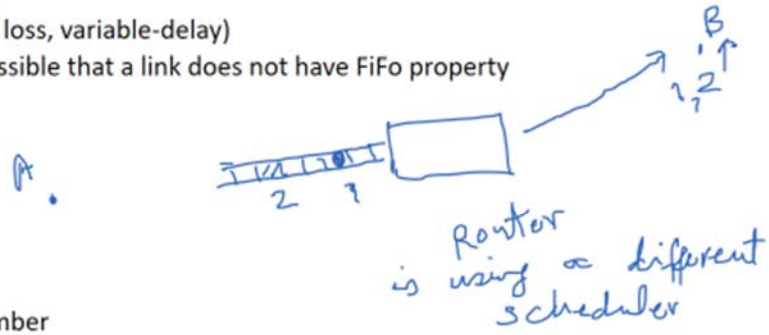
- ACK, TimeOut, Seq
  - (corruption of pkt, loss, variable-delay)
  - In internet, it is possible that a link does not have FiFo property

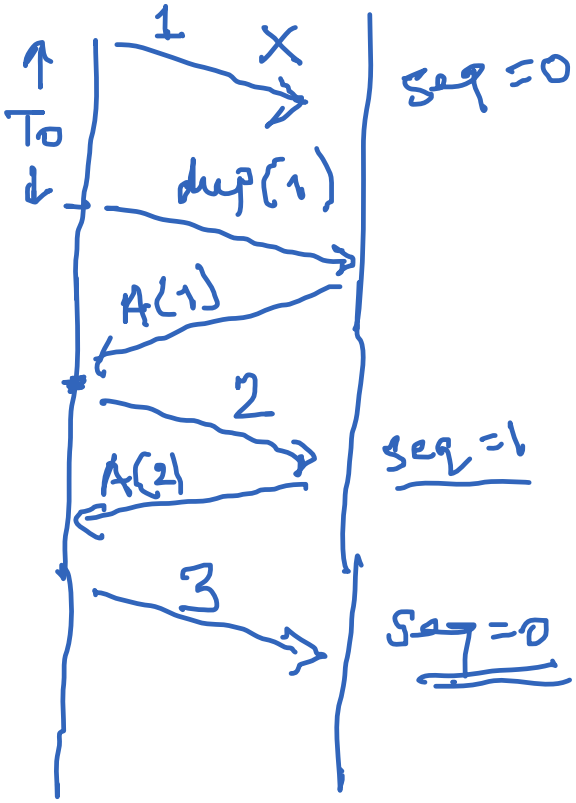


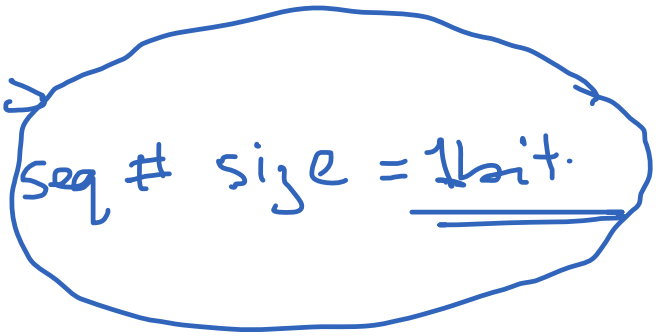
  - Size of the seq number
  - If several pkts are sent together, there may be a confusion about the pkt seq.

1st pkt didn't reach
5th pkt ≃ considered as 1st pkt

sender gets DD → ll

Limit # of pkts sent at a time = 1





What should be the seq # size if lets say, we send 4 pkts at a time?

→ 2 bit will work ?

→ 1 bit will work? X

2 pkt at a time.

Notion of Efficiency of a protocol

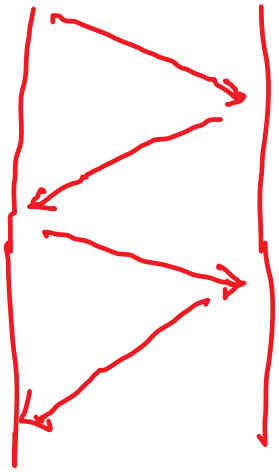


Fig-1

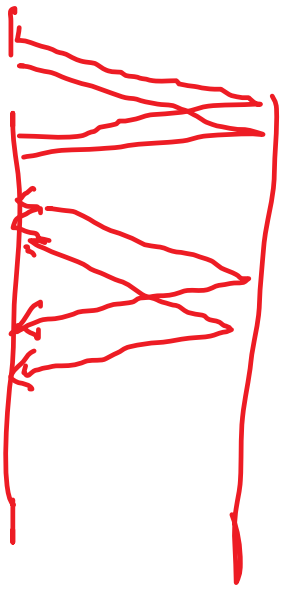


Fig-2

Networks : Bandwidth usage (bottleneck)

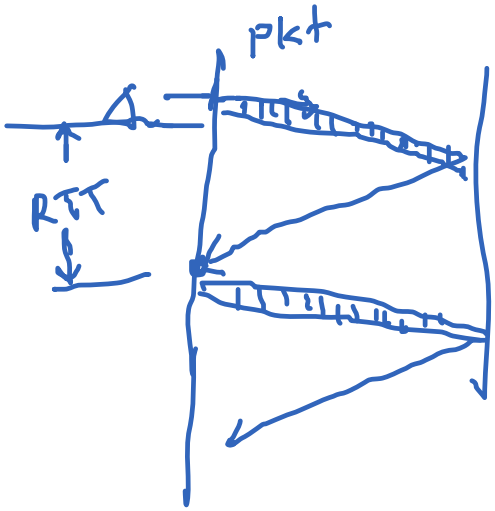
Good N/W protocol will use less b/w to do same amount of work.

η Efficiency of a protocol $\mathcal{Q}$ = $\frac{\text{b/w req. by an ideal protocol}}{\text{b/w req. by prot } \mathcal{Q}}$

≤ 1

another defn:

η = $\frac{\text{time taken to get the job done [ideal]}}{\text{time taken by the } \mathcal{Q} \text{ protocol}}$



Δ = pkt transmission time

$= \frac{\text{pkt size}}{\text{trans. rate}}$

Δ_{Ack} = negligible (Ack size ≈ 0)

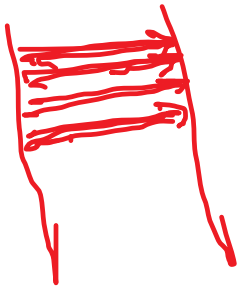
RTT = Round_Trip Time

How much time used by prot Q to send 1 pkt

$= (\Delta + RTT)$

Ideal protocol : [pkts never gets lost
no ack needed

Time to send one pkt = Δ



$\eta = \frac{\Delta}{\Delta + RTT} = \frac{1}{\left(1 + \frac{RTT}{\Delta}\right)}$

RTT = prop. delay + Queuing delay + Recv. delay

RTT $\geq \Delta$; η of single-bit sep. protocol is not good :

[STOP and WAIT]

- LAN : RTT $\approx$ 2 ms

very large pkt, low b/w : $\Delta \approx 5$ ms

e.g.

$\eta = \frac{5}{7} = .7$

Think of protocols as being represented by an FSM (finite state machine)

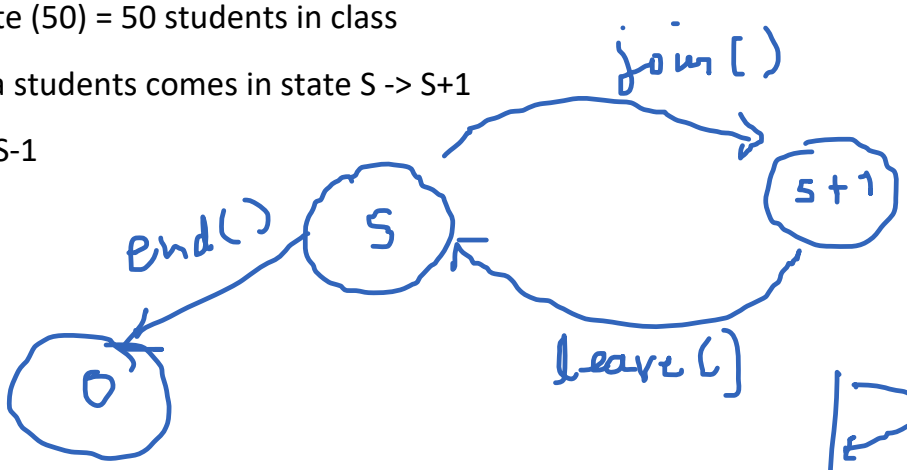
System that can be represented by a few (finite) number of parameters.

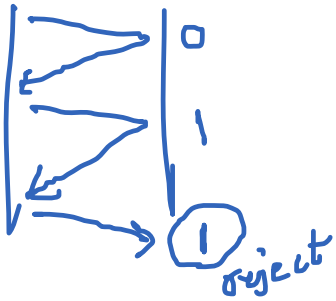
Classroom – (occupancy#)

Class1 = state (50) = 50 students in class

Every time a students comes in state S -> S+1

Leaves S -> S-1





STOP & WAIT

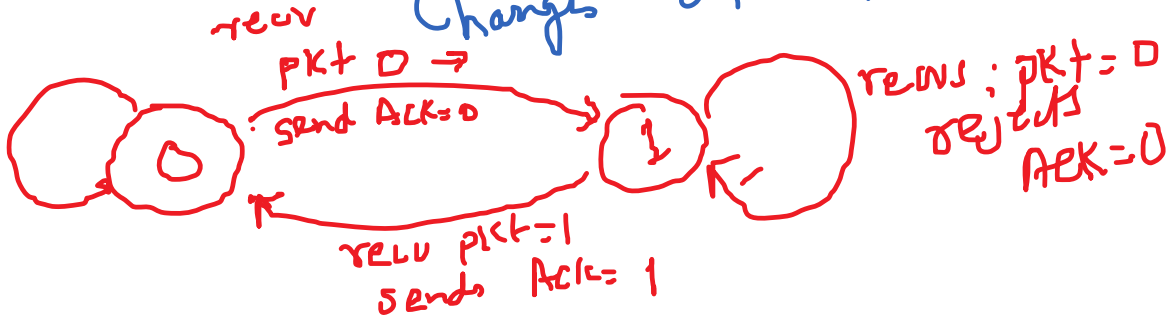
Receiver :

State 0 : waiting for pkt(seq#0)

if pkt=0 arrives, send ACK,

start waiting for seq#1

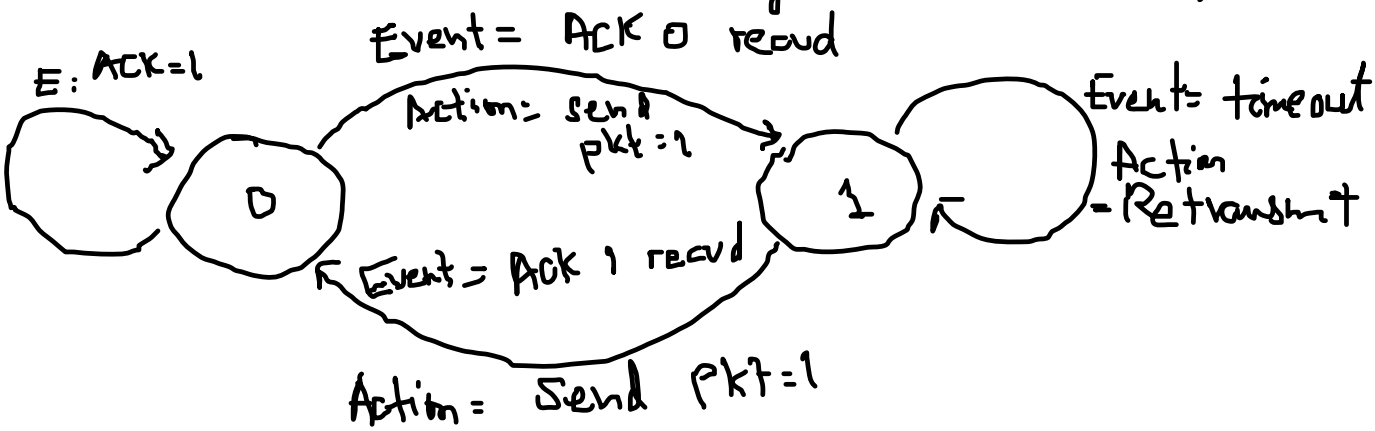
Changes state to 1



* Draw the sender FSM.

Sender FSM

- State 0 : Waiting for ACK=0
  State 1 : waiting for ACK 1



Create Combined FSM: $S(i, j)$

i : sendr param.
j : recv. param.

☐ Important: Go through the textbook section on Stop & Wait and FSM representation.