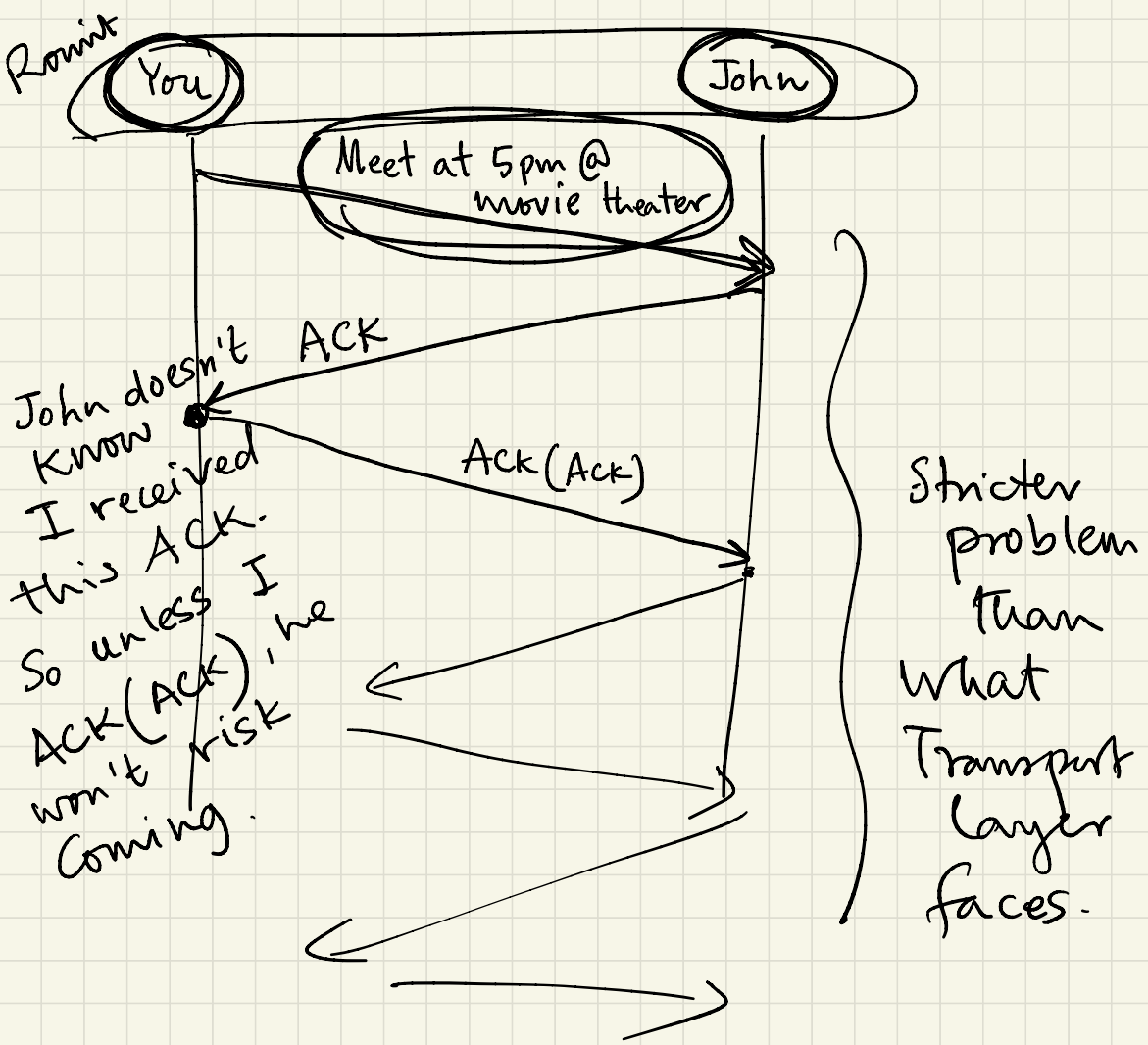
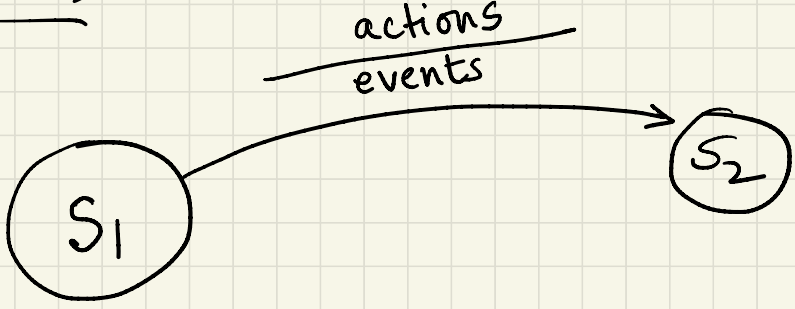
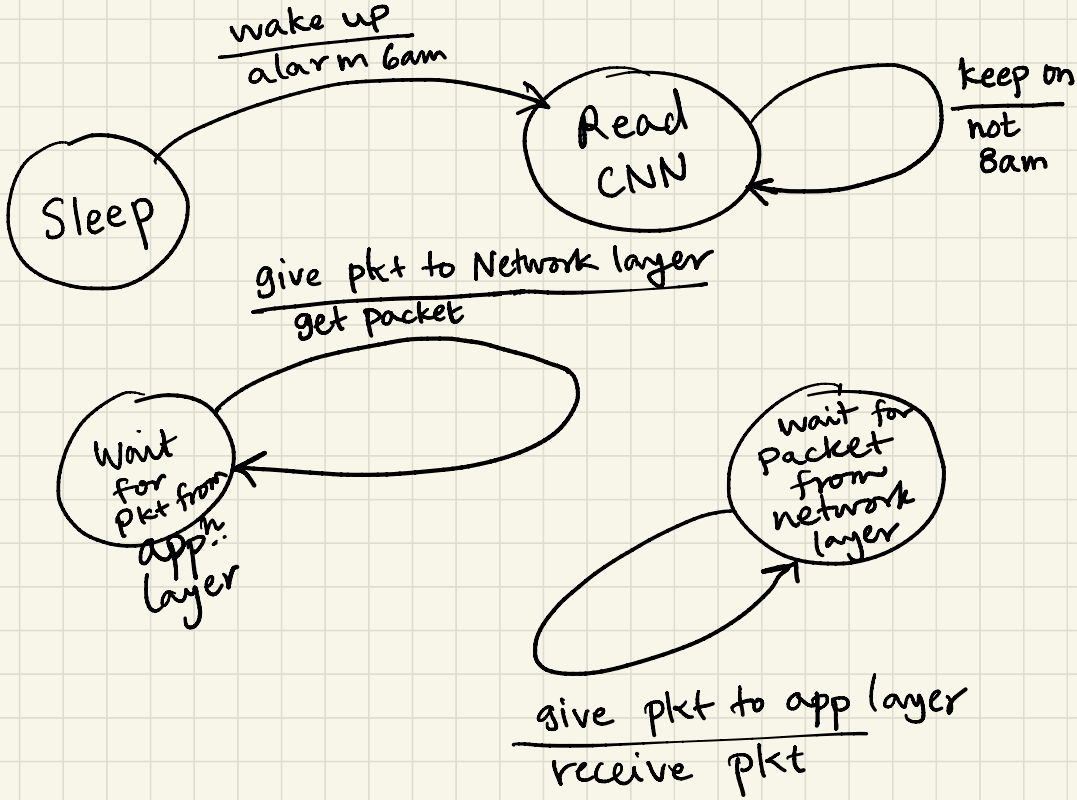


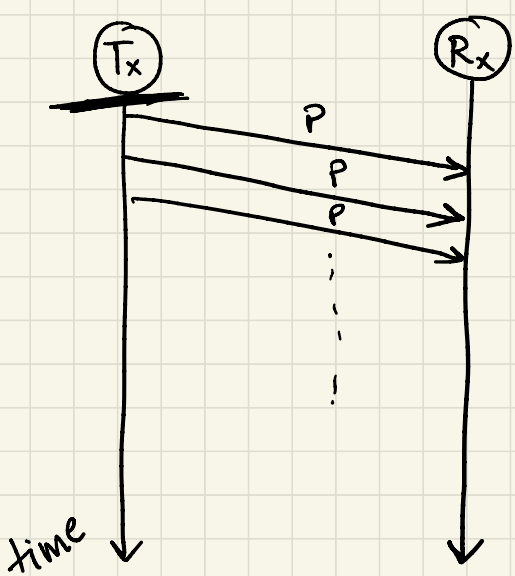
State Machines

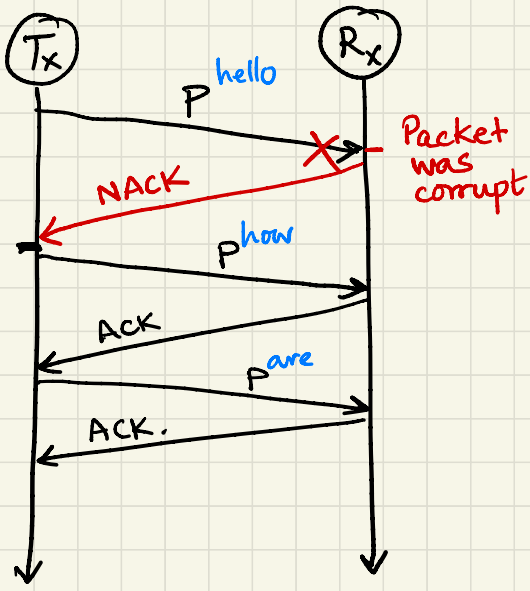


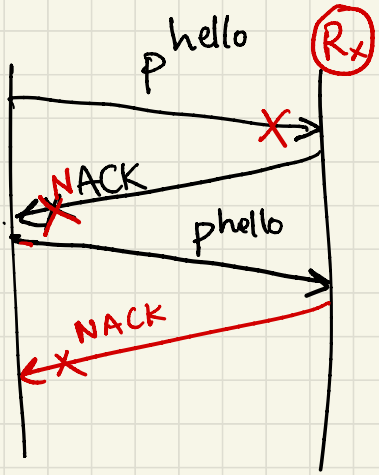


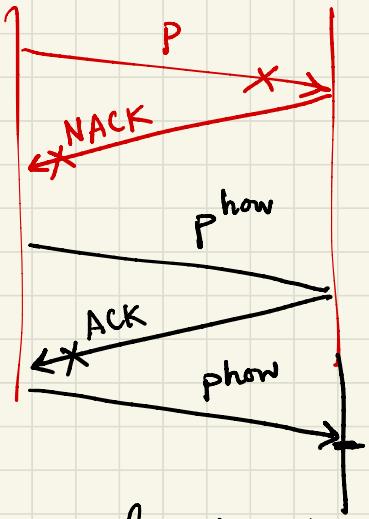
Reliable Pkt transmission.

1011 0 00 1 0100 0 1101





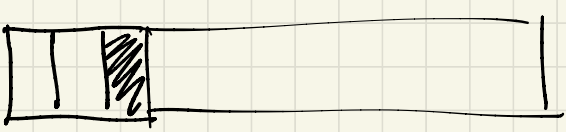


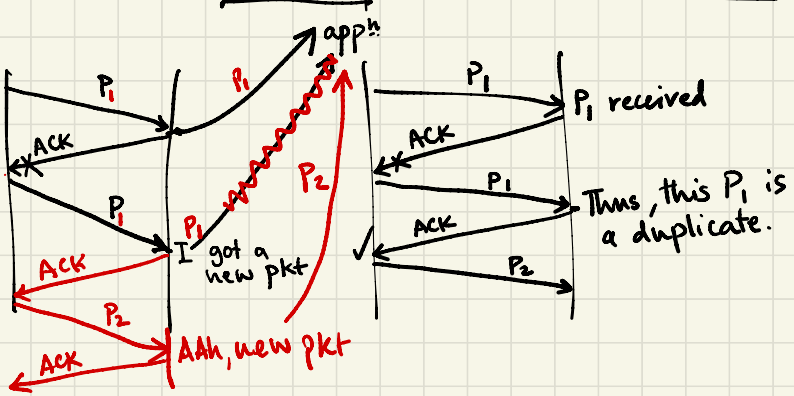


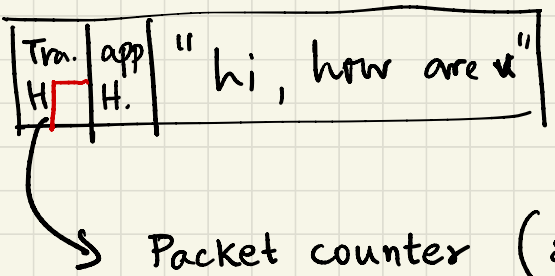
- ACK corruption causes duplication at R_x
- NACK corruption is fine.

⇓

Add seq. # to packets.

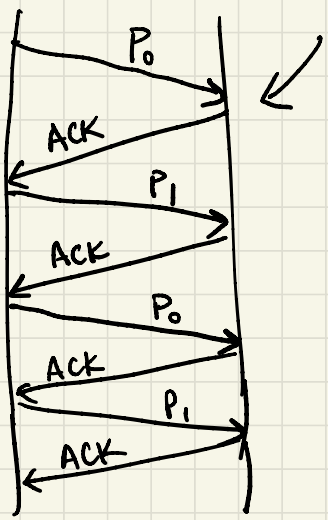




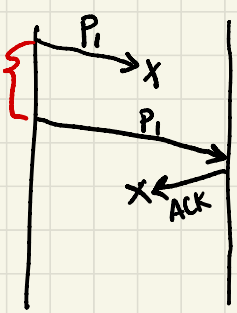


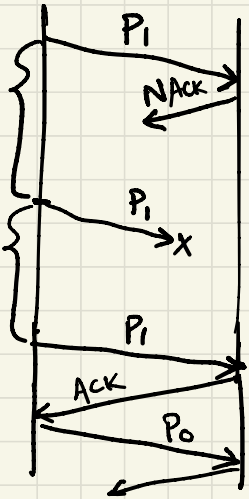
- Under bit errors (packet corruption)
  - ↳ 1 bit seq # is enough.

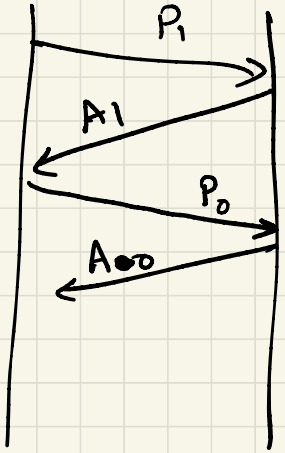
Error Model
↳ Pkts are either correctly delivered or are delivered with bit error.



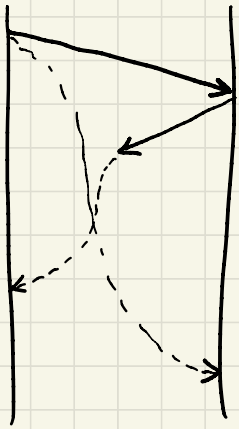
- Error Model : Bit error + pkt Loss
  - ↳ set timer / alarms
    - ↳ Ret$_x$ after some timeout

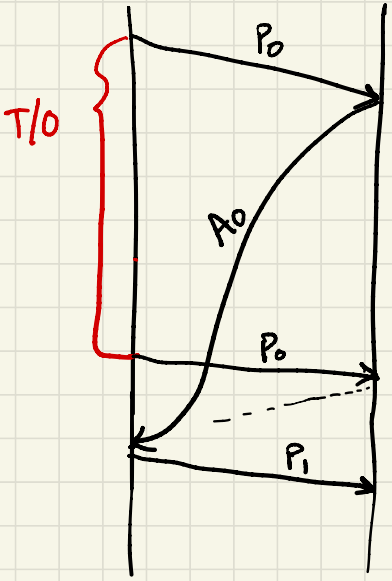


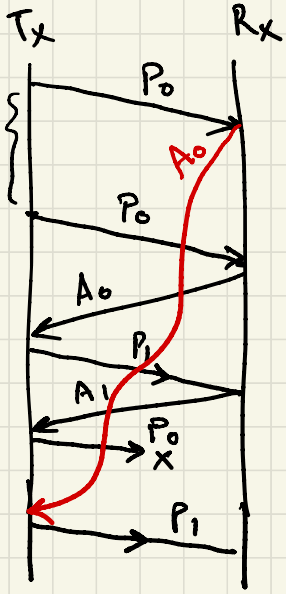


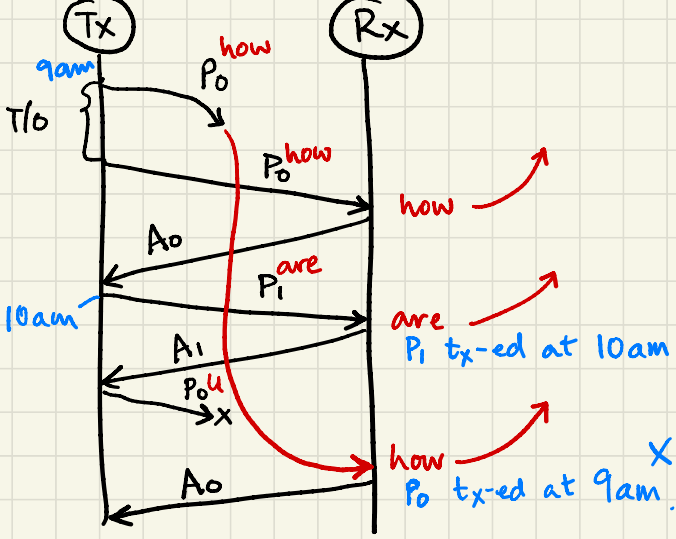


- Link Error : Bit Error + Loss + Delay



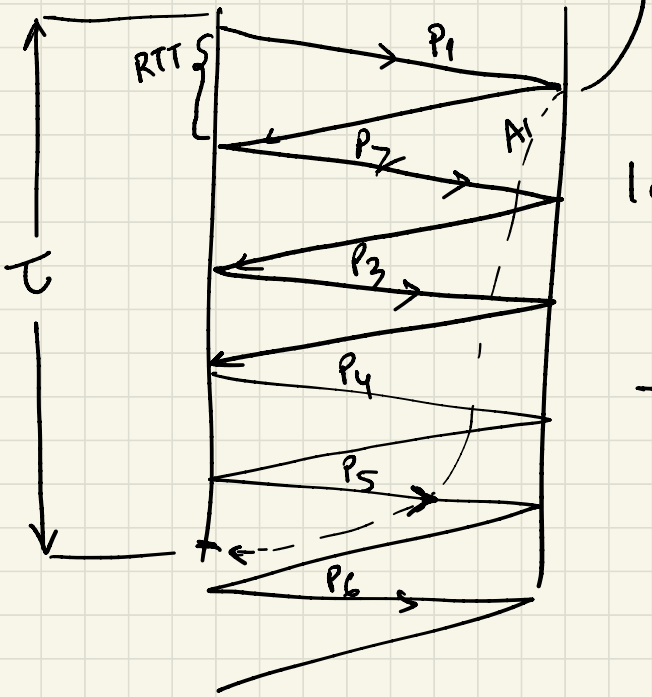






Pkt seq # needs to be increased.

Yes there is delay, but delay $\leq$ upper bound τ

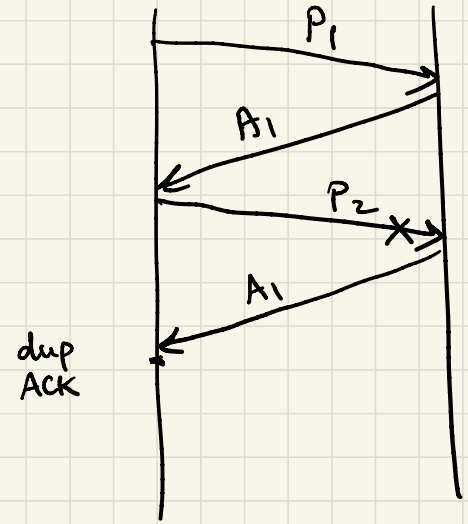


RTT = Round Trip time.

$$\log_2\left(\frac{\tau}{RTT}\right)$$

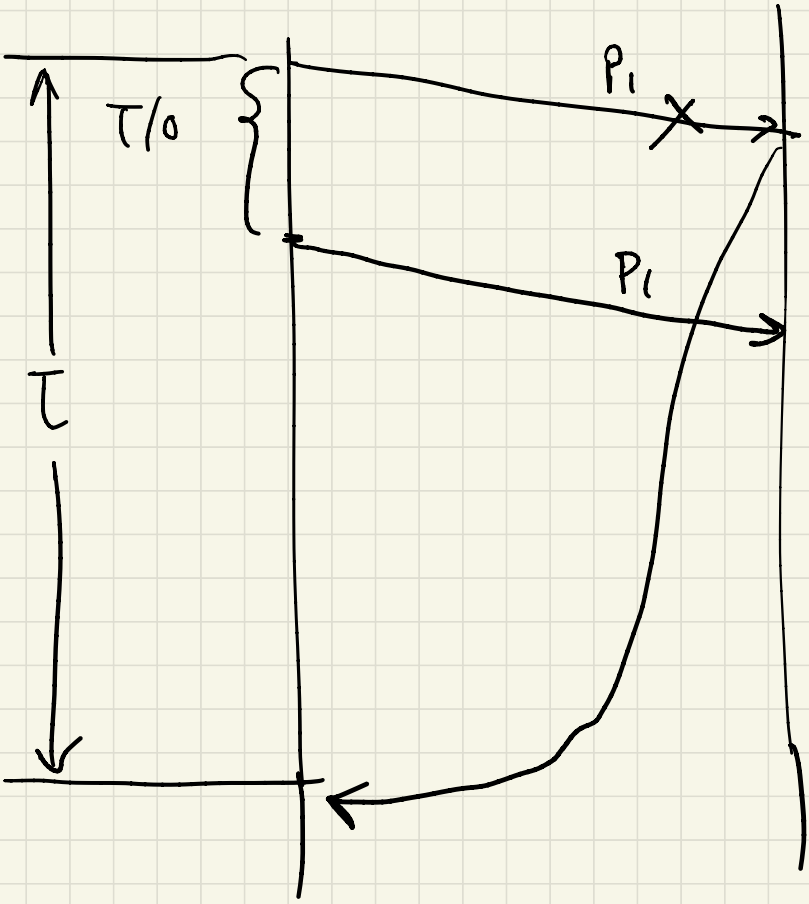
For simplicity

Replace NACKs with duplicate ACKs.

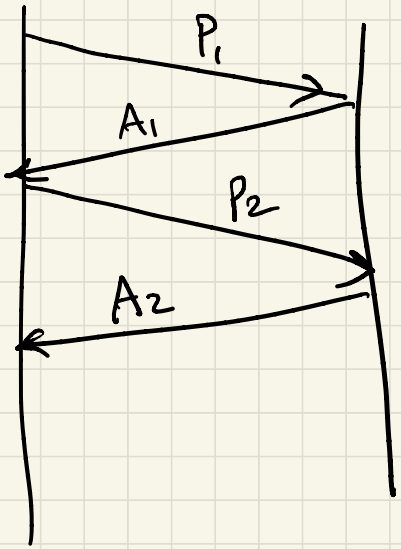


NACK - free protocol.

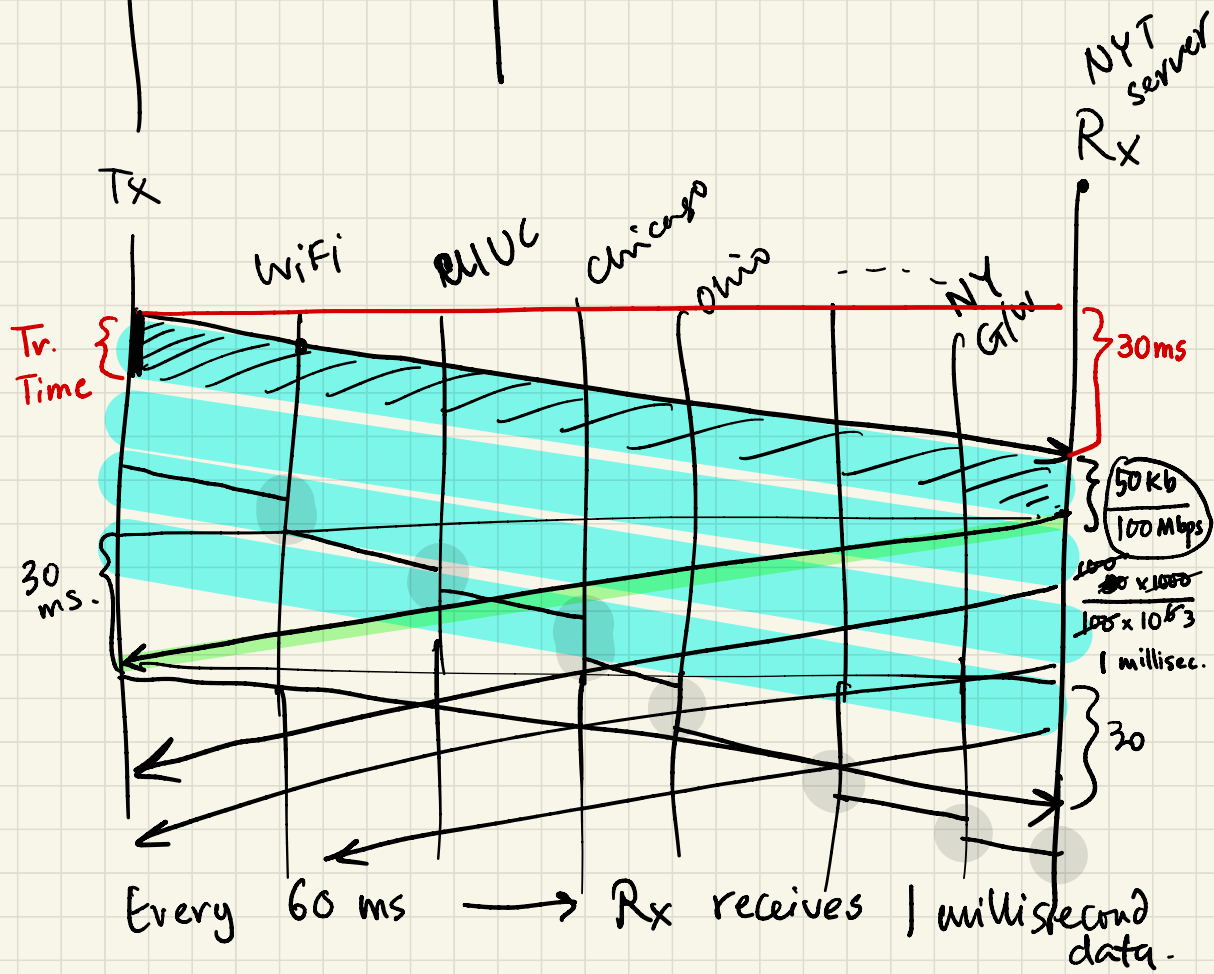
$$N > \log_2\left(\frac{T}{RTT}\right)$$



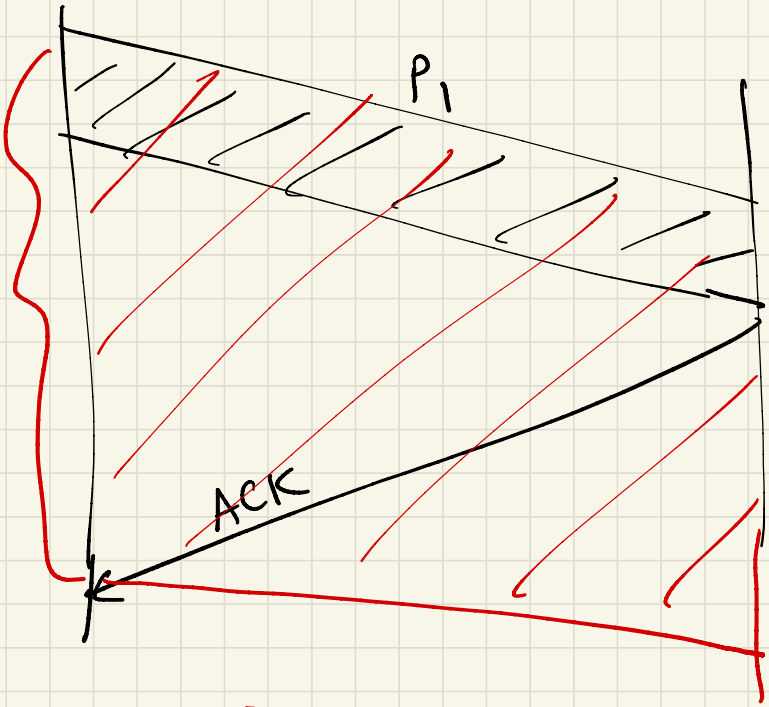
Stop and Go



→ How bad is this ?

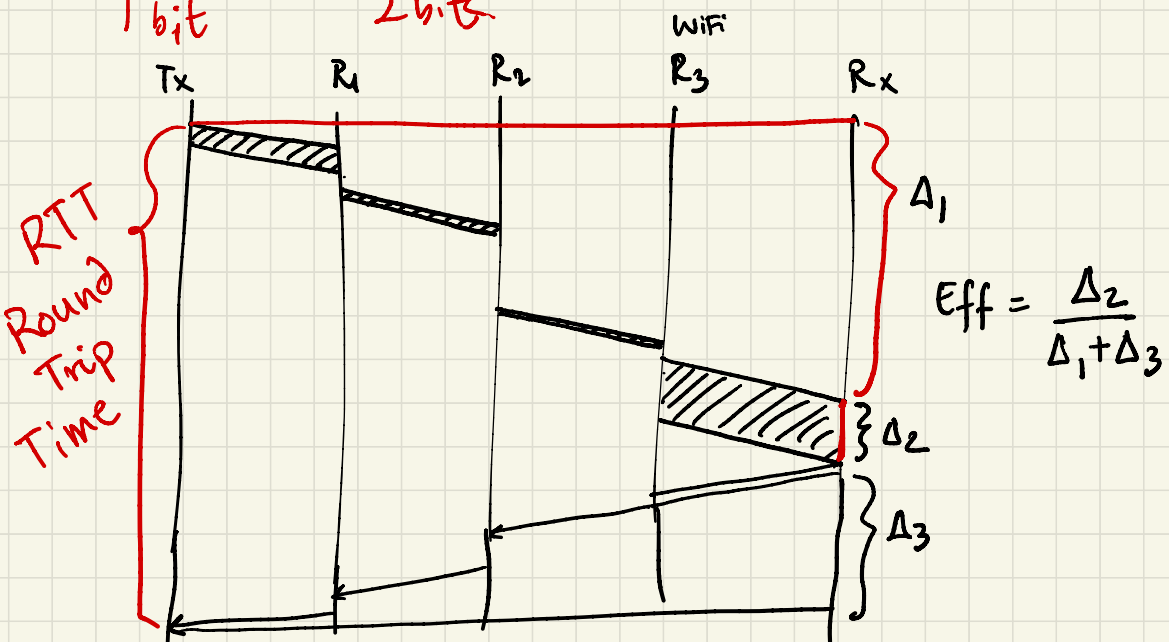


Every 60 ms → Rx receives 1 millisecond data.

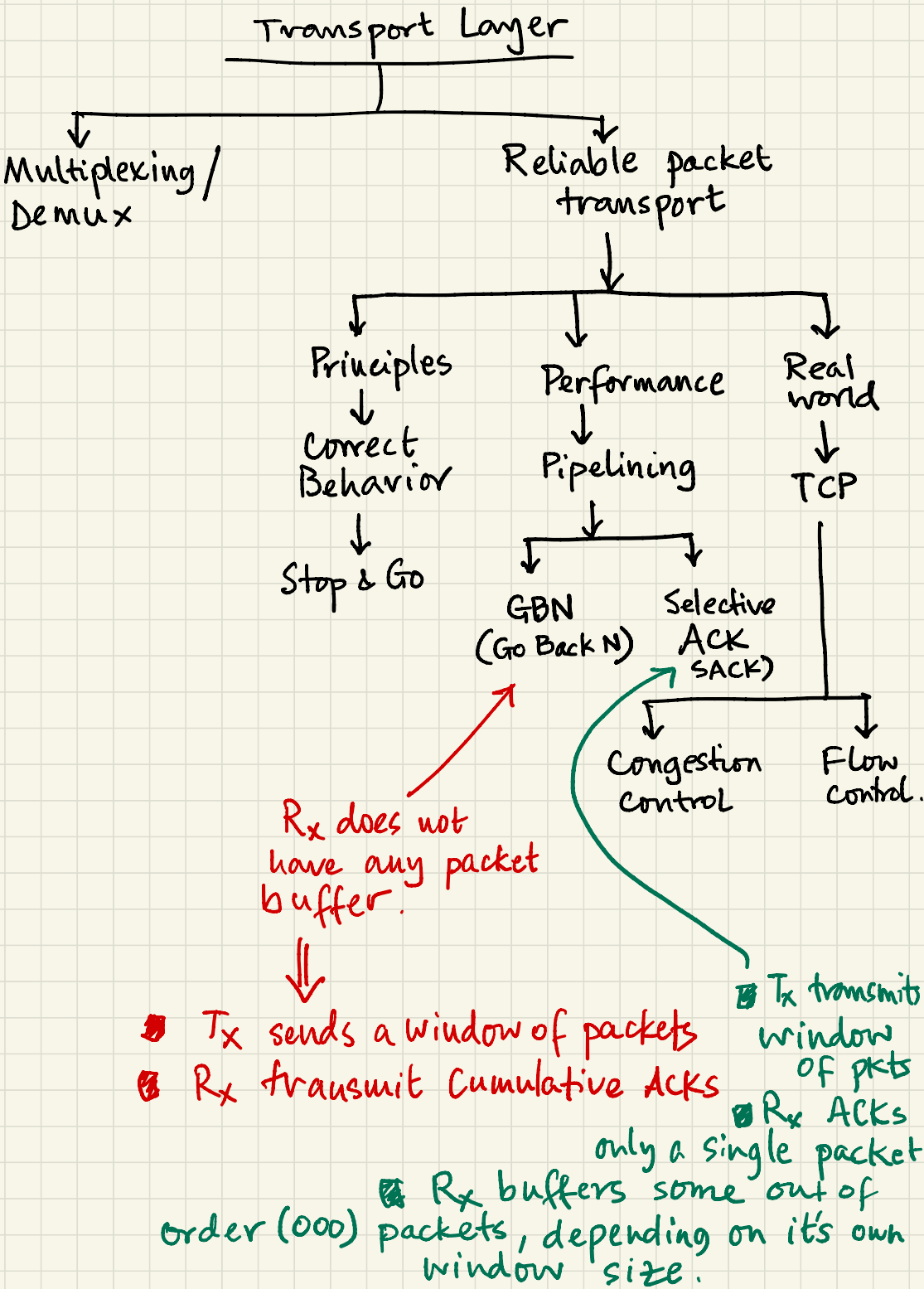


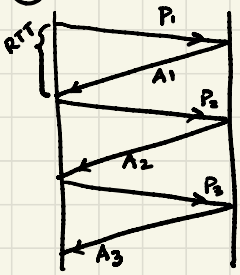
(2) (4) Stop & Go.

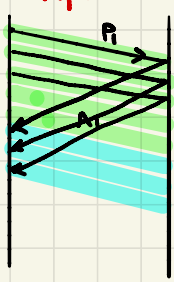
1 bit 2 bits

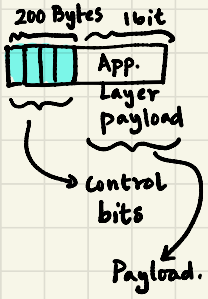


$$Eff = \frac{\Delta_2}{\Delta_1 + \Delta_3}$$


Transport Layer
Multiplexing / Demux
Reliable packet transport
Principles
Correct Behavior
Stop & Go
Performance
Pipelining
GBN (Go Back N)
Selective ACK (SACK)
Real world
TCP
Congestion control
Flow control
Rx does not have any packet buffer.
Tx sends a window of packets
Rx transmit Cumulative ACKs
Tx transmits window of pkts
Rx ACKs only a single packet
Rx buffers some out of order (OOO) packets, depending on it's own window size.

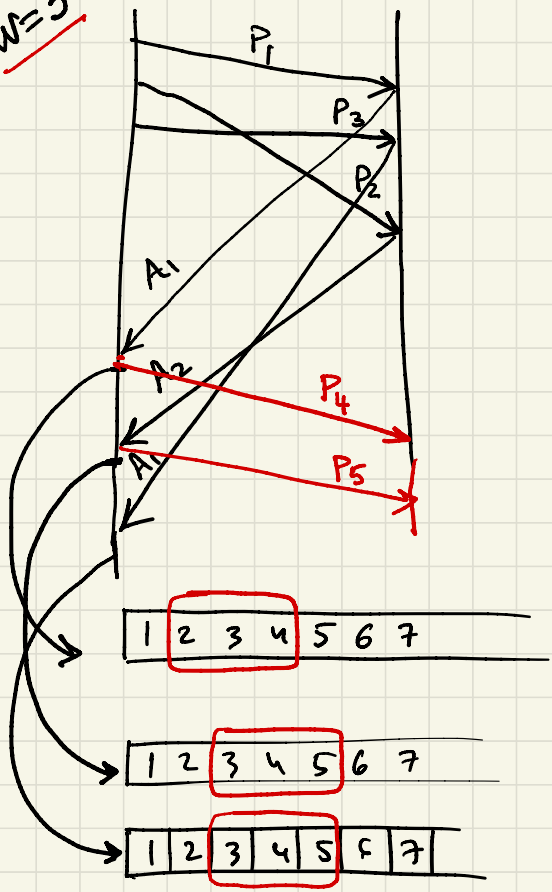
① 

Pipelined 



# of un-ACKed pkts T_x can send to not waste any time = $\frac{RTT}{T.T}$

W=3



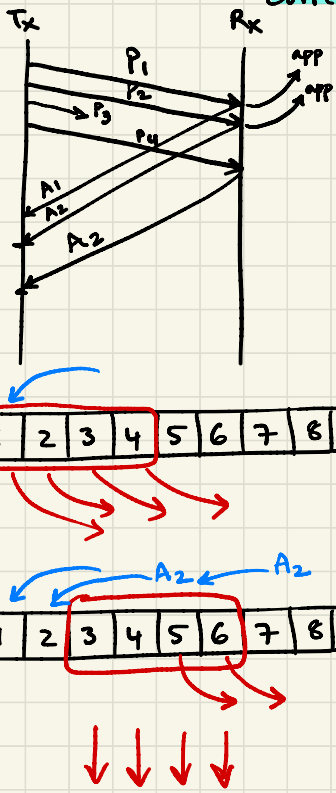
② Go Back N (GBN)

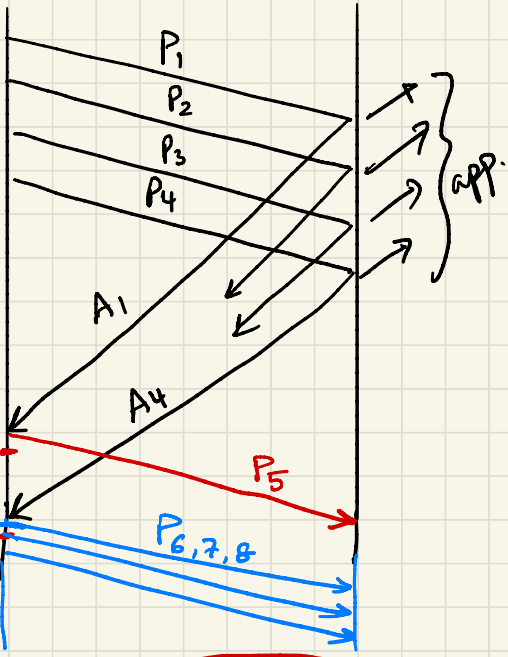
Tx

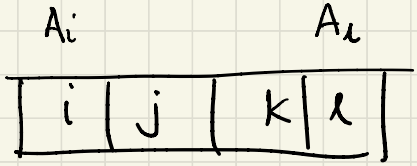
- transmits a window of packets (Window size = W = 4)

Rx

- Does not have a buffer
- ACKs the last in order packet received correctly.

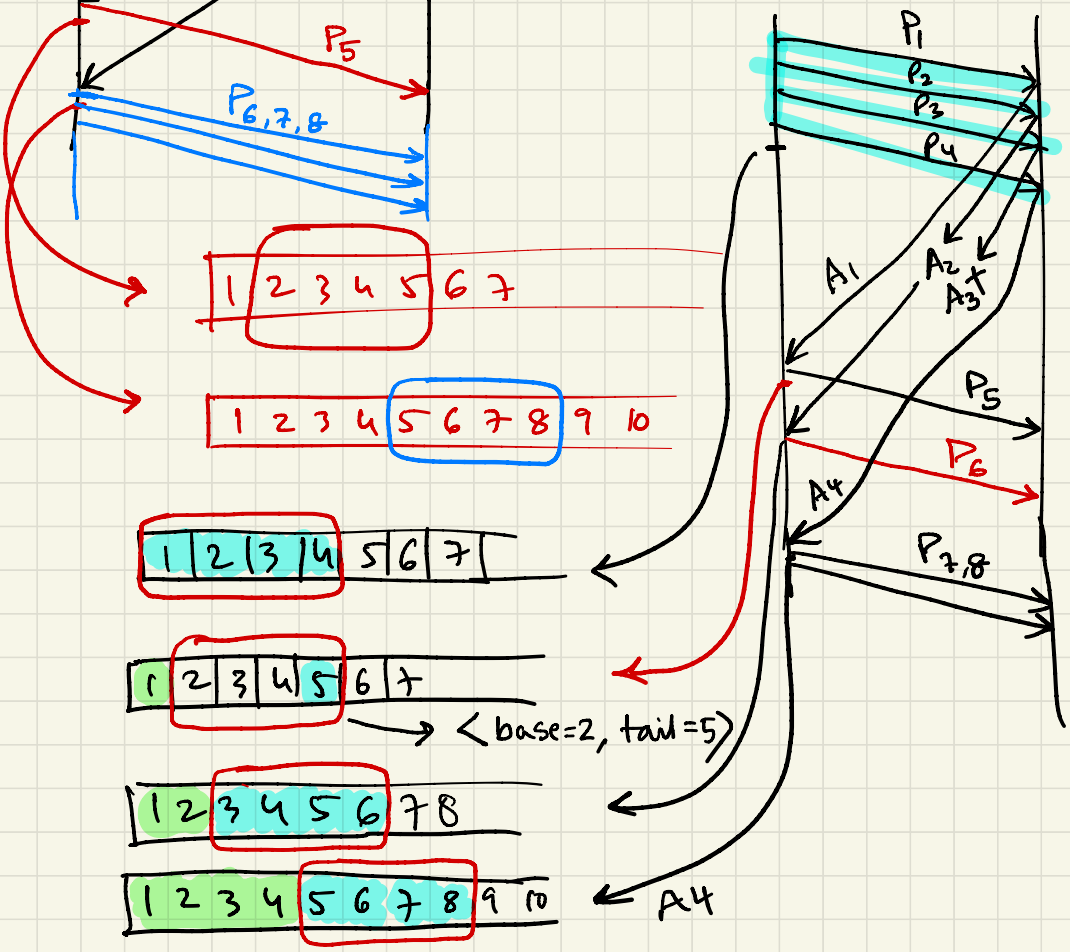




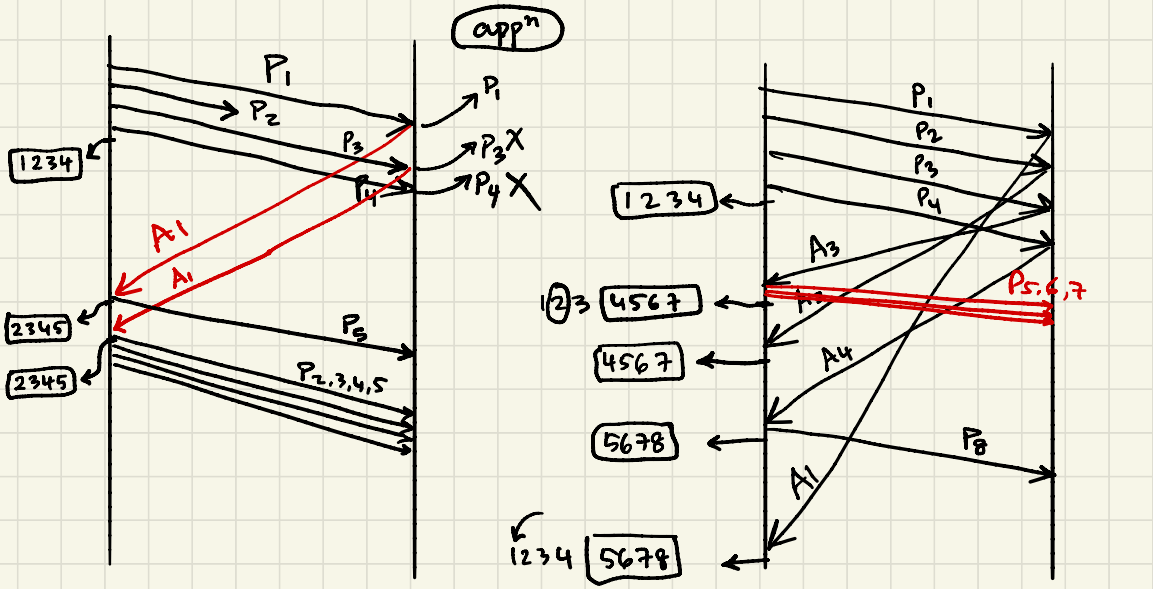


$A_p, \quad p > i$

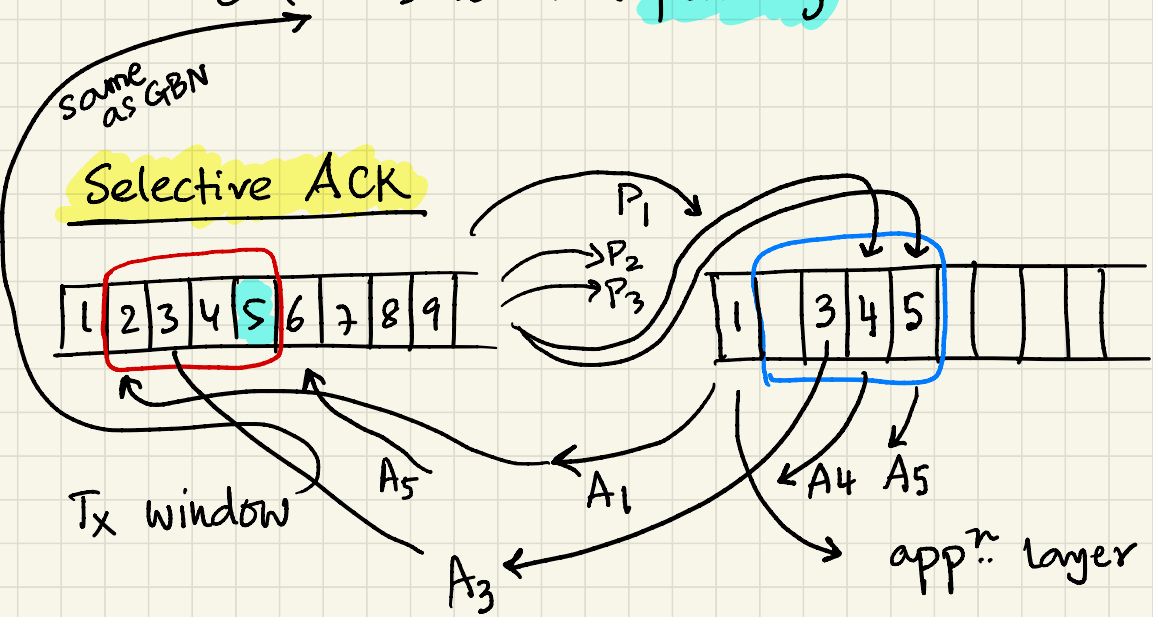
Yes, if ACKs get reordered.



Tx maintains a window = <base, tail>
5 8



GBN: Transmitter's window says which pkts can be sent with pending ACKs

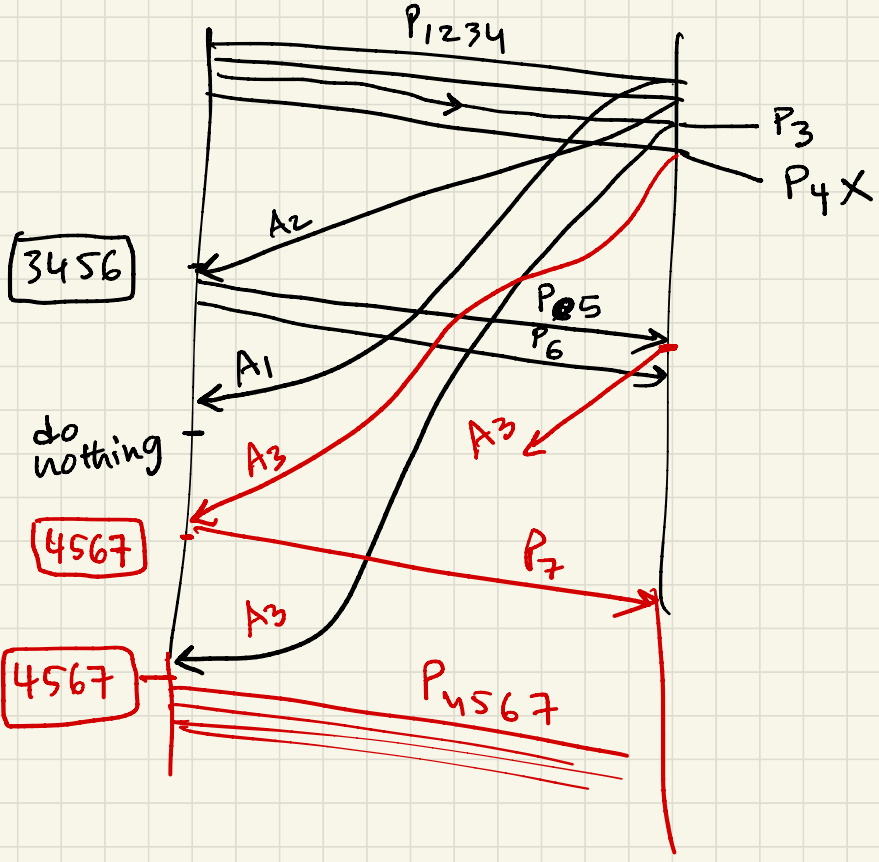


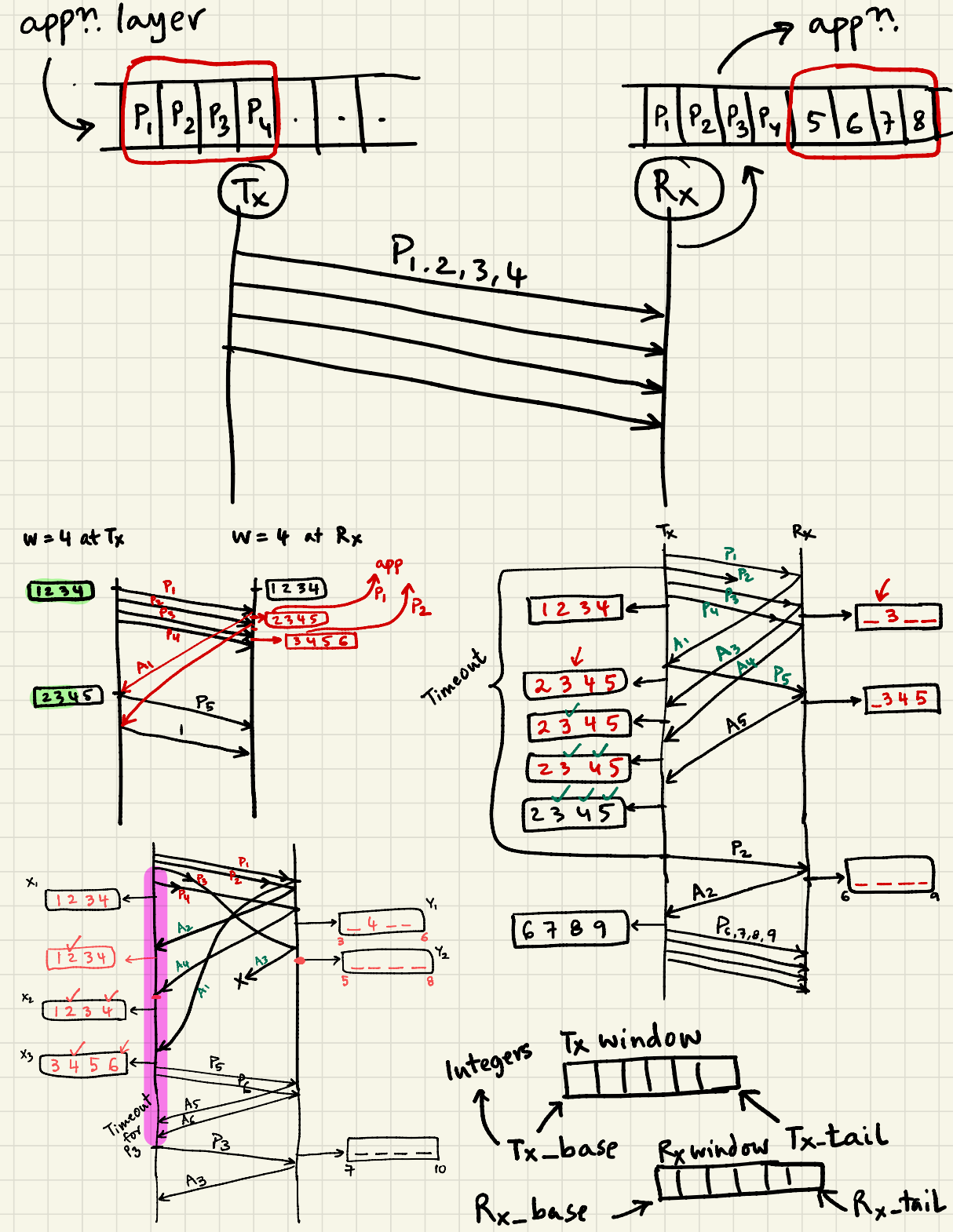
R_x window says how many out of order packet can I hold while waiting for ACKs.

In selective ACK (SACK), the ack A_i indicates that R_x has received packet P_i.

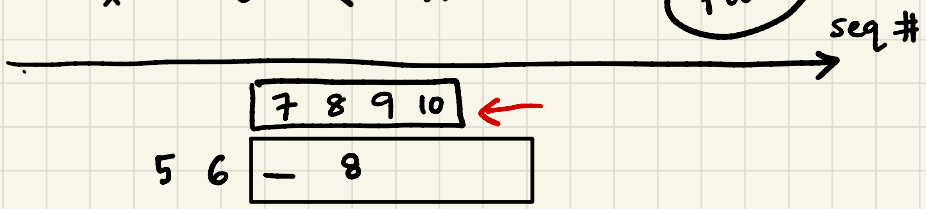
(In GBN, A_i indicates R_x has received every in-order pkt till and including P_i)

Example (after class)

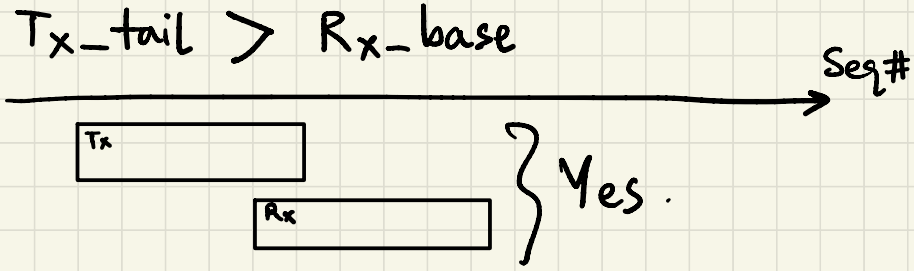



appn. layer
P1 P2 P3 P4
Tx
appn.
P1 P2 P3 P4 5 6 7 8
Rx
P1,2,3,4
w = 4 at Tx
w = 4 at Rx
1234
2345
3456
app
A1
P5
Timeout
6789
P6,7,8,9
X1
X2
X3
Y1
Y2
Timeout for P3
Integers
Tx window
Tx_base
Tx_tail
Rx window
Rx_base
Rx_tail

T/F : $R_x\_base < T_x\_base$ (False)

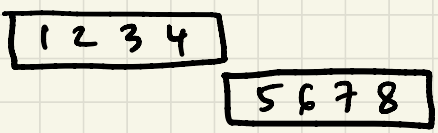


T/F : $T_x\_tail > R_x\_base$



Yes.

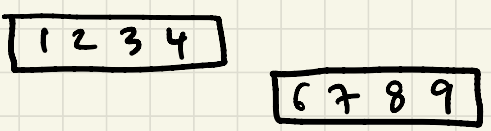
T/F : R_x_base can be $> T_x\_tail$



True.

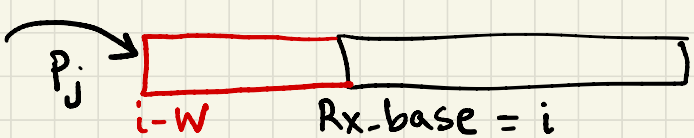
T/F : R_x_base can be $> T_x\_tail + 1$

False.



5 not sent but has been received. Impossible.

- Should R_x ACK packets that are less than R_x_base?
  - ↳ Yes should be acked



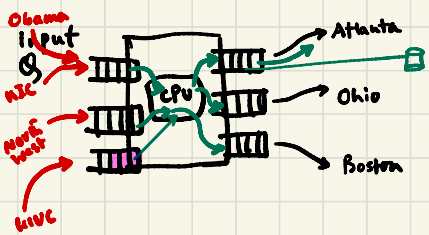
$j < i$

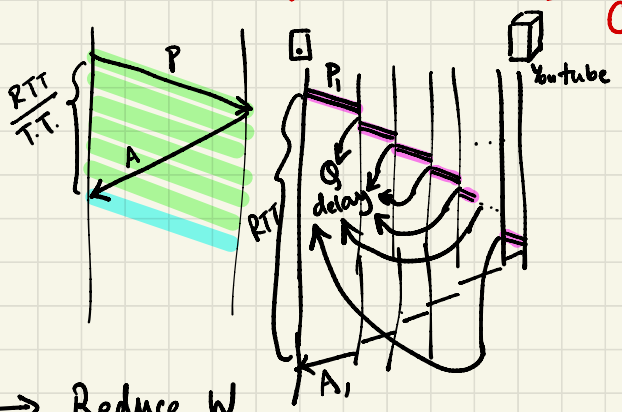
$j < (i - k)$

w = window size (equal for both Tx & Rx)

TCP (Transport Control Protocol)

↳ Congestion Control ⟶ Adaptively vary/Control window size W.





RTT goes up ⟶ Reduce W

TCP → self clocking protocol