Lecture 13 – Congestion Control
Eric Anderson
Peter Steenkiste
Fall 2014
www.cs.cmu.edu/~prs/15-441-F14

Outline
• Congestion control fundamentals
  • Challenges
  • Basic mechanisms
• TCP congestion control
  • TCP slow start

Internet Pipes?
• How should you control the faucet?

Internet Pipes?
• How should you control the faucet?
  • Too fast – sink overflows!
Internet Pipes?

• How should you control the faucet?
  • Too fast – sink overflows!
  • Too slow – what happens?

• Goals
  • Fill the bucket as quickly as possible
  • Avoid overflowing the sink
  • Solution – watch the sink

Plumbers Gone Wild!

• How do we prevent water loss?
  • Know the size of the pipes?

• Now what?
  • Feedback from the bucket or the funnels?
Congestion

• Different sources compete for resources inside network
• Why is it a problem?
  - Sources are unaware of current state of resource
  - Sources are unaware of each other
  - Coordinate all nodes in the Internet?
• Manifestations:
  - Lost packets (buffer overflow at routers)
  - Long delays (queuing in router buffers)
  - Can result in throughput less than bottleneck link (1.5Mbps for the above topology) → a.k.a. congestion collapse

Causes & Costs of Congestion

• Four senders – multihop paths
• Timeout/retransmit

Q: What happens as rate increases?

Causes & Costs of Congestion

• When packet dropped, any ″upstream transmission capacity used for that packet was wasted!″

Congestion Collapse

• Definition: Increase in network load results in decrease of useful work done
• Many possible causes
  - Spurious retransmissions of packets still in flight
    - Classical congestion collapse
    - How can this happen with packet conservation
    - Solution: better timers and TCP congestion control
  - Undelivered packets
    - Packets consume resources and are dropped elsewhere in network
    - Solution: congestion control for ALL traffic
Outline

• Congestion control fundamentals
  • Challenges
  • Basic mechanisms
• TCP congestion control
• TCP slow start

Approaches Towards Congestion Control

• Two broad approaches towards congestion control:
  End-to-end congestion control:
  • No explicit feedback from network
  • 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 (SNA, DECbit, TCP/IP ECN, ATM)
    • Explicit rate sender should send at (ATM)
  • Problem: makes routers more complicated
    • Per-flow state → poor scalability
    • Can sometimes be avoided

Congestion Control and Avoidance

• A mechanism that:
  • Uses network resources efficiently
  • Preserves fair network resource allocation
  • Prevents or avoids collapse
• Congestion collapse is not just a theory
  • Has been frequently observed in many networks

Congestion Control with Binary Feedback (TCP)

• Very simple mechanisms in network
  • FIFO scheduling with shared buffer pool
  • Feedback through packet drops (or binary feedback)
• TCP interprets packet drops as signs of congestion and sender slows down
  • This is an assumption: packet drops are not a sign of congestion in all networks, e.g., wireless networks
• Sender periodically probes the network to check whether more bandwidth has become available
• Key questions: how much to reduce (after a drop) and increase (when probing) rate
Objectives

- Simple router behavior
- Distributedness
- Efficiency: $X = \sum x_i(t)$
- Fairness: $(\sum x_i)^2/n(\sum x_i^2)$
  - What are the important properties of this function?
- Convergence: control system must be stable

Linear Control

- Many different possibilities for reaction to congestion and probing
  - Examine simple linear controls
    - $\text{Window}(t + 1) = a + b \text{Window}(t)$
    - Different $a/b_i$ for increase and $a/d_i$ for decrease
- Supports various reaction to signals
  - Increase/decrease additively
  - Increased/decrease multiplicatively
  - Which of the four combinations is optimal?

Phase Plots

- Simple way to visualize behavior of competing connections over time
- Sequence of steps with 2 synchronized senders

Phase Plots

- What are desirable properties?
- What if flows are not equal?
Additive Increase/Decrease

- Both $X_1$ and $X_2$ increase/decrease by the same amount over time
- Additive increase improves fairness and additive decrease reduces fairness

Muliplicative Increase/Decrease

- Both $X_1$ and $X_2$ increase by the same factor over time
- Extension along line through origin
- Constant fairness

Achieving Fairness AND Efficiency

- $a > 0$ and $b > 1$
- $a < 0$ and $b < 1$
- $a < 0$ and $b > 1$
- $a > 0$ and $b > 1$

What is the Right Choice?

- Constraints limit us to AIMD
  - Can have multiplicative term in increase (MAIMD)
  - AIMD moves towards optimal point
Outline

- Congestion control fundamentals
- TCP congestion control
  - Implementing AIMD
  - Packet pacing
  - Fast recovery
- TCP slow start

TCP Congestion Control: Implicit Feedback and AIMD

- Distributed, fair and efficient
- Packet loss is seen as sign of congestion and results in a multiplicative rate decrease: factor of 2
- TCP periodically probes for available bandwidth by increasing its rate: by one packet per RTT

Implementation Issue

- Operating system timers are very coarse – how to pace packets out smoothly?
- Implemented using a congestion window that limits how much data can be in the network.
  - Similar to using a flow control window to avoid flooding receiver
  - TCP also keeps track of how much data is in transit
- Data can only be sent when the amount of outstanding data is less than the congestion window.
  - The amount of outstanding data is increased on a “send” and decreased on “ack”
  - (last sent – last acked) < congestion window
- Window limited by both congestion and buffering
  - Sender’s maximum window = Min (advertised window, cwnd)

Packet Conservation

- At equilibrium, inject packet into network only when one is removed
  - Controlled by sliding window, not rate
  - But still need to avoid sending burst of packets → would overflow links
    - Need to carefully pace out packets
    - Helps provide stability
- Need to eliminate spurious retransmissions
  - Accurate RTO estimation
  - Better loss recovery techniques (e.g. fast retransmit)
TCP Packet Pacing

- Congestion window helps to “pace” the transmission of data packets
- In steady state, a packet is sent when an ACK is received
  - Data transmission remains smooth, once it is smooth
  - Self-clocking behavior

Congestion Avoidance

- If loss occurs when cwnd = W
  - Network can handle 0.5W ~ W segments
  - Set cwnd to 0.5W (multiplicative decrease)
- Upon receiving ACK
  - Increase cwnd by (1 packet)/cwnd
    - What is 1 packet? → 1 MSS worth of bytes
    - After cwnd packets have passed by → approximately increase of 1 MSS
  - Implements AIMD
Remember Fast Retransmit?

Fast Recovery

- With fast retransmit, TCP can often avoid timeout, but loss signals congestion → cut window in half
- Challenge: how do we maintain ack clocking?
- Observation: each duplicate ack notifies sender that a single packet has cleared the network
- When < new cwnd packets are outstanding
  - Allow new packets out with each new duplicate acknowledgement
- Behavior
  - Sender is idle for some time – waiting for ½ cwnd worth of dupacks
  - Transmits at original rate after wait with ack clocking

Outline

- TCP connection setup/data transfer
- TCP congestion avoidance
- TCP slow start
Reaching Steady State

- Doing AIMD is fine in steady state but how do we get started …
- How does TCP know what is a good initial rate to start with?
  - Should work both for a CDPD (10s of Kbps or less) and for supercomputer links (10 Gbps and growing)
  - Need quick initial phase to help TCP get up to speed
- Also, after a timeout, the "pipe has drained"
  - cwnd = 0.5 * cwnd
  - How do we restart ACK clocking?

Slow Start Packet Pacing

- How do we get this clocking behavior to start?
  - Initialize cwnd = 1
  - Upon receipt of every ack, cwnd = cwnd + 1
  - Packet loss means you are going too fast
    - Hopefully Fast Retransmit works!
- Allows TCP to quickly find a good window size
  - Exponential increase!
  - Reaches W in RTT * log₂(W)
  - Also starts packet pacing
- How is this slow?

Starting of Packet Pacing

- CC window increases by 1 packet/ACK
- Queuing separates packet pair
- This repeats each RTT
- Until pipe is full

Slow Start Sequence Plot

- Sequence No
- Time
- Packets
- Acks
TCP Sawtooth Behavior

- Time
- Congestion Window
- Initial Slowstart
- Slowstart to pace packets
- Fast Retransmit and Recovery
- Timeouts may still occur

Important Lessons
- TCP state diagram → setup/teardown
- TCP timeout calculation → how is RTT estimated
- Modern TCP loss recovery
  - Why are timeouts bad?
  - How to avoid them? → e.g. fast retransmit