Lecture 14 Overview

- TCP congestion control review
- XCP Overview
Challenge: how do we efficiently share network resources among billions of hosts?

- Today: TCP
  - Hosts adjust rate based on packet losses
- Alternative solutions
  - Fair queuing, RED (router support)
  - Vegas, packet pair (add functionality to TCP)
  - Rate control, credits
ACK Pacing in TCP

- ACKs open up slots in the congestion/advertised window
  - Bottleneck link determines rate to send
  - ACK indicates one packet has left the network
Problems with ACK Pacing

● ACK compression
  ◆ Variations in queuing delays on return path changes spacing between ACKs
  ◆ Example: ACK waits for single long packet
  ◆ Worse with bursty cross-traffic

● What happens after a timeout?
  ◆ Potentially, no ACKs to time packet transmissions

● Congestion avoidance
  ◆ Slow start back to last successful rate
  ◆ Back to linear increase/multiplicative increase at this point
Two TCP Connections

- Reach equilibrium independent of initial bandwidth (assuming equal RTTs)
TCP “Friendliness”

- Problem: many different TCP implementations
- If cut back more slowly after drops → grab bigger share
- If add more quickly after ACKs → grab bigger share
- Incentive to cause congestion collapse
  - Many TCP “accelerators”
  - Easy to improve perf at expense of network

- Solutions?
  - Per-flow fair queuing at router
Coupled because a *single* mechanism controls both

**Example:** In TCP, Additive-Increase Multiplicative-Decrease (AIMD) controls both

**XCP** argues **decoupling solves the problem:**

1. **To control congestion:** use **MIMD** which shows fast response
2. **To control fairness:** use **AIMD** which converges to fairness
XCP Advantages

- Tighter Congestion Control
  - Small queues
  - Almost no drops

- Scalable (no per-flow state)

- Flexible fairness definitions
  - Max/min
  - Proportional
  - Differential bandwidth
  - Etc.
XCP Overview

1. Congestion Controller
2. Fairness Controller
## XCP Example

### Congestion Control

<table>
<thead>
<tr>
<th>Feedback</th>
<th>Feedback = + 0.1 packet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round Trip Time</td>
<td>Congestion Window</td>
</tr>
</tbody>
</table>

CSE 222A – Lecture 14: Congestion Control
XCP Example

Round Trip Time
Congestion Window
Feedback = -0.3 packet
XCP Example

Congestion Window = Congestion Window + Feedback

Routers compute feedback without any per-flow state
Congestion Controller

**Goal:** Matches input traffic to link capacity & drains the queue

Looks at aggregate traffic & queue

**Algorithm:**

Aggregate traffic changes by $\Delta$

$\Delta \sim$ Spare Bandwidth

$\Delta \sim$ - Queue Size

So, $\Delta = \alpha d_{avg} \text{Spare} - \beta \text{Queue}$

Fairness Controller

**Goal:** Divides $\Delta$ between flows to converge to fairness

Looks at a flow’s state in Congestion Header

**Algorithm:**

If $\Delta > 0 \Rightarrow$ Divide $\Delta$ equally between flows

If $\Delta < 0 \Rightarrow$ Divide $\Delta$ between flows proportionally to their current rates
Efficiency

CSE 222A – Lecture 14: Congestion Control
Response to Dynamics

XCP Utilization
- Start 40 Flows
- Stop the 40 Flows

TCP-RED-ECN Utilization
- Time (seconds)

XCP Queue (Pkts)
- Time (seconds)

TCP-RED Queue (Pkts)
Short Flows

Average Utilization

Average Queue

Drops

CSE 222A – Lecture 14: Congestion Control
Fairness

Different RTT

Same RTT

Different RTT

CSE 222A – Lecture 14: Congestion Control
Datacenter Review

- Large purpose-built DCs
  - Huge investment: R&D, business

- Transport **inside** the DC
  - TCP rules (99.9% of traffic)

- How’s TCP doing?
TCP in the Data Center

- TCP is challenged to meet demands of apps.
  - Suffers from bursty packet drops, Incast [SIGCOMM ‘09], ...
  - Builds up large queues:
    - Adds significant latency.
    - Wastes precious buffers, esp. bad with shallow-buffered switches.

- Operators work around TCP problems.
  - Ad-hoc, inefficient, often expensive solutions
  - No solid understanding of consequences, tradeoffs
For Next Class…

- Read the DCTCP paper