Date Day

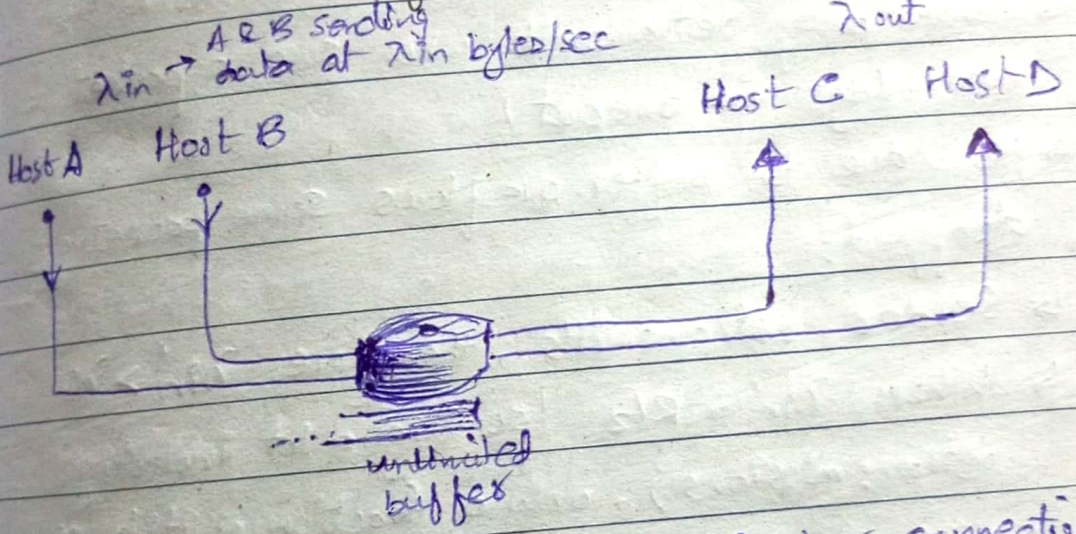
PRINCIPLES OF CONGESTION CONTROL

→ "Too many sources sending too much data too fast for network to handle" — congestion

→ Symptoms of congestion control
  ↳ long delays ↳ packet loss

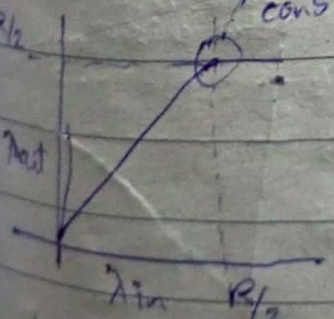
• Scenario #01: Two Senders, a Router with Infinite Buffer

λ_{in} → A & B sending data at λ_{in} bytes/sec

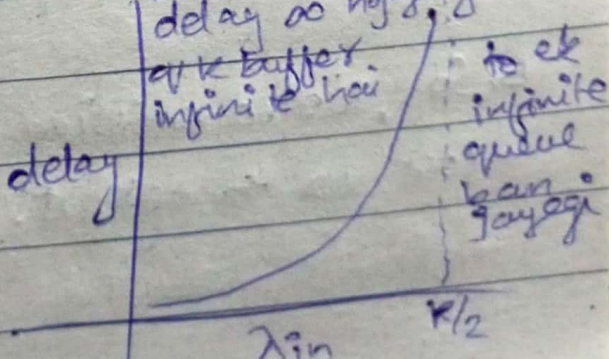


Agr link capacity R hai to ~~4~~ per connection R/2 hogi.

→ Maximum sender ki sending capacity R/2 hi hosakti hai.



agr rate to R/2 par send krne lagge to delay ao hojayega.
q ka buffer infinite hai



to ek infinite queue ban jayegi

Date _____ Day

- Scenario #02: Two Senders and a Router with Finite Buffers

→ Case #01:

↳ Sender knows when buffer is free and only sends then.

↳ No packet loss

↳ So throughput = $\lambda_{in} = \lambda_{in}'$ (original data + ~~retransmission~~)

↳ Max throughput/host = R/2

→ Case #02:

↳ Sender only retransmits when sure a packet is lost (eg: via timeout)

↳ Now due to buffer overflows some packets are dropped.

↳ Sender retransmits, therefore $\lambda_{in}' > \lambda_{in}$

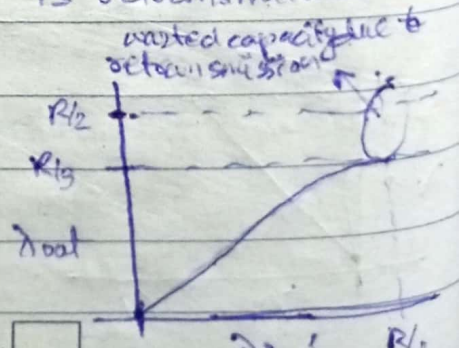
↳ When $\lambda_{in}' = R/2$ (sending a lot including retransmissions), only a fraction of it is new/original data

↳ Delivered throughput = R/3

↳ R/2 mein se R/3 sirf original data hai

↳ baaqi R/6 (0.166) is retransmitted

↳ So, retransmission cost us throughput.



Date Day

→ "sender must perform retransmission in order to compensate for dropped packets due to buffer overflow."

→ Case #03:

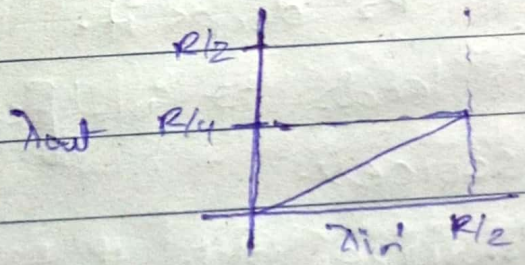
↳ Sender mistakenly thinks a packet is lost due to delay & retransmits it.

↳ But in reality original pkt is still in queue.

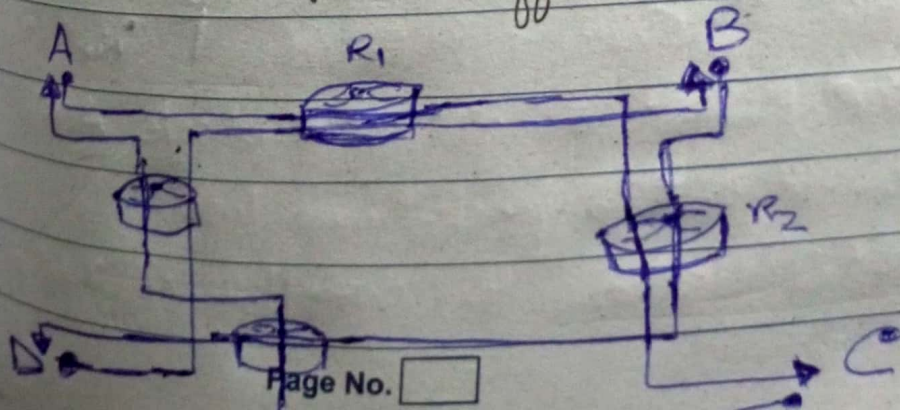
↳ The router wastes link bandwidth forwarding duplicate pkt.

↳ This bandwidth could have been used to send useful pkt.

↳ When each pkt is forwarded twice on avg: throughput becms R/4 when load is R/2



• Scenario #03: Four Senders and a Router with Finite Buffer & Multihop Paths



Page No.

Date _____ Day

↳ High Traffic :

→ Consider A → C path : passes thru $R_1 \rightarrow R_2$

→ Also, D → B shares R_1, and B → D shares

→ As λ_{in} increases for all.
R_2 ka buffer B-D connection se phele hi fill hojayega q k B ka 1st router R_2 hai jabke A k liye R_2 2nd router hai.

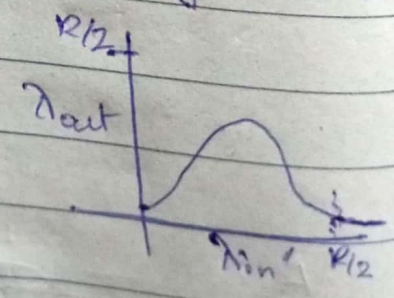
→ A-C pkts arriving from R_1 are frequently dropped at R_2.

→ Eventually AC ka throughput drops to 0, despite sending more data.

→ Packets dropped at 2nd hop router waste all upstream efforts.
↳ work done at R_1 to send AC pakts is wast if R_2 drops it.

→ This leads to inefficient use of network resources.

→ Prioritize pkts that have already traversed hops.



Date Day

- Costs of Congestion

① Throughput can never exceed capacity.
② Delay increases as capacity approached
③ Loss / retransmission decreases effective throughput
④ un-needed duplicates further decreases effective throughput
⑤ upstream transmission capacity / buffering wasted for pkt lost downstream.

- Approaches Towards Congestion Control

① End-To-End Congestion Control
↳ Network layer congestion k baare mein kuch nhi btati
↳ TCP ko khud infer krna parega k network mein congestion hai ya nhi. By the help of packet loss & delay.

② Network assisted Congestion Control
↳ Routers (Network layer) provide explicit feedback to sender & rec about congestion.
↳ May indicated congestion level or explicitly set sending rate

Date _____ Day

• TCP CONGESTION CONTROL :

→ TCP controls the sending rate based on network congestion.

→ If path is clear, it sends faster ; if it's congested it slows down.

Q: How TCP limits Sending Rate

→ TCP uses a variable called the congestion window (cwnd)

→ The sender can't have more than "cwnd" bytes of unack data in the network.

LastByte Sent − LastByte Acked ≤ min(cwnd, rwnd)
↓ ignore this

→ TCP sends "cwnd" bytes per RTT

→ Send rate ≈ cwnd / RTT

→ Cwnd directly affects how fast TCP can send data

→ congestion control is about dynamically adjusting cwnd based on perceived network conditions

Q: How TCP detects congestion?

1. Loss Event = Congested signal
   - If TCP timeout or get 3 duplicate ACKs, it assumes that a packet was lost
   - Packet is lost due to overflow of routers buff

Date Day

- Buffer is overflowed when there is network congestion.
- On packet loss → Decrease Rate
  - ↳ Loss means congestion
  - ↳ TCP [reduces] cwnd to reduce sending rate

2. ACKs = Network is healthy
   - If ACKs are arriving steadily.
   - TCP assumes no congestion & gradually incr. its send rate
   - TCP [increases] cwnd, probing for how much bandwidth it can safely use.

→ Self-Clocking Mechanism
- TCP uses ACKs to trigger data sending
- More/Faster ACKs = more data sent
- This automatically adjust the rate based on network feedback.

→ Bandwidth Probing Strategy
- TCP keeps increasing rate (cwnd) until pkt loss.
- Then it backs off, then slowly incr. again

Date _____ Day _____

▷ TCP Congestion Control Algorithm

The algorithm has 3 major component

1. Slow Start
2. Congestion Avoidance
3. Fast recovery

{ mandatory components of TCP, differing in how they incr size of cwnd in response to recd ACKs.

(1) Slow Start

→ Slow start ~~alg~~ increases the sending rate exponentially.

1. Initial Setup
   - cwnd starts at 1 MSS (Max seg size)
   - Only 1 segment is sent initially.
2. ACKs Double cwnd
   - Har ek ACK receive hone par 1 seg increase hoga.
   - ~~Isse~~ Jiski waja se exponential growth aayega
   - 1 → 2 → 4 → 8 → ... per RTT.

3. Kahen does Slow start end?
   - If timeout
     - ↳ set cwnd = 1
     - ↳ set ssthresh = cwnd/2 ← slow start threshold
     - ↳ Start slowstart again
   - If cwnd ≥ ssthresh
     - ↳ Switch to Congestion Avoidance (grows linearly)

Page No. ☐

Date Day

- 3 duplicate ACKs
  - Enter Fast Recovery instead of restarting

② Congestion Avoidance
- It starts when the slow start ends typically when cwnd ≥ ssthresh
- it takes more cautious, linear approach to avoid congestion
- How cwnd grows?
  - cwnd increases 1 MSS per RTT (yaha "per RTT" hota hai or slow start mein "per ACK")
  - cwnd increases by MSS/cwnd per sec
    - MSS = 1,460 bytes, cwnd = 14,600 bytes
    - No. seg = 10 (MSS/cwnd)
    - Each Ack increase cwnd by 1/10 MSS
- When Congestion is detected
  - Timeout:
    - set cwnd = 1 & ssthresh = cwnd/2
    - start slow start again
  - 3 Dup ACKs:
    - ssthresh = cwnd/2
    - cwnd = ssthresh + 3* MSS
    - Enter Fast Recovery

3 dup par fast recovery isliye kr rhe q k 3 dup ACKs ka matlab k kuch segments rec hogai hai sirf ek nhi hoga. MTlab network completely congested nhi. To aggressively Page No. kme ka need nhi, jese slow start mein hota hai.

Date _____ Day

(3) Fast Recovery

↳ It is triggered on 3 dup ACKs.

↳ Action :

- Retransmitt the lost segment immediately
- Set ssthresh = cwnd/2 (plus 3 dup segments)
- set cwnd = ssthresh + 3*MSS
- For each additional duplicate ACK, increase cwnd by 1 MSS.

3. dup ACK ke ilawa agr ACK aaye.

→ Exit Condition

↳ When ACK for lost seg arrives → enter congestion avoidance and reset cwnd = ssthresh

→ When timeout → go to slowstart (cwnd = 1 MSS)

→ Earlier version of TCP, known as TCP Tahoe unconditionally cuts cwnd = 1 MSS on timeout or 3 dup ACKs.

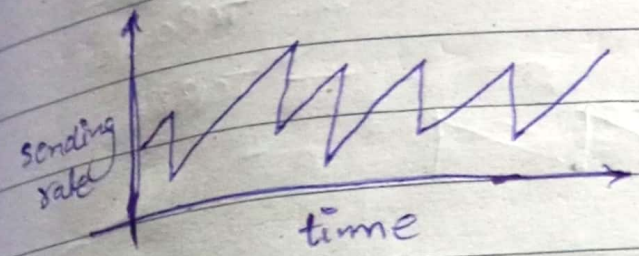
→ The newer version TCP Reno incorporated fast recovery.

Date Day

→ TCP congestion control is often referred to as an "additive-increase", "multiplicative-decrease".

↓ incr sending rate by 1 MSS / RTT until loss detected

↓ cut sending rate in half at each loss event



• TCP Cubic :

→ TCP Reno's additive-increase & multiplicative decrease is too cautious after plet loss

→ Slow growth after cutting the cwnd may underutilize bandwidth.

→ CUBIC fixes this by faster recovering to the pre loss sending rate & then probing carefully.

→ How it works?

W_{max} = cwnd size when last loss occured

K = Time needed (from loss) to reach W_{max} again

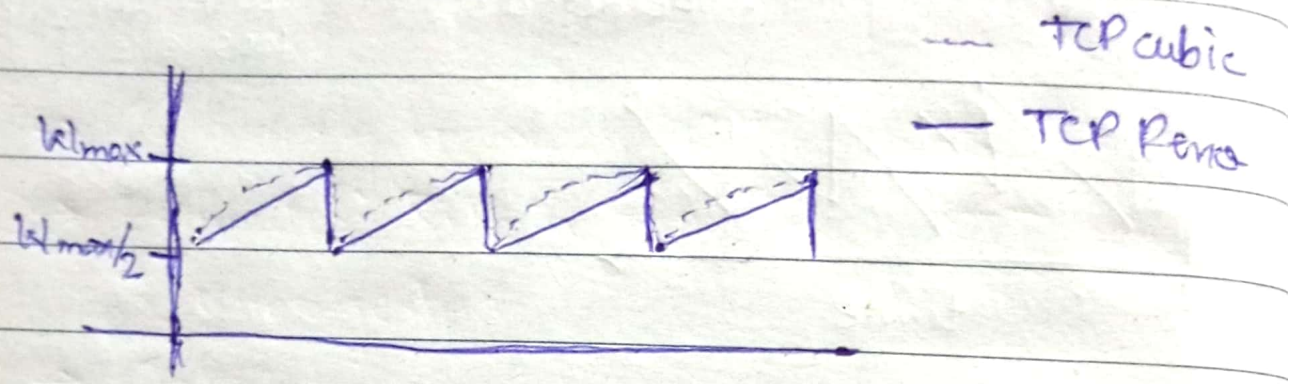
t = Current time

Increase W as a function of the cube of the distance b/w current time & K

Date _____ Day _____

↳ larger increases when farther away from K

↳ smaller increases (cautious) when near to K



(round)

→ avg throughput of a connection = $\frac{0.75 \cdot W}{RTT}$ → ?

Date Day

Network Assisted Congestion Control

① Explicit Congestion Notification

→ It allows routers to signal congestion without dropping pkts.

→ How ECN Works?

↳ IP Layer (Network Layer)

- Uses 2 bits in the IP Header's Type of Service (ToS) field.
- Router mark pkts instead of dropping them congestion is detected
- Marked pkts are forwarded to receiver.

↳ TCP Layer (Transport Layer)

- Receiver detects ECN marking and sets the ECE (ECN Echo) bit in the ACK
- Sender, upon rec ECE, reduces congestion window to half.
- Sender then sets the CWR (Congestion Window Reduced) bit in the next outgoing segment.

→ Proactive cong. control = Acts before pkt loss occur

→ Reduces retransmission

Date _____ Day _____

(2) Delay Based Congestion Control

→ Proactively detects congestion by monitoring increasing delay (RTT) rather than waiting for pkt loss

- TCP Vegas Approach

→ Continously measures RTT of pkts

→ Estimates RTTmin = minimum RTT when path is uncongested

→ Calculates:

Expected rate = cwnd / RTTmin

Actual rate = measured from ACKs (# bytes sent in last RTT / RTT measured)

→ if Actual < Expected → congestion detected
→ Decrease cwnd

→ if Actual ≥ Expected → Network uncongested
→ Increase cwnd

▷ TCP Fairness

→ If K TCP connection share a bottleneck link of rate R, fairness means each connection should ideally get R/K bandwidth.

→ Assumptions:
↳ Same MSS an RTT for both connections
↳ Operating in Cong Avoidance mode
↳ Both conn use AIMD

Date Day

→ Throughput Behaviour
↳ Each TCP connection incr cwnd by 1 MSS / RTT
↳ Throughput of both increases together until loss occur
↳ On pkt loss, both halve their cwnd
↳ Throughput drop
↳ This cycles repeat, causing throughput to fluctuate near fairness line
↳ Both will share equal bandwidth

→ Fairness can be voilated if
↳ Connections have diff RTTs
↳ Shorter RTTs = faster incr = more bandwidth
↳ Presence of UDP traffic which doesn't reduce rate on congestion.

↳ Fairness in UDP
→ Multimedia apps use UDP bcz they don't want rate throttled by congestion control
→ Send pkts at constant rate
→ Can tolerate pkt loss
→ It is possible for UDP sources to crowd out TCP traffic

Page No.