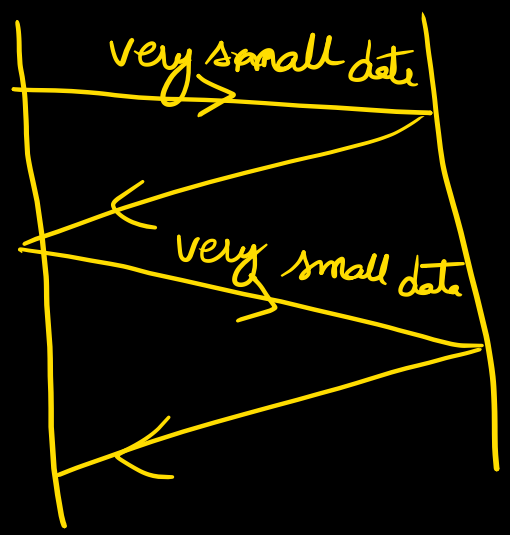
Silly window Syndrome:



(i) If Receiver is too slow
(ii) If sender is too slow
(iii) If Receiver consumes only 1B at at time.

(i) If Receiver is too slow: (adv = 0)

win is full.

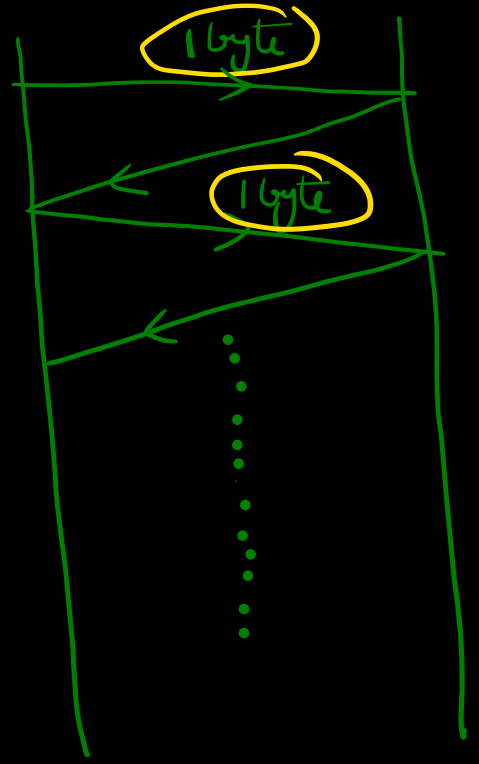
Solution: Restart the connection

RST = 1

Reset flag.

forever because Receiver is slow.

(ii) Sender is too slow and sending small packets:

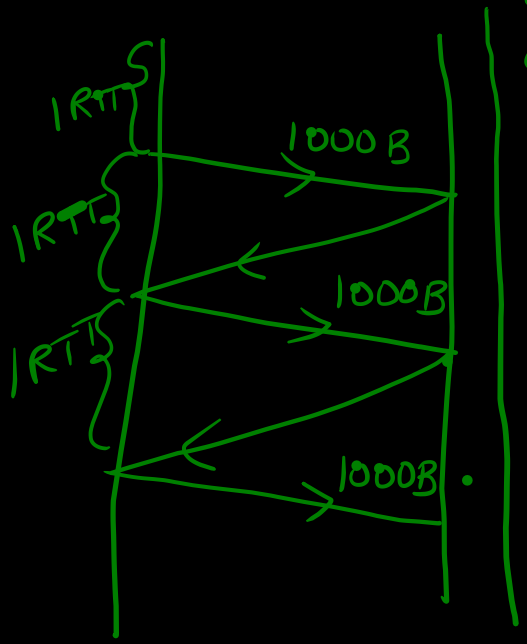


Sol: Nagle's algorithm

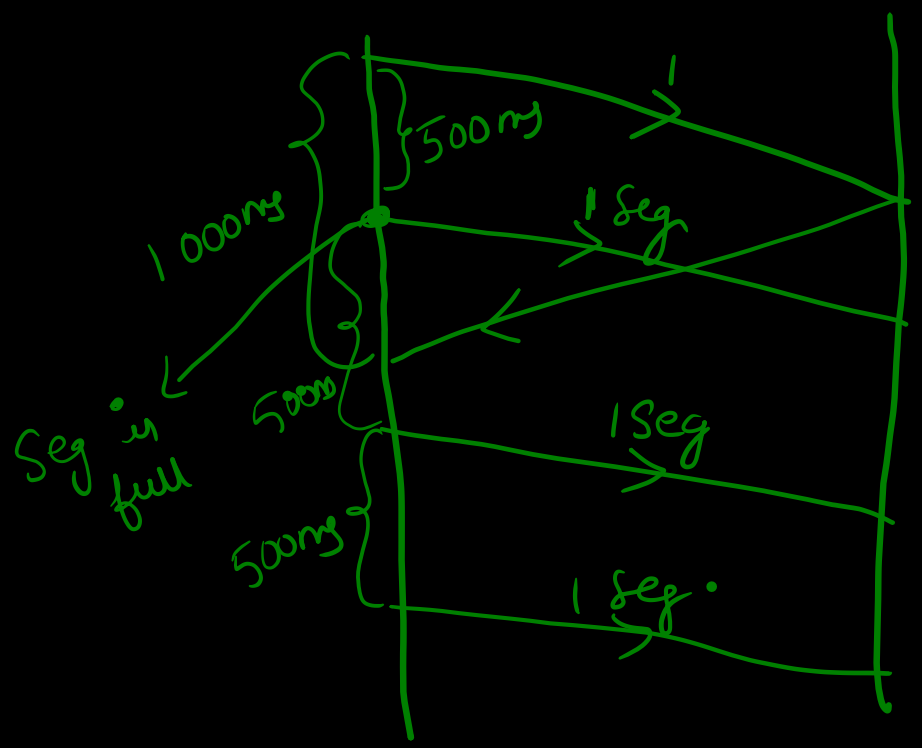
It says that → wait for 1 RTT and how much ever data is produced by AL in 1 RTT, can be ~~set~~ sent in one segment.

Ex: If Sender can produce 1B/ms and RTT = 1000 ms.

In 1 RTT = ? 1 RTT = 1000 B can be produced.



condition, if we get 1 segment size then send immediately.



1 RTT = 1000 ms, Sender speed is 1 B/ms

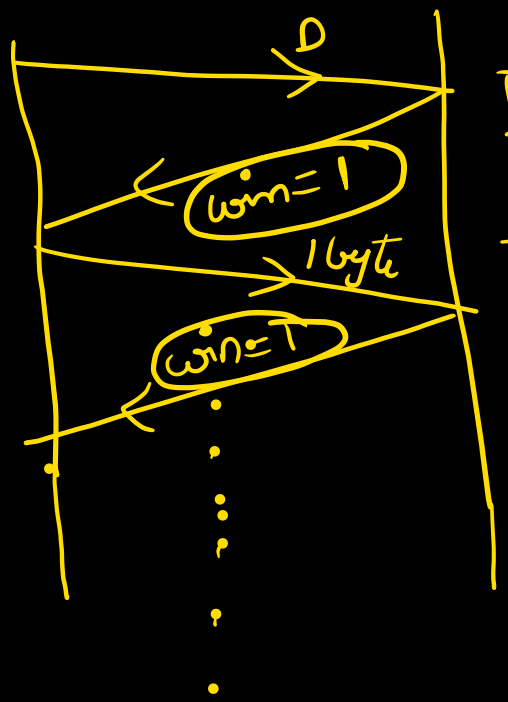
1 Seg Size = 500 B.

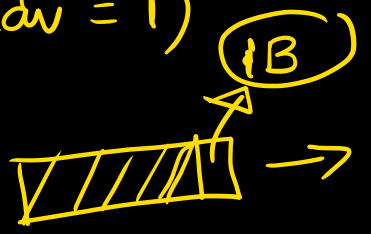
In 1 RTT → 1000 B → 2 Segments.

in 1/2 RTT → 1 Segment is produced

Nagles → wait for 1 seg of 1 RTT

(iii) Receiver is too slow: (adv = 1)

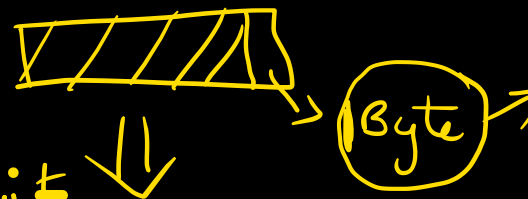


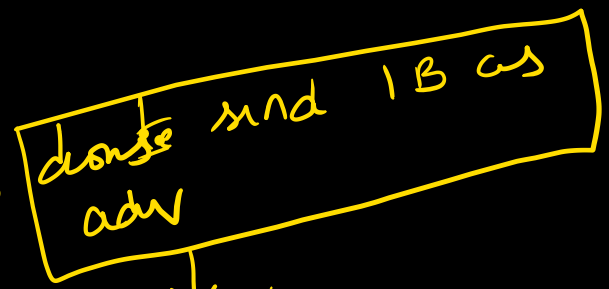


→ Rec is consuming only one Byte at a time.

Sol: clock solution:

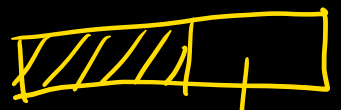
Rec should wait at least till ½ of buffer or 1 MSS and then advertise the window

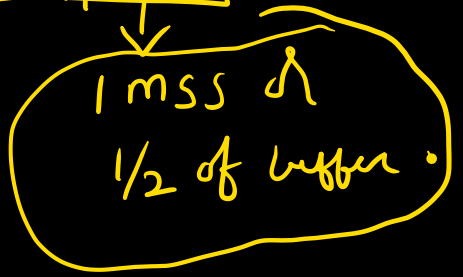




wait

wait



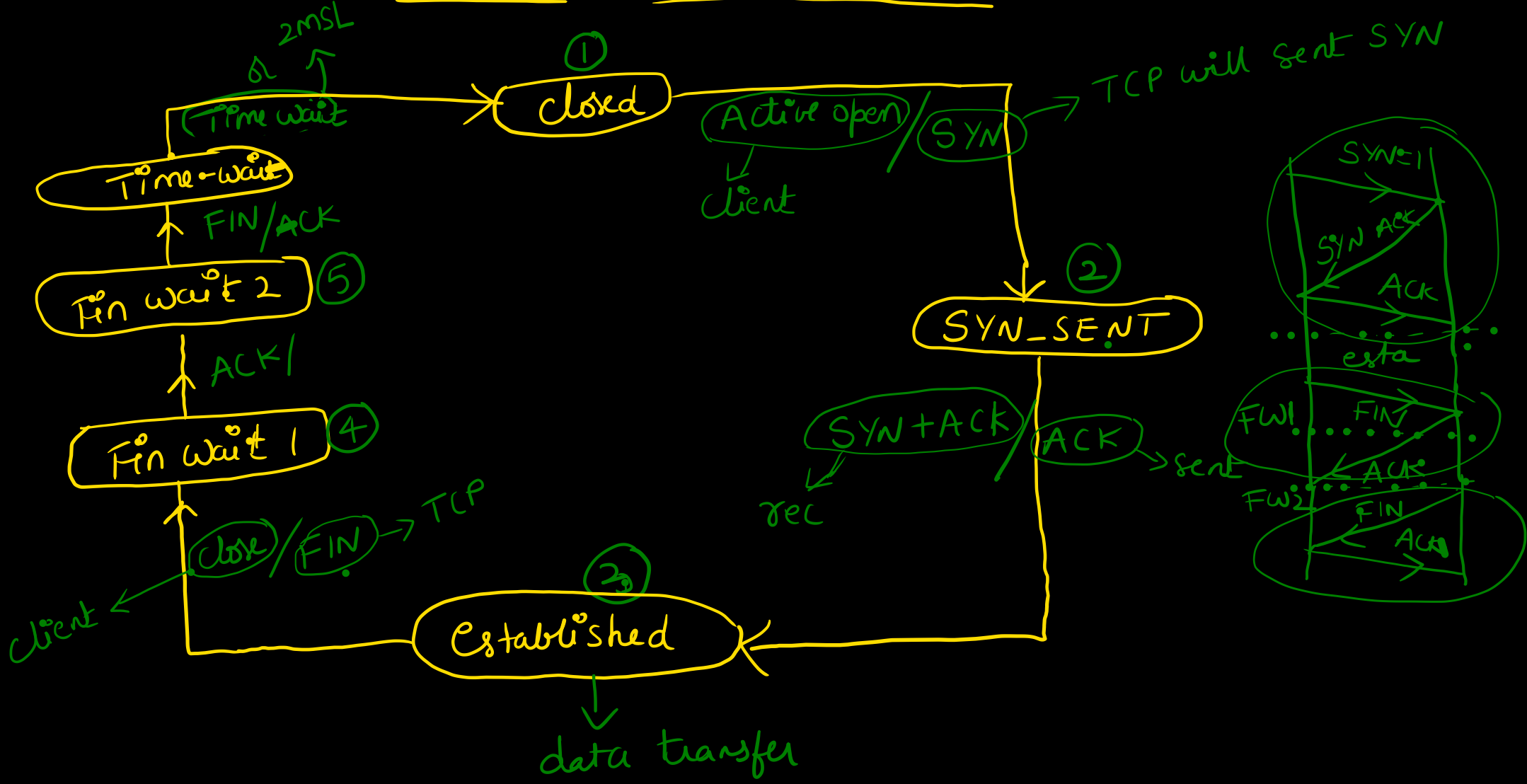


5 min break:

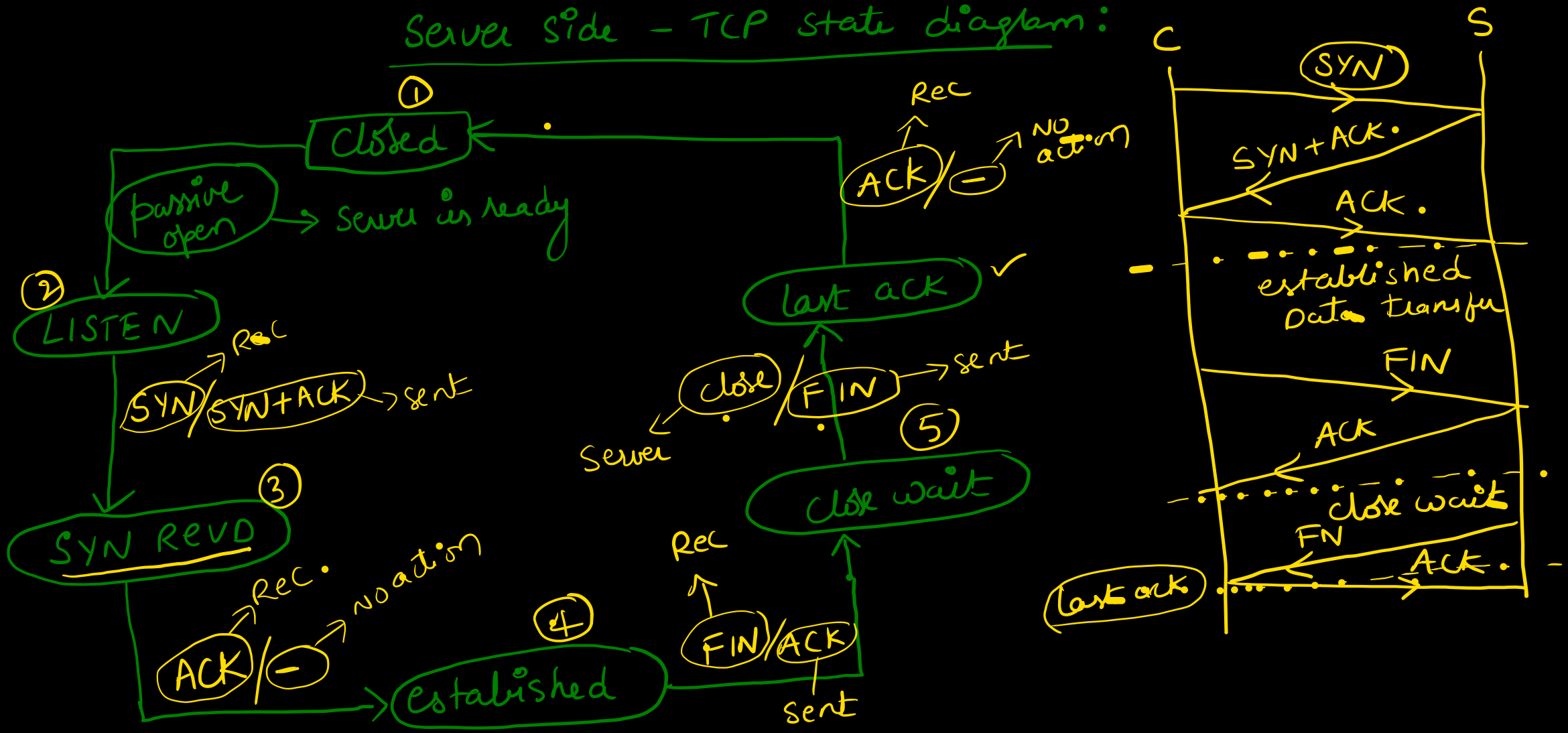
Traffic shaping:

CN will be over in 2 days.

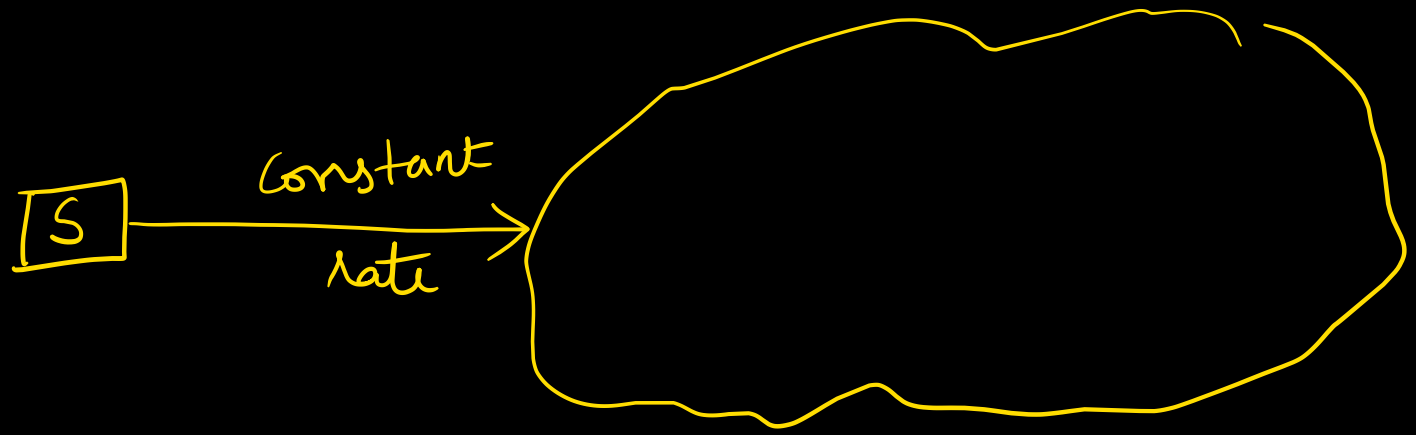
Client side TCP State diagram:



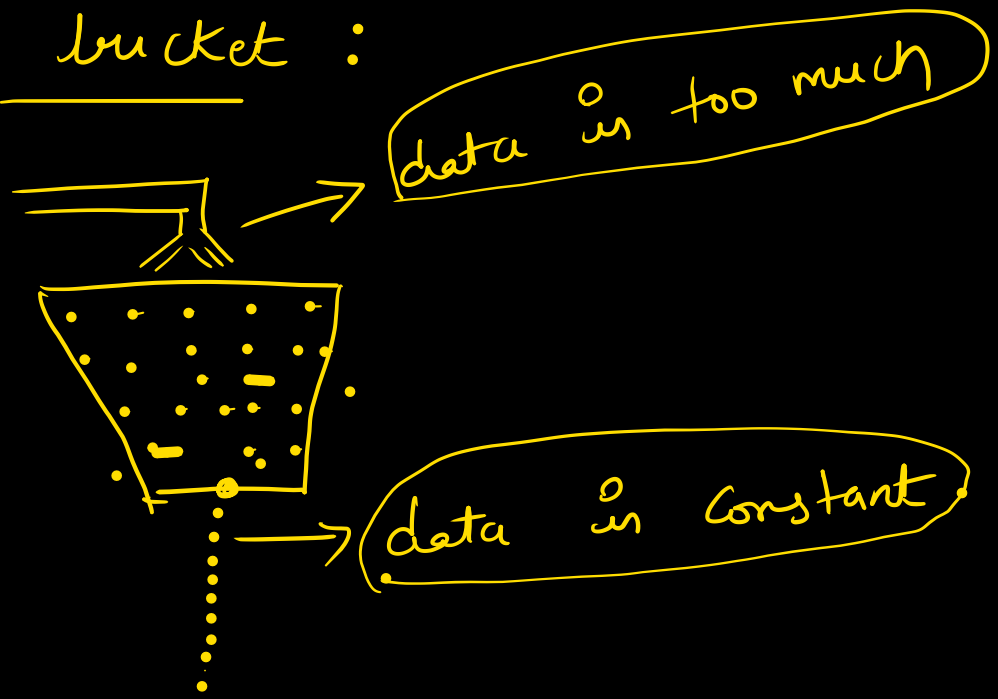
Server side - TCP state diagram:



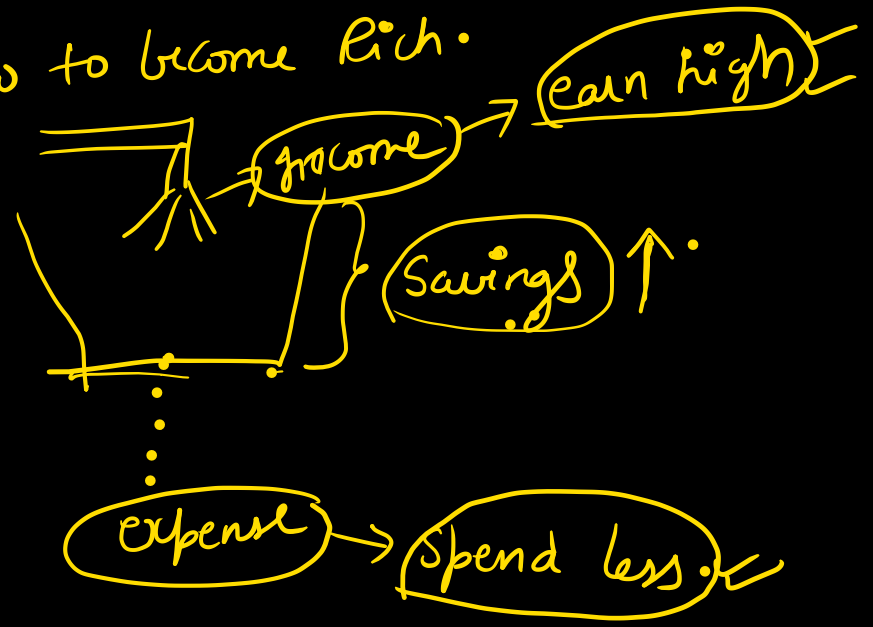
Congestion control at IP layer: (Traffic shaping)



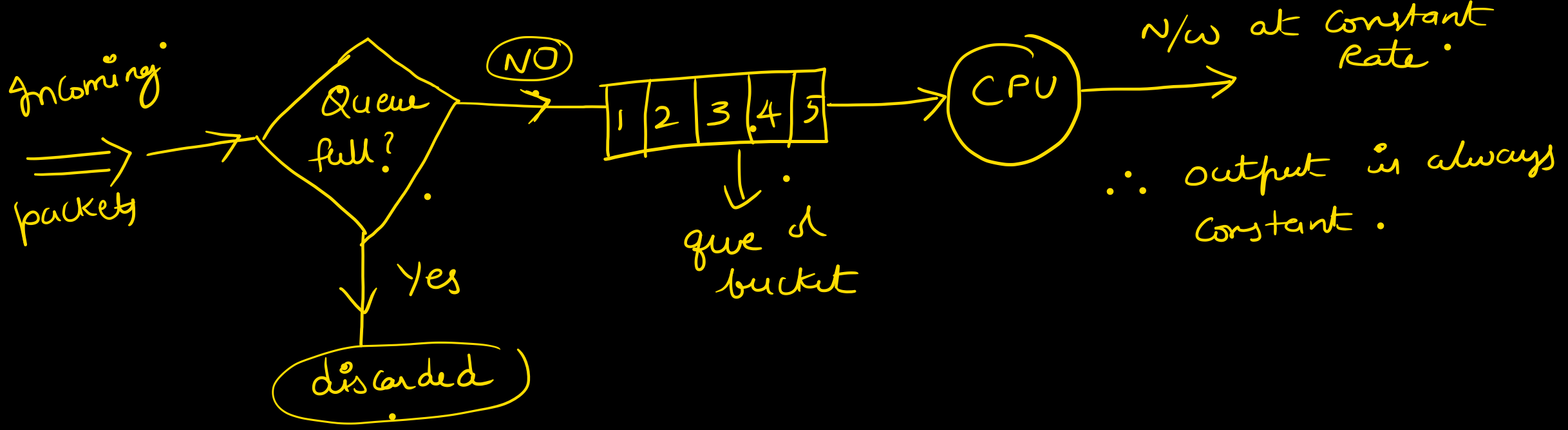
Leaky bucket :



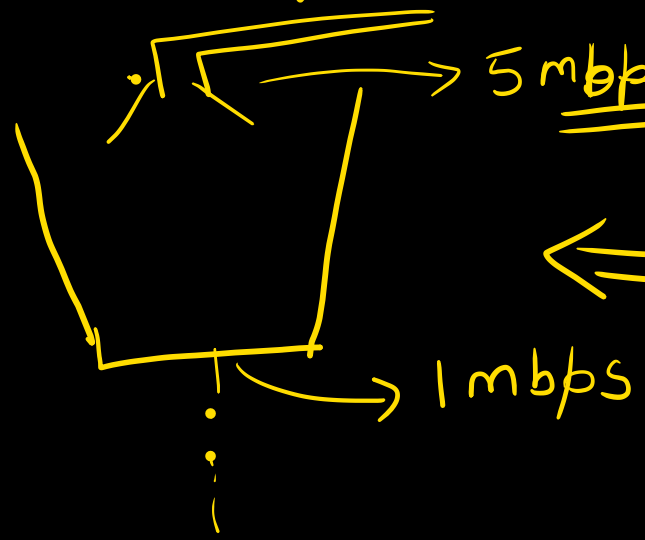
How to become Rich.



Leaky Bucket :

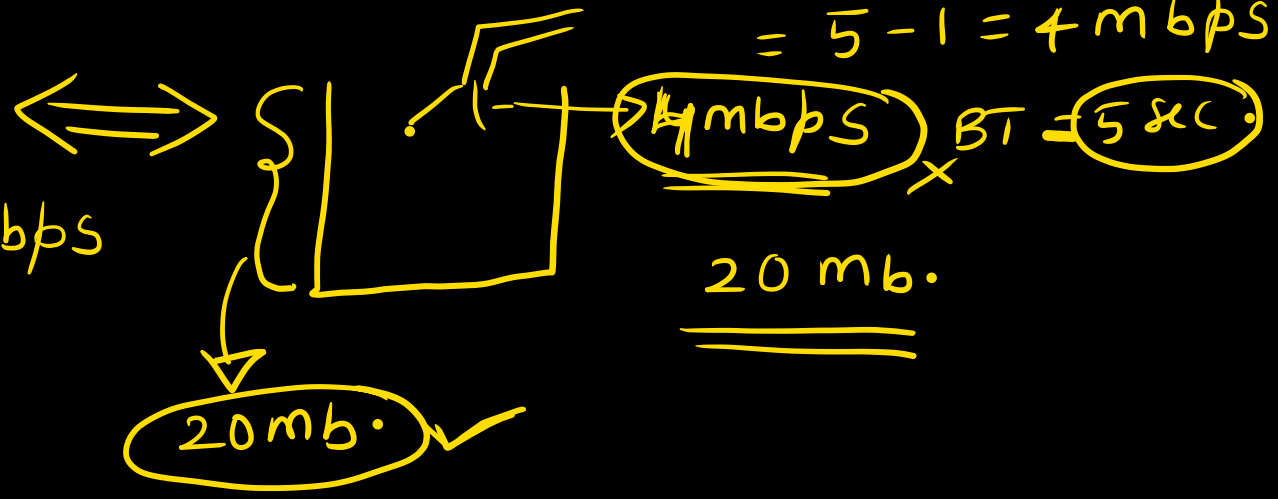


If inflow = 5 mbps and out flow = 1 mbps and the burst time = 5 sec, then what is the size of leaky bucket

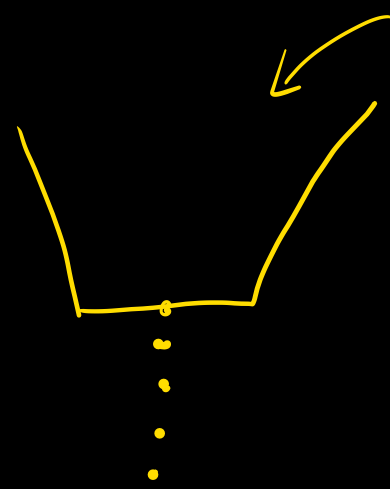


net inflow = inflow − outflow

= 5 − 1 = 4 mbps.



Token Bucket :

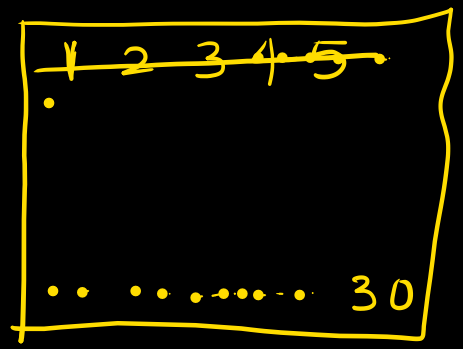


No inflow for some time

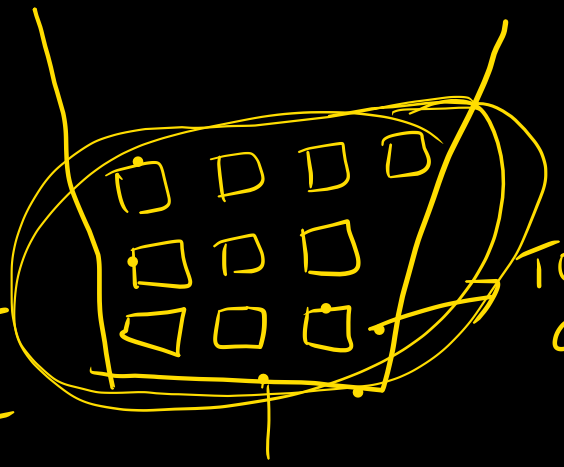
↓

Still sends
Cannot send more packets later

mess card



1 → didn't eat
2 - " "
3 - " "
4 - " "
5 → You can eat all

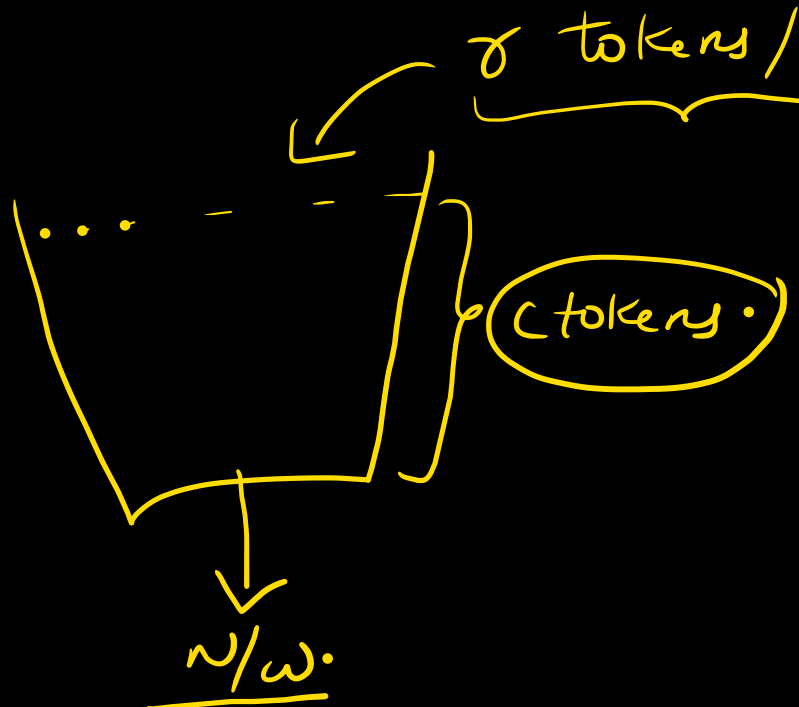


On a token bucket

Tokens will be accumulated

all of them at one

Let the capacity of a token bucket be 'C' tokens and tokens enter the bucket at a rate of 'r' tokens per second. The maximum num of packets that can enter the n/w during a time 't' is.



$$\text{max} = (C + t * r) \text{ packets}$$

$$\text{max rate} = \frac{C + tr}{t}$$

initial tokens in the bucket

Consider a token bucket of capacity 250 KB and tokens arrive at a rate of 2 MB/S. If the maximum output rate is 25 MB/S what is the burst time.

Gap $C = 250$ KB, $r = 2$ MBPS $T = ?$

$$\frac{C + Tr}{T} = 25 \text{ MBPS}$$

$$\Rightarrow T = 10.86 \text{ ms}.$$

TCP is covered

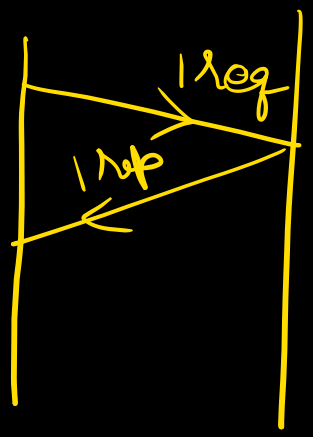
UDP – User datagram protocol:

TCP → heavy → lot of functionalities.

many application need simple TL · P

NTP → one req // reply.

NTP
- N/w Time
- Protocol



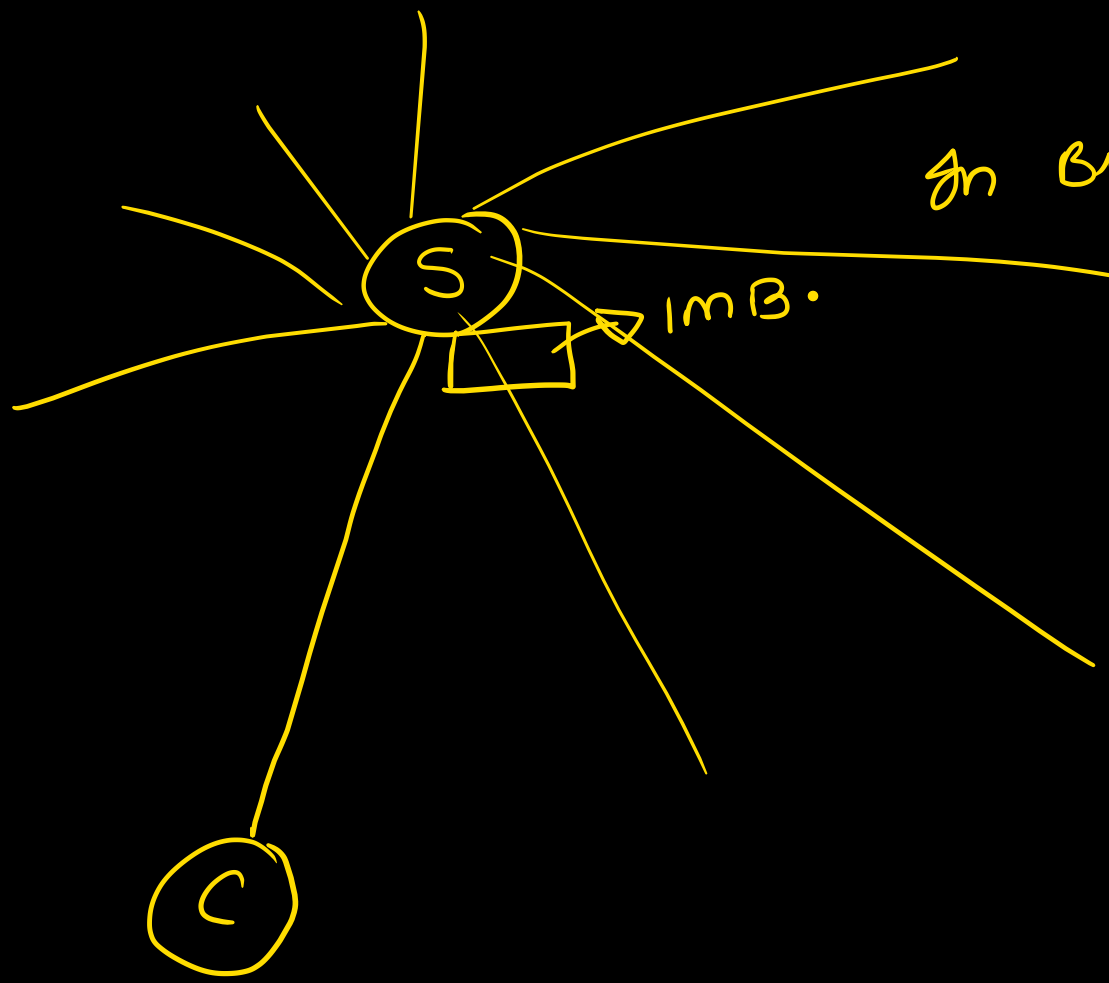
overhead.

1) NTP
2) BOOTP
   DHCP
3) NNTP
   |
   News
4) DNS
5) quote of the day.
.
.
.
.
.
.

1 Req / 1 reply.

→ TCP is a overhead

2) Broad Casting and multicasting → TCP is not suitable.



In Broadcasting
$\cong 2^{24}$ connection

$2^{24} \times 1MB \cong$ 16GB

Too many Resources are required

3) Some application require speed then reliability.

Ex: gaming.

multimedia ⇒ Youtube.

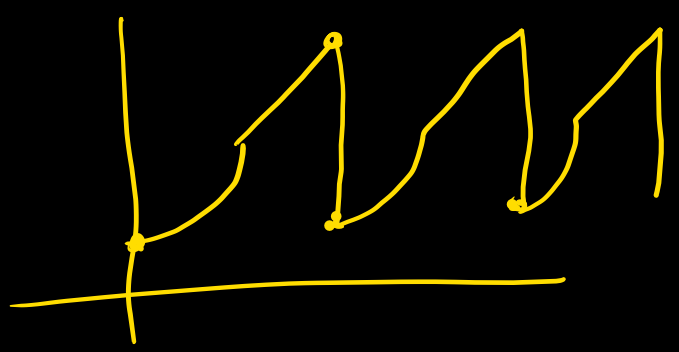
In TCP: we have AIMD algorithm

A I M D — Decrease.

Additive Increase multiplicative

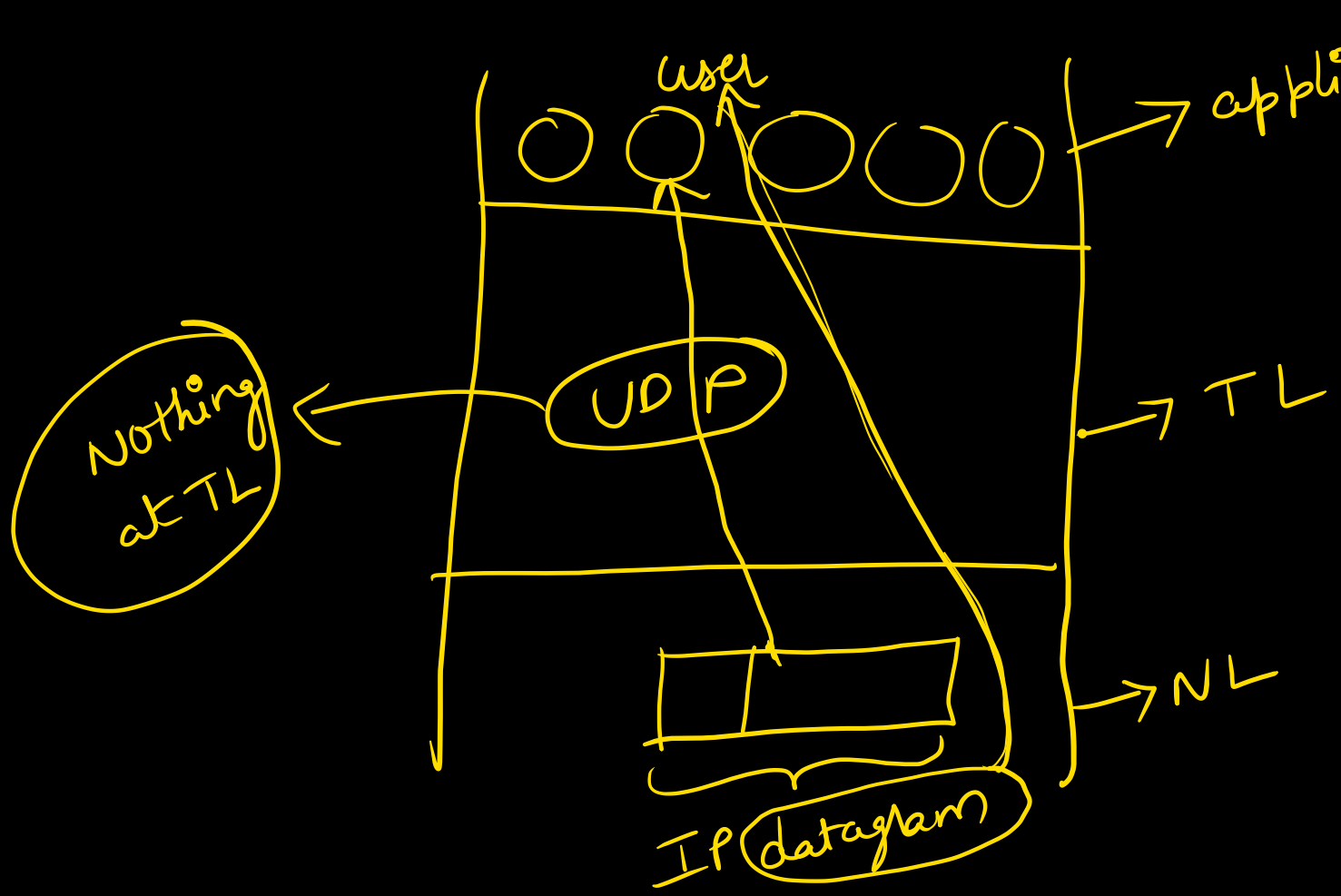
↓

Congestion control algo.



and data rate is not constant. It increases and decreases.

∴ TCP is not needed



Datagram → user

without doing anything at Transport layer in UDP.

NO connection establishment → NO SYN, Seq numbers

NO Acknowledgements → NO ACK, ack numbers

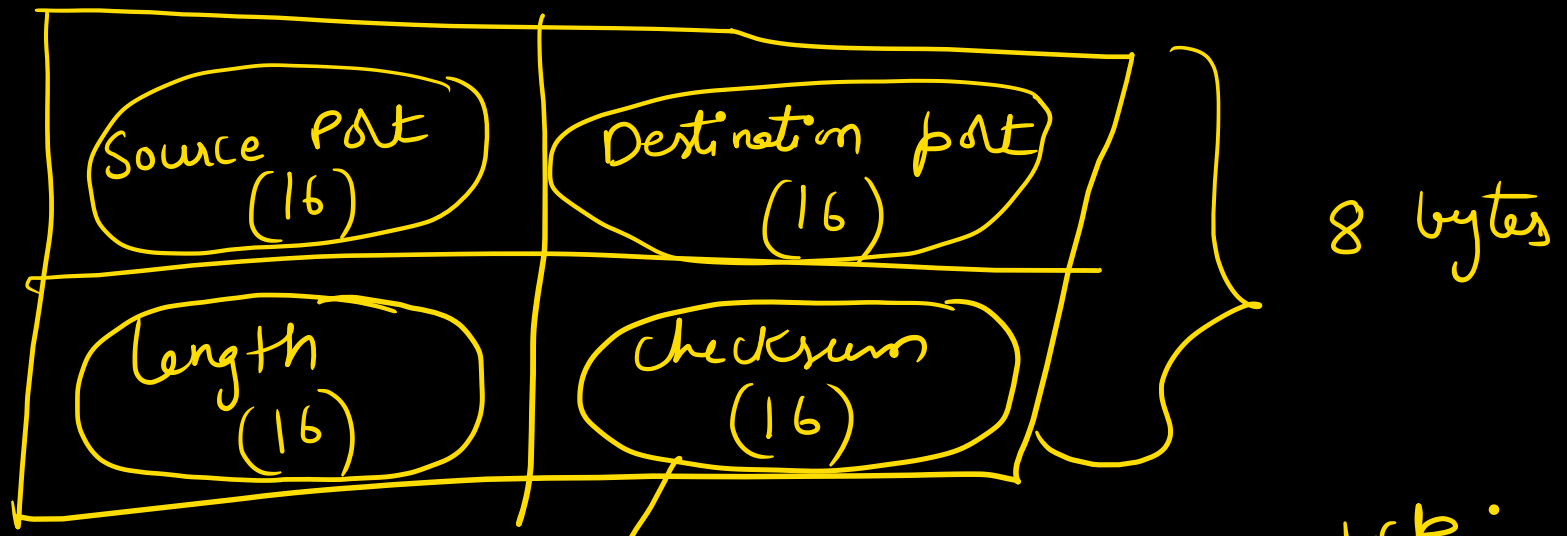
NO flow control → NO adv window

NO extra features → NO URG, RST, PSH....

NO options.

∴ UDP header is simple.

UDP header:



UDP Header
+
UDP data
+
Pseudo IP header

Same as tcp.

6th, 7th → CN will over

8th holiday → maha shivratri

9th → OS will start → 20

+

CN PYQ's