Congestion Control

- Problem: network performs badly if it gets congested (slow delivery, packets dropped)
- solution: detect congestion, then send more slowly
- details:
  - how to detect congestion
  - how much to slow down
  - fairness

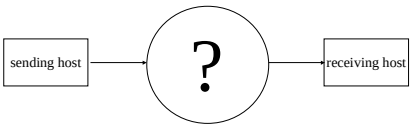
Approaches

- router-centric
  - routers tell hosts how much they can send
- host-centric
  - hosts figure out what to do
- TCP is mostly host-centric, but routers do some work too

Approaches

- reservation-based
  - hosts reserve capacity for each connection
  - routers provide the reserved level of service
- feedback-based
  - hosts learn how much capacity they can get at the moment
  - routers provide feedback to help them learn
    - » explicit: “please slow down”
    - » implicit: slowdown or timeout

Why is this hard?



Congestion Control: Goals

- slow down enough to alleviate congestion
- don’t slow down too much
- issue: fairness
  - each host makes its own decisions
  - Is the pain spread fairly?
- issue: compliance
  - virtuous hosts slow down, antisocial hosts keep going at full speed
  - tragedy of the commons

Routers and Congestion

- routers queue arriving packets until they can be forwarded
- queue sizes are finite, so queues can get full
  - limit on router memory
  - time proportional to queue length
- when queues fill, packets get dropped

Detecting Congestion

- TCP interprets a timeout as a symptom of congestion
  - good: doesn't require extra help from routers
  - bad: could be wrong; there are other causes of timeout
    - » mistake causes unneeded slowdown
    - » works OK as long as mistakes are fairly rare

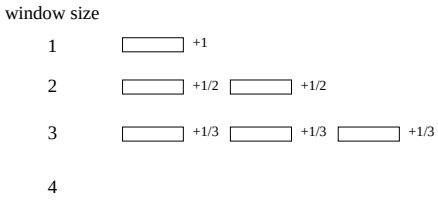
Window-Based Approach

- extension of TCP flow control
  - before: sender stayed within window size
    - » window size set by receiver
  - now: effective window size is minimum of
    - » flow control window (as before)
    - » congestion control window (new)
  - rationale: don't overwhelm the receiver, and don't overwhelm the network

Congestion Window

- strategy:
  - when all is well, increase window slowly
  - when congestion, decrease window rapidly
- additive increase
  - add C/window for each packet acked
    - » effect: increases by C for each window-sized batch
- multiplicative increase
  - cut window in half on timeout

Additive Increase



Sawtooth Behavior

Getting Started

- How to set congestion window for new connections?
  - start at 1: conservative, but usually too slow
  - start large: good for empty network, but overwhelms busy network
    - » made worse by manual restart of application
- better strategy: increase quickly from zero, to discover the right level

"Slow Start" Policy

- actual TCP algorithm:
  - start window at 1
  - increase window exponentially until first timeout
    - » add C to window for each successful packet
  - then revert to normal policy
- called "slow start" in comparison to starting with large window; but faster than starting with small window

"Slow Restart"

- another problem: after timeout, large burst of retransmissions all at once
  - causes temporary congestion
- solution: after packet drop:
  - reset window to 1
  - increase multiplicatively
  - on reaching half of pre-timeout window, switch over to additive increase

Behavior with Slow Start

How Well Does It Work?

- avoids disasters, even under heavy load
  - previous TCP versions didn't
- parameters and design carefully tuned via experience
- still some detrimental effects
  - network "hits the wall" suddenly
  - antisocial hosts might not slow down

Congestion Avoidance

- current scheme slows down when queues get *full*
- performance is better when queues are partially full
  - too empty: wasted chances to send
  - too full: long delays
- idea: try to slow down before queues get all the way full

DECbit Congestion Avoidance

- congestion bit in packet header
- router sets congestion bit if queue is long when packet arrives
- congestion bit echoed back in acknowledement
- sender slows down on receiving acks with congestion bit set
  - slow down a little bit
- good, but not widely used in Internet

RED Routers

- RED = Random Early Detection
- idea: when queue gets pretty full, router randomly drops a small fraction of packets
  - fraction depends on fullness of queues
- causes slight slowdown in sending rate, keeping queue length at best level
- can work in TCP with unmodified end hosts
- used on some Internet routers now

Host-Based Approach

- idea: measure effective bandwidth
  - normally, larger window should increase effective bandwidth
  - under congestion, larger window doesn't increase effective bandwidth
- slow down a little when window-size increases don't increase performance
- changes sending host TCP software only
- implemented in "TCP Vegas"

Congestion Control and the Web

- HTTP (Web browsing) protocol originally used a separate TCP connection for each file.
  - result: many short-lived connections
- congestion control relies on history of a connection
  - result: poor congestion control
  - result: serious Internet congestion
- fix: re-use connections

Coping with Antisocial Behavior

- problem: somebody who doesn't slow down gets an unfair share of resources
- many complicated solutions exist
- simple solution 1
  - pay for bandwidth, always get it
  - now antisocial behavior is the ISP's problem
  - at best a partial solution

Simple Solution 2

- divide packets into "compliant" and "non-compliant"
  - distinguished by bit in header
- routers drop non-compliant packets first
- sender buys the right to send a certain number of compliant packets per second
  - ISP checks, relabels packets that exceed limit
- ISPs apply same rules to each other