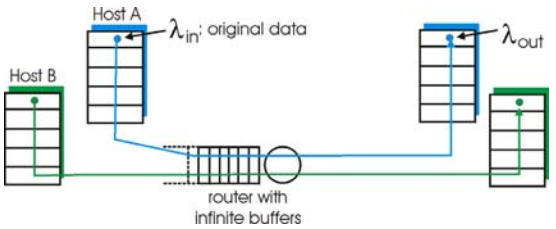
Principles of Congestion Control

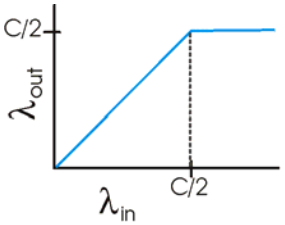
Congestion:

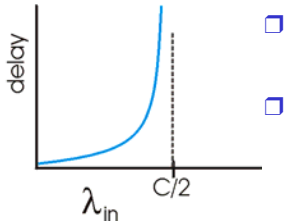
- informally: "too many sources sending too much data too fast for *network* to handle"
- different from flow control!
- manifestations:
  - lost packets (buffer overflow at routers)
  - long delays (queueing in router buffers)
- a top-10 problem!

Causes/costs of congestion: scenario 1

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



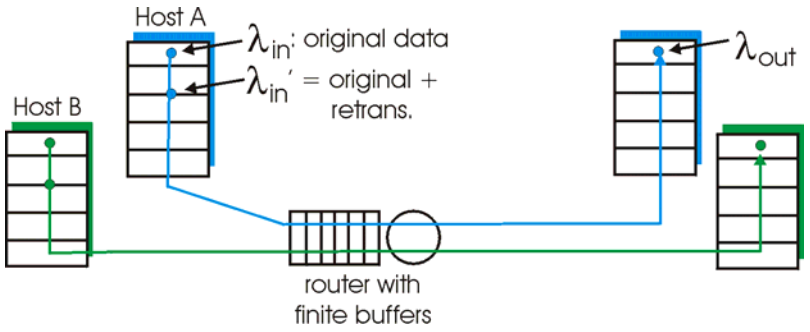




- large delays when congested
- maximum achievable throughput

Causes/costs of congestion: scenario 2

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



Costs of congestion: scenario 2

- always: $\lambda_{in} = \lambda_{out}$
- retransmission when loss: $\lambda_{in}' > \lambda_{out}$
- retransmission of delayed (not lost) packet makes λ_{in}' still larger for same λ_{out}

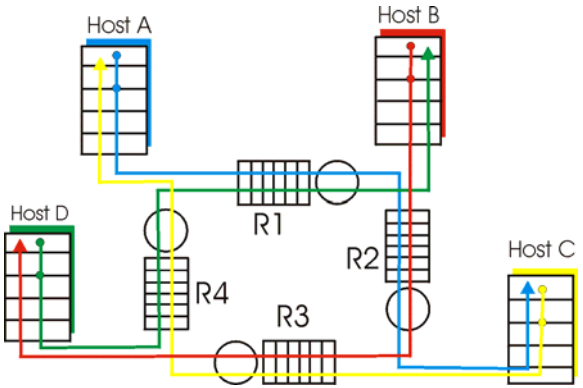
"costs" of congestion:

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

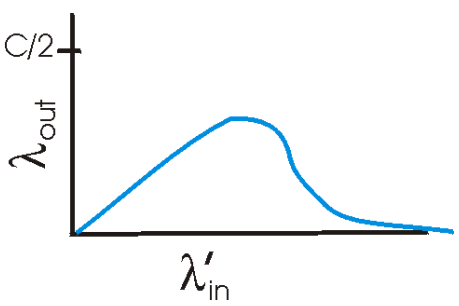
Causes/costs of congestion: scenario 3

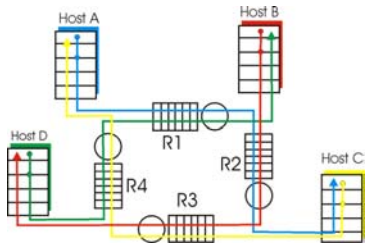
- four senders
- multihop paths
- timeout/retransmit

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!

Approaches towards congestion control

Two broad approaches towards congestion control:

End-to-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 indication to sender

Case study: ATM ABR congestion control

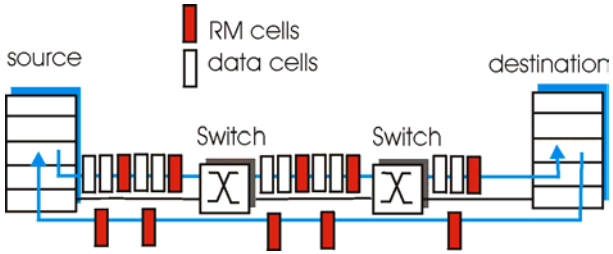
ABR: available bit rate:

- "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

TCP Congestion Control

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



- w segments, each with MSS bytes sent in one RTT:

$$\text{throughput} = \frac{w * MSS}{RTT} \text{ Bytes/sec}$$

TCP congestion control:

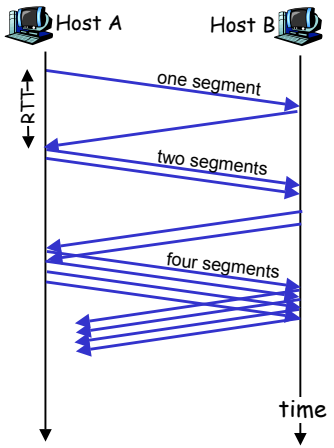
- "probing" for usable bandwidth:
  - ideally: transmit as fast as possible (`Congwin` as large as possible) without loss
  - *increase* `Congwin` until loss (congestion)
  - loss: *decrease* `Congwin`, then begin probing (increasing) again
- two "phases"
  - slow start
  - congestion avoidance
- important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

TCP Slowstart

Slowstart algorithm

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

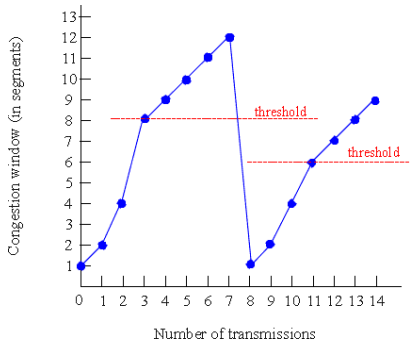
- exponential increase (per RTT) in window size (not so slow!)
- loss event: timeout (Tahoe TCP) and/or or three duplicate ACKs (Reno TCP)



TCP Congestion Avoidance

Congestion avoidance

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



1: TCP Reno skips slowstart (fast recovery) after three duplicate ACKs

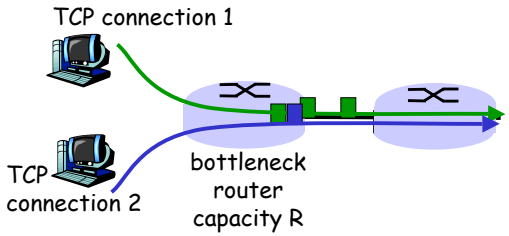
AIMD

TCP congestion avoidance:

- AIMD: *additive increase, multiplicative decrease*
  - increase window by 1 per RTT
  - decrease window by factor of 2 on loss event

TCP Fairness

Fairness goal: if N TCP sessions share same bottleneck link, each should get 1/N of link capacity



Why is TCP fair?

Two competing sessions:

- Additive increase gives slope of 1, as throughout increases
- multiplicative decrease decreases throughput proportionally

