Data Communication & Networks
G22.2262-001

Session 9 - Main Theme
Network Congestion: Causes, Effects, Controls

Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Agenda

- What is Congestion?
- Effects of Congestion
- Causes/Costs of Congestion
- Approaches Towards Congestion Control
- TCP Congestion Control
- TCP Fairness
- Conclusion

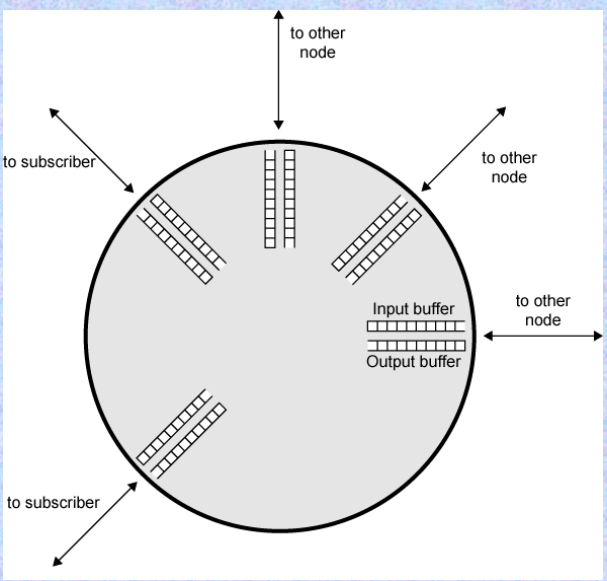
Part I

What is Congestion?

What is Congestion?

- Congestion occurs when the number of packets being transmitted through the network approaches the packet handling capacity of the network
- Congestion control aims to keep number of packets below level at which performance falls off dramatically
- Data network is a network of queues
- Generally 80% utilization is critical
- Finite queues mean data may be lost
- A top-10 problem!

Queues at a Node



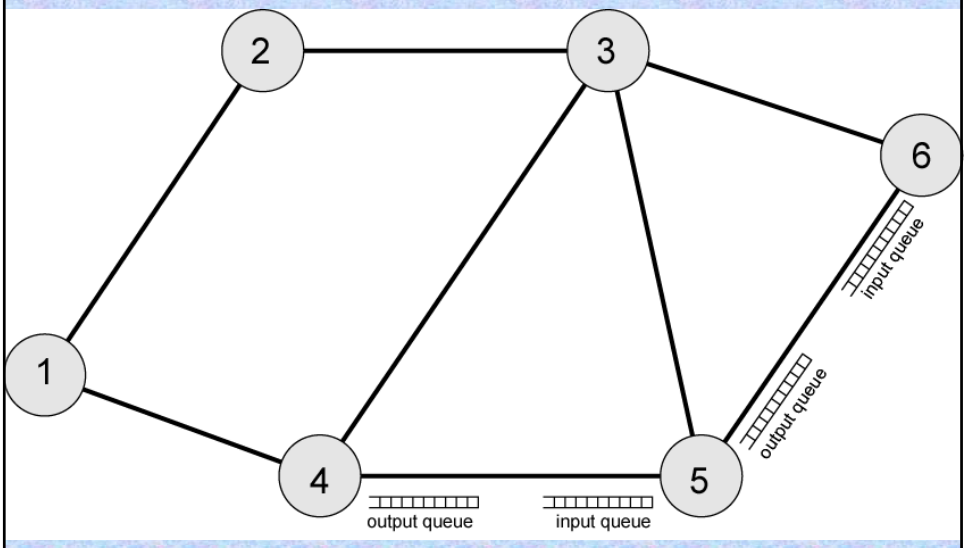
Part II

Effects of Congestion?

Effects of Congestion

- Packets arriving are stored at input buffers
- Routing decision made
- Packet moves to output buffer
- Packets queued for output transmitted as fast as possible
  - Statistical time division multiplexing
- If packets arrive to fast to be routed, or to be output, buffers will fill
- Can discard packets
- Can use flow control
  - Can propagate congestion through network

Interaction of Queues

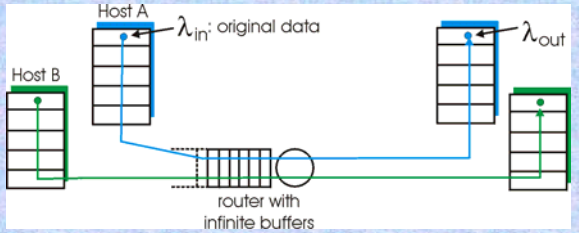


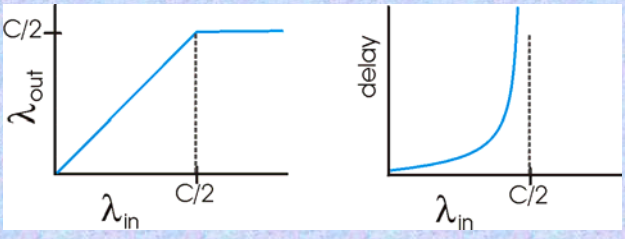
Part III

Causes/Costs of Congestion

Causes/Costs of Congestion: Scenario 1

- two senders, two receivers
- one router, infinite buffers
- no retransmission

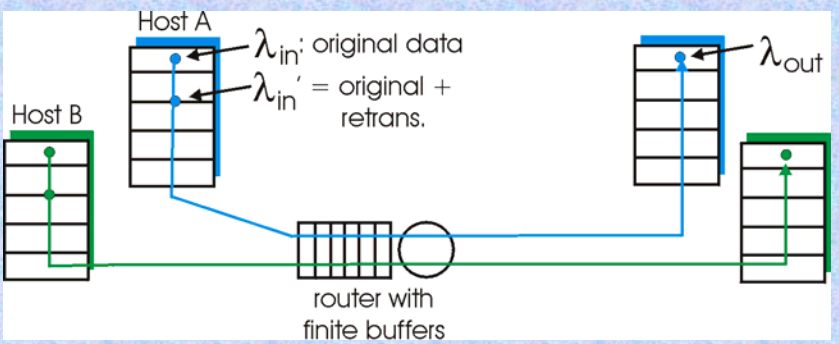




- large delays when congested
- maximum achievable throughput

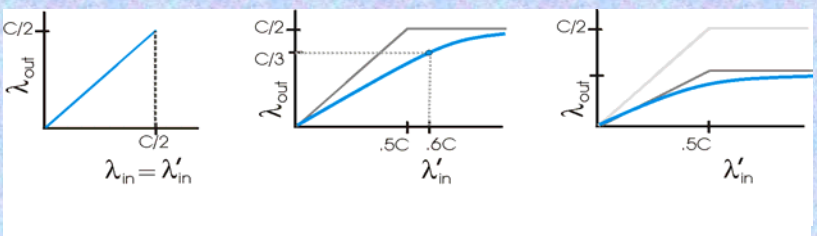
Causes/Costs of Congestion: Scenario 2

- one router, *finite* buffers
- sender retransmission of lost packet



Causes/Costs of Congestion: Scenario 2

- always: $\lambda_{in} = \lambda_{out}$ ($\lambda'_{in} = \lambda_{in}$)
- "perfect" retransmission only when loss: $\lambda'_{in} > \lambda_{out}$
- retransmission of delayed (not lost) packet makes λ'_{in} larger (than perfect case) for same λ_{out}



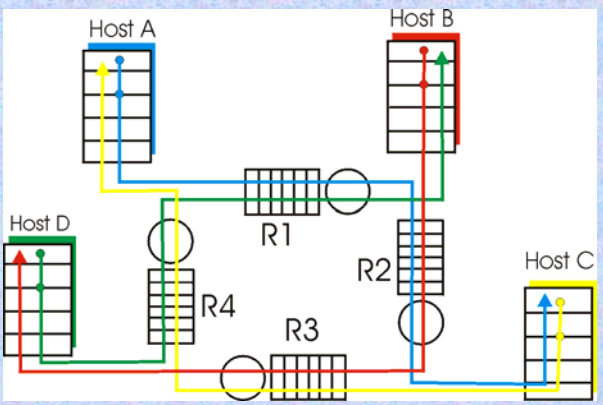
"costs" of congestion:

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

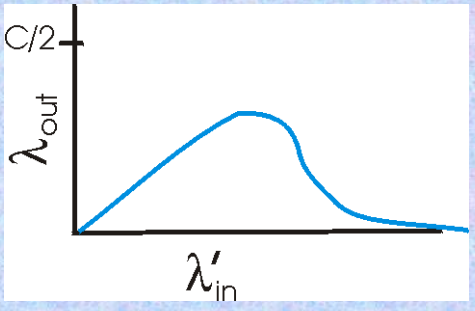
Causes/Costs of Congestion: Scenario 3

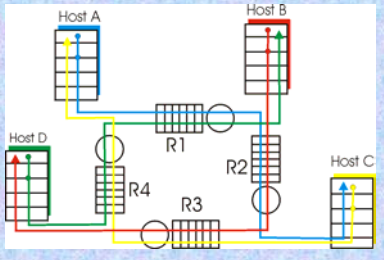
- four senders
- multihop paths
- timeout/retransmit

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!

Part IV

Approaches Towards Congestion Control

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

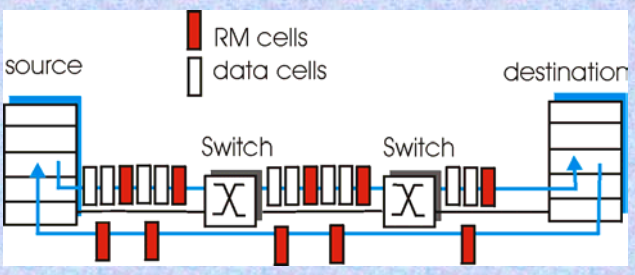
Case Study: ATM ABR Congestion Control

- **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' send rate thus minimum 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

Part V

TCP Congestion Control

TCP Congestion Control

- end-end control (no network assistance)
- 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

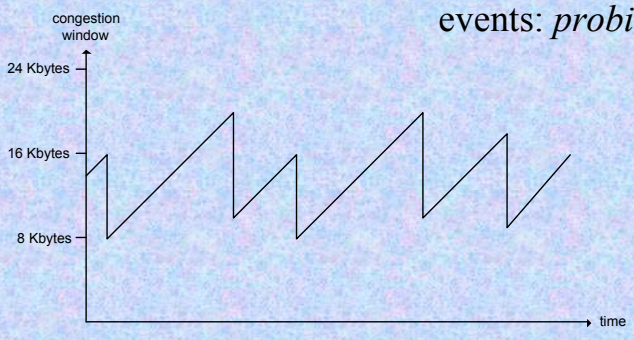
three mechanisms:

- AIMD
- slow start
- conservative after timeout events

TCP AIMD

multiplicative decrease: cut **CongWin** in half after loss event

additive increase: increase **`CongWin`** by 1 MSS every RTT in the absence of loss events: *probing*



Long-lived TCP connection

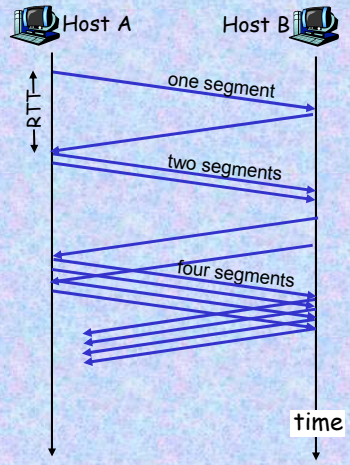
TCP Slow Start

- When connection begins, **CongWin** = 1 MSS
  - Example: MSS = 500 bytes & RTT = 200 msec
  - initial rate = 20 kbps
- available bandwidth may be >> 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

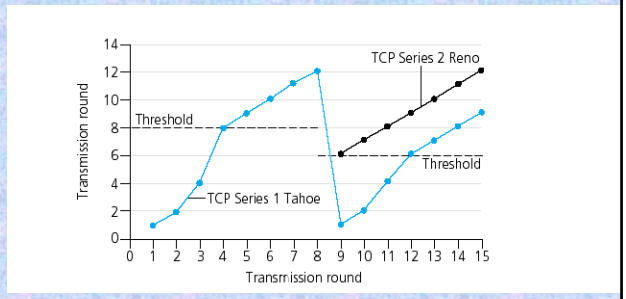
- After 3 dup ACKs:
  - **CongWin** is cut in half
  - window then grows linearly
- But after timeout event:
  - **CongWin** instead set to 1 MSS;
  - window then grows exponentially
  - to a threshold, then grows linearly

Philosophy:

- 3 dup ACKs indicates network capable of delivering some segments
- timeout before 3 dup ACKs is "more alarming"

Refinement (more)

- **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

Summary: TCP Congestion Control

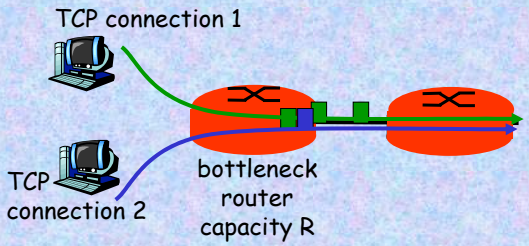
- When **CongWin** is below **Threshold**, sender in slow-start phase, window grows exponentially
- When **CongWin** is above **Threshold**, sender is in congestion-avoidance phase, window grows linearly
- When a triple duplicate ACK occurs, **Threshold** set to **CongWin/2** and **CongWin** set to **Threshold**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

Part VI

TCP Fairness

TCP Fairness

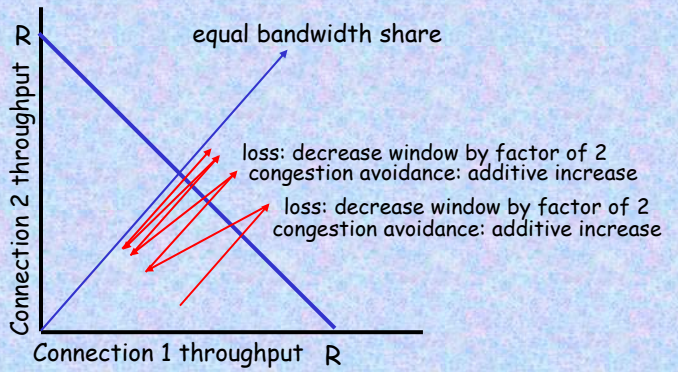
Fairness goal: if K TCP sessions share same bottleneck link of bandwidth R, each should have average rate of R/K



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:
  - pump audio/video at constant rate, tolerate packet loss
- Research area: TCP friendly

Fairness and parallel TCP connections

- nothing prevents app from opening parallel connections between 2 hosts.
- Web browsers do this
- Example: link of rate R supporting 9 connections;
  - new app asks for 1 TCP, gets rate R/10
  - new app asks for 11 TCPs, gets R/2 !

Part VII

Conclusion

Assignment & Readings

- Assignment #5 (due 04/15/10)
  - Assigned at the completion of Session 8
- Readings
  - Chapter 3 (3.6, 3.7)
  - RFC 2581

Next Session:
Java Sockets