

Part 1: Networking Review

Goals:

- review key topics from intro networks course
 - equalize backgrounds
 - identify remedial work
 - ease into course

Overview:

- overview
- error control
- **flow control**
- **congestion control**
- routing
- LANs
- addressing
- synthesis:
 - "a day in the life"
 - control timescales

1-42

Flow Control (in TCP)

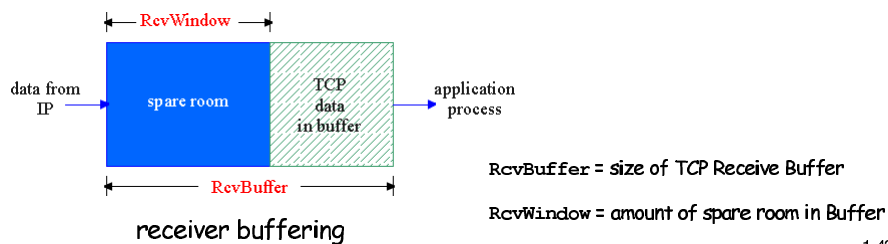
flow control

sender won't overrun receiver's buffers by transmitting too much, too fast

receiver: explicitly informs sender of (dynamically changing) amount of free buffer space

- **RcvWindow** field in TCP segment

sender: keeps the amount of transmitted, unACKed data less than most recently received **RcvWindow**



1-43

Principles of Congestion Control

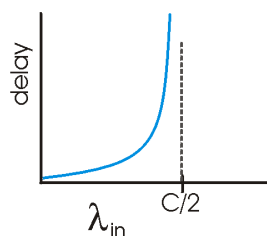
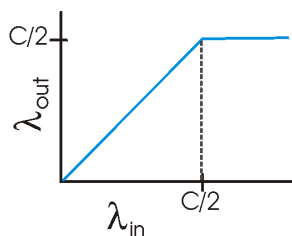
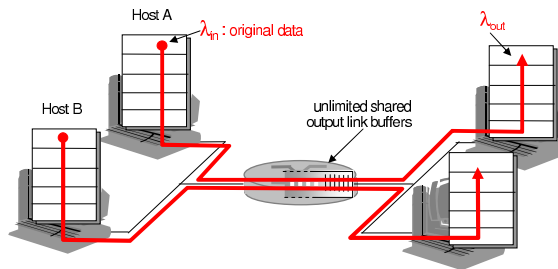
Congestion:

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

1-44

Causes/costs of congestion: scenario 1

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

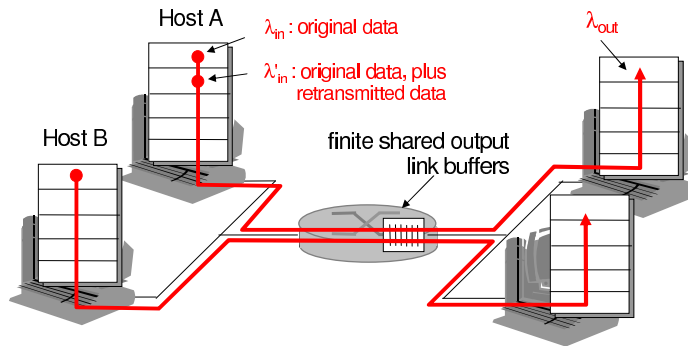


- ❑ large delays when congested
- ❑ maximum achievable throughput

1-45

Causes/costs of congestion: scenario 2

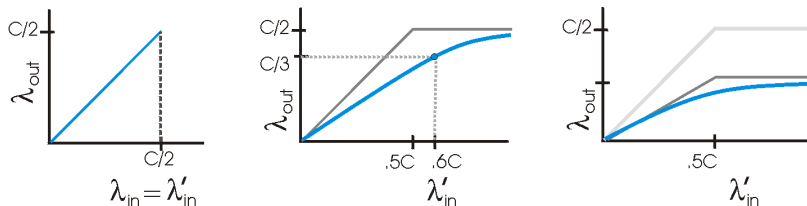
- ❑ one router, *finite* buffers
- ❑ sender retransmission of lost packet



1-46

Causes/costs of congestion: scenario 2

- ❑ always: $\lambda_{in} = \lambda_{out}$ (goodput)
- ❑ "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:

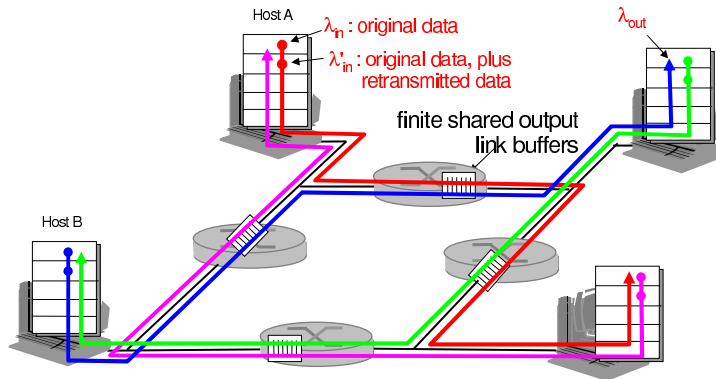
- ❑ more work (retrans) for given "goodput"
- ❑ unneeded retransmissions: link carries multiple copies of pkt

1-47

Causes/costs of congestion: scenario 3

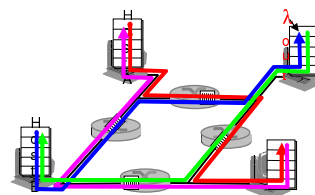
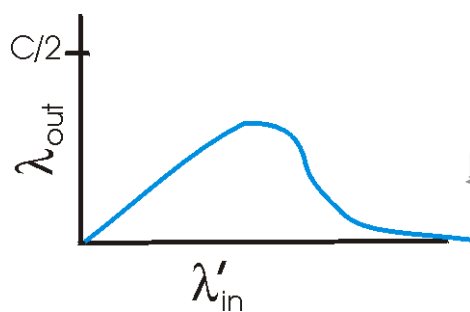
- four senders
- multihop paths
- timeout/retransmit

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



1-48

Causes/costs of congestion: scenario 3



Another "cost" of congestion:

- when packet dropped, any "upstream transmission capacity used for that packet was wasted!"

1-49

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
- Forms of feedback
 - sender: "choke" packet
 - receiver: packet hdr field

1-50

Case study: ATM ABR congestion control

ABR: available bit rate:

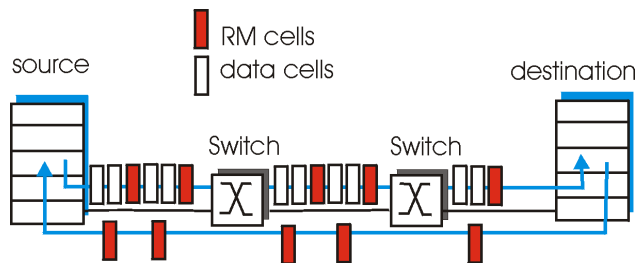
- "elastic service"
- if sender's path "underloaded":
 - sender should use available bandwidth
- if sender's path congested:
 - sender throttled to minimum guaranteed rate
- 3 signaling mechanisms:
 - NI/CI, ER, EFCI

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
 - Switches can also directly generate RM cells to source

1-51

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 (explicit forward congestion indication) 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

1-52

TCP Congestion Control

- end-end control (no network assistance)
- transmission rate limited by congestion window size, Congwin, over segments:



1-53

TCP congestion control:

- "probing" for usable bandwidth:
 - ideally: transmit as fast as possible (Congwin as large as possible) without loss
 - increase Congwin until loss (congestion)
 - loss: decrease Congwin, then begin probing (increasing) again
- Inherent assumption
 - Loss = congestion
- two "phases"
 - slow start
 - congestion avoidance
- important variables:
 - Congwin
 - threshold: defines threshold between the slow start and the congestion avoidance phase

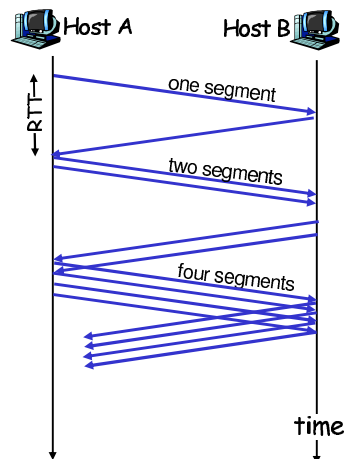
1-54

TCP Slowstart

Slowstart algorithm

initialize: Congwin = 1
for (each segment ACKed)
Congwin++
until (loss event OR
CongWin > threshold)

- exponential increase (per RTT) in window size (not so slow!)
- loss event: timeout (Tahoe TCP) and/or or three duplicate ACKs (Reno TCP)

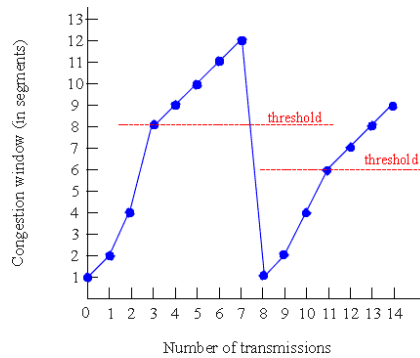


1-55

TCP Congestion Avoidance: Tahoe

~~TCP Tahoe Congestion avoidance~~

```
/* slowstart is over */
/* Congwin > threshold */
Until (loss event) {
  every w segments ACKed:
    Congwin++
}
threshold = Congwin/2
Congwin = 1
perform slowstart
```



Numerous improvements: TCP Reno, SACK

1-56