CSC2209
Computer Networks

Congestion Control (end-host view)

Stefan Saroiu
Computer Science
University of Toronto
Today’s Questions

• How fast should a transmitter send data?
  – Not too slow….
  – Not too fast…
  – Just about right!

• Shouldn’t be faster than receiver can process
  – This is called…

• Shouldn’t be faster than network can process
  – This is called…
Congestion Control Goals
Congestion Control Goals

• Efficiency
  – Utilize all available bandwidth

• Fairness
  – All hosts get equal access to bandwidth

• Distributed implementation
  – Only require state at endpoints

• Convergence
  – For constant load, arrive at single solution for using/sharing bandwidth
Questions

• How to detect congestion?

• How to limit sending data rate?

• How fast to send?

• How to achieve stability?
Detecting Congestion
Detecting Congestion

- **Implicit signaling**
  - Packet loss
    - Assumes congestion is primary cause of packet loss
  - Packet delay
    - RTT increases as packets queue
    - Packet inter-arrival time is a function of bottleneck link
    - Pros/cons?

- **Explicit signaling**
  - Source quench: router sends ICMP “Hey buddy, slow down”
  - ECN: router marks packet on how full queue is
  - Hop-by-hop backpressure
Questions

• How to detect congestion?

• How to limit sending data rate?

• How fast to send?

• How to achieve stability?
How to Limit Sending Rate?
How to Limit Sending Rate?

- Window-based (TCP)
  - Artificially constrain number of outstanding packets allowed
  - Increase window to send faster, decrease to send slower
  - Pro: cheap to implement, good failure properties
  - Cons:
How to Limit Sending Rate?

- **Window-based (TCP)**
  - Artificially constrain number of outstanding packets allowed
  - Increase window to send faster, decrease to send slower
  - Pro: cheap to implement, good failure properties
  - Cons: creates bursty traffic
How to Limit Sending Rate?

- **Window-based (TCP)**
  - Artificially constrain number of outstanding packets allowed
  - Increase window to send faster, decrease to send slower
  - Pro: cheap to implement, good failure properties
  - Cons: creates bursty traffic

- **Rate-based**
  - Two parameters (period, packets)
  - Send x packets in period y
  - Pro: smooth traffic
  - Cons:
How to Limit Sending Rate?

• Window-based (TCP)
  – Artificially constrain number of outstanding packets allowed
  – Increase window to send faster, decrease to send slower
  – Pro: cheap to implement, good failure properties
  – Cons: creates bursty traffic

• Rate-based
  – Two parameters (period, packets)
  – Send x packets in period y
  – Pro: smooth traffic
  – Cons: per connection timers, what if receiver fails
Questions

• How to detect congestion?

• How to limit sending data rate?

• How fast to send?

• How to achieve stability?
How fast to send?

• Ideally: keep equilibrium at “knee” of power curve
  – Find “knee” somehow
  – Keep number of packets in flight the same
  – Don’t inject new packet until old one left the network
  – What if you guessed wrong?

• Compromise: adaptive approximation
  – If congestion signaled, reduce sending rate by x
  – If data delivered successfully, increase sending rate by y
  – How should x and y be related? Convergence?
Questions

• How to detect congestion?

• How to limit sending data rate?

• How fast to send?

• How to achieve stability?
How to achieve stability?

• Additive increase, multiplicative decrease (AIMD)
  – Increase sending rate by a constant (e.g., 1500 bytes)
  – Decrease sending rate by a linear factor (e.g., divide by 2)

• Rough intuition why this works
  – Let \( L_i \) be length of queue at time \( i \)
  – In steady state: \( L_i = N \), where \( N \) is a constant
  – During congestion: \( L_i = N + yL_{i-1} \), where \( y > 0 \)
  – If \( y \) is large (close to 1), queue size increases exponentially
Resulting TCP/IP Improvements

- Slow-start
- Round-trip time variance estimation
- Exponential retransmit timer backoff
- More aggressive receiver ack policy
- Dynamic window sizing on congestion
- Clamped retransmit backoff (Karn)
- Fast Retransmit

Congestion control means: “Finding places that violate the conservation of packets principle and then fixing them.”
Slow-start

- Goal: find equilibrium sending rate

- Quickly increase sending rate until congestion detected

- Algorithm:
  - On new connection, or after timeout, set cwnd=1
  - For each segment acknowledged, cwnd += 1
  - If timeout then cwnd /= 2, set ssthresh = cwnd
  - If cwnd >= ssthresh then exit slow start

- Very confusing name
Adaptive Timing

• How long should we wait for a packet’s acknowledgement
  – Too short: spurious timeouts and retransmissions
  – Too long: wasteful

• Old TCP
  – Maintain weighted average of RTT samples: R
  – Timeout set to B*R, where B=2
  – Under high load, this scheme doesn’t reflect variation

• Jacobson’s contributions
  – Estimate variation, B based on some samples
  – After loss, increase timeout exponentially (by 2)
Fast Retransmit & Fast Recovery

• Fast retransmit
  – Timeouts are slow (1 second is shortest TCP timeout)
  – When packet is lost, receiver still acks last in-order packet
  – Use 3 duplicate ACKs to indicate loss
    • Why 3? When wouldn’t this work?

• Fast recovery
  – If ACKs are still arriving, then no need for slow start
  – Divide cwnd by 2 after fast retransmit
  – Increment cwnd by 1 for each duplicate ACK
A TCP Taxonomy

- TCP Tahoe (1988)
  - Slow-start, fast retransmit, congestion avoidance
- TCP Reno (1990)
  - Tahoe + fast recovery
- TCP New-Reno (1996)
  - Reno + partial ACKs
- SACK TCP (1996)
  - Selective acknowledgements
- TCP Vegas (1993)
  - Uses RTT variation to measure congestion
- TCP BIC (2004)
  - Binary search between $W_{\text{min}}$ and $W_{\text{max}}$
Congestion Control in POTS

• How is it done?
Congestion Control in POTS

- How is it done?
  - Pricing
  - Call doesn’t get through (admission control)
Short Connections

• How are short connections affected by slow-start?
  – What happens if drop in slow-start?
  – What happens when SYN dropped?
• Bottom line: which packet gets dropped matters a lot!
• Are most flows long or short?
Cooperation

• TCP is designed around the premise of cooperation
  – What if receiver lies about receiving packets?

• Does over-provisioning always help?
  – Let’s look at router queues