ECE 435 – Network Engineering
Lecture 15

Vince Weaver

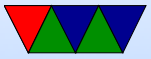
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

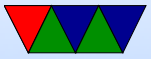
10 March 2022

Announcements

- HW#7 was posted, due after break Last year Pentagon activated some of its vast IPv4 collection turns out had been unused people using them as unroutable numbers, including China military. Oops.
- Question from last time: about ID field in IPv4. Actually quite complex and some RFCs conflict. Often you can just set it to 0 if DNF. Otherwise you set it, but can't find much info on if you just increment or should do random or what. It can wrap pretty quickly on modern

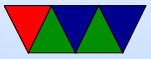


network speeds. Also people like to try to re-use it for other uses if DNF is set but it's unclear if all routers will be OK with that.



HW#5 Coding Notes

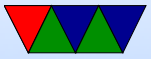
- Remember to comment your code!
- Getting source port from incoming connection
- Note this is not the IP address
- Getting it from the struct is sort of hard
- Also remember it's in network endian, need to convert with ntohs() In general would be an ephemeral port above 40000



HW#6 Review

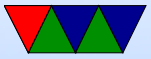
- Header length was the most trouble, top 4 bits of nibble (0x8) multiplied by 32
  can sanity check with size.
- Decode the flags
- Timestamp not necessarily actual times, used for more advanced congestion
- Data is ASCII, handy thing to recognize

```
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
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
```

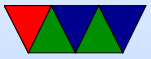
- It's a web request
- Size: 0x46 = 70 bytes, 4/70 = 5.7%
- 3-way handshake SYN/SYN+ACK/ACK
- Sends hi / ack / sends back HI / ack. Note PSH sent



so that it doesn't wait and piggyback

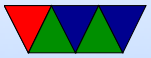
- Closing connection. FIN/ACK+FIN/ACK
- Network connections
  - CLOSE-WAIT: received a FIN and ACKed it, waiting to close

    Only a few, https and imap
  - ESTAB: established, a few ssh, https, imap connections
  - SYN-RECV: way too many, SYN flood
  - TIME-WAIT: connection closed, waiting a bit before re-using port

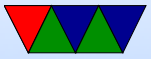


- UNCONN – UDP listening. 789? ipp, mdns (multicast DNS, bonjour, can find names on network w/o running DNS), lsof -i udp:789, rpcbind
- LISTEN – listening. Can see ipp (CUPS printing), netbios/microsoft, apparently have SAMBA running,

- Synflood, by default Linux uses SYN cookies to defend against this

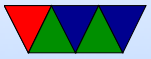


IPv4 Catastrophe



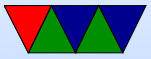
Out of IPv4 Addresses Problem

- 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



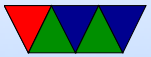
- So many devices – 4G mobile devices all have one
- Inefficiencies originally handing out. Companies like Apple, MIT, DEC, all got 16 million address Class A addresses even if didn't need them
- Despite being out, in 2011 reportedly only 14% of addresses being used

- Why not reclaim unused, such as Class E? The bane of network programmers, the out-of-date router that makes assumptions
- Stanford gave back a class A in 2000

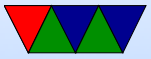


Network Address Translation (NAT)

- 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

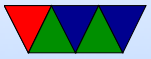


- Basic NAT has one to one mapping of external to internal IPs. Each internal host maps to unique external IP

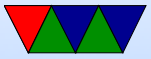


Network Address Port Translation (NAPT)

- 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

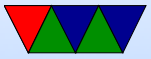


port numbers
  - Symmetric – best security – outgoing packets mapped to different eaddr/port if the destination or port differs
- When passing through, NAT needs to re-write dest/source/port and recompute header checksum
- Linux: IP-masquerade/iptables



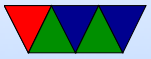
Many IP people hate NAT

- Violates the IP identifies one machine rule
- Hard to connect two machines if both behind different NATs (NAT tranversal)
- 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



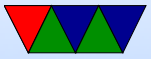
too

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

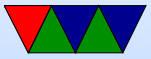


Carrier Grade NAT (CGN, CGNAT, LSN)

- 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
- Breaks port-forwarding for users, as you're in a NAT

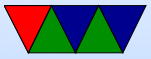


inside a NAT (port control protocol RFC 6887 tries to work around this)

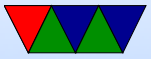


The Internet Protocol v6

- 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
- As of July 2016 12.5% of traffic is IPv6

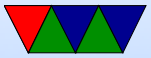


According to Google, March 2022 around 34%



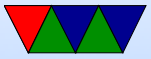
The Internet Protocol v6 Goals

- 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

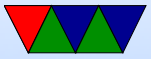


The Internet Protocol v6 features

- 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

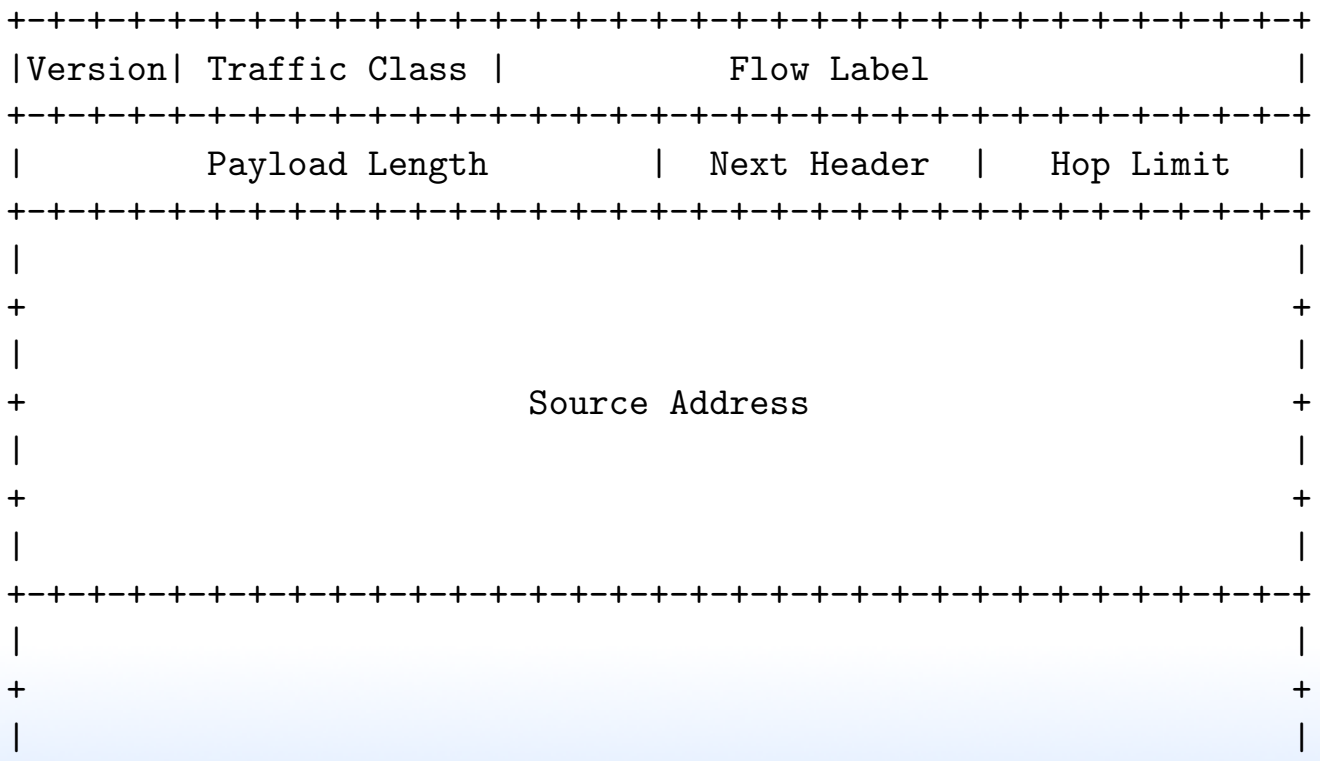


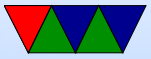
- Quality of service (???)
- Anycast (see end of slides)
- Autoconfiguration (like DHCP)
- Minimum fragment size 1280 (up from 576)
- No checksum – was slow and recalculated often

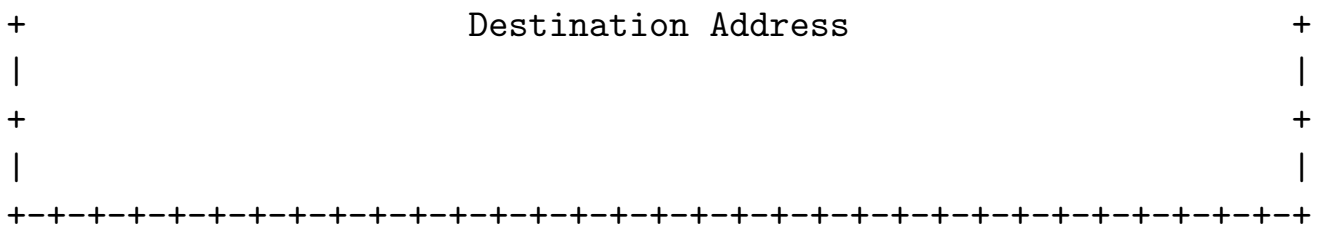


IPv6 header

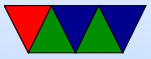
- Header fixed length of 40 bytes, with Extension headers
- ASCII art from RFC 2460





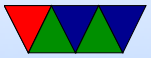


- **Version Number** (1 nibble [4-bits]) = 6
- **Differential Services/Traffic Class** (8-bits) (QoS/congesti
- **Flow label** (20-bits) (for streaming?)
- **Payload Length** (16-bits) 64k (header bytes not counted anymore) (if you want longer, extension for Jumbograms)
- **Next header** (8-bits) If nothing special identifies TCP or UDP If special options (fragmentation, security)



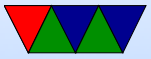
indicated
(TCP=0x6, UDP=0x11)

- **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?

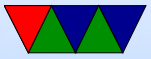


IPv6 addresses

- 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.

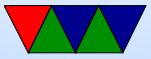


- Do not have classes RFC 4291
- encoding ipv4 addresses in ipv6?
- autoconfig – generate a unique ID, often based on MAC address and send it to router (Router solicitation). Router sends back router advertisement which has subnet prefix and can be used to generate global address
  Stateless address auto-config (SLAAC). Can also use DHCPv6
- ::1 ip6-localhost, fe00::0 ip6-localnet

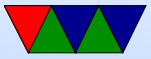


IPv6 Options

- 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

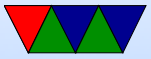


- Hop-by-hop (so far only used for jumbo frames, if that set header length is set to 0)
- Source Routing
- Fragmentation
- Authentication
- Encryption

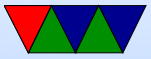


IPv6 fragmentation

- 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

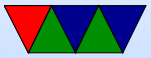


the way being able to



IPv6 Privacy Concerns

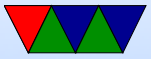
- Security – have MAC in your IPv6 address?



IPv6/IPv4 compat

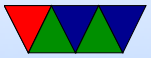
- Dual stack – host runs both IPv4 and IPv6
  or internal is IPv6 but router converts to IPv4 before passing on

- Tunneling – encapsulate IPv6 inside of IPv4, tunnels across IPv4, split back out to IPv6 on other side of tunnel



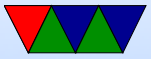
IPv6 Routing

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

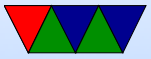


IPv6 Setup

- 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 StateLess Address AutoConfiguration (SLAAC) assumes on /64 subnet (so every subnet contains 1000 times the IPv4 space)
- Essentially large enough a system could just pick a random address and it would work

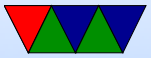


- Three ways:
  - EUI-64 (RFC 4291) – based on MAC address
  - Stable Private (RFC 7217) – hash based, don't give away MAC
  - Privacy Extension Addresses (RFC 4941) – like above but change over time to preserve anonymity
- No broadcast, so nodes coming up listen to multicast address
- Neighbour Solicitation (NS) (RFC 4861) use with SLAAC described in RFC 4862 to get MAC address
- One has address, sends out a packet to see if collision,



if not, it's configured

- 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 address
- Needs to figure out DHCP6 or SLAAC



IPv6 setup issues

- It can be hard to subnet. You get 2^{64} addresses but it's hard to split those up into multiple subnets
- Some ISPs will give up to a /56 but often not, hard if you want multiple subnets at home (for wireless, DMZ, etc)
- Setting up Firewall. Linux has separate ipv4 and ipv6 firewalls