ECE 435 – Network Engineering
Lecture 10

Vince Weaver

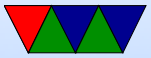
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

28 September 2017

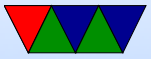
Announcements

- HW#4 was due
- HW#5 will be posted. You'll have 2 weeks due to midterm/fall break
- Midterm Tuesday
- Out of town Tues+Wed for MEMSYS'17. Plan to be back by class on Thursday, will send e-mail if something goes wrong.



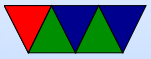
Midterm Review

- Can have one page (8.5" x 11") of notes if you want, otherwise closed everything. I do not think you should need a calculator.
- Mostly short answer questions. No long coding exercises or protocol memorization. Maybe some sockets code, but analyzing it not writing it.
- Know the OSI layers and what each one is for.
- Know at a high level the following protocols:
  - WWW/http



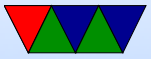
- e-mail
- DNS

- UDP + TCP
  - Know the 3-way handshake
  - Know the tradeoffs between UDP and TCP
  - Why does DNS use UDP
  - Why do webservers use TCP

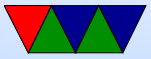


Flow Control

- How much data can be sent before receiving an ACK
- Extreme – just 1 byte. Inefficient (overhead). Also modern systems, a fibre line coast to coast a long time to ACK packet
- Sliding Window Protocol
- Circular buffer (see figure)
- Size of the sliding window: how many outstanding there can be. Once it gets ACKed, can slide window. grow/shrink size of window.



Sent+ACKed		Sent, no ACK yet			Can be Sent Now									Empty	
		10	11	12	13	14	15	16	17	18	19	20	21		
					⇑ Next										

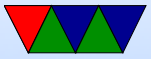


Error Correction

- Ways to Catch Errors
  - Checksum
  - Acknowledgement
  - Time-out
- Example types of Errors
  - Corrupted packet

    SEQ 1401 + 200 bytes, SEQ 1601+200, SEQ 1801+200 sent

    Last one corrupted receiver only acks through



ACK=1601

Eventually timeout, and sender will retransmit

- Lost packet

  Same as previous

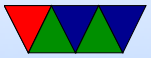
- Duplicate packet (how can happen? a timeout happens and is resent just before ACK gets in)

  TCP discards packets with duplicate SEQ

- Out-of-order packet

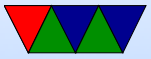
  Do not ACK packet until preceding ones make it.

  For performance can queue up out of order ones so they don't have to be resent



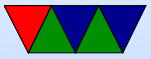
- Lost ACK

  ACKs cumulative, so if the next packet causes an ACK then it doesn't matter. Otherwise a timeout?

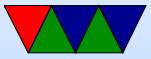


TCP Timer Management

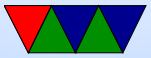
- What should the timer value be? Too short, send extra packets, too long and takes long time to notice lost packets.
- On the fly measures round trip time. (RTT) When send segment, start timer, updates. Various algorithms. Often 2 or 4x
- Connection Timer – send SYN. If no response in time, reset



- Retransmission Timer – retransmit data if no ACK
- Delayed ACK timer – if send a packet, tag an ACK along if timer expires and no outgoing data, have to send stanadlone ACK
- Persist Timer – solve deadlock where window was 0, so waiting, and missed the update that said window was open again.
  Sends special probe packet. Keep trying every 60s?
- Keepalive Timer – if connection idle for a long time, sends probe to make sure still up

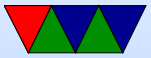


- FIN_WAIT_2 Timer – avoid waiting in this state forever if other side crashes
- TIME_WAIT_TIMER – used in TIME_WAIT to give other side time to finish before CLOSE



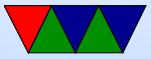
TCP Congestion Control

- Fast network feeding small receiver (flow control?)
- Slow network feeding big receiver (problem on sending side, lose packets, congestion control)
- Two windows, receiver window (RWND) looked at previously, and congestion window (CWND) (kept locally)
- Amount can send is the minimum of the two windows
- Setup
  - Congestion window set to max segment size



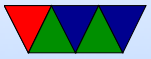
- Send one max segment, arm timer
- If ACK received before timer goes off, good. Double the size.
- Repeat, exponential growth. Called "slow start"
- "internet (?)" has a limit where it stops exponential and moves to linear growth
- eventually hit receiver window size and stop

- Changed over the years.
- Initial implementation no congestion control, not needed (not that much traffic)
- After 8 years (1980s) introduced by Van Jacobson (Van



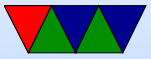
is his first name) – internet facing "congestion collapse" – would send as fast as possible, packets would be dropped, hosts retransmit, even more congestion

- Usually assume corrupted packets are rate, at least on wired. Wireless. More lost packets, so shouldn't slow down but maybe try harder in congestion



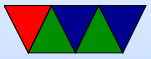
TCP Tahoe

- TCP Tahoe (v2) (BSD 4.2 1988) added congestion avoidance, fast retransmit (Van Jacobsen)
- Slow start – probing bandwidth with few rounds.
- CWND set to 1 and exponentially increases with each ACK until hits ssthresh
- congestion avoidance – slow probing but rapid respond to congestion
- AIMD additive increase multiple decrease.
- Fast retransmit– transmitting lost packets immediately,



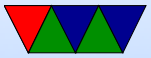
no wait for timer. If get three duplicate ACKS in a row, assume packet loss, resend. Drop ssthresh to half and start slow-start again

- retransmission timeout – halve sshthreash and restart slow-start
- When to retransmit
  - If packet lost, then will receive duplicate ACK on next transmission
  - If get three identical ACKs, probably means packet lost, resend
  - Why not two? Because if packets arrive out of order



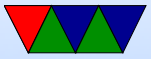
can also cause duplicate ACK

- TCP Reno (v3) added fast recovery
  set sshthread to cwnd+3 because of triple ack
- TCP New Reno



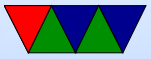
TCP Options

- One-byte
  - End of option – end of all options. Only one allowed (not always needed?)
  - No operation – for padding
- Multi-byte
  - MSS maximum segment size (only in initial SYN packet)
    Byte1: 2, Byte 2: len=4, max size=2 bytes
  - RFC1323 –PAWS, window scaling factor, specify



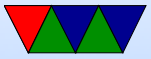
larger transfer size as on long-latency high-bandwidth connections can end up idle a lot waiting for ACK
Scales the window size by value
Byte 1: 3, Byte 2: Len=3, Scale factor=1 byte

- Timestamp: used to calculate round-trip time. Leaks info? Send along timestamp when you send, other side keeps that and returns it (as echo) with the relevant ACK.
  10 bytes long Byte 1: 8, Byte 2: Len=10, Timestamp=4 bytes, Echo=4 bytes
- RFC1106 allows selective resend – if lost packet in



long stream, instead of sending all, just resend missing

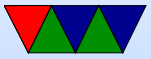
- Fast connections sequence can wrap quickly.



Security Issues

- SYN Flood attack – Denial of service – spoof IP address, send lots of spurious SYNs followed by ACK, tie up lots of resources
  Spoof, because responds to wrong address which just ignores. Causes lots of half-open connections
  One solution is SYN cookies – (pick special sequence that allow throwing out connection info but able to reconstruct if an ACK comes back).

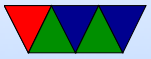
- Connection hijacking – guess a proper sequence number



and forge a packet that looks like it should be next. If you can also take down the real IP (DOS?) can take over the connection Helps to have good random sequence numbers

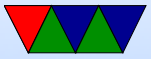
- TCP veto – inject packet with sequence and payload of next expected. That way when the real actual next one comes in, it will be silently dropped as a duplicate

- NMAP port scanning – send packets and find if connections are open, determine host system. Christmas Tree packets



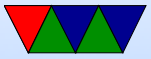
Find out host? options supported, sequence numbers. Also can find out uptime of system as timestamps from extension usually aren't reset each time.

- Martian packets – packets with a source of a reserved network found on routed internet
- UDP broadcast storm/amplification attack – broadcast bad packet to 10000 machines, all reply at once with error



Making things faster

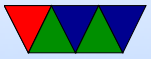
- Offload engines



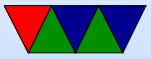
Proposed Replacements

- T/TCP – Transactional TCP.
  To send one small message (w/o UDP) can require up to 9 packets. (handshake, data transfer, shutdown)
  Instead, in the initial packet put SYN, DATA, AND FIN.
  The small message done in 3 packets.

- SCTP – stream control transmission protocol. More complex
  four-way handshake (why? prevent SYN flood attacks)



- QUIC – from google
  Over UDP for now (NATs won't route protocols they don't know)
  Used by Youtube, Google, etc. with Chromium
  Head of Line problem with TCP
  Single-way handshake if version match, otherwise has to negotiate.
  Encrypted
  Once encrypted connection set up once, assumed still there and so sends single HELLO packet followed by data.



Sends redundancy in packets which can be used with XOR to reconstruct missing packets. (but shown not to help much?)