**Principle of congestion control**

Congestion:

* Informally: “too many sources sending too much data too fast for the network to handle”
* Different from flow control
* Manifestations:
  + Lost packets (buffer overflow at routers)
  + Long delays (queuing in router buffers)
* Flow is like making progress but congestion preventing bad things happening on the network that can affect your sending

Approaches towards congestion control

Two broad approaches towards congestion control:

End-end congestion control:

* No explicit feedback from network: network doesn’t tell you anything about whats going on e.g. timeouts, acknowledgement, what you see from the destination
* 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 (states): because you need to tell the senders about what is happening
  + Explicit rate for the sender to send at

Additive increase & multiplicative decrease (AMD)

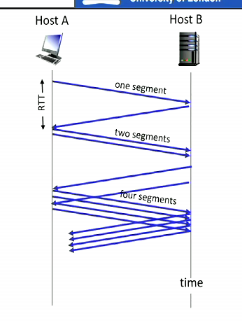
Approach: sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs

* Additive increase: increases cwnd (congestion window, a TCP state variable that limits the amount of data the TCP can send before receiving an ACK) by 1mss (could be anything, but slowly increase linearly) every RRT as long as no loss is detached (slowly increases)
* Multiplicative decrease: cut cwnd in half after loss

This is principle rather than implementation

Two phases in TCP congestion control:

TCP Slow Start

* When connection begins, increase rate exponentially until first loss event:
  + Initial cwnd = 1mss
  + Double cwnd ever RTT
  + Done by increment cwnd for every ACK received
* Summary: initial rate is slow but ramps up exponentially fast, it is just learning what the network can handle
* Send double segment for each ACK segment
* This is not additive increase, since it is not linear

Detecting and reacting to loss

* Loss indicated by timeout:
  + Cwnd set to 1mss;
  + window then grows exponentially (slow start) to threshold (ssthresh) then grows linearly (congestion avoidance). You restart the exponentially growth because you need to relearn what the network can do
* Loss indicated by 3 duplicate ACKs
  + The duplicate ACKs indicate that the network is capable of delivering some segments still. The network still works and does work but you have to cut the congestion window by half (cwnd); then it grows linearly

Phase transitions: switching between slow start and CA

Q: when should the exponential increase switch to linear?

A: when cwnd gets 1/2 of its value before timeout