ECE 435 – Network Engineering Lecture 13

Vince Weaver

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

vincent.weaver@maine.edu

21 February 2025

Announcements

- HW#4 due today, don't forget
- HW#5 will be posted

HW#5 Notes – Hexdumps

- Decoding a hexdump

```
hexdump -C ece435_lec08.pdf
00000000  25 50 44 46 2d 31 2e 35  0a 25 d0 d4 c5 d8 0a 39  |%PDF-1.5.%.....9|
00000010  20 30 20 6f 62 6a 0a 3c  3c 0a 2f 4c 65 6e 67 74  | 0 obj.<<./Lengt|
00000020  68 20 33 37 33 20 20 20  20 20 20 20 0a 2f 46 69  |h 373       ./Fi|
00000030  6c 74 65 72 20 2f 46 6c  61 74 65 44 65 63 6f 64  |lter /FlateDecod|
00000040  65 0a 3e 3e 0a 73 74 72  65 61 6d 0a 78 da 9d 52  |e.>>.stream.x..R|
```

- First column is offset into the file or packet (usually in hex).
- The next set of columns are the raw bytes, in hex.
- The last column is the ASCII char equivalent of the raw data. a '.' often indicates non-printable ASCII.

HW#3 Review

- md5sum/encryption, seems to have gone well
- How to validate PGP key is indeed for who it says?
  - https isn't enough, what if the person who admins the webserver is evil?
  - Certificate Authority (costs money)
  - Distributed Web of Trust (key signing party).
  - Compare in person/phone, key fingerprint if not want to send whole thing
  - Can you trust phone/video calls anymore?

- Encrypted message went fine
- Why not use SHA-1 for git anymore? It's been "broken" which means possible to generate a collision
- umaine website certificate
  - aside: it stores 107k of private data on you?
  - Internet2 certificate, Public key 4096bit RSA, valid for a year, possibly SHA-256 ECDSA
  - That was signed by Incommon RSA, certificate valid for 10 years
  - That was then signed by Usertrust, good until Jan 2038 (!?)

- md5sum extra credit, this is first year people actually made proper collisions. Though birthday and/or chosen-prefix of course

Transmission Control Protocol (TCP)

- RFC 793 (from 1981) / 1122 / 1323

  2018 / 2581 / 2873 / 2988 / 3105, summary in 4614
- Generally attributed to Vint Cerf and Bob Kahn
- Reliable, in-order delivery.
- Adapts to network congestion
- Takes data stream, breaks into pieces smaller than 64k (usually 1460 to fit in Ethernet) and sends as IP

- No guarantees all packets will get there, so need to retransmit if needed.
- Multiple connections can share same port (i.e. webserver on port 80 can handle multiple simultaneous requests)
- Point-to-point (can't multicast)
- Full duplex
- Byte stream, if program does 4 1024byte writes there's no guarantee the other end sees 4 chunks of 1024, only 4k stream of bytes is guaranteed.

TCP Header

Fixed 20-byte header. From RFC793:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |       |C|E|U|A|P|R|S|F|                               |
| Offset|Reservd|W|C|R|C|S|S|Y|I|            Window             |
|       |       |R|E|G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TCP Header Format

- 16-bit source port
- 16-bit dest port
- 32-bit sequence number
- 32-bit ack number

  next byte expected, not last one received

TCP Header – Offset / Flags

- 4-bit data offset (header length) points to start of data. NOTE: must multiply by 4. (minimum is 5 (20 bytes), max 15 (60 bytes))
- 3-bit reserved zero (not used)
- NS / CWR / ECE – for ECN congestion
- ACK (acknowledge) – 1 if ack field valid, otherwise ack field ignored

TCP Header – Flags (Continued)

- U (URGent) – urgent pointer points to urgent byte
  - URGENT flag can be sent that says to transmit everything and send a signal on the other side that things are urgent.
  - 16-bit urgent pointer
- PSH – receiver should process the data immediately and not buffer it waiting for more to come in
  - PUSH flag can be sent that says not to buffer (For example, if interactive command line)

TCP Header – Flags (Continued)

- RST (reset) – reset a connection because something has gone wrong
- SYN (synchronize) – used to establish connection CONNECTION REQUEST (SYN=1,ACK=0) and CONNECTION ACCEPTED (SYN=1,ACK=1)
- FIN – used to release a connection

TCP Header continued – Window

- 16-bit window size
- We'll discuss this more later
- Only in ACK, says how many bytes to send back.
- This can be 0, which means I received everything but I am busy and can't take any more right now (can send another ACK with same number and nonzero window to restart)

TCP Header continued – Checksum

- 16-bit ones' complement checksum
- Same calculation as UDP
- As with UDP also add in pseudo header

TCP Header – Options

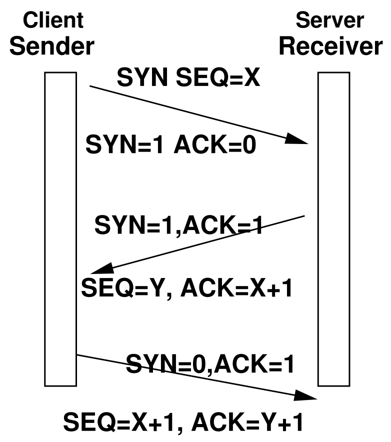
- options (32-bit words) – we'll discuss these later
- **type=0** End of option
  End of all options. Only one allowed (not always needed?)
- **type=1** No operation (for padding to 32-bit boundary)
- **type=2, Len=4, Value=16-bits** Maximum Segment Size
  only in initial SYN packet
- **type=3, Len=3, Value=8-bits** Window size

Scaling factor to shift window size by (0..14), raising limit to 1GB. Only set during handshake

- **type=4, len=2** Selective ACK permitted
- **type=5, len=?** Selective ACK
  list of 1-4 blocks being selectively acknowledged, as 32-bit begin/end pointers
  allows only resending missing packets instead of having to restart at last ACK (RFC1106?)
- **type=8, len=10** Timestamp and echo of last timestamp
  Not necessarily current time. (RFC1323) PAWS, Protection against Wrapped Sequence-number

High bandwidth, seq num can wrap. Use timestamps to recognize when this happens.
Fast connections sequence can wrap quickly (orig internet 56k, modern 1Gb connection wrap in seconds rather than weeks)

TCP Opening Connection



- Three-way handshake (Tomlinson 1975)
  - Server does LISTEN/ACCEPT to wait for connection.
  - Client issues CONNECT: destination/port/size, etc.
  - CONNECT chooses random initial sequence number (ISN) X

Sends SYN(SEQ=X) (SYN=1 ACK=0) with port and sequence number

- Server receives packet. Checks if listening on that port; if not send back a packet with RST to reject.
- Otherwise it can accept

  sends back ACK(X+1) plus SYN(SEQ=Y) with random sequence# of own
- Client then responds with the server SYN ACK(Y+1) SEQ=x+1
- Connection is established

Good Sequence Numbers

- SEQ number picked, not to be 0
- Originally clock based (random these days).
- If machine reboots should wait for maximum lifetime to make sure all close
- Why do this? What happens with simultaneous connection? What if attacker can easily guess your sequence number?

TCP Closing Connection

- TODO: diagram
- Closing connection
- Although full duplex, almost like two independent one-way connections, released independently
  - one side sends packet with FIN
  - other side sends ACK of FIN, that direction is shut down
  - other direction can keep sending data though
  - at some point other side sends FIN

◦ this is ACKed

Can you Guarantee Progress on Closing?

- What if one of packets lost?
- Two army problem?
  - Two generals on opposite side trying to co-ordinate attack.
  - Any message can be intercepted by enemy. So say "attack at 9pm" but that could be lost. Could require other side to send reply, but that could be lost.
  - You need infinite messages to guarantee it got through.
- Use timeout

- If FIN not ACKed within two packet lifetimes, will close anyway.
- The other side eventually notices and closes too.