



Transport Layer

-Congestion Control (Cont...)

Lec 19

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

Case study: ATM ABR congestion control



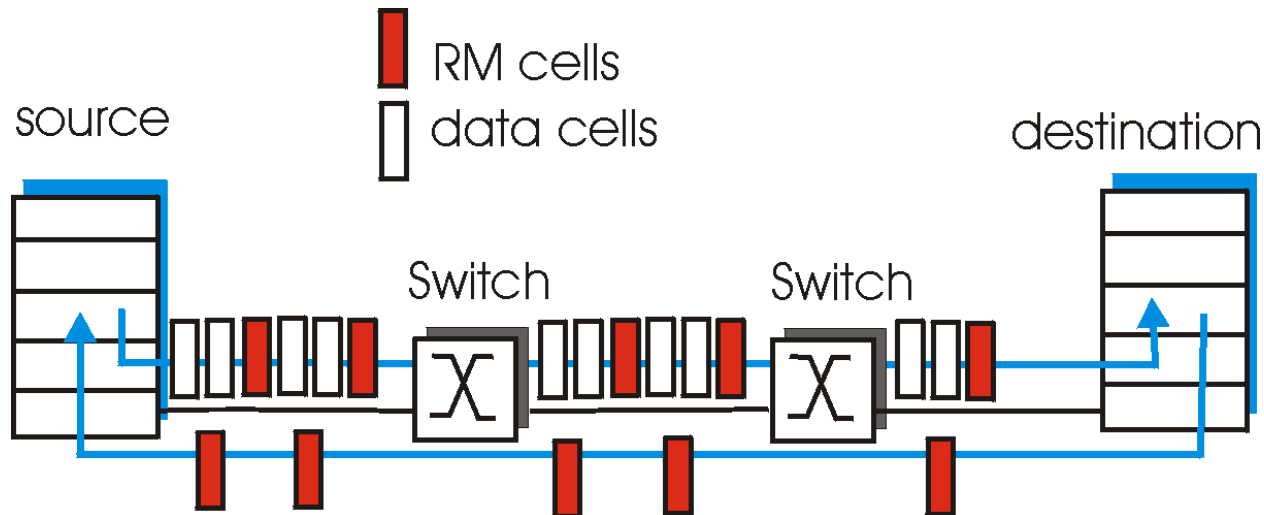
ABR: available bit rate:

- “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
 - sender's send rate thus maximum supportable rate on path
- EFCI bit in data cells: set to 1 in congested switch
 - if data cell preceding RM cell has EFCI set, sender sets CI bit in returned RM cell

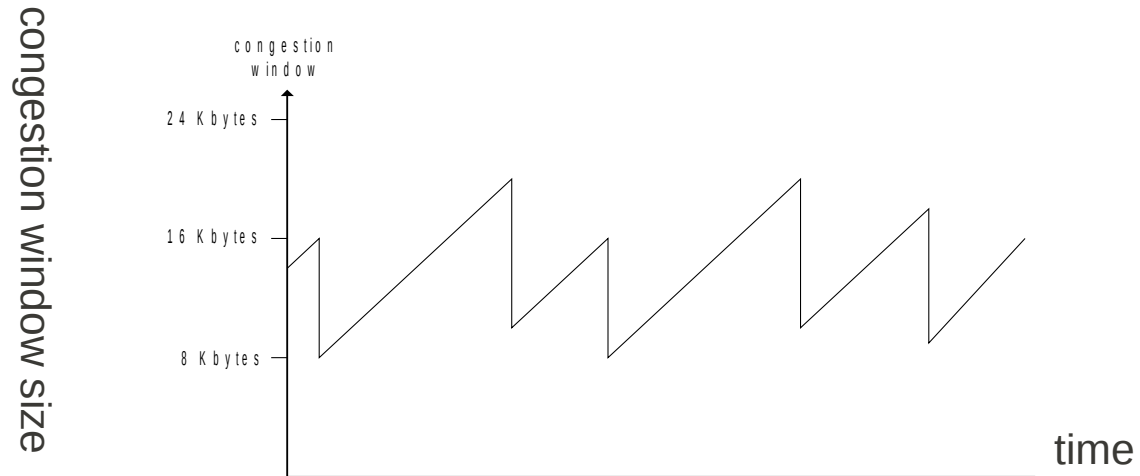
TCP congestion control:



additive increase, multiplicative decrease

- *Approach*: increase transmission rate (window size), probing for usable bandwidth, until loss occurs
 - *additive increase*: increase **CongWin** by 1 MSS every RTT until loss detected
 - *multiplicative decrease*: cut **CongWin** in half after loss

Saw tooth
behavior: probing
for bandwidth





TCP Congestion Control: details

- sender limits transmission:
LastByteSent - LastByteAcked
≤ CongWin

- Roughly,

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

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

How does sender perceive congestion?

- loss event = timeout or 3 duplicate acks
- TCP sender reduces rate (**CongWin**) after loss event

three mechanisms:

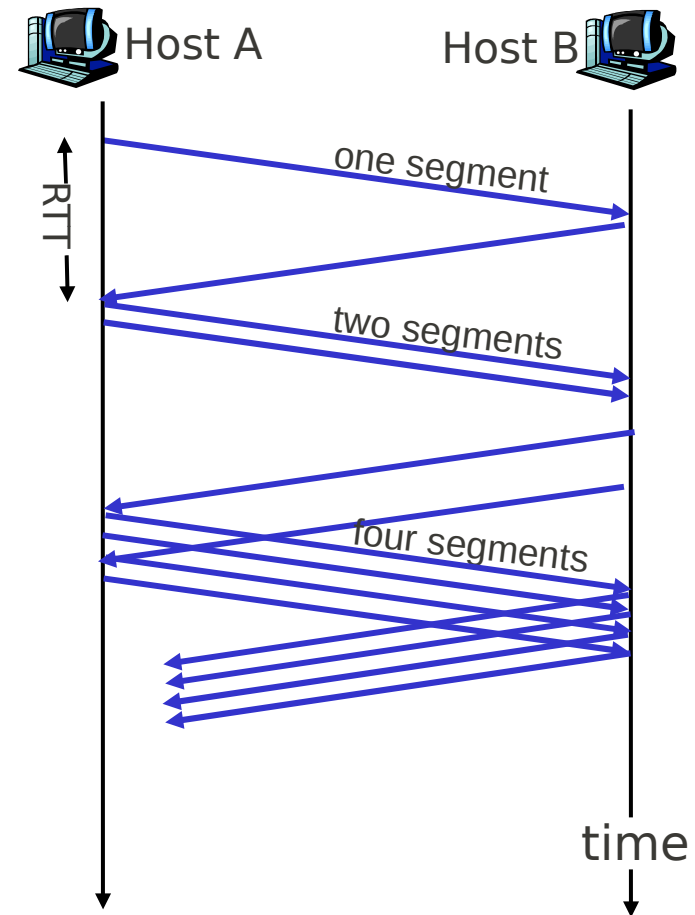
- AIMD
- slow start
- conservative after timeout events

TCP Slow Start

- When connection begins, **CongWin** = 1 MSS
 - Example: MSS = 500 bytes & RTT = 200 msec
 - initial rate = 20 kbps
- available bandwidth may be \gg MSS/RTT
 - desirable to quickly ramp up to respectable rate
- When connection begins, increase rate exponentially fast until first loss event

TCP Slow Start (more)

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

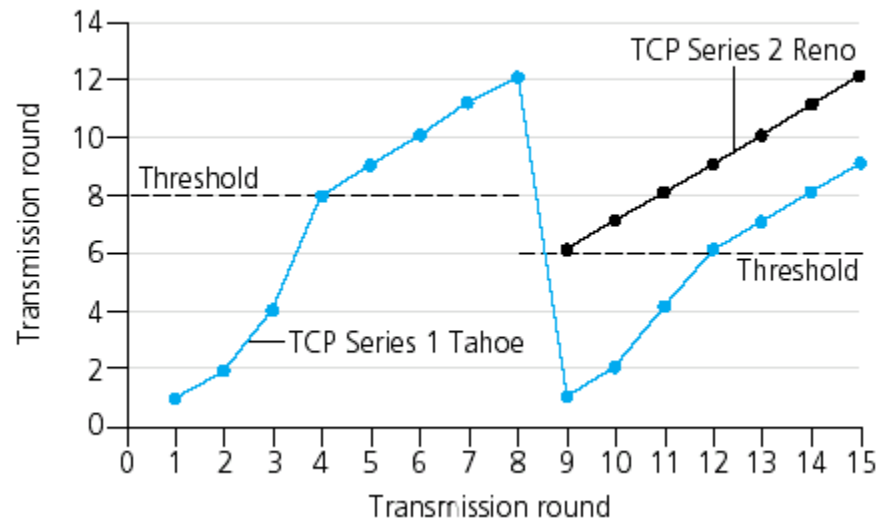


Refinement



Q: When should the exponential increase switch to linear?

A: When **CongWin** gets to 1/2 of its value before timeout.



Implementation:

- Variable Threshold
- At loss event, Threshold is set to 1/2 of CongWin just before loss event

Refinement: inferring loss

- After 3 dup ACKs:
 - **CongWin** is cut in half
 - window then grows linearly
- But after timeout event:
 - **CongWin** instead set to 1 MSS;
 - window then grows exponentially
 - to a threshold, then grows linearly

Philosophy:

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

Summary: TCP Congestion Control

- When **CongWin** is below **Threshold**, sender in **slow-start** phase, window grows exponentially.
- When **CongWin** is above **Threshold**, sender is in **congestion-avoidance** phase, window grows linearly.
- When a **triple duplicate ACK** occurs, **Threshold** set to **CongWin/2** and **CongWin** set to **Threshold**.
- When **timeout** occurs, **Threshold** set to **CongWin/2** and **CongWin** is set to 1 MSS.