Transport Layer Protocols -4

A/PROF. DUY NGO

Learning Objectives

3.6 principles of congestion control

3.7 TCP congestion control

Principles of Congestion Control

congestion:

informally: "too many sources sending too much data too fast for **network** to handle"

different from flow control!

manifestations:

- lost packets (buffer overflow at routers)
- long delays (queueing in router buffers)

a top-10 problem!

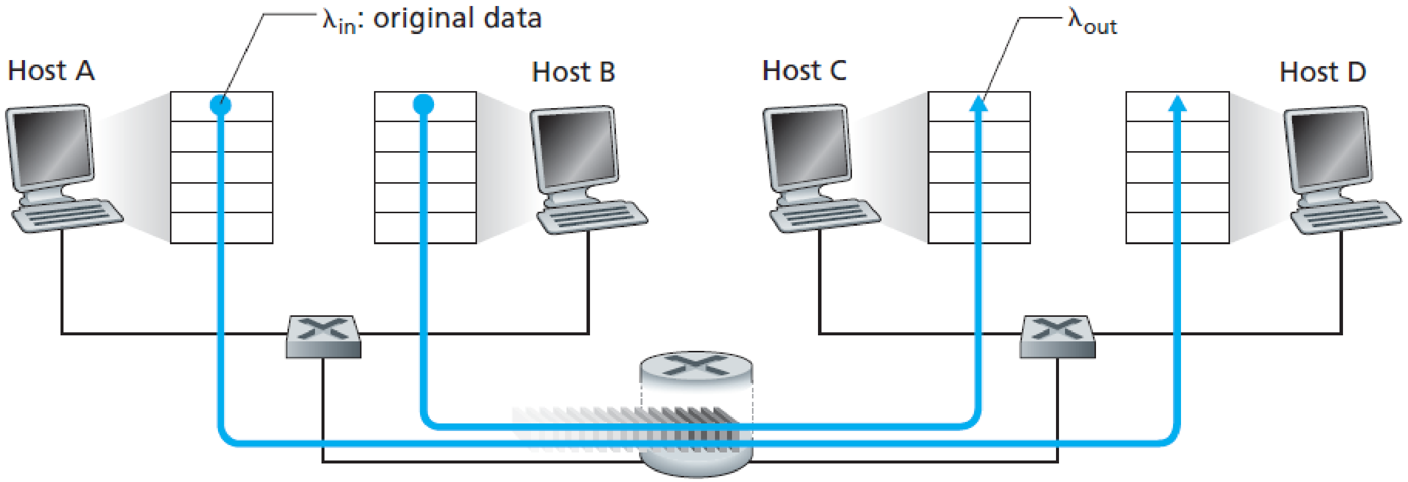
Causes/Costs of Congestion: Scenario 1 (1 of 2)

two senders, two receivers

one router, infinite buffers

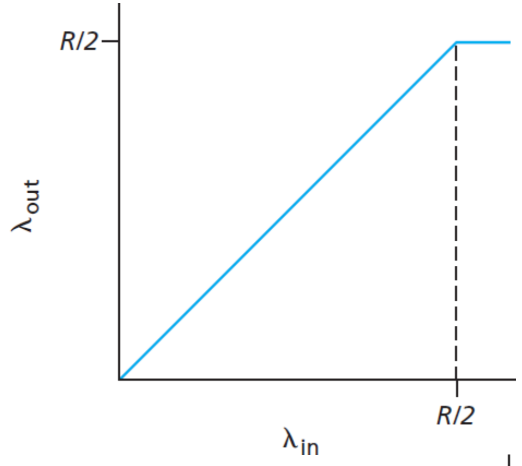
output link capacity: R

no retransmission

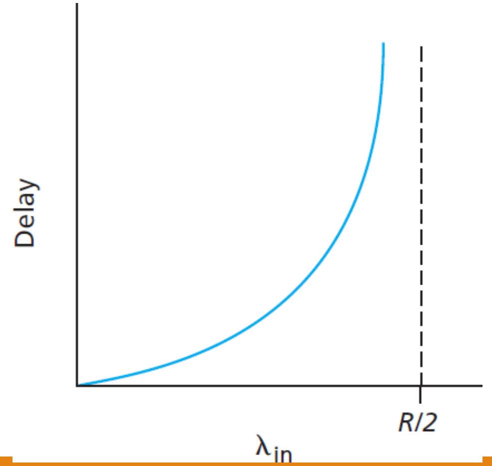


Causes/Costs of Congestion: Scenario 1 (2 of 2)

- maximum per-connection throughput: $\frac{R}{2}$



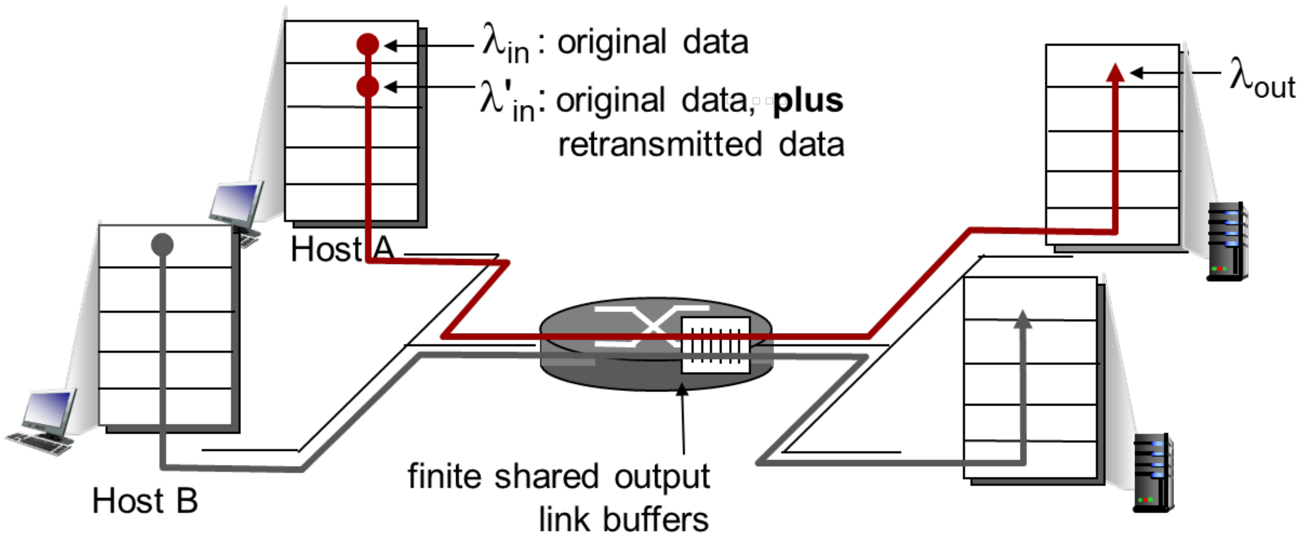
- large delays as arrival rate, λ_{in}, approaches capacity



Causes/Costs of Congestion: Scenario 2 (1 of 6)

- one router, **finite** buffers
- sender retransmission of timed-out packet
  - application-layer input = application-layer output:
  - transport-layer input includes **retransmissions** :

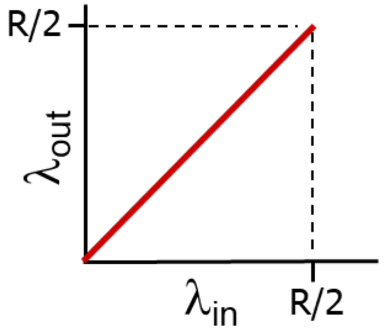
$$\lambda_{in} = \lambda_{out} \quad \lambda'_{in} \geq \lambda_{in}$$

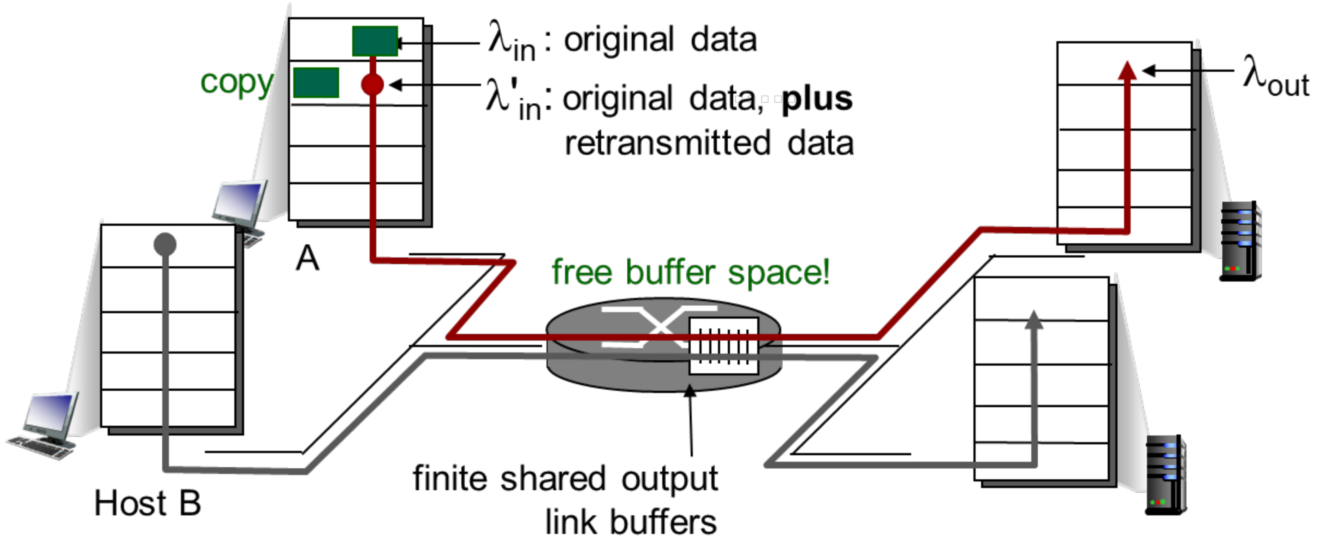


Causes/Costs of Congestion: Scenario 2 (2 of 6)

idealization: perfect knowledge

- sender sends only when router buffers available

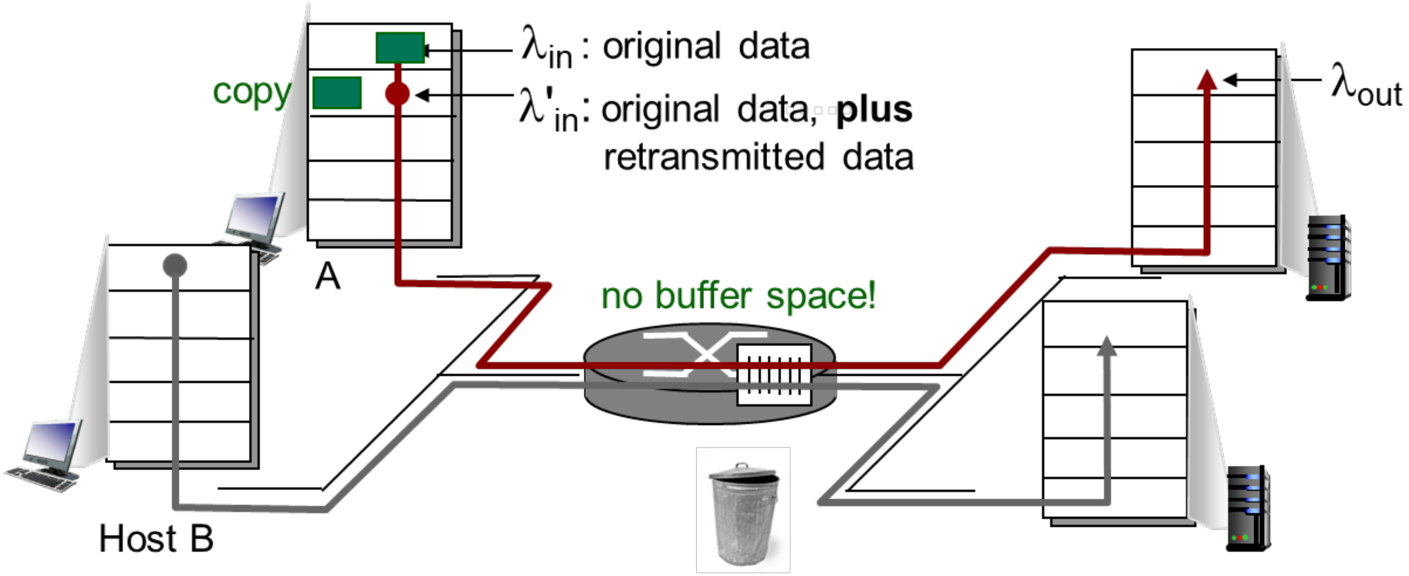




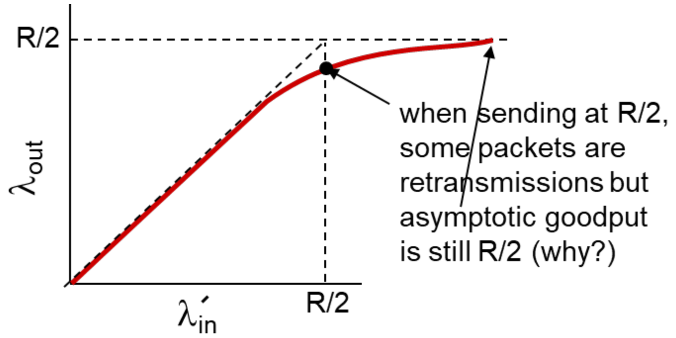
Causes/Costs of Congestion: Scenario 2 (3 of 6)

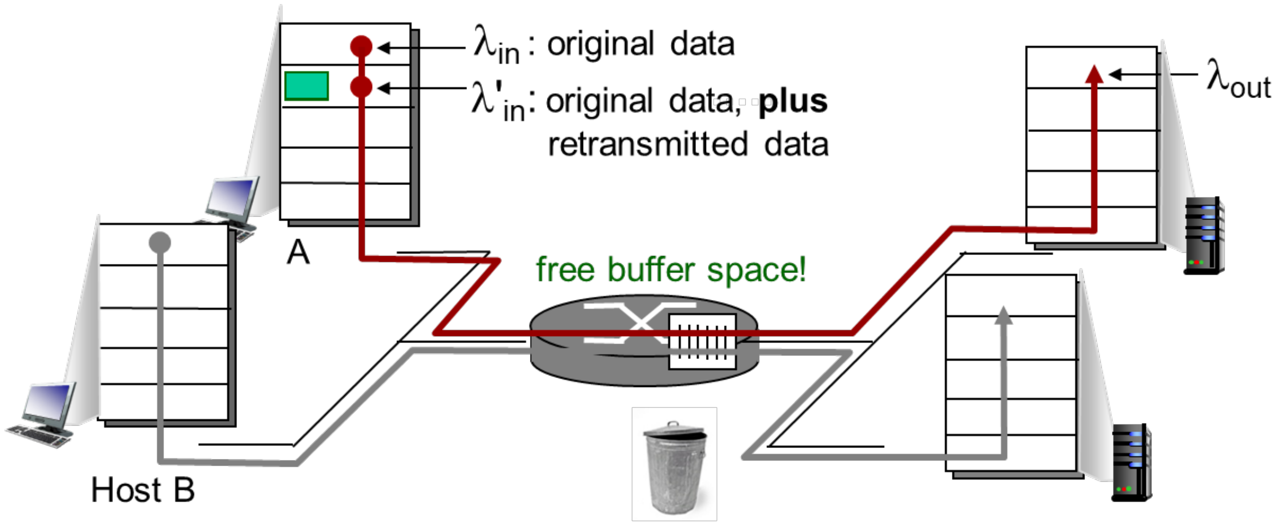
Idealization: known loss packets can be lost, dropped at router due to full buffers

sender only resends if packet **known** to be lost



Causes/Costs of Congestion: Scenario 2 (4 of 6)



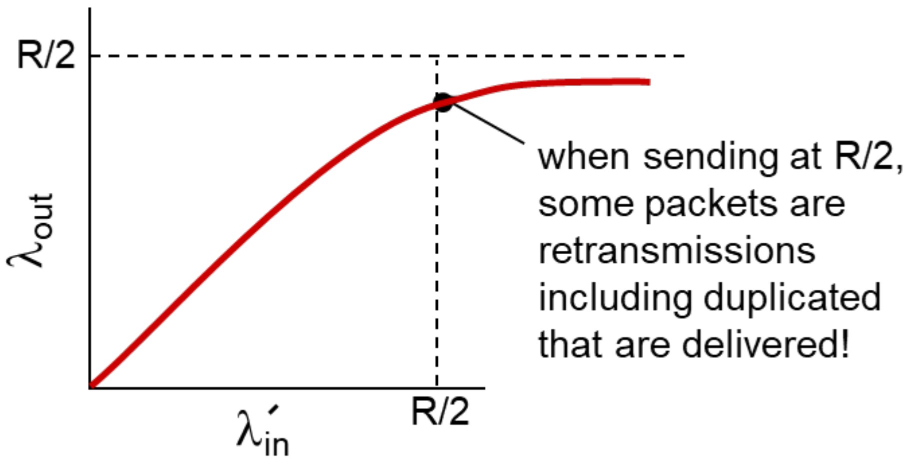


Causes/Costs of Congestion: Scenario 2 (5 of 6)

Realistic: duplicates

packets can be lost, dropped at router due to full buffers

sender times out prematurely, sending **two** copies, both of which are delivered



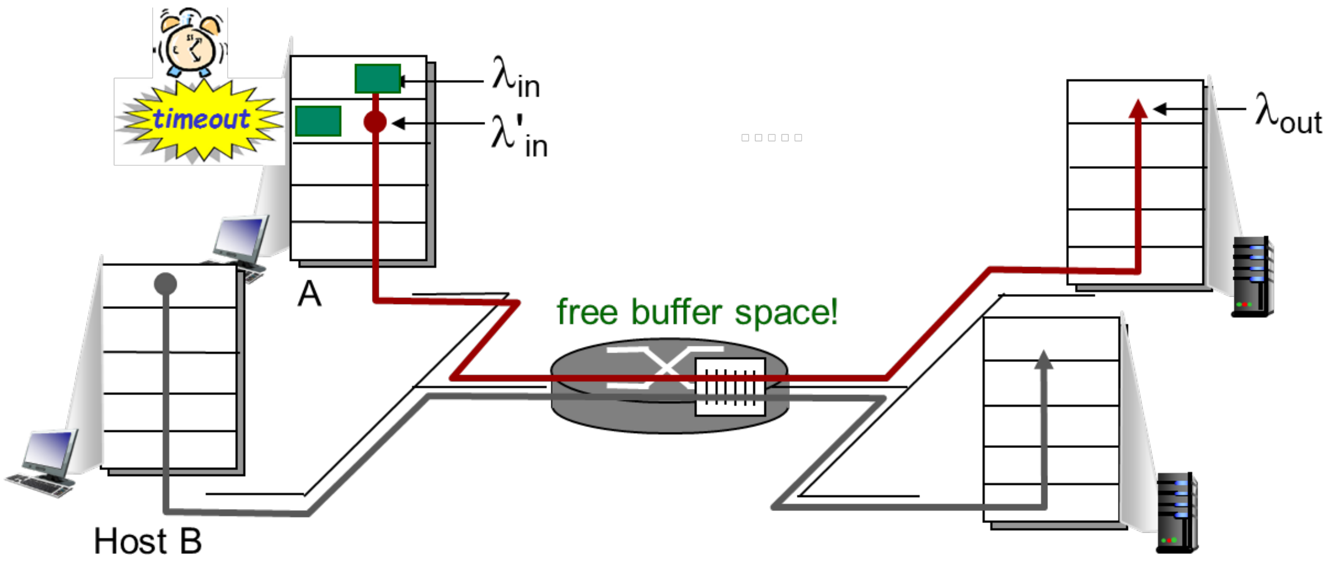
Causes/Costs of Congestion: Scenario 2 (6 of 6)

"costs" of congestion:

more work (retrans) for given "goodput"

unneeded retransmissions: link carries multiple copies of pkt

- decreasing goodput

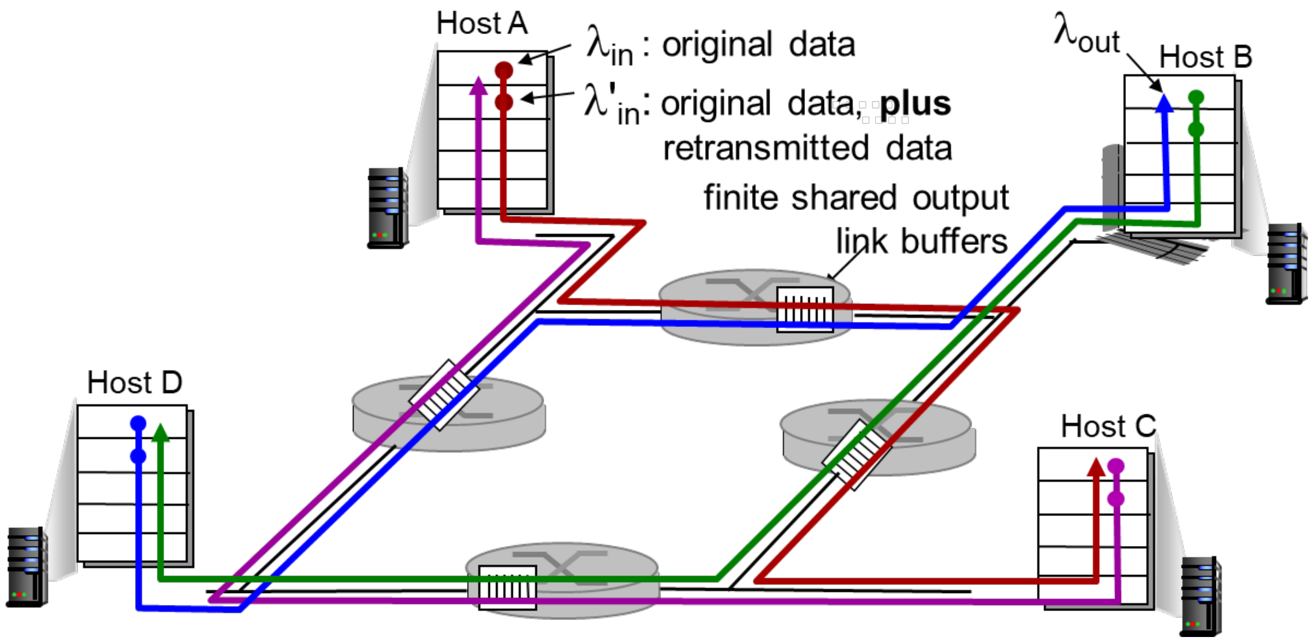


Causes/Costs of Congestion: Scenario 3 (1 of 2)

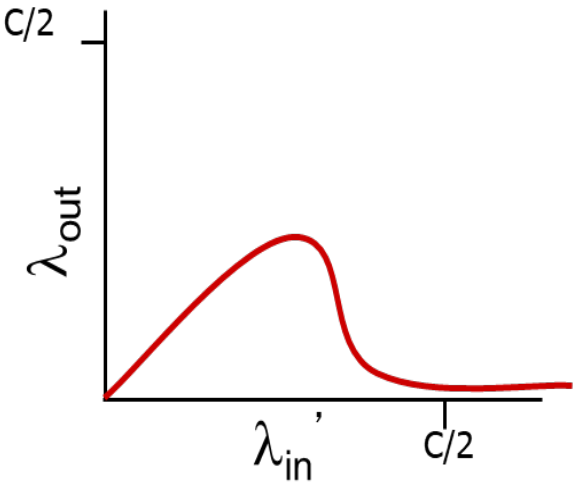
- four senders
- multihop paths
- timeout/retransmit

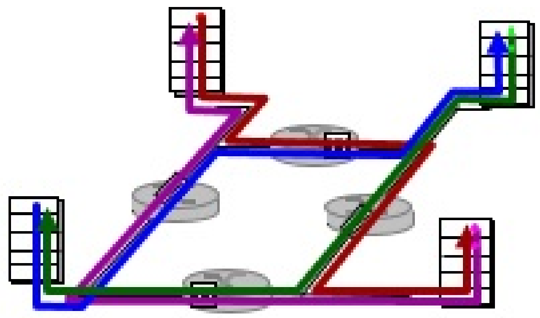
Q: what happens as λ_{in} and λ_{in}' increase **?**

A: as red λ_{in}' increases, all arriving blue pkts at upper queue are dropped, blue throughput $\rightarrow 0$



Causes/Costs of Congestion: Scenario 3 (2 of 2)





another "cost" of congestion:

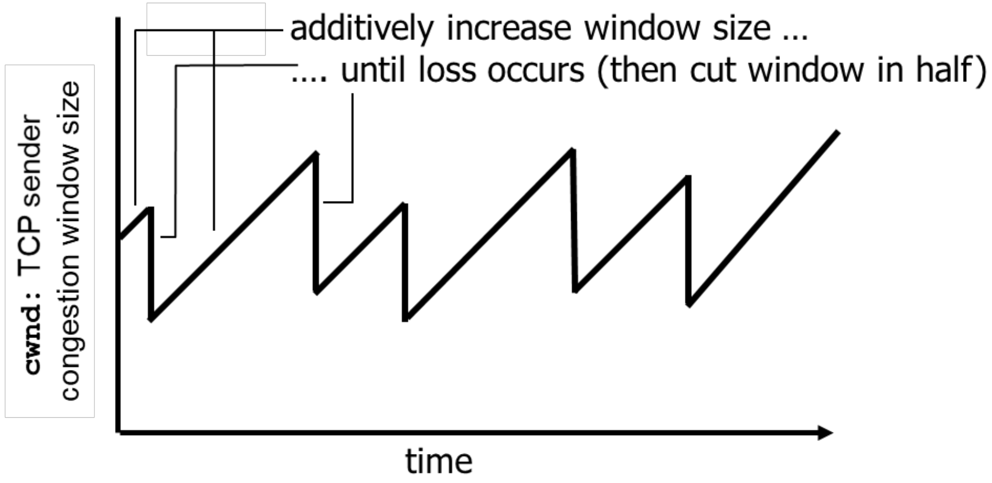
when packet dropped, any "upstream transmission capacity used for that packet was wasted!

TCP Congestion Control: Additive Increase Multiplicative Decrease

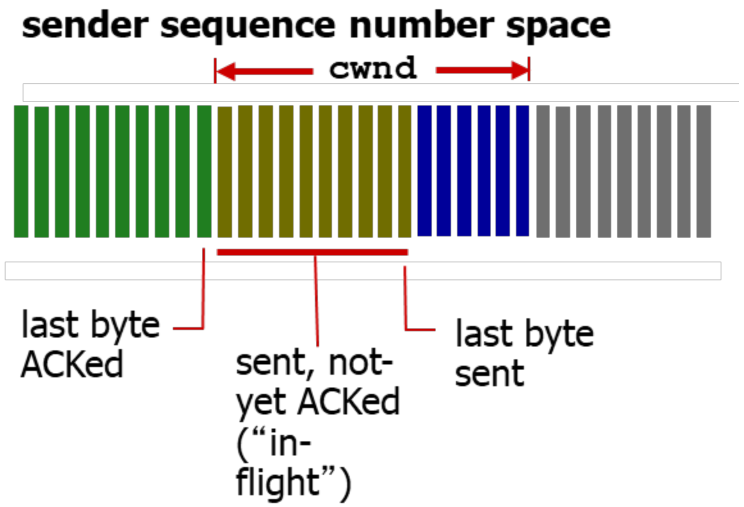
approach: sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs

- **additive increase**: increase **`cwnd`** by 1 MSS every RTT until loss detected
- **multiplicative decrease**: cut **`cwnd`** in half after loss

AIMD saw tooth behavior: probing for bandwidth



TCP Congestion Control: Details



- sender limits transmission:

$$\boxed{\texttt{LastByteSent-LastByteAcked} \leq \texttt{cwnd}}$$

- **cwnd** is dynamic, function of perceived network congestion

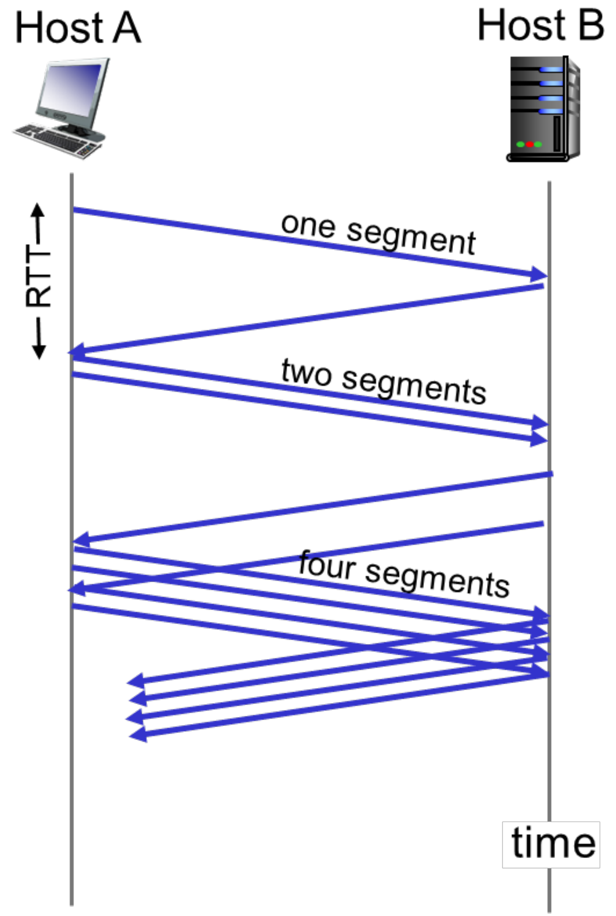
TCP sending rate:

- **roughly:** send cwnd bytes, wait RTT for ACKS, then send more bytes

$$\boxed{\text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}}$$

TCP Slow Start

- when connection begins, increase rate exponentially until first loss event:
  - initially **`cwnd`** = 1 MSS
  - double **`cwnd`** every RTT
  - done by incrementing **`cwnd`** for every ACK received
- **summary:** initial rate is slow but ramps up exponentially fast



TCP: Detecting, Reacting to Loss

loss indicated by **timeout**: **TCP RENO**

- **`cwnd`** set to 1 MSS;
- window then grows exponentially (as in Slow Start) to threshold, then grows linearly

loss indicated by **3 duplicate ACKs**: **TCP RENO**

- dup ACKs indicate network capable of delivering some segments
- **`cwnd`** is cut in half window plus 3 (for Fast Recovery), then grows linearly

TCP Tahoe <u>always</u> sets **`cwnd`** to 1 (for both timeout and 3 duplicate acks) and enter Slow Start

TCP: Switching from Slow Start to CA

Q: when should the exponential increase switch to linear?

A: when **`cwnd`** gets to $\frac{1}{2}$ of its value before timeout.

Implementation:

- variable **`ssthresh`**
- on loss event, **`ssthresh`** is set to $\frac{1}{2}$ of **`cwnd`** just before loss event

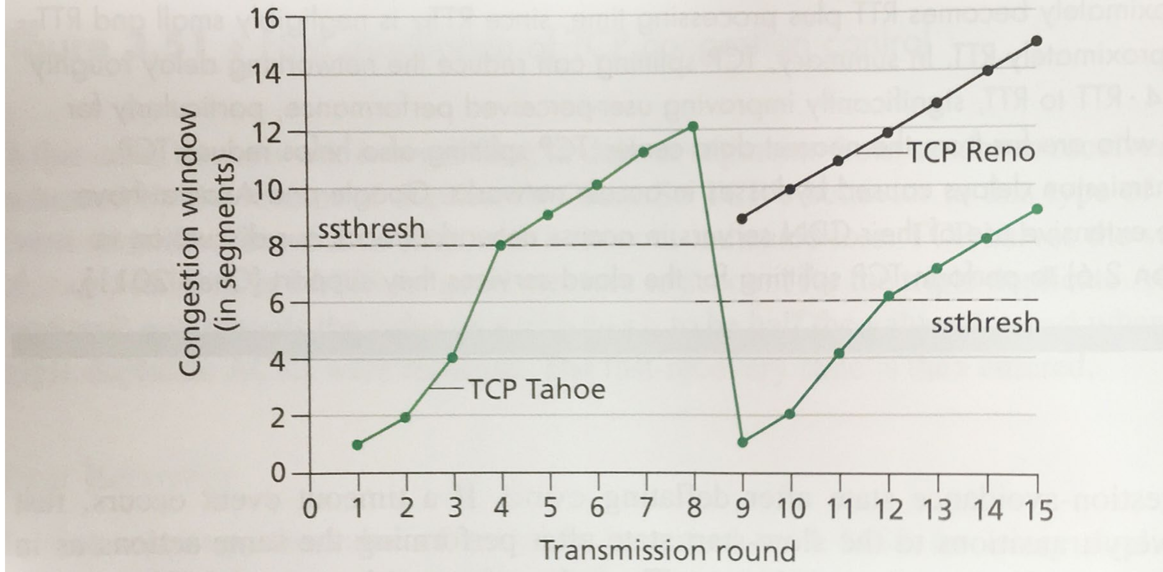


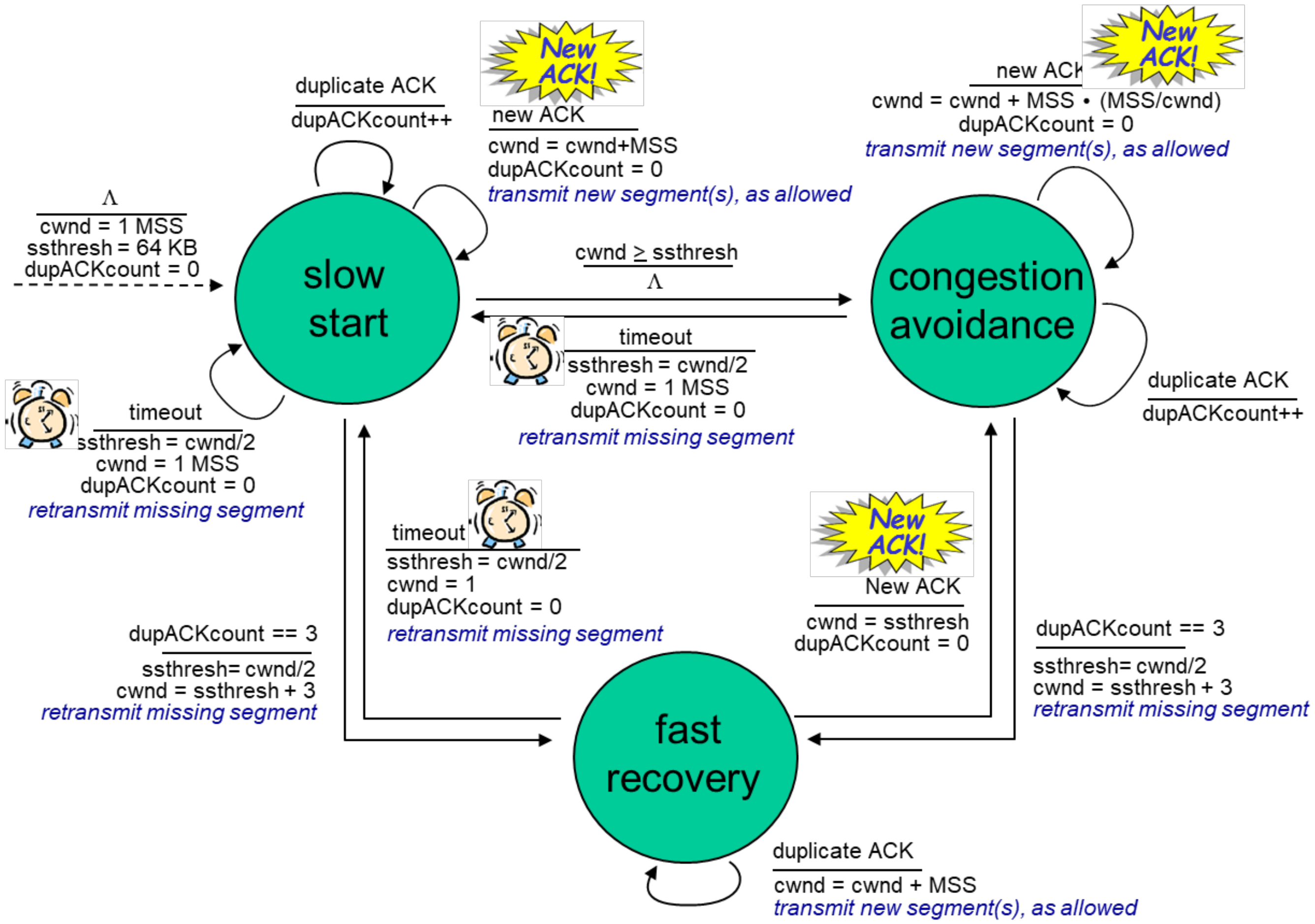
Figure 3.52 ♦ Evolution of TCP's congestion window (Tahoe and Reno)

In Fig 3.52, at transmission round 9, triple duplicate ACKs occur

- ssthresh := cwnd/2 = 12/2 = 6 MSS
- **Reno**: cwnd := ssthresh + 3 = 6 + 3 = 9 MSS
- **Tahoe**: cwnd := 1 MSS

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

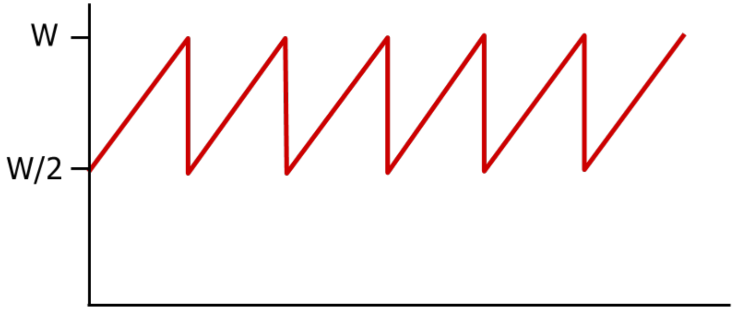
Summary: TCP Congestion Control



TCP Throughput

- average TCP thruput as function of window size, RTT?
  - ignore slow start, assume always data to send
- W: window size (measured in bytes) where loss occurs
  - average window size (# in-flight bytes) is $\frac{3}{4}W$
  - average thruput is $\frac{3}{4}W$ per RTT

$$\text{avg TCP thruput} = \frac{3}{4}\frac{W}{RTT} \text{ bytes/sec}$$



TCP Futures: TCP over "Long, Fat Pipes"

- example: 1500 byte segments, 100ms RTT, want 10 Gbps throughput
- requires W = 83,333 in-flight segments
- throughput in terms of segment loss probability, L [Mathis 1997]:

$$\text{TCP throughput} = \frac{1.22 \cdot \text{MSS}}{\text{RTT}\sqrt{L}}$$

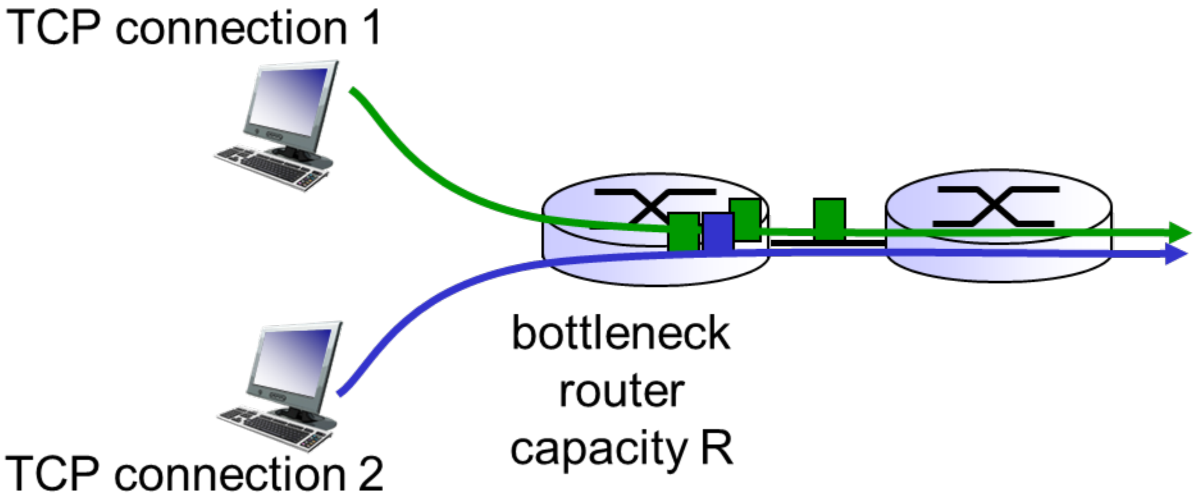
→ to achieve 10 Gbps throughput, need a loss rate of

$L = 2 \cdot 10^{-10}$ **– a very small loss rate!**

- new versions of TCP for high-speed

TCP Fairness

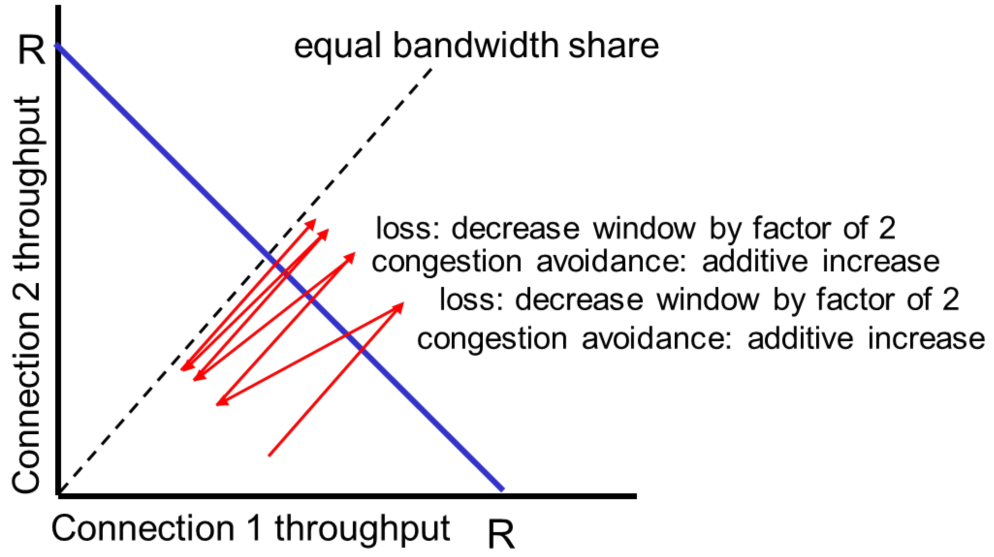
fairness goal: if K TCP sessions share same bottleneck link of bandwidth R, each should have average rate of $\frac{R}{K}$



Why is TCP Fair?

two competing sessions:

- additive increase gives slope of 1, as throughout increases
- multiplicative decrease decreases throughput proportionally



Fairness (More)

Fairness and UDP

- multimedia apps often do not use TCP
  - do not want rate throttled by congestion control
- instead use UDP:
  - send audio/video at constant rate, tolerate packet loss

Fairness, parallel TCP connections

- application can open multiple parallel connections between two hosts
- web browsers do this
- example, link of rate R with 9 existing connections:
  - new app asks for 1 TCP, gets rate $\frac{R}{10}$
  - new app asks for 11 TCPs, gets $\frac{R}{2}$

Explicit Congestion Notification (ECN)

network-assisted congestion control:

- two bits in IP header (ToS field) marked **by network router** to indicate congestion
- congestion indication carried to receiving host
- receiver (seeing congestion indication in IP datagram)) sets ECE bit on receiver-to-sender ACK segment to notify sender of congestion