CS 3251- Computer Networks I: TCP (II)

Professor Patrick Traynor

Lecture 10

9/19/13

Reminders

- Homework 1 has been returned.
    - Check for issues *now*...
- Homework 2 is due in less than 2 weeks.
    - Please start it
- Project 2 is due in exactly 2 weeks.
    - Like Project 1, submitted solely via T-Square, by 5pm EDT.
- Please avail yourself of office hours.
    - I like meeting and chatting with you all. Feel free to stop by when you have a question, or if you want to talk about an advanced topic!

Review

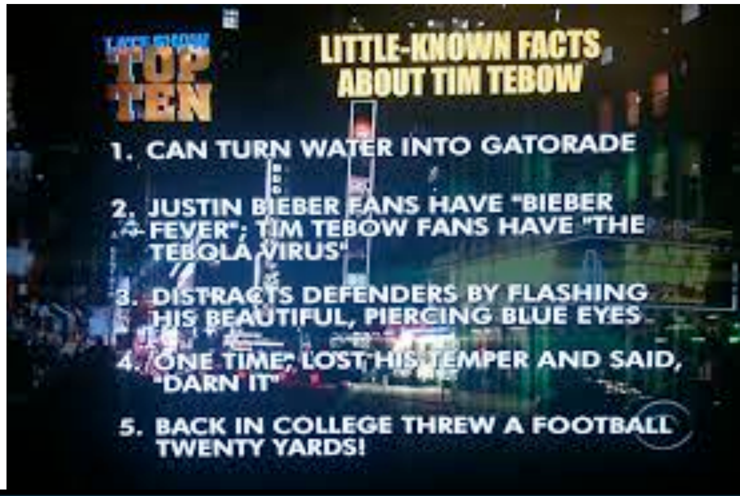
- What are the six flags in the TCP header?
  - What do they do?
- Why does TCP use an exponential weighted moving average for estimated RTT?
- What does the sequence number correspond to in a TCP packet? The Acknowledgement number?
- What is fast retransmit?

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

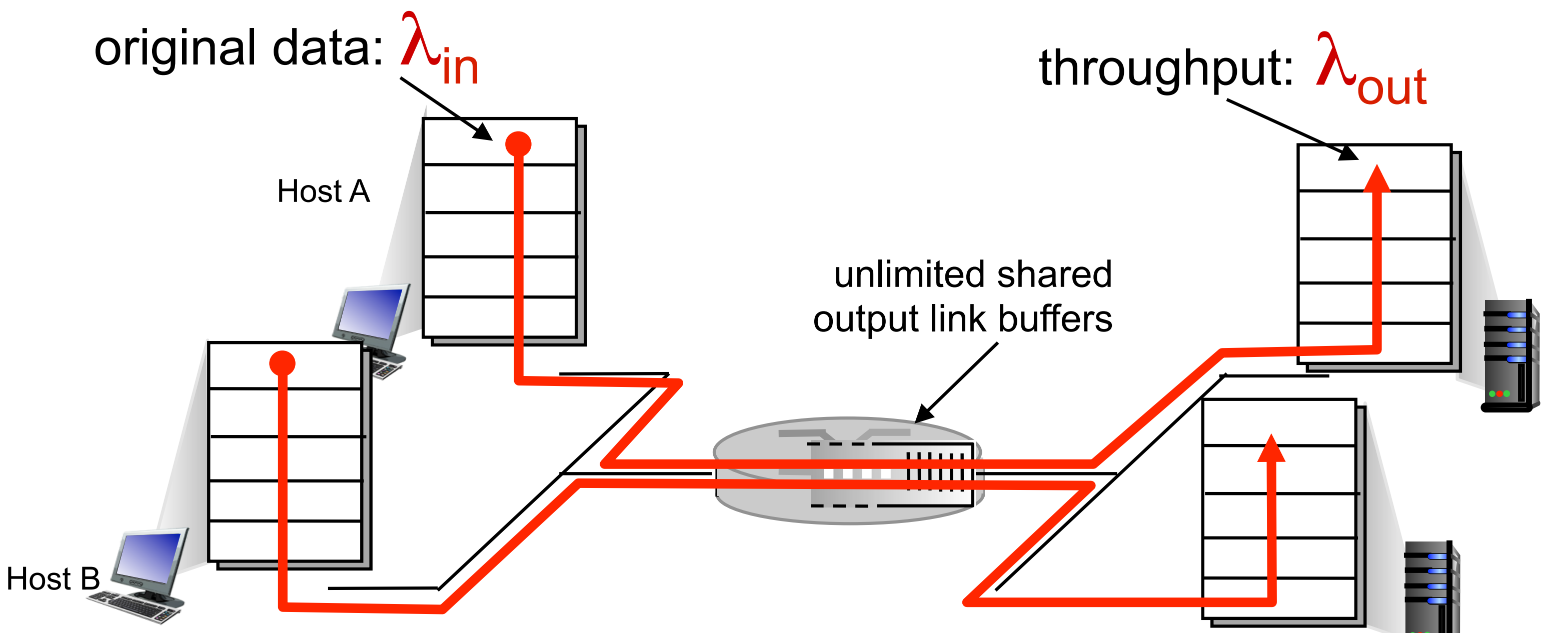
Principles of Congestion Control

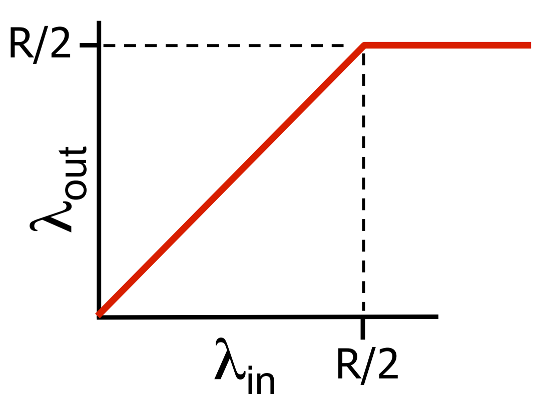
- Informally: "too many sources sending too much data too fast for *network* to handle"
  - different from flow control!
- manifestations:
  - lost packets (buffer overfill at routers)
  - long delays (queueing in router buffers)
- a top-10 problem!

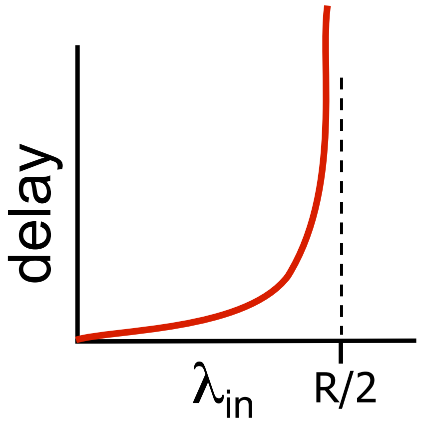


Causes/costs of congestion: Scenario 1

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

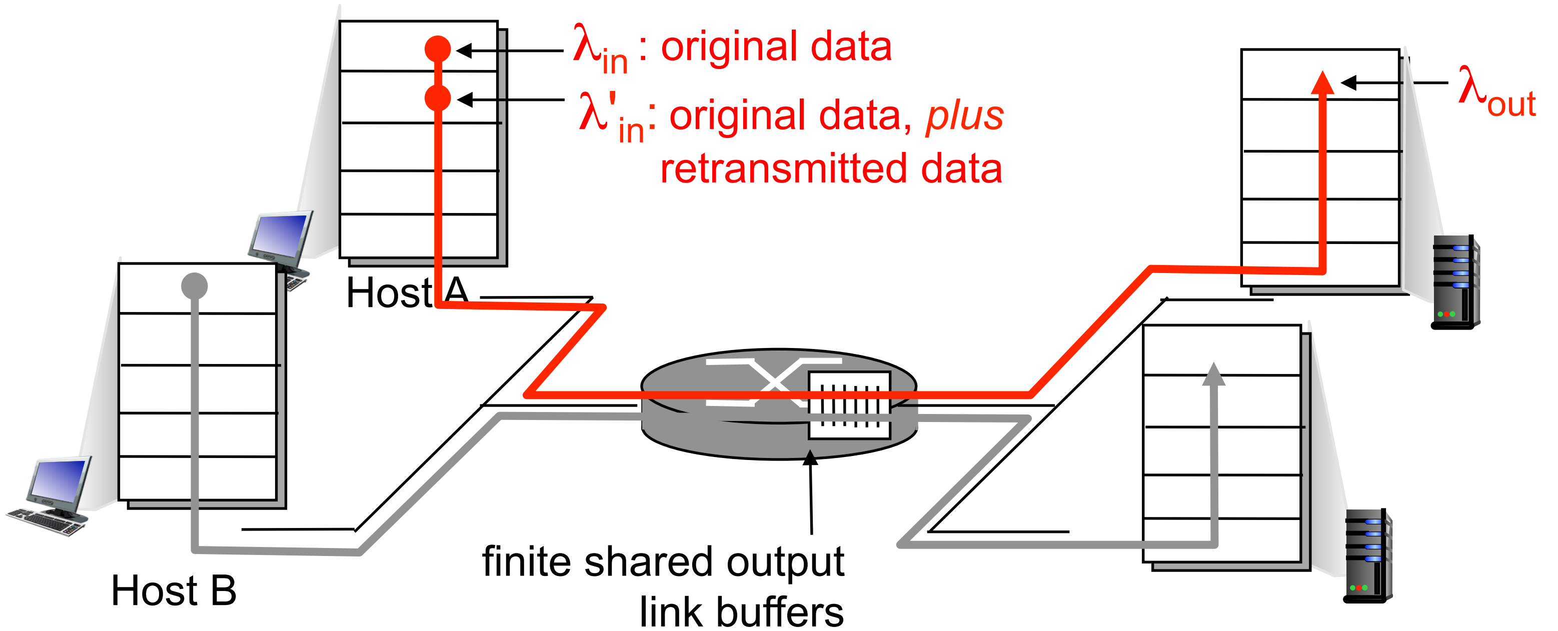






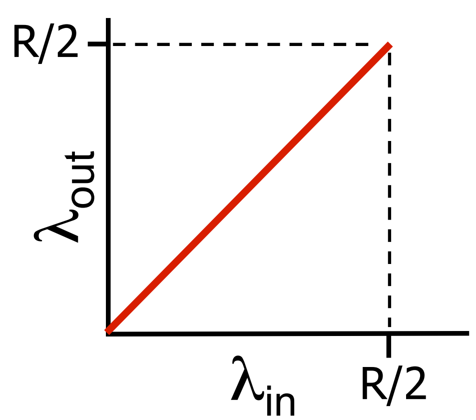
Causes/costs of congestion: Scenario 2

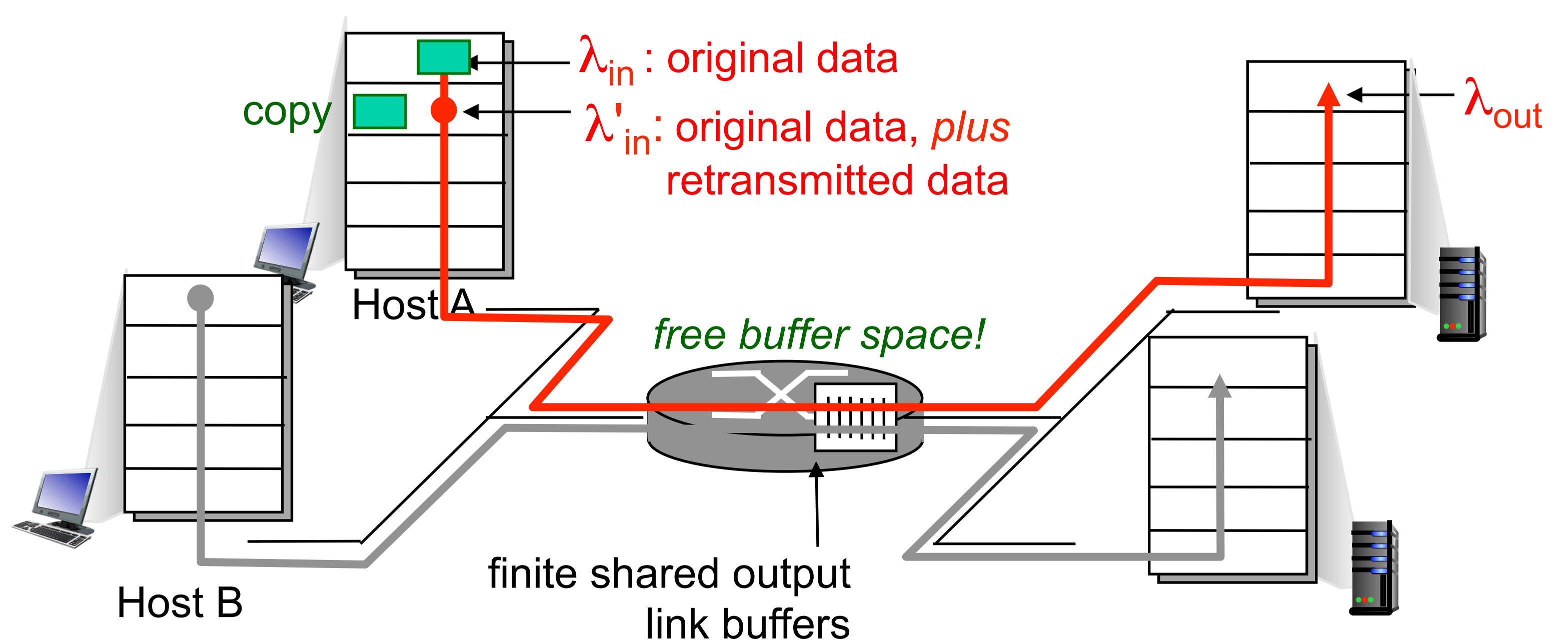
- one router, *finite* buffers
- sender retransmission of timed-out packet
  - application-layer input = application-layer output:
  - $\lambda_{in} = \lambda_{out}$
- transport-layer input includes retransmissions:
  - $\lambda_{in}' \geq \lambda_{in}$



Causes/costs of congestion: Scenario 2

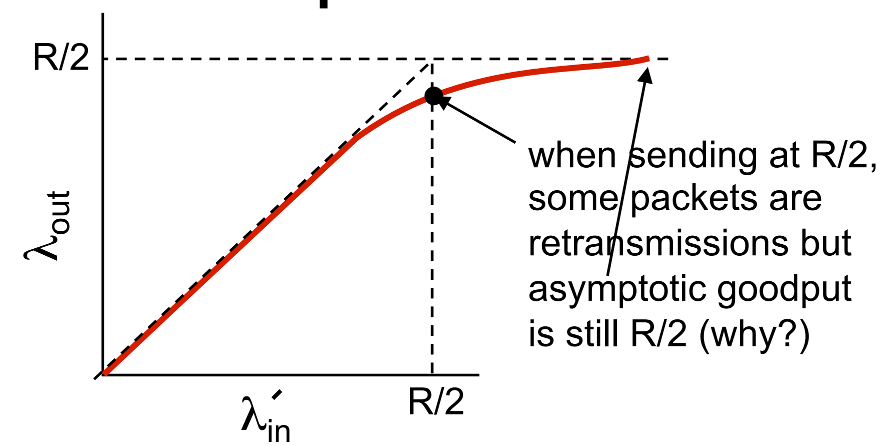
- Idealization: *perfect knowledge*
  - sender sends only when router buffers available

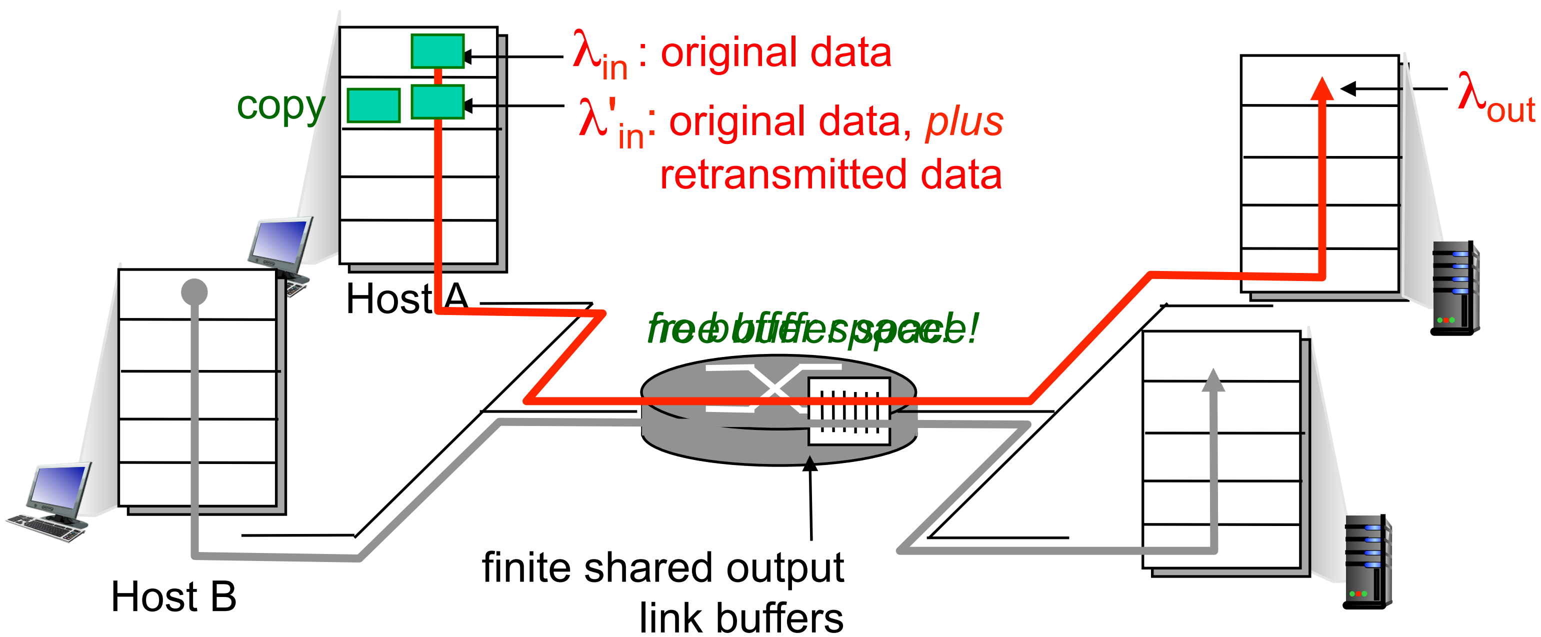




Causes/costs of congestion: Scenario 2

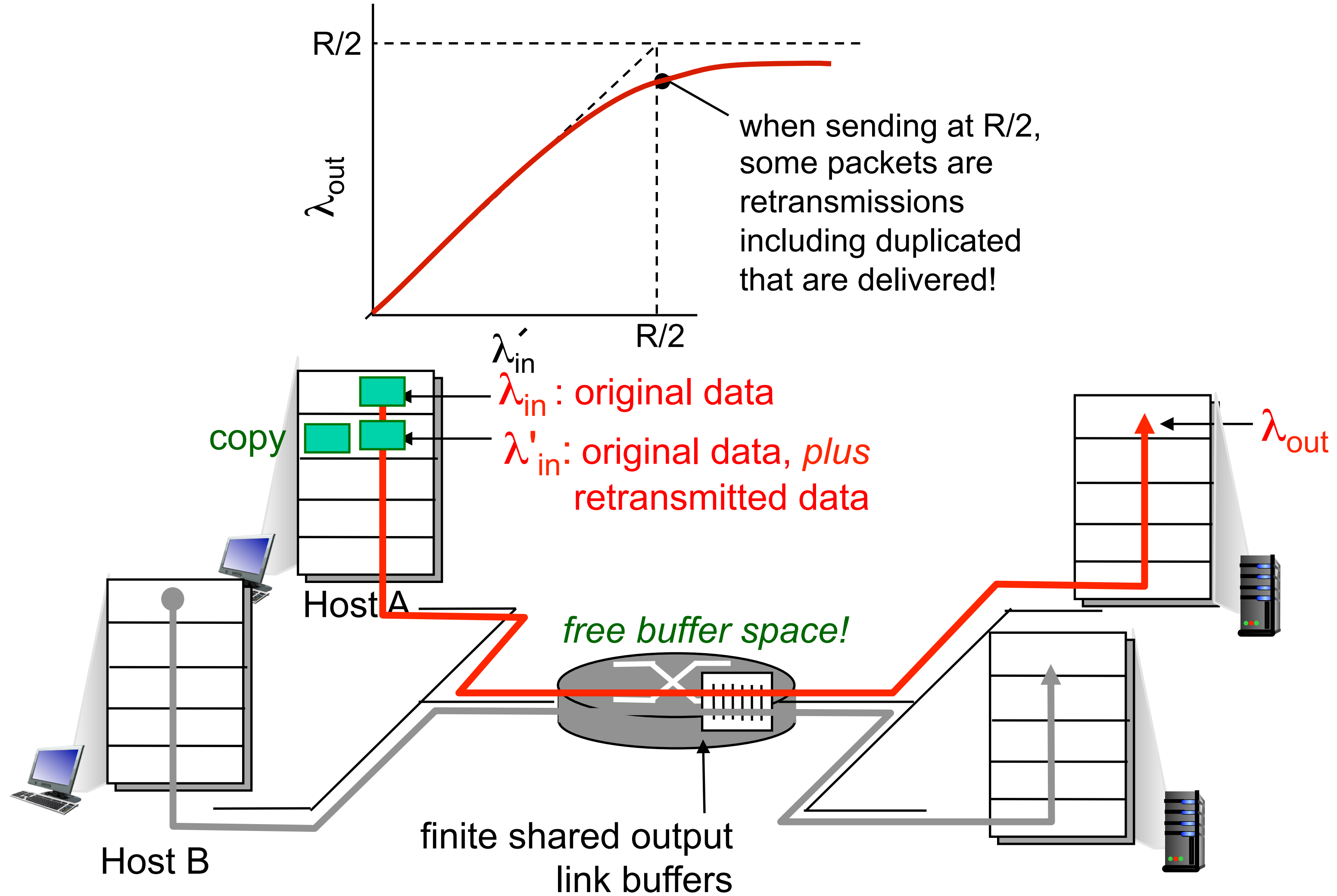
- Idealization: *known loss*
  - packets can be lost, dropped at router due to full buffers
  - sender only resends if packet known to be lost





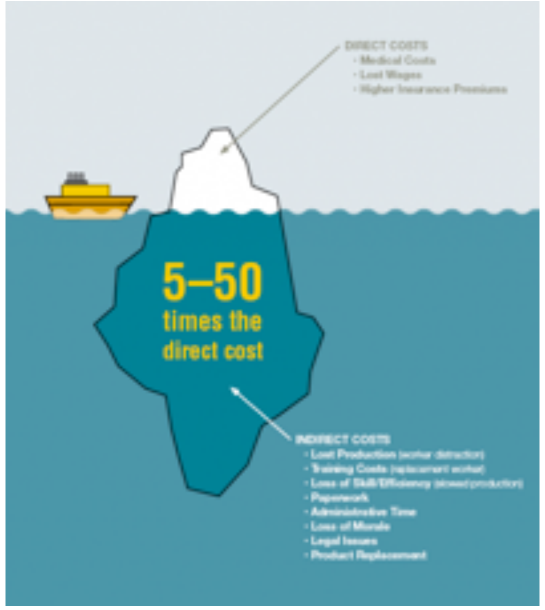
Causes/costs of congestion: Scenario 2

- 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



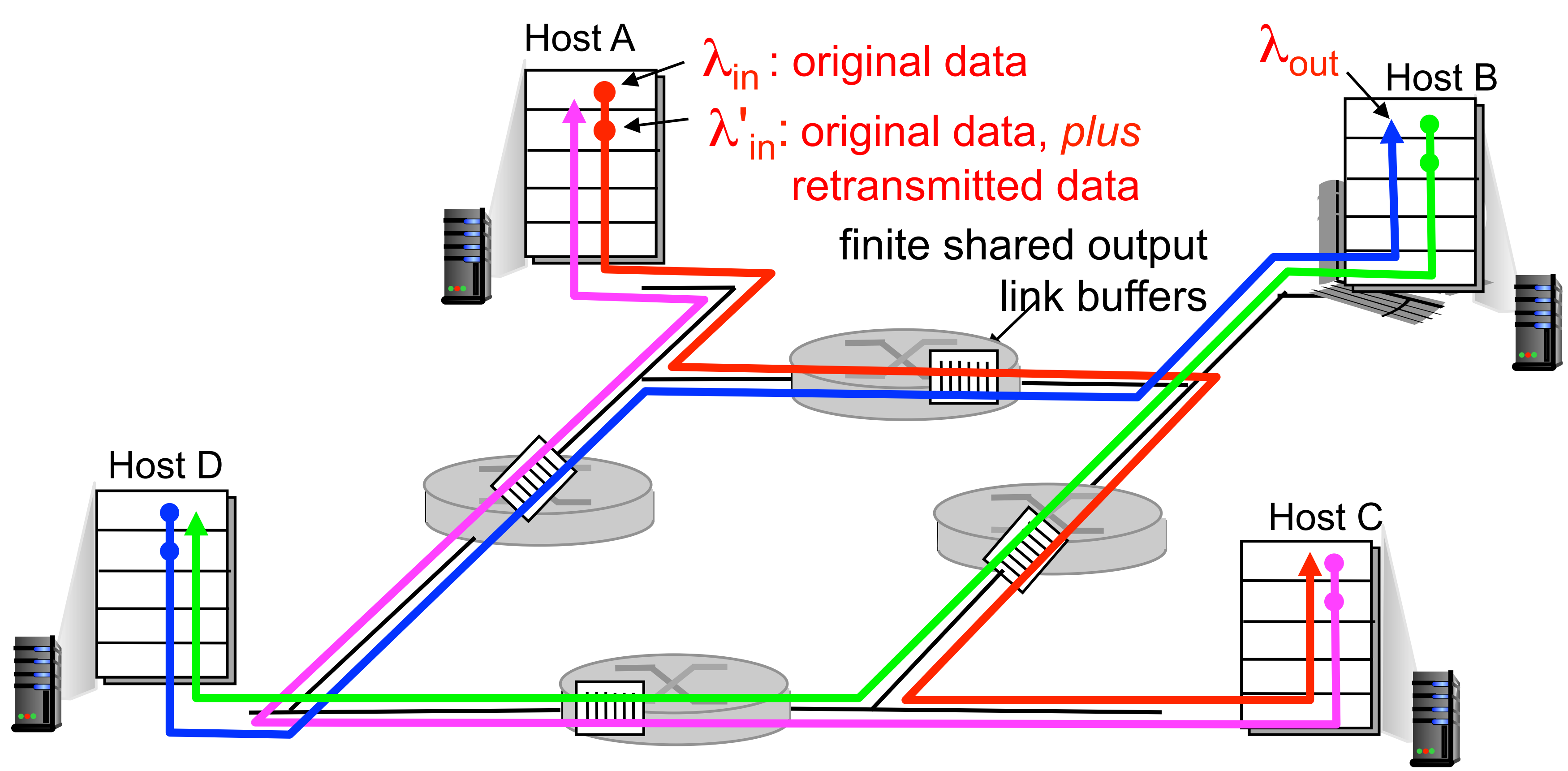
Causes/costs of congestion: Scenario 2

- 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



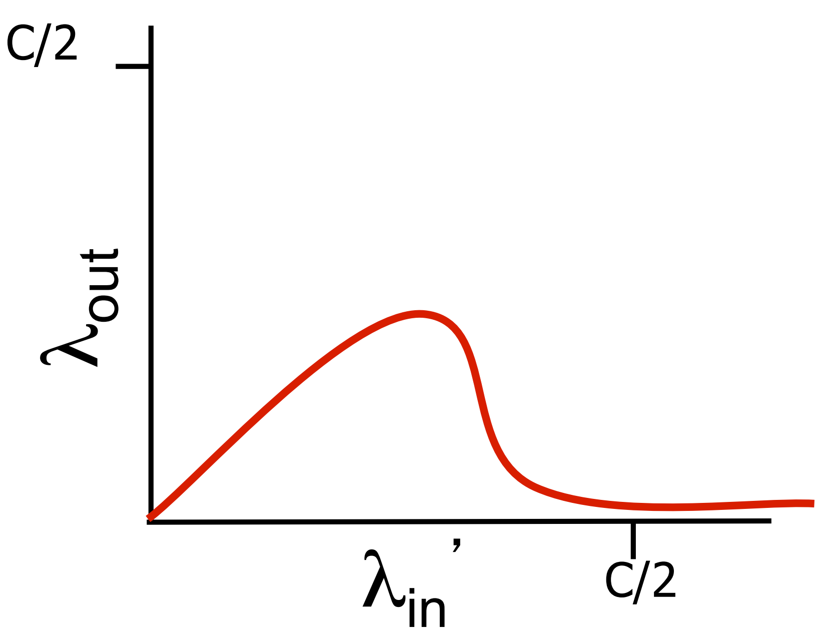
Causes/costs of congestion: Scenario 3

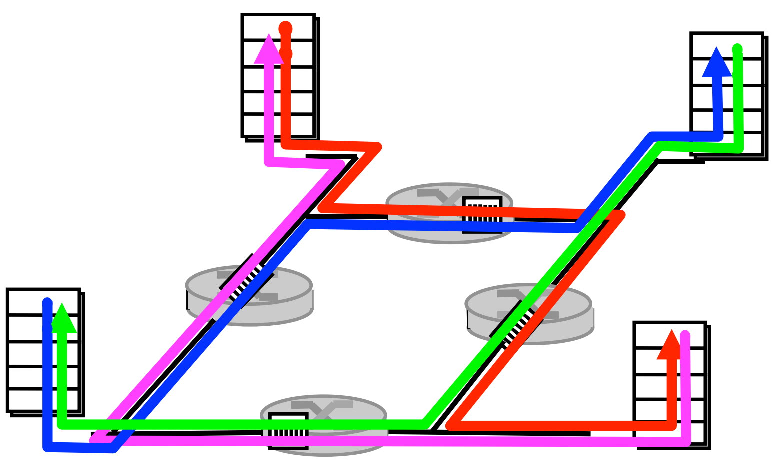
- Scenario: four senders, multihop paths, timeout/retransmit
- *Q*: What happens as λ_{in} and λ_{in}' increase ?
- *A*: as red λ_{in}' increases, all arriving blue pkts at upper queue are dropped, blue throughput $\rightarrow$ 0



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:

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

End-end congestion control

• no explicit feedback from network

• congestion inferred from end-system observed loss, delay

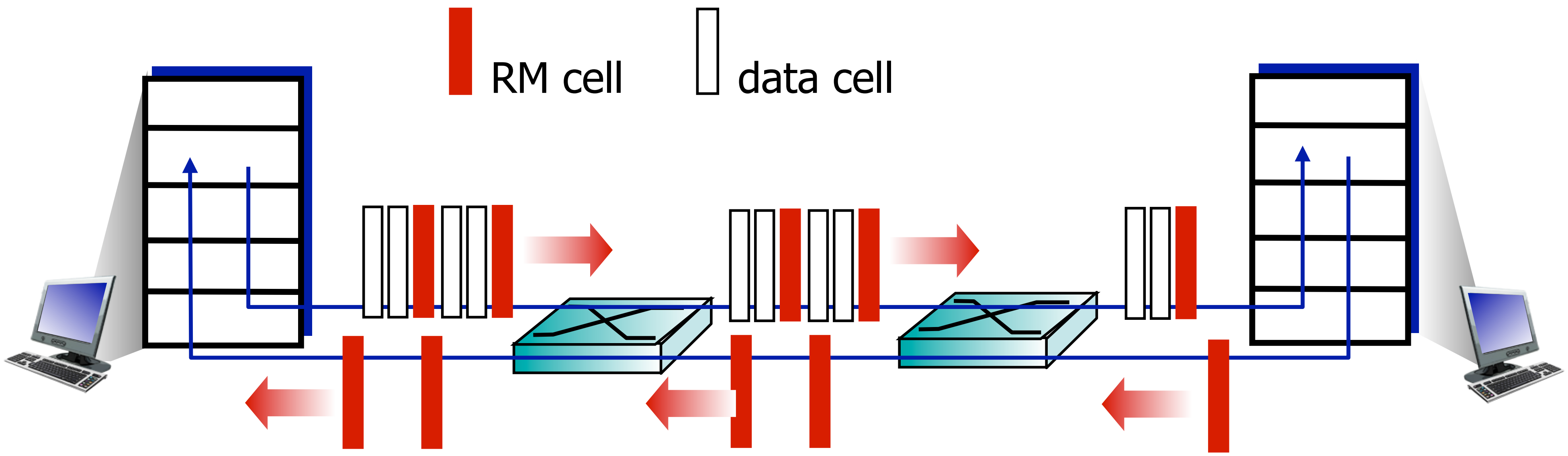
• approach taken by TCP

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



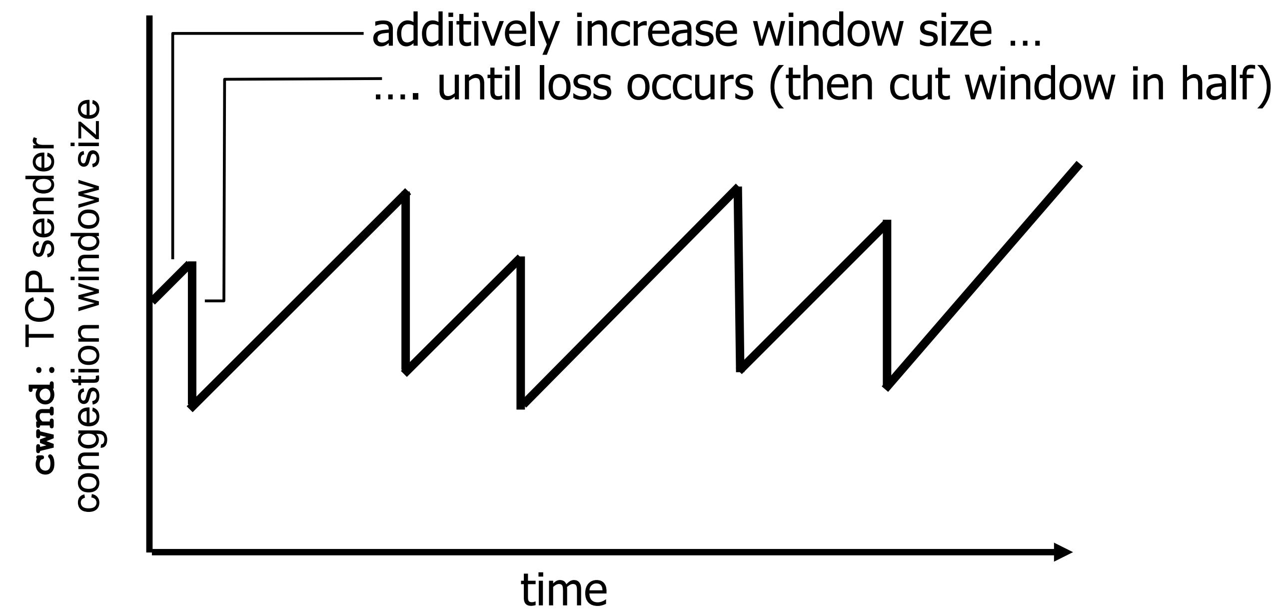
- two-byte ER (explicit rate) field in RM cell
  - congested switch may lower ER value in cell
  - senders' send rate thus max supportable rate on path
- EFCI bit in data cells: set to 1 in congested switch
  - if data cell preceding RM cell has EFCI set, receiver sets CI bit in returned RM cell

Chapter 3 outline

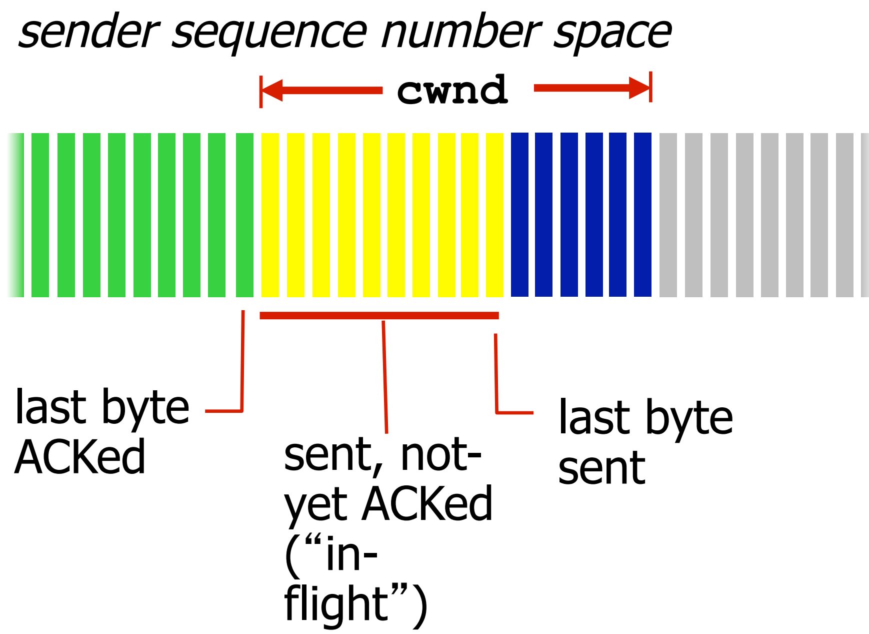
- 3.1 Transport-layer services
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Principles of reliable data transfer
- 3.5 Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- 3.6 Principles of congestion control
- 3.7 TCP congestion control

TCP Congestion Control

- **A**dditive **I**ncrease, **M**ultiplicative **D**ecrease (AIMD)
  - 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



TCP Congestion Control: Details

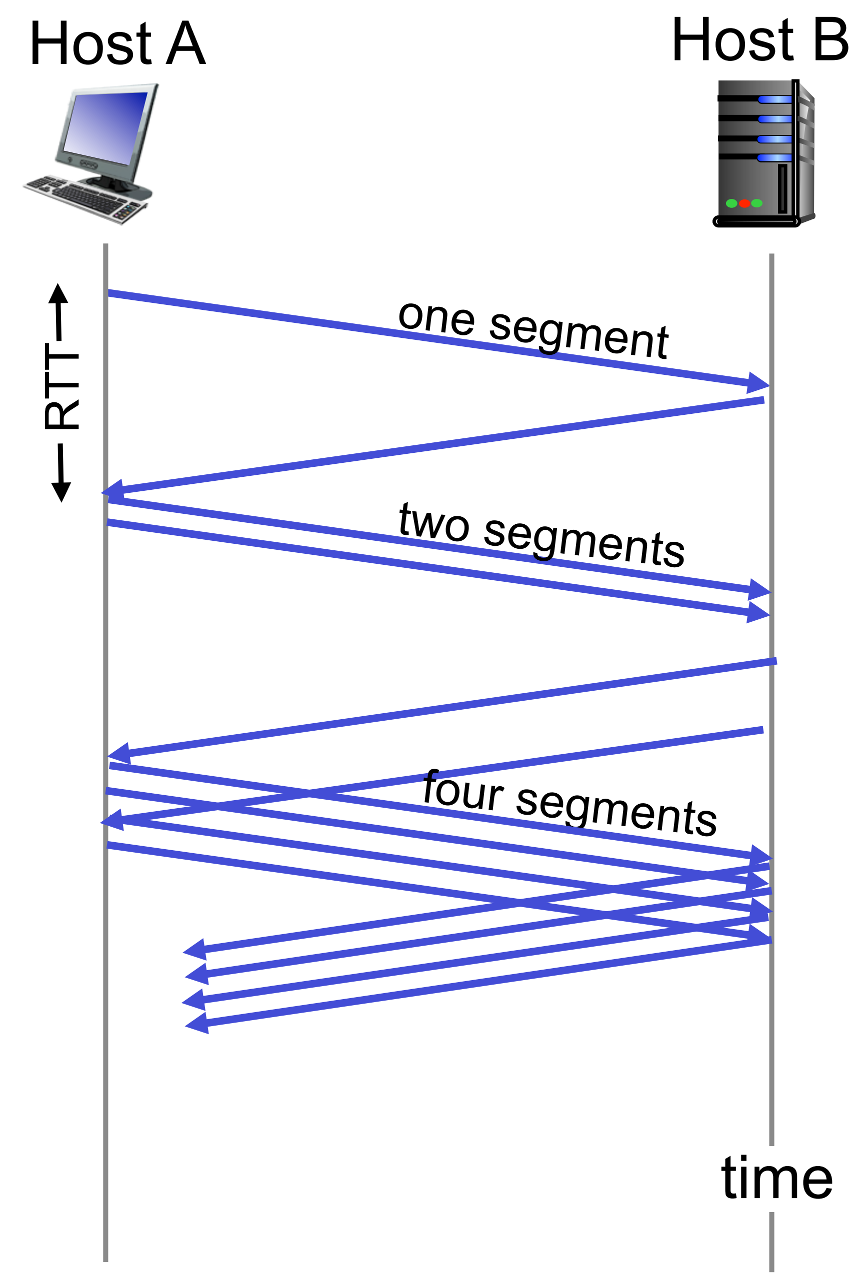


- sender limits transmission: `LastByteSent- LastByteAcked` ≤ `cwnd`
- **`cwnd`** is dynamic, function of perceived network congestion
- TCP sending rate:
  - roughly: send cwnd bytes, wait RTT for ACKS, send more bytes

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

TCP Slow Start

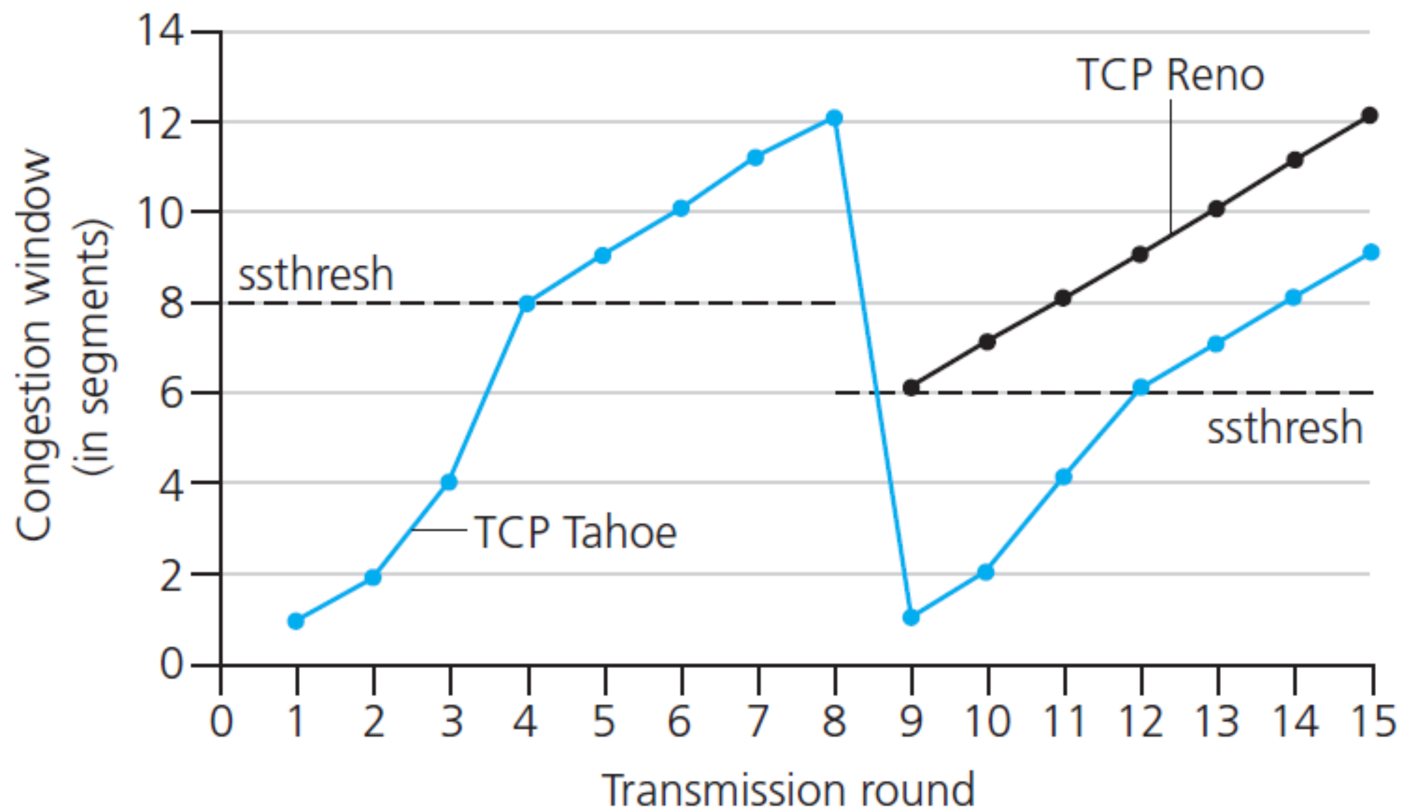
- When connection begins, increase rate exponentially until first loss event:
  - initially **`cwnd`** = 1 MSS
  - double **`cwnd`** every RTT
  - done by incrementing cwnd for every ACK received
- *Summary*: initial rate is slow but ramps up exponentially fast



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)

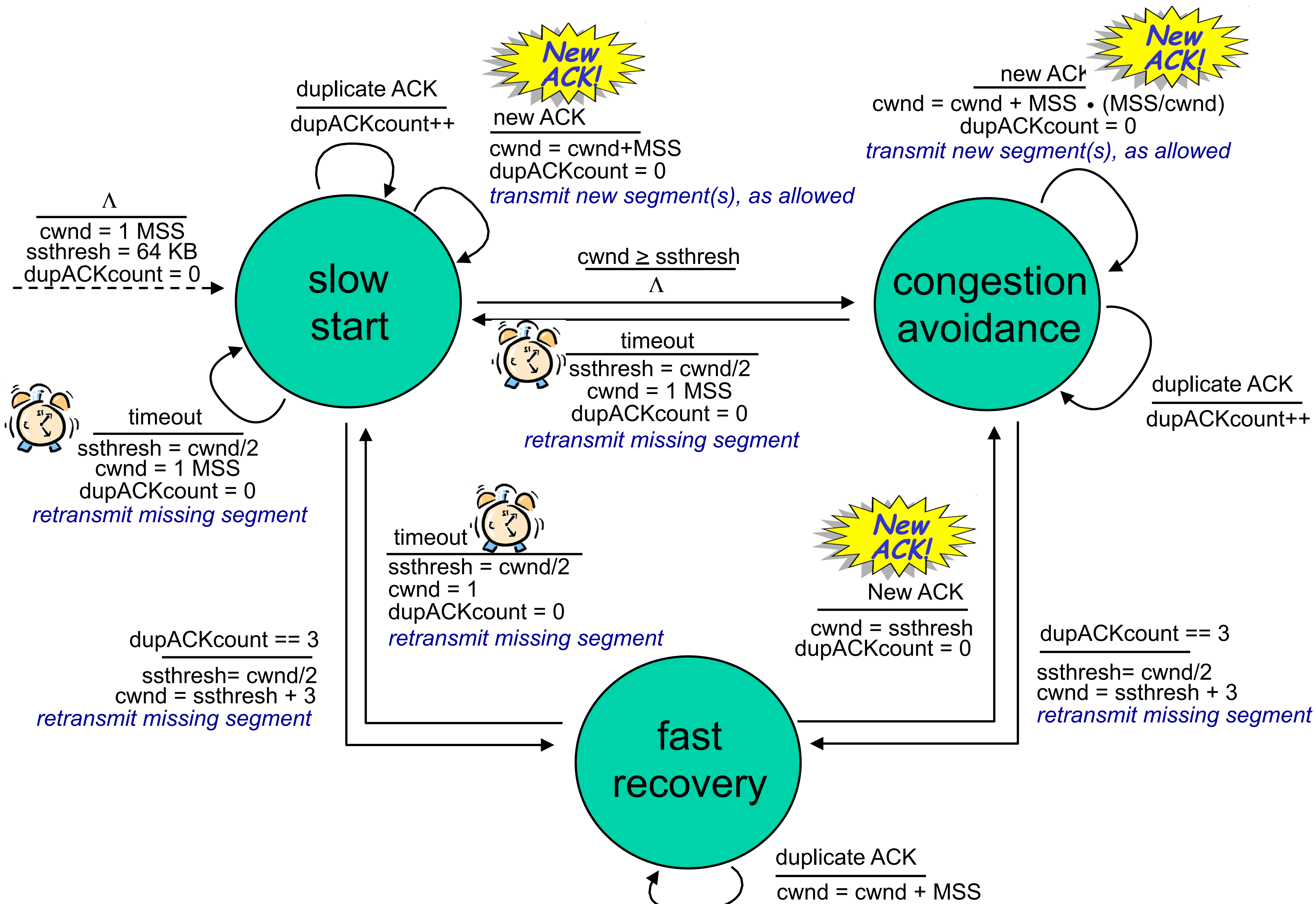
TCP: Switching from Slow Start to CA



- *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

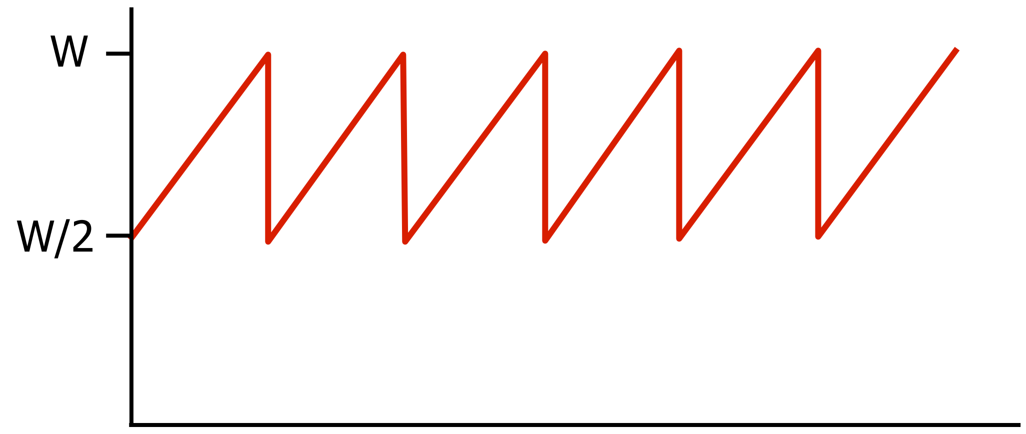
Summary: TCP Congestion Control



TCP throughput

- avg. TCP 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

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



Fine... but WHY?

- This saw-toothed pattern is "normal" TCP behavior.
- Is that good or bad?
  - Are there lost opportunities?

TCP Futures: TCP over "long, fat pipes"

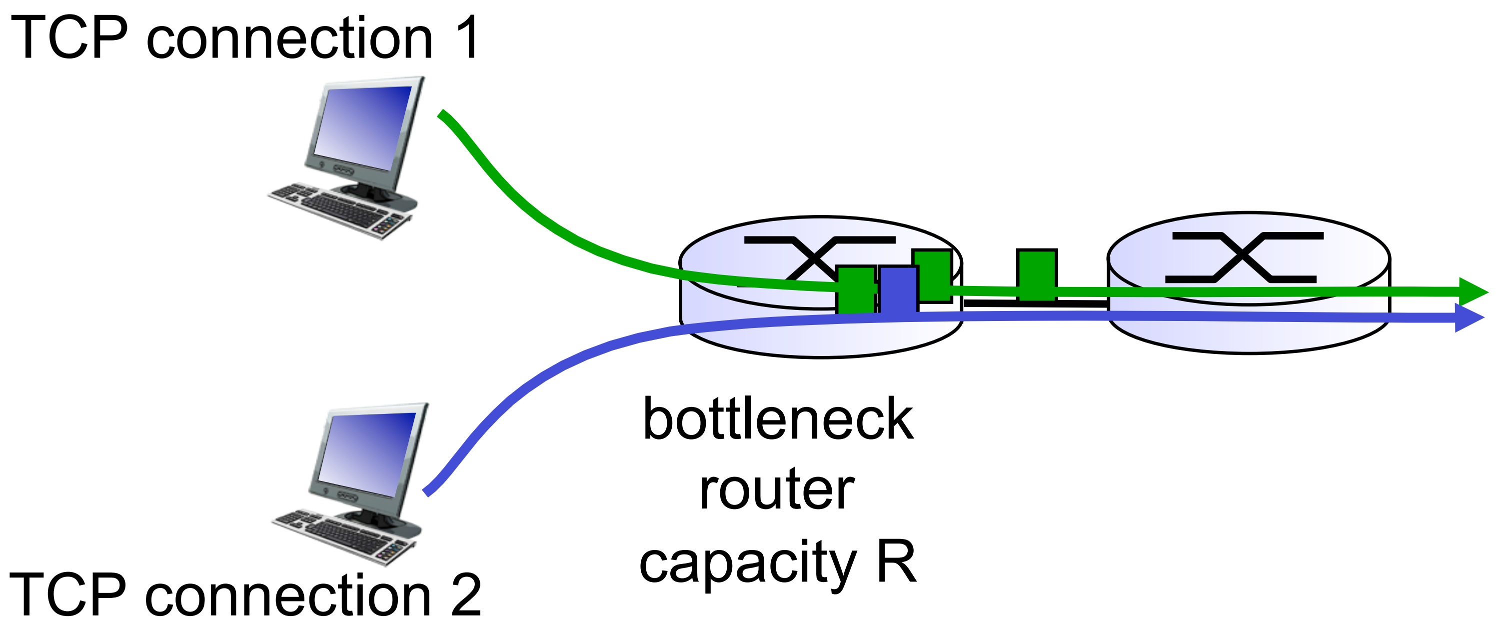
- example: 1500 byte segments, 100ms RTT, want 10 Gbps throughput
- requires W = 83,333 in-flight segments
- throughput in terms of segment loss probability, L [Mathis 1997]:

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

- ➜ 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

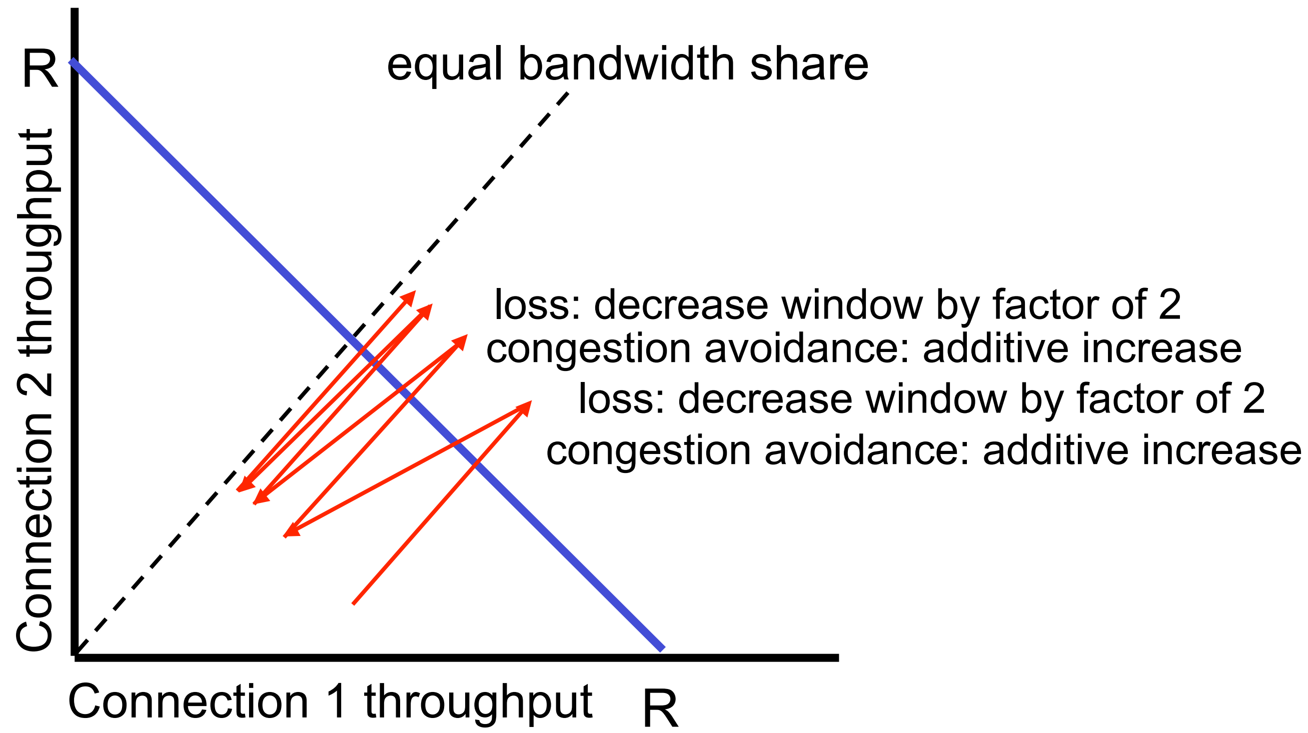
TCP Fairness

- *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 throughout increases
  - multiplicative decrease decreases throughput proportionally



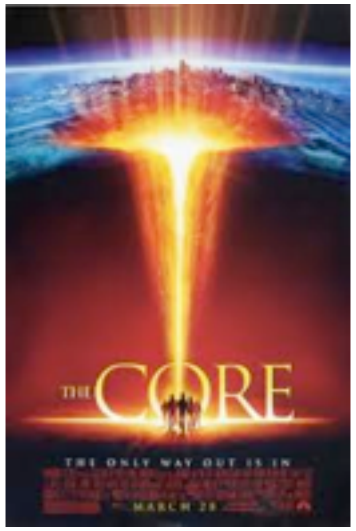
Fairness (more)

- 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
  - 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"

Wrap-Up

- Look at your homework, project and get started!
- Read Sections 4.1 - 4.3