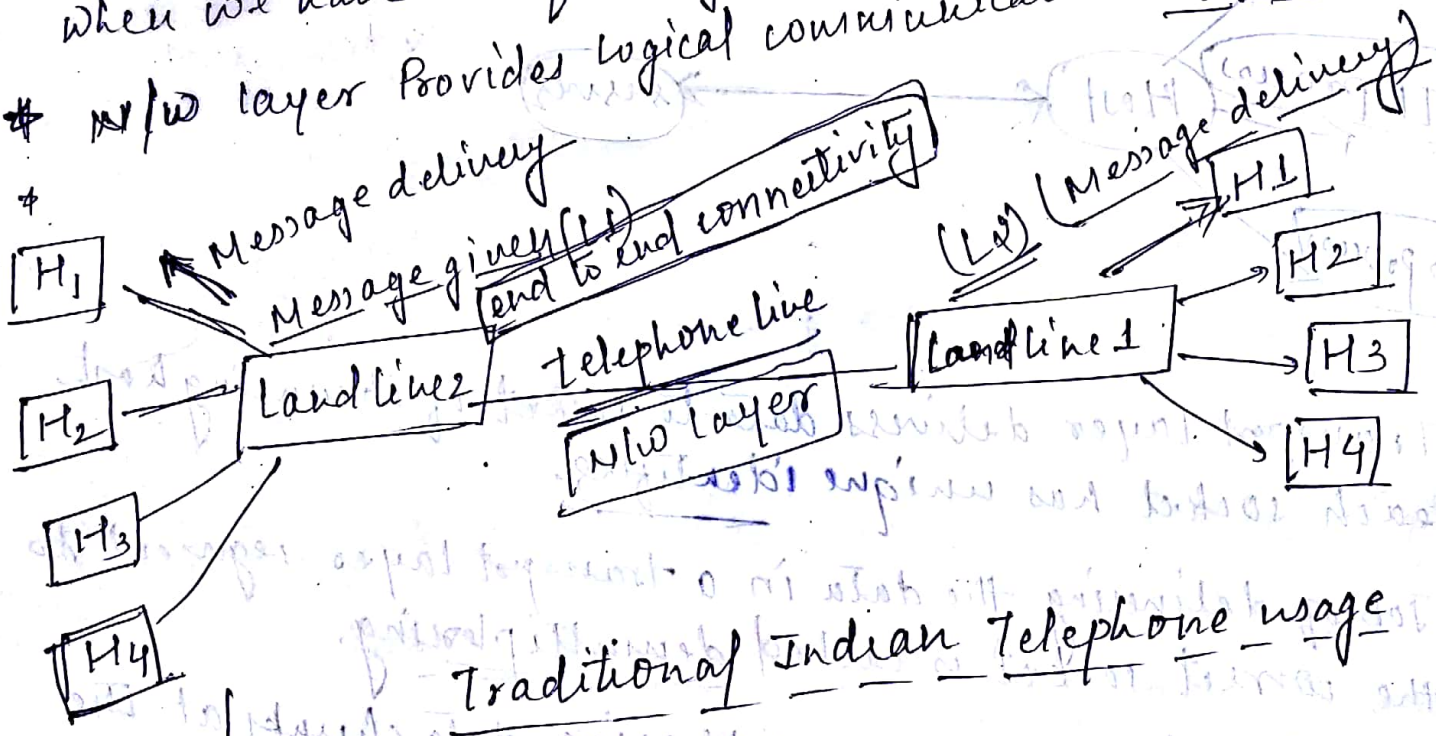
Transport layer

- Provides logical communication b/w application process running on diff host.
- from application Perspective, Not directly connected host
- Transport layer protocol implemented in end system not in routes.
- breaks into segments, add a transport layer headers to each chunk.
- Why do we need transport layer?
- Why do transport layer need error correction, error control when we have this facility in data link layer?
- N/w layer Provides logical communication b/w host.



Traditional Indian Telephone usage

TCP

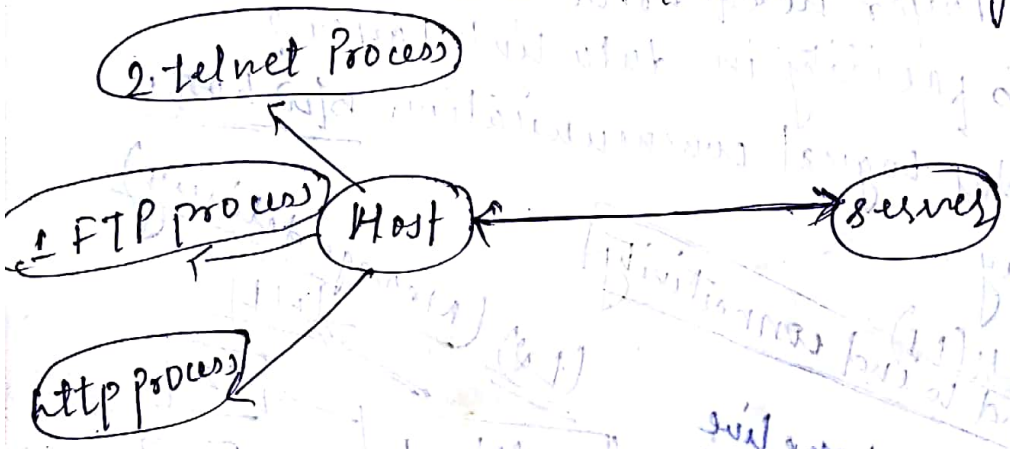
→ Now after sometime you are frustrated occansionally when phone ring you say, person is not there at their house, means dropping message occasionally

* IP protocol is based on "best effort delivery service" guarantee of segment delivery, no orderly delivery of segments, NO guarentee of integrity of data, IP unreliable service.

* integrity checking by error detection field.

TCP

* reliable data transfer → flow control, sequence numbers, acknowledgements, Timers.
* congestion control
* byte stream oriented transport protocol.
* TCP uses sliding window protocol.



* Transport layer delivers data to socket of incoming host, each socket has unique identifier.
* Job of delivering the data in a transport layer segment to the correct socket is called demultiplexing.
* Multiplexing - Job of gathering data chunks at the source host from diff sockets, encapsulating each data chunk at the source host from diff sockets encapsulating each data chunk with header info. to create segments and passing the segment to the NW layer is called multiplexing.

Numbers – 16 bit, 0 to 65535
0 to 1023 → well known port numbers.

source port numbers and destination port numbers in header.

* UDP as transport protocol does multiplexing/demultiplexing function and some light error checking.

* UDP – connectionless, No handshaking, process to process delivery and error checking

* Why UDP is needed when TCP is already there?

→ Fines application level control over what data is send and when – UDP takes data and send, TCP uses congestion control mechanism that throttles the transport layer & TCP uses resend mechanism regardless of how long it takes. Not suited for Real time application.

→ No connection establishment – No delay for connection establishment.

→ No connection state – buffers, congestion control para, seq no, ack no.

→ small packet header overhead –

* ~~Pipelined Reliable data transfer Protocol~~

TCP connection

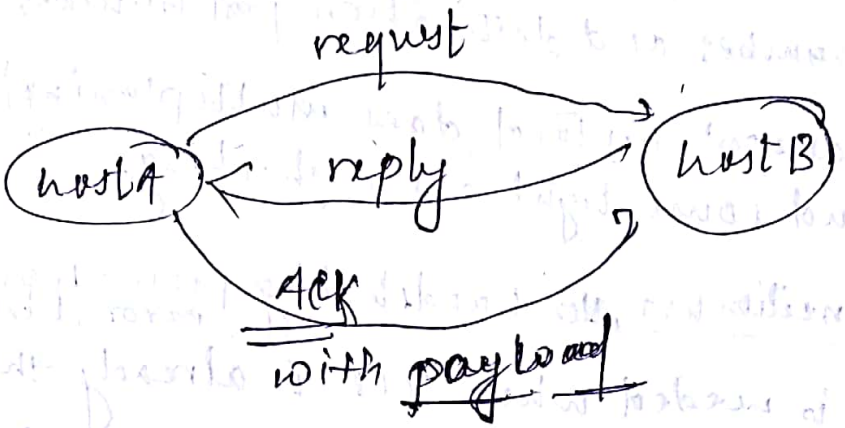
* for connection establishment two process must handshake

* TCP protocol runs on end systems, the intermediate ~~state~~ network do not maintain TCP connection state.

* full duplex service

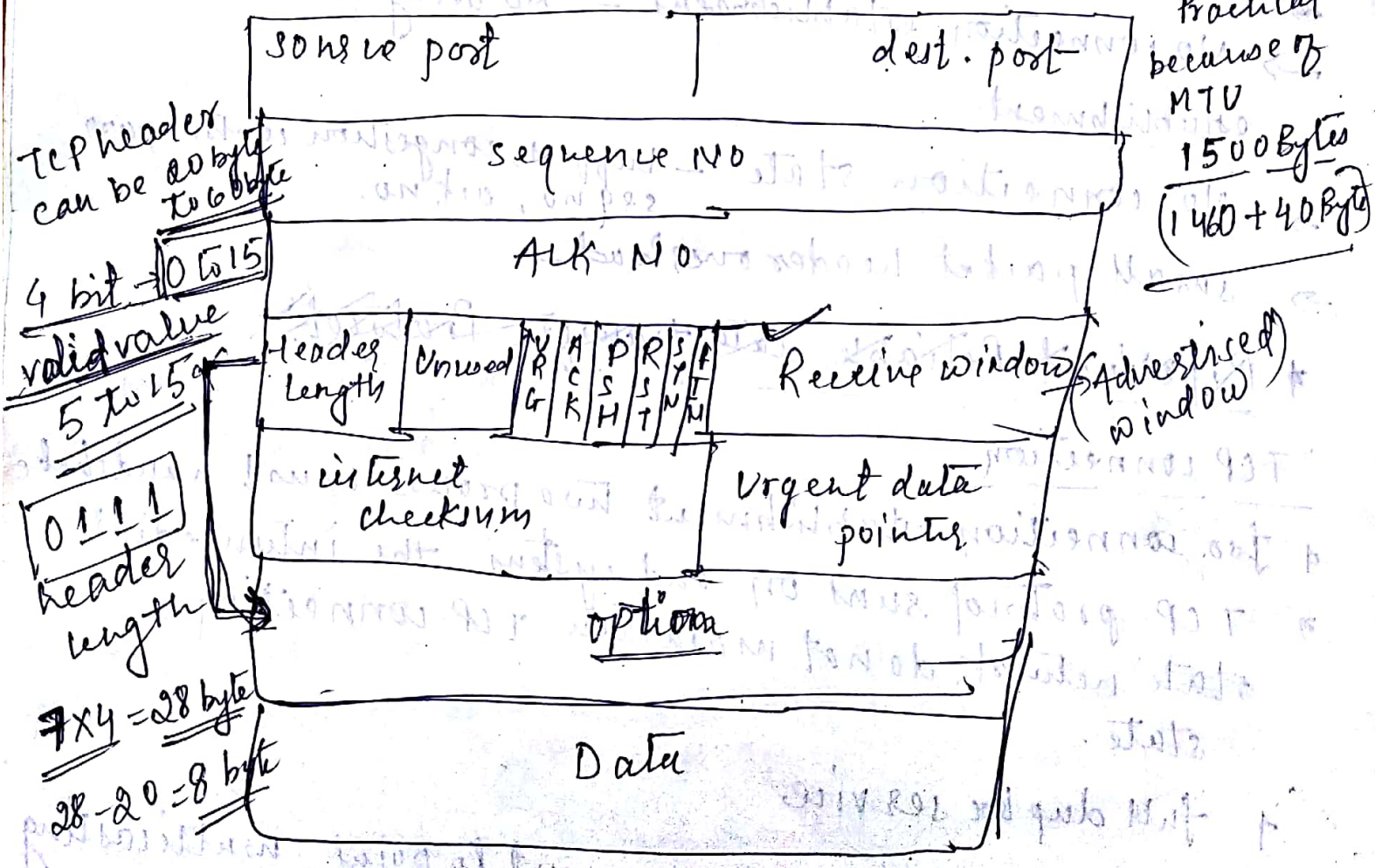
* TCP connection are always point to point, multicasting is not possible.

4 three way handshake



* MSS - is the Maximum amount of application layer data in the segment, not the maximum size of TCP segment including headers.

TCP segment structure (Max packet size of TCP connection 64kB), but not Practical because of MTU 1500 Bytes (1460 + 40 Byte)



sequence no & Ack no → to implement a reliable data transfer service and ordered delivery.

* Receive window (16 bit) – used for flow control.
* header length field (4 bit) – specifies the length of TCP header in 32 bit words. The TCP header can be variable length due to TCP option field.
* The option field is used when a sender and receiver negotiate the Maximum segment size or as a window scaling factor for use in high speed network.
* Flag field – 6 bits –

ACK=1 data+ACK
ACK=0 invalid

ACK – the segment contain Ack for a segment that has been successfully received.

RST – connection setup & teardown.

PSH – receiver should pass the data to upper layer immediately.

URG – there is a data in segment that the sending side upper layer entity marked as "urgent"

* 16 bit urgent Pointer – The location of the last byte of urgent data is indicated by 16 bit urgent data pointer field. TCP must inform the receiving side upper layer entity when urgent data exist and pass it a pointer to the end of urgent data.

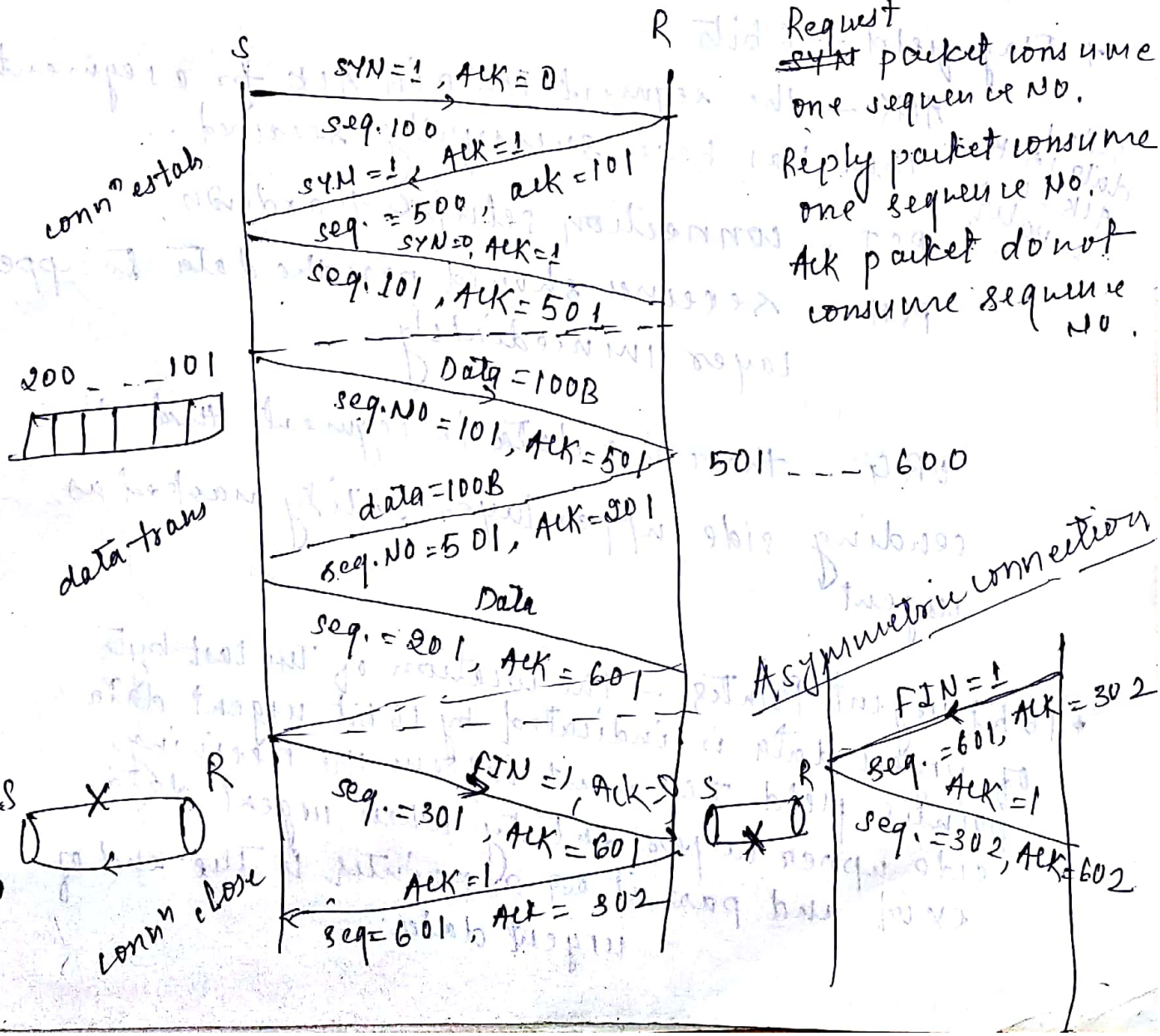
Packet recognition :-

	SYN	ACK
request packet	1	0
reply	1	1
data/~~data~~ ACK	0	1
purely data packet	0	0

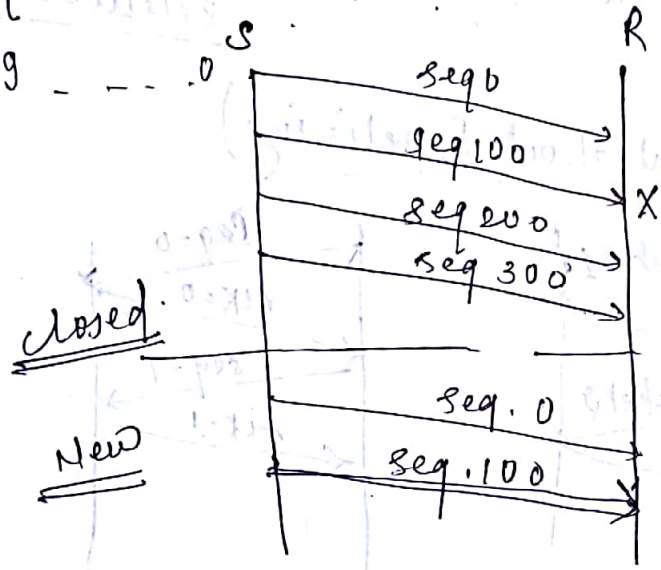
FIN - used to terminate the connection.

RST Flag - it is used to refresh the connection.

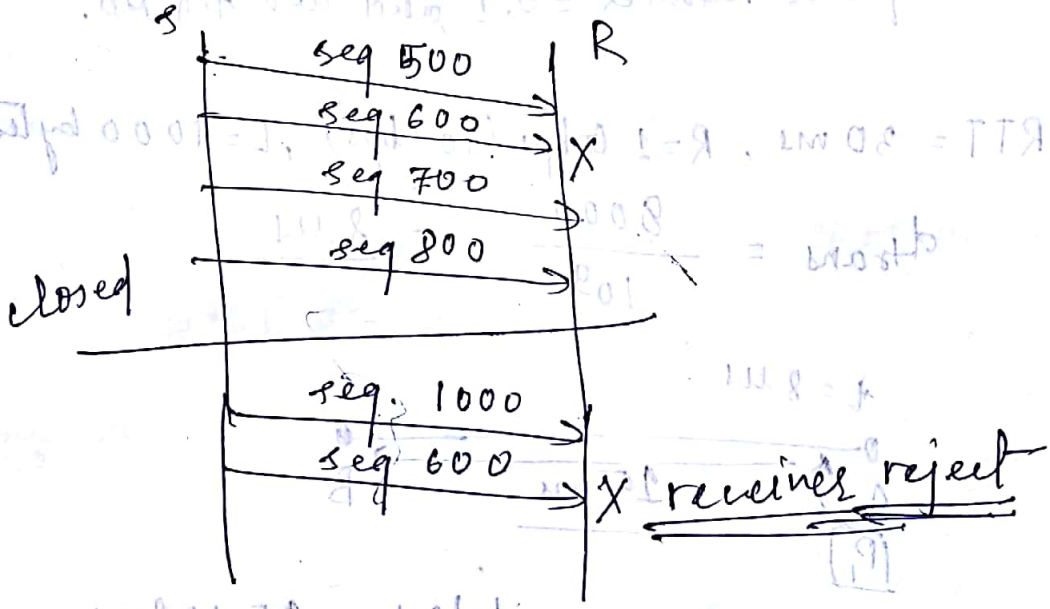
~~sequence~~ sequence Number and Acknowledgement NO.



Why each connection starts with random initial sequence number.



जाना था जापान पहुँच गये
नींद वाली condition
receives need to reject
but did not



~~Security issue~~

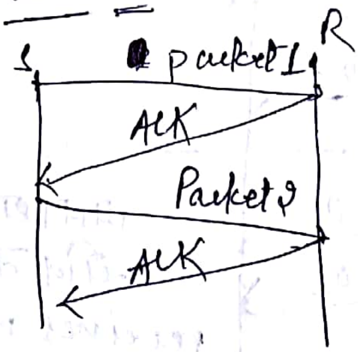
- TCP views data as ordered stream of bytes. Sequence No reflects this view as the stream of transmitted bytes and not ones the series of transmitted segment.
- The acknowledgement number that receiver puts in the segment is the sequence number of the next byte receiver is expecting from host A.

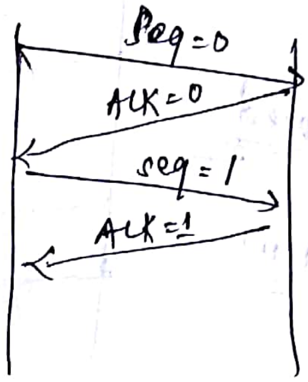
Pipelined reliable data transfer protocol :-

→ Also called sliding window protocol → Go Back N & Selective Repeat

Non pipelined protocol

① STOP and wait (Without Pipelining)

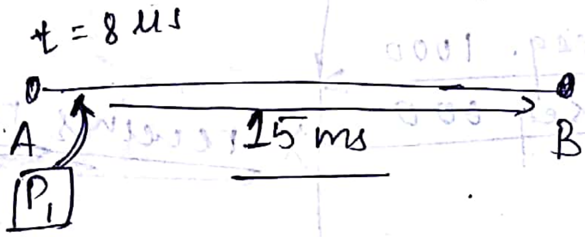




* 2 sequence number = 0, 1 and two ACK NO.

* RTT = 30 ms, R = 1 Gbps (10^9 bps), L = 1000 bytes

$$d_{trans} = \frac{8000}{10^9} = 8\ \mu s$$



One way delay = 15.008 ms

NO ACK is very small, ACK will take 15 ms to reach to A.

$$t = RTT + L/R = 30.008\ ms$$

in 30.008 ms, host is sending for only = 0.008 ms

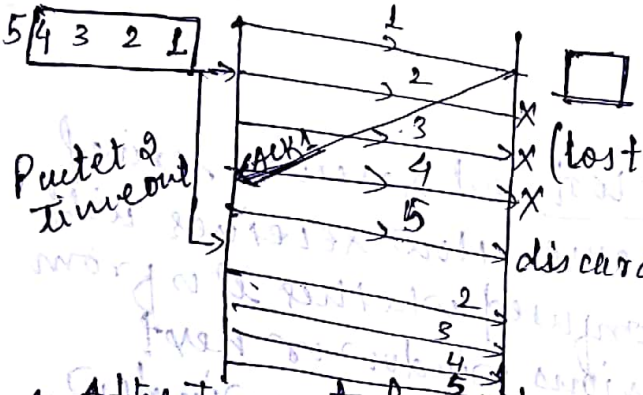
$$\text{utilization} = \frac{L/R}{RTT + L/R} = \frac{0.008}{30.008} = 0.00027$$

$$\text{throughput} = \frac{1000}{30.008} = 267\ \text{Kbps} = \text{[illegible]}$$

Go Back N :-

- sender is allowed to transmit multiple packet without waiting for an acknowledgement.
- Not more than N ~~packet~~ unacknowledged packet is allowed. (6 outstanding packets)
- It requires large buffer size.
- More sequence numbers are needed.

window size = 4



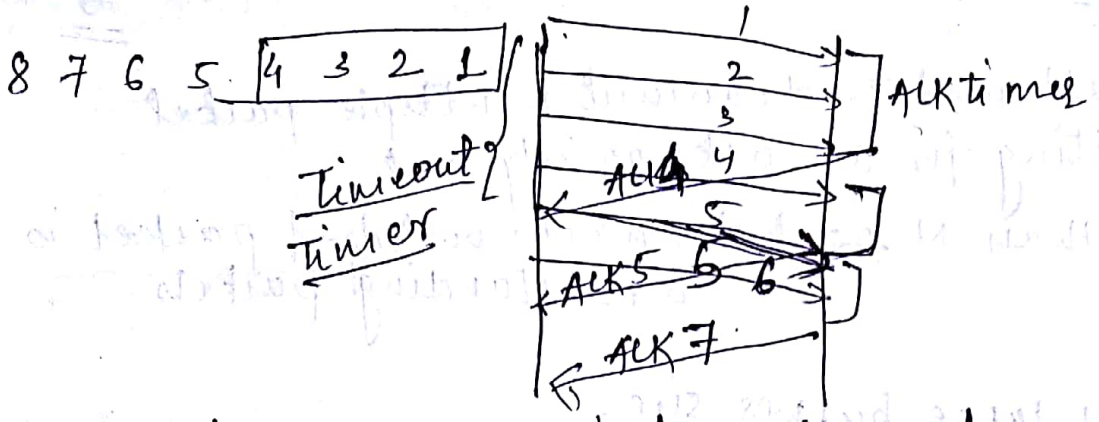
window size = 1

window size = min(receiver window, congestion window)

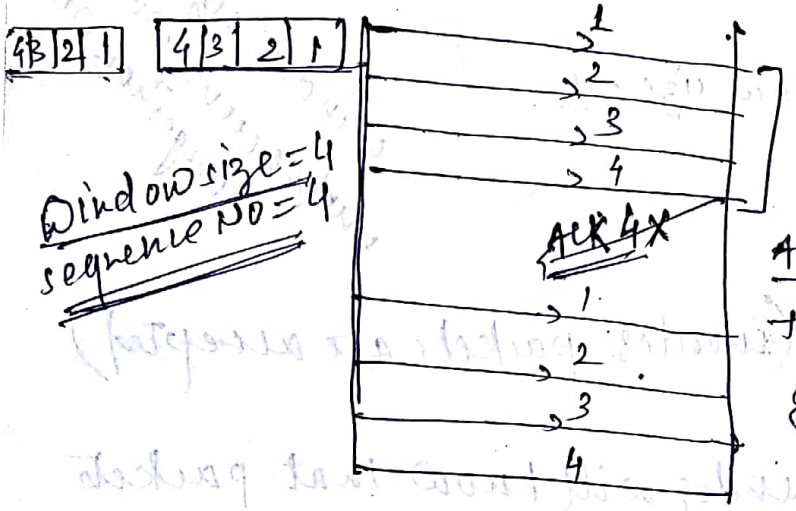
- After timeout for 2nd packet sender will know that packets in windows 2, 3, 4, 5 are lost so resend.
- ~~Here~~ acknowledgement are two types :-
  1. independent
  2. cumulative
- For GBN cumulative ACK is used :-

GBN maintains a timer called acknowledgement timer which starts when any packet is received. when ACK timer expires a single acknowledgement will be sent for all the packets within this time.

If ack times is too big, it leads to timeout & retransmission if Ack timer is too small, it becomes independent Ack & more traffic.



Relation between window size and sequence numbers :-



ACK is lost but receiver received the frame, Now receiver will get confused whether it is from previous window or next window.

- so if window size is N, then min no of seq. no required = N+1.

- If k is the max no of bits available in sequence number of field, then what is max window size

$= 2^k - 1$

$W_s + W_R \leq$ Available sequence NO

Suppose you are designing a sliding window protocol for a 1 Mbps point to point link to stationary satellite revolving around the earth at 3×10^4 km altitude. Assuming that each frame carries 1 KB of data, what is the minimum number of bits that you need for the sequence number in the following case? Assuming speed of light 3×10^8 m/s

(a) RWS = 1

(b) RWS = SWS

$S_W + W_R \leq$ Available sequence no

Soln → One way latency = $\frac{3 \times 10^4 \times 10^3}{3 \times 10^8}$

= 100 ms

RTT = 200 ms = 0.2 s

data send in one RTT = BW × RTT

= 1 Mb × 0.2

= 0.2 Mb

No of packets simultaneously send

= $\frac{0.2 \times 10^6}{8 \times 10^3}$ = 25 per frame

(a) RWS = 1, necessary sequence number = 25 + 1 = 26

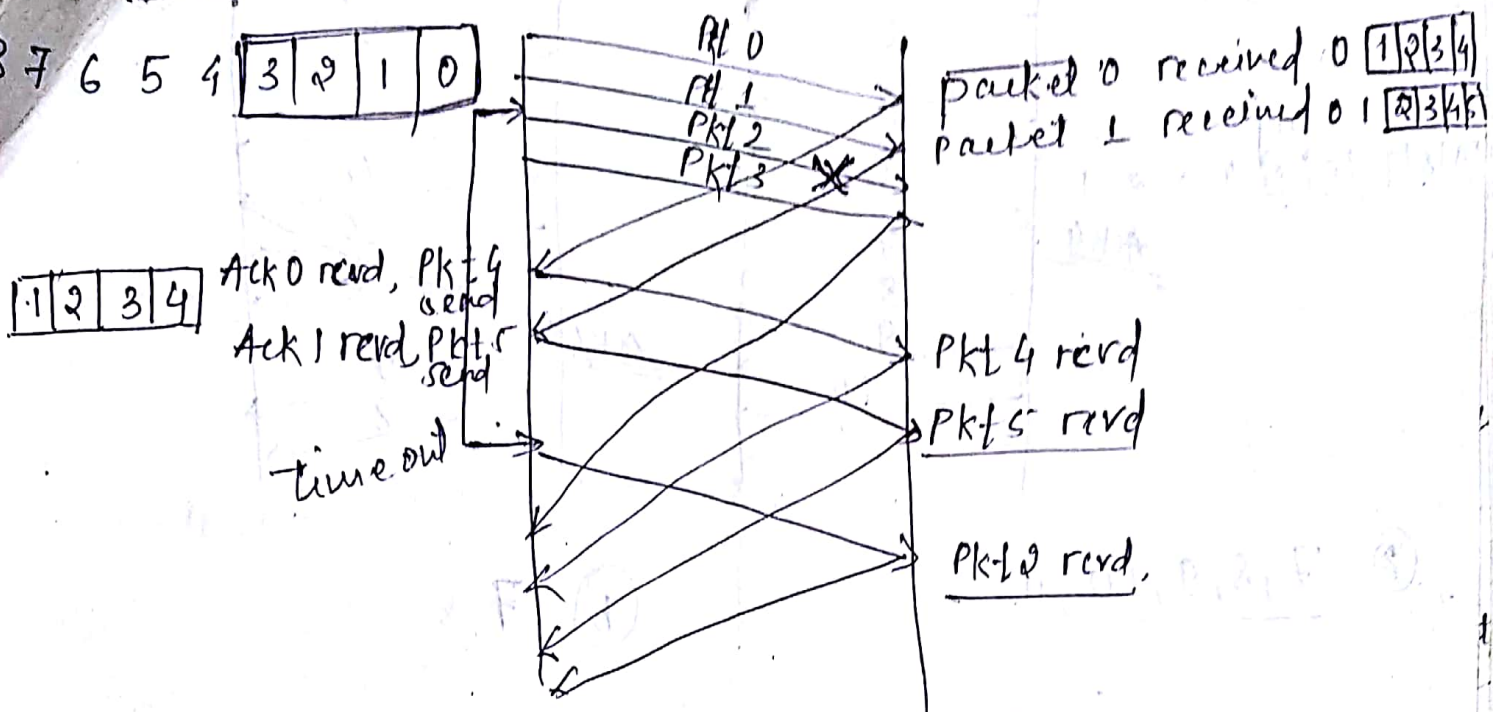
No of bits = 5

(b) RWS = SWS, necessary seq. no required

= 25 + 25 = 50, No of bits = 6

Selective Repeat operation

$w = 4$



- Receiver can accept out of order packets.
- Sender will send only lost packet.
- Here sender window = Receiver window
- If k is no of bits in sequence numbers then window size:-

$$w_s = \frac{2^k}{2} = 2^{k-1}, \quad w_R = \frac{2^k}{2} = 2^{k-1}$$

$2^m \geq S + r$
$\geq 2x$
$2^{m-1} \geq x$

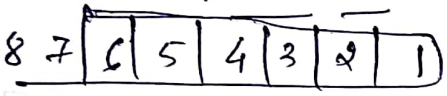
Sum 1

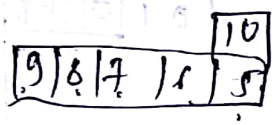
① Go back N protocol, window size = 7, Frames sent = 1, 2, 3, 4, 5, 6, sender received an Ack for frame 4. Frames 7, 8, 9, 10, 11, 12 waiting to be sent. Draw Timeline.

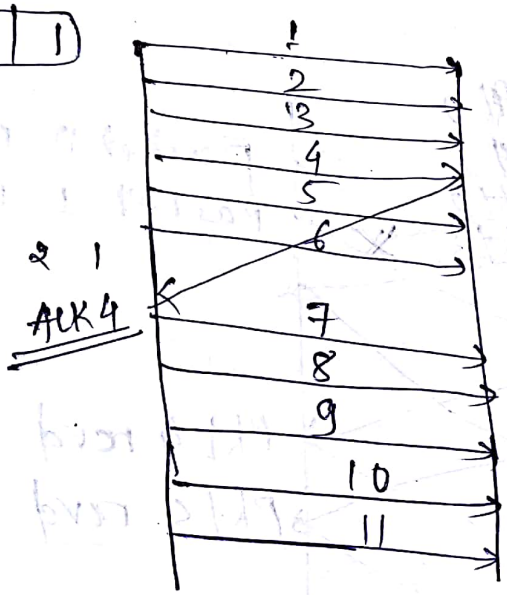
② Which frames can sender send before it must wait for the next Ack from the receiver?

③ Repeat for selective Repeat.

Go back N

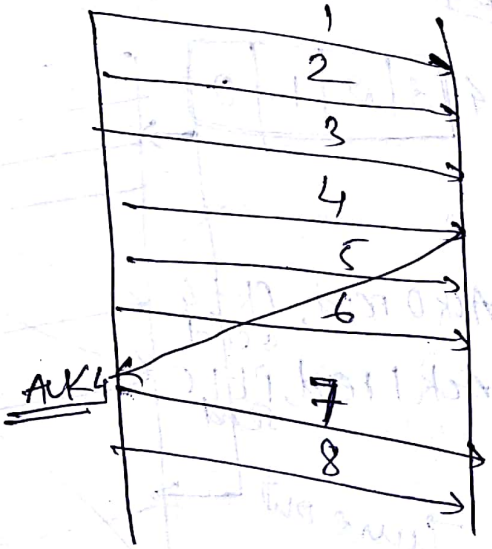






(a) 7, 8, 9, 10, 11

Selective Repeat



(b) 7, 8