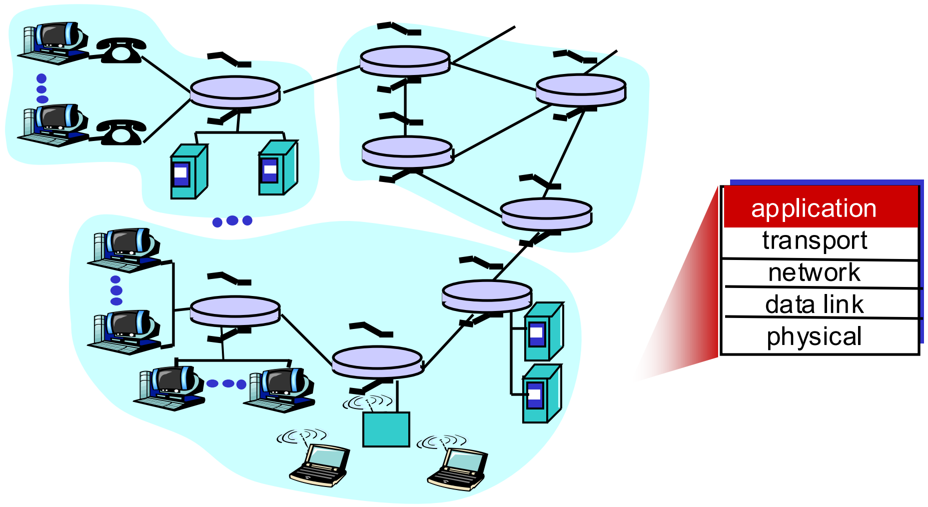
CS 4390
Computer Networks

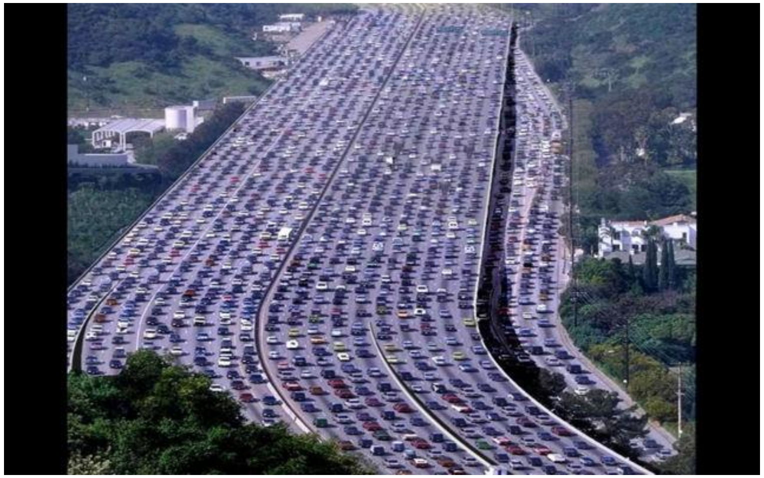


Transport Layer – Congestion Control Introduction

Principles of Congestion Control

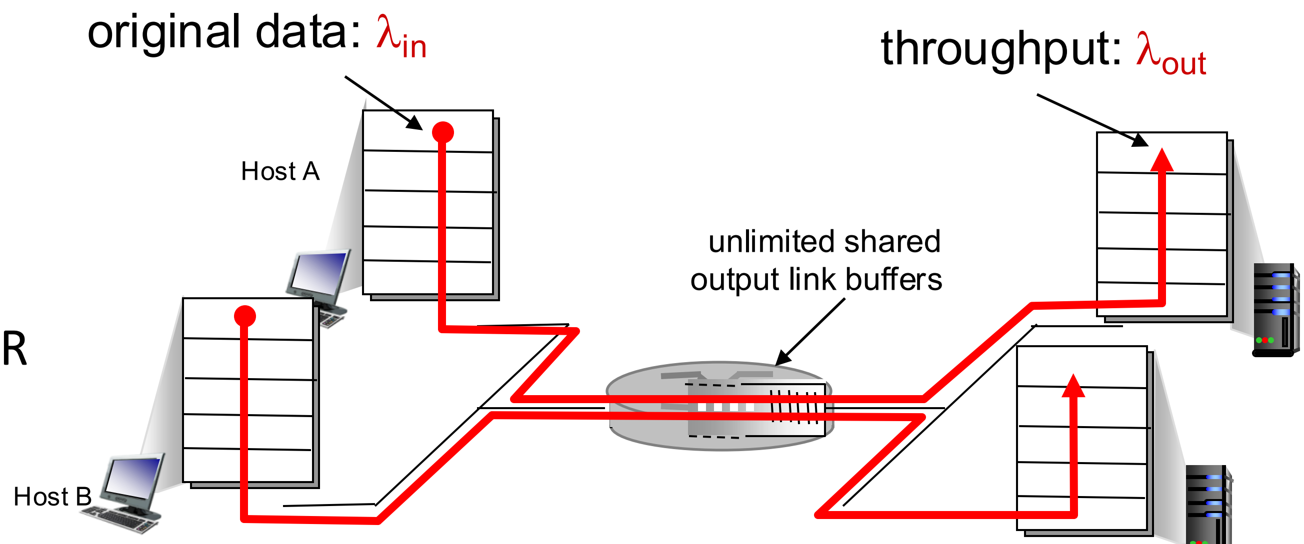
congestion:

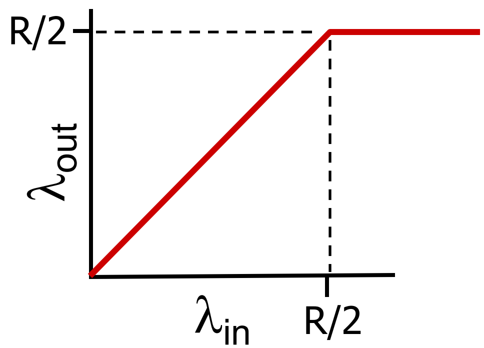
- informally: "too many sources sending too much data too fast for *network* to handle"
- *different from flow control!*
- manifestations:
  - lost packets (buffer overflow at routers)
  - long delays (queueing in router buffers)
- a top-ten problem!

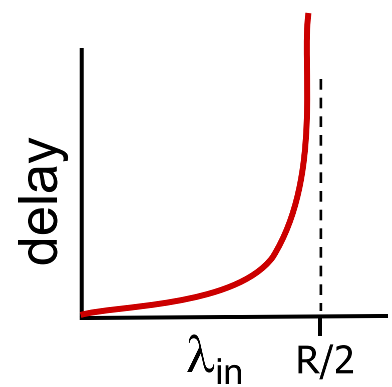


Causes/costs of Congestion: scenario 1

- two senders, two receivers
- one router, infinite buffers
- output link capacity: R
- no retransmission



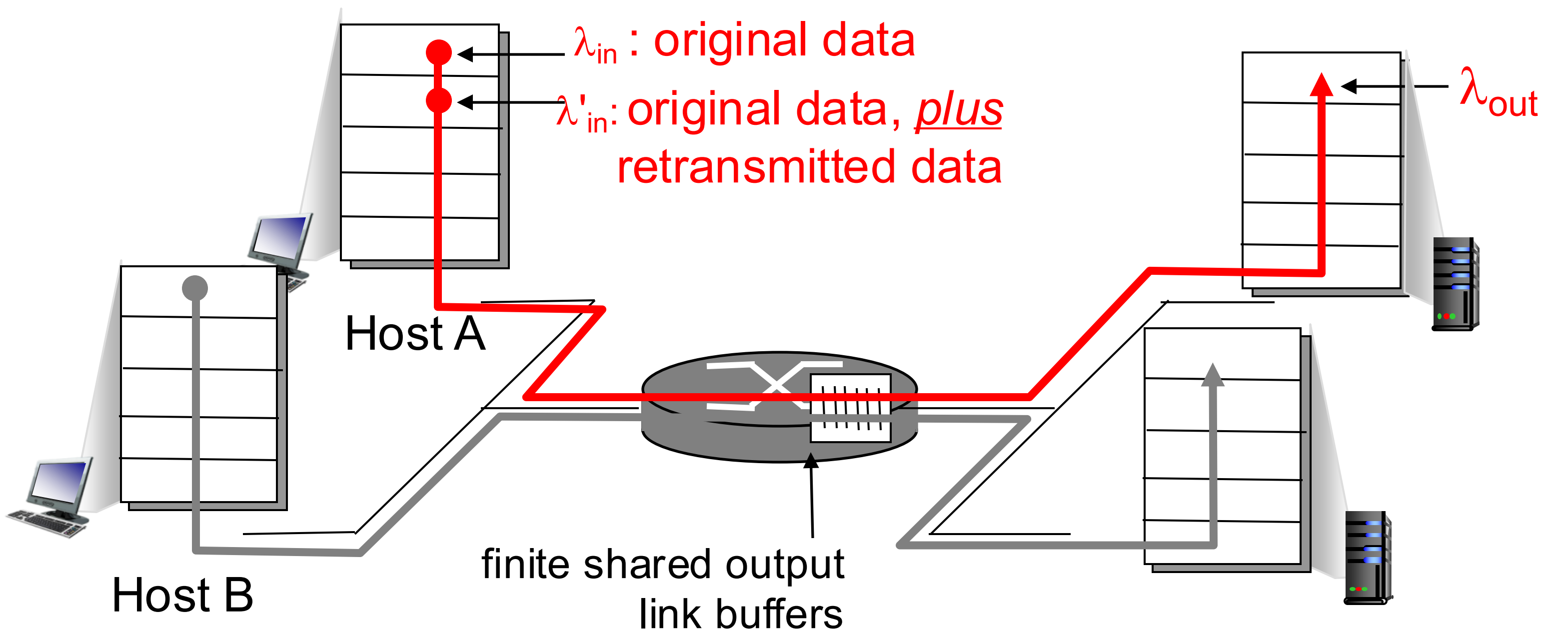




- maximum per-connection throughput: R/2
- large delays as arrival rate, λ_{in}, approaches capacity

Causes/costs of Congestion: scenario 2

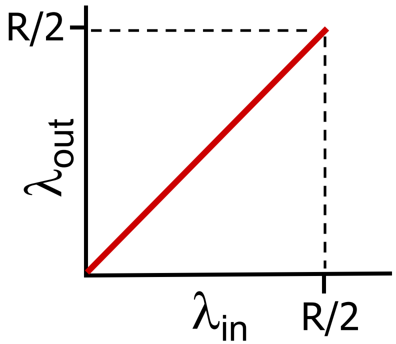
- one router, *finite* buffers
- sender retransmission of timed-out packet
  - application-layer input = application-layer output: $\lambda_{in} = \lambda_{out}$
  - transport-layer input includes *retransmissions* : $\lambda'_{in} \geq \lambda_{in}$

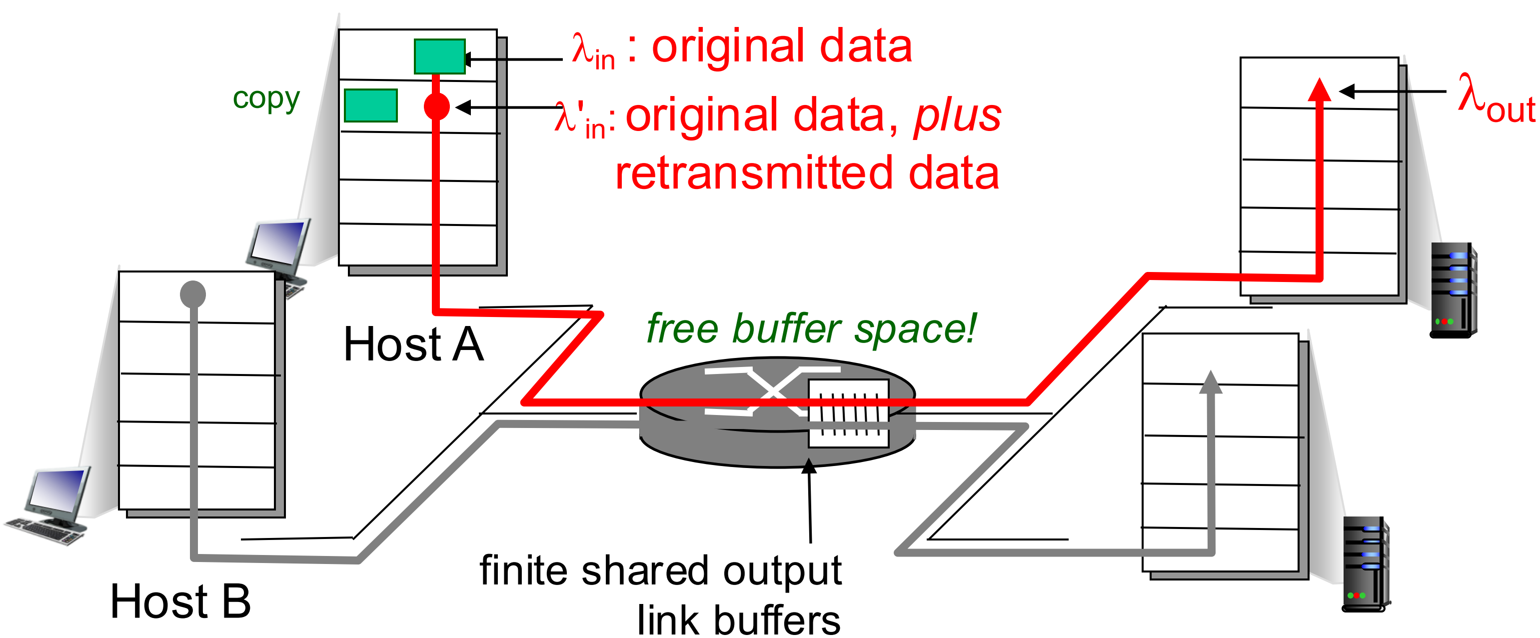


Causes/costs of Congestion: scenario 2

idealization: *perfect knowledge*

- sender sends only when router buffers available

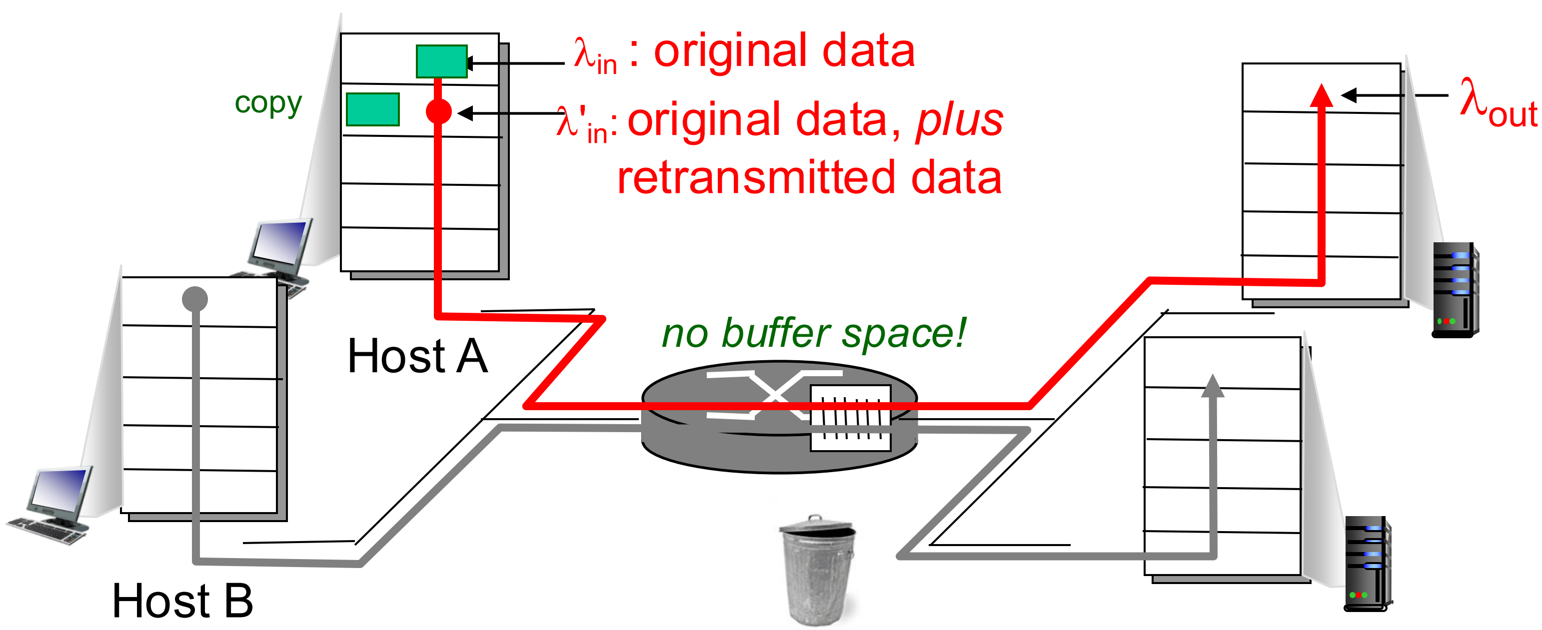




Causes/costs of Congestion: scenario 2

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

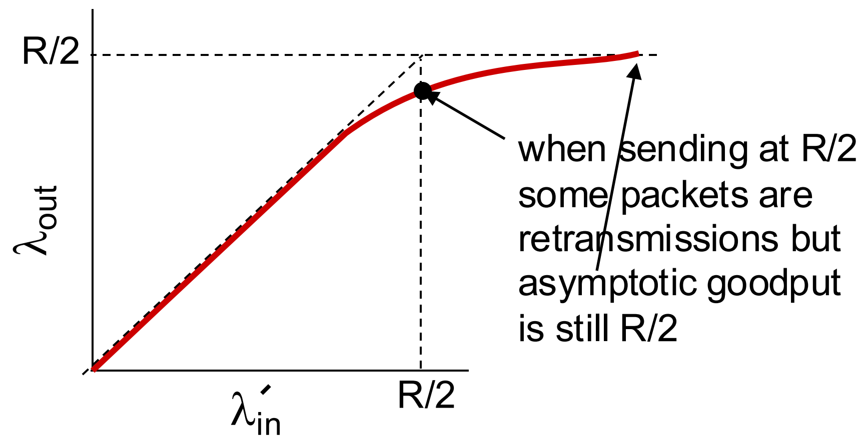
- sender only resends if packet *known* to be lost

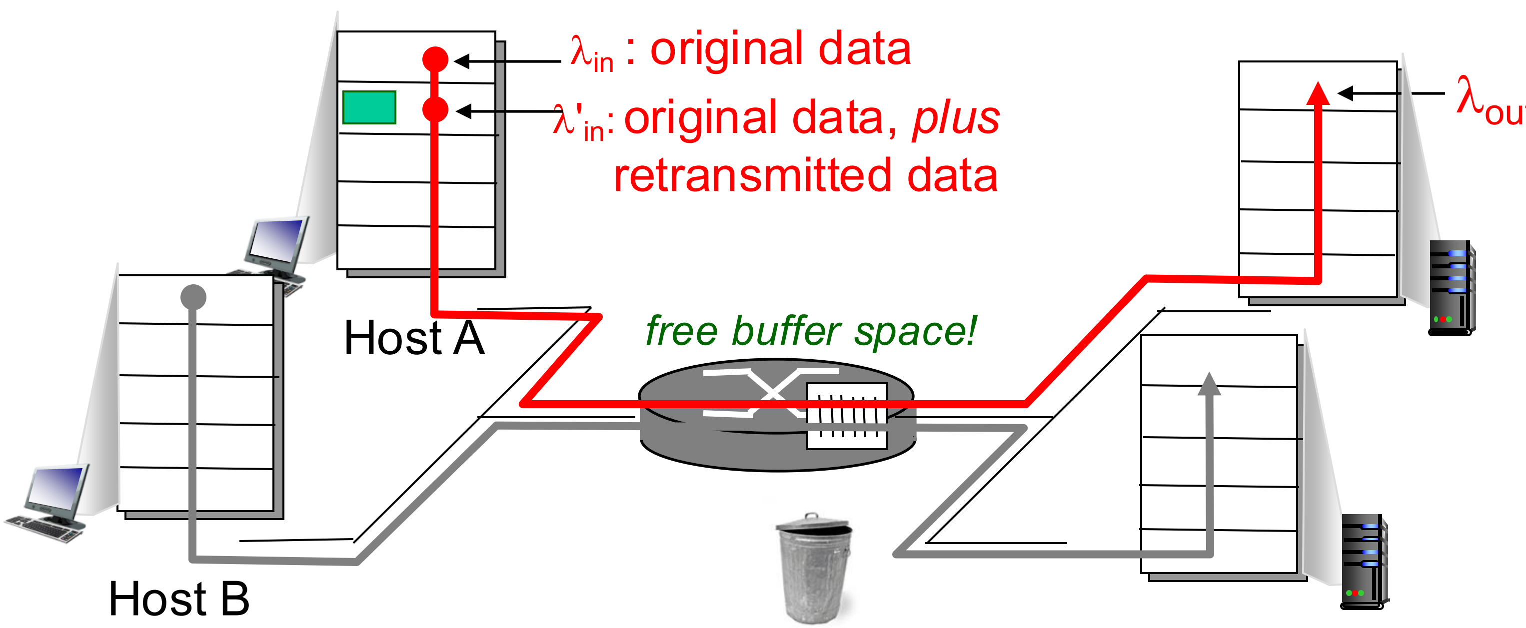


Causes/costs of congestion: scenario 2

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

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

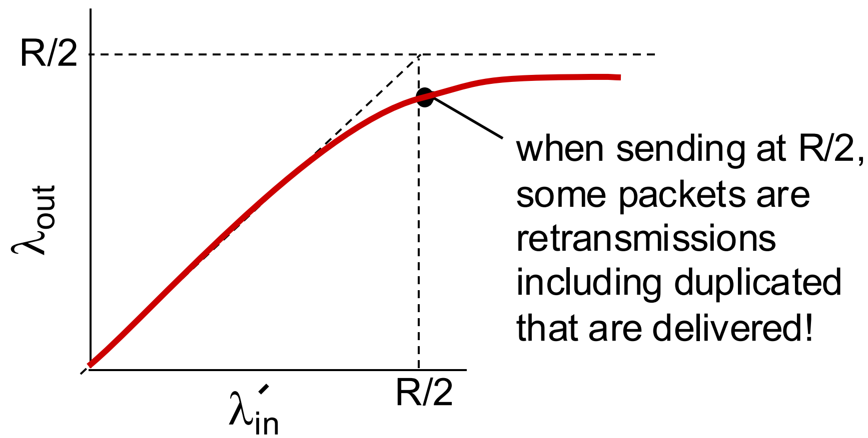


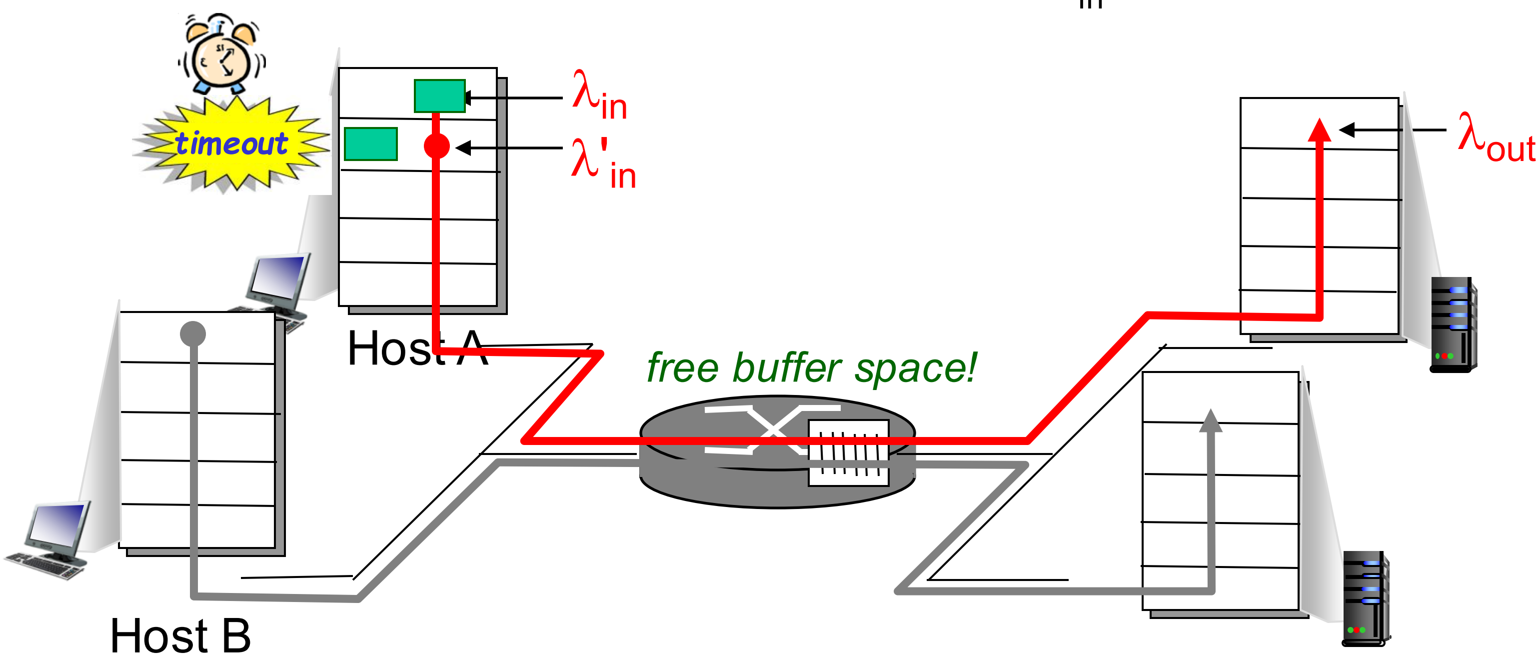


Causes/costs of Congestion: scenario 2

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

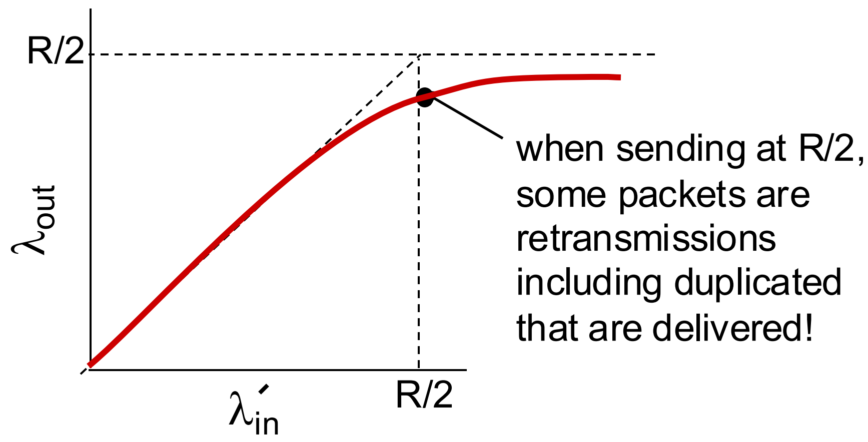




Causes/costs of Congestion: scenario 2

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



"costs" of congestion:

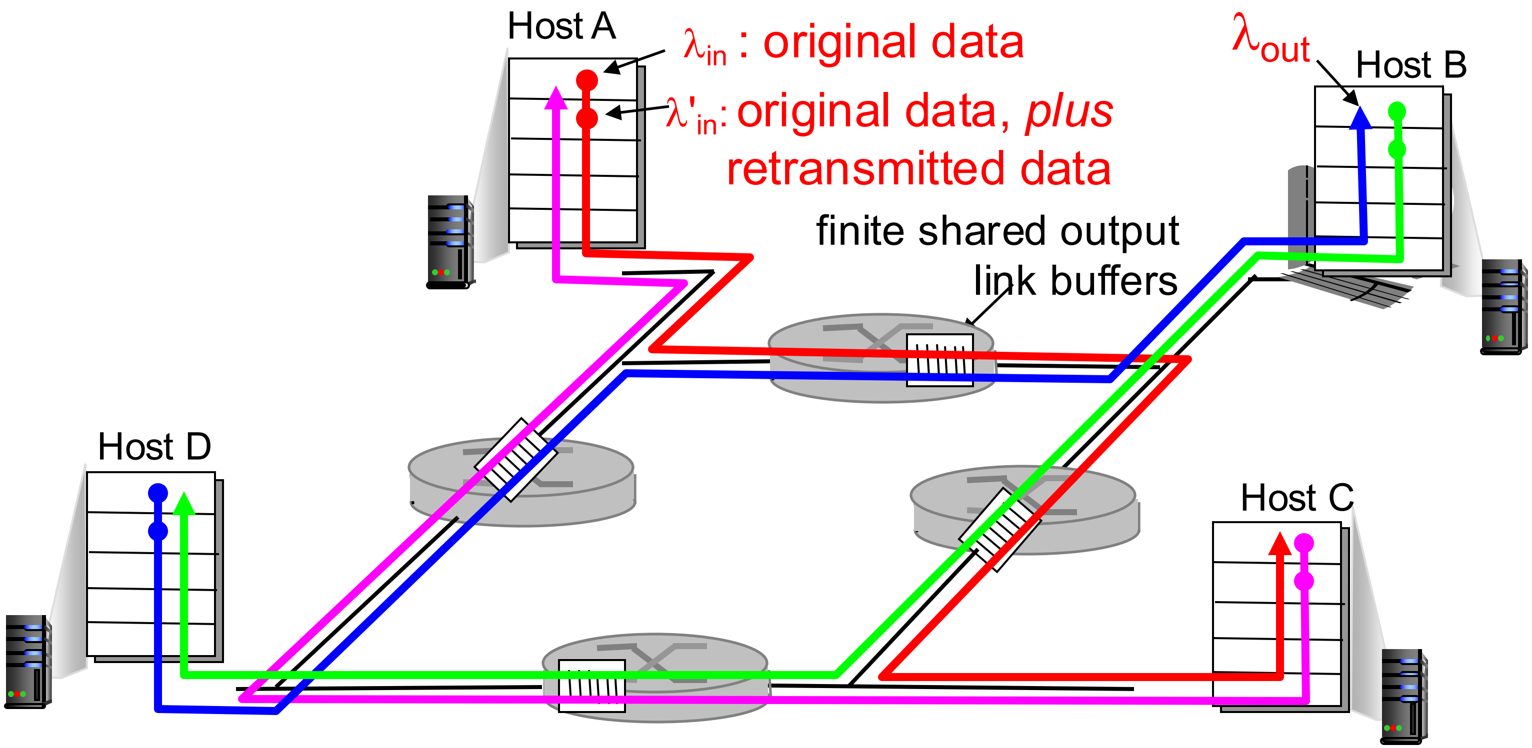
- more work (retrans) for given "goodput"
- unneeded retransmissions: link carries multiple copies of pkt
  - decreasing goodput

Causes/costs of Congestion: scenario 3

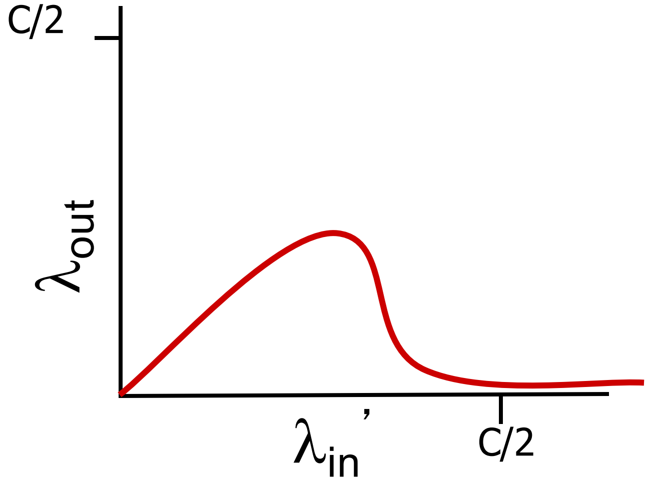
- four senders
- multihop paths
- timeout/retransmit

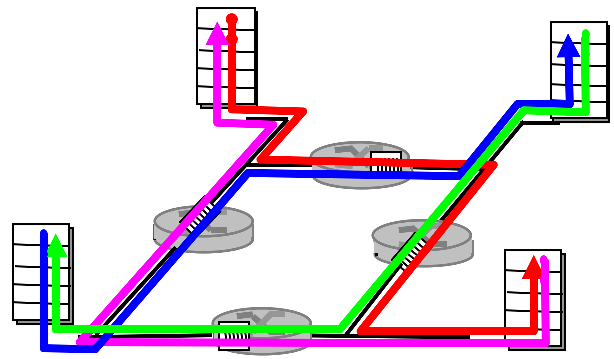
Q: what happens as λ_{in} and λ'_{in} increase ?

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



Causes/costs of Congestion: scenario 3





another "cost" of congestion:

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

Approaches towards Congestion Control

two broad approaches towards congestion control:

end-end congestion control:

- no explicit feedback from network
- congestion inferred from end-system observed loss, delay
- approach taken by TCP

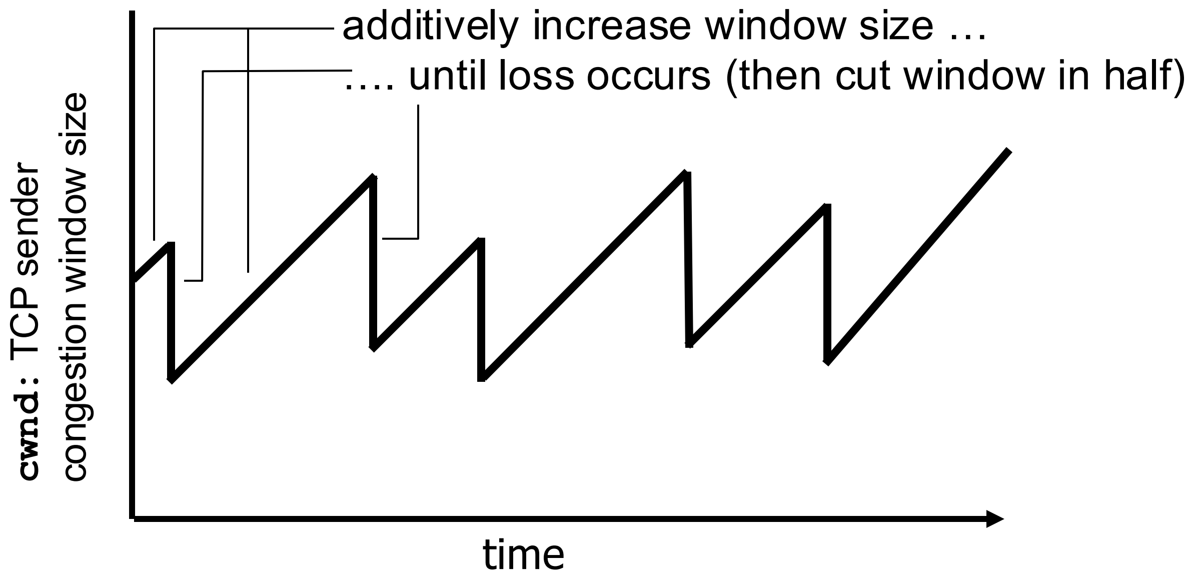
network-assisted congestion control:

- routers provide feedback to end systems
  - single bit indicating congestion (SNA, DECbit, TCP/IP ECN, ATM)
  - explicit rate for sender to send at

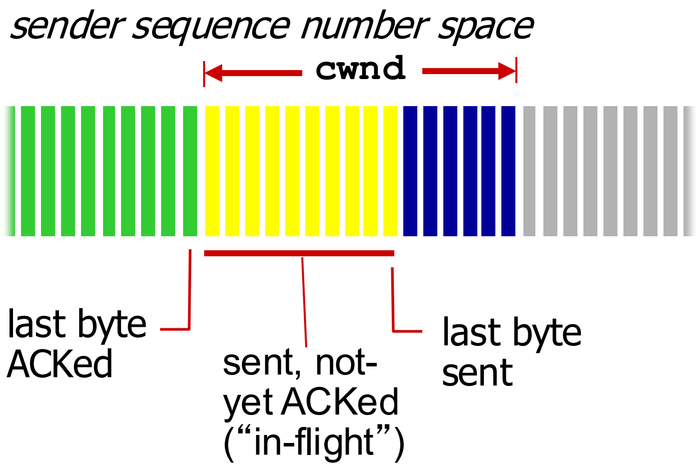
TCP Congestion Control: *additive increase multiplicative decrease*

- approach*:* sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs
  - *additive increase:* increase **cwnd** by 1 MSS (Maximum Segment Size) 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:

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

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

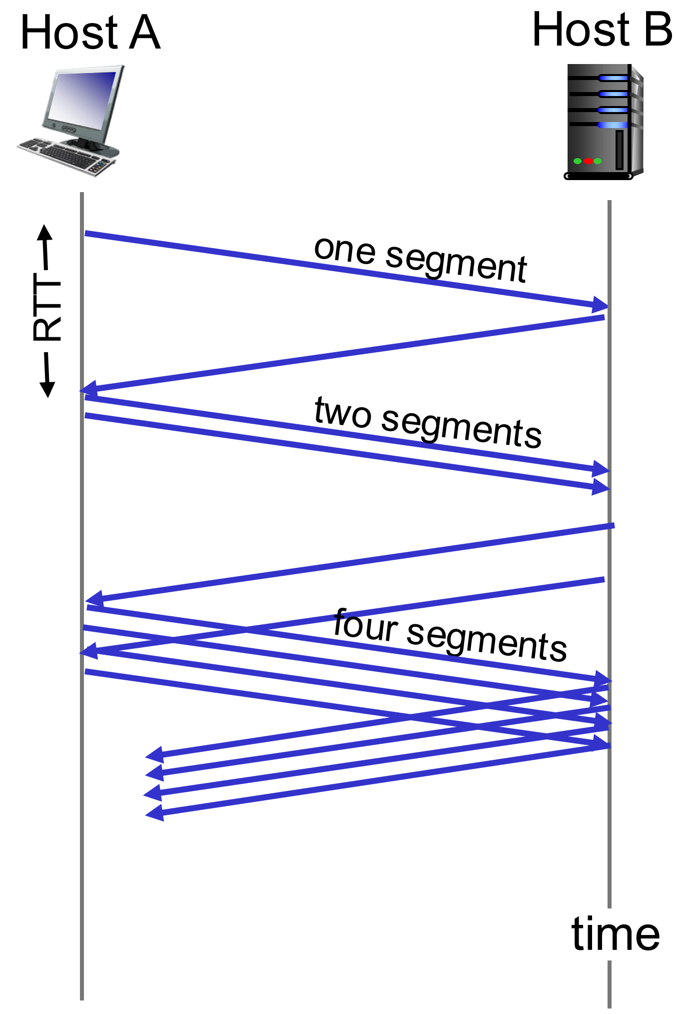
TCP sending rate:

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

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

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



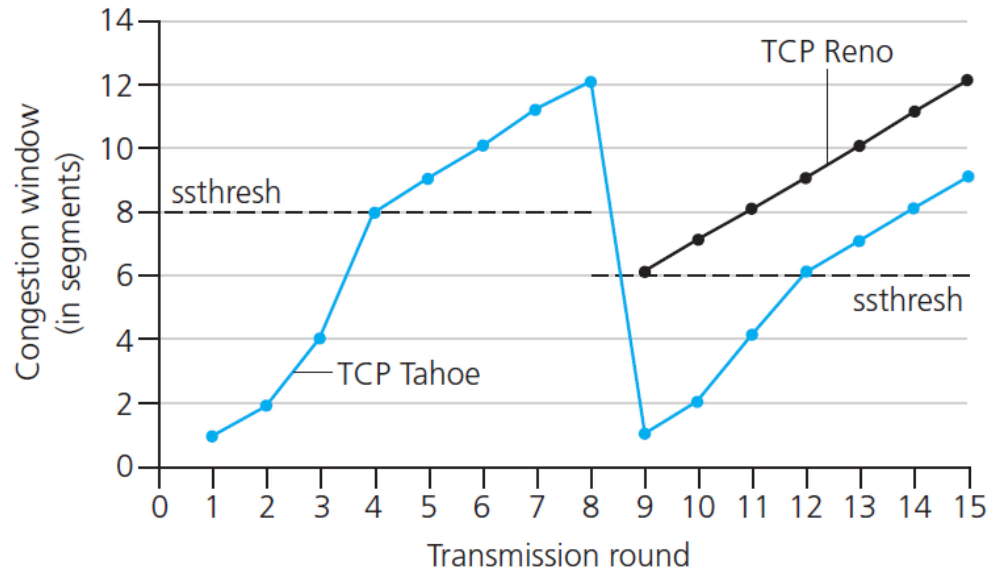
TCP: Detecting, Reacting to Loss

- loss indicated by timeout:
  - **`cwnd`** set to 1 MSS (TCP Tahoe) or is cut in half (TCP Reno)
  - window then grows exponentially (as in slow start) to **threshold**, then grows linearly
- loss indicated by 3 duplicate ACKs:
  - dup ACKs indicate network capable of delivering some segments -> move to fast recovery state
  - **`cwnd`** is set to **threshold + 3**
- **threshold** is set to ½ of **`cwnd`** in both cases of loss

TCP: Switching from Slow Start to CA

Q: when should the exponential increase switch to linear?

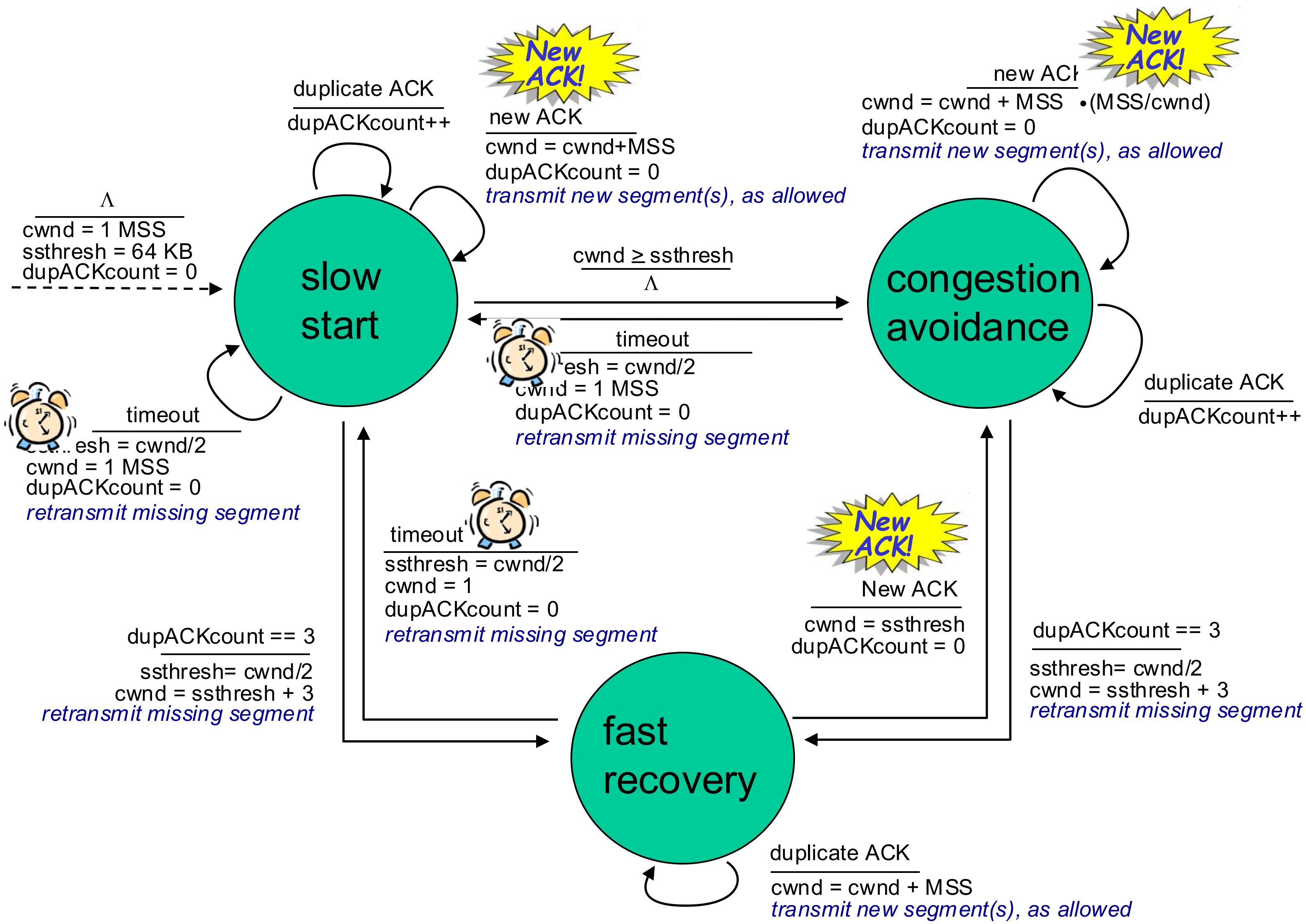
A: when **`cwnd`** gets to 1/2 of its value before timeout.



Implementation:

- variable **`ssthresh`**
- on loss event, **`ssthresh`** is set to 1/2 of **`cwnd`** just before loss event

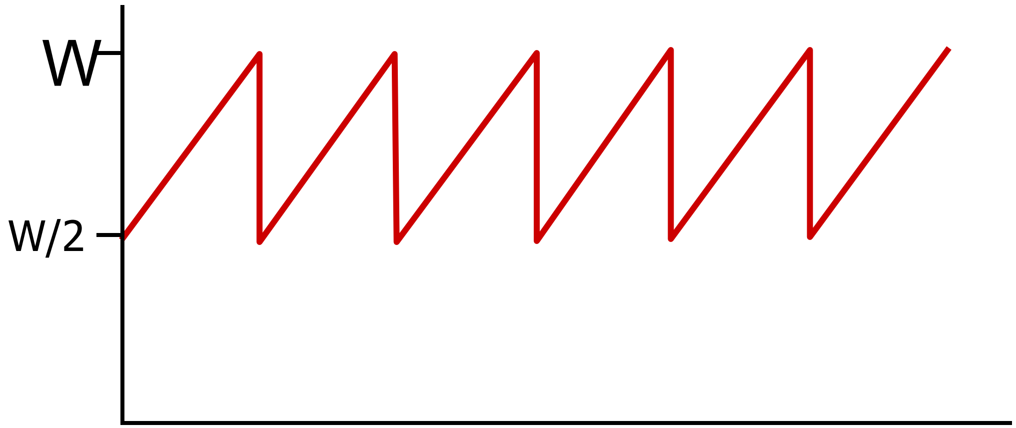
Summary: TCP Congestion Control



TCP Throughput

- Average TCP throughput as function of window size, RTT?
  - ignore slow start, assume always data to send
- W: window size (measured in bytes) where loss occurs
  - avg. window size (# in-flight bytes) is ¾ W
  - avg. thruput is 3/4W 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]:

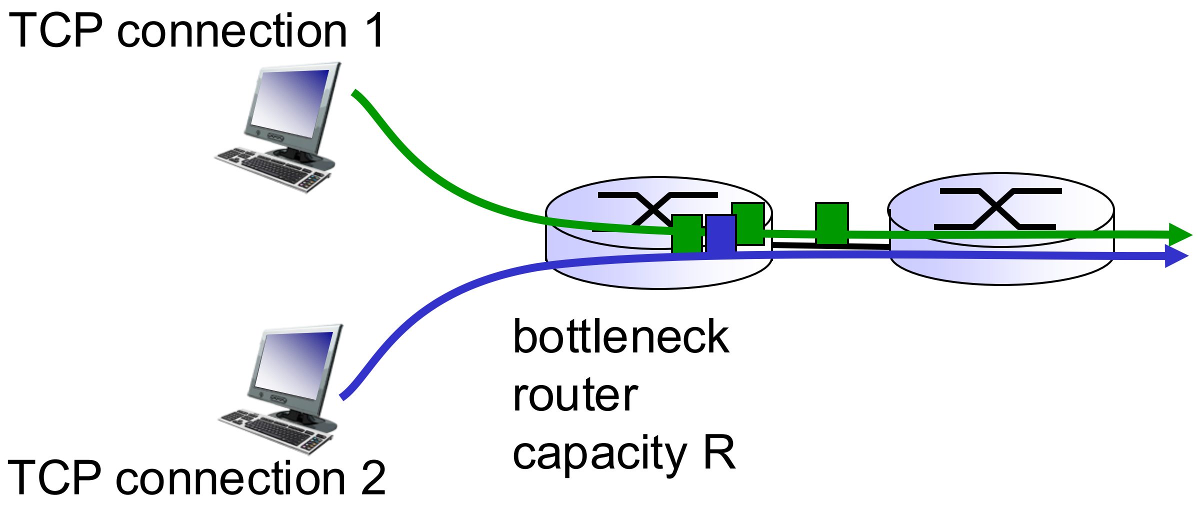
$$\text{TCP throughput} = \frac{1.22 \cdot MSS}{RTT \sqrt{L}}$$

➜ to achieve 10 Gbps throughput, need a loss rate of L = $2 \cdot 10^{-10}$ *– a very small loss rate!*

- new versions of TCP needed for high-speed connections

TCP Fairness

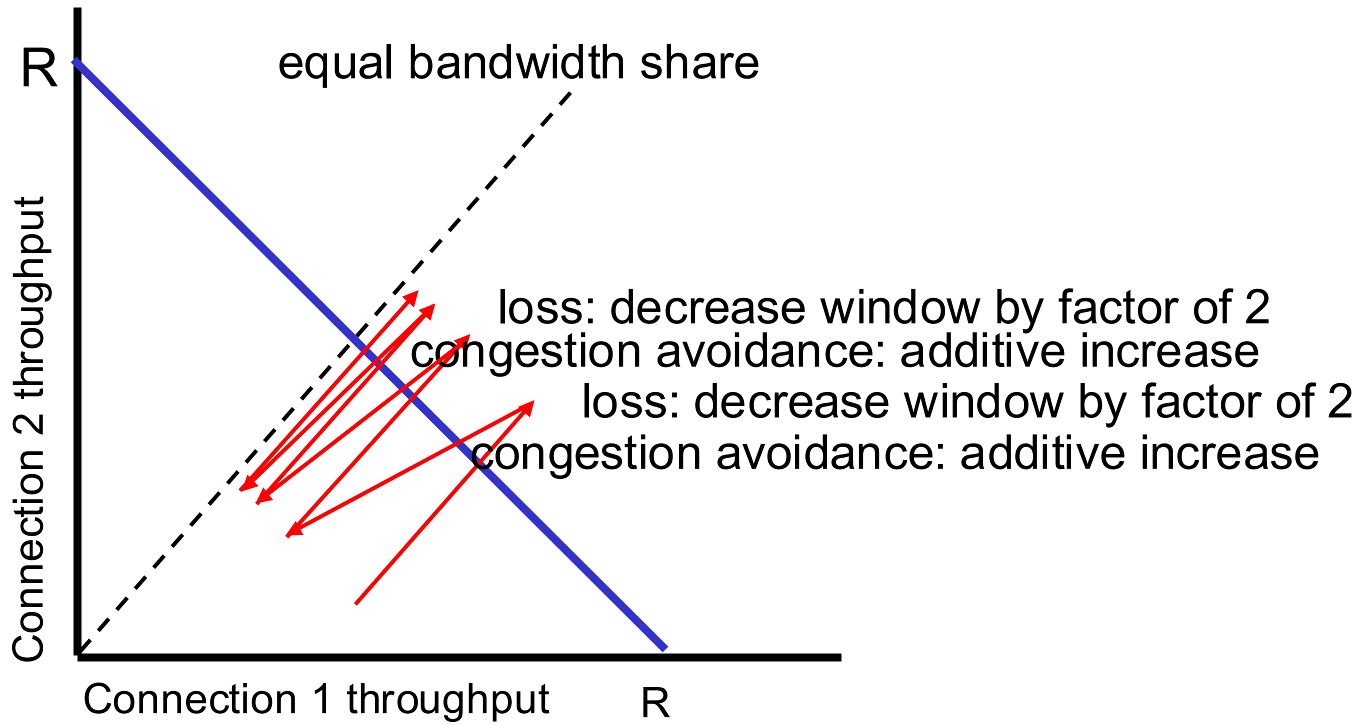
fairness goal: if k TCP sessions share same bottleneck link of bandwidth R, each should have average rate of R/k



Why is TCP fair?

two competing sessions:

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



Fairness – cont'd

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
- e.g., link of rate R with 9 existing connections:
  - new app asks for 1 TCP, gets rate R/10
  - new app asks for 11 TCPs, gets R/2