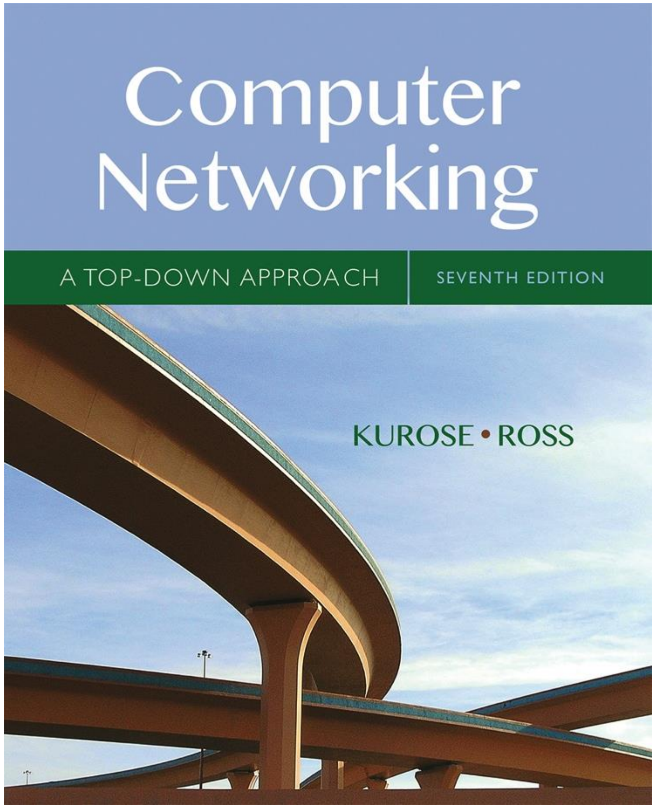
Chapter 3 Transport Layer Part 5/5

A note on the use of these Powerpoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR



Computer Networking: A Top Down Approach

7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

Chapter 3 outline

3.1 transport-layer services

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

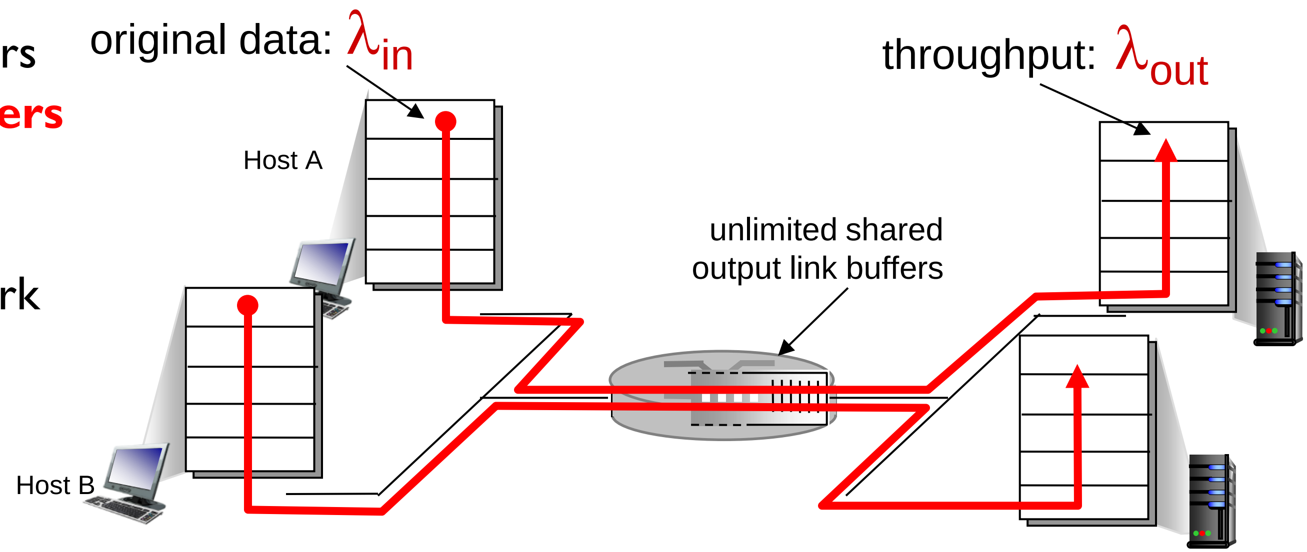
Principles of congestion control

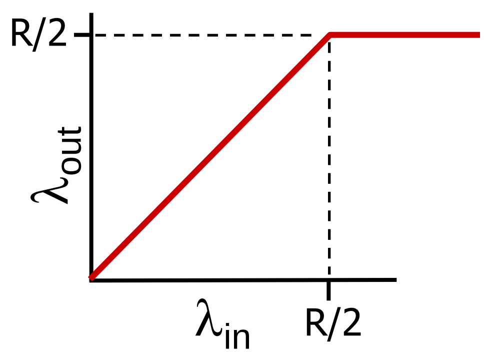
congestion:

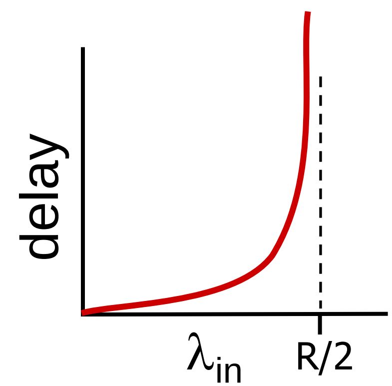
- informally: "too many sources sending too much data too fast for *network* to handle"
- different from flow control!
- Signs of Congestion:
  - lost packets (buffer overflow at routers)
  - long delays (queueing in router buffers)
- a top-10 problem!

Causes/costs of congestion: scenario (ideal 1)

- two senders, two receivers
- one router, **infinite buffers**
- output link capacity: R
- no retransmission
- Cost of congested network
  - Large queuing delay





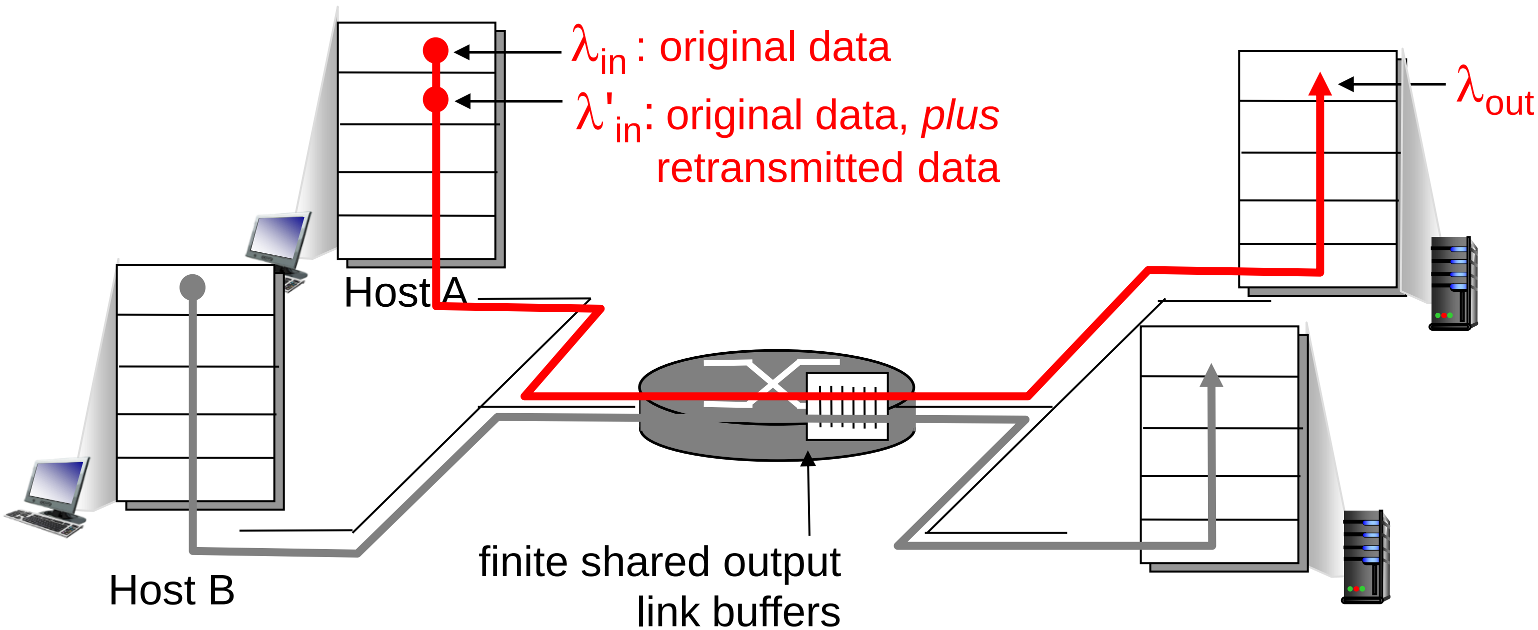


- maximum per-connection throughput: R/2

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

Causes/costs of congestion: scenario 2

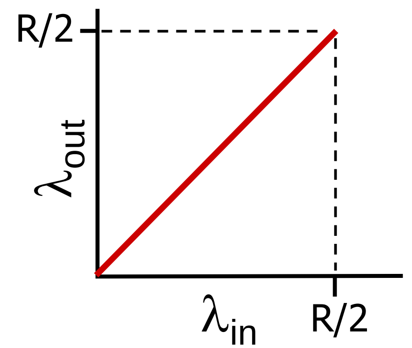
- Two senders, one router, *finite* buffers
- sender retransmission of timed-out packet
  - application-layer input = application-layer output: $\lambda_{in} = \lambda_{out}$
  - transport-layer input includes *retransmissions* : $\lambda'_{in} \geq \lambda_{in}$

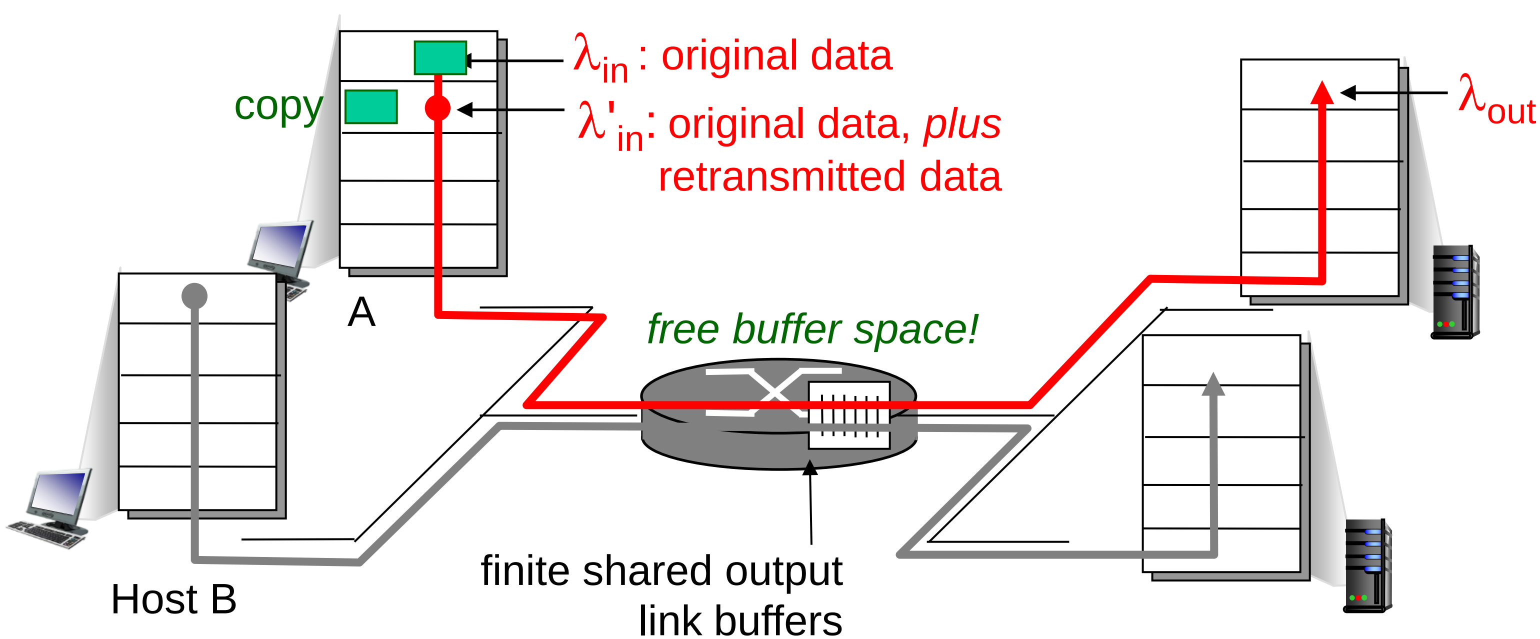


Causes/costs of congestion: scenario 2(a)

idealization: perfect knowledge

- sender sends only when router buffers available
- No retransmission
- Packet loss is assumed never to occur



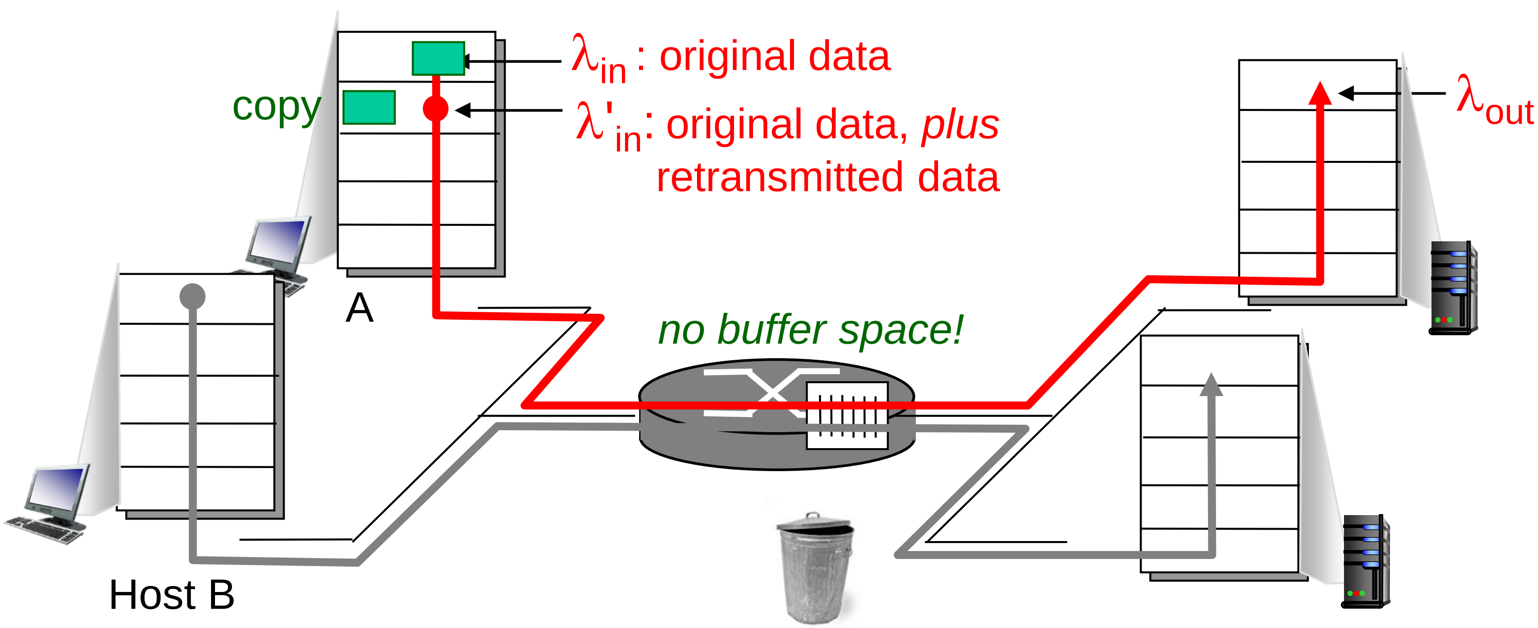


Causes/costs of congestion: scenario 2(b)

Idealization: known loss

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

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

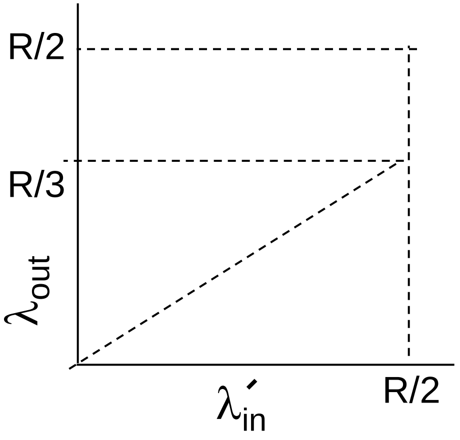


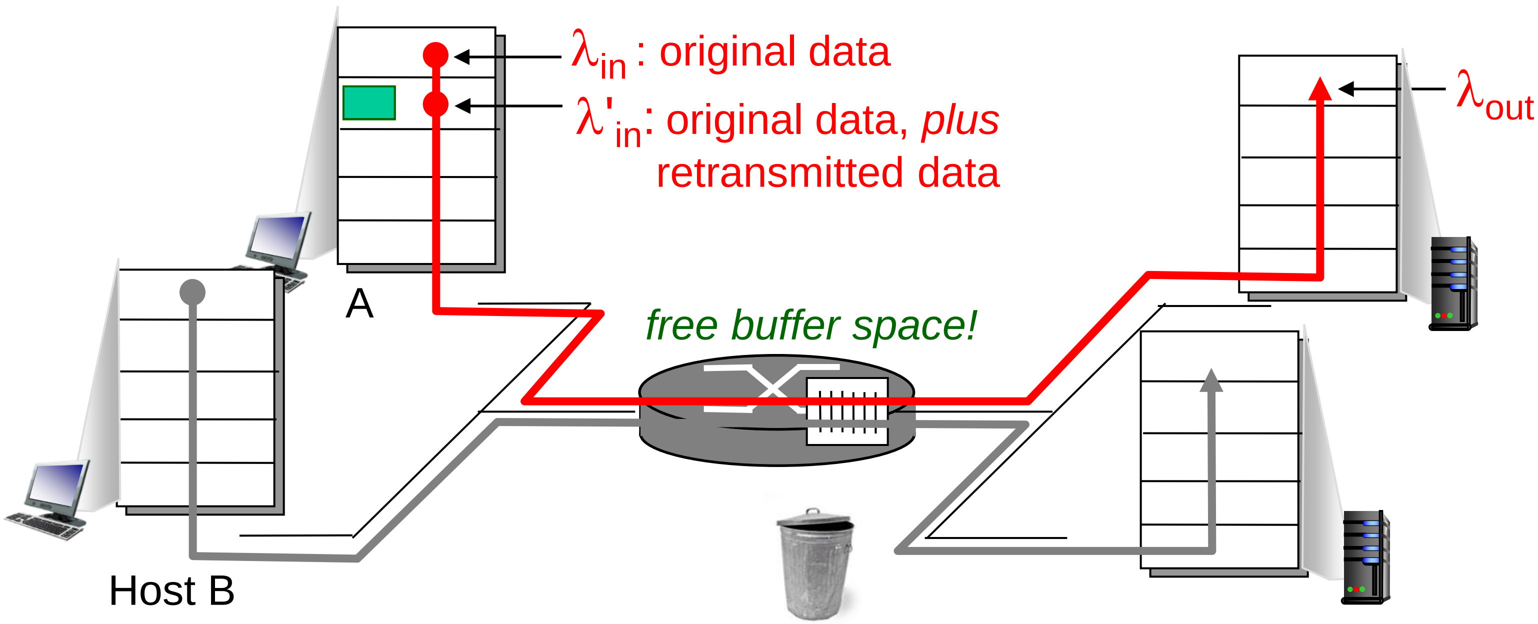
Causes/costs of congestion: scenario 2(b)

Idealization: known loss

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

- sender only resends if packet *known* to be lost
- Cost of congestion:
  - Retransmission to compensate lost pktss

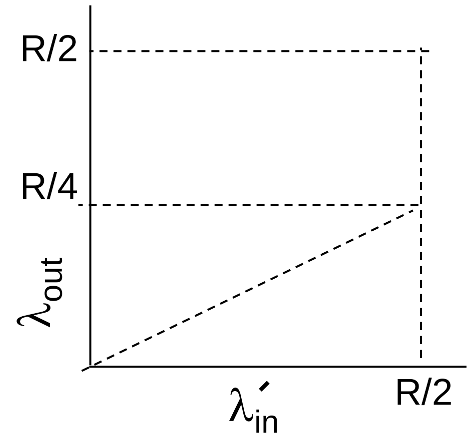


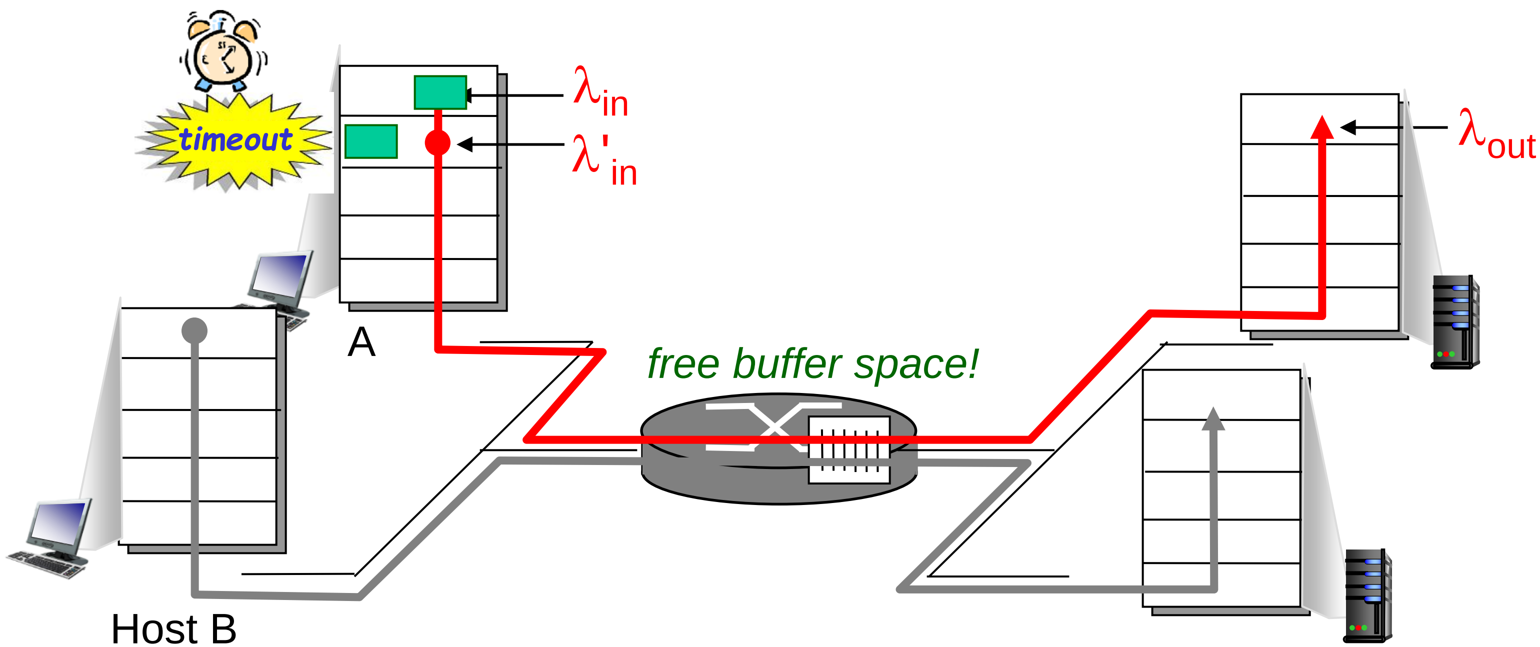


Causes/costs of congestion: scenario 2(c)

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

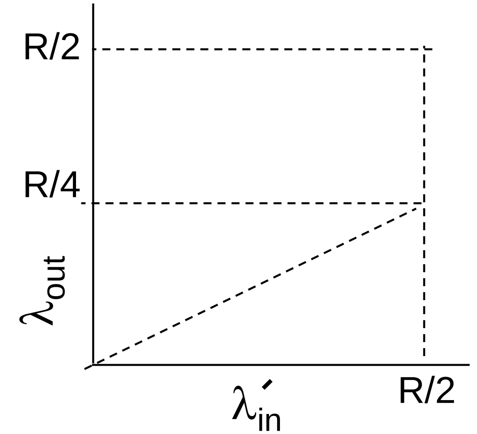




Causes/costs of congestion: scenario 2(c)

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:

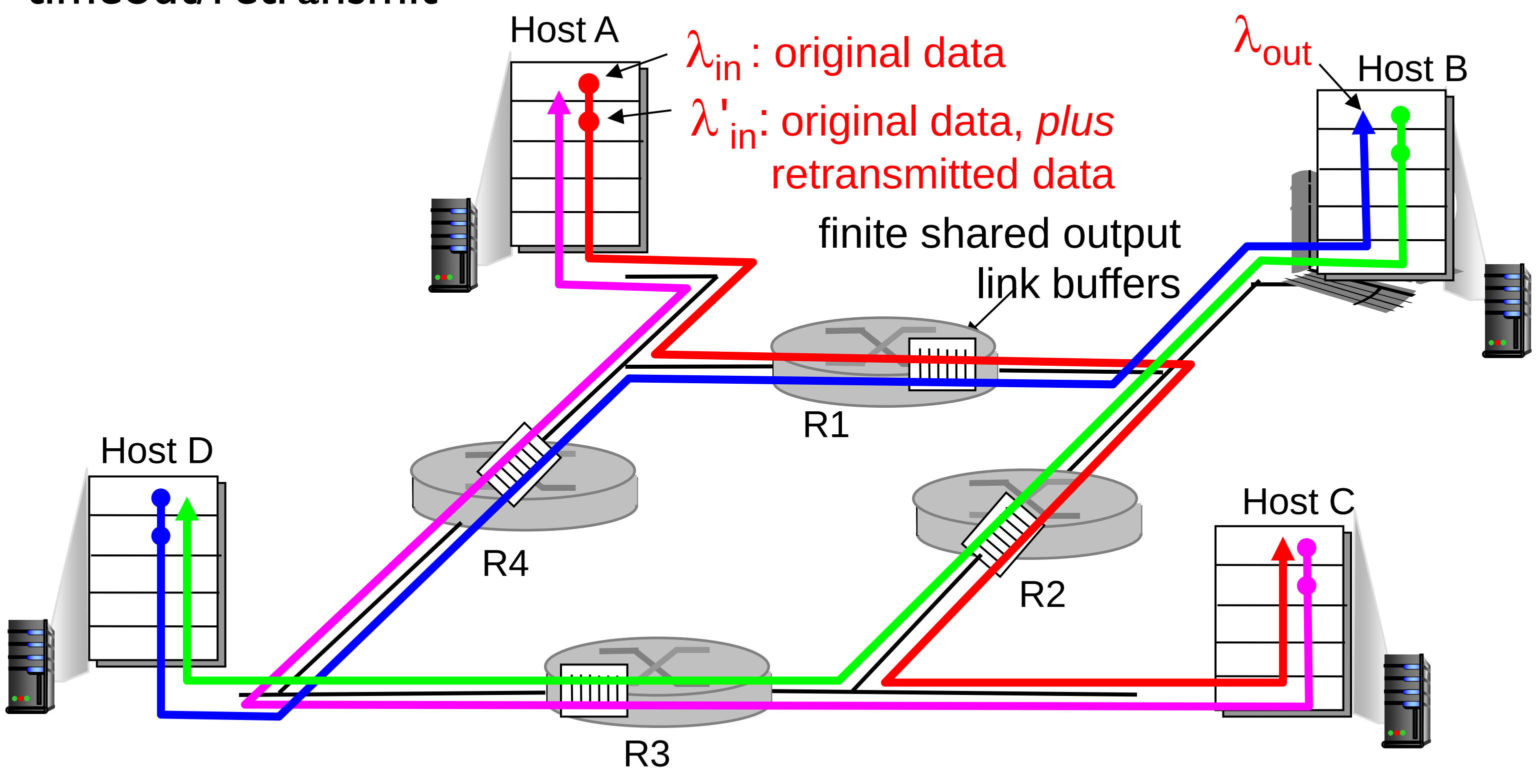
- more work (retransmissions) for given "goodput"
- unneeded retransmissions: link carries multiple copies of pkt
  - decreasing goodput

Causes/costs of congestion: scenario 3

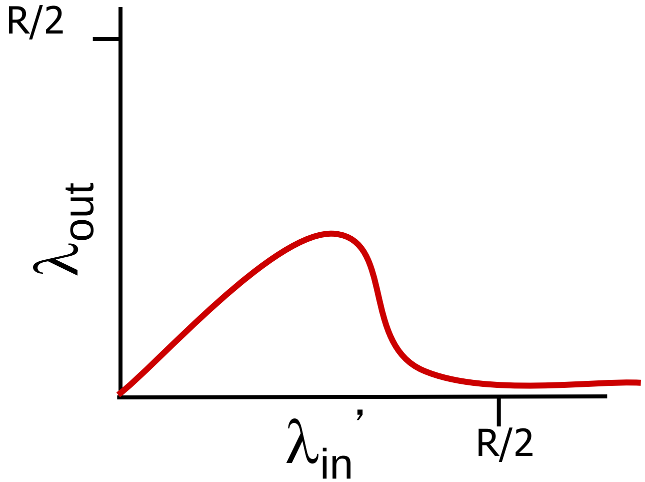
- four senders, four routers,
- Finite buffers
- multihop paths
- Shared path between **A-C** and **B-D**
- timeout/retransmit

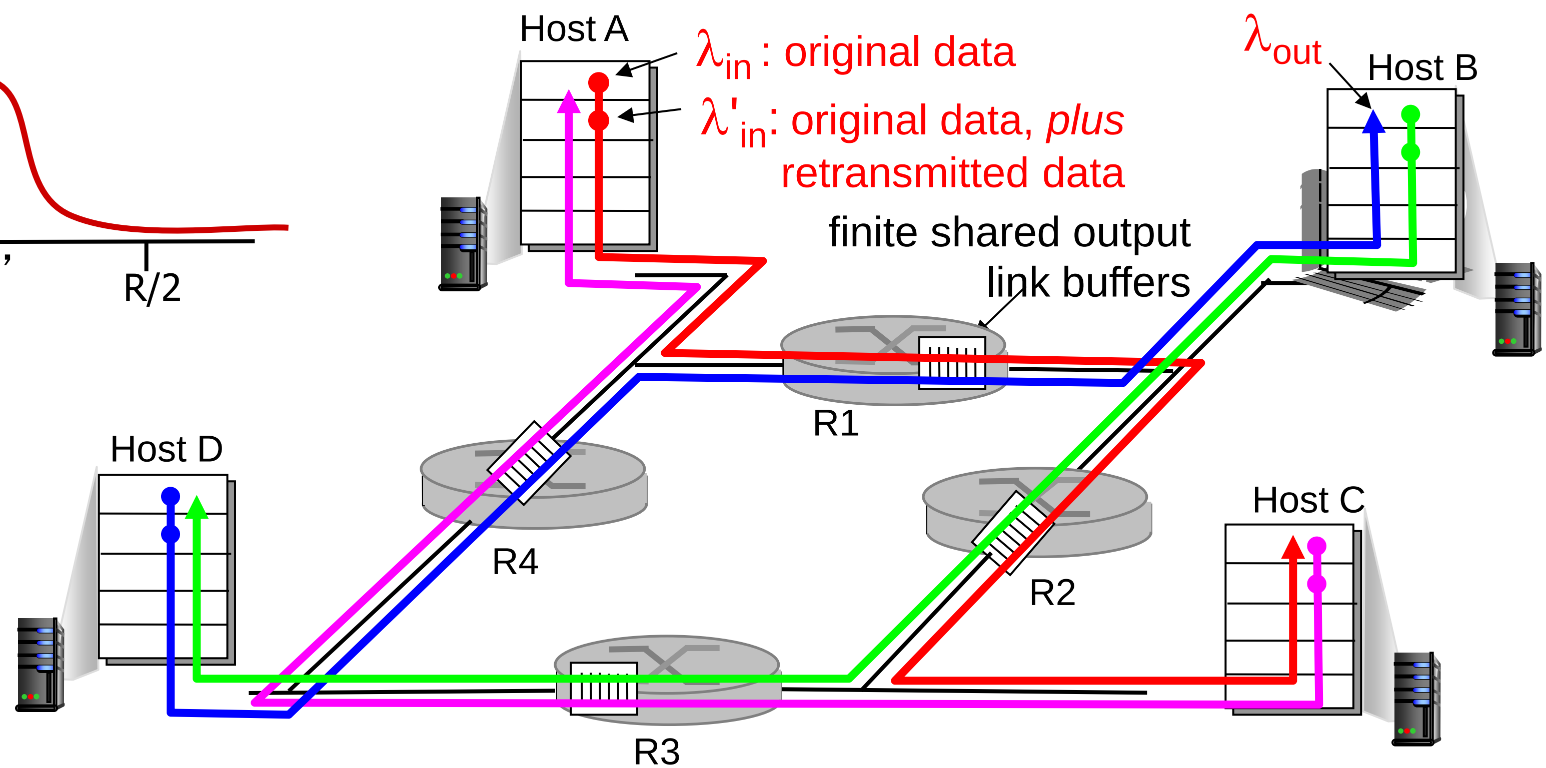
Q: what happens as λ_{in} and λ_{in}' increase ?

A: as B-D (green) λ_{in}' increases, all arriving A-C pkts (red) at R2 are dropped, red throughput → 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
  - explicit rate sender should send at

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

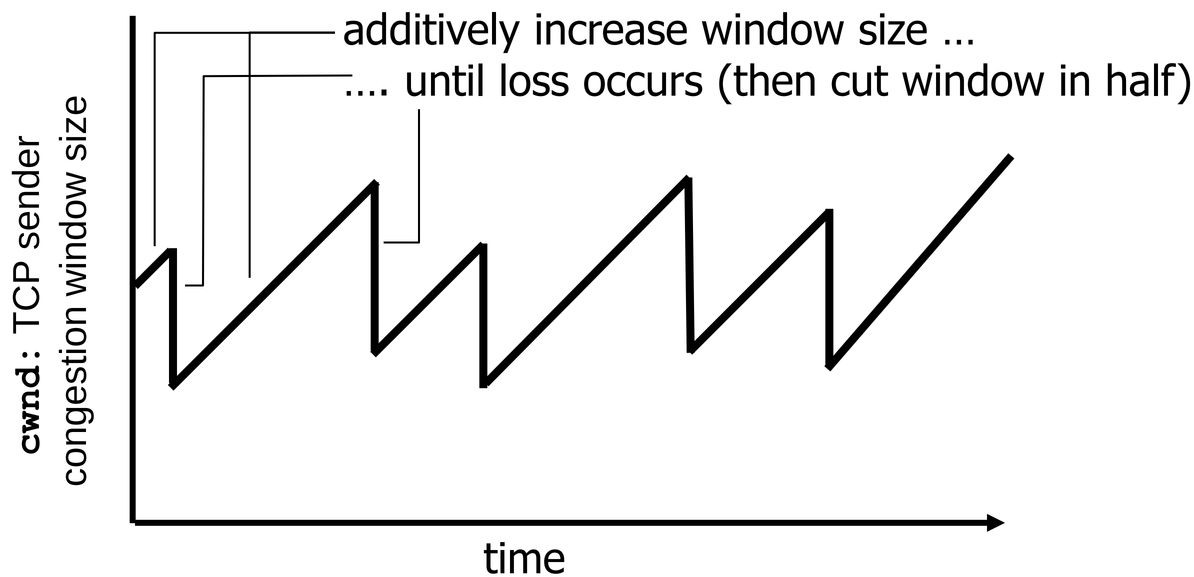
3.6 principles of congestion control

3.7 TCP congestion control

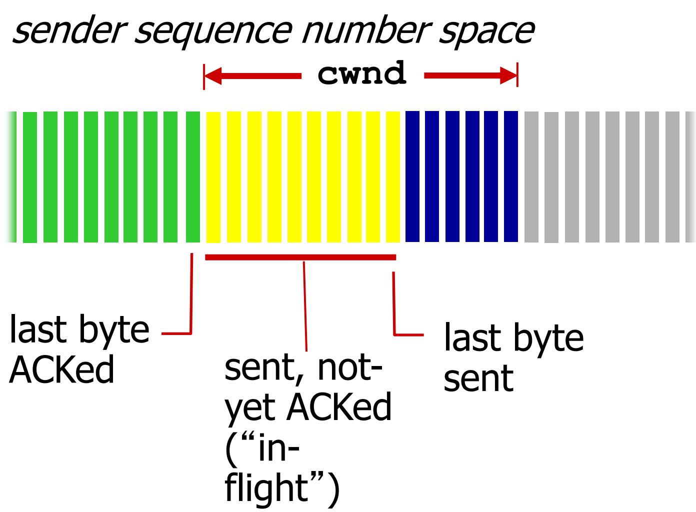
TCP congestion control: additive increase multiplicative decrease

- *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:

```
LastByteSent- LastByteAcked   ≤   cwnd
```

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

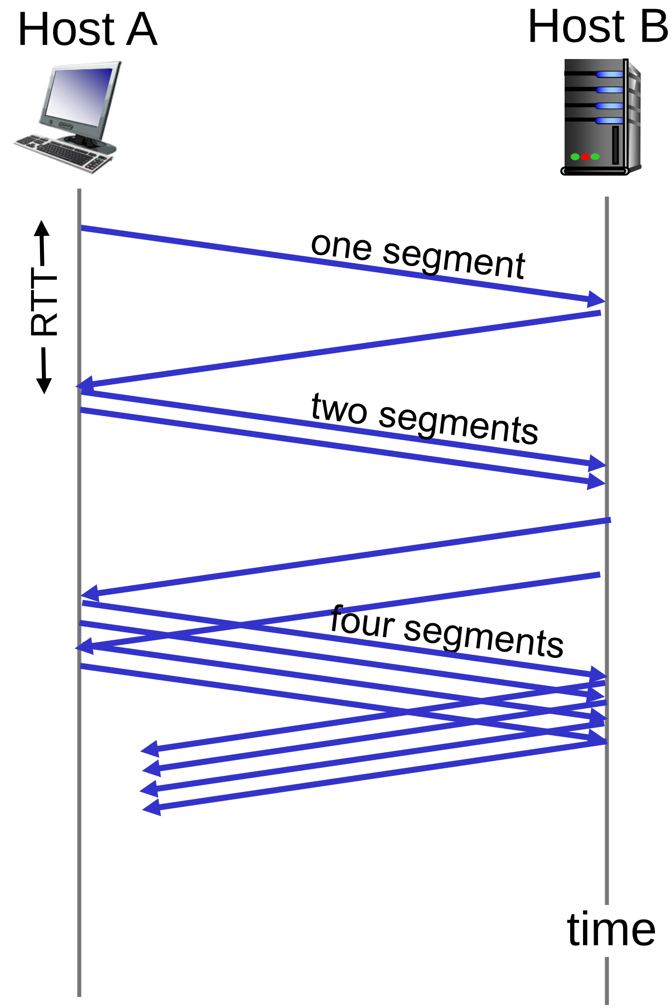
TCP sending rate:

- *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



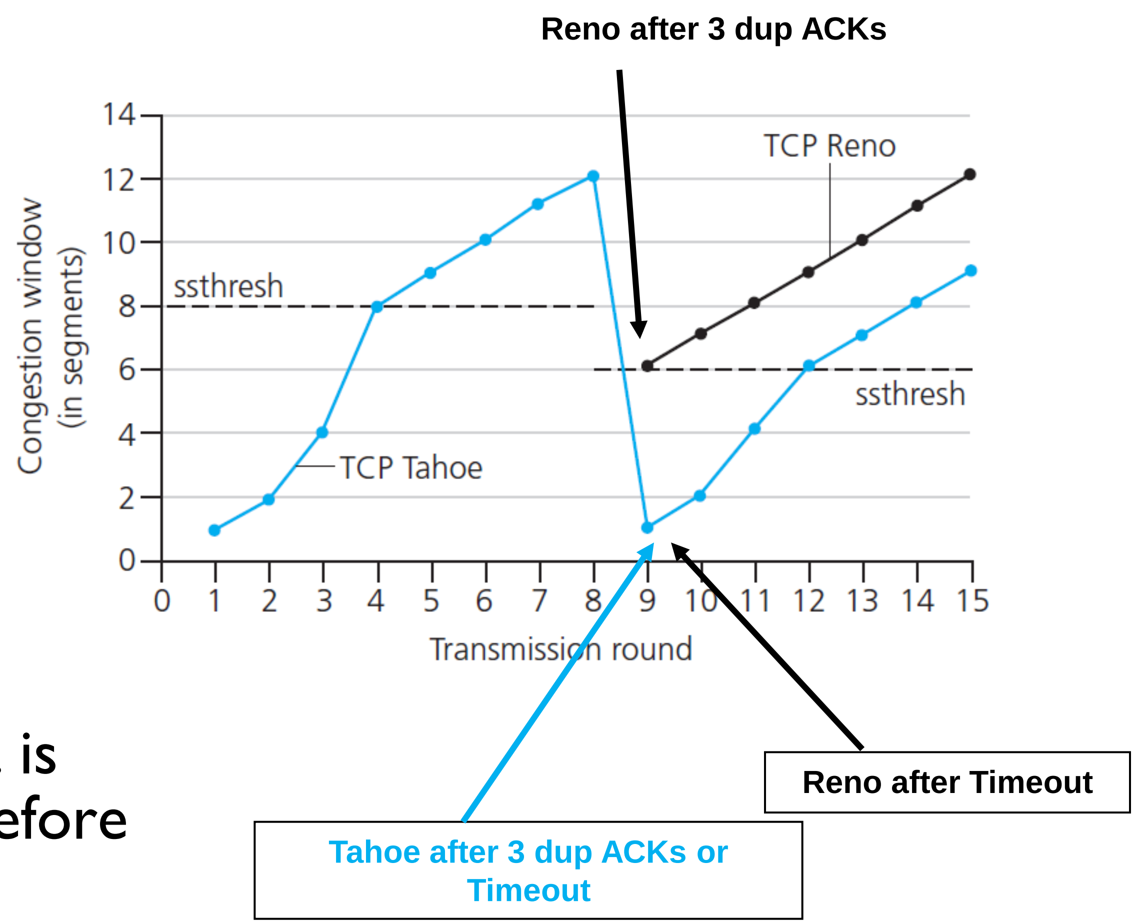
TCP: detecting, reacting to loss

- loss indicated by timeout: ***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 then grows linearly
- ***TCP Tahoe*** always sets **`cwnd`** to 1 (timeout or 3 duplicate acks)
  - window then grows exponentially (as in slow start) to threshold, then grows linearly

TCP: switching from slow start to CA(Congestion Avoidance)

Q: when should the exponential increase switch to linear?

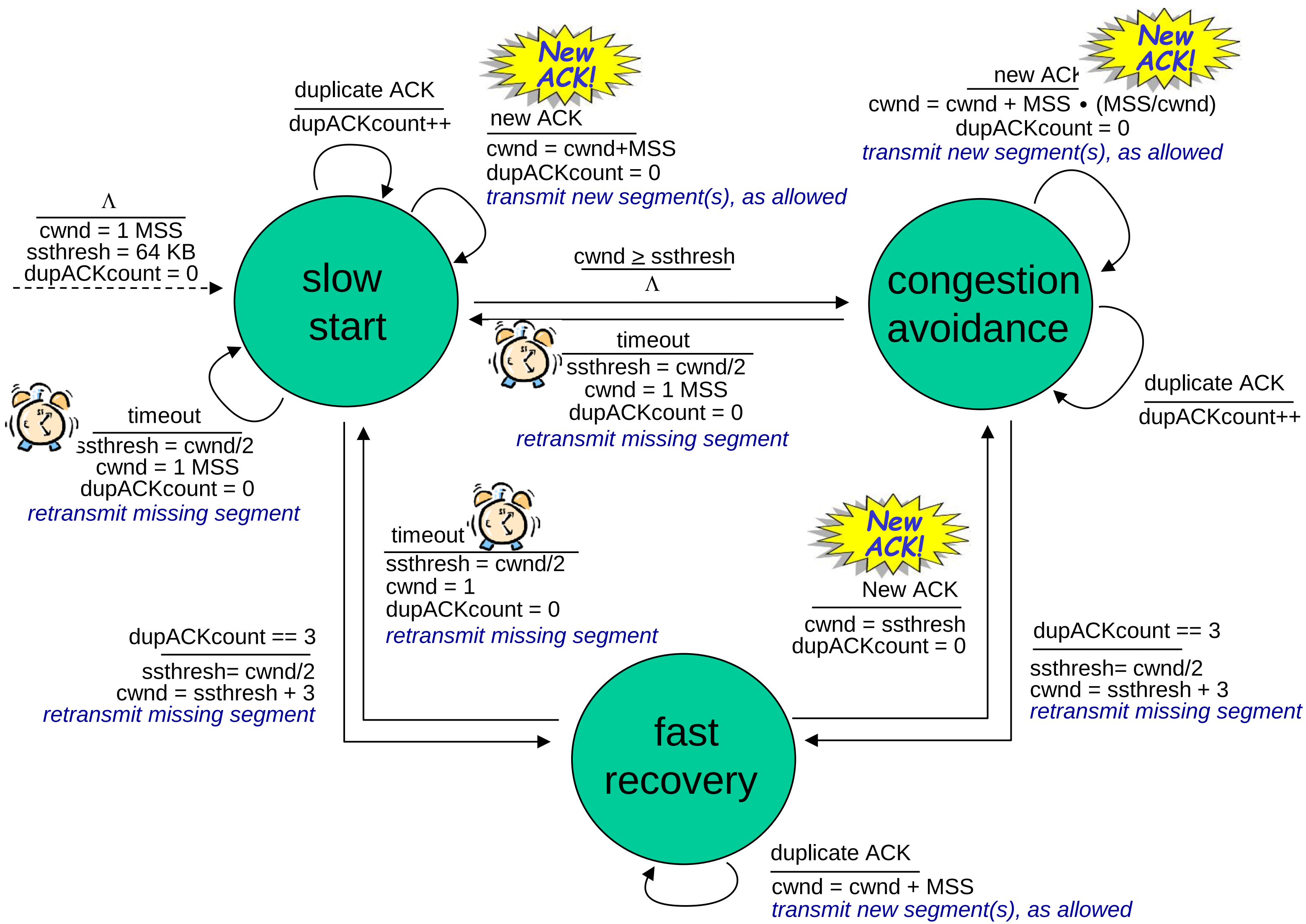
A: when **cwnd** gets to 1/2 of its value before timeout.



Implementation:

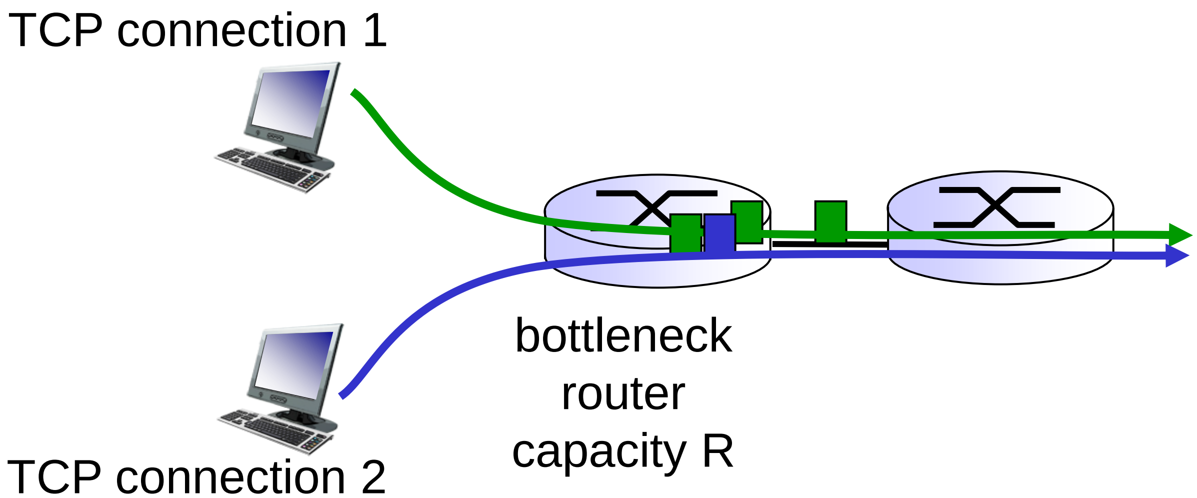
- variable **ssthresh**
- on loss event, **ssthresh** is set to 1/2 of **cwnd** just before loss event

Summary: TCP Congestion Control



TCP Fairness

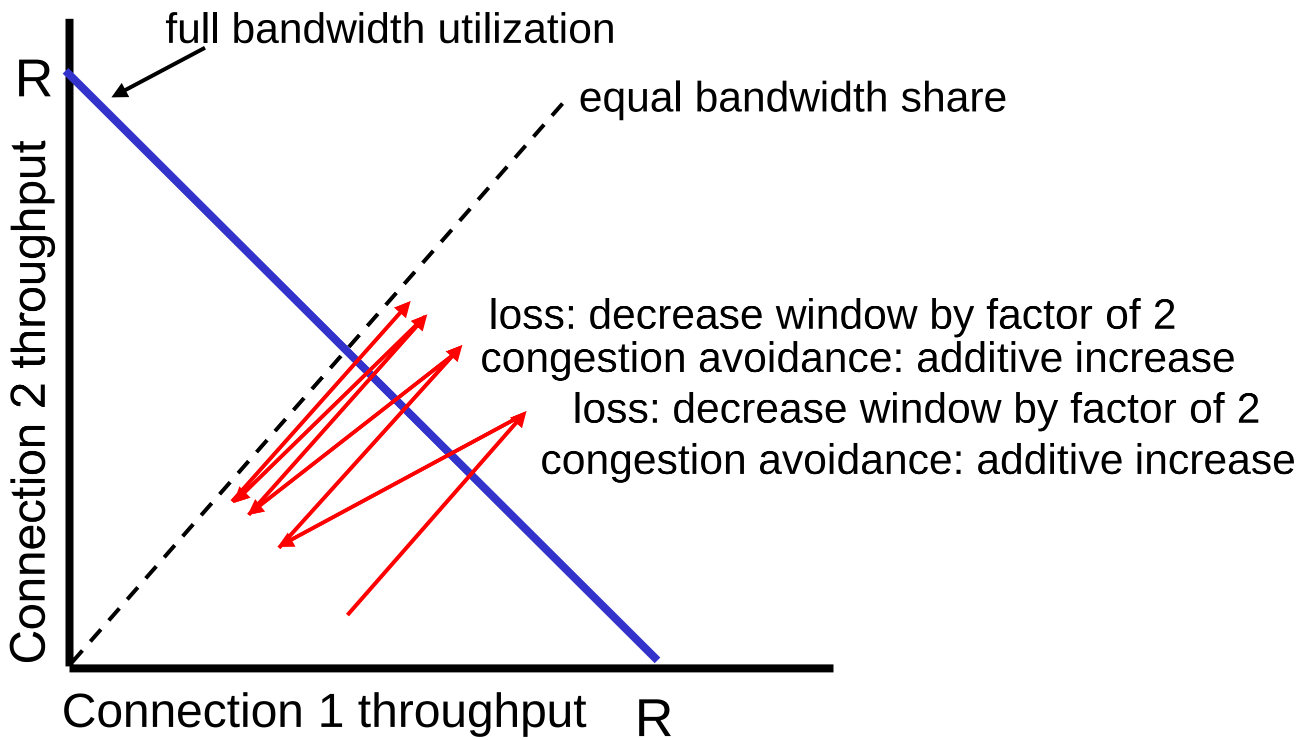
fairness goal: if K TCP sessions share same bottleneck link of bandwidth R, each should have average rate of R/K



Why is TCP fair?

two competing sessions:

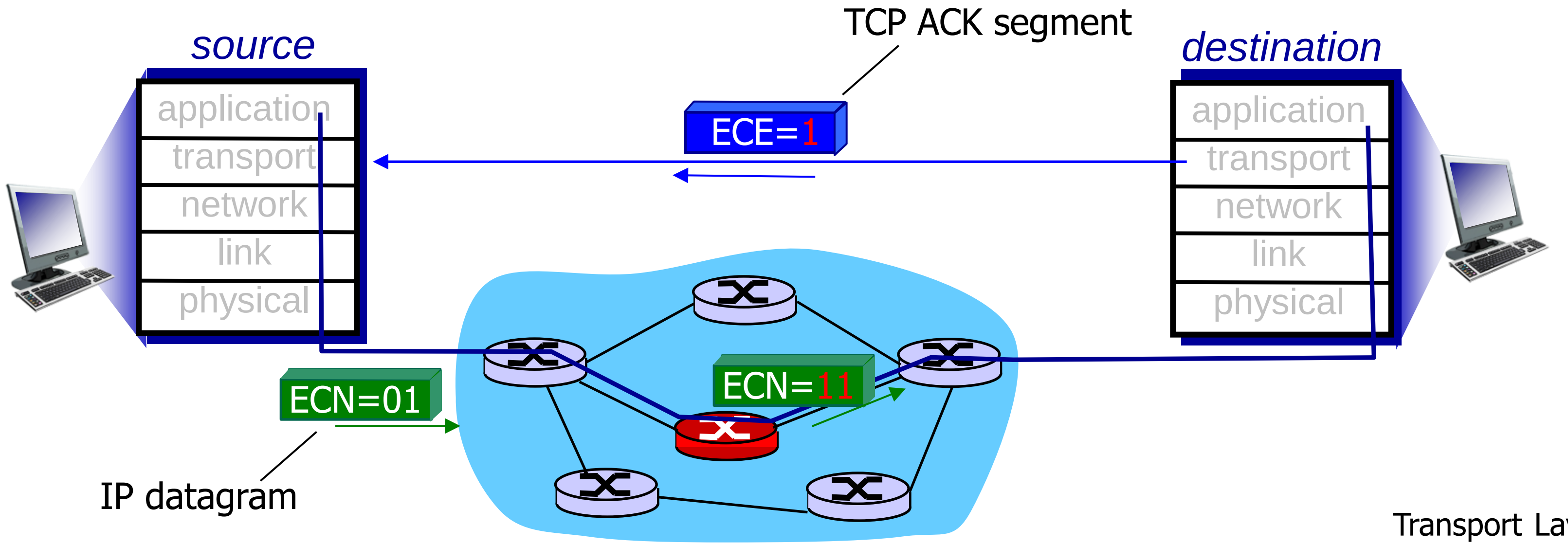
- ***additive increase*** throughout increases proportionally
- ***multiplicative decrease*** decreases throughput proportionally
- Both connections have equal RTT and MSS



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 (ECN Echoe) bit on receiver-to-sender ACK segment to notify sender of congestion



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"
- two network layer chapters:
  - data plane
  - control plane