There is no Such Thing as TCP: TCP Congestion Control

Wolf Richter
Background

- RFC 793 – Original TCP RFC
- RFC 2001 – Close language to class
- RFC 5681 – More up-to-date RFC 2001
- Vint Cerf is here Friday
- 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: $\text{cwnd } += \frac{\text{SMSS} \times \text{SMSS}}{\text{cwnd}}$
- Takes over when $\text{cwnd } > \text{ssthresh}$
- $\text{ssthresh } = \min(\text{cwnd}, \text{rwnd}) / 2$ when congestion
- If congestion is a timeout, $\text{cwnd } = \text{SMSS}$
CA: Multiplicative Decrease

- Appears depending on congestion control
  - Most likely [Reno]: 3 Duplicate ACKs
- On a timeout, set \( cwnd = \frac{cwnd}{2} \)
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
Fast Recovery [Reno, New Reno]

- $\text{ssthresh} = \frac{\text{cwnd}}{2}$
- $\text{cwnd} = \text{ssthresh} + 3 \times \text{SMSS}$ (in RFC)
- Each time another duplicate ACK arrives,
  - $\text{cwnd} += \text{SMSS}$
  - Transmit new segment if allowed [New Reno]
- When ACK for new data arrives
  - $\text{cwnd} = \text{ssthresh}$
- If timeout again, slow start with $\text{cwnd} = \text{SMSS}$
Timeout Events [Tahoe, Reno]

Both treat these the same: drop to slow start

\[
\text{ssthresh} = \frac{\text{cwnd}}{2}
\]

\[
\text{cwnd} = \text{SMSS}
\]
Experimenting on Your Own

- `getsockopt()` – on a TCP socket
- Transfer large amounts of data
- Check out `TCP_INFO`
- Returns a `struct tcp_info`;
struct tcp_info
{
    u_int8_t tcpi_state;
    u_int8_t tcpi_ca_state;
    u_int8_t tcpi_retransmits;
    u_int8_t tcpi_probes;
    u_int8_t tcpi_backoff;
    u_int8_t tcpi_options;
    u_int8_t tcpi_snd_wscale : 4, tcpi_rcv_wscale : 4;

    u_int32_t tcpi_rto;
    u_int32_t tcpi_ato;
    u_int32_t tcpi_snd_mss;
    u_int32_t tcpi_rcv_mss;

    u_int32_t tcpi_unacked;
    u_int32_t tcpi_sacked;
    u_int32_t tcpi_lost;
    u_int32_t tcpi_retrans;
    u_int32_t tcpi_fackets;

    /* Times. */
    u_int32_t tcpi_last_data_sent;
    u_int32_t tcpi_last_ack_sent; /* Not remembered, sorry. */
    u_int32_t tcpi_last_data_recv;
    u_int32_t tcpi_last_ack_recv;

    /* Metrics. */
    u_int32_t tcpi_pmtu;
    u_int32_t tcpi_rcv_ssthresh;
    u_int32_t tcpi_rtt;
    u_int32_t tcpi_rttvar;
    u_int32_t tcpi_snd_ssthresh;
    u_int32_t tcpi_snd_cwnd;
    u_int32_t tcpi_advmss;
    u_int32_t tcpi_reordering;

    u_int32_t tcpi_rcv_rtt;
    u_int32_t tcpi_rcv_space;

    u_int32_t tcpi_total_retrans;
};
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
P2P Research: Bandwidth Trading

- UVA limited dorm links in dorm rooms
- We had high-speed WiFi between us
- What if we all colluded?
- Merging many TCP flows out-of-band :-)
- Fun senior thesis project
- P2P Bandwidth Trading (economics+CS)
LPTHW EC [10 Points]

Due November 19

Email Wolf
GitHub:

Git it, got it, good.

git clone git://github.com/theonewolf/15-441-Recitation-Sessions.git