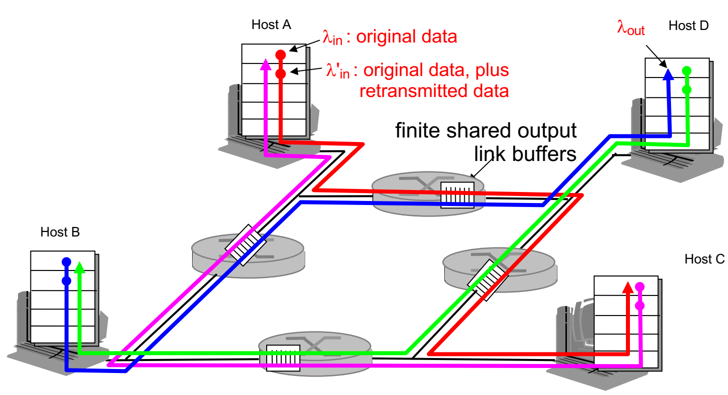
Lecture 18

- Sections 3.6.1, 3.6.2 and 3.7.1
  - Principles of congestion control
    - causes and cost of congestion
  - Approaches to congestion control
  - TCP congestion control
    - Classic TCP congestion control

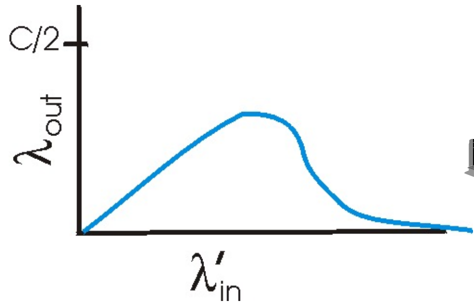
Causes/costs of congestion: scenario 3

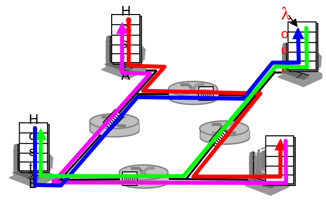
- four senders
- multihop paths
- timeout/retransmit

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



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 sender should send at (ATM)

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
- TCP timeout and round trip time
- reliable data transfer
- flow control
- connection management

3.6 Principles of congestion control
3.7 TCP congestion control

Transport Layer 3-123

123

TCP congestion controL

- Maximum transmission rate (bytes/sec) is limited by receiver window (implements flow control)
  - (receiver window)/RTT
- Sender uses a window (congestion window - cwnd) that does not exceed receiver window:
  - window expands if there is no congestion -> increase rate
  - window shrinks if there is congestion -> decrease rate
- window size increase should be conservative in order to investigate bandwidth, & avoid sudden congestion
- window size decrease should be aggressive in order to reduce congestion quickly

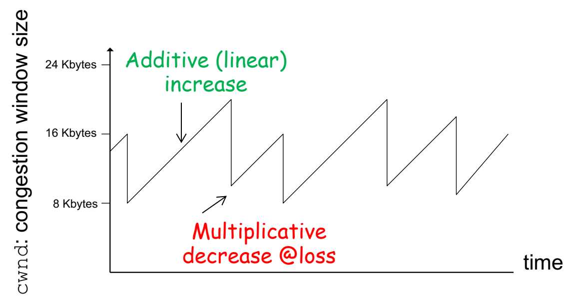
Transport Layer 3-124

124

TCP congestion control: additive increase, multiplicative decrease

- *approach:* increase 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

saw tooth behavior: probing for bandwidth



TCP Congestion Control: details

- **cwnd** = congestion window
- **cwnd** ≤ receiver window
- sender limits transmission:

  LastByteSent-LastByteAcked ≤ cwnd
- roughly,

$$\boxed{\text{rate} = \frac{\text{cwnd}}{\text{RTT}} \text{ Bytes/sec}}$$

- **cwnd** is dynamic, function of perceived network congestion:
  - increases in the absence of congestion
  - decreases when there is congestion

TCP Congestion Control: details

- **cwnd** = congestion window
- **cwnd** $\leq$ receiver window
- sender limits transmission:
  LastByteSent-LastByteAcked
  $\leq$ cwnd
- roughly,

$$\text{rate} = \frac{\text{cwnd}}{\text{RTT}} \text{ Bytes/sec}$$

- **cwnd** is dynamic, function of perceived network congestion:
  - increases in the absence of congestion
  - decreases when there is congestion

How does sender perceive congestion?

- loss event:
  - timeout, *or*
  - 3 duplicate acks
- TCP sender reduces rate (**cwnd**) after loss event ->**cwnd= cwnd/2**

three mechanisms:

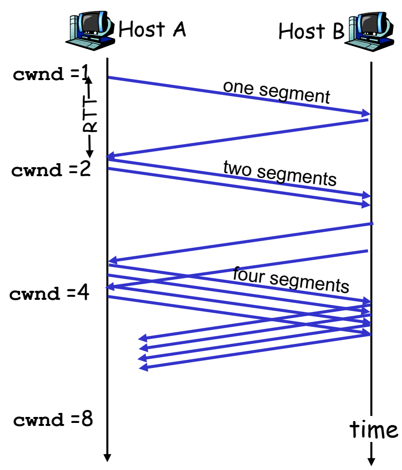
- AIMD
- slow start
- conservative probe of bandwidth

AIMD:
Additive Increase-Multiplicative Decrease

- Additive increase: increase cwnd by 1 MSS every RTT
  - assume every segment has MSS
  - cwnd is in bytes
  - cwnd is sent using cwnd/MSS segments
  - with each rcvd ACK, add number of bytes = MSS/(cwnd/MSS) = MSS^2/cwnd
- Multiplicative decrease: reduce window size by $\frac{1}{2}$
  - cwnd = cwnd/2

TCP Slow Start

- when connection begins, increase rate until first loss event:
  - initially **cwnd** = 1 MSS
  - with every ACK:
    - reinstate credit
    - add 1 MSS to **cwnd**
  - double **cwnd** every RTT
  - **cwnd** increases exponentially
- summary: initial rate is slow but ramps up exponentially fast



Refinement: inferring loss

- after 3 dup ACKs (packet lost, but following packets made it):
  - **cwnd** is cut in half
  - window then grows linearly

→*Congestion Avoidance*

- but after timeout event (lost packet & following packets):
  - **cwnd** set to 1 MSS;
  - window then grows exponentially (slow start) to a threshold, then grows linearly (congestion avoidance)

Philosophy:

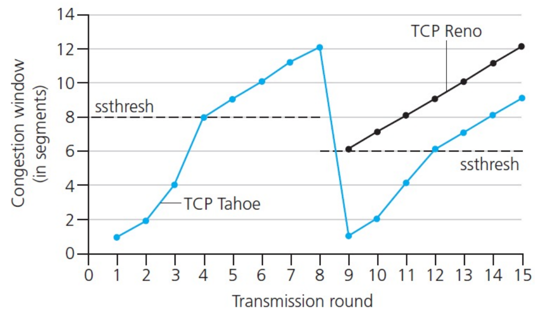
- 3 dup ACKs indicates network capable of delivering some segments
- timeout indicates a "more alarming" congestion scenario

Refinement

What is the threshold?

when **cwnd** gets to ½ of its value before timeout.

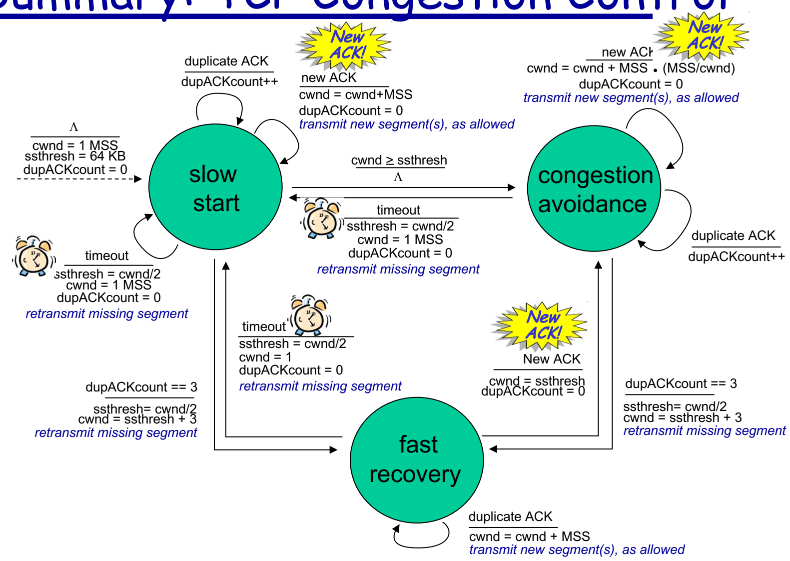
Slow Start Threshold (**ssthresh**)



Implementation:

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

Summary: TCP Congestion Control



TCP throughput

- what's the average throughout of TCP as a function of window size and RTT?
  - ignore slow start
- let W be the window size (in #MSSs) when loss occurs, and MSS=B bytes
  - when window is W, throughput is (W.B)/RTT
  - just after loss, window drops to W/2, throughput to (W.B)/2RTT.
  - average throughout: (.75 W.B)/RTT

TCP Futures: TCP over "long, fat pipes"

- example: 1500 byte segments, 100ms RTT, want 10 Gbps throughput
- requires window size W = ??? segments
- throughput in terms of loss rate:

$$\frac{1.22 \cdot MSS}{RTT\sqrt{L}}$$

- ➜ $L = 2 \cdot 10^{-10}$ *– a very small loss rate!*
- new versions of TCP for high-speed

Transport Layer 3-135

135

Chapter 3: Summary

What have we covered?

- understanding principles behind transport layer services:
  - multiplexing/demultiplexing
  - reliable data transfer
  - flow control
  - congestion control
- learning about transport layer protocols in the Internet:
  - UDP: connectionless
  - TCP: connection-oriented transport
  - TCP congestion control

Transport Layer 3-136

136