Defeating TCP Congestion Control in Three Easy Steps

Stefan Savage
Neal Cardwell, David Wetherall and Tom Anderson

Department of Computer Science and Engineering
University of Washington
Overview

● Simple Question
  - Can a TCP client influence how fast a TCP server sends it data?

● Simple Answer
  - Oh yeah… Big time.
ACK Division

- Send M ACKs for one pkt
- Exponential growth factor proportional to M!
- Preserves end-to-end semantics
**DupACK Spoofing**

- Send extra duplicate ACKs
- Sender sends one pkt for each duplicate ACK
- Preserves end-to-end semantics
Optimistic ACKing

- Send ACKs early
- Sender sends pkts in proportion to ACK rate
- Violates end-to-end semantics
Implementation experience

- TCP Daytona
  - Easy to implement (<75 lines in Linux)
  - Works with all popular sender TCP stacks

- We have the world’s fastest Web browser
What to do?

- Sender-side TCP modifications can mitigate these problems
- It's possible to remove the *incentive* for receiver misbehavior using a variant on *nonces*.
  - Receiver can only hurt itself by lying
- Details can be found in:
  TCP Congestion Control with a Misbehaving Receiver,
  to appear *ACM Computer Communications Review*,
  October, 1999.
RFC 2581
Controlling the congestion window

During slow start, TCP increments cwnd by at most SMSS bytes for each ACK received that acknowledges new data.

... 

During congestion avoidance, cwnd is incremented by 1 full-sized segment per round-trip time (RTT).
Set cwnd to ssthresh plus 3*SMSS. This artificially “inflates” the congestion window by the number of segments (three) that have left the network and which the receiver has buffered.

... For each additional duplicate ACK received, increment cwnd by SMSS. This artificially inflates the congestion window in order to reflect the additional segment that has left the network.