EECS 122, Lecture 21

Today’s Topics:
Congestion Control Metrics
TCP Congestion Control

Kevin Fall, kfall@cs.berkeley.edu

Evaluation Criteria
- Effectiveness
  - want to fully utilize links in network, but filling all queues increases end-to-end delay
  - how to measure throughput/delay tradeoff?
- Fairness
  - how do multiple flows share a common network?
  - if we assume fair means equal, how to measure if a set of flows are receiving equal treatment?

Effectiveness
- Throughput/delay tradeoff
  - with stat muxing (and a work-conserving service discipline), outgoing link is always fully utilized if any packet present
  - want to avoid empty queues, but larger queues mean larger delays
- Network power:
  - Power = (Throughput) / (Delay)
  - 0 < α < 1

Network Power
Jain’s Fairness Index
- A definition for fairness:
  - $0 \leq f() \leq 1$, given flow throughputs $x$
  - locally equal partitioning of bandwidth achieves index of 1. If only $k$ of $n$ flows receive equal bw (and others get none), index is $k/n$
  - what about different-length flows? (p.401)

Properties of the Index
- population size independence
- scale and metric independent
- bounded on $[0..1]$
- continuous

Allocation between A & B

Congestion Control with TCP
- Congestion control added to TCP in late 80s as a result of congestion collapse problem
- Idea:
  - host figures out how many packets it can safely inject into network
  - each received indicates 1 (or possibly more) packets have been removed from network, allowing host to inject another
  - self-clocking property ensures stability

Challenges for TCP
- How to determine how many packets to inject into network?
  - Too many: overrun buffers
  - too few: underutilization of link
- Additional problems:
available bandwidth changes over time as new connections start and terminate

**Congestion Window Maintenance**

- TCP maintains a congestion window (cwnd), based on packets
- Sender’s window limited to MIN(receiver’s window, cwnd)
- Maintenance policy:
  - on congestion signal, multiplicative decrease
  - on success, additive increase
- Additive increase/multiplicative decrease produces stability [CJ 89]

**Window Increase/Decrease**

- TCP Congestion Avoidance:
  - use packet loss as indicator of congestion
  - on loss, divide cwnd by 2
  - on successful ACK, increase cwnd by 1/cwnd
- Results in window growth of 1 packet for each window’s worth of ACKs [linear]

**TCP Congestion Avoidance**

- TCP Congestion Avoidance makes sense when the connection is operating near capacity (in steady-state)
- What about when a connection starts up, or there has been a long pause (where the state of the
world may have changed)?
• Need a way to get to equilibrium ... next time ...