CSCI-1680
Transport Layer II

Data over TCP: Congestion Control I

Nick DeMarinis

Based partly on lecture notes by Rodrigo Fonseca, David Mazières, Phil Levis, John Jannotti
Sign up for TCP milestone meetings by Thursday, April 14, Friday, April 15.

TCP gearup for milestone II
- Likely next Tuesday, 7pm—details to follow.

HW3 (short): out before Thursday, related to what you need for TCP.

Grading is in progress…
Topics for today

• Connection termination
• More on flow control
• Overview of congestion control
• When you have no more data to send, send a FIN
  – Sent by close() or shutdown()
• Both sides close connection separately!
• TIME_WAIT: initiating side should wait for 2*MSL before deleting TCB
  – MSL = Longest time a segment might be delayed (configurable, ~1min)
When to Transmit?

Nagle’s algorithm

• Goal: reduce the overhead of small packets
  
  if (there is data to send) and (window >= MSS)
  
  Send a MSS segment
  
  else
  
  if there is unAcked data in flight
  
  buffer the new data until ACK arrives
  
  else
  
  send all the new data now
  
• Receiver should avoid advertising a window <= MSS after advertising a window of 0
Delayed Acknowledgments

• Goal: Piggy-back ACKs on data
  – Delay ACK for 200ms in case application sends data
  – If more data received, immediately ACK second segment
  – Note: never delay duplicate ACKs (if missing a segment)

• Warning: can interact badly with Nagle for some applications
  – Nagle waits for ACK until send => Temporary deadlock
  – App can disable Nagle with TCP_NODELAY
  – App should also avoid many small writes
The story so far

• Flow control: reliable, in-order delivery
• Goal: send as much data as receiver can handle
  – Receiver’s advertised window: sent with every ACK
• Sliding window: increase throughput by having multiple packets in flight
Flow control provides **correctness**: reliable, in order delivery

Need more for performance

- What if the network is the bottleneck?

Sending too fast will cause queue overflows, heavy packet loss

Need more for performance: congestion control
A Short History of TCP

• 1974: 3-way handshake
• 1978: IP and TCP split
• 1983: January 1\textsuperscript{st}, ARPAnet switches to TCP/IP
• 1984: Nagle predicts congestion collapses
• 1986: Internet begins to suffer congestion collapses
  – LBL to Berkeley drops from 32Kbps to 40bps
• 1987/8: Van Jacobson fixes TCP, publishes seminal paper*: (TCP Tahoe)
• 1990: Fast transmit and fast recovery added (TCP Reno)

* Van Jacobson and Michael Karels. Congestion avoidance and control. SIGCOMM ’88
• Mid 1980’s: Problem with the protocol *implementations*, not the protocol!
• What was happening?
• If close to capacity, and, e.g., a large flow arrives suddenly…
  – RTT estimates become too short
  – Lots of retransmissions → increase in queue size
  – Eventually many drops happen (full queues)
  – Fraction of useful packets (not copies) decreases
The problem

• [https://witestlab.poly.edu/respond/sites/genitutorial/files/tcp-aimd.ogv](https://witestlab.poly.edu/respond/sites/genitutorial/files/tcp-aimd.ogv)
TCP Congestion Control

• 3 Key Challenges
  – Determining the available capacity in the first place
  – Adjusting to changes in the available capacity
  – Sharing capacity between flows

• Idea
  – Each source determines network capacity for itself
  – Rate is determined by window size
  – Uses implicit feedback (drops, delay)
  – ACKs pace transmission (self-clocking)
Congestion control has a long history

• Active research area for ~40 years

• I am nowhere close to being an expert

• My hope is to get you to understand the problems involved
Just a few TCP implementations

What’s the difference?

General usage
• Reno (1980s)
• Tahoe
• Vegas
• New Vegas
• Westwood
• Cubic
• BBR (2016)
• …
Dealing with Congestion

• Maintain two windows:
  – Advertised Window (from receiver)
  – Congestion window (cwnd)

  Sending rate = \( \min(\text{Advertised Window, cwnd}) \)

• Ideally, want to have sending rate: \( \sim \frac{\text{Window}}{\text{RTT}} \)
Dealing with Congestion

- Assume losses are due to congestion
- After a loss, reduce congestion window
  - How much to reduce?
- Idea: conservation of packets at equilibrium
  - Want to keep roughly the same number of packets in network
  - Analogy with water in fixed-size pipe
  - Put new packet into network when one exits
Next time

- TCP Tahoe/Reno
- Overview of other CC schemes