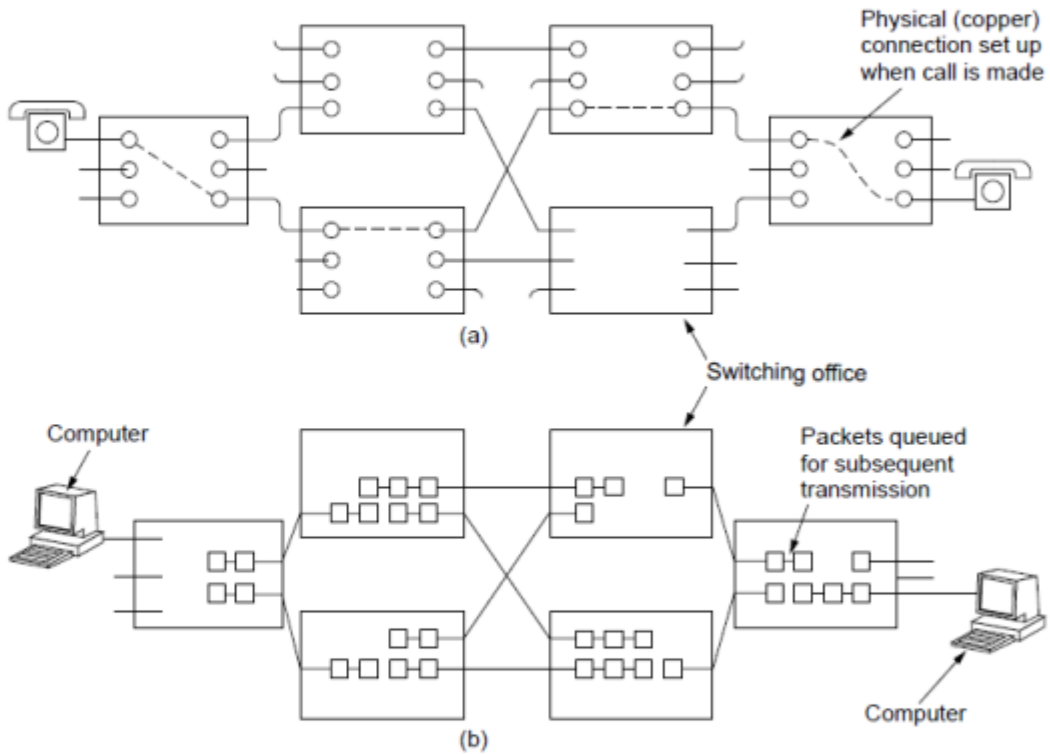
Congestion control

Overview

- Congestion in the network
  - Connection model and flows
  - What routers do
- Avoiding congestion collapse
  - Congestion control by senders
  - Slow down sending for the greater good
- TCP congestion control algorithm
  - Slow start, fast retransmit, fast recovery
- Congestion avoidance
  - Detecting eminent before packet loss

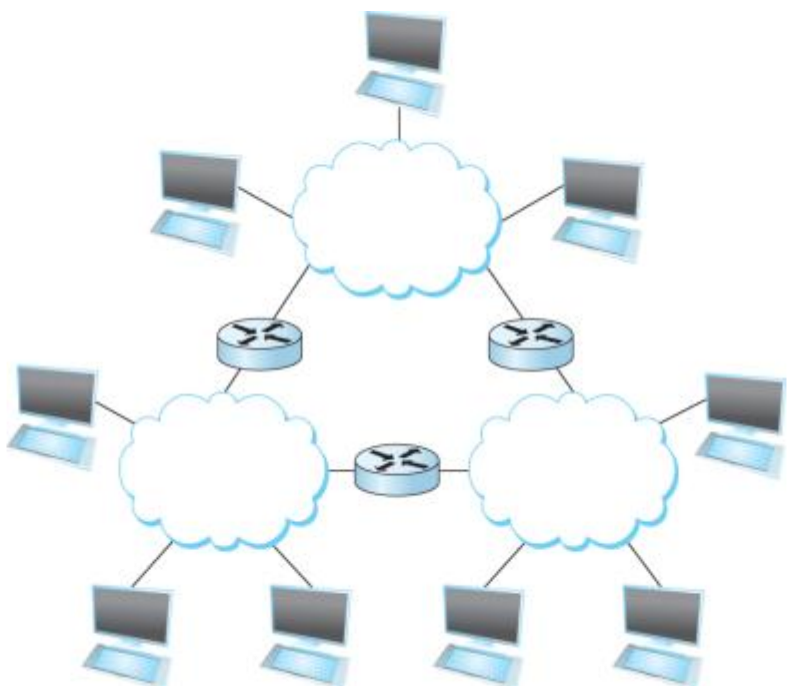
Not a problem with circuit switching

- Connection-oriented (circuit switched)
  - Nodes reserve resources (e.g. buffer space along path)
  - Circuit is rejected if resources aren't available
  - Cannot exceed what the network can handle



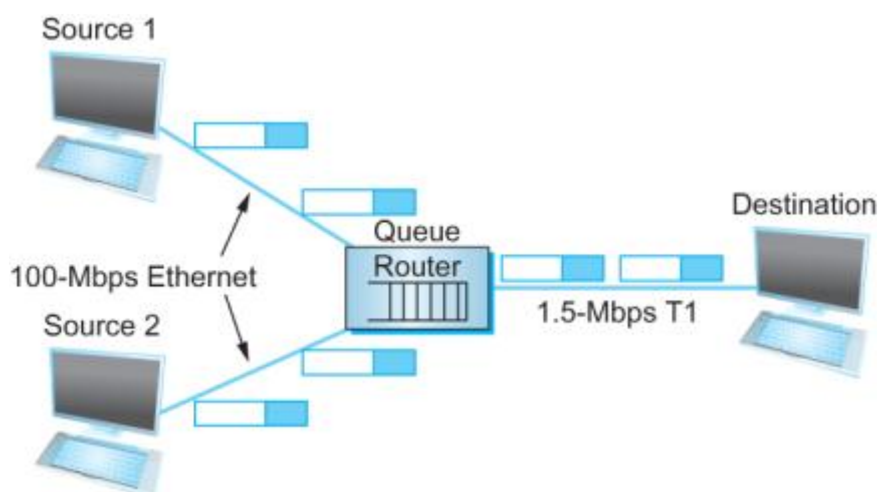
IP best-effort network

- Best-effort model
  - Everybody can send
  - Network does the best it can to deliver
  - Delivery not guaranteed, some traffic may be dropped



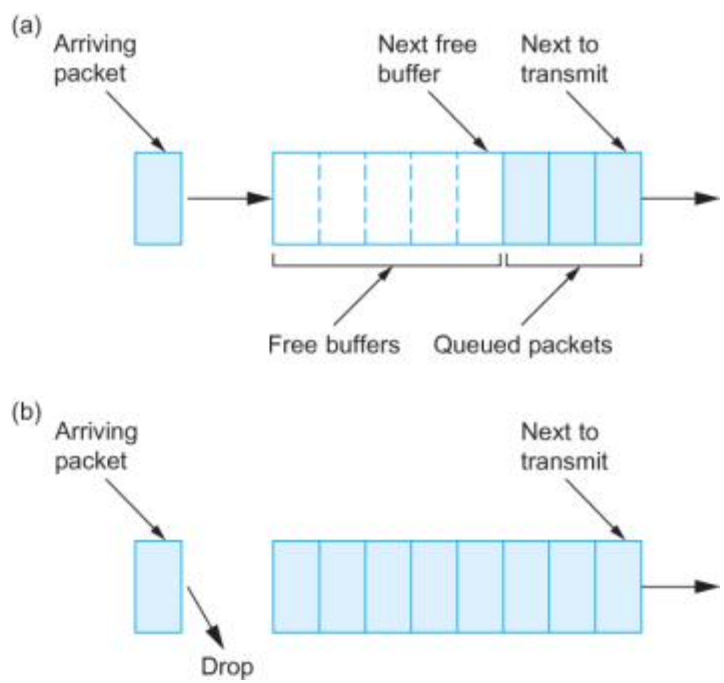
Congestion unavoidable

- Multiple packets arrive at same time
  - Router can only transmit one
  - Router has to buffer remaining
- If too many arrive in a short time window
  - Buffer may overflow
  - Router has to choose some packets to drop



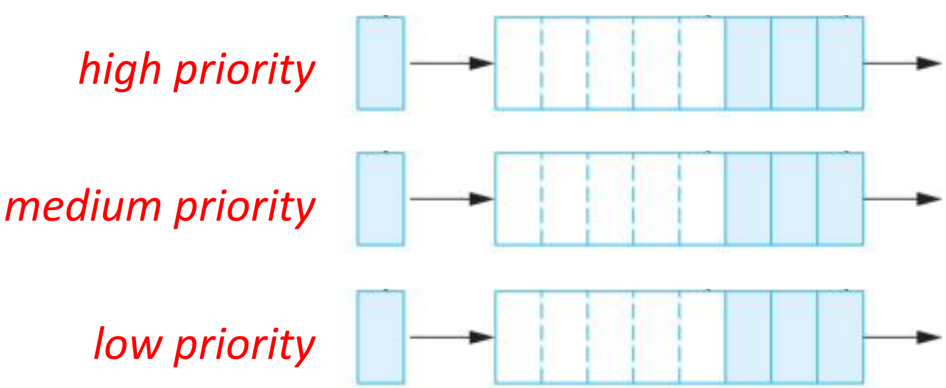
What routers do

- Too many packets arrive too quickly
  - Which packets should we drop?
- First-in first-out (FIFO) with tail drop
  - Simple, drop the new guy that doesn't fit in your buffer



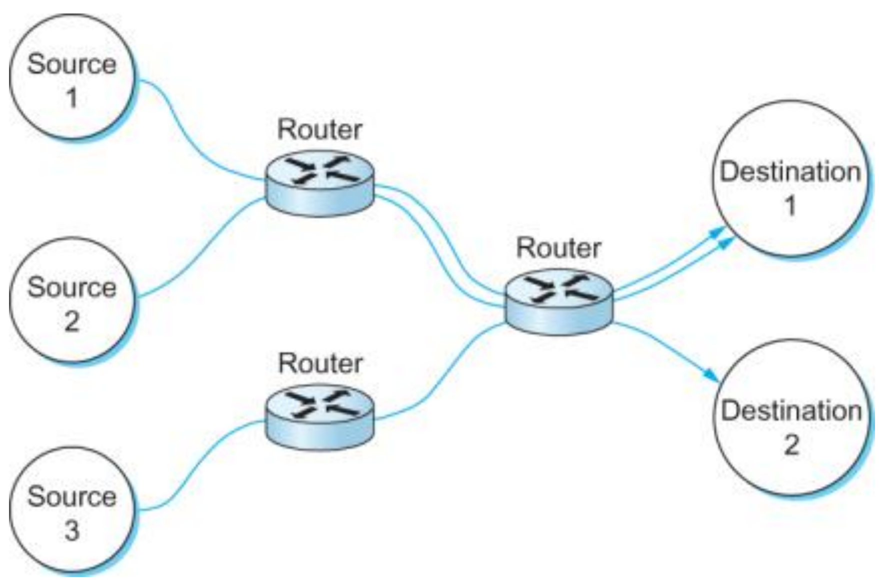
Queuing disciplines

- Priority queuing
  - Packets marked with priority in header
  - Multiple FIFO queues, one for each priority class
  - Transmit high priority queues first
  - Who is allowed to set priority bit?



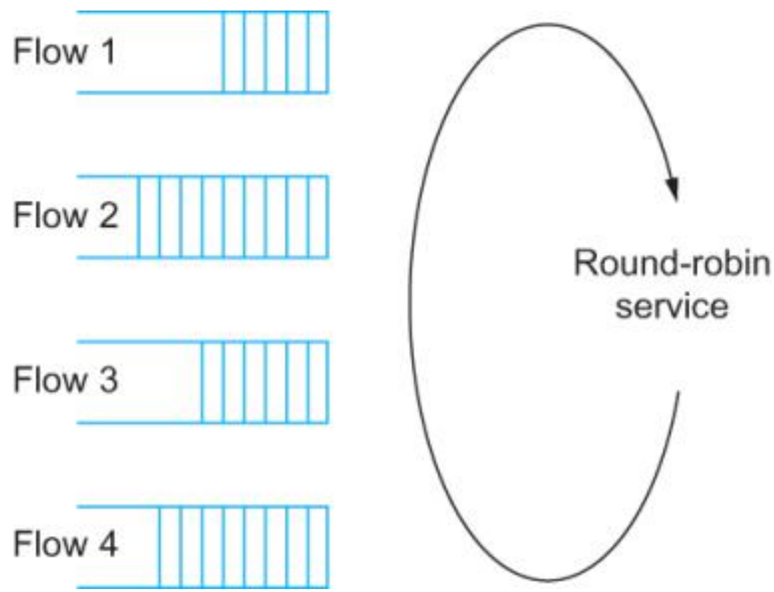
Network flows

- Connection flows
  - IP network is connectionless
  - Datagrams really not independent
  - Stream of datagrams between two hosts
  - Routers can infer current flows, "soft state"



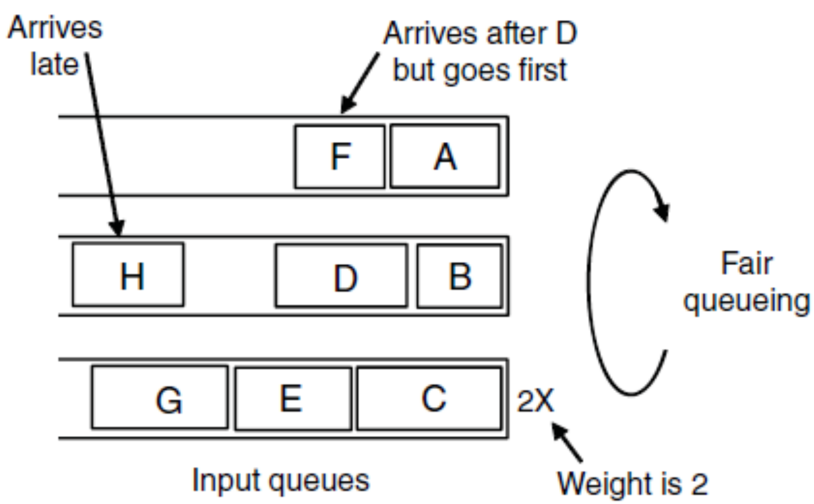
Fair queuing

- Use flows to determine scheduling
  - Prevent hosts from hogging all the router resources
  - Important if hosts don't implement host-based congestion control (e.g. TCP congestion control)
  - Each flow gets its own queue, served round-robin



Fair queuing

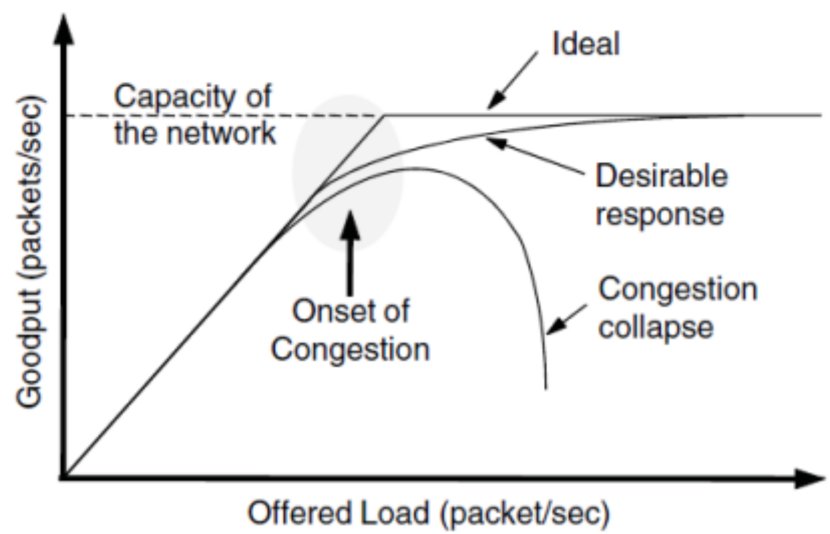
- Round-robin scheduling
  - Packets different lengths, approximate bit-level round-robin
  - Compute virtual finish time assuming each "round" drains byte from each queue
  - Sort in order of virtual finish time
  - Different flows might be assigned weights



Packet	Arrival time	Length	Finish time	Output order
A	0	8	8	1
B	5	6	11	3
C	5	10	10	2
D	8	9	20	7
E	8	8	14	4
F	10	6	16	5
G	11	10	19	6
H	20	8	28	8

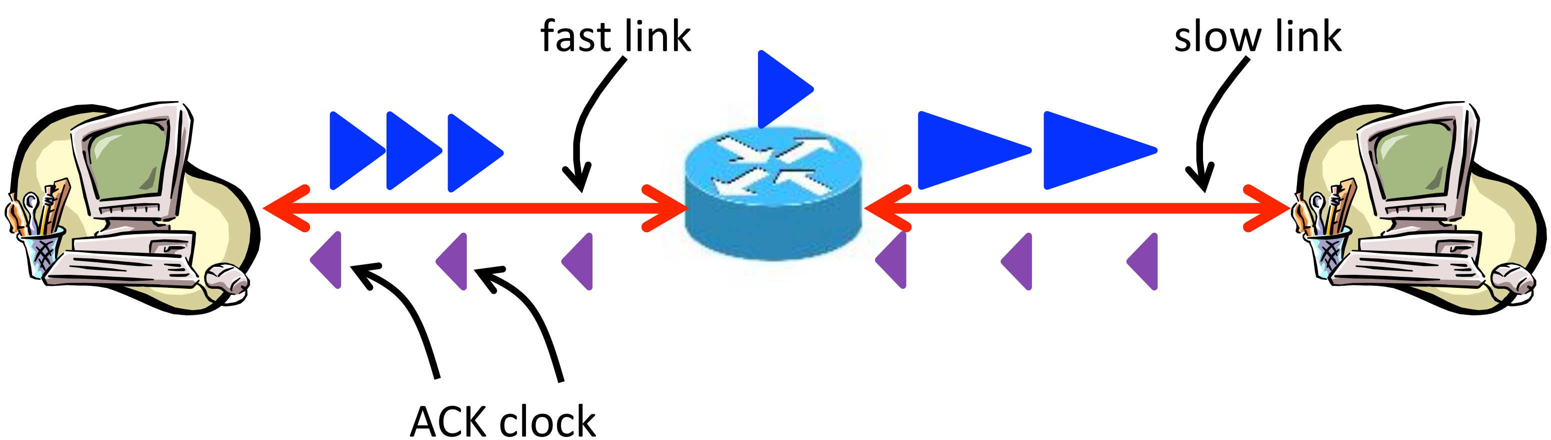
Congestion collapse

- Congestion collapse
  - 1986, NSF backbone dropped from 32 kbps to 40 bps
    - Hosts send packets as fast as advertised window allowed
    - When packets dropped, hosts retransmit causing more congestion
  - Goodput = useful bits delivered per unit time
    - Excludes header overhead, retransmissions, etc.



TCP congestion control

- TCP congestion control
  - Introduced by Van Jacobson in the late 80's
  - Done without changing headers or routers
  - Senders try and determine capacity of network
  - Implicit congestion signal: packet loss
  - ACK from previous packet determines when to send more data, "self-clocking"

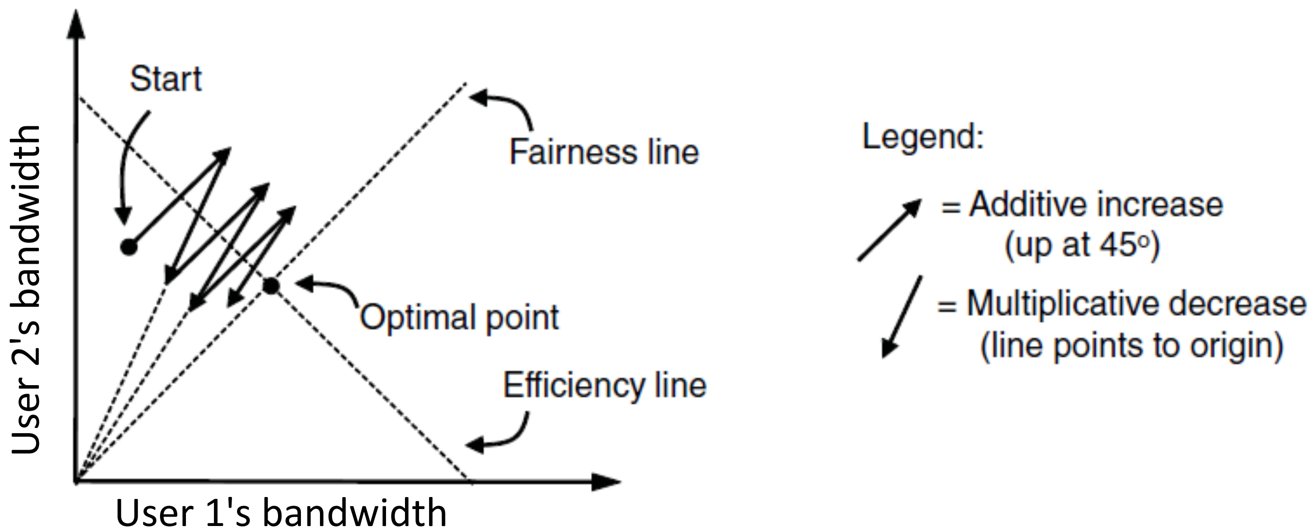


TCP congestion control

- Each TCP sender tracks:
  - Advertised window, for flow control
  - Congestion window, for congestion control
- Sender uses minimum of the two:
  - Advertised window prevent overrunning receiver's buffer
  - Congestion window present overloading network
- Situation is dynamic:
  - Network changes
    - e.g. new high bandwidth link, other hosts start/stop sending
  - Sender always searching for best sending rate

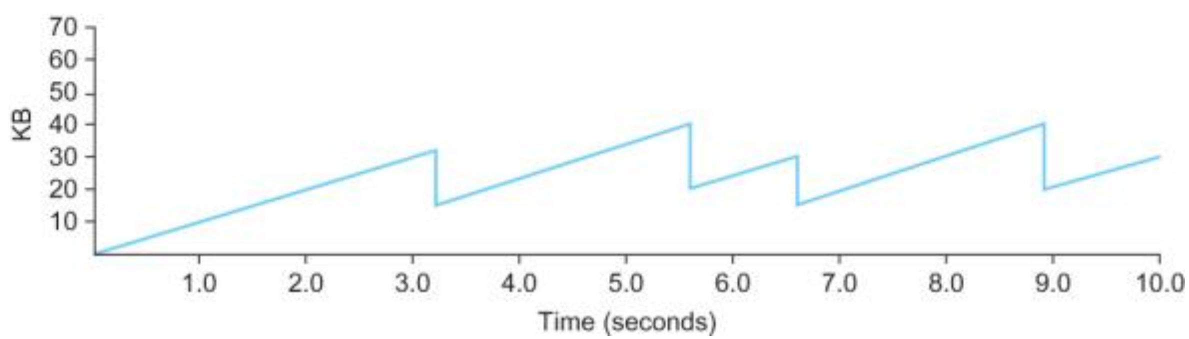
AIMD

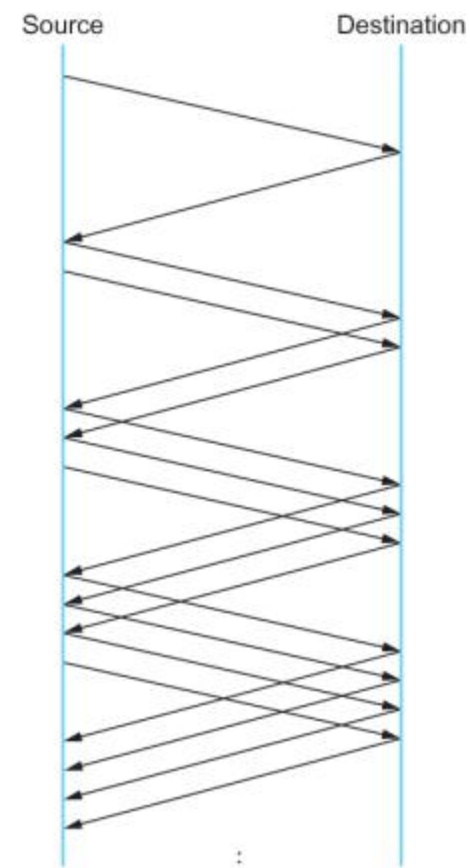
- Additive increase, multiplicative decrease (AIMD)
  - Additive increase: On success of last packet, increase window by 1 Max Segment Size (MSS)
  - Multiplicative decrease: On loss of packet, divice congestion window in half



Basic TCP congestion control

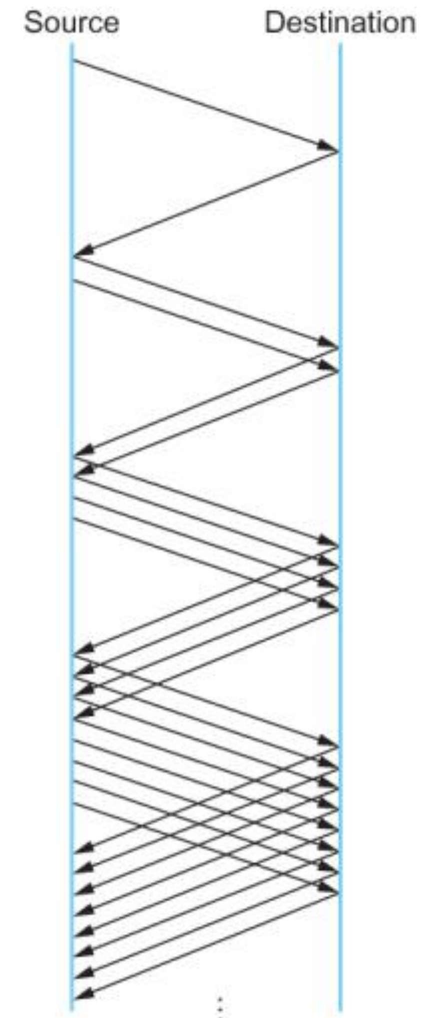
- Add one packet to window per RTT
  - Works well if we start near capacity
  - Otherwise could take a long time to discover real network capacity





Slow start

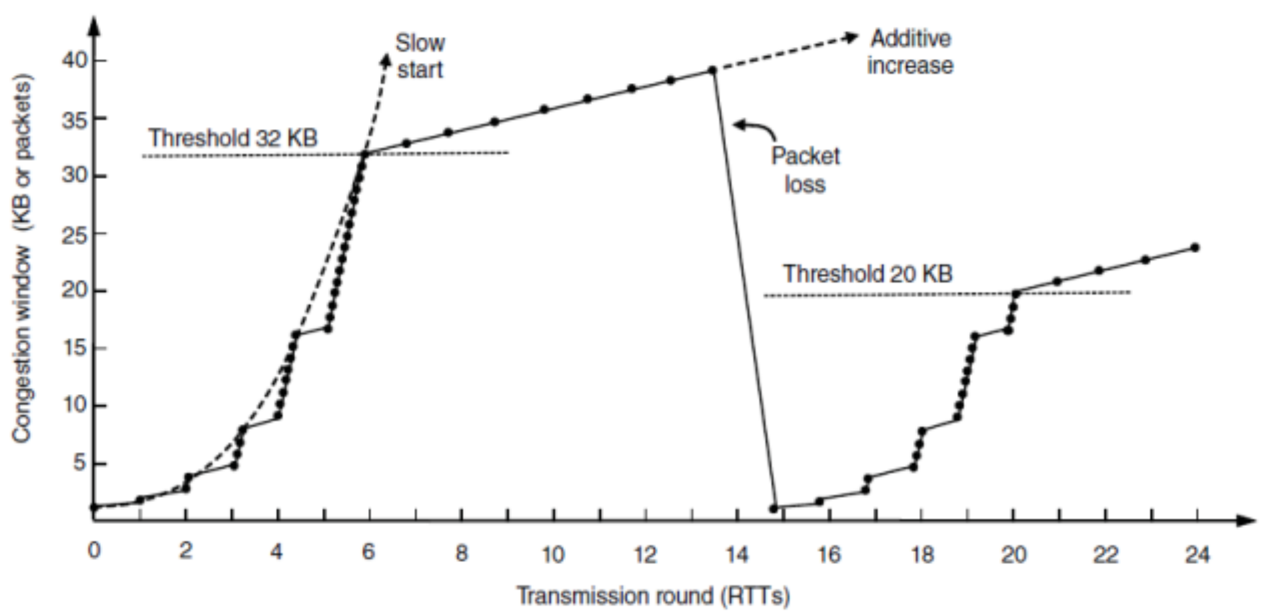
- Slow start
  - Increase congestion window rapidly from cold start of 1
  - Add one to window for every good ACK
    - Exponential increase in packets in flight
  - On packet loss, start over at 1
  - Slow in comparison to original TCP
    - Immediate sending up to advertised window (caused congestion collapse)



http://histrory.visualland.net/tcp_swnd.html

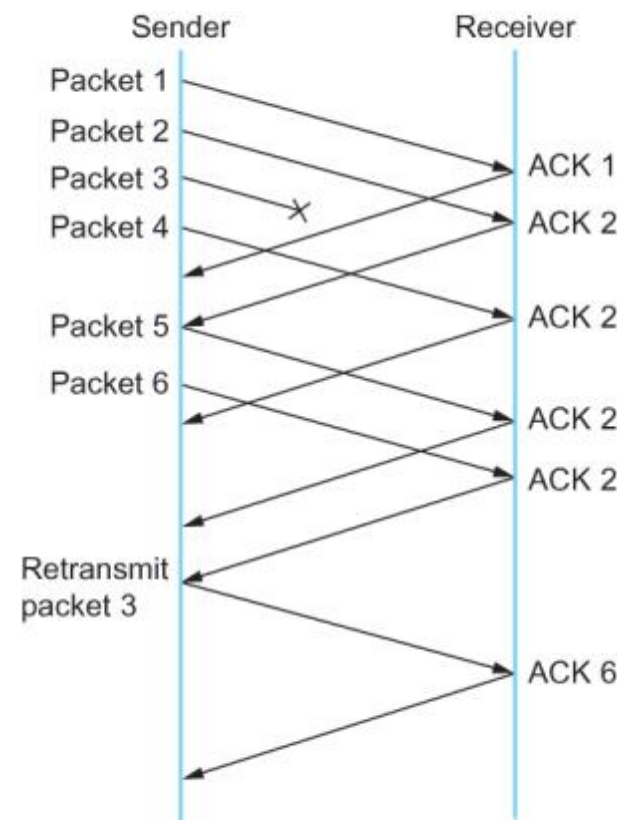
Slow start

- Congestion threshold (slow start threshold)
  - Initially set to large value
  - Updated on a multiplicative decrease
  - When we ramp up, switch to additive when we reach



Fast retransmission

- Problem: Timeouts take a long time
  - Connection sits idle waiting for a packet we are pretty sure is never going to be ACK'd
- Fast retransmission
  - Heuristic to retransmit packet we suspect was lost
  - Triggered when we observe 3 duplicate ACKs
  - 20% increase in throughput
- TCP "Tahoe"

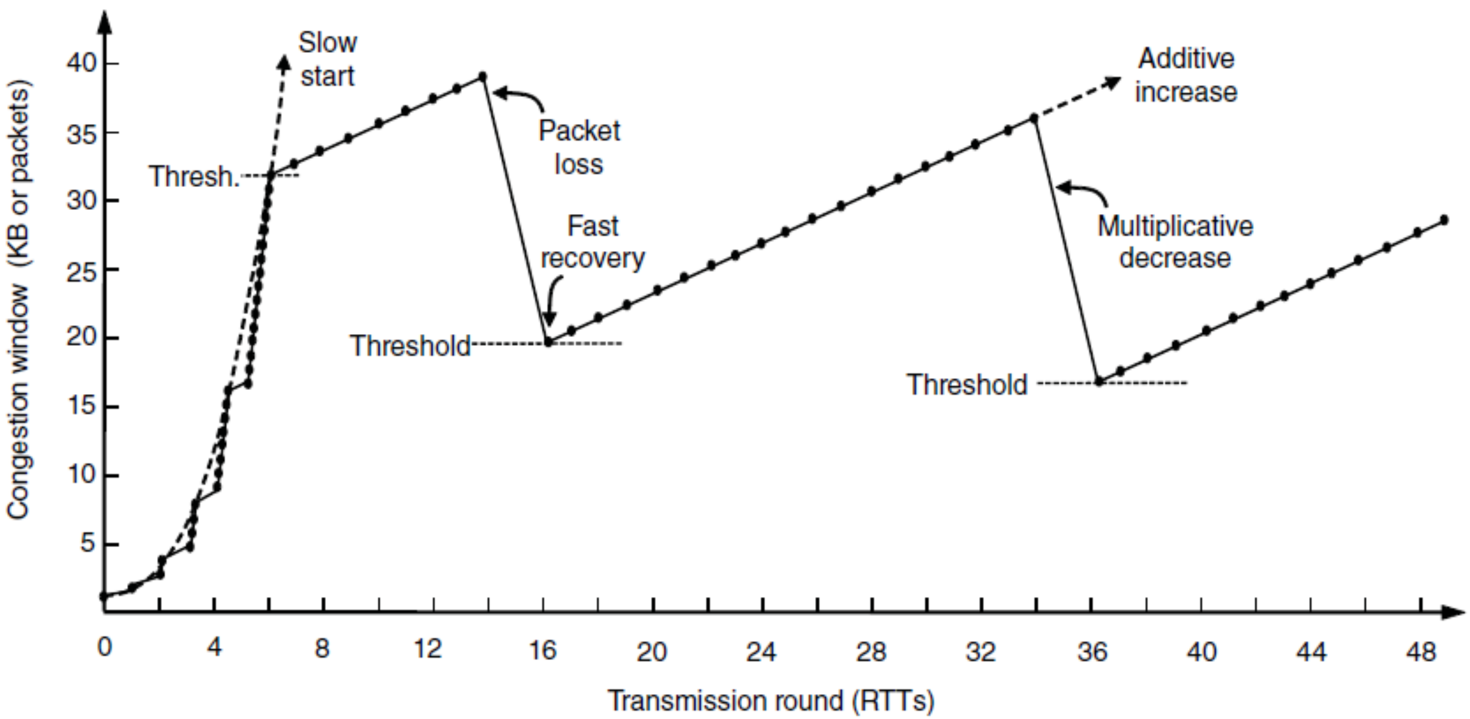


Fast recovery

- Problem: Restarting from 1 takes too long
  - We spend too long below "known" network limit
- Fast recovery
  - ACK clock is still working even though packet was lost
  - Count up dup ACKs (including 3 that triggered fast retransmission)
  - Once packets in flight has reached new threshold, start sending packet on each dup ACK
  - Once lost packet ACK's, exit fast recovery and start linear increase

Fast recovery

- "TCP Reno"
  - Tahoe + fast recovery



Wireless networks

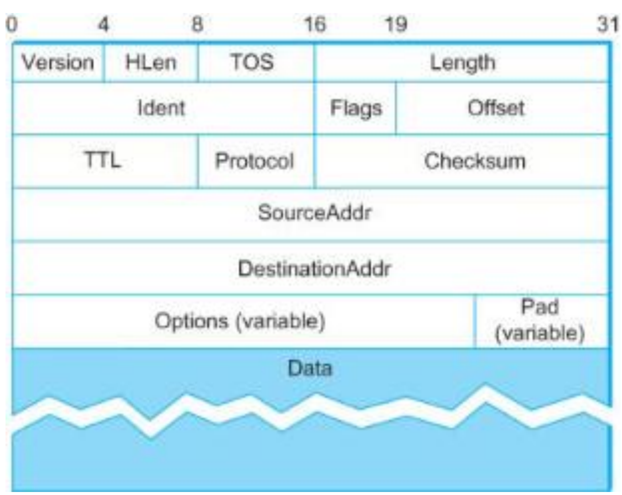
- TCP congestion control uses packet loss as signal
  - Wireless/satellite links = high error rate
  - TCP could think loss is due to congestion not bit errors
- Possible solutions:
  - Link layer acknowledgements and retransmission
  - Forward error correction
  - Split connection into wireless/wired segments
  - Use other signals than packet loss: increasing RTT

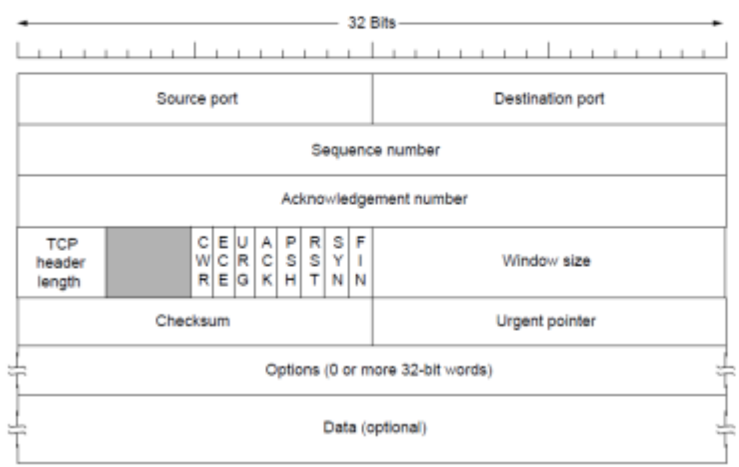
Control vs. avoidance

- Congestion control
  - Dealing with packet loss once it occurs
- Congestion avoidance
  - Attempt to control send rates before packets dropped
  - Explicit signal generated by routers
  - Implicit signal inferred by hosts
  - Currently not widely adopted

Router signaling

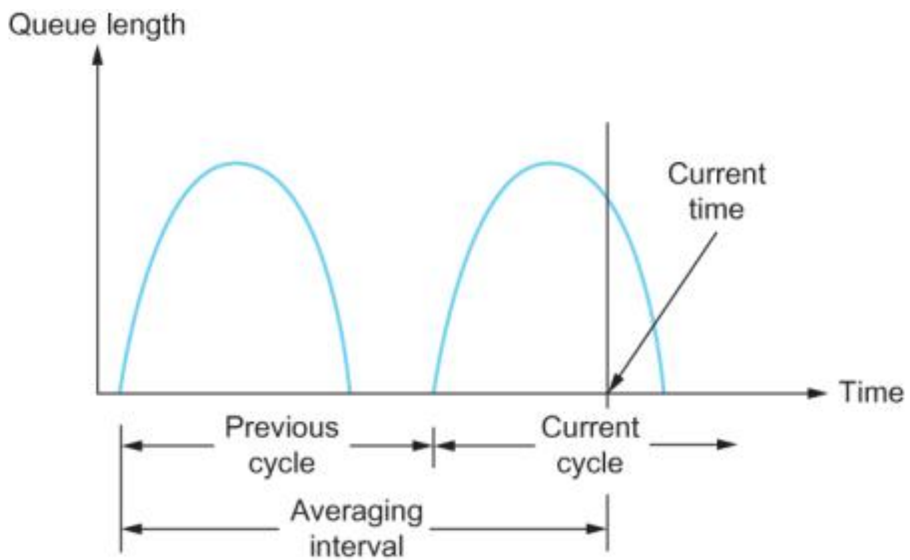
- Explicit Congestion Notification (ECN)
  - Sender sets TOS IP header bit saying it supports ECN
  - If ECN-aware router is congested, marks another TOS bit
  - TCP receiver sees IP congestion bit, informs sender via TCP segment ECN-Echo (ECE) bit
  - TCP sender confirms receipt of ECE with Congestion Window Reduced (CWR) bit





TCP congestion avoidance

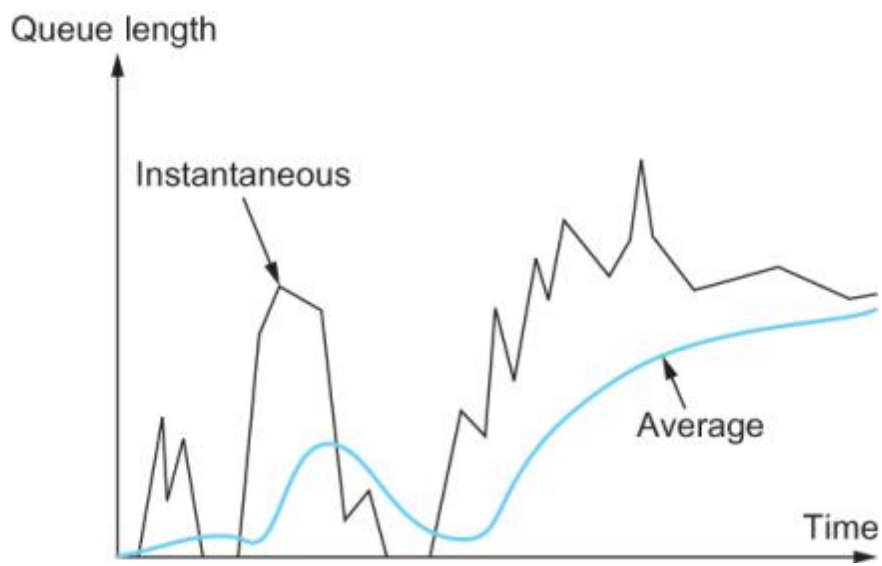
- How does router determine congestion?
  - Checks avg. queue length spanning last busy + idle cycle



- What does TCP sender do with congestion signals?
  - Checks fraction of last window's worth of packets
  - If < 50%, increase congestion window
  - If > 50%, decrease congestion window by 0.875

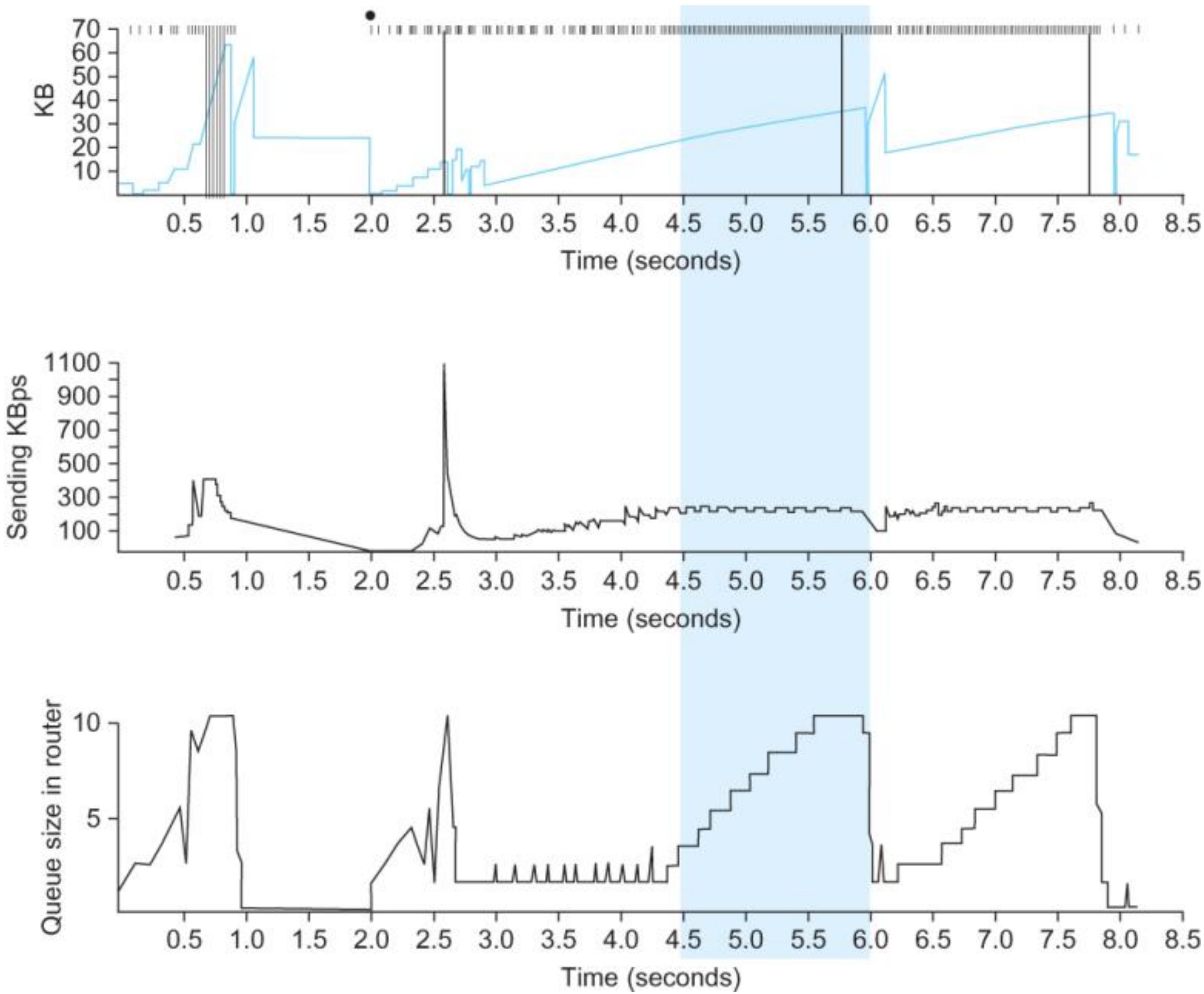
What if hosts don't support ECN?

- Random early detection (RED)
  - If router approaching congestion: drop a random packet
  - Source detects packet loss and can adjust send rate
  - Randomness approximates fairness since more likely to signal host sending lots of packets
  - Various parameters controlling drop behavior



Source based avoidance

- Hosts watch for signs of congestion
  - Adjust before packets actually dropped
  - Possible signals:
    - Increasing RTT
    - Flattening of sending rate
    - Changes in the sending rate



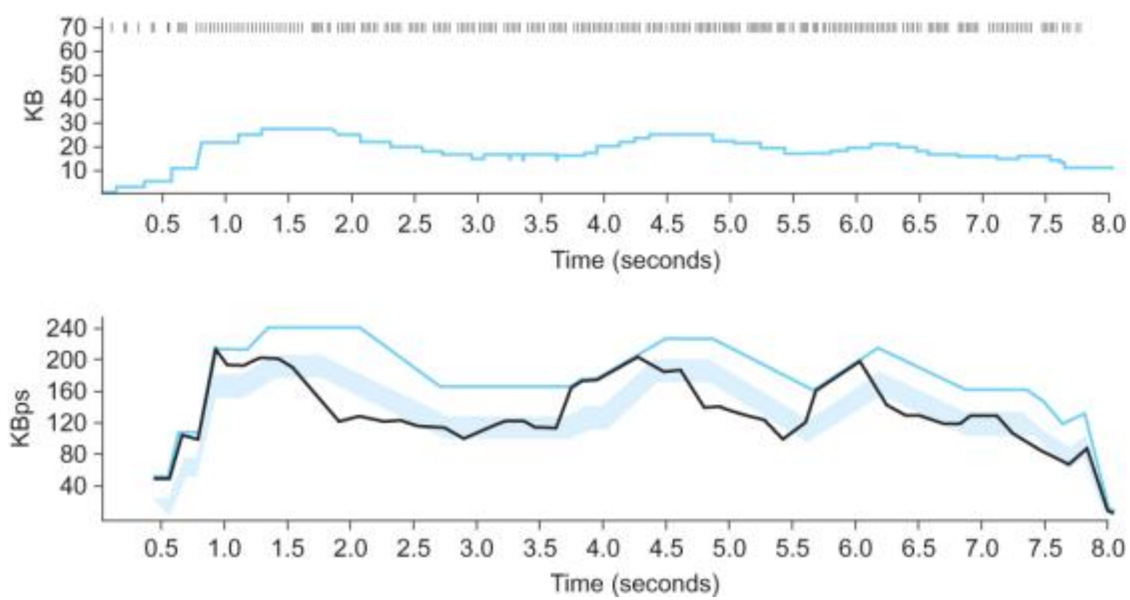
Congestion window

Observed throughput

Buffer space taken at router

Source based avoidance

- TCP "Vegas"
  - Monitor for signs of increasing congestion using RTT
    - Track minimum RTT
    - Measure actual rate for one RTT
    - Compare with expected rate (using minimum RTT), increase or decrease window linearly
    - Use multiplicative decrease if actual loss



Cheating

- Not everybody plays fair:
  - Run multiple TCP connections in parallel
  - Change the TCP implementation
    - Starts your TCP connection off with > 1 MSS
  - Use a protocol without congestion control (e.g. UDP)
  - Good guys slow down to make way so others can have unfair share of bandwidth
- Possible solutions?
  - Routers detect cheating and drop excess traffic
  - Fair queuing

Summary

- Network congestion
  - Too many packets, routers have to drop
  - Routers can do this in various ways
    - FIFO tail drop, fair queuing, Random Early Detection (RED)
- Congestion control
  - Senders use dropped packets as signal to slow down
  - TCP congestion control
    - Slow start, fast retransmission, fast recovery
- Congestion avoidance
  - Router signaling, e.g. ECN
  - Host monitoring, e.g. TCP "Vegas"