Lecture 21: TCP CC and Fluid flow

CSE 123: Computer Networks
Alex C. Snoeren

HW 4 due Mon 12/4
Lecture 21 Overview

- TCP Congestion Control
- Fluid Flow
- Quality of Service
TCP Bandwidth Probing

TCP uses AIMD to adjust congestion window
- Converges to fair share of bottleneck link
- Increases modestly in good times
- Cuts drastically in bad times

But what rate should a TCP flow use initially?
- Need some initial congestion window
- We’d like to TCP to work on all manner of links
- Need to span 6+ orders of magnitude, e.g., 10 K to 10 Gbps.
- Starting too fast is catastrophic!
Goal: quickly find the equilibrium sending rate

Quickly increase sending rate until congestion detected
- Remember last rate that worked and don’t overshoot it

TCP Reno Algorithm:
- On new connection, or after timeout, set $cwnd=1$ MSS
- For each segment acknowledged, increment $cwnd$ by 1 MSS
- If timeout then divide $cwnd$ by 2, and set $ssthresh = cwnd$
- If $cwnd \geq ssthresh$ then exit slow start

Why called slow? Its exponential after all…
Slow Start Example

Sender

Receiver

$cwnd=1$

Ack 2

$cwnd=2$

2

3

Ack 3

Ack 4

$cwnd=4$

4

5

Ack 5

Ack 6

Ack 7

Ack 8

$cwnd=8$

$cwnd$ vs. round-trip times

CSE 123 – Lecture 21: TCP CC & Fluid Flow
Basic Mechanisms

Slow Start + Congestion Avoidance

- **cwnd**
- **round-trip times**

- **Timeout**
- **Congestion avoidance**
- **ssthresh**
- **Slow start**
Fast Retransmit & Recovery

- Fast retransmit
  - Timeouts are slow (default often 200 ms or 1 second)
  - When packet is lost, receiver still ACKs last in-order packet
  - Use 3 duplicate ACKs to indicate a loss; detect losses quickly
    » Why 3? When wouldn’t this work?

- Fast recovery
  - Goal: avoid stalling after loss
  - If there are still ACKs coming in, then no need for slow start
  - If a packet has made it through -> we can send another one
  - Divide $cwnd$ by 2 after fast retransmit
  - Increment $cwnd$ by 1 MSS for each additional duplicate ACK
Fast Retransmit Example

Fast recovery (increase cwnd by 1)

Fast retransmit
The image contains a graph displaying the concepts of Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery in the context of TCP congestion control. The horizontal axis represents round-trip times, while the vertical axis shows the change in cwnd (cumulative window size). The graph illustrates how these mechanisms interact over time, with annotations indicating points of Fast Recovery.
Short Connections

- Short connection: only contains a few pkts
- How do short connections and Slow-Start interact?
  - What happens when a packet is lost during Slow-Start?
  - What happens when the SYN is dropped?
- Bottom line: Which packet gets dropped matters a lot
  - SYN
  - Slow-Start
  - Congestion avoidance
- Do you think most flows are short or long?
- Do you think most traffic is in short flows or long flows?
TCP is designed around the premise of cooperation
- What happens to TCP if it competes with a UDP flow?
- What if we divide $cwnd$ by 3 instead of 2 after a loss?

There are a bunch of magic numbers
- Decrease by 2x, increase by $1/cwnd$, 3 duplicate acks, initial timeout = 3 seconds, etc.

But overall it works really well!
- Still being constantly tweaked…
TCP CC Summary

- TCP Probes the network for bandwidth, assuming that loss signals congestion

- The congestion window is managed with an additive increase/multiplicative decrease policy
  - It took fast retransmit and fast recovery to get there
  - Fast recovery keeps pipe “full” while recovering from a loss

- Slow start is used to avoid lengthy initial delays
  - Ramp up to near target rate, then switch to AIMD
● Flows can be served one bit at a time

● WFQ can be implemented using bit-by-bit weighted round robin
  ◆ During each round from each flow that has data to send, send a number of bits equal to the flow’s weight
Fluid Flow Example

- Orange flow has packets backlogged between time 0 and 10
- Other flows have packets continuously backlogged
- All packets have the same size
Packet (Real) system: packet transmission cannot be preempted. Why?

Solution: serve packets in the order in which they would have finished being transmitted in the fluid flow system.
Packet-Based Example

- Select the first packet that finishes in the fluid flow system
Quality of Service (QoS)

- So far, we have assumed all traffic is equal and provided best effort delivery
  - Perhaps with enforcement to throttle non-responsive senders

- Not always best model. Why?
  - Application demands
    - I want low-delay low-loss for phone service
    - For backup, I just need bandwidth… don’t care about delay
  - Market differentiation
    - I want to sell better service for more money
  - Bandwidth management
    - Don’t let BitTorrent eat up all UCSD bandwidth
Different Demands

- Elastic
- Delay-adaptive
- Hard real-time

Utility vs. Bandwidth
Packet Classification

- Want to treat some traffic better/worse than others
  - How to identify the more important traffic?
  - How much better do we want to treat it?
  - How do we actually treat it better?

- Router **classifies** based on packet header
  - Aggregates
    - From particular network (IP src address)
    - For particular protocol (e.g., port 80 traffic)
  - Individual network flows
    - 5-tuple (src, dst, src port, dst port, protocol)
  - Special header field that indicates traffic “type”
Service Classes

● Best-effort
  ◆ Vanilla IP

● Differentiated service
  ◆ Bronze, Silver, Gold, etc… (effectively priorities, up to some amount of bandwidth per time)
  ◆ E.g., best service up to 10Mbps, then best effort

● Predicted service (soft real-time)
  ◆ Network guarantees good performance on average
  ◆ Application promises only send as fast as negotiated

● Guaranteed service (hard real-time)
  ◆ Network guarantees good performance always
  ◆ Application promises only send as fast as negotiated
More Complicated Routers

Routing Messages

QoS Control messages

Routing

Signaling

Admission Control?

Forwarding Table

Per Flow QoS Table

Data In

Dest Lookup

Classifier

Scheduler

Data Out

Routing

Messages

Control Plane

Data Plane
QoS Summary

- Routers manage their own resources
  - Buffer management may entail marking/dropping
  - Scheduling discipline determines outgoing packet order

- Token bucket and RED
  - Mechanisms to control traffic flowing through routers

- Networks can provide quality of service
  - Combines per-router traffic policing with network signaling
  - IntServ and DiffServ are contrasting approaches
For Next Class

- Read 2.2