DATA COMMUNICATOIN NETWORKING

Instructor: Ouldooz Baghban Karimi

Course Book: Computer Networking, A Top-Down Approach
By: Kurose, Ross

Course Overview

- **Basics of Computer Networks**
  - Internet & Protocol Stack
  - Application Layer
  - Transport Layer
  - Network Layer
  - Data Link Layer

- **Advanced Topics**
  - Case Studies of Computer Networks
  - Internet Applications
  - Network Management
  - Network Security

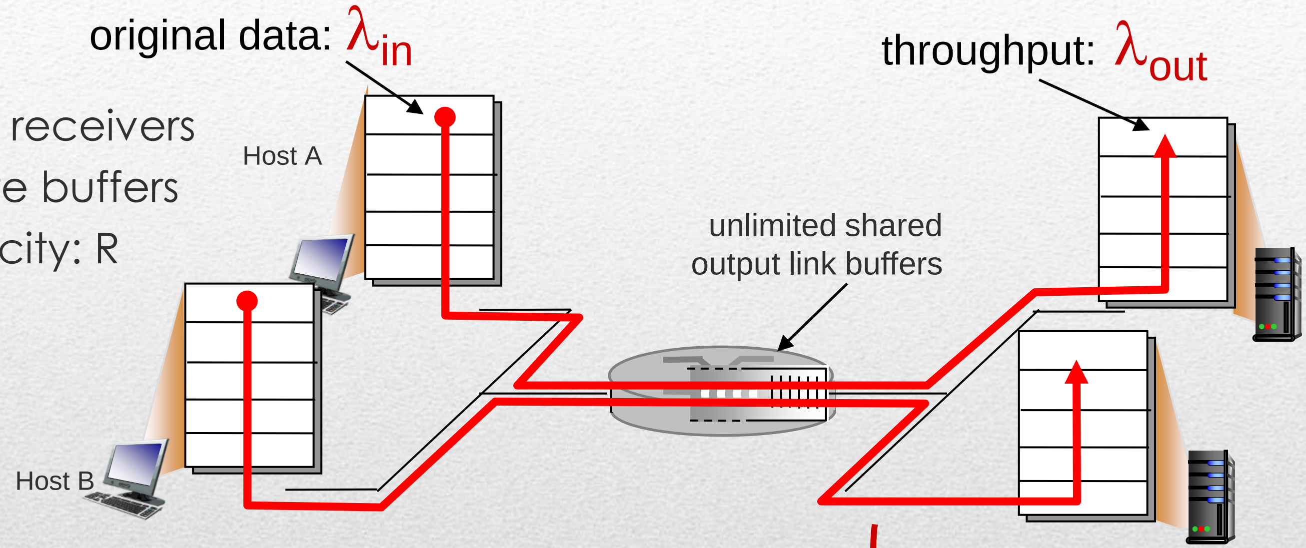
Congestion

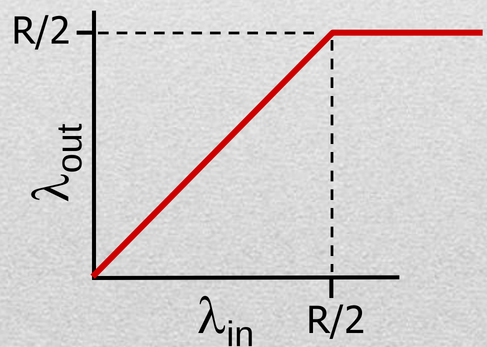
- **Too many sources sending too much data too fast for network to handle**
- Different from flow control
- **Manifestations**
  - Lost packets (buffer overflow at routers)
  - Long delays (queuing in router buffers)

Causes & Costs of Congestion

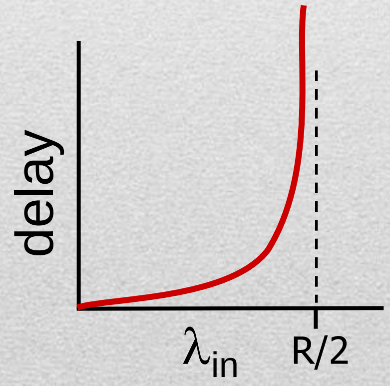
Scenario (1)

- Two senders, two receivers
- One router, infinite buffers
- Output link capacity: R
- No retransmission





Maximum per-connection throughput: R/2



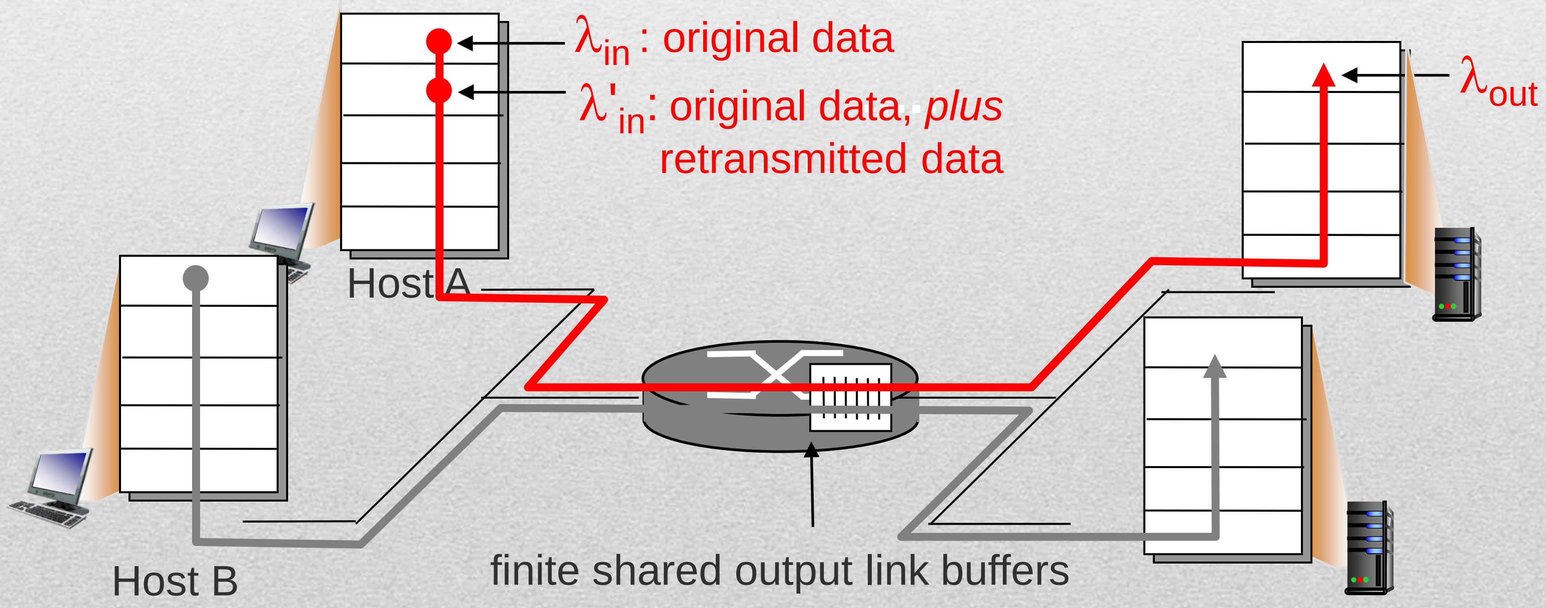
Large delays as arrival rate, λ_{in}, approaches capacity

Causes & Costs of Congestion

Scenario (2)

- One router, *finite* buffers
- Sender retransmission of timed-out packet
  - Application-layer input = application-layer output: $\lambda_{in} = \lambda_{out}$

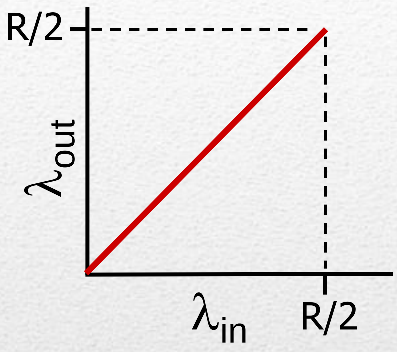
  Transport-layer input includes *retransmissions* : $\lambda_{in} \geq \lambda_{in}$

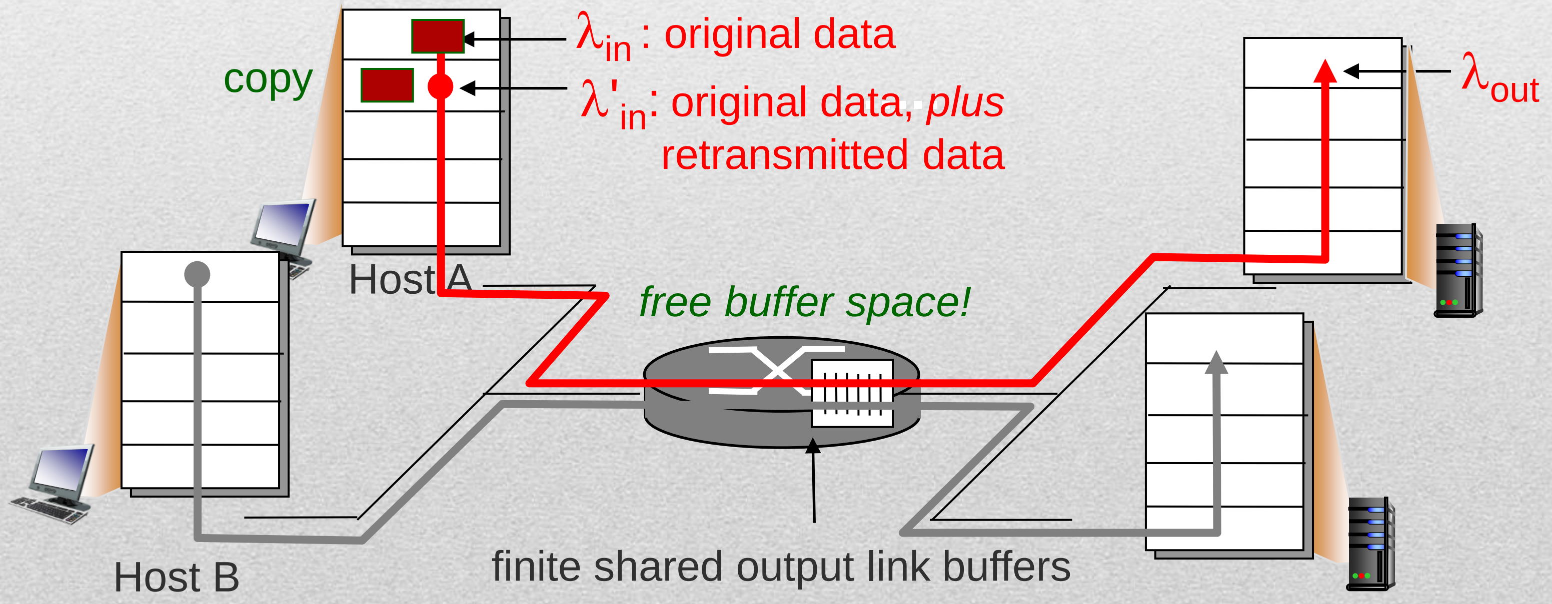


Causes & Costs of Congestion

Idealization: Perfect Knowledge

- Sender sends only when router buffers available

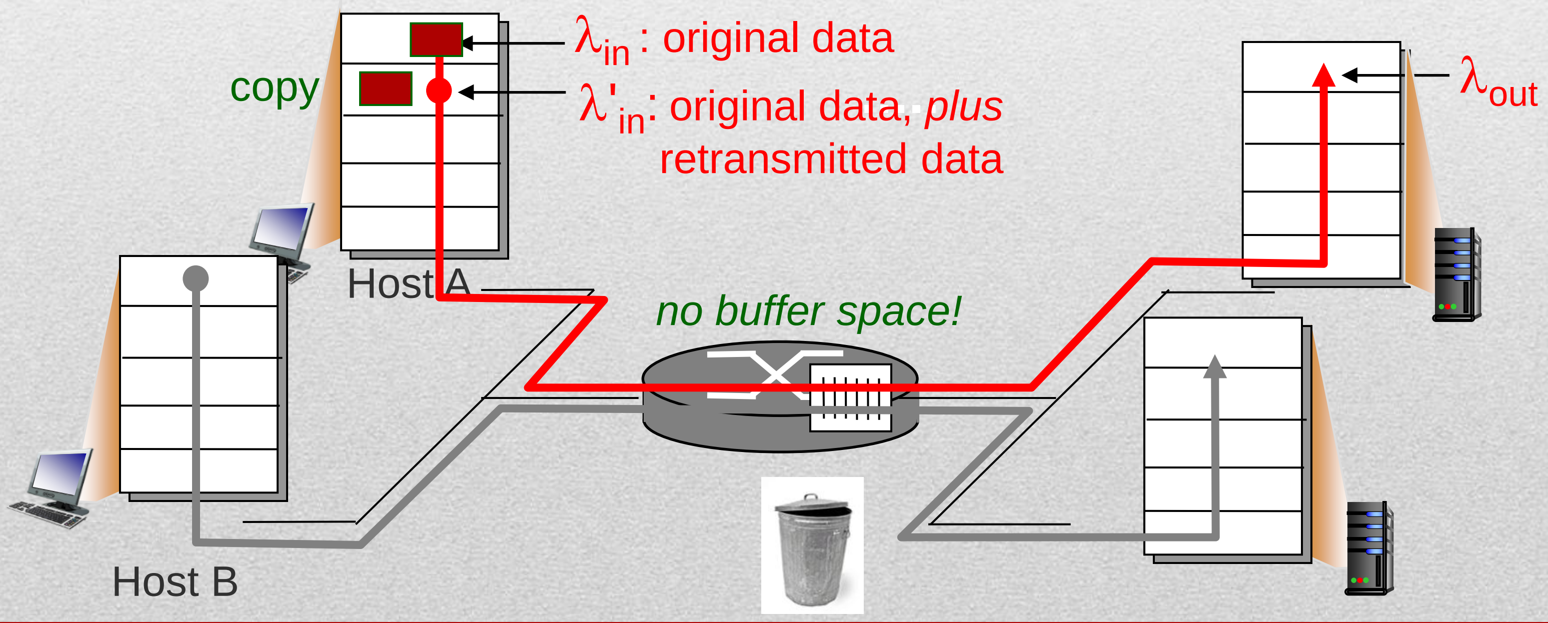




Causes & Costs of Congestion

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

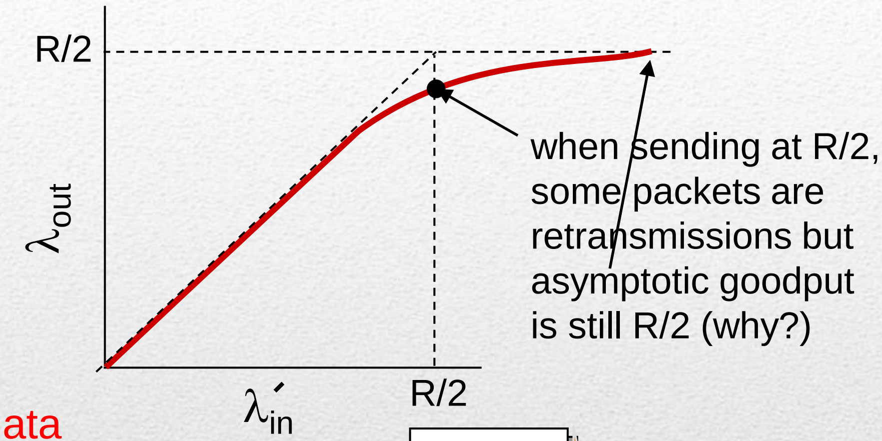
- Sender only resends if packet known to be lost

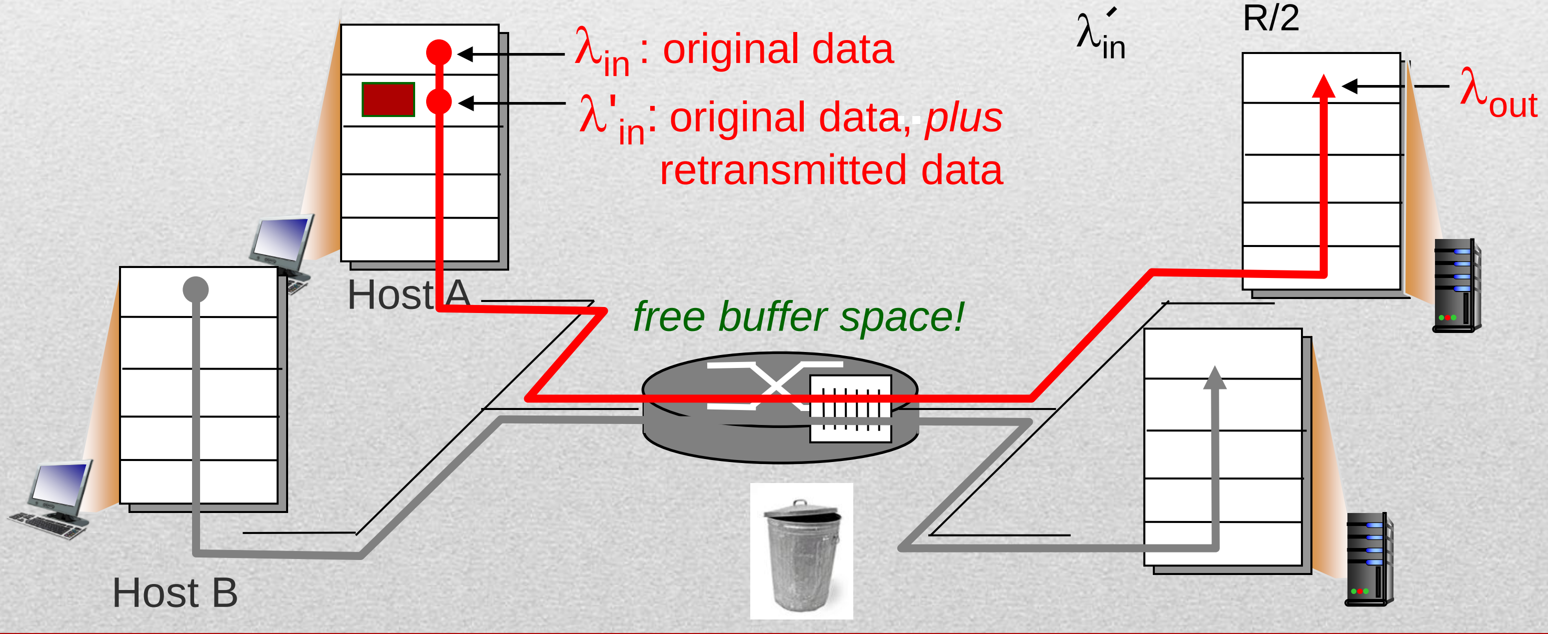


Causes & Costs of Congestion

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

- Sender only resends if packet known to be lost

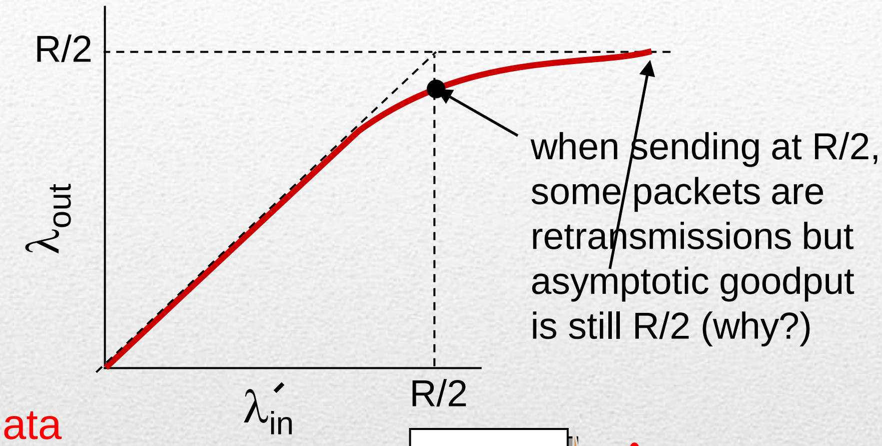


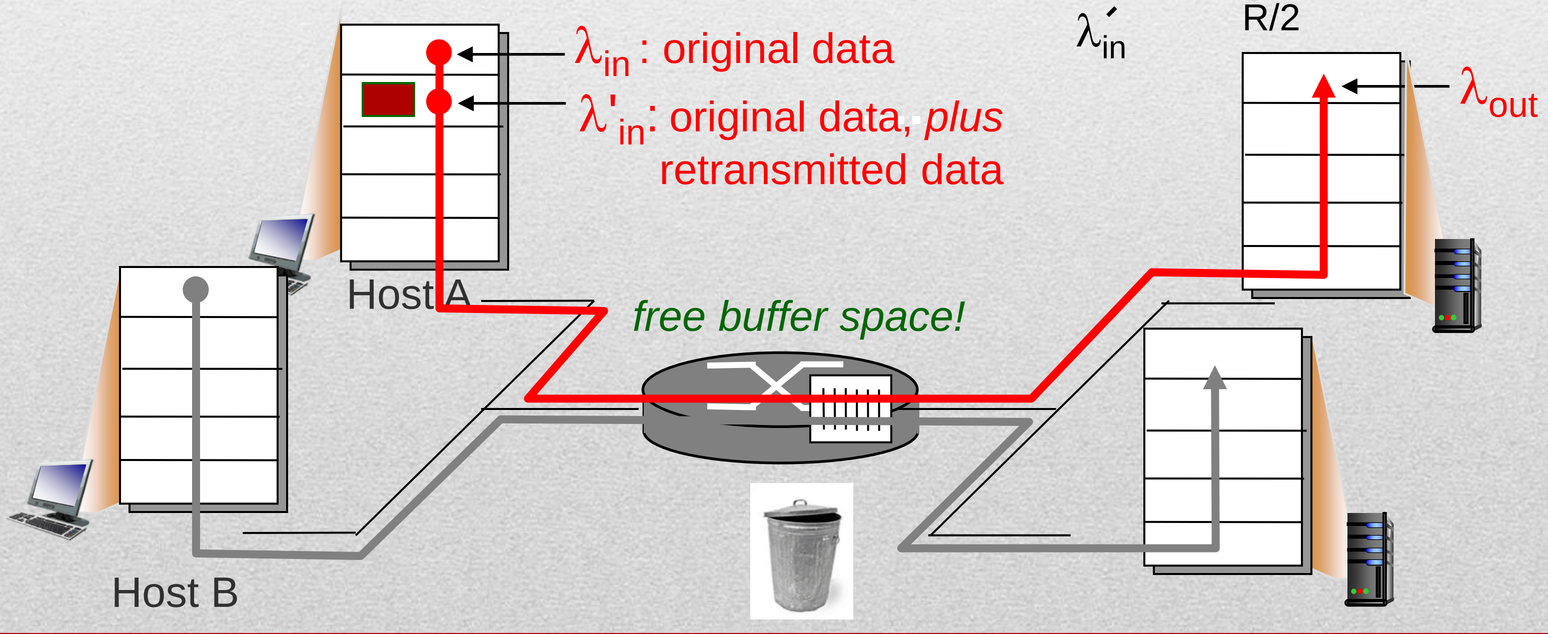


Causes & Costs of Congestion

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

- Sender only resends if packet known to be lost

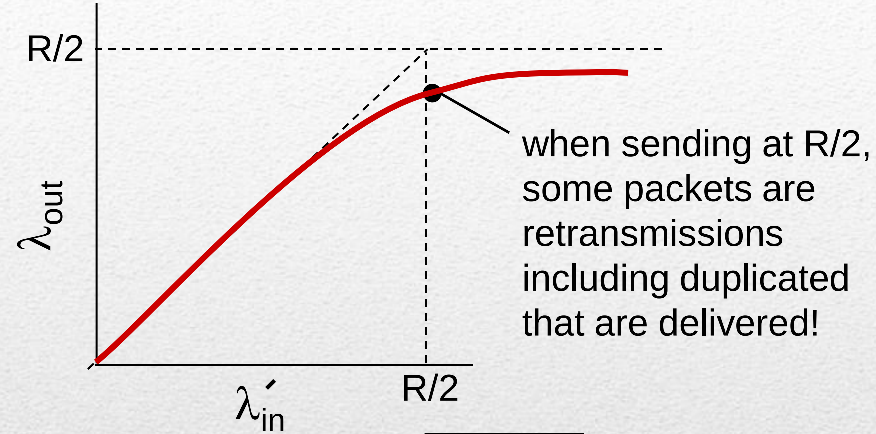


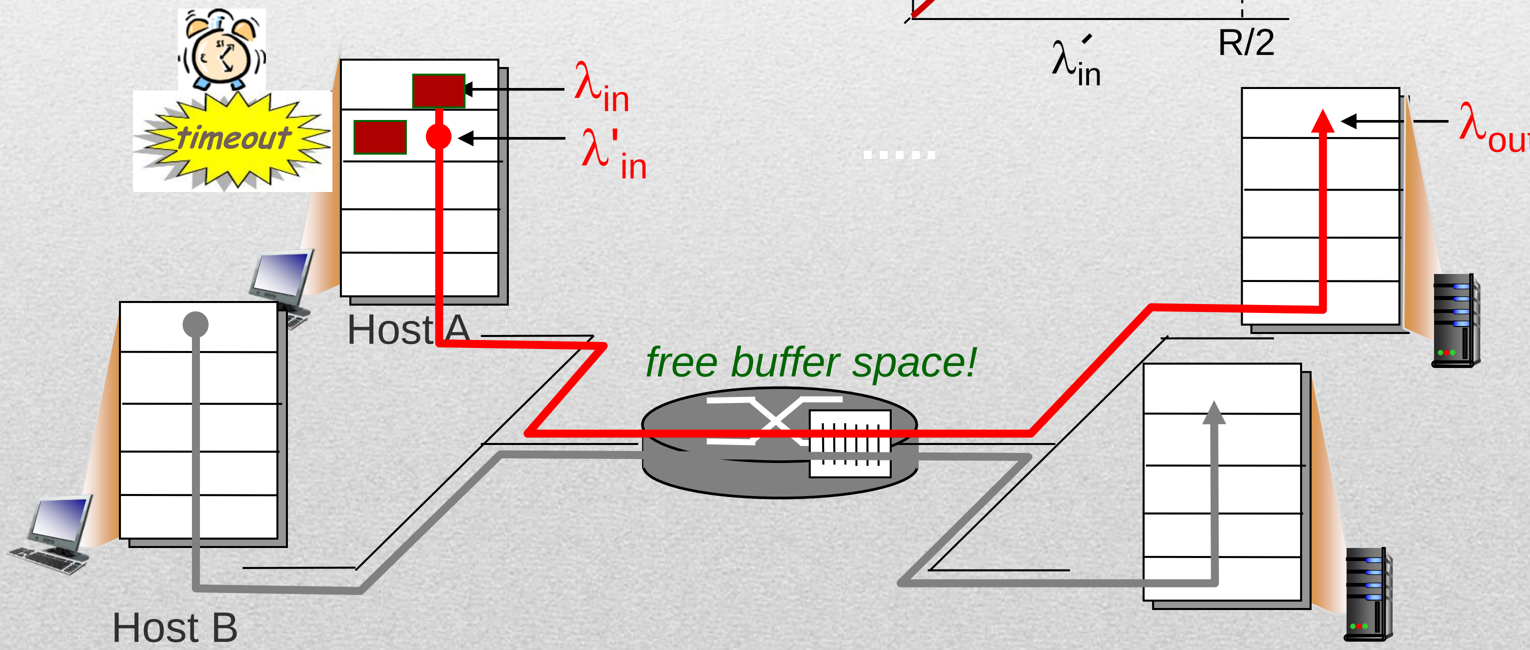


Causes & Costs of Congestion

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

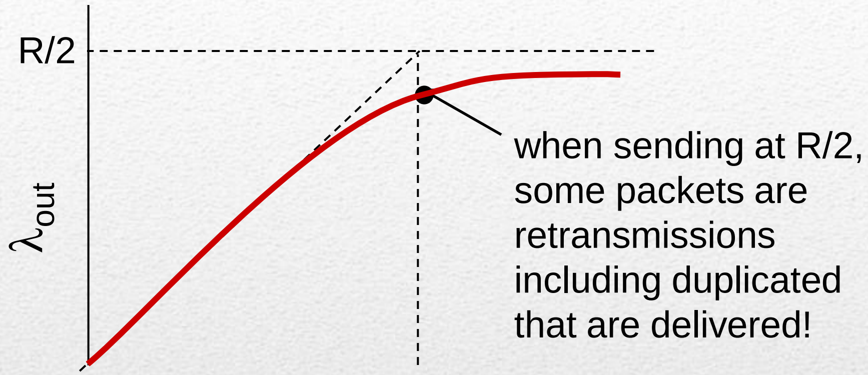




Causes & Costs of Congestion

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 (retransmit) for given "goodput"
- Unneeded retransmissions: link carries multiple copies of packet
  - Decreasing goodput

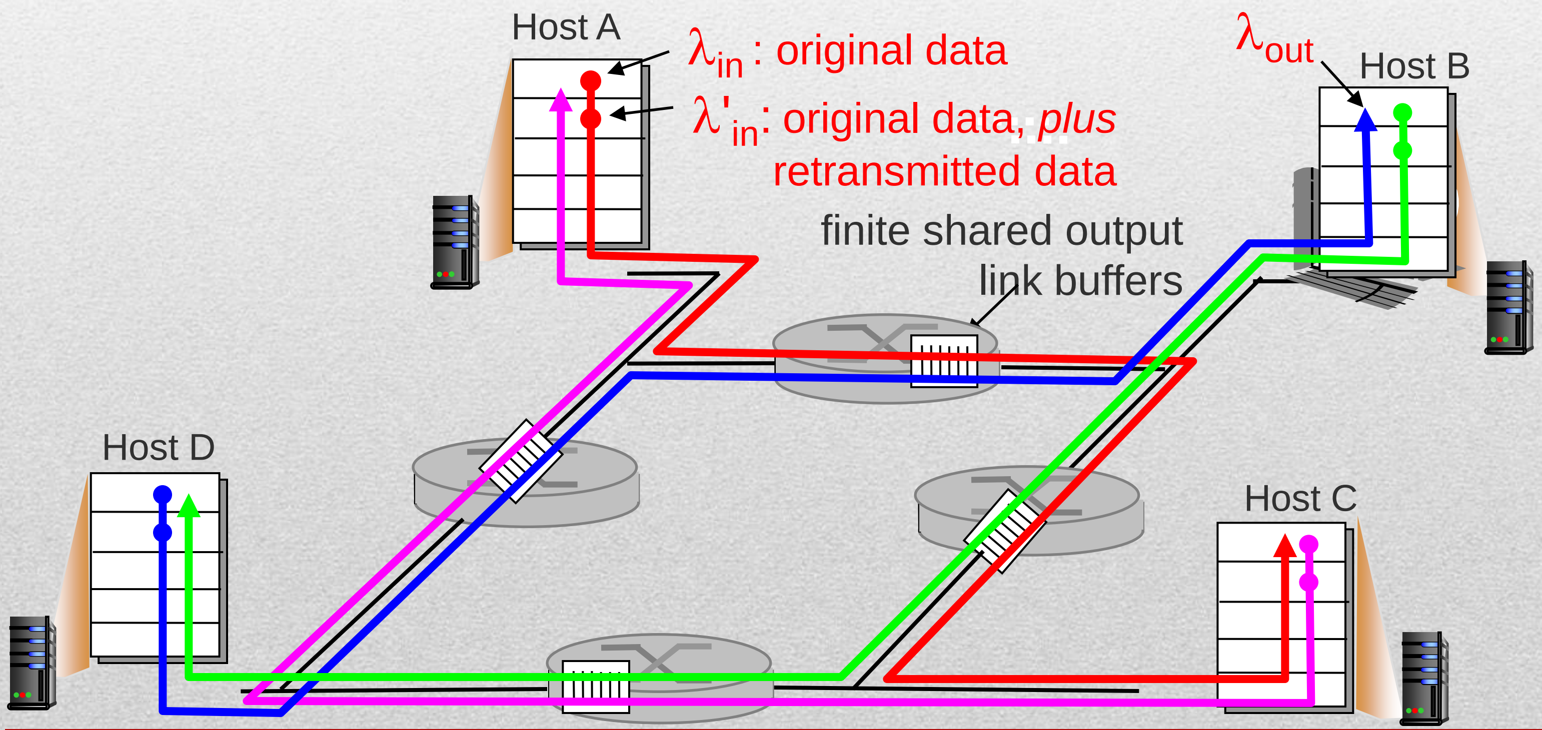
Causes & Costs of Congestion

Scenario (3)

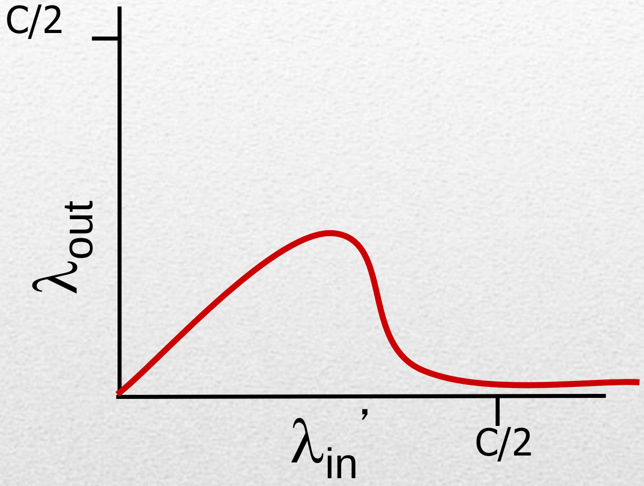
- Four senders
- Multi-hop paths
- Timeout/retransmit

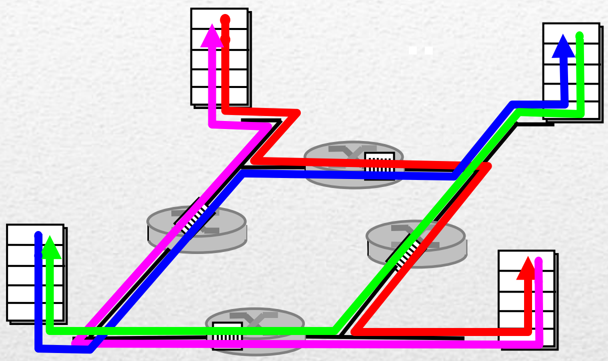
Q: what happens as λ_{in} and λ_{in}' increase ?

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



Causes & Costs of Congestion





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

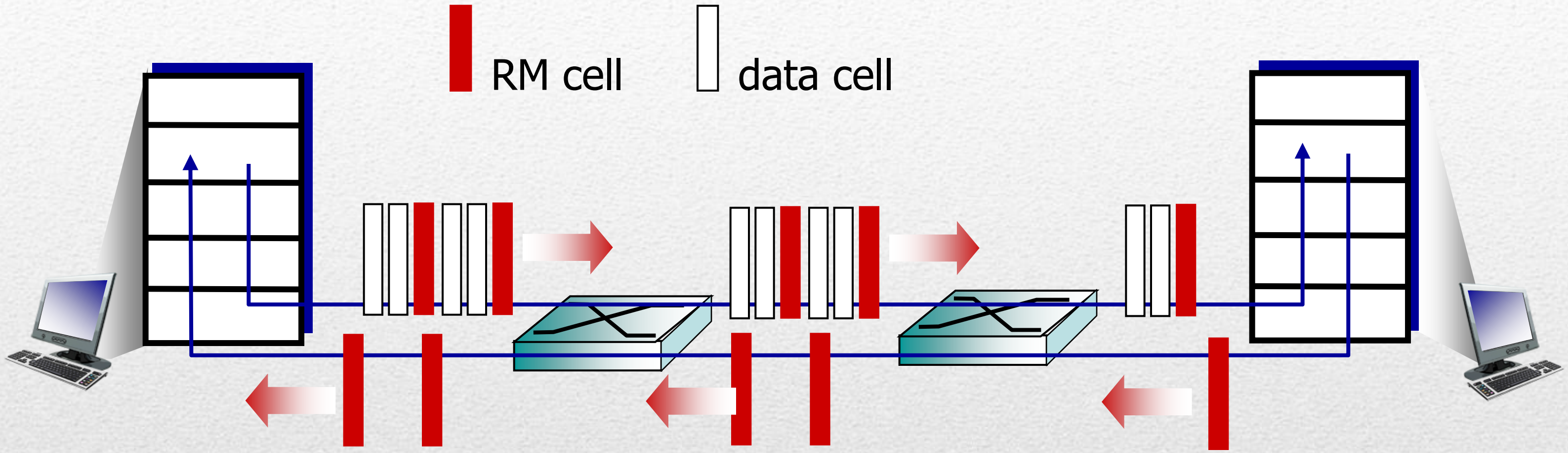
- Routers provide feedback to end systems
  - Single bit indicating congestion (SNA, DECbit, TCP/IP ECN, ATM)
  - Explicit rate for sender to send at

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

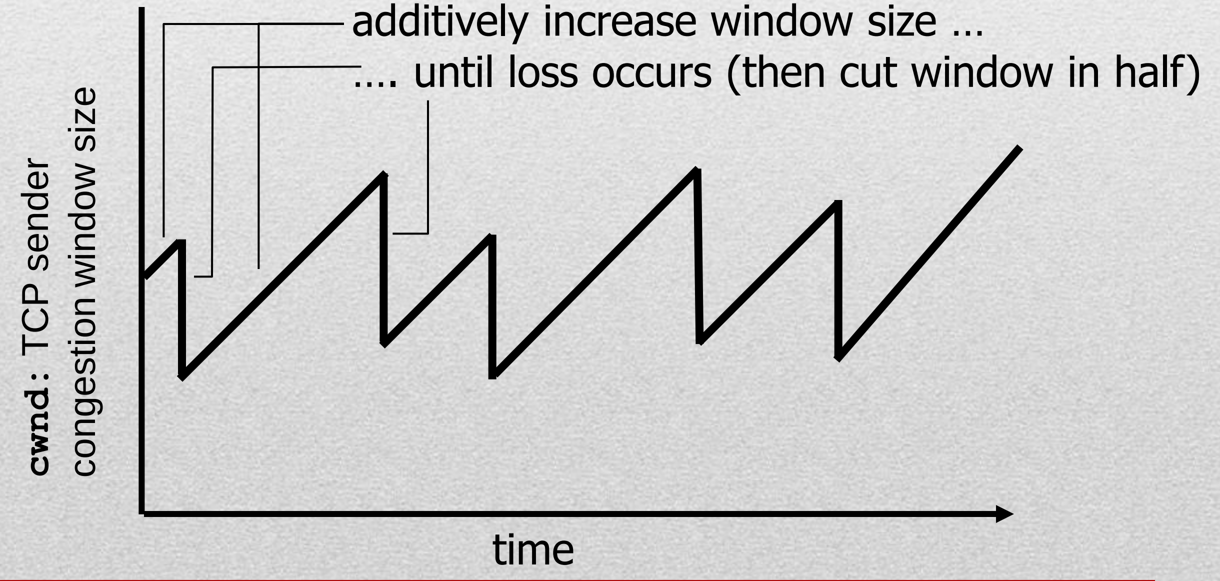
- **EFCI bit in data cells: set to 1 in congested switch**
  - If data cell preceding RM cell has EFCI set, receiver sets CI bit in returned RM cell

TCP Congestion Control

Additive Increase Multiplicative Decrease

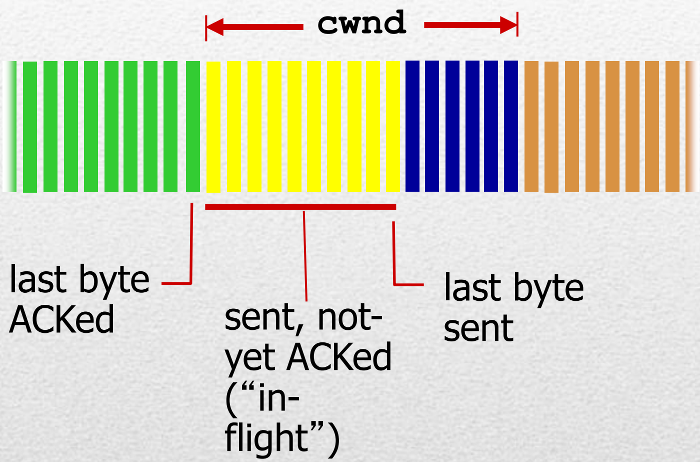
- Approach: sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs
  - Additive increase: increase **`cwnd`** by 1 MSS every RTT until loss detected
  - Multiplicative decrease: cut **`cwnd`** in half after loss

AIMD saw tooth behavior: probing for bandwidth



TCP Congestion Control

sender sequence number space



- Sender limits transmission:

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

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

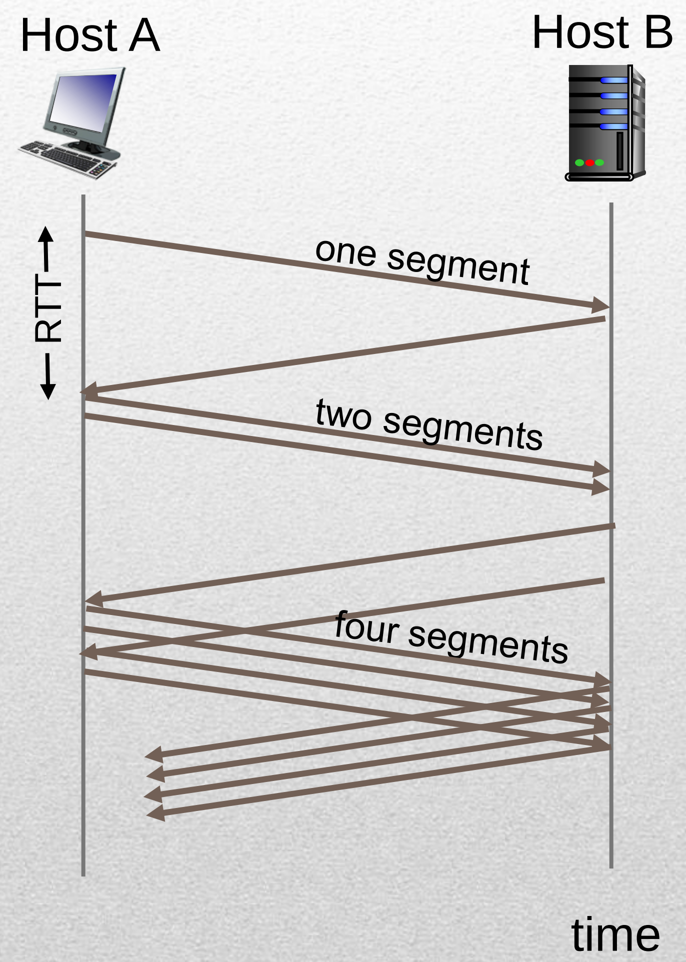
TCP sending rate

- 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



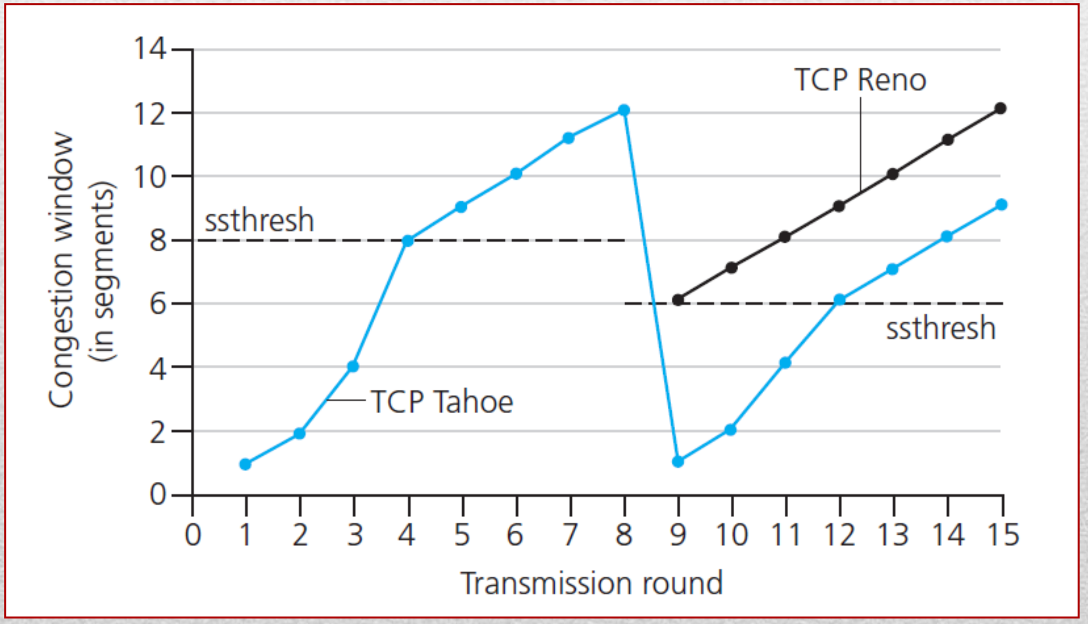
TCP: Detecting & Reacting to Loss

- Loss indicated by timeout
  - **`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)

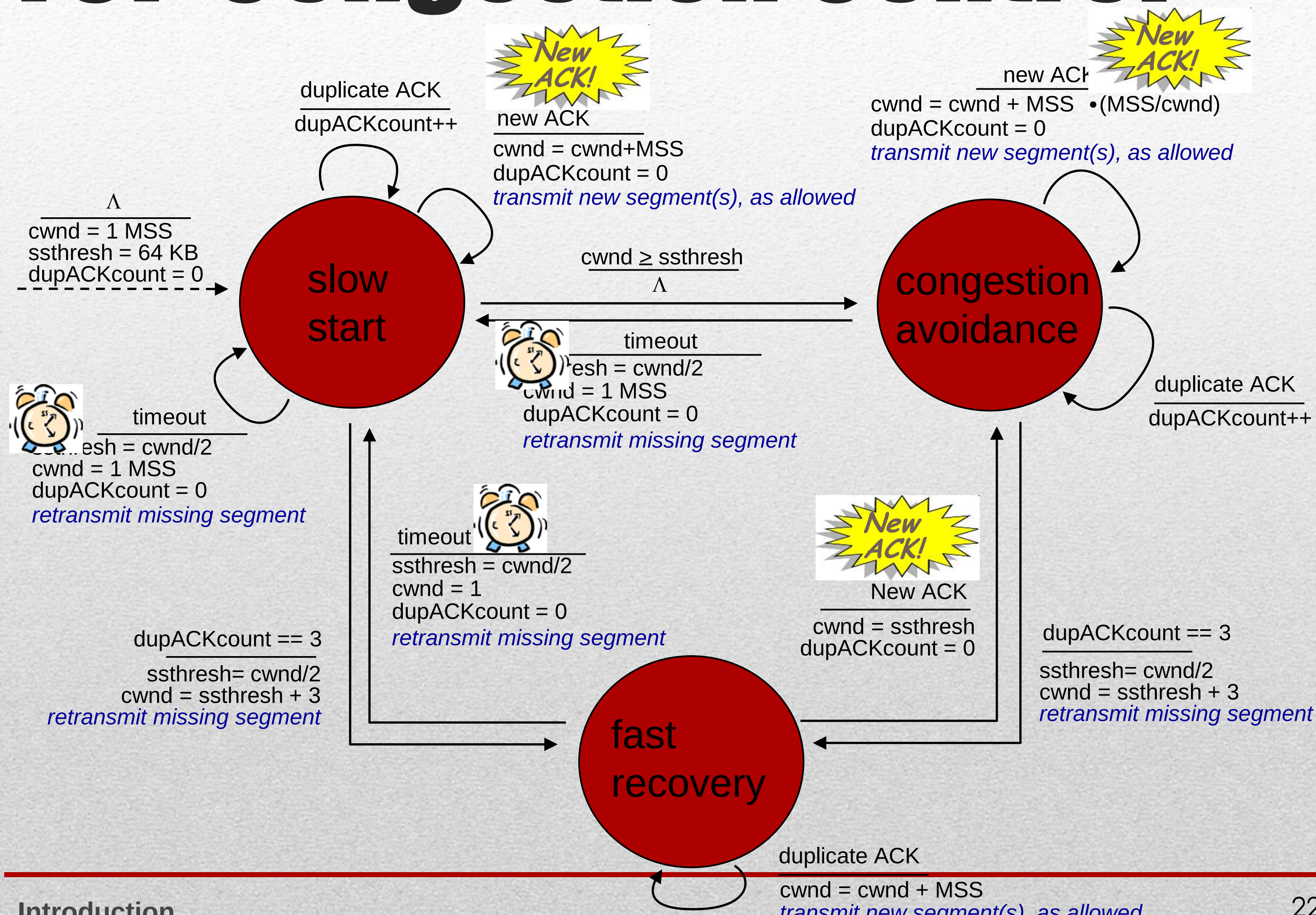
TCP: Switching from Slow Start to CA

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



TCP Congestion Control



TCP Throughput

- Average TCP throughput as function of window size, RTT?
  - Ignore slow start, assume always data to send
- Window size (measured in bytes) where loss occurs
  - Average window size (number of in-flight bytes) is ¾ W
  - Average throughput is 3/4W per RTT

average TCP throughput

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

TCP Throughput 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]:

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

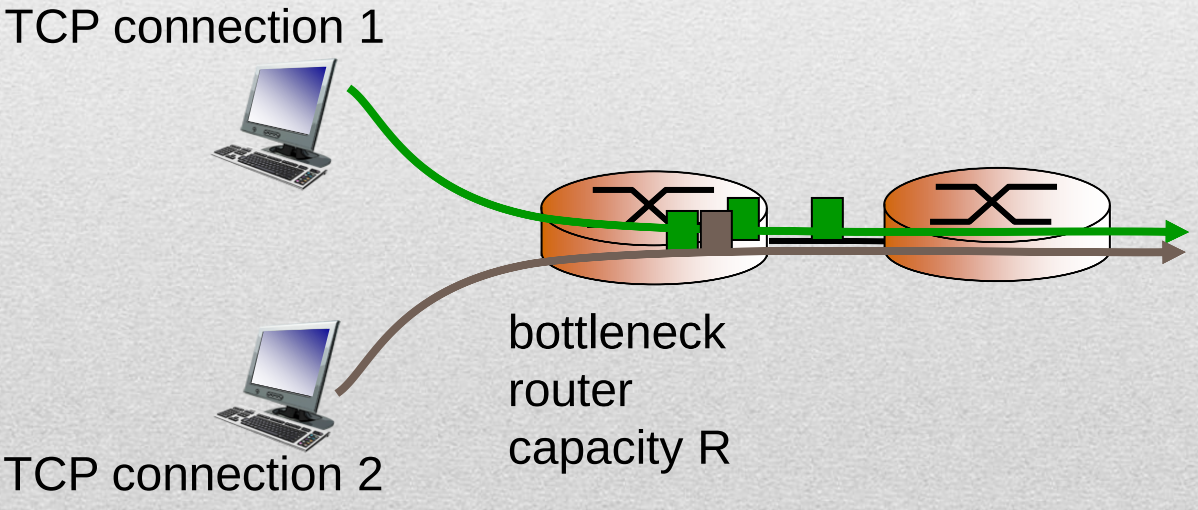
➜ to achieve 10 Gbps throughput, need a loss rate of $L = 2 \cdot 10^{-10}$ *– a very small loss rate!*

- New versions of TCP for high-speed

TCP Fairness

Fairness goal

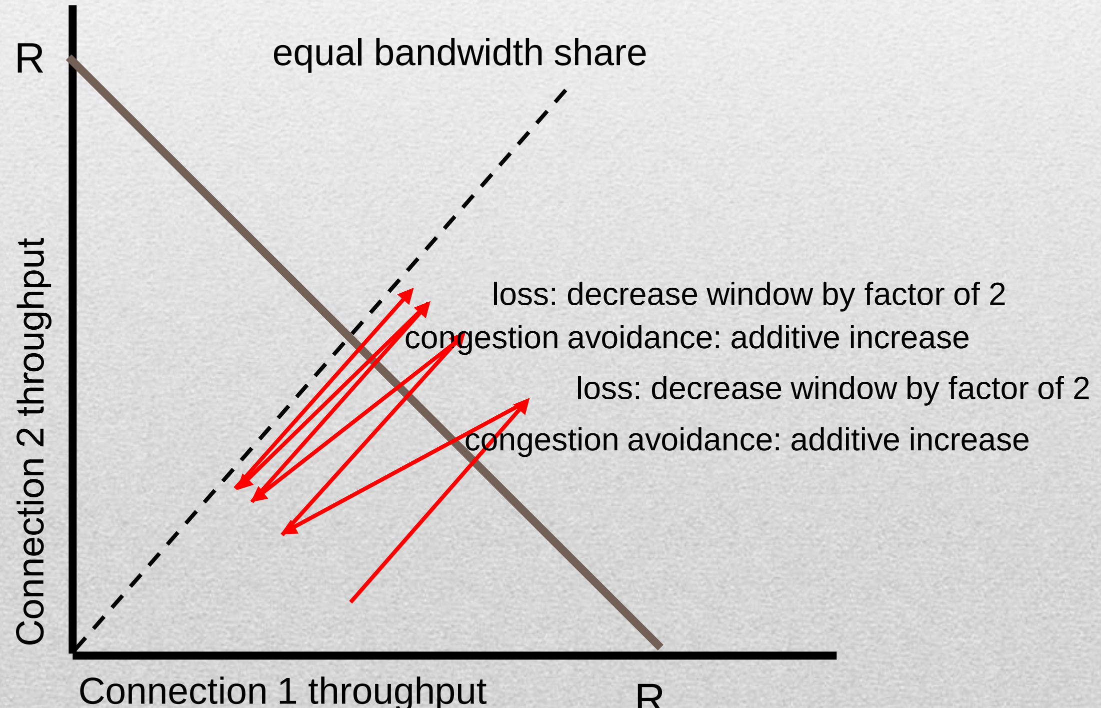
if K TCP sessions share same bottleneck link of bandwidth R, each should have average rate of R/K



Why is TCP fair?

Two competing sessions:

- Additive increase gives slope of 1, as throughout increases
- Multiplicative decrease decreases throughput proportionally



Fairness

- 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 application asks for 1 TCP, gets rate R/10
  - New application asks for 11 TCPs, gets R/2