

Chapter 3

Transport Layer

A S M Touhidul Hasan, Ph.D.
Assistant Professor,
Department of CSE, UAP

Chapter 3 Outline of Lecture 10

3.6 principles of congestion control

3.7 TCP congestion control

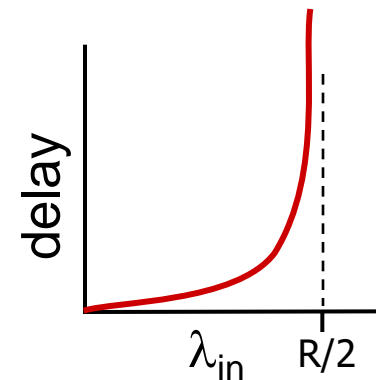
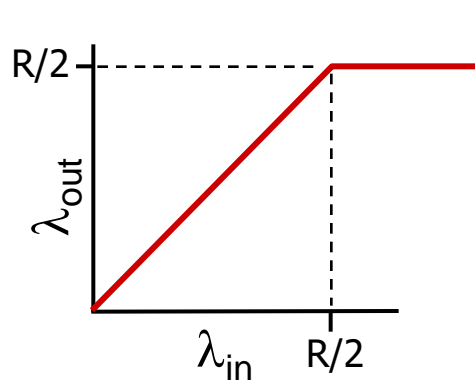
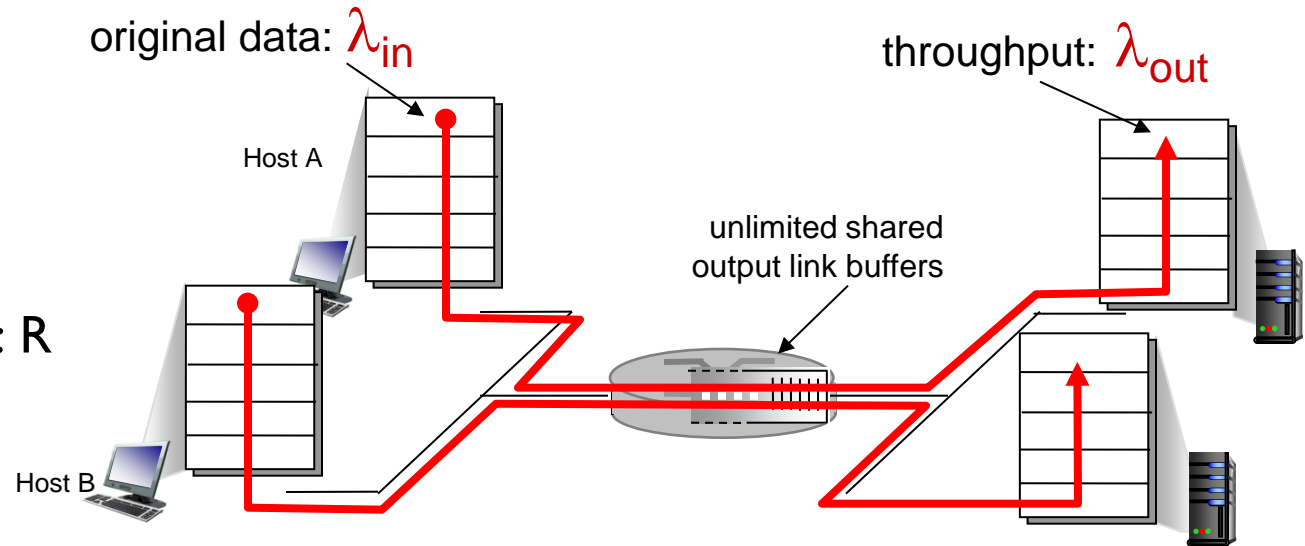
Principles of congestion control

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 I

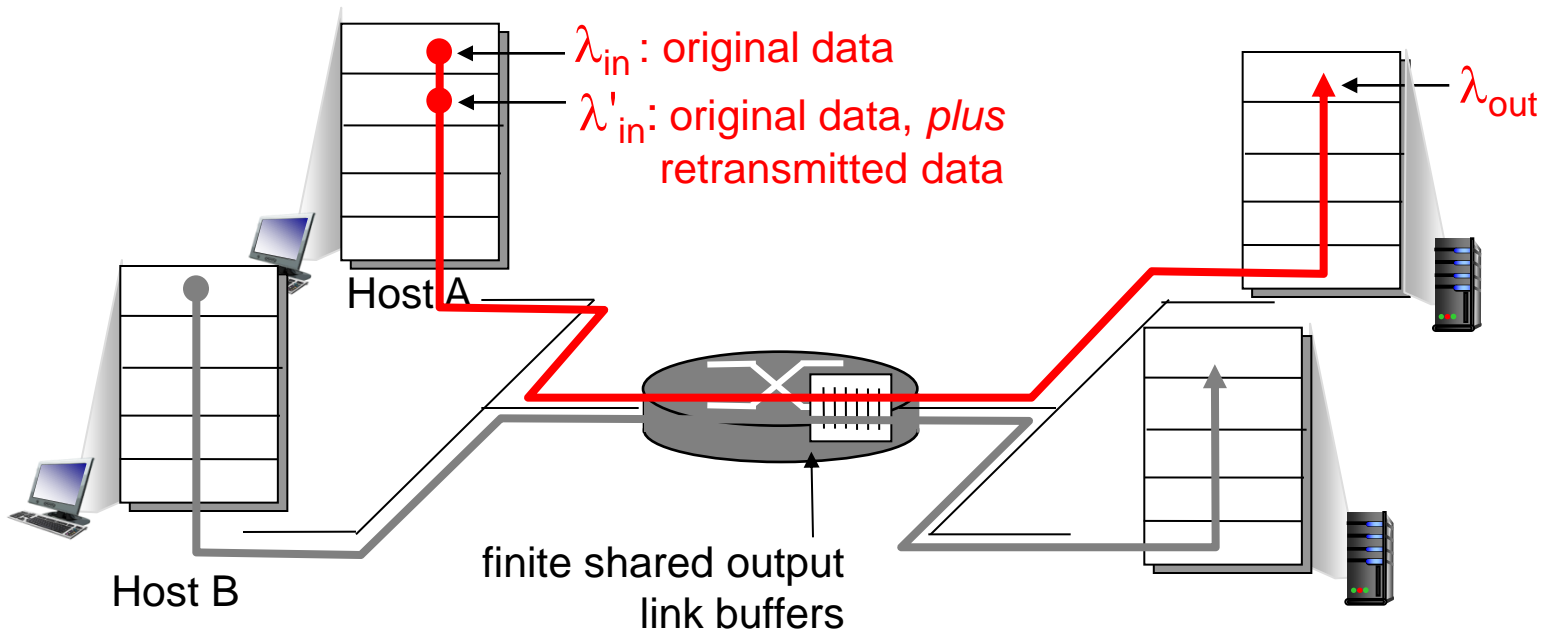
- ❖ two senders, two receivers
- ❖ one router, infinite buffers
- ❖ output link capacity: R
- ❖ no retransmission



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

Causes/costs of congestion: scenario 2

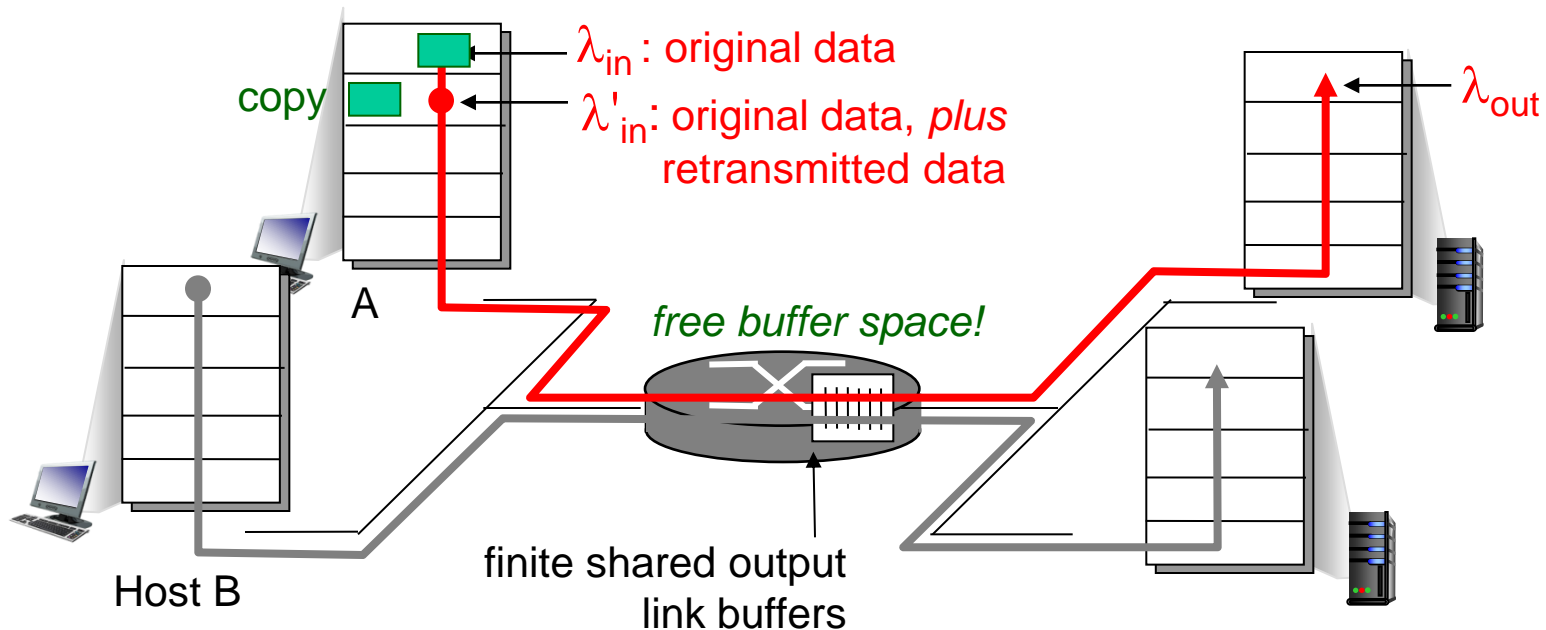
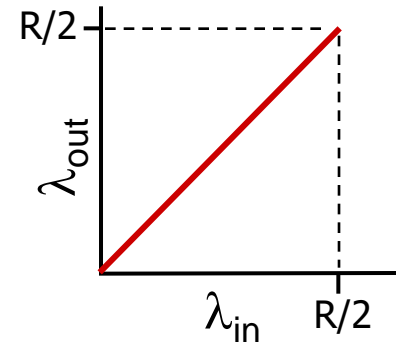
- ❖ one router, *finite* buffers
- ❖ sender retransmission of timed-out packet
 - application-layer input = application-layer output: $\lambda_{in} = \lambda_{out}$
 - transport-layer input includes *retransmissions* : $\lambda'_{in} \geq \lambda_{in}$



Causes/costs of congestion: scenario 2

idealization: perfect knowledge

- ❖ sender sends only when router buffers available

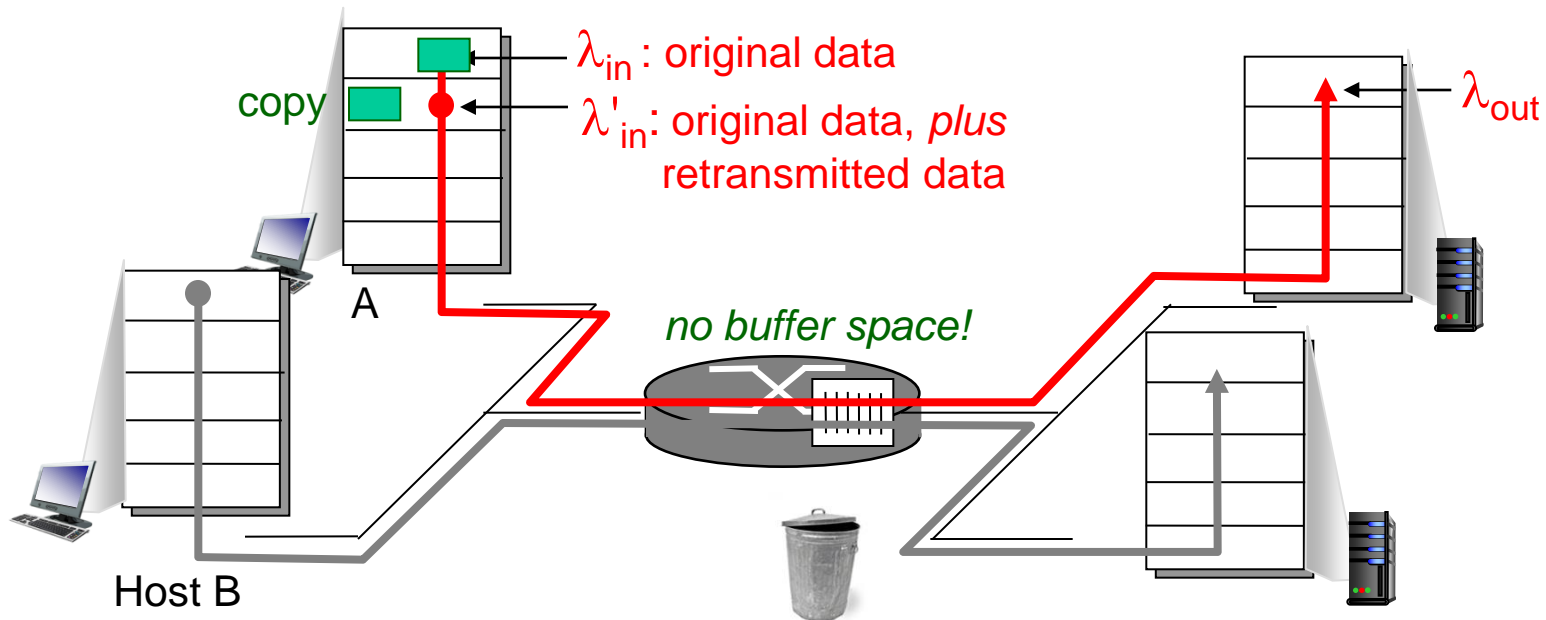


Causes/costs of congestion: scenario 2

Idealization: known loss

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

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

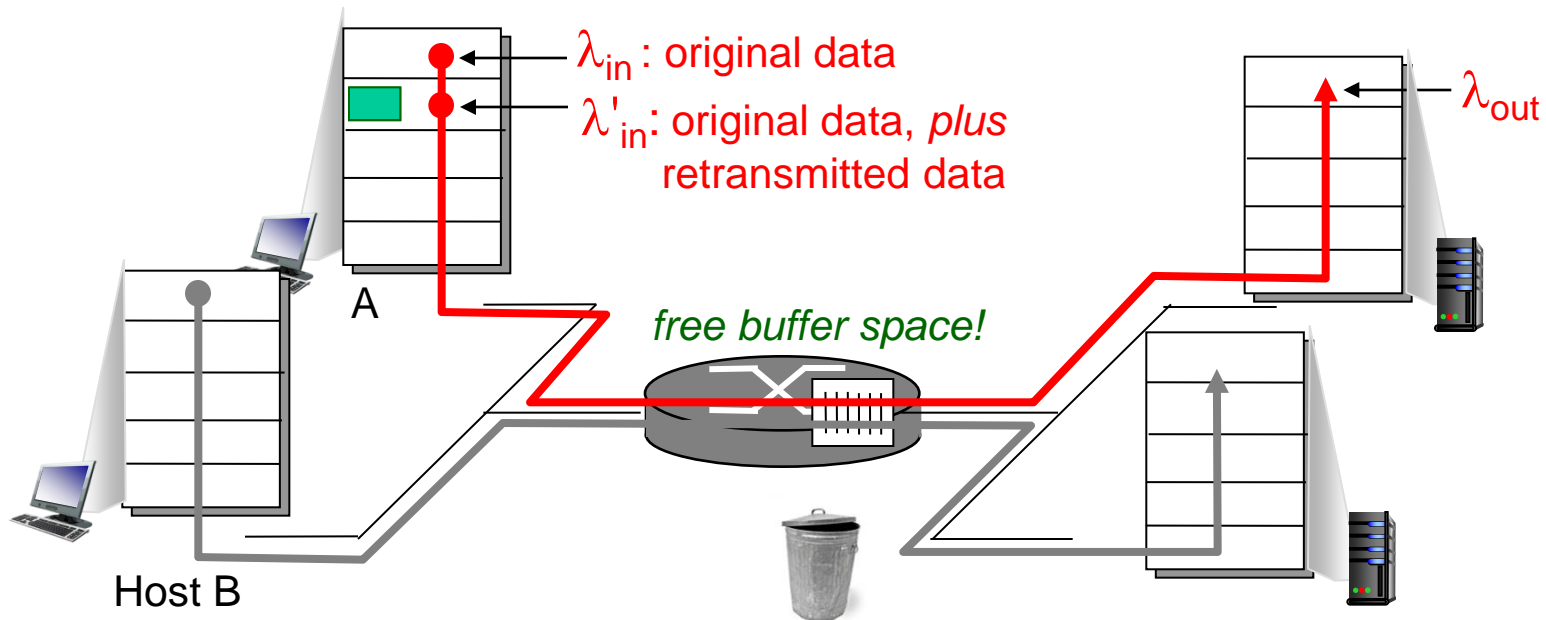
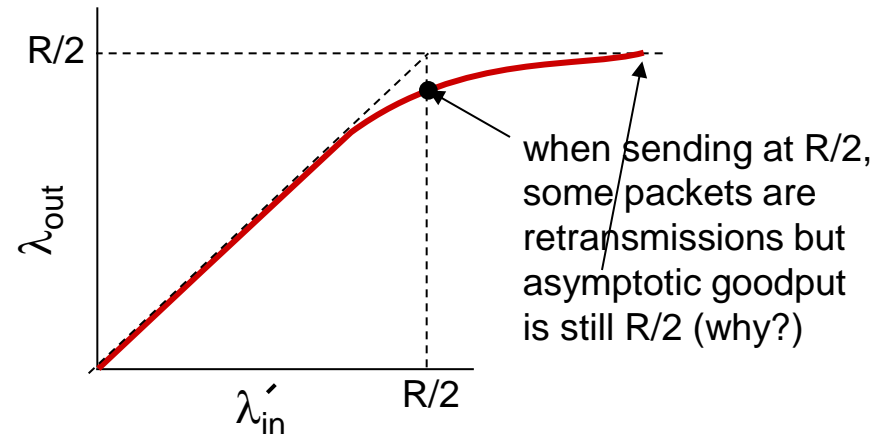


Causes/costs of congestion: scenario 2

Idealization: *known loss*

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

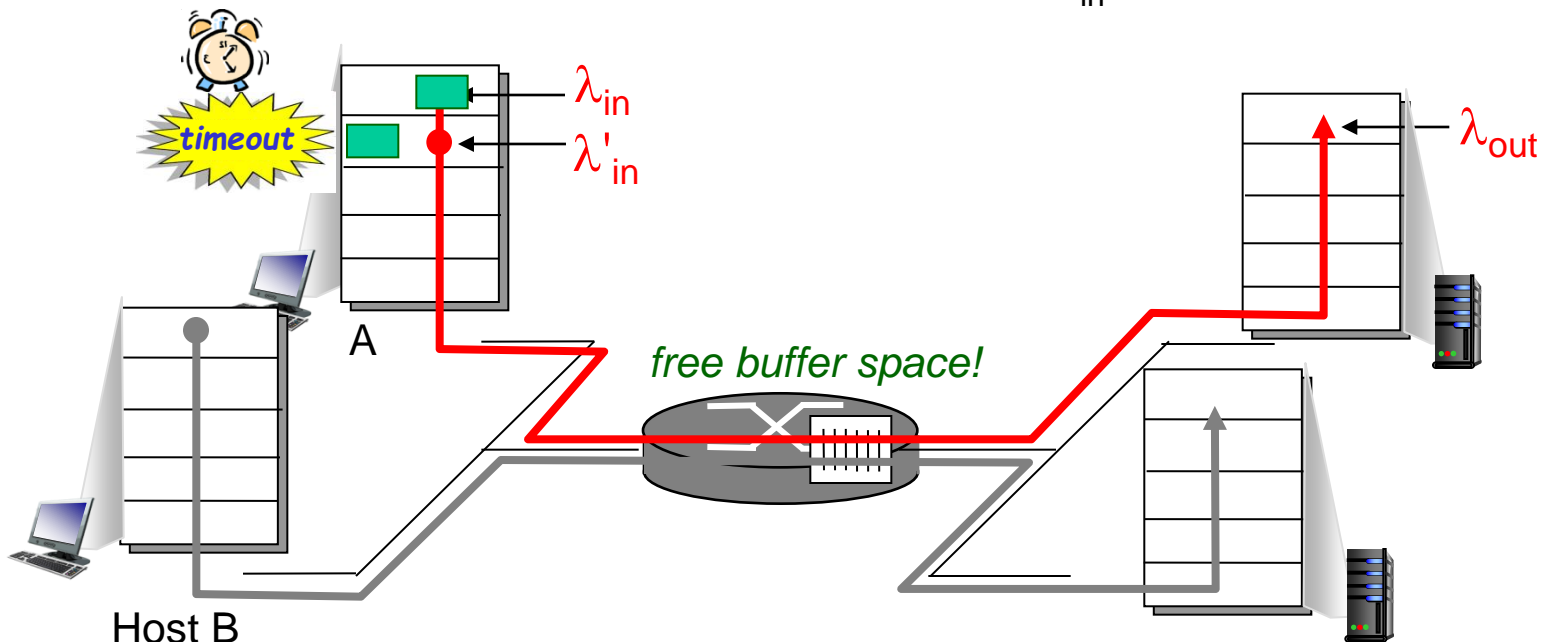
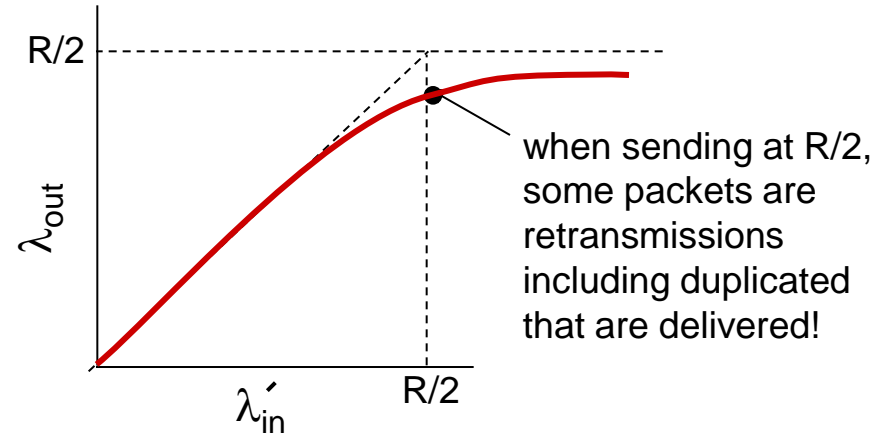
- ❖ sender only resends if
packet *known* to be lost



Causes/costs of congestion: scenario 2

Realistic: *duplicates*

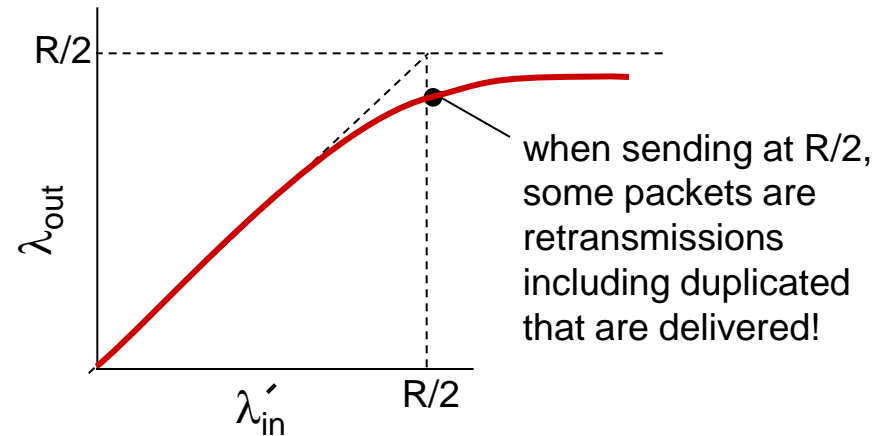
- ❖ packets can be lost, dropped at router due to full buffers
- ❖ sender times out prematurely, sending *two* copies, both of which are delivered



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:

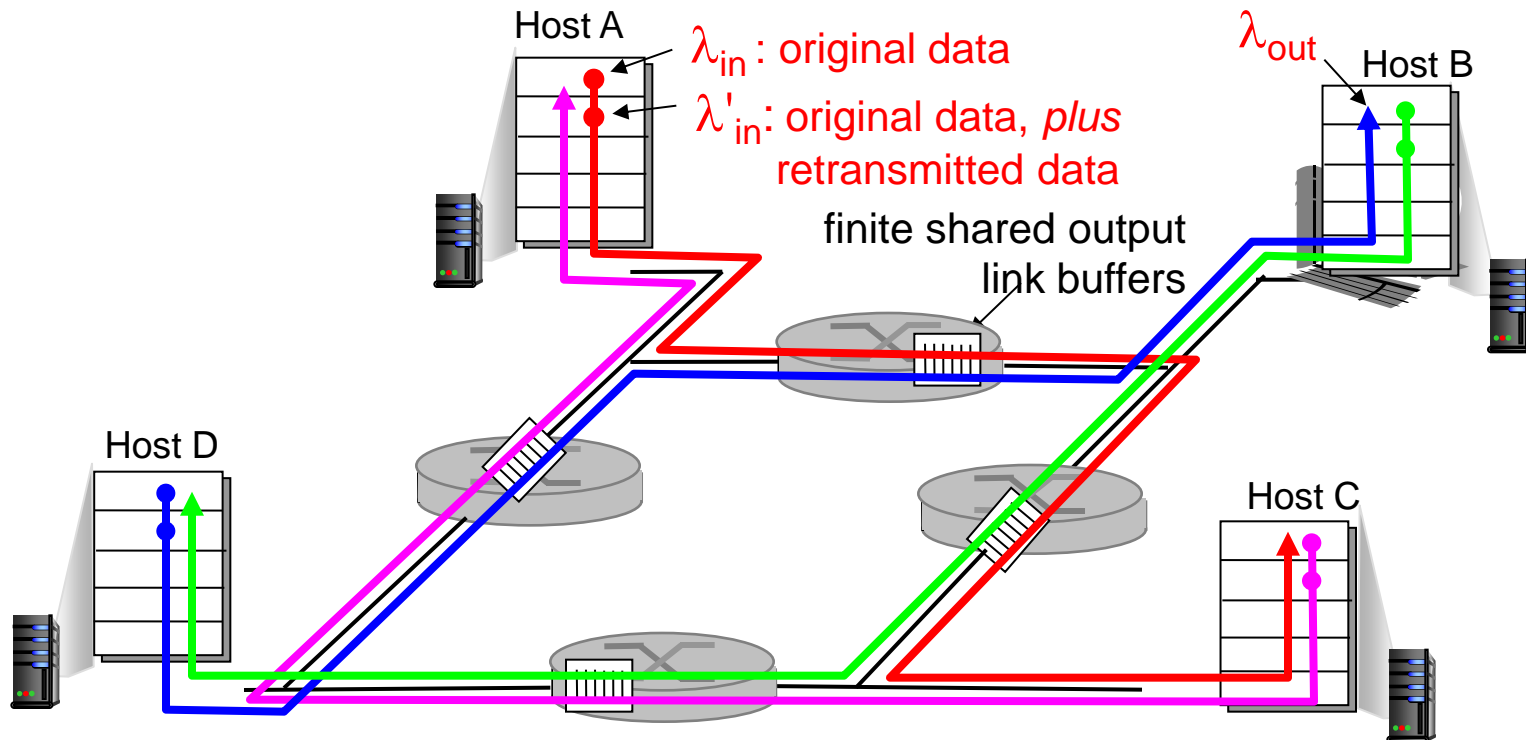
- ❖ more work (retrans) for given “goodput”
- ❖ unneeded retransmissions: link carries multiple copies of pkt
 - decreasing goodput

Causes/costs of congestion: scenario 3

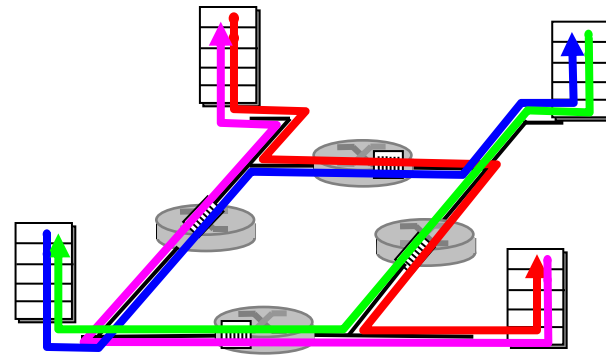
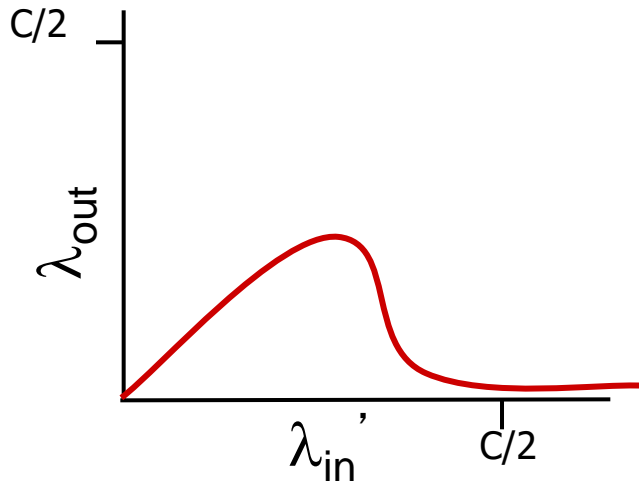
- ❖ four senders
- ❖ multihop paths
- ❖ timeout/retransmit

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 $\rightarrow 0$



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

Case study: ATM ABR congestion control

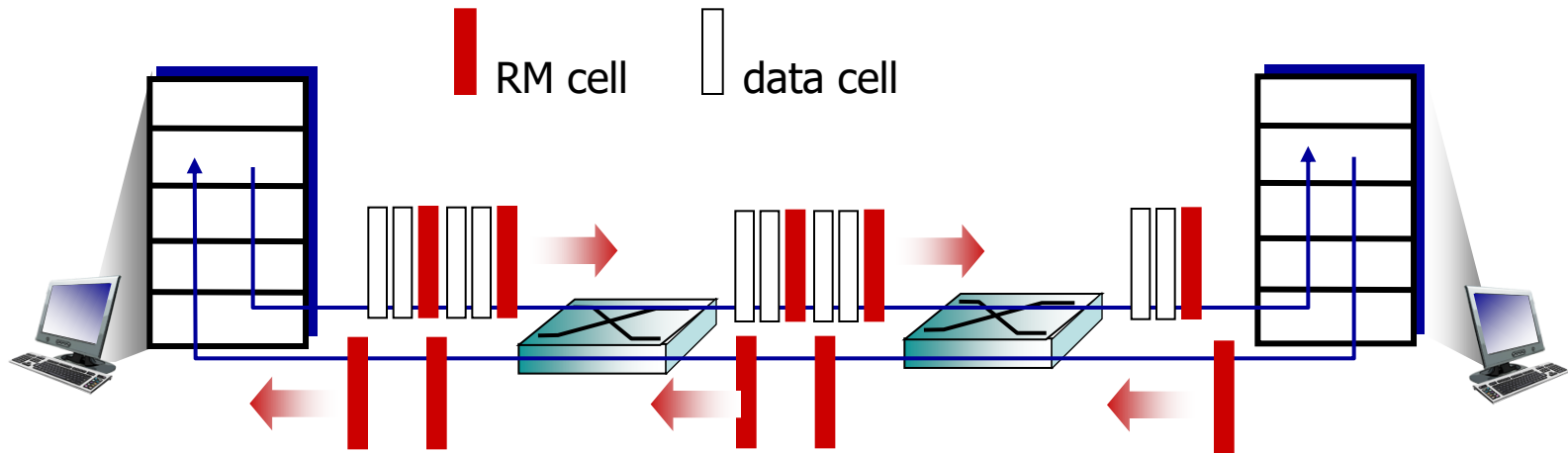
ABR: available bit rate:

- ❖ “elastic service”
- ❖ if sender's path “underloaded”:
 - sender should use available bandwidth
- ❖ if sender's path congested:
 - sender throttled to minimum guaranteed rate

RM (resource management) cells:

- ❖ sent by sender, interspersed with data cells
- ❖ bits in RM cell set by switches (“*network-assisted*”)
 - *NI bit*: no increase in rate (mild congestion)
 - *CI bit*: congestion indication
- ❖ RM cells returned to sender by receiver, with bits intact

Case study: ATM ABR congestion control



- ❖ two-byte ER (explicit rate) field in RM cell
 - congested switch may lower ER value in cell
 - senders' send rate thus max supportable rate on path
- ❖ EFCI bit in data cells: set to 1 in congested switch
 - if data cell preceding RM cell has EFCI set, receiver sets CI bit in returned RM cell

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

- segment structure
- reliable data transfer
- flow control
- connection management

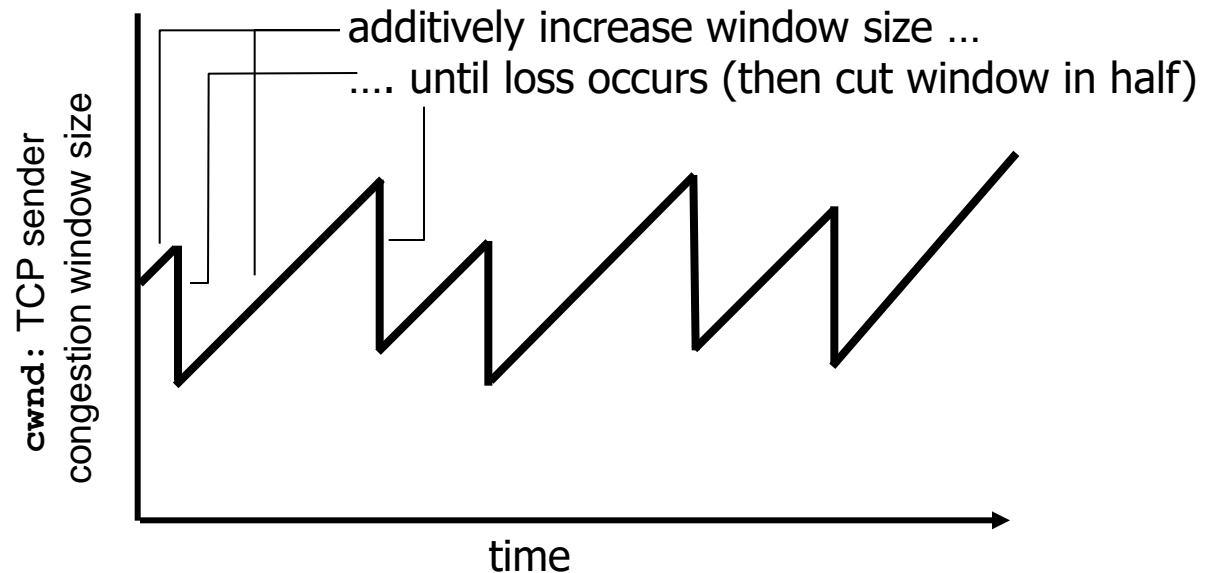
3.6 principles of congestion control

3.7 TCP congestion control

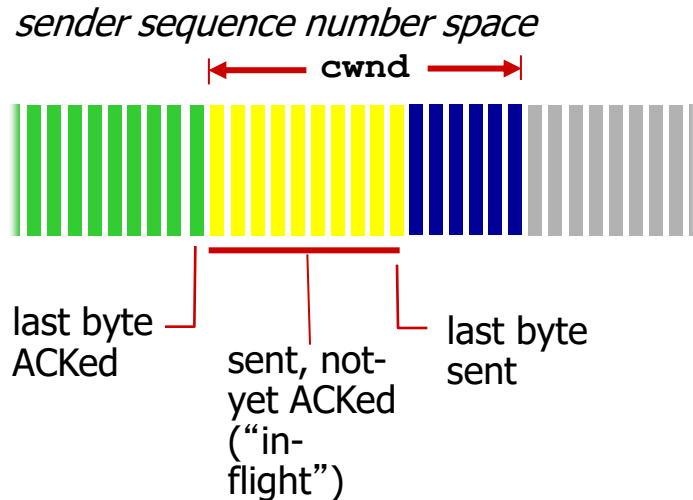
TCP congestion control: additive increase multiplicative decrease

- ❖ *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:

$$\text{LastByteSent} - \text{LastByteAked} \leq \text{cwnd}$$

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

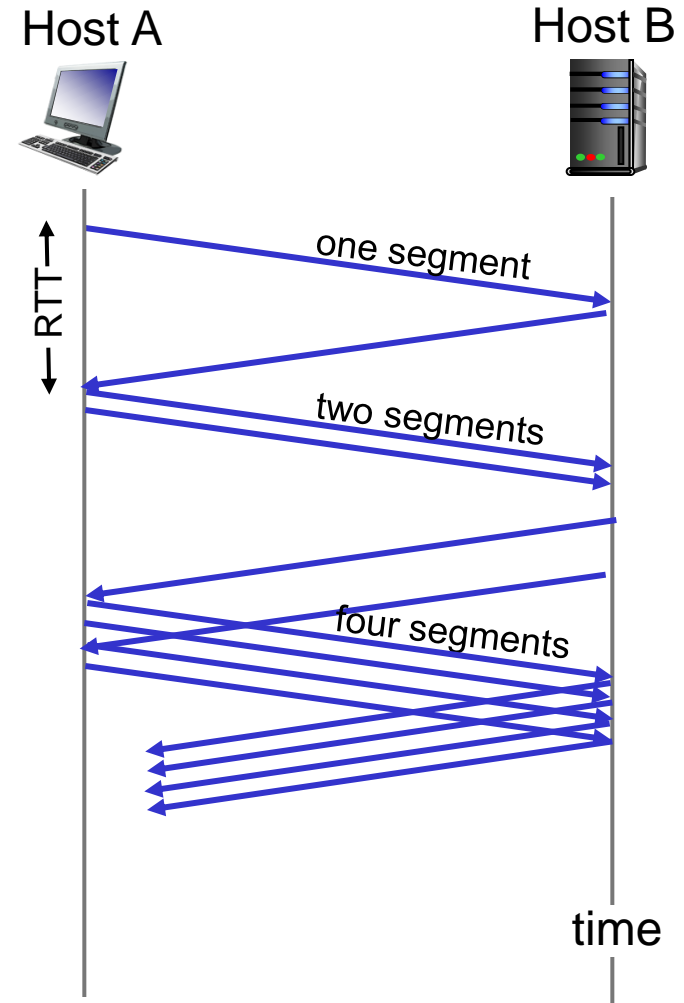
TCP sending rate:

- ❖ *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;
 - 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 then grows linearly
- ❖ TCP Tahoe always sets `cwnd` to 1 (timeout or 3 duplicate acks)

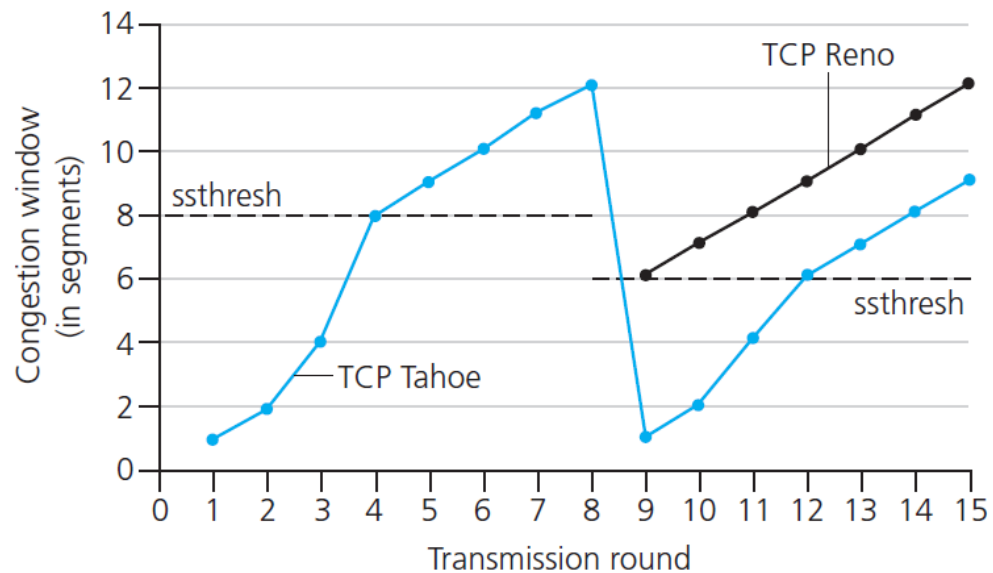
TCP: switching from slow start to CA

Q: when should the exponential increase switch to linear?

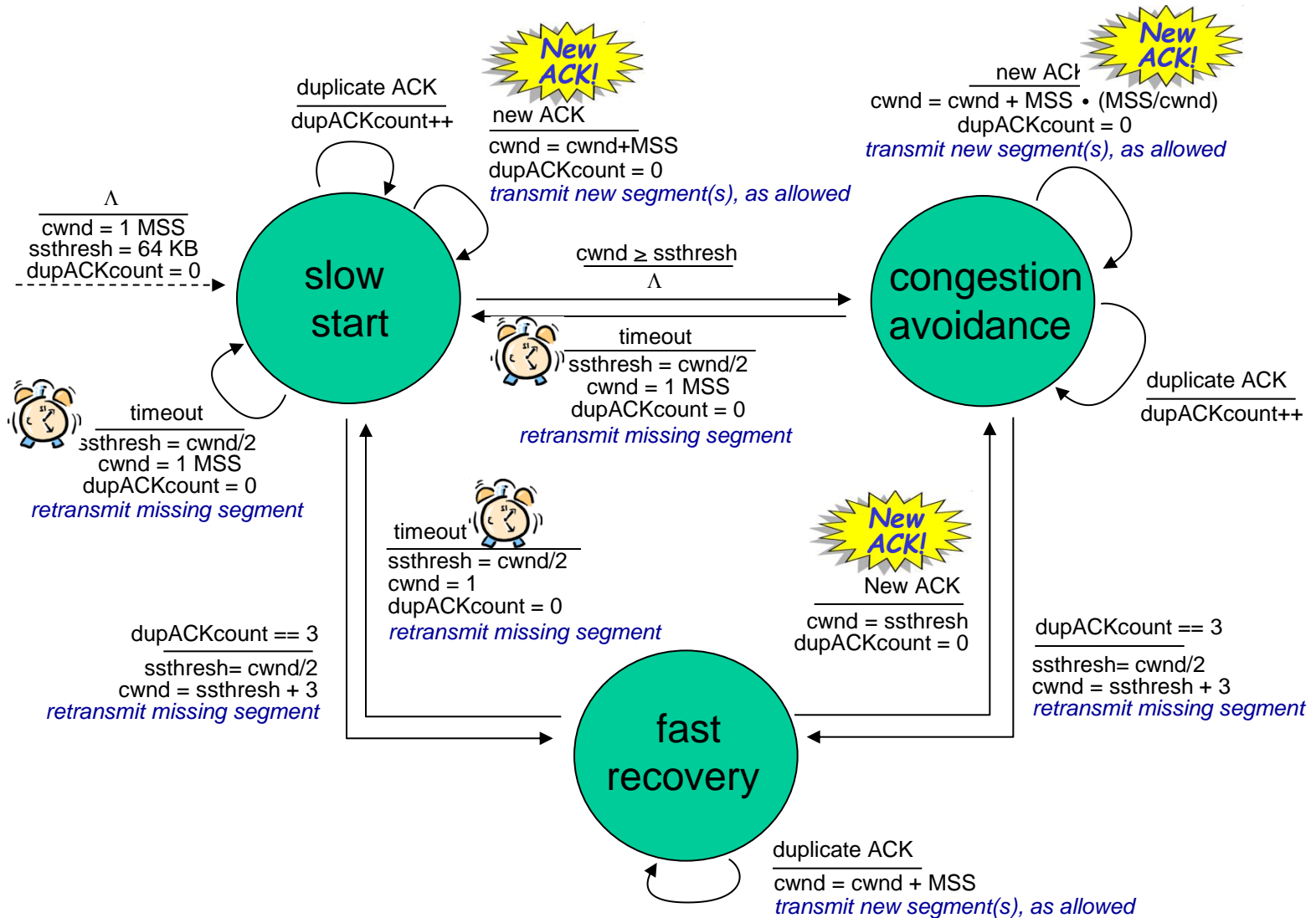
A: when **cwnd** gets to 1/2 of its value before timeout.

Implementation:

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



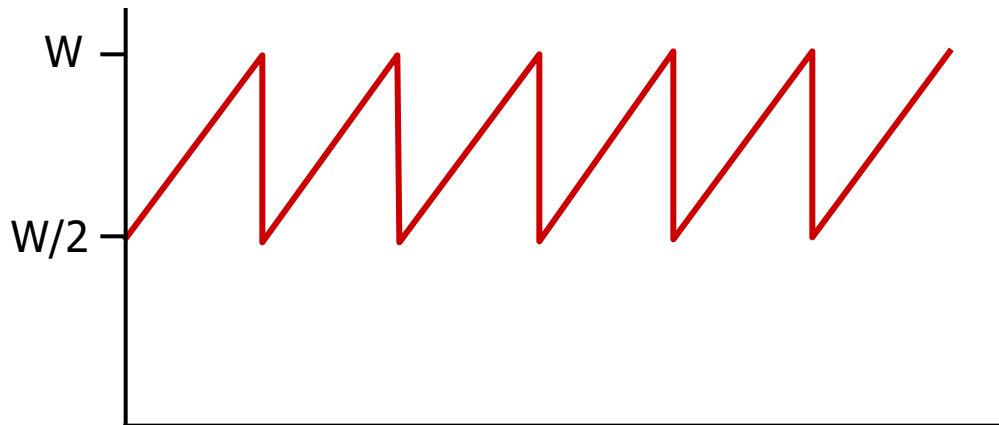
Summary: TCP Congestion Control



TCP throughput

- ❖ avg. 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**
 - avg. window size (# in-flight bytes) is $\frac{3}{4} W$
 - avg. thruput is $\frac{3}{4}W$ per RTT

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



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]:

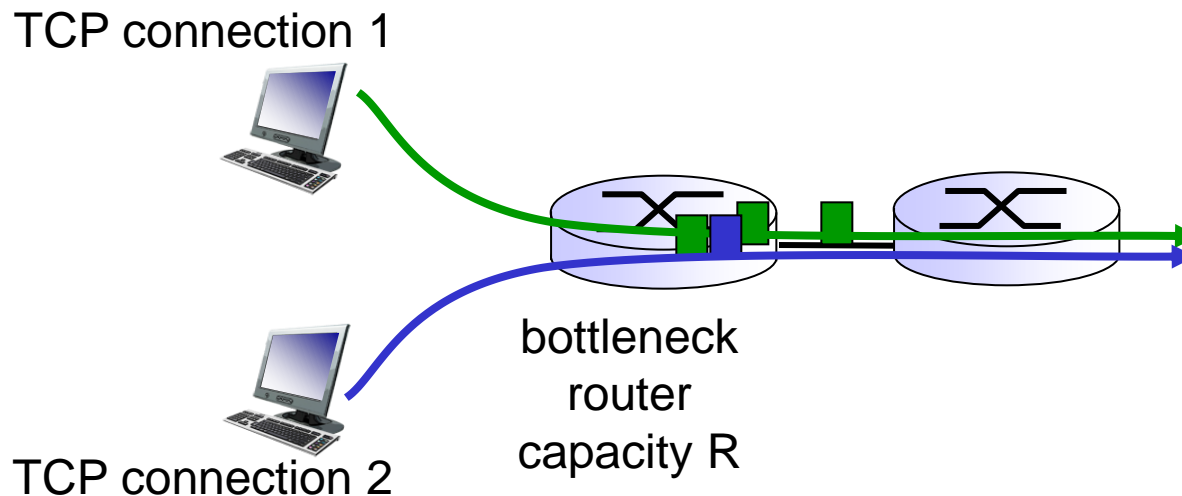
$$\text{TCP throughput} = \frac{1.22 \cdot \text{MSS}}{\text{RTT} \sqrt{L}}$$

→ 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

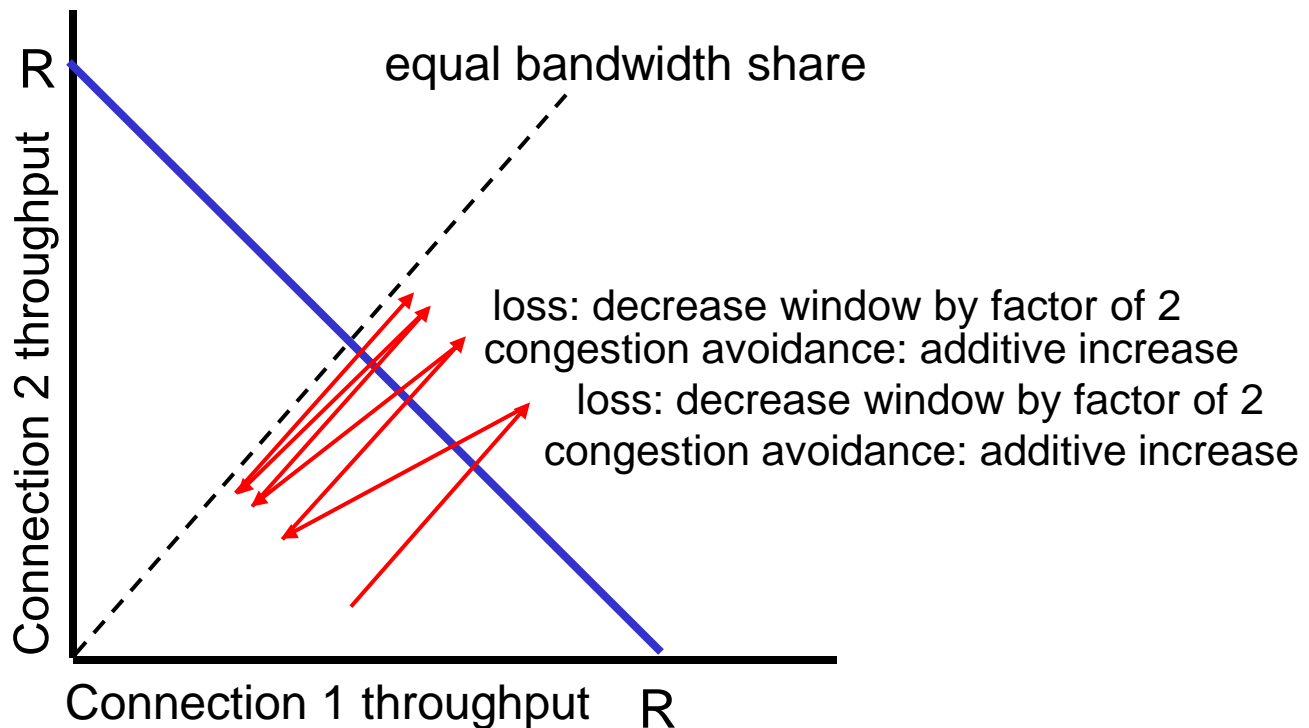
fairness goal: if K TCP sessions share same bottleneck link of bandwidth R , each should have average rate of R/K



Why is TCP fair?

two competing sessions:

- ❖ additive increase gives slope of 1, as throughput 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
- ❖ 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 $R/2$

Chapter 3: summary

- ❖ principles behind transport layer services:
 - multiplexing, demultiplexing
 - reliable data transfer
 - flow control
 - congestion control
- ❖ instantiation, implementation in the Internet
 - UDP
 - TCP

next:

- ❖ leaving the network “edge” (application, transport layers)
- ❖ into the network “core”