TCP Congestion Control

15-441: Computer Networks

Titouan Rigoudy
Junchen Jiang
Yuchen Wu

Why?

- Project 2 asks you to build a P2P system
- On top of UDP
- Reliable
- Using congestion control
- Only for DATA transmissions
- Not for control packets
    - WHOHAS, IHAVE, GET, DENIED, etc.

Background

- RFC 793 – Original TCP RFC
- RFC 2001 – Close language to class
- RFC 5681 – More up-to-date RFC 2001
- http://dl.acm.org/citation.cfm?id=52356 – Van Jacobson, Congestion Avoidance and Control
- Linux: `man tcp`

The Learning TCP Problem

- Slide's versions
- Book's version
- RFC versions
- Research paper versions
- Version in your head
- Then, there's the multiple real-world implementations

Learn Exact Versions of TCP

- Tahoe
- Reno
- New Reno
- Vegas
- That's the goal here unfortunately

As always, experimenting on your own with a real implementation is the **only way you will learn anything valuable**.

So, we're making

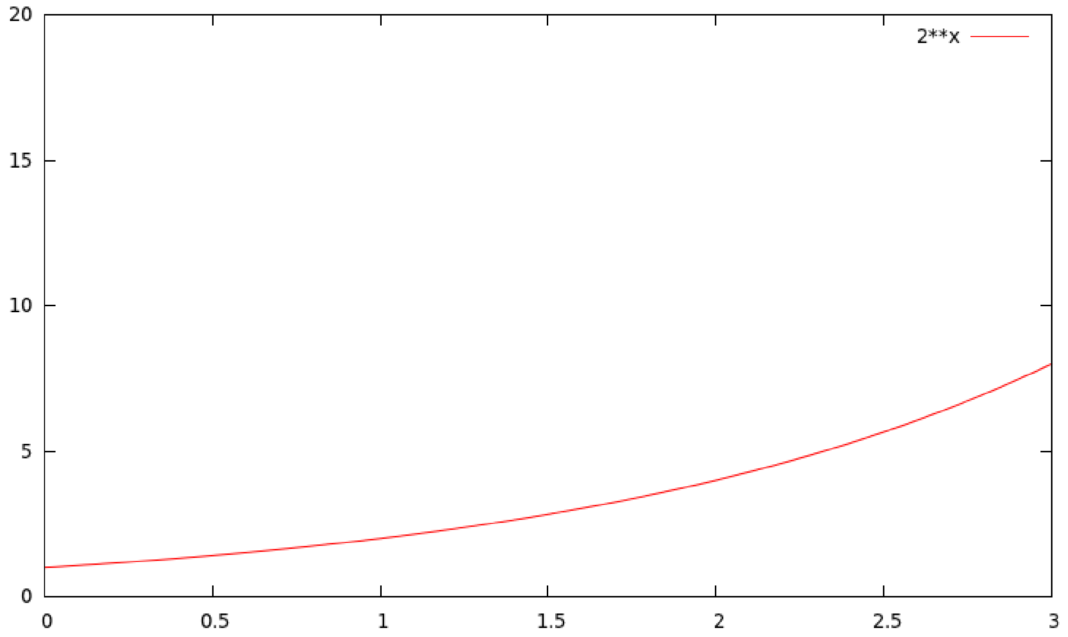
you implement your own.

Problem: Avoid congestion with no central coordination, no knowledge from peers, and no direct network feedback.

All you see are, essentially, ACKs.

New Connection: Slow Start [Tahoe]

- Intuition: Don't flood, but quickly optimize
- Start really small: 1 SMSS
- Grow really fast: exponentially
- Occurs: beginning of TCP, after timeout



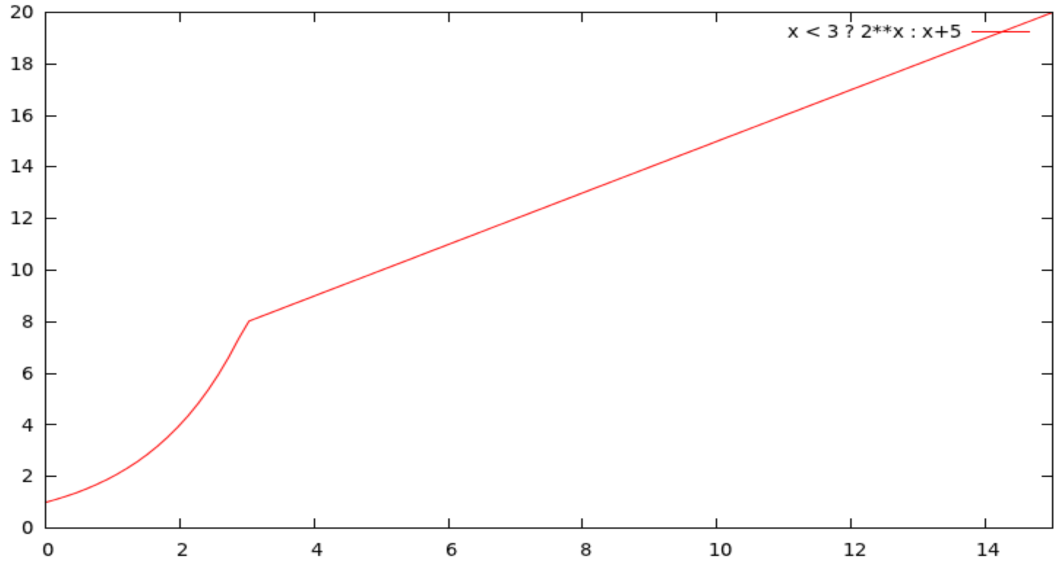
ssthresh

- cwnd – congestion window
  - Governs data transmission (with rwnd)
  - SMSS == sender maximum segment size
  - On segment ACK, `cwnd += SMSS`
- ssthresh – slow start threshold
  - Use slow start when cwnd < ssthresh
  - Use congestion avoidance when cwnd > ssthresh

Typically, ssthresh starts at 65535 bytes.

CA: Additive Increase

- On ACK: `cwnd += SMSS*SMSS/cwnd`
- Takes over when `cwnd > ssthresh`
- `ssthresh = min(cwnd,rwnd) / 2` when congestion
- If congestion is a timeout, `cwnd = SMSS`

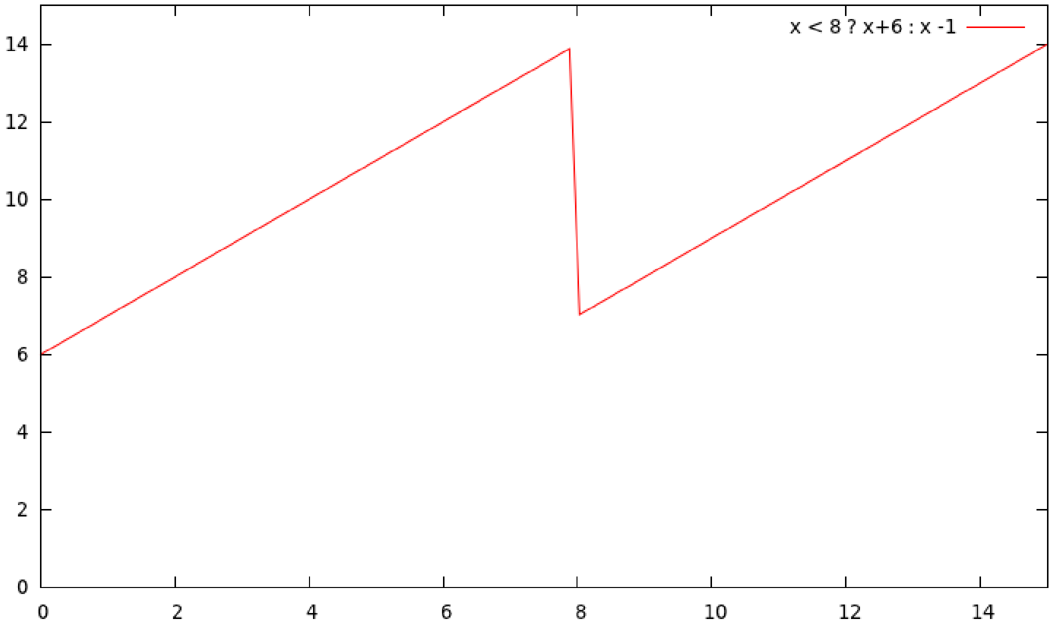


Fast Retransmit [Tahoe]

- Receiver sends duplicate ACKs
- Immediately on out-of-order segment
- Sender receives >= 3 duplicate ACKs
- Immediately retransmit segment
  - `cwnd = SMSS`
  - Slow start
- [Reno] Fast Recovery until non-duplicate ACK

CA: Multiplicative Decrease

- Appears depending on congestion control
  - Most likely [Reno]: 3 Duplicate ACKs
- On a timeout, set `cwnd = cwnd / 2`



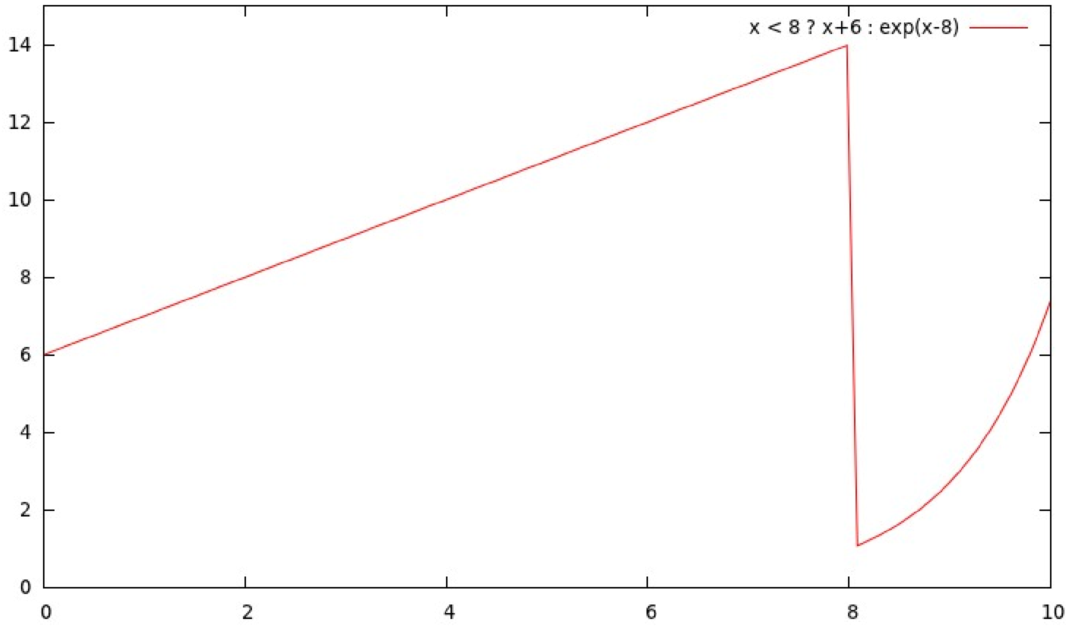
Fast Recovery [Reno, New Reno]

- ssthresh = cwnd / 2
- cwnd = ssthresh [+ 3*SMSS] (in RFC)
- Each time another duplicate ACK arrives,
  - cwnd += SMSS
  - Transmit new segment if allowed [New Reno]
- When ACK for new data arrives
  - cwnd = ssthresh
- If timeout again, slow start with cwnd = SMSS

Timeout Events [Tahoe, Reno]

Both treat these the same: drop to slow start

```
ssthresh = cwnd / 2
    cwnd = SMSS
```



Cheating TCP: Foul Play

- What happens with two TCP streams, one from each host, on a 10 Mbps link?

Cheating TCP: Foul Play

- What happens with two TCP streams, one from each host, on a 10 Mbps link?
- Name them host A and host B. What if host A opens 10 TCP streams? Host B keeps only 1 TCP stream?

Cheating TCP: Foul Play

- What happens with two TCP streams, one from each host, on a 10 Mbps link?
- Name them host A and host B. What if host A opens 10 TCP streams? Host B keeps only 1 TCP stream?
- Fair sharing across streams...
- No notion of logical peers
- That's what download managers do!

Cheating TCP: Foul Play

- What if I implement my own protocol with no congestion control?
- Or designed to grab all the bandwidth?

Cheating TCP: Foul Play

- What if I implement my own protocol with no congestion control?
- Or designed to grab all the bandwidth?

- Internet will be angry
- IETF requires congestion control for RFC

P2P Research: Bandwidth Trading

- Limited dorm links in dorm rooms
- High-speed WiFi between rooms
- What if students all colluded?
- Merging many TCP flows out-of-band :-)
- Fun senior thesis project
- P2P Bandwidth Trading (economics+CS)