ECE 435 – Network Engineering
Lecture 14

Vince Weaver

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

vincent.weaver@maine.edu

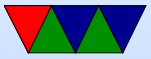
12:30pm, Barrows 127

29 February 2024

Announcements

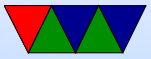
- HW#6 due Friday
- Will try to grade HW#5 and HW#6 before midterm
- HW#7 will be posted
- In 2021 Pentagon activated some of its vast IPv4 collection turns out had been unused people using them as unroutable numbers, including China military.

  https://www.theregister.com/2021/04/26/defense_department_ipv6/
- Office hours cancelled Monday due to faculty interview schedule



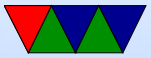
Midterm Preview

- 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.

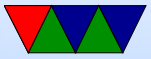


Midterm Preview – Topics

- 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)

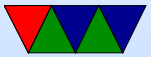


- UDP + TCP
  - Know the 3-way handshake
  - Know the tradeoffs between UDP and TCP
  - Why does DNS use UDP
  - Why does HTTP1.1 use TCP



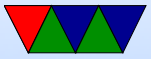
Brief HW#5 Review

- source/destination/size/checksum
  - src: a9a0 = 43424 (note, hex dumps are naturally big endian)
  - dest: 35 = 53 (DNS)
  - size: 2a = 42 bytes
  - yes checksum
  - protocol is DNS (how can you tell?)
- Why use UDP vs TCP
  lower latency, lower overhead (no need to handshake),



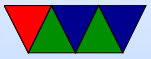
simpler

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

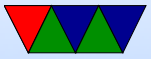


HW#5 Coding Notes

- 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

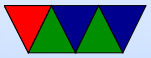


The IPv4 Catastrophe



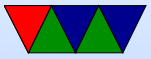
Out of IPv4 Addresses Problem

- IPv4 address exhaustion
- CIDR not enough
- Addresses managed by IANA globally and five regional registrars (RIR)
- Top level ran out in 2011
- All 5 RIRs finally ran out on Nov 25th, 2019

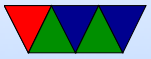


Why are we out?

- Always active connections – unlike dialup, many machine are on all the time
- So many devices – 4G mobile devices all have one
- Inefficiencies originally handing out. Companies like Apple, MIT, DEC, all got 16 million address Class A addresses even if didn't need them
  (Stanford gave back a class A in 2000)
- Despite being out, in 2011 reportedly only 14% of addresses being used

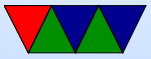


- Why not reclaim unused, such as Class E? The bane of network programmers, the out-of-date router that makes assumptions



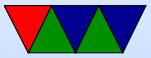
Ways to mitigate lack of addresses

- Add extra bits for addresses in ipv4 in a backward compatible way (this was generally determined to not be practical)
- Replace ipv4 with new protocol
- Have private subnetworks live behind a gateway that only requires one IPv4 address

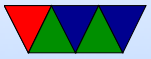


Network Address Translation (NAT)

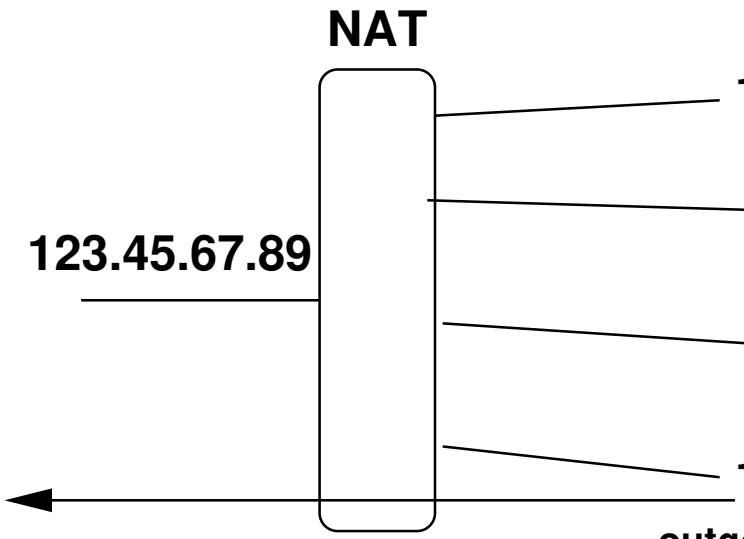
- Private IP ranges, defined in RFC 1918
  - 1 Class A: 10.0.0.0 - 10.255.255.255 (10.0.0.0/8)
  - 16 Class B: 172.16.0.0 - 172.31.255.255 (172.16.0.0/12)
  - 256 Class C: 192.168.0.0 - 192.168.255.255 (192.168.0.0/16)
- Can use for various reasons, most recently due to network depletion
- NAT: map IP addresses from one group to another. often public to private.
- NAT and NAPT (port translation) RFC 3022



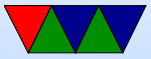
- Basic NAT has one to one mapping of external to internal IPs. Each internal host maps to unique external IP



NAT Example

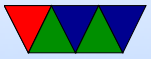


outgoing connection to 8.8.8.8 port 53
source is 192.168.8.5 port 40000
NAT remaps to 123.45.67.89 port 60000
incoming data to 123.45.67.89 60000 remapped back to 192.168.8.5 port 40000



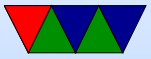
Network Address Port Translation (NAPT)

- NAPT: based on port, only one external IP
  - Full cone – most common
    once an internal address (iaddr/port) has been mapped to an external (eaddr/port) all packets from iaddr/port are sent out and any incoming are passed through with no additional checks
  - Restricted cone – same as above, but only external that have received packets from internal can send through
  - Port restricted cone – same as above, but also checks



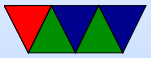
port numbers

- Symmetric – best security – outgoing packets mapped to different eaddr/port if the destination or port differs



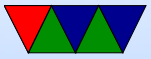
NAT Implementation

- When passing through, NAT needs to re-write dest/source/port and recompute header checksum
- Linux: IP-masquerade/iptables



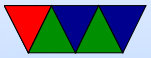
Many IP people hate NAT

- Violates the IP identifies one machine rule
- Hard to connect two machines if both behind different NATs (NAT transversal)
- Changes IP to be connection oriented, router has to remember connections
- Layering violation, looks at TCP/UDP port numbers
- Only works for TCP/UDP
- Some protocols (like FTP) are even more annoying, send address in plain text in data and that has to be adjusted



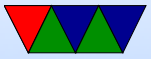
too

- Can only NAT up to 64k machines (why? how many ports are there?)



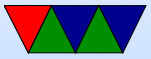
Carrier Grade NAT (CGN, CGNAT, LSN)

- Internal network uses private IP range
- Public facing server channels these through a set of external IP addresses
- NAT444 – potentially traverse 4 different IP (private in home, private in ISP, external IP)
- RFC6598 – allocate 100.64.0.0/10 for this, to avoid complications where internal/external collisions of the RFC1918 ranges



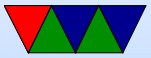
CGNAT Downsides

- Breaks end-to-end connections
- Stateful
- Doesn't fully solve IPv4 exhaustion problem cases where need a visible IP address (SSL web server?)
- Lots of devices behind a few IPs, what if get banned for spamming/security?
- Breaks port-forwarding for users, as you're in a NAT inside a NAT (port control protocol (PCP) RFC 6887 tries to work around this)



The Internet Protocol v6

- RFC2460 - RFCC466
- Started work in 1991
- Many problems with IPv4. Most notable shortage of addresses.
- IPng. (IPv5 was an experimental stream protocol)
- Migration happening, a large amount of web traffic, especially that from phones, is already switched.
- *not* backwards compatible

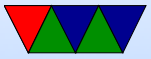


IPv6 uptake

- As of July 2016 12.5% of traffic is IPv6
- According to Google connecting to network

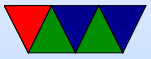
  `https://www.google.com/intl/en/ipv6/statistics.html`

  - March 2022: 34%
  - March 2023: 38%
  - March 2024: 43%
- This worldwide, some countries higher (US 48%, India/France 75%)



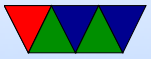
The Internet Protocol v6 Goals

- Support billions of hosts
- Reduce size of routing tables
- Simplify the protocol (so routers can be faster)
- Better security
- Pay more attention to type of service
- Aid multicasting
- Allow roaming w/o changing address
- Co-exist with existing protocols

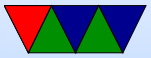


The Internet Protocol v6 features

- Address size 128 bits
  a lot of addresses. 7×10^{23} for ever square meter
- Simpler fixed length header (speeds up processing)
  Many fields not really used in IPv4 dropped (or made optional)
- Better support for options
- Better security support
  IPSEC. Originally mandatory, made optional
  Can encrypt packets at the network layer

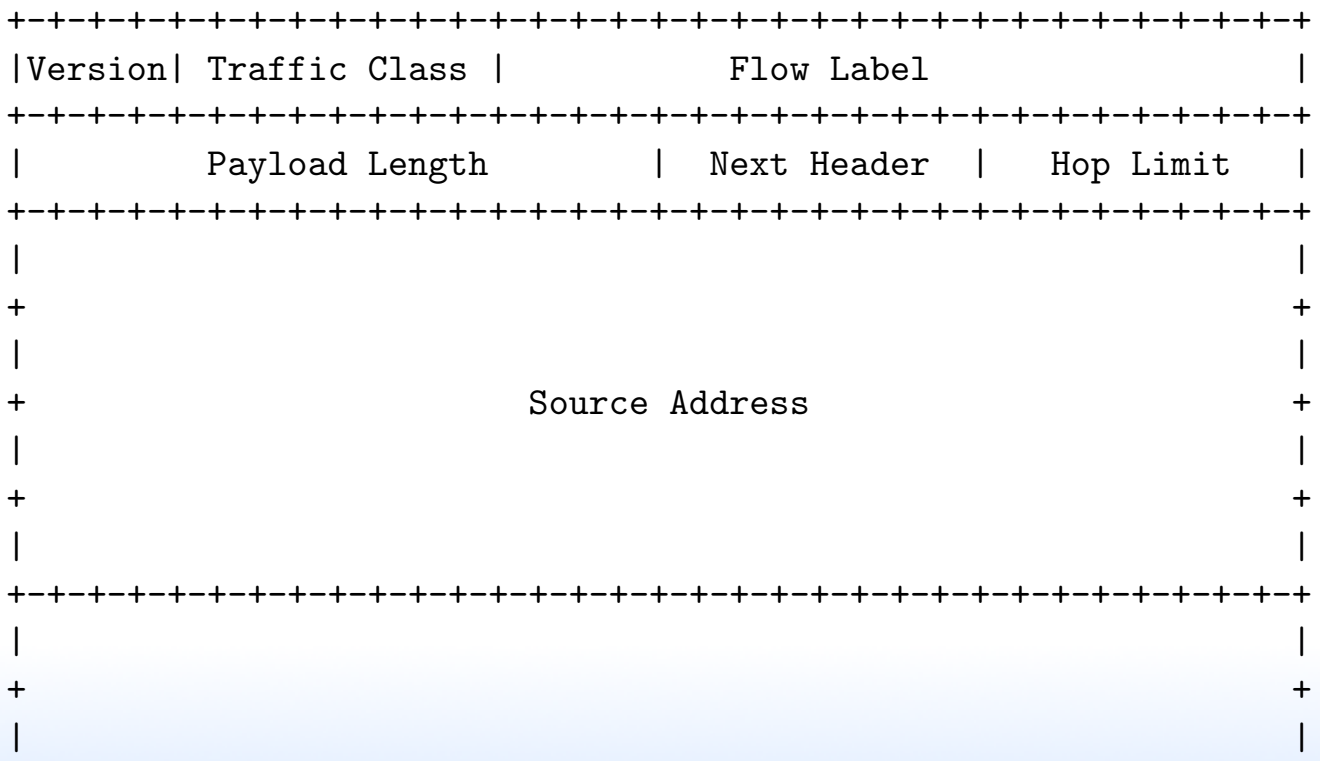


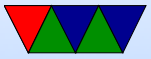
- Quality of service (???)
- Anycast (see end of slides)
- Autoconfiguration (like DHCP)
- Minimum fragment size 1280 (up from 576)
- No checksum – was slow and recalculated often

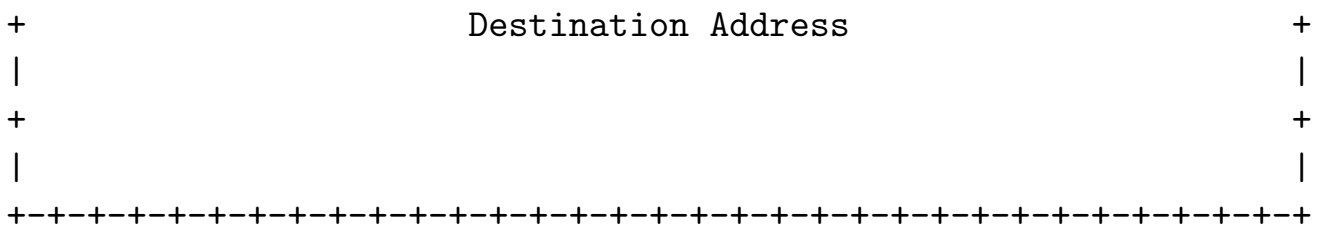


IPv6 header

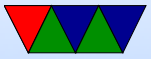
- Header fixed length of 40 bytes, with Extension headers
- ASCII art from RFC 2460





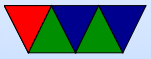


- **Version Number** (1 nibble [4-bits]) = 6
- **Differential Services/Traffic Class** (8-bits) (QoS/congesti
  6-bit differential services, 2 bits ECN (sort of like recent IPv4?)
- **Flow label** (20-bits) (for streaming?) (Recently for ECMP (Equal Cost Multipath) Line sharing. Sending packets different router paths can be bad with TCP as packets more likely to arrive late/out-of-order. Instead

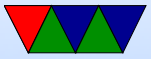


send all packets with same flow label same path, but balance different paths)

- **Payload Length** (16-bits) 64k (header bytes not counted anymore) (if you want longer, extension for Jumbograms (up to 4GB))
- **Next header** (8-bits) If nothing special identifies TCP or UDP If special options (fragmentation, security) indicated
  (TCP=0x6, UDP=0x11)
- **Hop Limit** (8-bits) TTL, big debate about whether 8-bits was enough

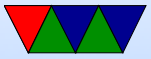


- **Source Address** (128-bits)
- **Destination Address** (128-bits)
- Why not 64-bit addresses?
- No checksum, link or transport catches issues
  What does this mean for UDP?



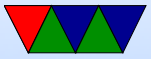
IPv6 addresses

- 2^{128} is a lot. $7x10^{23}$ per m^2 of Earth surface
- Too long for dotted decimal, use colon hexadecimal
- Why colons? .BE is a valid domain ending for one...
- X:X:X:X:X:X:X:X where X is 16 bit chunk
- F000:0123:5678:0000:0000:ABCD:0001:CAFE
- Can drop leading zeros, as well as groups of zeros
  F000:123:5678::ABCD:1:CAFE
  Note, cannot drop two sets of groups of zeros. Why? Ambiguous.



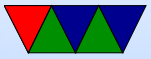
IPv6 reserved addresses

- ::1 ip6-localhost, fe00::0 ip6-localnet
- fe80:: link-local?
- 2001:: special? reserved?
- 2002:: 6to4 (deprecated)
- 64:: more 6to4 (?)
- ff00:: multicast

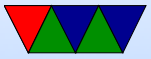


IPv6 Options

- Happen immediately after the header.
- Should occur in numerical order (though routers might be able to handle if they don't)
- Routers should inspect in order as some later might depend on earlier.
- Plain: IPV6:Next=TCP, TCP, Data
- Example: IPV6:Next=Routing, Routing:Next=TCP, TCP, Data
- Various types

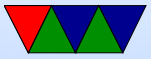


- Hop-by-hop (so far only used for jumbo frames, if that set header length is set to 0)
- Source Routing
- Fragmentation
- Authentication
- Encryption

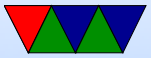


IPv6 fragmentation

- Info is in an extension header
- Routers cannot fragment, only at source
- How can this work when not know MTU?
- MTU is always greater than 1280
- Path MTU discovery protocol to discover MTU along the way (RFC 1981). (IPv4 too, set DNF and get error via ICMP) If too big, sends an error back and source needs to fragment it smaller
- Easier to have source fix things then every router along

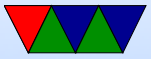


the way being able to



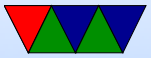
IPv6/IPv4 compat

- Dual stack – host runs both IPv4 and IPv6
  or internal is IPv6 but router converts to IPv4 before passing on
- Tunneling – encapsulate IPv6 inside of IPv4, tunnels across IPv4, split back out to IPv6 on other side of tunnel
- IPv4 mapped to IPv6 with special 96-bit prefix
  there are specs for this, not sure if really implemented

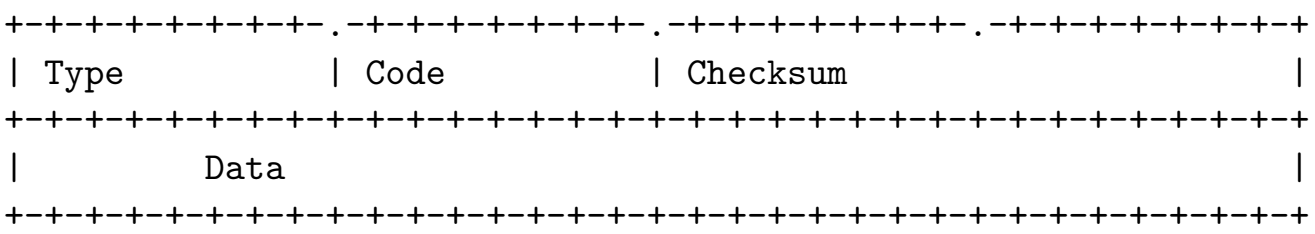


IPv6 Routing

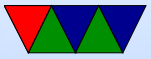
- Much like IPv4
- IPv6 network address, “prefix-length” instead of netmask
- Routing table as before



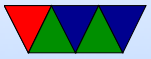
ICMP6



- Checksum: similar to TCP, also includes pseudo-header
- Type 0, Destination Uncertain
  - Various
- Type 1, Packet too big
- Time Exceeded
- Bad Parameters

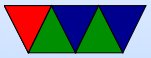


- ECHO / ECHO Reply
- Neighbor Discovery Protocol (NDP)
- SEND – Secure Neighbor Discovery Protocol
- Multicast Listener Discovery (MLD)



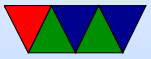
IPv6 Anycast / Multicast

- We'll talk about this later when we talk about broadcast



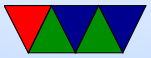
How do you get an IPv6 address?

- Manual (hard-coded)
- DHCPv6
- SLAAC



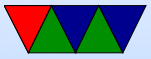
IPv6 Stateless Address Auto-Config (SLAAC)

- 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



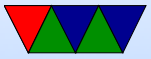
SLAAC Methods

- 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

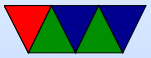


EUI-64

- Linux seems to do this to set up link-local addresses
- Link-local, non-routable address on fe80::/10
- 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 do specify interface, something like:

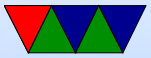


```
ssh fe80::8edc:d4ff:fe24:7d45%eth0
```



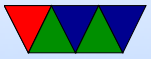
Duplicate Address Discovery (DAD)

- 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 it anyone else has address
- `ip maddr show` will show in-use multicast addresses

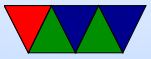


IPv6 Neighbor Discovery

- 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 to use, either directly or has bits set to indicate it should use

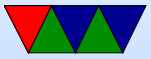


DHCPv6



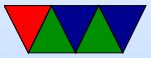
IPv6 DHCPv6

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



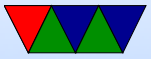
IPv6 Setup

- 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



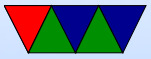
IPv6 setup issues

- 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



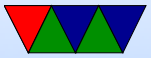
firewalls

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



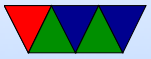
IPv6 Security Issues

- 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



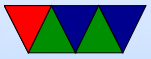
Modern IPv6 vs NAT Concerns

- 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



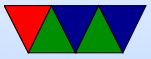
IPv4 / IPv6 Interop

- 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);
```