ECE 435 – Network Engineering
Lecture 20

Vince Weaver

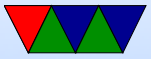
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

10 March 2025

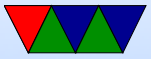
Announcements

- HW#7 was posted
- HW#5 and HW#6 will be graded soon
- Don't forget Midterm is this Wednesday (the 12th)



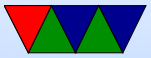
Midterm Preview

- 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.
- There might be some sockets code, but analyzing it not writing it.

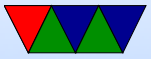


Midterm Preview – Topics

- Know the OSI layers and what each one is for.
- Be aware of socket programming in C, and what the common syscalls do (bind(), listen(), accept(), read(), write(), etc.)
- Know at a high level the following protocols:
  - WWW/http
  - e-mail
  - DNS
- Encryption (at a high level)

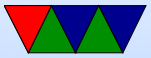


- UDP + TCP
  - Know the 3-way handshake
  - Know the tradeoffs between UDP and TCP
- No detailed general network layer stuff (i.e. no Dijkstra algorithm question)
- Brief IPv4 questions, not a lot as we haven't finished the homework for that yet



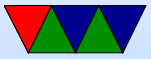
Brief HW#5 Review

- source/destination/size/checksum
  - src: a9a0 = 43424 (note, hex dumps are naturally big endian)
  - dest: 35 = 53 (DNS)
  - size: 2a = 42 bytes
  - yes checksum (note: 0000 means no checksum. ffff is a valid one)
  - protocol is DNS (how can you tell?)
- Why use UDP vs TCP



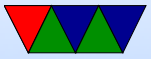
lower latency, lower overhead (no need to handshake), simpler

Be careful just saying “faster”, need to explain more what you mean by that.



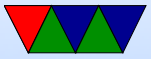
HW#5 Coding Notes

- Remember to comment your code!
- Getting source port from incoming connection
- Note this is not the IP address
- Getting it from the struct is sort of hard
- Also remember it's in network endian, need to convert with `ntohs()` In general would be an ephemeral port above 40000

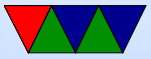


HW#6 Review – TCP Header

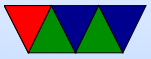
```
0x0022:  bda5 ____________________ Source port (48549)
0x0024:  0050 ____________________ Destination port (80)
0x0026:  cdc4 6a49 _______________ Sequence Number
0x002a:  3c7b 6ca5 _______________ Acknowledgement Number
0x002e:  80 ______________________ 1000 header length = 8*4=32
0x002f:  18 ______________________ 11000 ACK+PSH
0x0030:  00e5 ____________________ Window Size = 229 (likely *128)
0x0032:  79f4 ____________________ Checksum = 0x79f4
0x0034:  0000 ____________________ Urgent = ?
0x0036:  01       _Option: NOP (padding)
0x0037:  01        _Option: NOP (padding)
0x0038:  080a      _Option: Timestamp, 10 bytes
0x003a:  0104 3e58 _Timstamp TSval
0x003e:  34a8 7bc3 _Timestamp TSecr Echo Reply
```



- Header offset/length was the most trouble, top 4 bits of nibble (0x8) multiplied by 32
  can sanity check with size.
- Decode the flags (ACK and PSH)
- Timestamp not necessarily actual times, used for more advanced congestion
- Data is ASCII, handy thing to recognize
- People getting offset, mostly confused by reserved/flags. Count bits. Note things, like checksum is mandatory on TCP so can't be 0.
- It's a web request

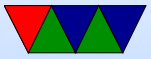


- Size: 0x46 = 70 bytes, 4/70 = 5.7%
  trouble counting bytes vs nybbles
  “useful data” issue this year



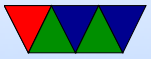
HW#6 Review – TCP Connections

- 3-way handshake SYN/SYN+ACK/ACK
  note also other things in packet, windowscale, sackOK (selective), TS, val, ecr (timestamp: value, echo-reply)
- Sends hi / ack / sends back HI / ack.
  Note PSH sent so that it doesn't wait and piggyback
  Why is PSH sent? Most(?) TCP stacks when you do a write() will set PSH on the last packet containing data from the write.
- Closing connection. FIN/ACK+FIN/ACK



HW#6 Review – Noticing Congestion

- Timeout
- Multiple duplicate ACKs
- Note: not multiple timeouts
- ECN can notice congestion, but in this case it happens before packets start getting lost (otherwise you'd never get the packets with the ECN info)



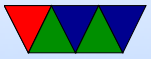
HW#6 Review – Security

- Network connections: Should you worry?
  - CLOSE-WAIT: received a FIN and ACKed it, waiting to close

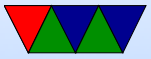
    Only a few, https and imap
  - ESTAB: established, a few ssh, https, imap connections
  - SYN-RECV: way too many, SYN flood

    Could a lot of legitimate ssh connections cause this?

    Unlikely. Have to stop handshake mid-way Why attack

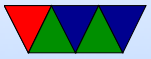


ssh? Have to try a socket someone is listening on
  - TIME-WAIT: connection closed, waiting a bit before re-using port
  - UNCONN – UDP listening. 789? ipp, mdns (multi-cast DNS, bonjour, can find names on network w/o running DNS), `lsof -i udp:789`, rpcbind
  - LISTEN – listening. Can see ipp (CUPS printing), netbios/microsoft, apparently have SAMBA running,
- Synflood, by default Linux uses SYN cookies to defend against this



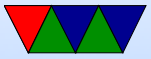
How do you get an IPv6 address?

- Manual (hard-coded)
- DHCPv6
- SLAAC



IPv6 Stateless Address Auto-Config (SLAAC)

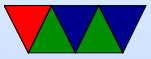
- IPv6 StateLess Address AutoConfiguration (SLAAC) assumes on /64 subnet (so every subnet contains orders of magnitude more than the total IPv4 space for their own local network)
- Essentially large enough a system could just pick a random address and it would work



SLAAC Methods

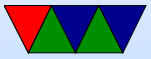
- Three ways:
  - EUI-64 (RFC 4291) – based on MAC address
  - Stable Private (RFC 7217) – hash based, don't give away MAC
  - Privacy Extension Addresses (RFC 4941) – like above but change over time to preserve anonymity

    For security refresh daily, this does happen on MacOS/Windows, but not necessarily on Linux



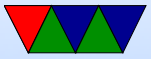
EUI-64 Link Local Example

- Linux seems to do this to set up link-local addresses
- Link-local is a non-routable IP address only used to your LAN (Local Area Network)
- IPv4 has these too but I've only ever seen microsoft use them
- For IPv6 addresses they are on fe80::/10



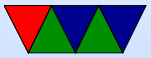
EUI-64 Link Local Example Continued

- Take MAC address (i.e. 8c:dc:d4:24:7d:45)
- Split up, put fffe in middle, flip bit 7 of top byte
  - 8c:dc:d4:24:7d:45
  - fe80::8edc:d4ff:fe24:7d45
- Can see these addresses with `ip addr`
- `ping6` can ping them on local network
- To ssh you have to specify interface, something like:
  `ssh fe80::8edc:d4ff:fe24:7d45%eth0`



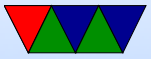
Duplicate Address Discovery (DAD)

- Once has link-local address, joins special multicast address
- `ff02::1:ffXX:XXXX` where last 6 bytes are bottom half of IP address it picks
- Sends packet to see if anyone else has address
- `ip maddr show` will show in-use multicast addresses



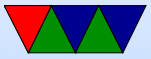
IPv6 Neighbor Discovery

- Neighbor Solicitation (NS) (RFC 4861) use with SLAAC described in RFC 4862 to get address
- Once has address, does DAD
- Once has link local address, sends out Router solicitation (RS) to multicast address ff02::2
- Router replies with (RA) router advertisement packet with info on router, maybe DNS, etc
- Now needs to get global routable address prefix: gets directly or has bits set to indicate it should use DHCPv6



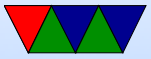
IPv6 DHCPv6

- Can provide info just like IPv4
- Not just router info, but also things like DNS servers, etc



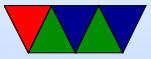
IPv6 Setup

- I've set up many many IPv4 networks, not any IPv6
- `https://lwn.net/Articles/831854/`
  Article by James Bottomley
- With IPv4, DHCP can take care of everything



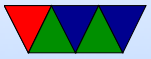
IPv6 setup issues

- It can be hard to subnet.
- It's recommended an ISP gives you a /56 but often they will just give you a /64
- That's a lot of addresses, but due to SLAAC it's assumed a network has a minimum of 2^{64} addresses so you can't split it up easily
- Annoying if you want multiple subnets at home (for wireless, DMZ, etc)
- Setting up Firewall. Linux has separate ipv4 and ipv6



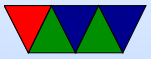
firewalls

- Having a NAT set up sort of gives you a firewall for free, you don't necessarily get that with IPv6



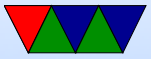
IPv6 Security Issues

- Shadow Networks – if you have a primarily ipv4 setup but various devices start up IPv6 connections without you realizing it
- Fragmentation – even though only on ends, can still have issues like IPv4 where it's hard to handle fragments as TCP port info and such only in first fragment



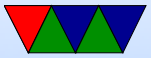
Modern IPv6 vs NAT Concerns

- Performance, NAT takes extra processing. Can small routers keep up at 1Gbps?
- Security, implicit security in NAT (internal devices not visible at all unless open outgoing connection). Can configure a firewall for ipv6 but requires extra work
  Also to get similar NAT-like behavior (blocked by default unless outgoing connection) maybe difficult
- Generally ipv6 not used by as many so dependent on your ISP not breaking things and not noticing



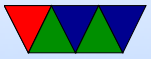
IPv4 / IPv6 Interop

- IPv6 NAT? What would that even mean?
- Can you have an internal network that's IPv4 connected to an external IPv6 network?
- Can you have an internal network that's IPv6 connected to an external IPv4 network?
  - Dual stack. Run both IPv4 and IPv6. Can fall back if one doesn't work. Need to configure two parallel network infrastructures.
  - Stateless IP/ICMP Translation



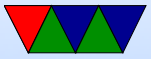
Internally fully IPv6, but each server has equivalent IPv4 on outside and the router converts them

- Tunneling, IPv6, tunnel/encapsulate inside of IPv4, then return to IPv6
- NAT64 – IPv6 internally, but has single external IPv4 gateway and does NAT/conversion to inside
- 464XLAT – nat64 at network level, SIIT internal
  Used in carrier-grade type situations, PLAT/CLAT



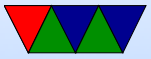
IPv6 Socket Programming

```
struct sockaddr_in6 server_addr;

sock_fd = socket(AF_INET6, SOCK_STREAM, IPPROTO_TCP);

server_addr.sin6_family = AF_INET6;

inet_pton(AF_INET6, "::1", &server_addr.sin6_addr);

server_addr.sin6_port = htons(SERVER_PORT);
```



Multistack (IPv4 or IPv6) Sockets

- Can you open a socket that handles both IPv4 and IPv6?
- On most machines, yes
- Create IPv6 socket
- There's an option you can disable, IPV6_V6_ONLY
- With that disabled can accept incoming connections for either IPv4 or IPv6
- Linux (possibly) allows by default. Other OSes (including BSD and Windows) are IPv6 only by default.



Multistack Sockers for Client

- You can do this too
- `getaddrinfo()` is smart enough when you lookup by name on DNS to in theory pick IPv6 or IPv4 for you and set up the various structs properly
- TODO: write some sample code for both the above cases