ECE 435 – Network Engineering
Lecture 11

Vince Weaver

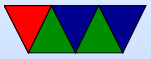
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

21 February 2023

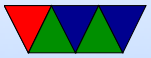
Announcements

- HW#5 was posted
- Note that you can see what states sockets in with `netstat` (obsolete) or `ss`

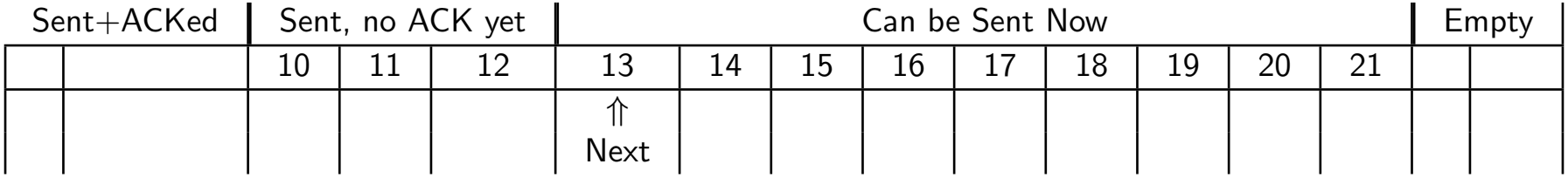


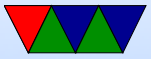
Flow Control

- 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
- How does operating system / TCP stack keep track of this
  - Sliding Window Protocol
  - Circular buffer (see figure)
  - Size of the sliding window: how many outstanding



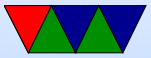
there can be. Once it gets ACKed, can slide window. grow/shrink size of window.





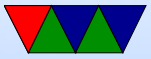
Receiver Window (RWND)

- Receiver "advertises" a window, how much incoming data it wants.
- Example:
  - Receiver has 4k buffer
  - Sender does 2k write (2k/SEQ=0)
  - Receiver sends back ACK=2k, WIN=2048 (can take up to 2k)
  - Application sends 2k (2k, SEQ=2k)
  - If it is full, receiver might send ACK=4k, WIN=0



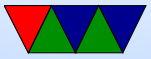
- Later once buffer clears up a bit (application reads 2k maybe) sends ACK=4096, WIN=2k
- Sender then sends some more

- When happens when waiting on a WIN=0?
  - URG can come to kill the connection
  - Sender can send a "window probe", a 1-byte packet with retransmit window and next byte expected (in case the ACK restarting was lost, otherwise deadlock)

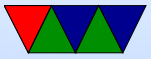


Window Management / Flow Control

- Senders do not have to transmit incoming data immediately
- Receivers do not have to ACK immediately
- Can have a lot of overhead to send 40byte packet for one byte payload
- Senders can buffer data, for example if know window is 4k can wait until they have 4k. Can help performance.
- For example, typing at keyboard on telnet/ssh might send when an editor. Press key, send a packet. Get

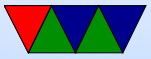


ACK. Then when read, another ACK updating window size. Then finally draw char on screen, send packet with that. 4 packets for one keypress (total of 160x overhead)

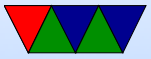


Sender Window Problems

- What if sender only sending 1-byte at time?
- Can do "delayed acknowledgement" where you buffer up to 500ms for additional input. Adds lot of latency.
- Nagel's Algorithm
  - (John Nagel, 1984) RFC896
  - When only sending one byte at a time, send one packet, but buffer the rest until the outstanding data is ACKed
  - Also take into account window size.

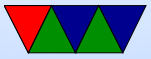


- Widely used, can be bad for things like X window forwarding as mouse movements bunched together.
- Interacts poorly with Delayed ACKs
- Often causes despair to people unaware and having to debug latency problems
- TCP_NODELAY option disables.



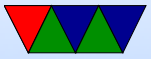
Receiver Window Problems

- Silly Window Syndrome
- Slow reader on receive side. Application reads one byte out, stack immediately advertises 1-byte window and causes back-and-forth
- Clark's solution
  - If receiving small amounts, close window until buffer half empty and the open again.



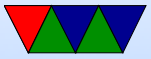
TCP Congestion Control

- Fast network feeding small receiver (flow control?)
- Slow network feeding big receiver (problem on sending side, lose packets, congestion control)



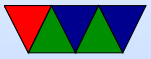
TCP Congestion Control – History

- Originally no congestion control, not needed (not that much traffic)
- 1980s Internet facing "congestion collapse" – would send as fast as possible, packets would be dropped, hosts retransmit, even more congestion



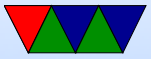
TCP Congestion Control

- Solution: detect dropped packets as congestion, throttle down if it happens
- Additive Increase/Multiplicative Decrease (AIMD) linearly increase bandwidth attempts, but if start losing packets back off exponentially
- Add CWND (congestion window) to control data sent in one round-trip-time with a MWND maximum-window size.
- Use the smallest of CWND/MWND



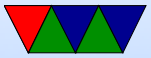
TCP Congestion Control – Slow Start

- Quickly probe bandwidth with a few rounds.
- CWND set to 1, 2, 4, or usually 10 packets
  packet size is Maximum Segment Size (MSS)
  MSS usually 1460 bytes on ethernet
- CWND increased exponentially by doubling each time gets ACK. until hits ssthresh (Slow-start threshold)
- Once ssthresh reached, enter congestion avoidance



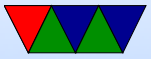
TCP Congestion Control – Congestion Avoidance

- Enter once slow-start hits ssthresh
- Only increase linearly (AIMD)
- Each ACK received, increase CWND by $MSS \times \frac{MSS}{CWND}$
- If get three duplicate ACKS in a row, assume packet loss congestion
  - Drop ssthresh to half
  - Start slow-start again



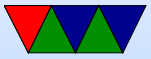
TCP Congestion Control – Fast Retransmit

- re-transmit lost packets immediately, no wait for timer.



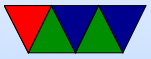
Why Three ACKs?

- 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



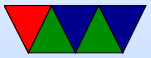
TCP Tahoe

- TCP Tahoe (v2) (BSD 4.2 1988)
- added congestion avoidance, fast retransmit
- (Van Jacobsen) [Van is his first name]
- Can have bad performance on workloads like http where each connection for a file has to go through slow-start
- Also can perform poorly on networks where packets being lost for other reasons



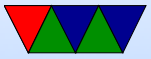
TCP Reno

- Fast recovery if three ACKs
  - Fast retransmit sent
  - Half of CWND saved as ssthresh and new CWND
  - Skip slow start and go directly to congestion avoidance
- Sawtooth Pattern
- Commonly used for 20 years



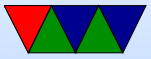
TCP New Reno

- RFC 3782
- RFC 6582

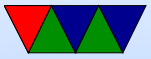


Explicit Congestion Notification (ECN)

- Instead of dropping packet during congestion, router sets ECN bits to let receiver know
- CE (congestion encountered) bit set by router if there's congestion and both sides have marked that they have ECN enabled
- ECE bit echos back to sender the CE bit
- CWR – congestion window received
- Ability to talk ECN negotiated during the handshake at beginning

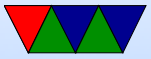


- Since these bits were originally marked as reserved, some routers didn't support them and if you enabled ECN on Linux your packets might silently get dropped. In theory things are better now.



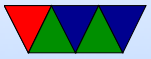
TCP BIC

- Binary Increase Congestion Control
- Optimized for high-speed networks
- Linux used from 2.6.8 to 2.6.18



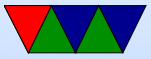
TCP CUBIC

- Update congestion window bases on last duplicate ACK, rather than on ACKs
- Congestion window with Cubic function
- Used by Linux post 2.6.19 and older Windows



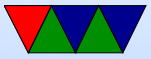
Compound TCP (Microsoft)

- Two congestion windows?
- Used since Windows Vista



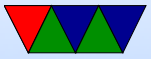
TCP BBR

- TCP BBR (Bottleneck BW and Roun-trip) Google
- Model-based
- Concerns about fairness, if it can result in getting more than fair share of bandwidth



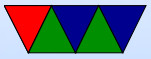
TCP other, So Many, this isn't Exhaustive

- TCP Hybla
- Agile-SD TCP
- TCP Westwood
- TCP Proportional Rate Reduction
- Others (many many)



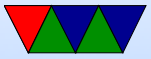
Real World Impacts

- `https://endtimes.dev/why-your-website-should-be-under-14kb-in-size/`
- Article says if you keep your website under 14k (under 10 MSS units) it will load faster as the first 10 packets are sent under slow start but then it waits for an ACK before requesting more



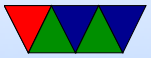
Wireless Issues

- TCP detects congestion from packet loss
- Wireless networks can have a lot of packet loss, so connections might be throttled unnecessarily



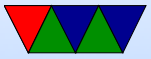
Buffer Bloat

- Routers have more RAM these days and can buffer packets temporarily adding delay
- Especially when congested the buffers can fill up and packets can get stuck in queue for long time
- This can keep TCP congestion control from working well



Security Issues

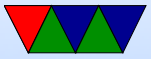
- SYN Flood attack – Denial of service
  Send SYN (but no ACK) or spoof IP address, send lots of spurious SYNs followed by ACK, waits for final ACK tie up lots of resources
  Spoof, because responds to wrong address which just ignores. Causes lots of half-open connections
  One solution is SYN cookies – (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

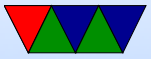
- NMAP port scanning – send packets and find if connections are open, determine host system. Christmas



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.

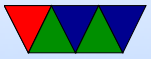
- Martian packets – packets with a source of a reserved network found on routed internet
- UDP broadcast storm/amplification attack – broadcast bad packet to 10000 machines, all reply at once with error
- Smurf attack (same as above?) – spoof return address



and send broadcast packet. All ICMP error replied overwhelm spoofed address (this doesn't work as well anymore)

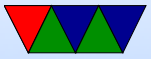
- Fraggle attack –UDP attack targeting the chargen and echo ports

- winnuke – set urgent pointer out of bounds, window didn't handle properly (as unusual)



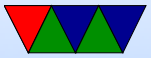
Making things faster

- Offload engines



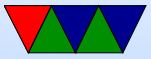
Proposed Replacements

- TCP does have issues, various new protocols have been proposed
- Often held back by routers that drop packets with protocols or parameters that they don't recognize

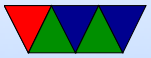


Issues with TCP

- Any error can cause a stall
- If a single connection has multiple internal streams (HTTP/2) all streams get stopped by a single error, even if the packet isn't important for the others
  "Head of Line Blocking"
- Many packet overhead to start/end connection
- TCP is a raw data stream ignorant of stuff inside
  If you need to negotiate further protocol stuff (like TLS) need to have a handshake after the initial TCP handshake

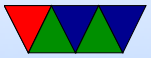


so lots of wasted packets



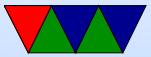
T/TCP (never caught on)

- 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.



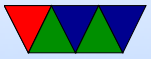
TCP Fast Open

- Store cookie from previous connection
- Allow quick re-open/re-start without waiting for ACK

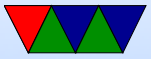


SCTP – Stream Control Transmission Protocol

- More complex
  four-way handshake (why? prevent SYN flood attacks)
- Each packet multiple control chunks and multiple data chunks
- Preserve boundaries (100 bytes then 50 bytes seen on other end that way)
- Allows un-ordered packet delivery
- Multi-homing/Multi-streaming

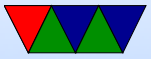


- Designed for telecommunication use?



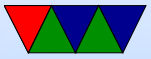
MPTCP – Multipath TCP

- Balance connection across multiple interfaces
- Useful for things like multiple ethernet ports, or simultaneous mobile/wifi



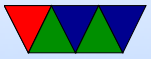
QUIC

- (originally Quick UDP Internet Connections)
- RFC9000 (May 2021)
- Designed to lower latency of HTTP connections
- Over UDP for now (NATs won't route protocols they don't know)
- Used by Youtube, Google, etc. with Chromium, Firefox
- Head of Line problem with TCP
  One missing packet blocks processing all previous
- Single-way handshake if version match, otherwise has to



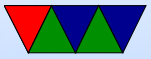
negotiate.

- Encrypted
  encrypted by packet rather than full stream so can handle missing packets better
- 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?)
- Help in roaming, TCP have to rely on timeouts to notice



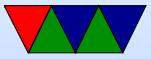
this. QUIC has identifier for connection that can restart

- Implemented in userspace? Not socket interface
- Chrome would open both QUIC and TCP and drop back to TCP if no QUIC

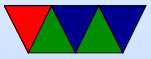


Sample Packets

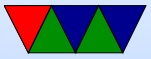
```
sudo tcpdump -XX -i eth0 tcp port 31337
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
15:00:07.507769 IP 192.168.8.38.57962 > 192.168.8.138.31337: Flags [S],
seq 2755900677, win 29200,
options [mss 1460,sackOK,TS val 2941869143 ecr 0,nop,wscale 7], length 0
0x0000:  b827 ebf8 7742 0050 b647 1cde 0800 4510  .'..wB.P.G....E.
0x0010:  003c c79f 4000 4006 e10b c0a8 0826 c0a8  .<..@.@......&..
0x0020:  088a e26a 7a69 a443 b505 0000 0000 a002  ...jzi.C........
0x0030:  7210 8220 0000 0204 05b4 0402 080a af59  r..............Y
0x0040:  5c57 0000 0000 0103 0307                 \W........
15:00:07.507901 IP 192.168.8.138.31337 > 192.168.8.38.57962: Flags [S.],
seq 3080782625, ack 2755900678, win 28960,
options [mss 1460,sackOK,TS val 3332076052 ecr 2941869143,nop,wscale 7],
length 0
0x0000:  0050 b647 1cde b827 ebf8 7742 0800 4500  .P.G...'..wB..E.
0x0010:  003c 0000 4000 4006 a8bb c0a8 088a c0a8  .<..@.@.........
0x0020:  0826 7a69 e26a b7a1 0321 a443 b506 a012  .&zi.j...!.C....
0x0030:  7120 922f 0000 0204 05b4 0402 080a c69b  q../............
0x0040:  7214 af59 5c57 0103 0307                 r..Y\W....
15:00:07.508278 IP 192.168.8.38.57962 > 192.168.8.138.31337: Flags [.],
ack 1, win 229, options [nop,nop,TS val 2941869143 ecr 3332076052], length 0
```



```
0x0000:  b827 ebf8 7742 0050 b647 1cde 0800 4510  .'..wB.P.G....E.
0x0010:  0034 c7a0 4000 4006 e112 c0a8 0826 c0a8  .4..@.@......&..
0x0020:  088a e26a 7a69 a443 b506 b7a1 0322 8010  ...jzi.C.....".."
0x0030:  00e5 2e94 0000 0101 080a af59 5c57 c69b  ...........Y\W..
0x0040:  7214                                     r.
15:00:09.037573 IP 192.168.8.38.57962 > 192.168.8.138.31337: Flags [P.],
seq 1:6, ack 1, win 229,
options [nop,nop,TS val 2941870672 ecr 3332076052],
 length 5
0x0000:  b827 ebf8 7742 0050 b647 1cde 0800 4510  .'..wB.P.G....E.
0x0010:  0039 c7a1 4000 4006 e10c c0a8 0826 c0a8  .9..@.@......&..
0x0020:  088a e26a 7a69 a443 b506 b7a1 0322 8018  ...jzi.C.....".."
0x0030:  00e5 3d1b 0000 0101 080a af59 6250 c69b  ..=........YbP..
0x0040:  7214 6865 790d 0a                        r.hey..
15:00:09.037634 IP 192.168.8.138.31337 > 192.168.8.38.57962: Flags [.],
ack 6, win 227, options [nop,nop,TS val 3332077582 ecr 2941870672], length 0
0x0000:  0050 b647 1cde b827 ebf8 7742 0800 4500  .P.G...'..wB..E.
0x0010:  0034 3e1c 4000 4006 6aa7 c0a8 088a c0a8  .4>.@.@.j.......
0x0020:  0826 7a69 e26a b7a1 0322 a443 b50b 8010  .&zi.j..."..C....
0x0030:  00e3 9227 0000 0101 080a c69b 780e af59  ...'........x..Y
0x0040:  6250                                     bP
15:00:09.039363 IP 192.168.8.138.31337 > 192.168.8.38.57962: Flags [P.],
 seq 1:6, ack 6, win 227, options [nop,nop,TS val 3332077584 ecr 2941870672],
 length 5
```



```
0x0000:  0050 b647 1cde b827 ebf8 7742 0800 4500  .P.G...'..wB..E.
0x0010:  0039 3e1d 4000 4006 6aa1 c0a8 088a c0a8  .9>.@.@.j.......
0x0020:  0826 7a69 e26a b7a1 0322 a443 b50b 8018  .&zi.j..."".C....
0x0030:  00e3 922c 0000 0101 080a c69b 7810 af59  ...,........x..Y
0x0040:  6250 4845 590d 0a                        bPHEY..
15:00:09.039650 IP 192.168.8.38.57962 > 192.168.8.138.31337: Flags [.],
ack 6, win 229, options [nop,nop,TS val 2941870674 ecr 3332077584], length 0
0x0000:  b827 ebf8 7742 0050 b647 1cde 0800 4510  .'..wB.P.G....E.
0x0010:  0034 c7a2 4000 4006 e110 c0a8 0826 c0a8  .4..@.@......&..
0x0020:  088a e26a 7a69 a443 b50b b7a1 0327 8010  ...jzi.C.....'..
0x0030:  00e5 2293 0000 0101 080a af59 6252 c69b  ..".........YbR..
0x0040:  7810                                     x.
15:00:12.469960 IP 192.168.8.38.57962 > 192.168.8.138.31337: Flags [F.],
 seq 6, ack 6, win 229, options [nop,nop,TS val 2941874105 ecr 3332077584],
 length 0
0x0000:  b827 ebf8 7742 0050 b647 1cde 0800 4510  .'..wB.P.G....E.
0x0010:  0034 c7a3 4000 4006 e10f c0a8 0826 c0a8  .4..@.@......&..
0x0020:  088a e26a 7a69 a443 b50b b7a1 0327 8011  ...jzi.C.....'..
0x0030:  00e5 152b 0000 0101 080a af59 6fb9 c69b  ...+.......Yo...
0x0040:  7810                                     x.
15:00:12.471639 IP 192.168.8.138.31337 > 192.168.8.38.57962: Flags [F.],
 seq 6, ack 7, win 227, options [nop,nop,TS val 3332081016 ecr 2941874105],
 length 0
0x0000:  0050 b647 1cde b827 ebf8 7742 0800 4500  .P.G...'..wB..E.
```



```
0x0010:  0034 3e1e 4000 4006 6aa5 c0a8 088a c0a8  .4>.@.@.j.......
0x0020:  0826 7a69 e26a b7a1 0327 a443 b50c 8011  .&zi.j...'.C....
0x0030:  00e3 9227 0000 0101 080a c69b 8578 af59  ...'.........x.Y
0x0040:  6fb9                                     o.
15:00:12.471921 IP 192.168.8.38.57962 > 192.168.8.138.31337: Flags [.],
 ack 7, win 229, options [nop,nop,TS val 2941874107 ecr 3332081016], length 0
0x0000:  b827 ebf8 7742 0050 b647 1cde 0800 4510  .'..wB.P.G....E.
0x0010:  0034 c7a4 4000 4006 e10e c0a8 0826 c0a8  .4..@.@......&..
0x0020:  088a e26a 7a69 a443 b50c b7a1 0328 8010  ...jzi.C.....(..
0x0030:  00e5 07c0 0000 0101 080a af59 6fbb c69b  ...........Yo...
0x0040:  8578                                     .x
```