ECE 435 – Network Engineering Lecture 18

Vince Weaver

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

vincent.weaver@maine.edu

6 March 2026

Announcements

- HW#6 will be posted
- Midterm on Wednesday March 11th (next week)
- Final project info posted to website
  - Topic selection by March 27th
  - Went over briefly some topic ideas

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
- IPv4

Which Protocols at Net Layer?

- Obsolete
  - IPX (IPX is net, SPX is transport)
  - Appletalk (DDP for routing)
- Current
  - IPv4
  - IPv6
  - ICMP

The Internet Protocol

- Last time talked about routing
- The Internet Protocol (IP) is used for routing packets across the internet
- Given the destination address, packet hops from router to router until gets to final address

IPv4 Addresses

- IP version 4 was the original version
- Each IPv4 address is 32-bits, split between network address and host ID
- Can write many ways: decimal, hex, (all equivalent) but most common is dotted decimal (i.e. `12.34.56.78`)
- Unique to *interface* not necessarily to *host*.
- Top level ran out in 2011, last NIC ran out 2019

Who Hands these Out?

- ICANN and various regional authorities Internet Corporation for Assigned Names and Numbers Internet Assigned Numbers Authority (IANA)
- Regional Internet Registrars
  - AfriNIC (Africa)
  - ARIN (N America),
  - APNIC (Asia-pacific)
  - LACNIC (latin america),
  - RIPE NCC (Europe and rest)

Subnets

- Having routing table for entire internet would be huge
- Instead address space split up into separate networks (subnets)
- All hosts on subnet have the same prefix (leftmost bits)

Subnet Masks

- Mask can be used to determine which bits are for network and which for host
- If top 24 bits describe network, 0xffffff00 (255.255.255.0)
- Alternately can write this as 192.168.8.0/24 (24 is number of leading binary 1s in mask)

Classful IP Routing (Not used since 1993)

- Class A: 8 bit network (high bit 0) (24 bits of hosts)

  0.0.0.0 to 127.255.255.255 (128 possible)
- Class B: 16 bit network, (high bits 10)

  128.0.0.0 to 191.255.255.255 (16384 possible)
- Class C: 24 bit network (high bits 110)

  192.0.0.0 to 223.255.255.255 (2M possible)
- Class D: multicast (high bits 1110)

  224.0.0.0 to 239.255.255.255
- Class E: reserved (high bits 1111) – 240.0.0.0 to 255.255.255.255

Classful IP Routing (No Longer Used)

- Why so simple? In 80s memory and processors were expensive!
- Network type can be found by looking at top 4 bits
- Routers shift right to separate prefix/host
- Looked up A and B in table, C in hash table to find where to send
- Had a routing entry for each Class A (128), an entry for each class B (16k). Class C (2 million) a bit much, so hash table (possible with data from slower storage)

Reserved IP Ranges

- Private Networks
  - 10.0.0.0/8 private network (RFC 1918)
  - 172.16.0.0/12 private network (RFC 1918)
  - 192.168.0.0/16 Private Network (RFC 1918)
- Loopback
  - 127.0.0.0/8 loopback (RFC 6890)

Reserved IP Ranges – Other

- 0.0.0.0/8 reserved for current network (RFC 6890)
- 100.64.0.0/10 shared address space (RFC 6598)
- 169.254.0.0/16 link-local (RFC 3927)
- 192.0.0.0/24 IETF (RFC 6890)
- 192.0.2.0/24 test (RFC 5737)
- 192.88.99.0/24 IPv6 to IPv4 relay (RFC 3068)
- 224.0.0.0/4 IP Multicast (class D) (RFC 5771)
- 240.0.0.0/4 Reserved (class E) (RFC 1700)
- 255.255.255.255/32 Broadcast (RFC 919)

Other IPv4 Conventions

- .0 represents a subnet
  See `https://lwn.net/Articles/850374/` really old UNIX treated .0 (or all host bits 0) as another broadcast, there's a push to reclaim it as unicast
- .1 is often (but not always) a router
- If all host bits 1, broadcast for that subnet
- 255.255.255.255 is broadcast for device that doesn't know own IP yet (DHCP)
- What if /31, address 0 and 1?

Classless Inter-Domain Routing (CIDR)

- RFC 1519
- Running out (have run out) of network addresses
- For many groups, Class-A too big, Class-C too small (three bears problem?)
- Merge neighboring class-C together
- Scalability problem: each network takes up space in routing table
- Solution, group neighboring class Cs together

CIDR Addressing

- Variable-Length Subnet Masking (VLSM)
  subnet sizes variable (not fixed like classful)
- Routing tables track triplet: IP address, subnet mask, outgoing line
- With CIDR some ranges can overlap, eg 44/9 and 44.128/10 so routers have to handle this. If multiple matches, one with longest mask is used.
- There are algorithms to make this go faster.

IP on local networks?

- If strictly on a LAN you don't need to use IP
- You could use some other protocol, and people did in the old days
- However TCP/IP is sort of the standard these days even on local networks

Local IP Routing

- If to same host, skip network.
- If on same subnet, send packet directly to destination (Ethernet)
- Otherwise, send to default router. See Linux route command. Often a "default router" 0.0.0.0/0. If doesn't match any other, sent out over default route
- If multiple network interfaces: If to this machine, deliver it, If to directly connected subnet, directly deliver, else deliver to next hop router

Local IP Routing Details

- How do we know if on network? If ((hostIP XOR destip)&subnetmask)==0
- If local, how do we map IP to MAC?
  ARP, We'll talk about this in a few lectures.
- Due to CIDR, longest prefix matching. If match both a /21 and /24 then 24 is the one to send to as it's the longest.

Routing in the OS

- Your OS can be configured to act as router if has multiple network interfaces
- Data structures. Hashes? Trie?
  - Linux: two level hashing
  - BSD - trie (prefix tree)

Linux/UNIX routing setup

- Was `route` command, has been replaced by `ip route`
- `route add default gateway` sets default gateway (router) for packets leaving the local network
- also set up local subnets you are on, those packets don't need a router
- more complicated if you are configuring your Linux box to *be* a router

Linux/UNIX routing example

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.8.2     0.0.0.0         UG    600    0        0 wlp2s0
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 wlp2s0
192.168.8.0     0.0.0.0         255.255.255.0   U     600    0        0 wlp2s0
```

IPv4 Packet Format

- Header, followed by data, multiple of 4-bytes, big-endian
- ASCII from RFC791 — `https://tools.ietf.org/html/rfc791`

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IPv4 Header – Version/Length

- **Version** (4-bits) version number: IPv4 this is 4
- **Header Length** (4-bits) in 4-byte chunks
  - Can vary in size
  - Often is 5 (20 bytes) the minimum
  - max is 15 (60 bytes)

IPv4 Header – Precedence / ToS

- **Precedence / Type of Service** (1 byte)
  - ○ Precedence (RFC 791, high bits):

    111 (net control)

    110 (internetwork control)

    101 (critic/ecp)

    100 (Flash override)

    011 (flash)

    010 (intermediate)

    001 (priority)

000 (routine)

- TOS (RFC 1349):

  1000 minimize delay

  0100 maximize throughput

  0010 maximize reliability

  0001 minimize cost

  0000 normal

  1111 maximize security
- R: reserved
- Replaced with DSCP (differentiated services code point) (RFC 2474) and ECN congestion (RFC 3168)

IPv4 Header – Length

- **Total Length** (2 bytes) – max is 64kB

IPv4 Header – Fragmentation

- More on this later...
- **Identification** (2 bytes) – also called sequence
- **Fragmentation** (2 bytes) – fragmentation:
  - ◦ **flags** (3 bits): for fragmentation control.
    high bit always 0, (joke April Fools proposal: ‘evil bit’)
    next is “do not fragment”
    last is “more fragments”
  - ◦ **fragmentation offset** (13-bits): all but last fragment must be a multiple of 8-bytes as only have 13 bits)

IPv4 Header – TTL

- **TTL** (1 byte) time-to-live, max routers allowed to pass though
  - (was supposed to be time, but ended up as a hop limit)
  - each router decreases TTL by one, if reaches zero discarded and ICMP error sent to source
  - Max is 255. why? prevent packets from wandering lost forever

IPv4 Header – Protocol / Checksum

- **Upper-layer protocol** (1 byte)
  Originally in RFC 1700, now see `www.iana.org` (ICMP=1, TCP=6, UDP=17) (many many more)
- **Header Checksum** (2 bytes)
  - Sum using 16-bit 1s complement, then complementing.
  - Not as strong as CRC-16, but faster and easier in software.
  - Only checksums header (not payload).
  - **Must be recomputed each hop as TTL changes**

IPv4 Header – Addresses

- **Source address** (4 bytes)
- **Destination Address** (4 bytes)

IPv4 Header – Options

- **Options** – not required. rare, debugging
  - security: how secret it is (usually ignored)
  - strict source: gives a list of IPs of routers to traverse
  - loose: list of routers not to miss
  - record route: record IPs pass on way (debugging)
  - timestamp(debugging)