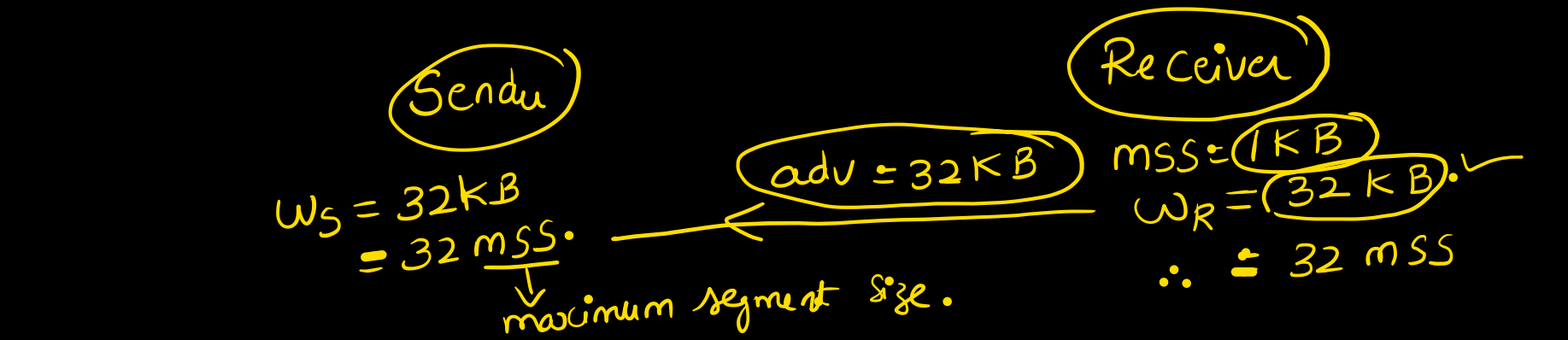
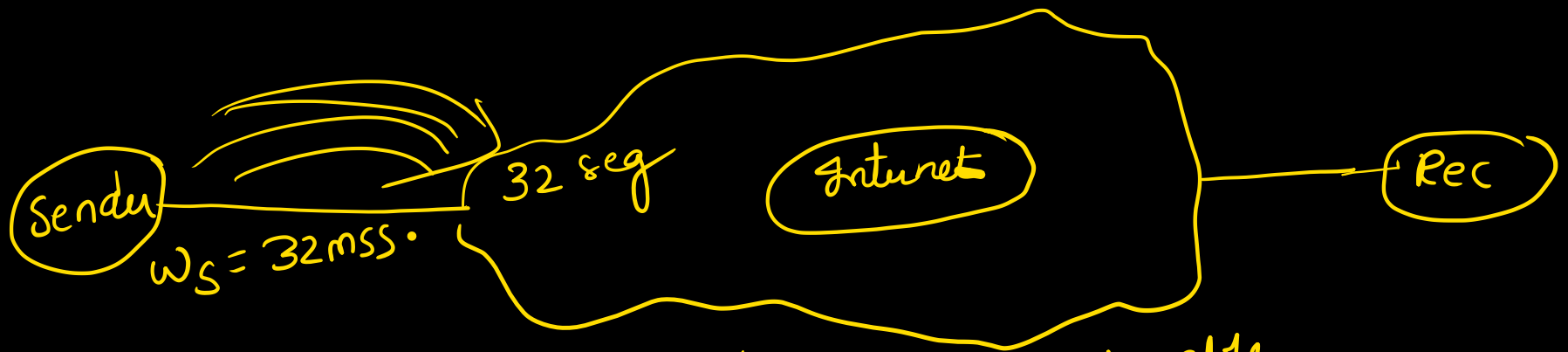
TCP Congestion Control:





N/w may not be able to handle.

Rec Capacity is known by adv win.

But N/w Capacity is not known.

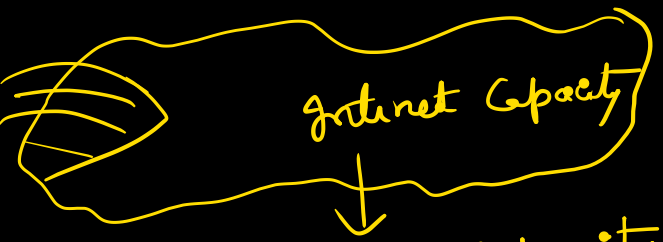
So we use Congestion control algorith

$W_C = 1\,MSS$

$W_S = \min(W_C, W_R)$

$1 \quad 32$

$W_S = 1\,MSS.$



$W_R = 32\,MSS$ ✓ (adv)

$Th = \frac{W_R}{2} = 16$ ✓

n/w capacity unknown

$W_C \rightarrow$ Congestion window $\rightarrow$ initially 1 MSS

1 2 4 8 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 32

exponentially grow till threshold

32 32

4 max.

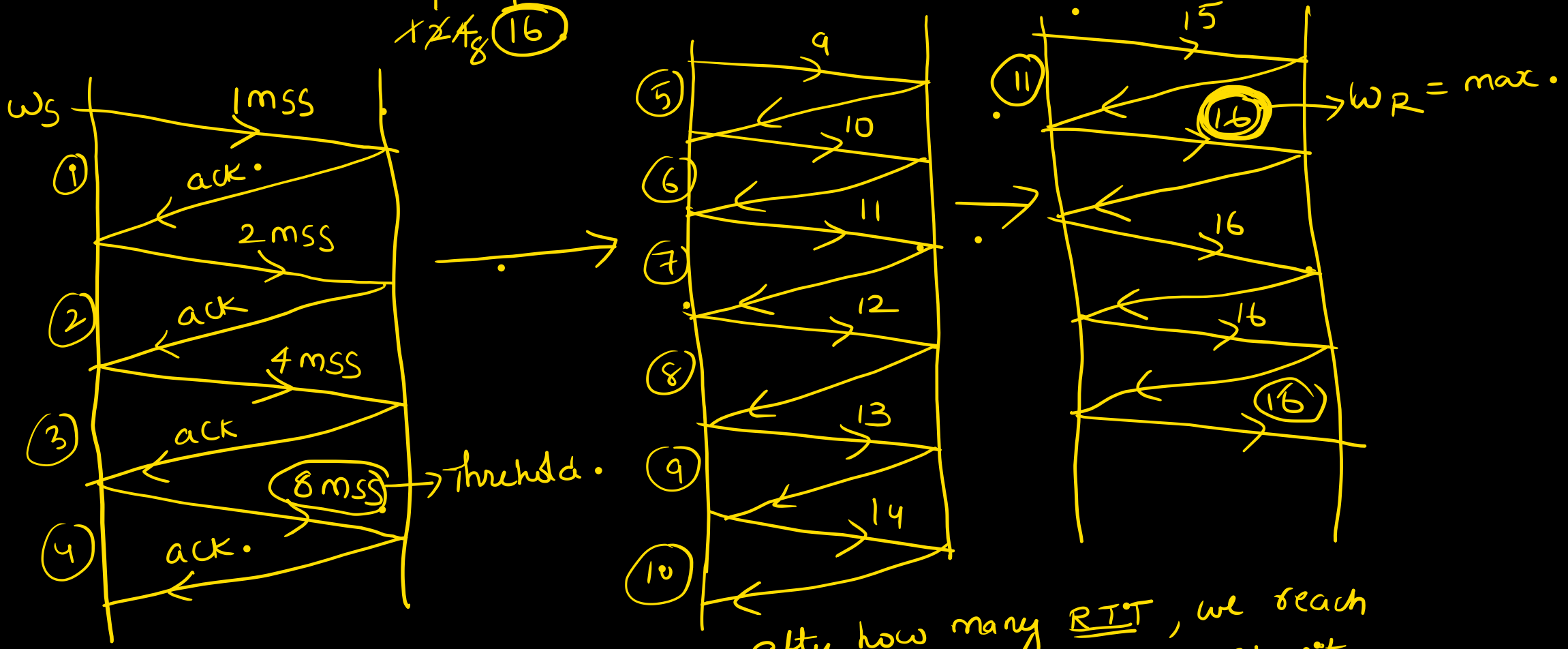
W_C = 1 mss → initially

$W_S = \min(W_C, W_R)$ = ~~1 2 4 8~~

1 ~~1 2 4 8~~ 16

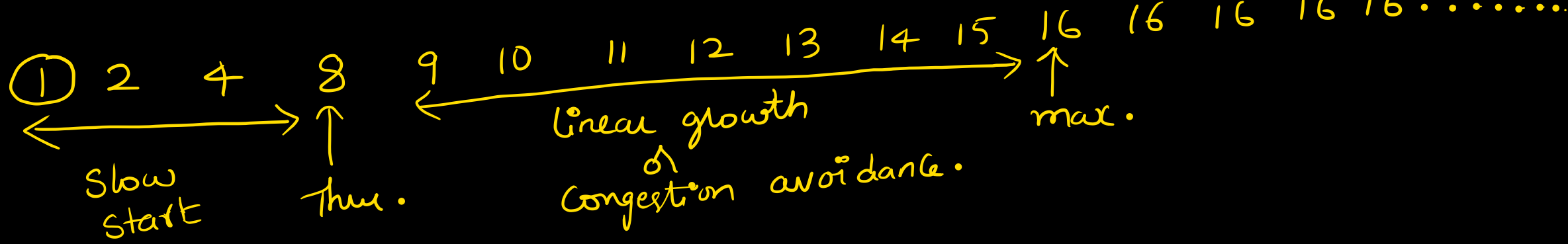
W_R = 16 mss ✓

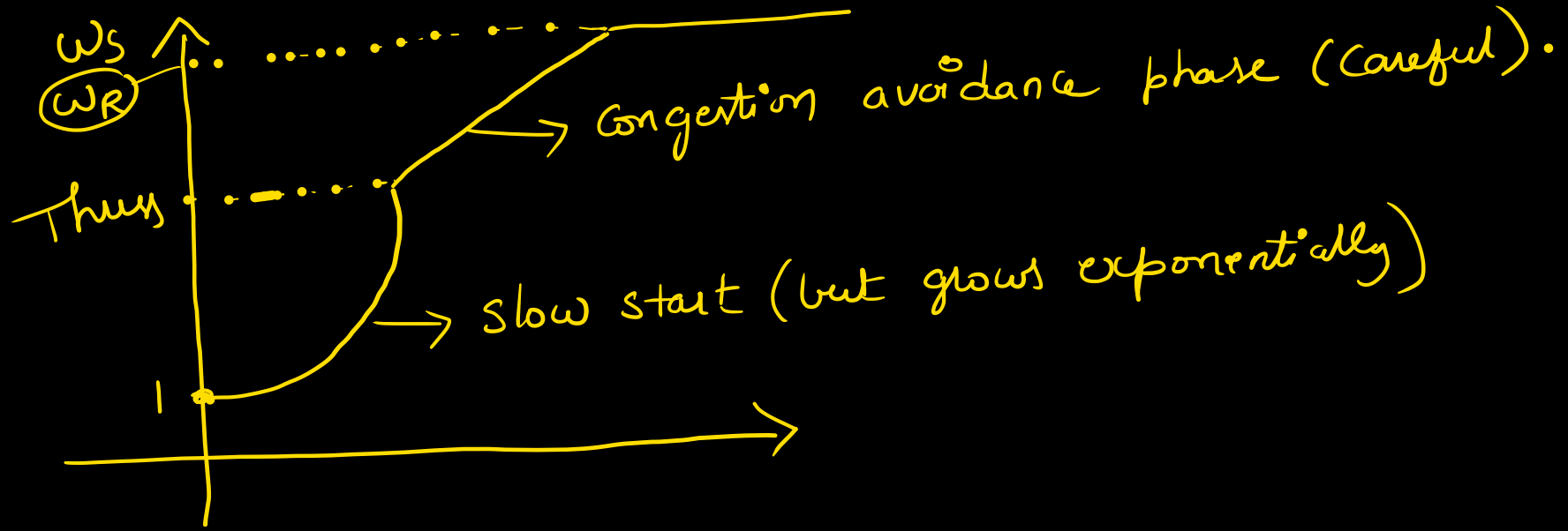
Threshold = $\frac{W_R}{2}$ = 8 mss

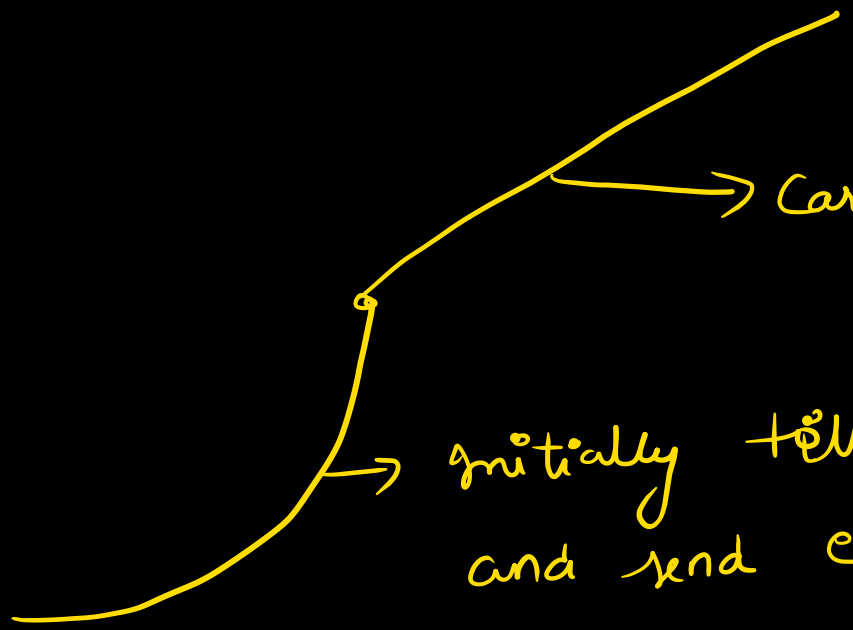


after how many RIT, we reach full capacity.

$W_R = 16$ MSS $Thr = 8$ MSS $W_C = 1$ MSS $W_S = \min(W_C, W_R)$



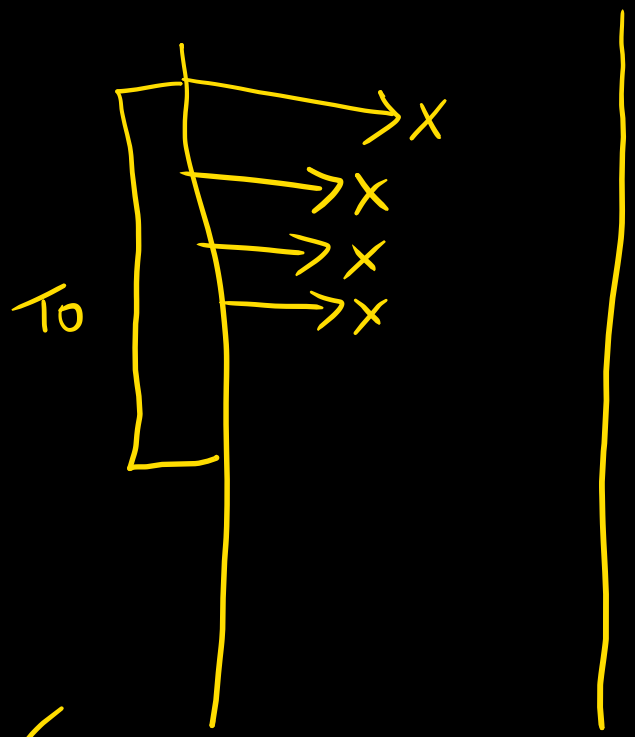



Careful to avoid congestion.
Initially till threshold, we we will be careless and send exponential packets.

what is congestion → All the n/w devices are too busy.

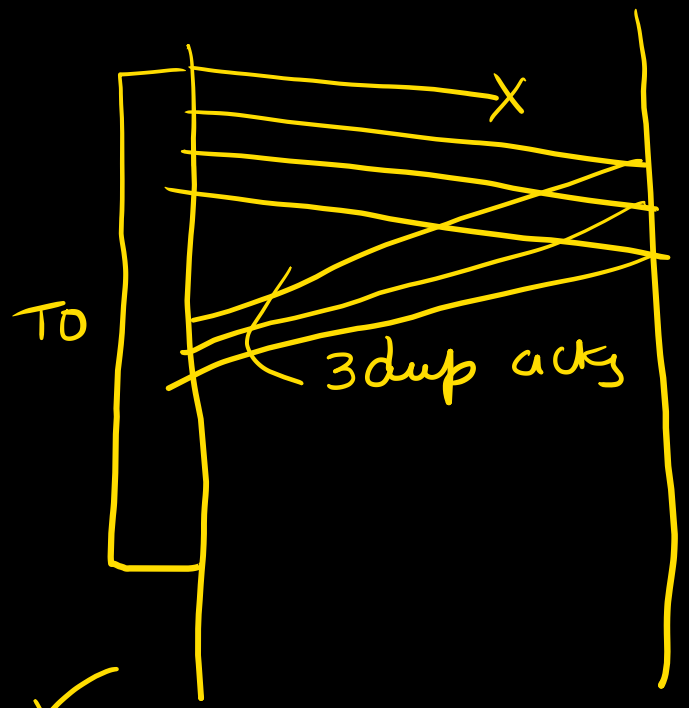
- Packects will be discarded.
- How will we know Congestion → when packet is lost.

How will know a packet is lost? TO, 3 dup ack.



a) TO will occur

Severe Congestion



b) 3 dup ack.

mild Congestion

ICMP source quench

↓

not reg

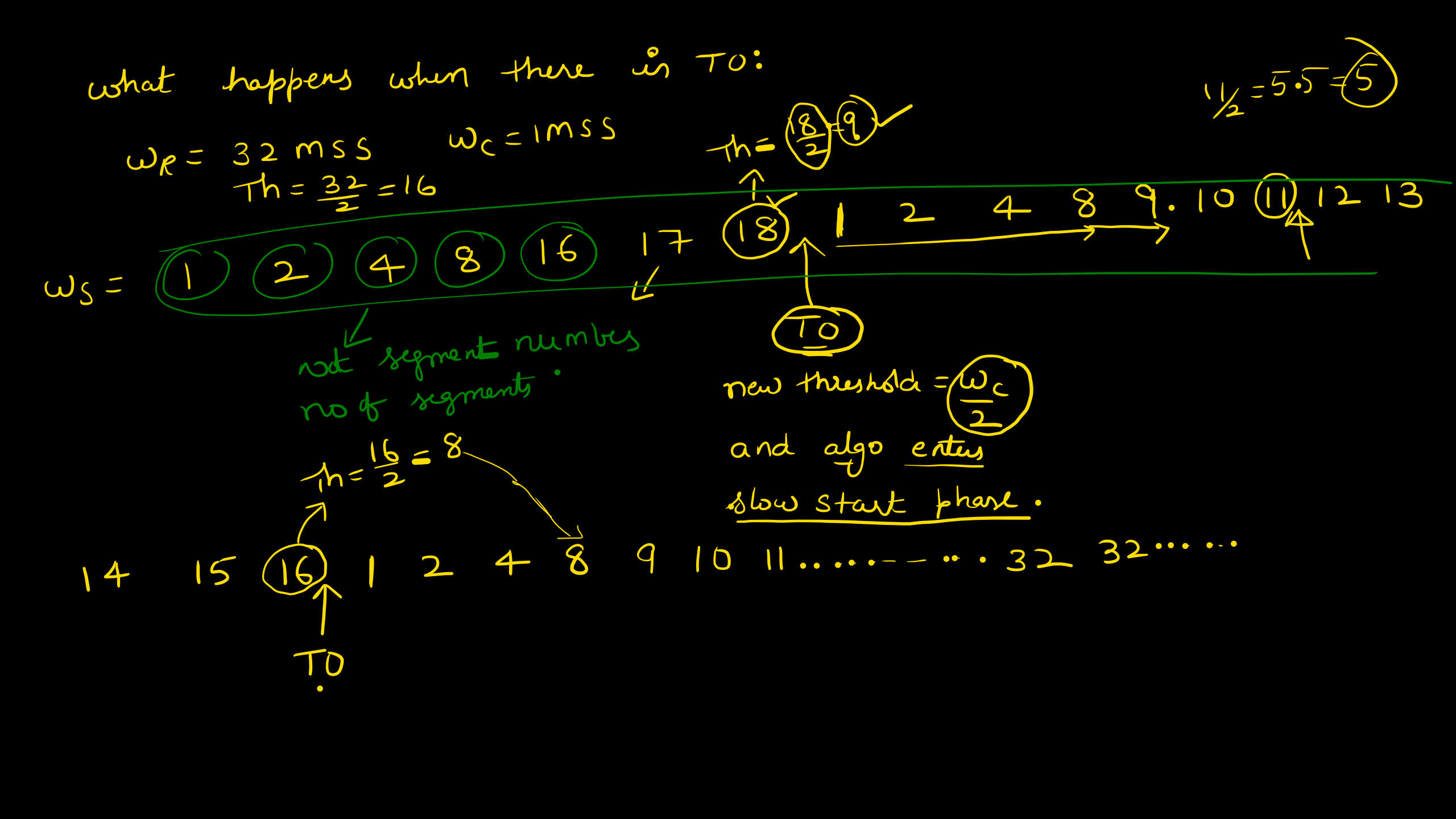
∴ Congestion algo will react differently for TO and 3 dup acks.

TCP Congestion Control algorithm:

Phase 1: Slow start phase
Phase 2: Congestion avoidance phase } seen

Phase 3: Congestion detection phase

a) Time out (Severe Congestion)
b) 3 dup ack (mild Congestion)
c) ICMP Source quench (not required)


what happens when there is TO:
w_R = 32 mss
Th = $\frac{32}{2}$ = 16
w_C = 1 mss
w_S = 1 2 4 8 16 17
not segment numbers
no of segments.
Th = $\frac{16}{2}$ = 8
14 15 16 1 2 4 8 9 10 11 ..... 32 32 .....
TO
Th = $\frac{18}{2}$ = 9
18 1 2 4 8 9 10 11 12 13
TO
new threshold = $\frac{w_c}{2}$
and algo enters
slow start phase.
11½ = 5.5 = 5

what happens if there are 3 dup acks: → Congestion is mild, so don't start from (1).

$W_R = 32$ mss $\quad W_C = 1$ mss

Th = 16 mss.

1 2 4 8 16 17 18 19 (20) 10 11 12 13 14

$Th = \frac{W_C}{2} = 10$

3 dup acks. and algo enters Congestion avoidance phase.

$Th = \frac{W_C}{2} = 8$

15 (16) 8 9 10 11 12 13 ..... 32...

3 dup ack.

Ex: Combine both : ~~ms~~ $w_R = 32$ mss, $w_c = 1$ mss, Th = 16.

Th = $\frac{17}{2}$ = 8 ✓

Th = 5

1 2 4 8 16 17 1 2 4 8 9 10 5 6 7 .....

To

3 dup

TCP timer management :

(i) Time - wait timer

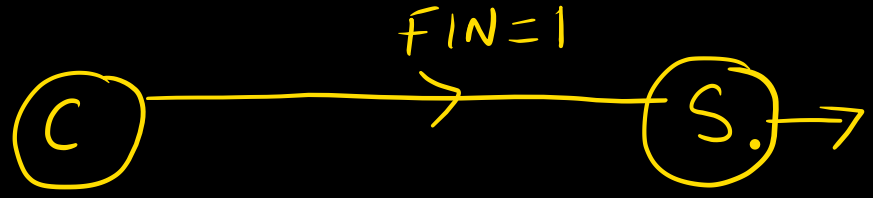
(ii) Keep alive timer

(iii) Persistent timer

(iv) Acknowledgment timer

(v) Time out timer.

Time wait timer: ∴ Time wait timer = 2 MSL



Should not close connection immediately and it has to wait for late packets.

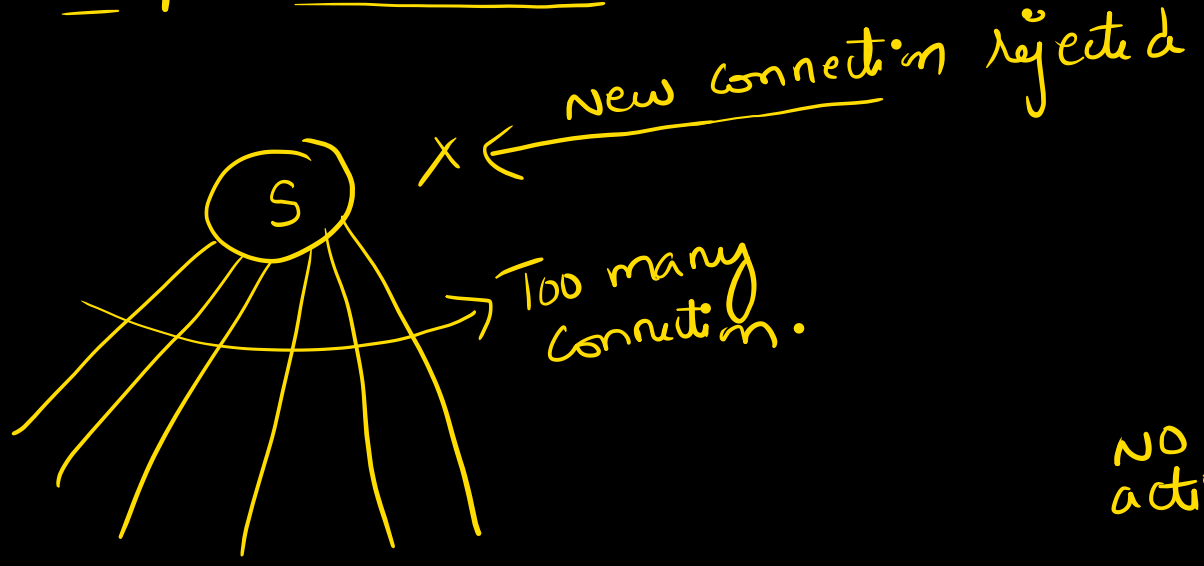
It should wait for 2 MSL → max segment life time.

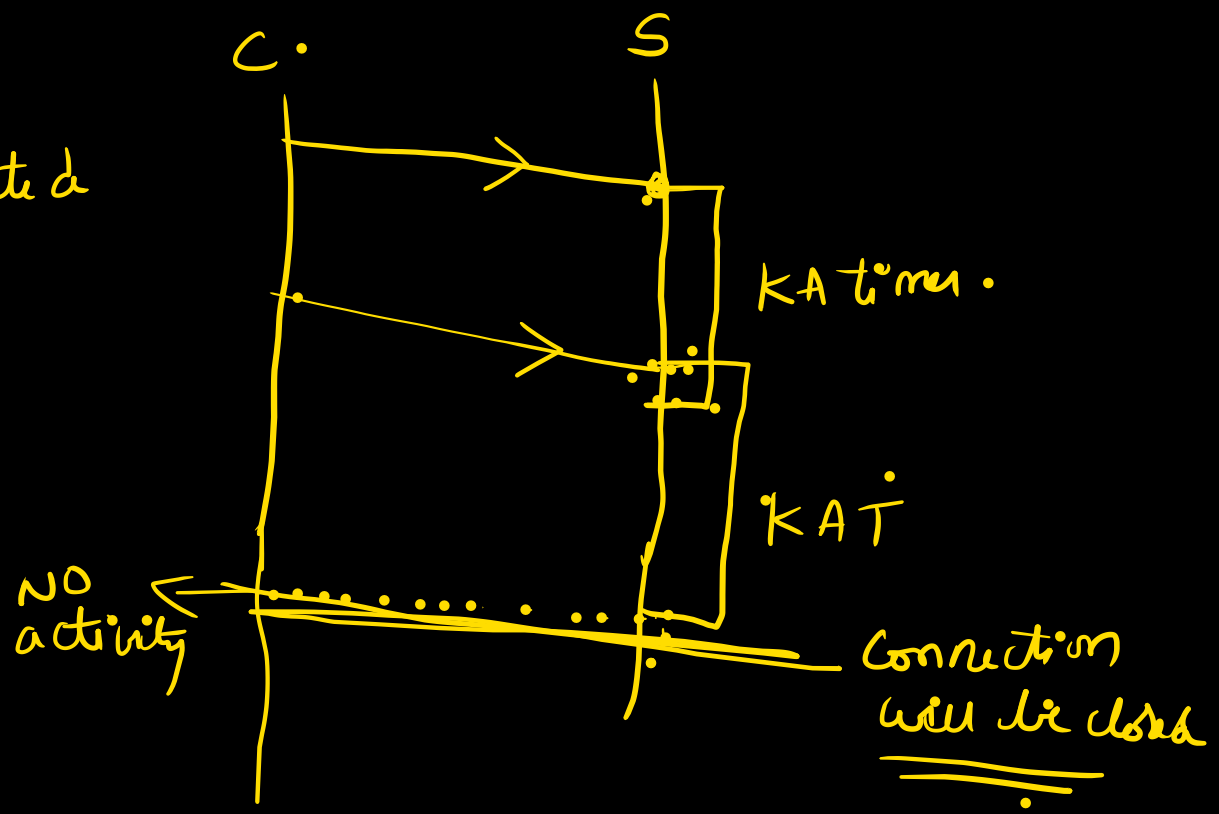
MSL = 10 sec / 30 sec / 1 min / 2 min / 3 min.

↓

depends on implementation.

Keep alive timer:

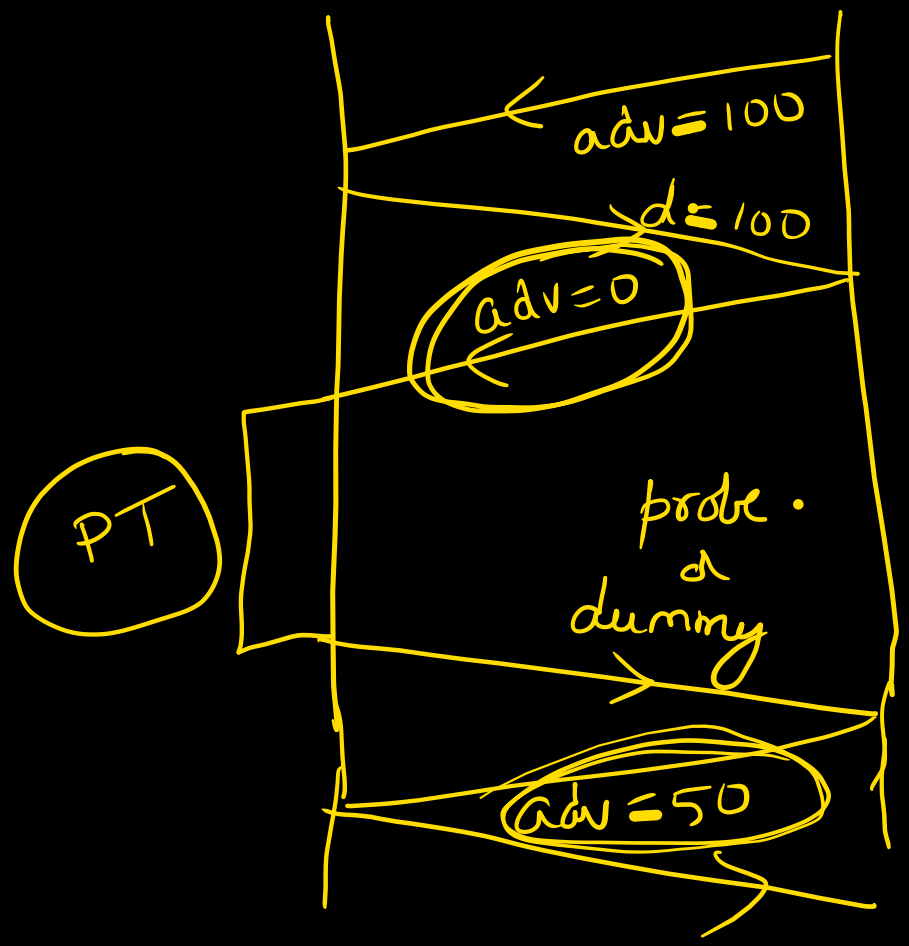




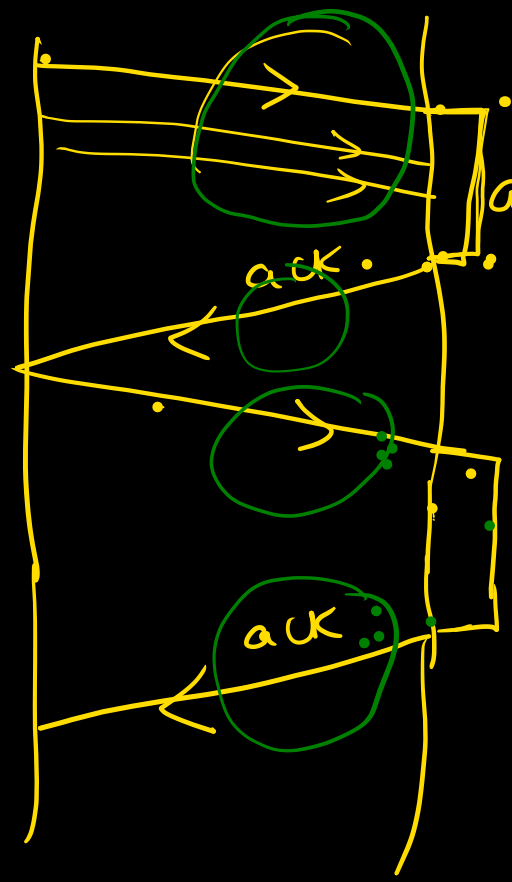
Server should close idle connections.

To check idle connections keep alive timer is used.

Persistent timer:



acknowledgement timer:



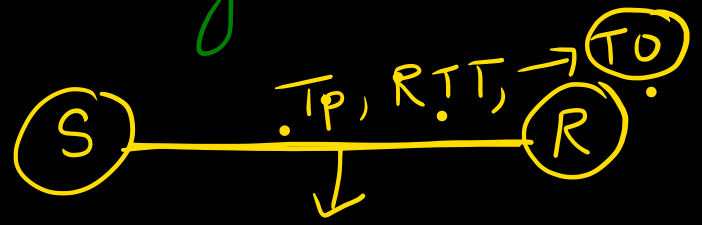
ack timer is used to implement cummulative acks.

↓

for many packets one ack.

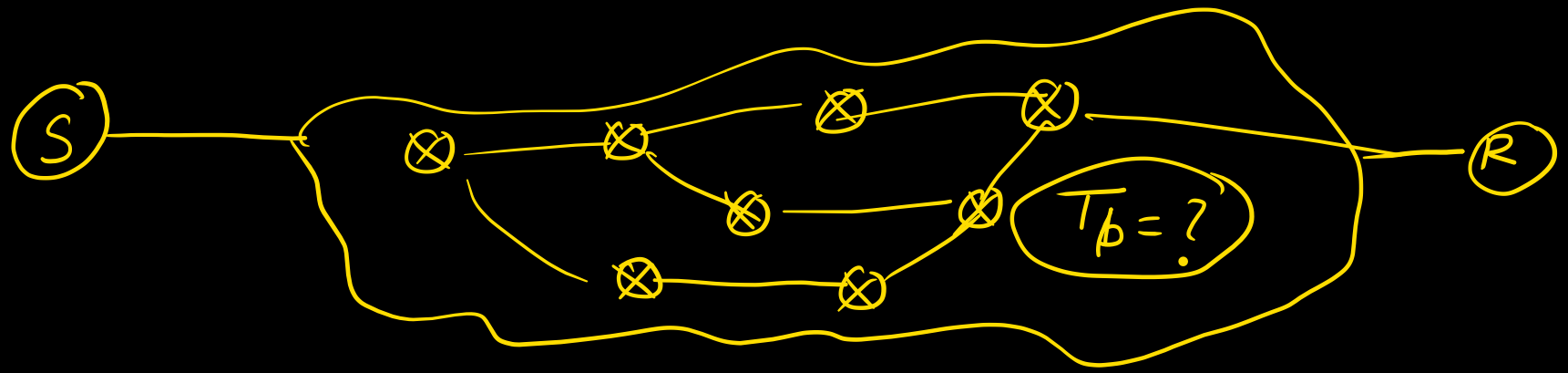
Time out time:

Generally in one hop $T_p = \frac{d}{v}$, $RTT = 2 * T_p$, $TO = 2 * RTT$



one hop.

But in end to end, we dont know $T_p = ?$.

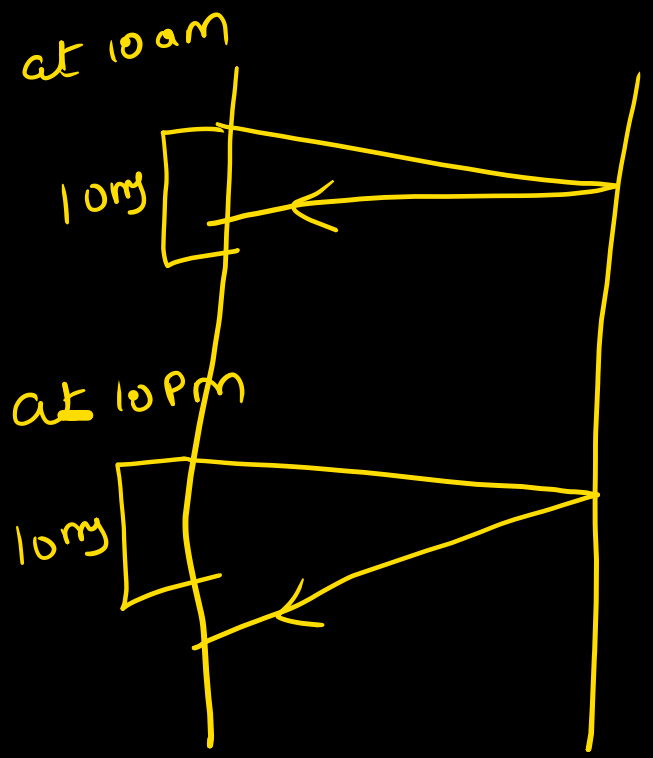


Every packet takes a diff route.

→ In DLL → one hop → RTT → fixed
→ TO → fixed.

hop
– one edge

→ In TL → many hops → RTT → values
TO → dynamic.
↓
Keeps changing.



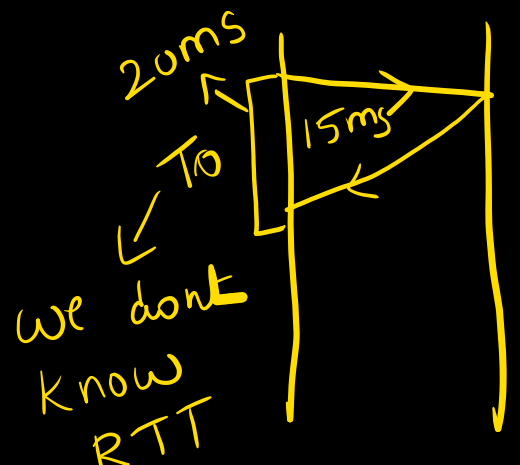
∴ RTT changes based on traffic.

So TO should change based on RTT

So to Calculate TO in TCP, we have 2 algorithms

1) Basic
2) Jacobsons

Basic algorithm for Timeout Calculation:



$\therefore$ Initial RTT = 10ms (Guess) I packet.

$\therefore$ TO = 2 * RTT = 20ms.

actual RTT = 15 ms

Next RTT = ? α (IRTT) + (1-α) ARTT

α : Smoothening factor

If $\alpha = 1$ NRTT = IRTT → Static

$\alpha = 0$ NRTT = ARTT → Too dyamic.

Generally $0 \leq \alpha \leq 1$

Let $\alpha = 0.5$ for this example.

IRTT = 10 ms (guess)

TO = 2 * IRTT

= 20 ms

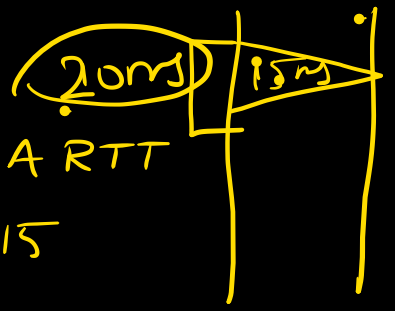
ARTT = 15 ms

NRTT = α(IRTT) + (1 − α) ARTT

= 0.5 * 10 + 0.5 * 15

= 12.5 ms

I packet.



IRTT = 12.5ms ✓

TO = 2 * IRTT

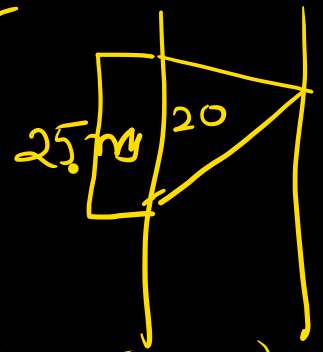
= 25 ms.

ARTT = 20ms ✓

NRTT = α (IRTT) + (1−α)(ARTT)

= 16.25 ms.

II



IRTT = 16.25ms III

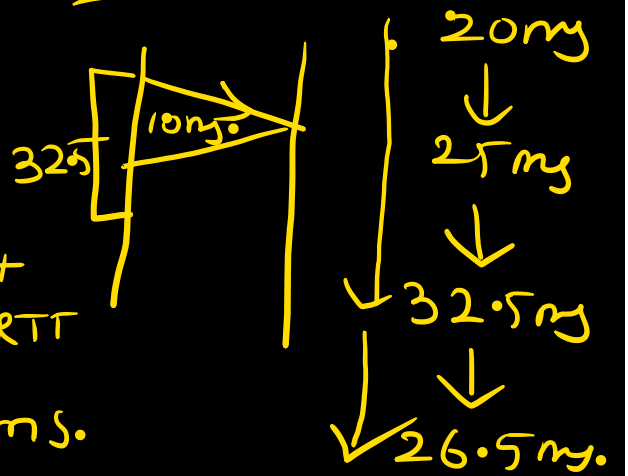
TO = 2 x RTT

= 32.5 ms

ARTT = 10ms

NRTT = α IRTT + (1 − α) ARTT

= 13.125 ms.



IRTT = 13.125ms IV

TO = 2 x IRTT

= 26.25ms.

TO = 2 * RTT

NO logic

Jacobson's algo.

Jacobsons algorithm:

IRTT = 10 ms (guess)

Initial deviation = 5 ms (guess)

RTT 10 ± 5 → (5 -15)

TO = 4 * D + RTT → Diff from Basic algo → TO = 2 * RTT

= 4 * 5 + 10

= 30 ms.

ARTT = 20 ms

AD = 10 ms ✓ | IRTT - ARTT |

NRTT = α (IRTT) + (1 - α) (ARTT) = 15 ms.

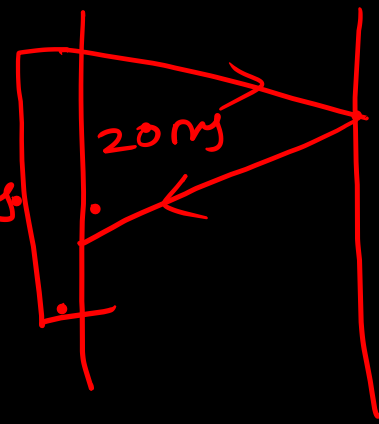
IRTT → II

ND = α (ID) + (1 - α) (AD)

= 7.5 ms.

ID → II

I



II

assume α = 0.5,

II packet

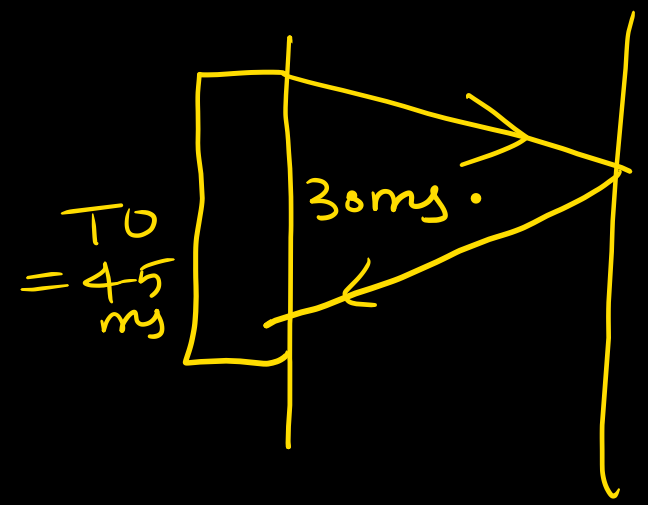
IRTT = 15 ms

ID = 7.5 ms

TO = 4XD + RTT

= 45 ms.

ARTT = 30 ms.



AD = 15 ms.

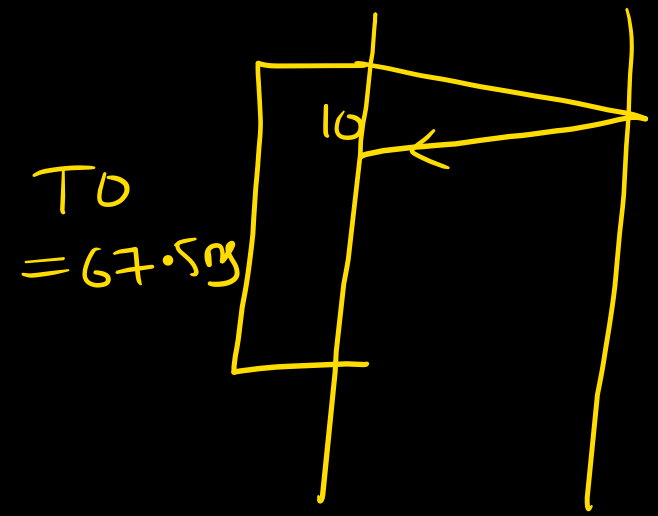
NRTT = 22.5 ms

ND = 11.25 ms

→ III packet.

III packet:

- SRTT = 22.5 ms
- SD = 11.25 ms
- TO = 4 × D + RTT
  = 67.5 ms.
- ARTT = 10 ms



AD = 12.5 ms.

NRTT = 16.25 ms.

ND = 11.875 ms.

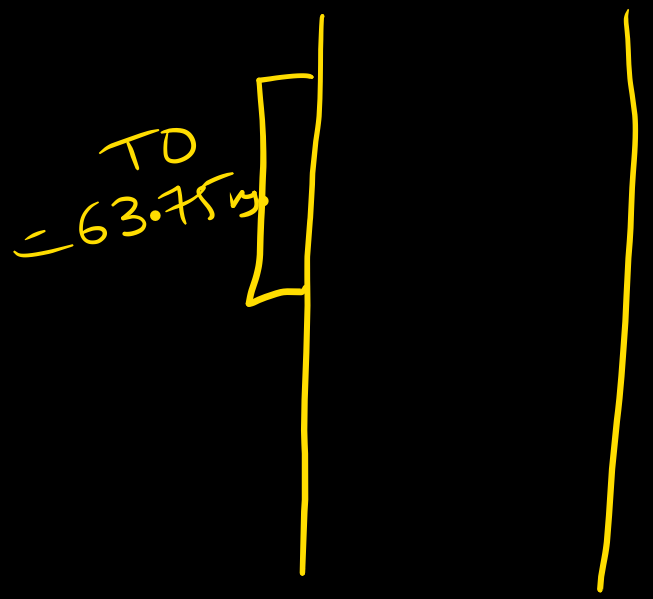
→ IV packet.

IV packet :

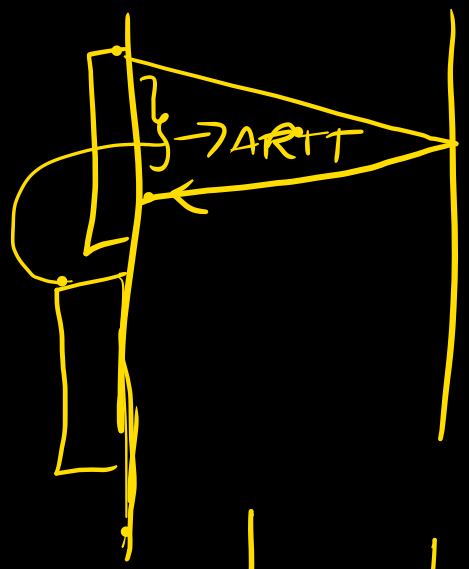
$IRTT = 16.25\ ms$

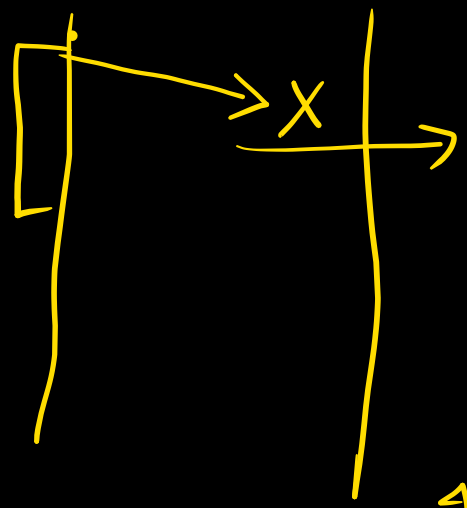
$ID = 11.875\ ms$

$TO = 4 \times D + RTT = 63.75\ ms.$



Both Basic and Jacobson's algo are based on ARTT → actual Round trip time.





what is ARTT, what is TO for next packet.

↓

Kaun modification.

It says that whenever a packet is lost, just double the previous timeout

Keep doubling