Lecture 16: Network Layer Addressing, Control Plane, and Routing

COMP 332, Spring 2024

Victoria Manfredi

WESLEYAN UNIVERSITY

Acknowledgements: materials adapted from Computer Networking: A Top Down Approach 7th edition: ©1996-2016, J.F Kurose and K.W. Ross, All Rights Reserved as well as from slides by Abraham Matta at Boston University, and some material from Computer Networks by Tannenbaum and Wetherall.

Today

Announcements

- Homework 6 due next Wednesday by 11:59p

Internet protocol

Addressing

- IPV4 addresses
- usage in routing
- how to get an IP address

Network programming

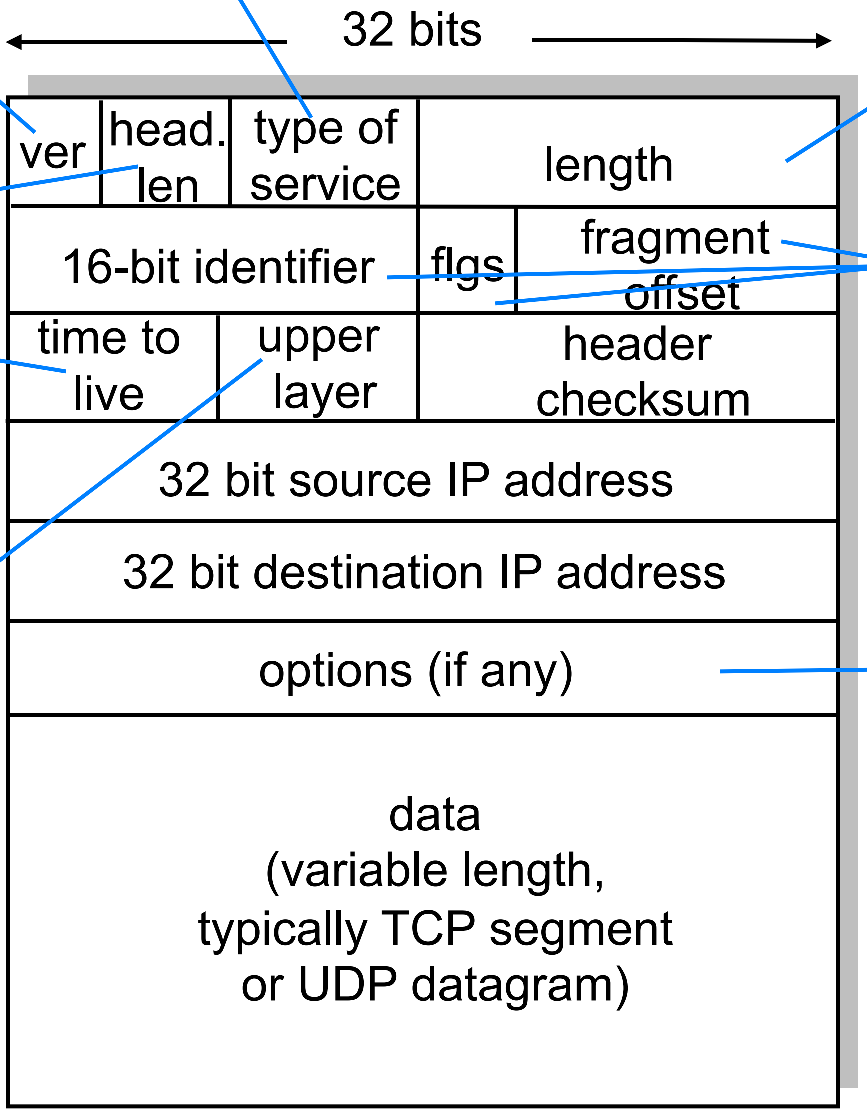
- raw sockets and byte packing

Network Layer

INTERNET PROTOCOL

IP packet format

Q: Why is version 1st?
So you know how to parse pkt



32 bits

ver | head. len | type of service | length

16-bit identifier | flgs | fragment offset

time to live | upper layer | header checksum

32 bit source IP address

32 bit destination IP address

options (if any)

data
(variable length,
typically TCP segment
or UDP datagram)

Not widely used

IP protocol version number

header length (# of 32bit words)

max number remaining hops (decremented at each router)

upper layer protocol to deliver payload to
ICMP=1, UDP=17, TCP=6

total packet length (bytes)

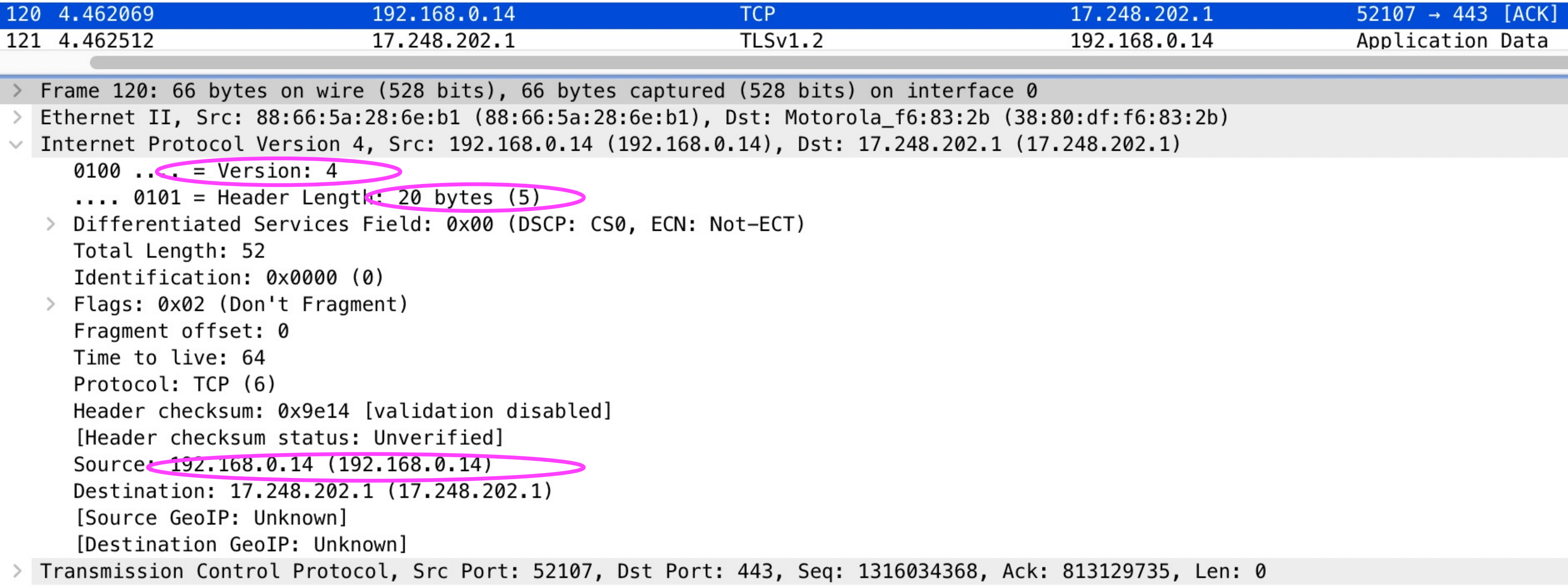
for fragmentation/ reassembly

e.g. timestamp, record route taken, specify routers to visit. When used usually indicate suspicious activity

How much overhead?
20 bytes of TCP
20 bytes of IP
= 40 bytes + app layer overhead

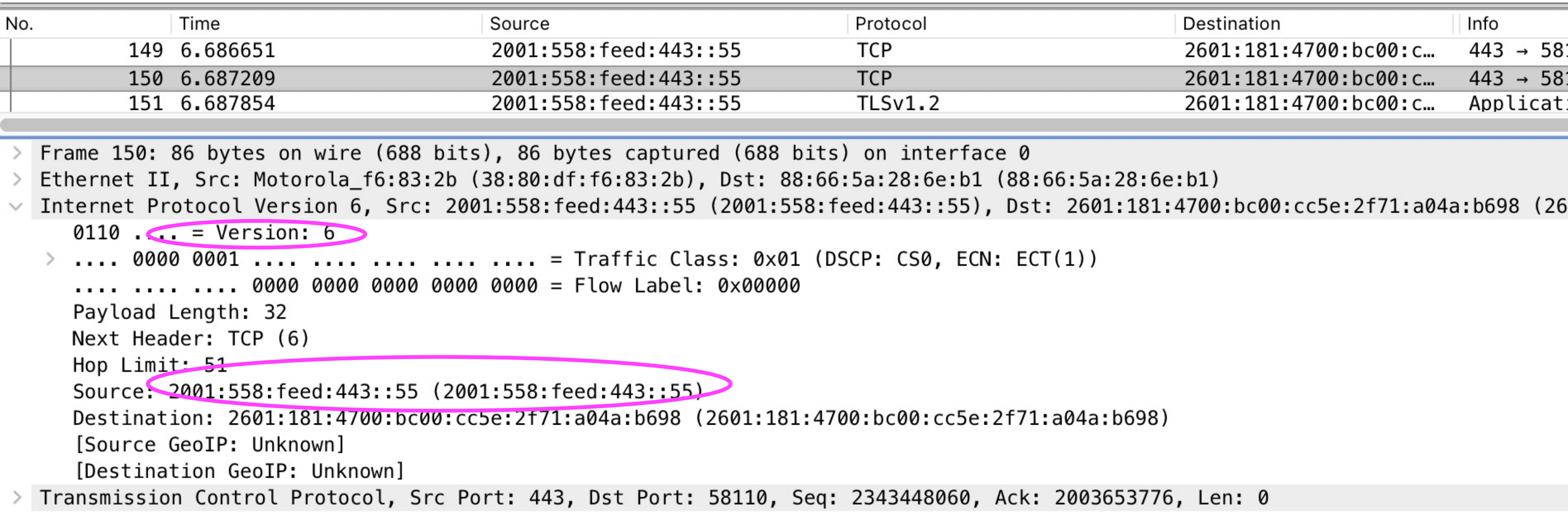
Bits transmitted left to right, top to bottom

Wireshark: IPv4



```
120  4.462069        192.168.0.14        TCP        17.248.202.1        52107 → 443 [ACK]
121  4.462512        17.248.202.1        TLSv1.2    192.168.0.14        Application Data

> Frame 120: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: 88:66:5a:28:6e:b1 (88:66:5a:28:6e:b1), Dst: Motorola_f6:83:2b (38:80:df:f6:83:2b)
v Internet Protocol Version 4, Src: 192.168.0.14 (192.168.0.14), Dst: 17.248.202.1 (17.248.202.1)
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 52
    Identification: 0x0000 (0)
  > Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0x9e14 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.0.14 (192.168.0.14)
    Destination: 17.248.202.1 (17.248.202.1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
> Transmission Control Protocol, Src Port: 52107, Dst Port: 443, Seq: 1316034368, Ack: 813129735, Len: 0
```

Wireshark: IPv6



No.	Time	Source	Protocol	Destination	Info
149	6.686651	2001:558:feed:443::55	TCP	2601:181:4700:bc00:c…	443 → 58
150	6.687209	2001:558:feed:443::55	TCP	2601:181:4700:bc00:c…	443 → 58
151	6.687854	2001:558:feed:443::55	TLSv1.2	2601:181:4700:bc00:c…	Applicat

```
> Frame 150: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
> Ethernet II, Src: Motorola_f6:83:2b (38:80:df:f6:83:2b), Dst: 88:66:5a:28:6e:b1 (88:66:5a:28:6e:b1)
v Internet Protocol Version 6, Src: 2001:558:feed:443::55 (2001:558:feed:443::55), Dst: 2601:181:4700:bc00:cc5e:2f71:a04a:b698 (26
    0110 .... = Version: 6
  > .... 0000 0001 .... .... .... .... .... = Traffic Class: 0x01 (DSCP: CS0, ECN: ECT(1))
    .... .... .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
    Payload Length: 32
    Next Header: TCP (6)
    Hop Limit: 51
    Source: 2001:558:feed:443::55 (2001:558:feed:443::55)
    Destination: 2601:181:4700:bc00:cc5e:2f71:a04a:b698 (2601:181:4700:bc00:cc5e:2f71:a04a:b698)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
> Transmission Control Protocol, Src Port: 443, Dst Port: 58110, Seq: 2343448060, Ack: 2003653776, Len: 0
```

Wireshark

Look at IP headers and ping/traceroute

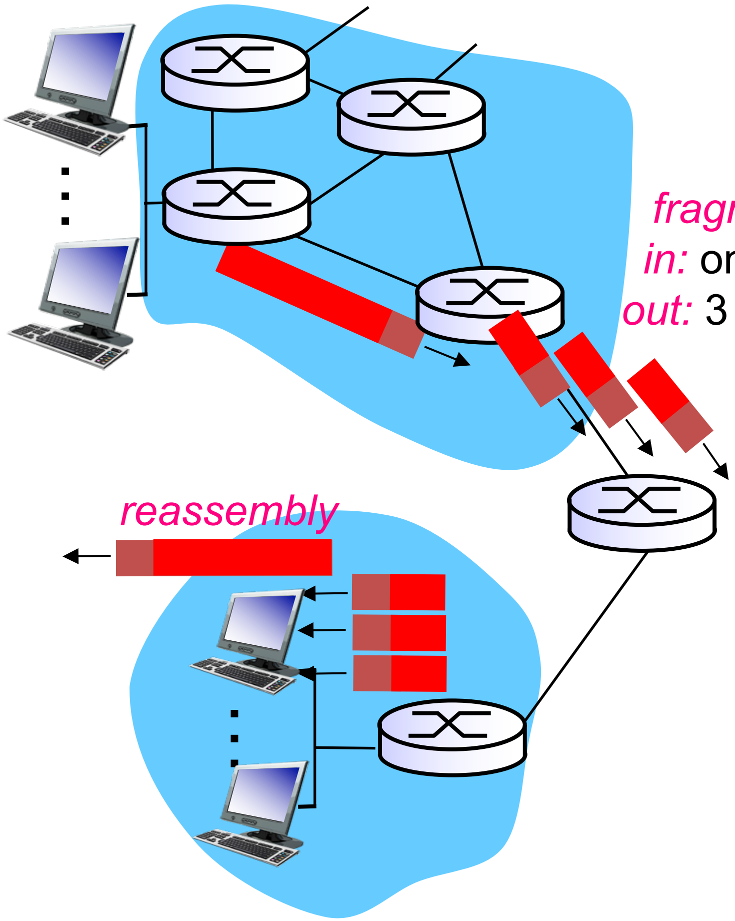
IP fragmentation and reassembly

Network links have MTU

- largest possible link-level frame
- different link types have different MTUs

Fragment when pkt > MTU

- 1 pkt becomes several pkts
  - IP header bits used to identify and order related fragments
- reassembled only at final dst
- re-fragmentation possible
- don’t recover from lost fragments
- (IPv6 does not support)



fragmentation:
in: one large pkt
out: 3 smaller pkts

reassembly

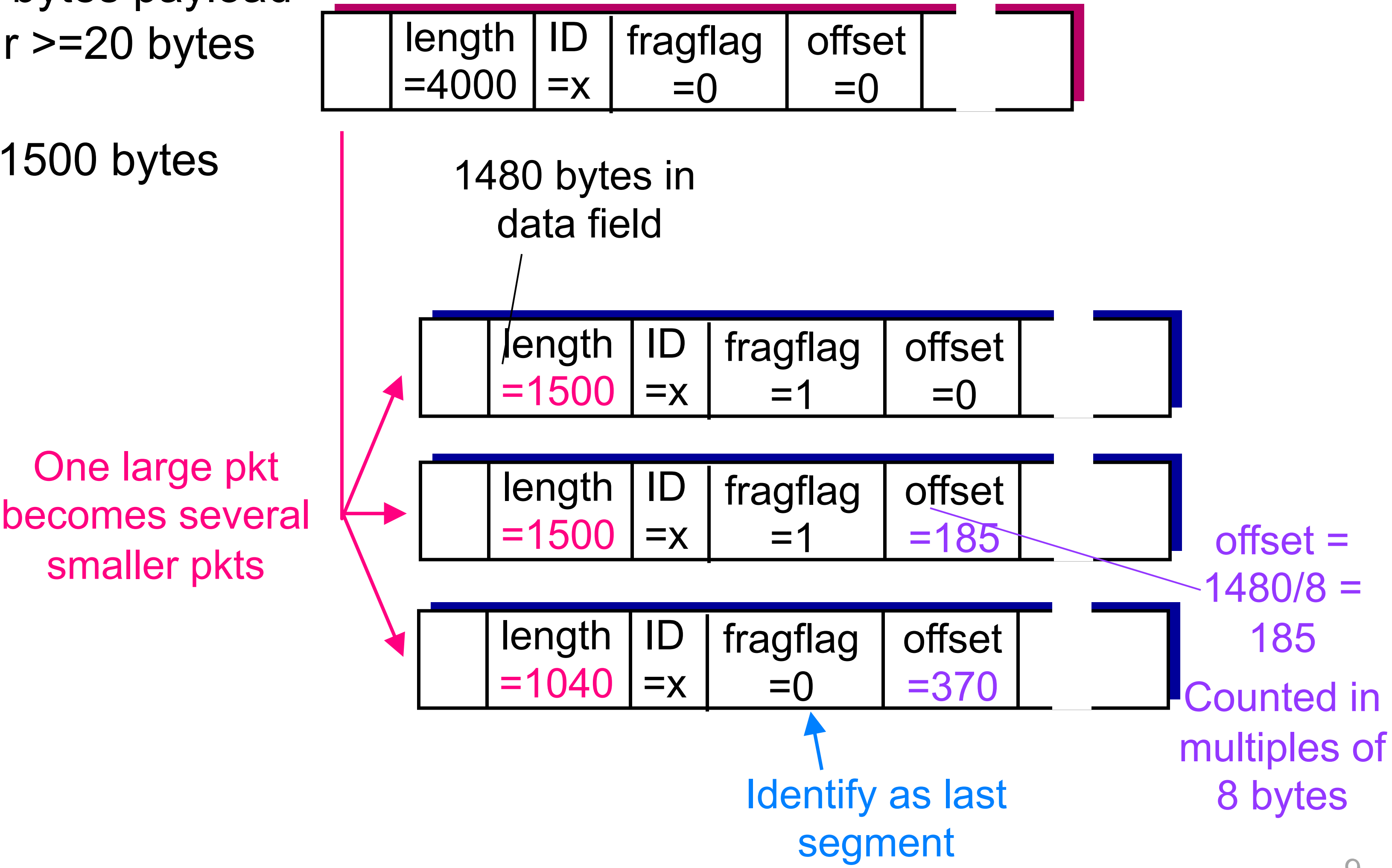
DoS attack: send fragmented pkts but leave one out

IP fragmentation and reassembly

4000 byte packet

- 3980 bytes payload
- IP hdr >=20 bytes

MTU = 1500 bytes



	length	ID	fragflag	offset	
	=4000	=x	=0	=0	

1480 bytes in data field

	length	ID	fragflag	offset	
	=1500	=x	=1	=0	
	=1500	=x	=1	=185	
	=1040	=x	=0	=370	

One large pkt becomes several smaller pkts

offset = 1480/8 = 185

Counted in multiples of 8 bytes

Identify as last segment

Addressing

IPV4 ADDRESSES

IPv4 addresses

Globally unique 32-bit identifier

- associated with host or router interface
- interface: connection between host/router and physical link
  - **host:** usually 1 or 2 interfaces
  - **router:** usually many interfaces

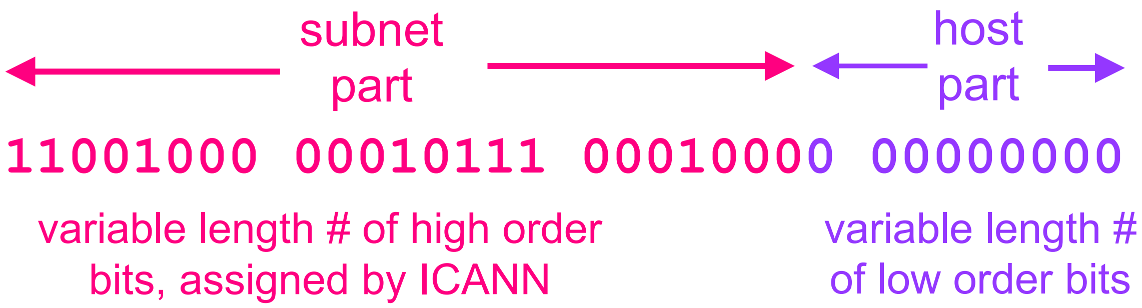
Address format is hierarchical

- CIDR: Classless InterDomain Routing
- split into subnet part and host part
  - a.b.c.d/x, where x is # bits in subnet part

IPv4 addresses

subnet part and host part

– a.b.c.d/x, where x is # of bits in subnet part



subnet part

host part

11001000 00010111 00010000 00000000

variable length # of high order bits, assigned by ICANN

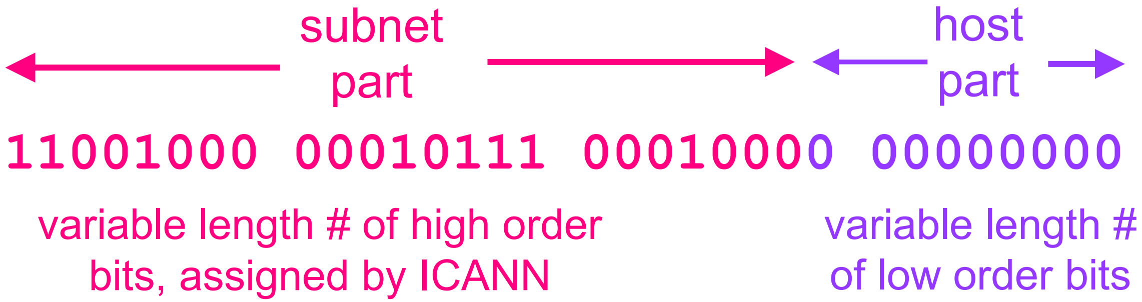
variable length # of low order bits

3 min: what is a.b.c.d for this? What is /x?

IPv4 addresses

subnet part and host part

- a.b.c.d/x, where x is # of bits in subnet part



subnet part

host part

11001000 00010111 00010000 00000000

variable length # of high order bits, assigned by ICANN

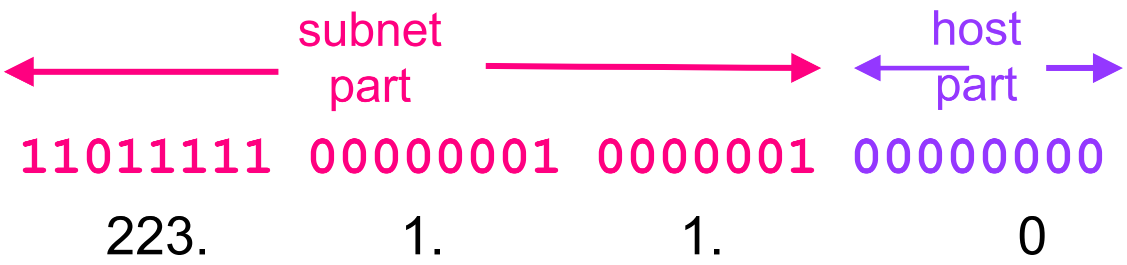
variable length # of low order bits

3 min: How many addresses in this block of addresses?

Dividing up an address block

Suppose given 223.1.1.0/24

- a.b.c.d/x, where x is # bits in subnet part



subnet part

host part

11011111 00000001 0000001 00000000

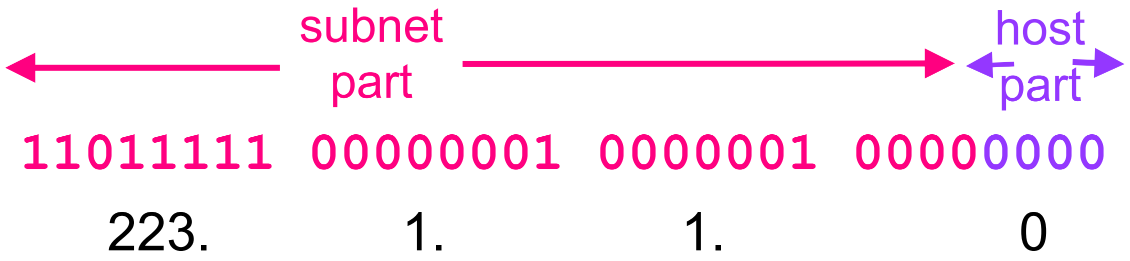
223. 1. 1. 0

How many addresses are there in this block?

Dividing up an address block

Suppose given 223.1.1.0/28

- a.b.c.d/x, where x is # bits in subnet part



subnet part

host part

`11011111 00000001 0000001 00000000`

223. 1. 1. 0

How many addresses are there in this block?

What's a subnet?

3 min: this network comprises how many subnets? Why?

Subnet

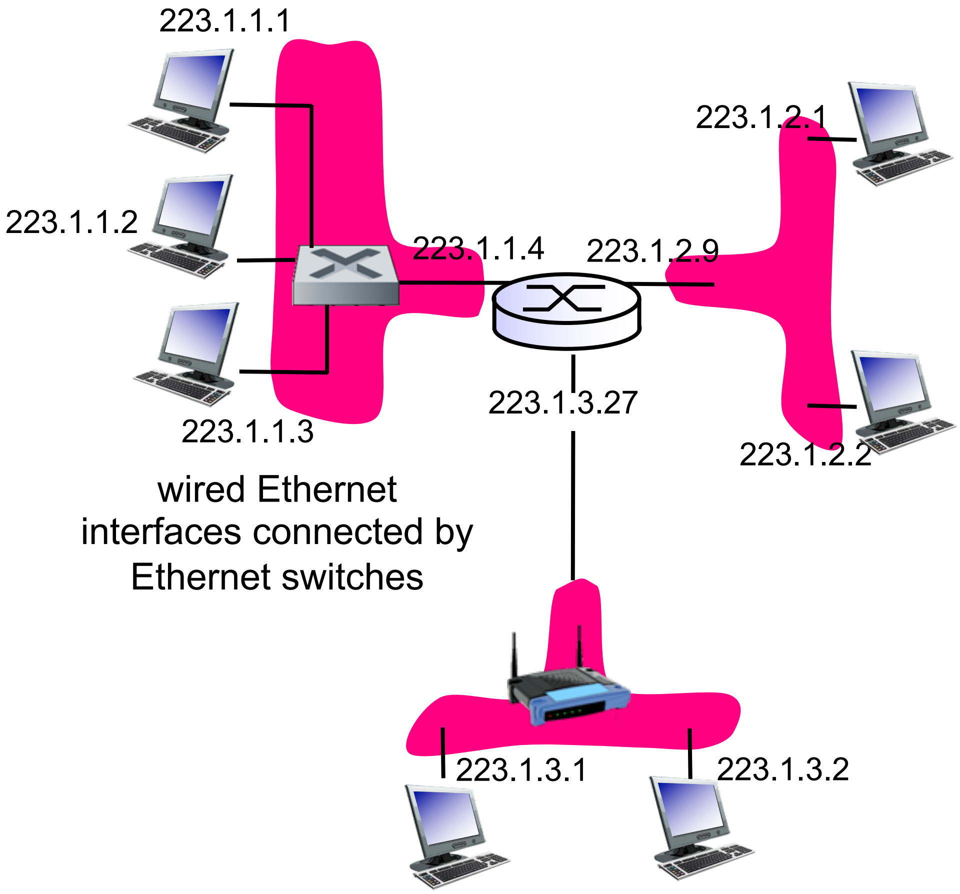
- set of interfaces with same subnet part of IP addr
- devices reachable without intervening routers

Subnet mask

- divides IP addr into subnet addr + host addr
- included in routing info given to routers

Recipe to find subnets

- detach each interface from its host or router
- create islands of isolated networks, i.e., subnets



wired Ethernet interfaces connected by Ethernet switches

wireless WiFi interfaces connected by WiFi base station

What's a subnet?

Subnet

- set of interfaces with same subnet part of IP addr
- devices reachable without intervening routers

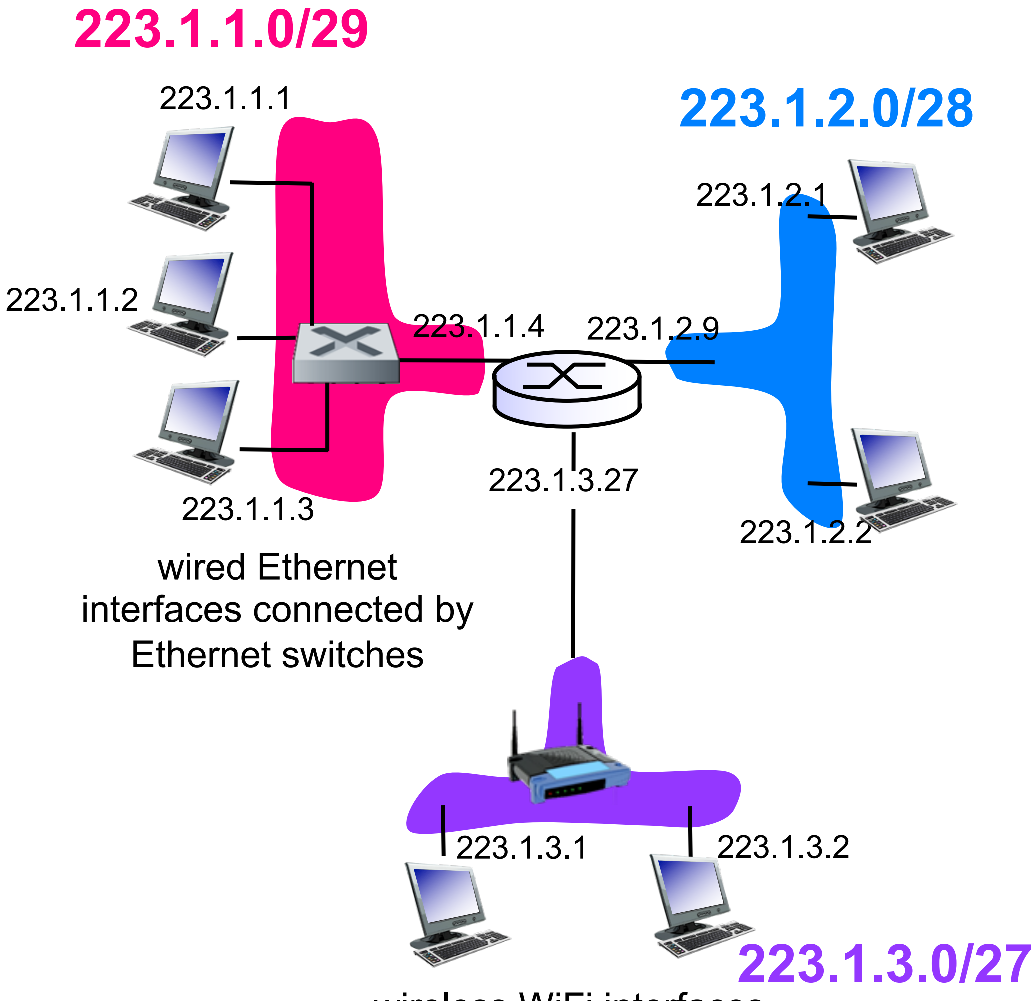
Subnet mask

- divides IP addr into subnet addr + host addr
- included in routing info given to routers

Recipe to find subnets

- detach each interface from its host or router
- create islands of isolated networks, i.e., subnets

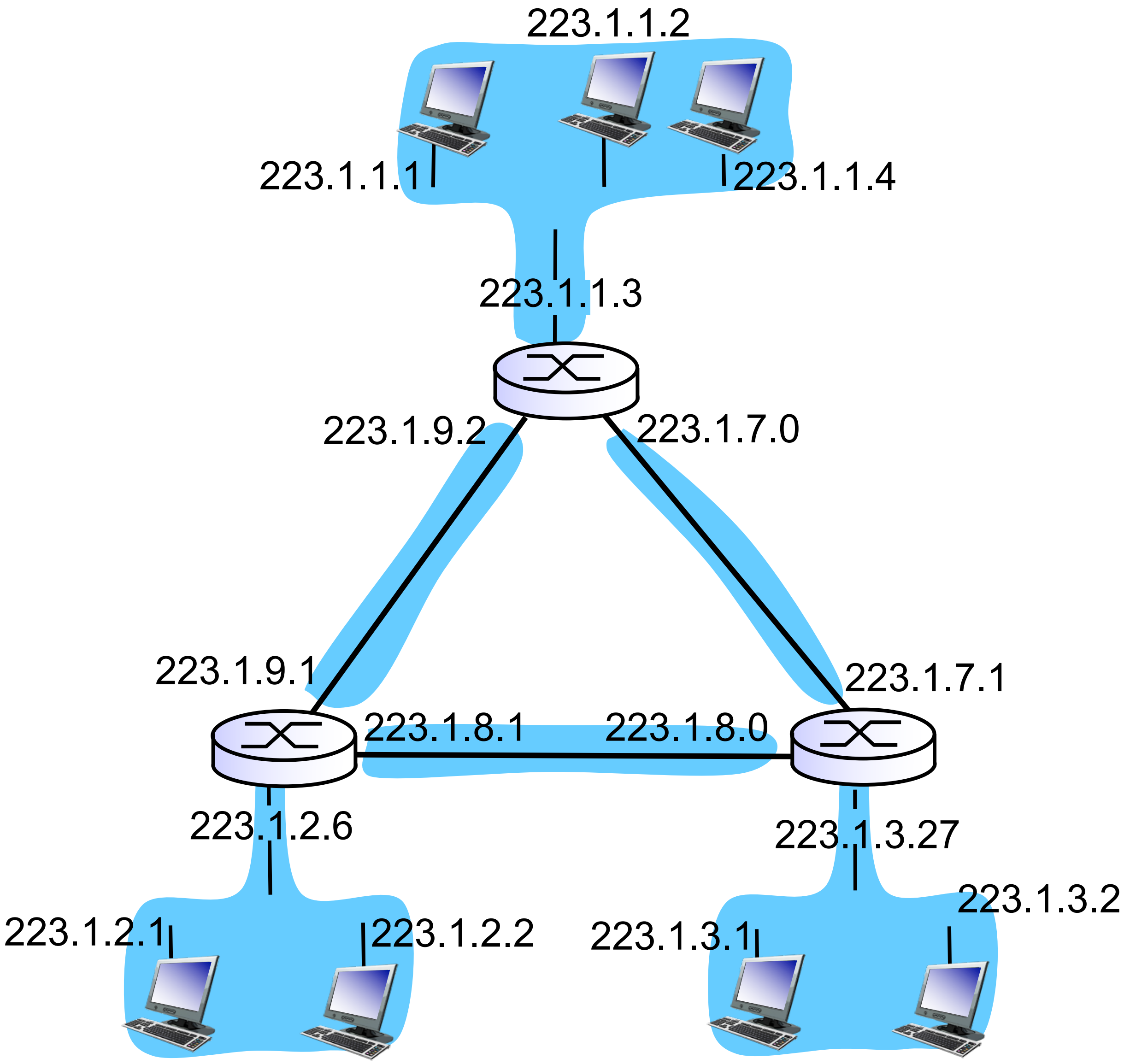
3 min: this network comprises how many subnets? Why?



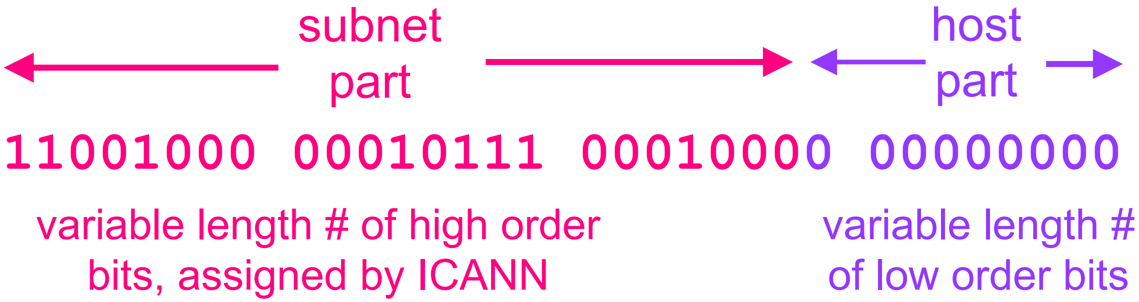
wired Ethernet interfaces connected by Ethernet switches

wireless WiFi interfaces connected by WiFi base station

How many subnets? What are address blocks?



Subnet mask example



subnet part

host part

11001000 00010111 00010000 00000000

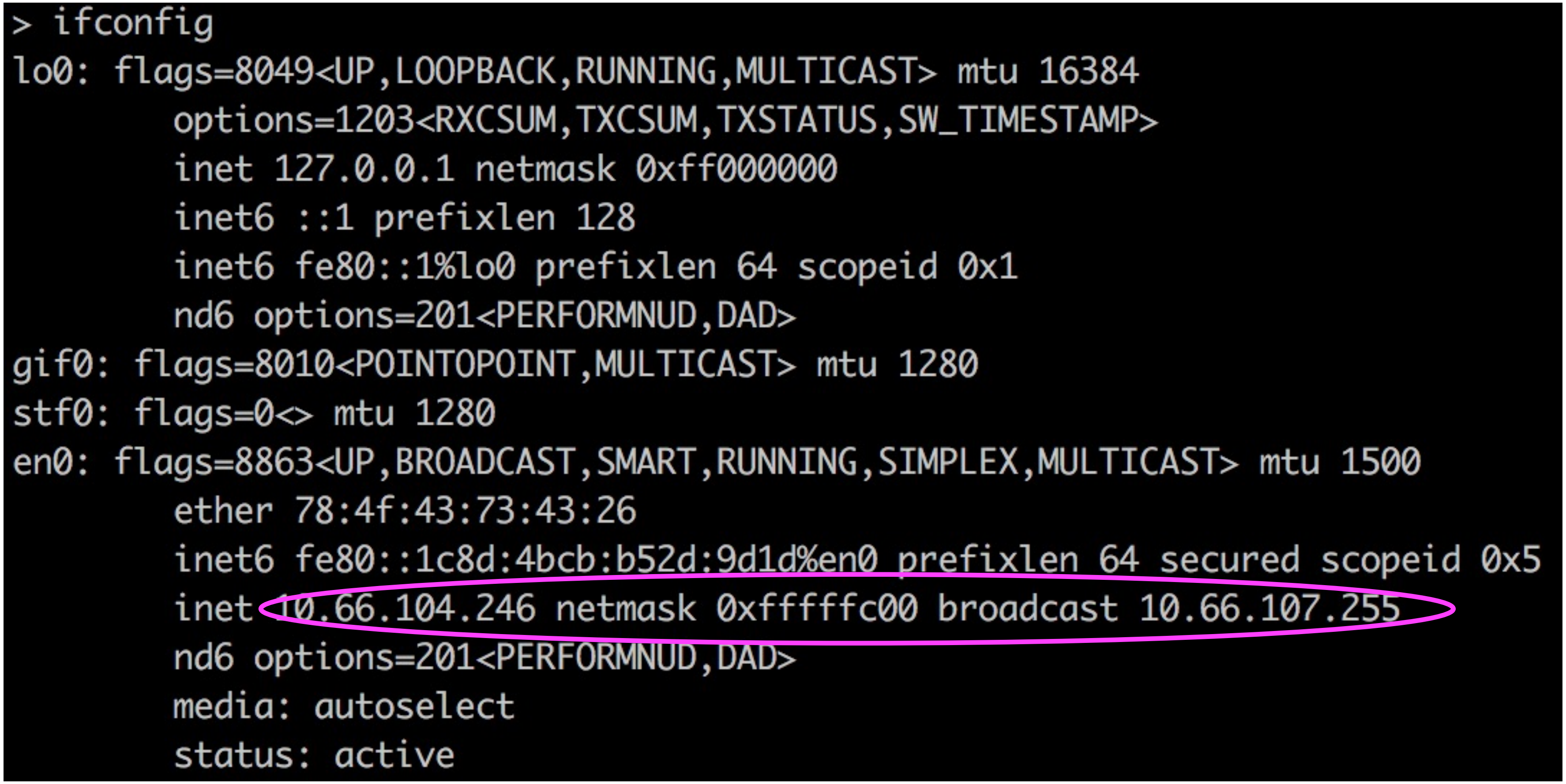
variable length # of high order bits, assigned by ICANN

variable length # of low order bits

Subnet mask

- zeroes out host part
- e.g., 200.23.16.0/23
  - 11111111 11111111 11111110 00000000
- take logical “and” of subnet mask with address to get subnet part
  - 1 AND 1 → 1
  - 1 AND 0 → 0
  - 0 AND 1 → 0
  - 0 AND 0 → 0

Ifconfig example



```
> ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        options=1203<RXCSUM,TXCSUM,TXSTATUS,SW_TIMESTAMP>
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
        nd6 options=201<PERFORMNUD,DAD>
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether 78:4f:43:73:43:26
        inet6 fe80::1c8d:4bcb:b52d:9d1d%en0 prefixlen 64 secured scopeid 0x5
        inet 10.66.104.246 netmask 0xfffffc00 broadcast 10.66.107.255
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
```

Hex is [0:15] where A=10, B=11, C=12, D=13, E=14, F=15

```
1111  1111  1111  1111  1111  1100 0000 0000
 f     f     f     f     f     c    0    0
```

Q: Why is broadcast addr 10.66.107.255?

Because .0 and .255 not assigned

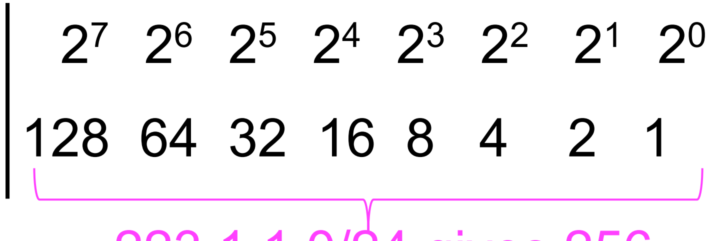
Subnet masks and address blocks

Suppose

- we must have 223.1.1 as network prefix
- we need block of 90 addresses

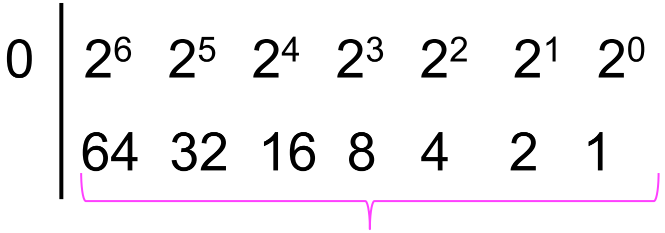
What should subnet mask be?

- how many bits for 90 addresses?



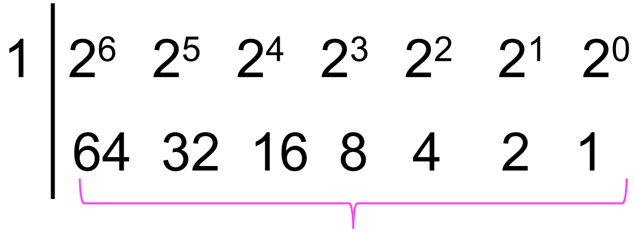
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

223.1.1.0/24 gives 256 addresses [0-255]



0	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	64	32	16	8	4	2	1

223.1.1.0/25 gives 128 addresses [0-127]



1	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	64	32	16	8	4	2	1

223.1.1.128/25 gives a different set of 128 addresses [128-255]

IP addresses are hierarchical

Pros

- scalable: routers don't need to look at host part
- all pkts on same network forwarded in same direction
  - only when pkt reaches network does host matter

Cons

- every IP addr belongs to specific network
- what if host moves networks and wants to keep same addr?
  - mobile IP
  - contrast with fixed Ethernet link layer addr

Special addresses

Private subnet (used in NAT), do not appear on Internet

- 172.16-31.*.*
- 10.*.*.*
- 192.168.*.*

Loopback address:

- 127.*.*.*

Addresses you can’t assign to devices

- *.*.*.255: broadcast addr
- *.*.*.0: used for subnet name

Broadcast address

- 255.255.255.255: broadcast to all hosts on network indicated
  - if no mask: local network
  - if mask: broadcast on that network

Address when device booting up

- 0.0.0.0

Addressing

USAGE IN ROUTING

Routers forward traffic to networks not hosts

Forwarding table

- does not contain row for every dest IP address
- instead computes routes between subnets (blocks of addresses)

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

What if address ranges don’t divide up nicely?

Longest prefix matching

– use longest address prefix that matches destination address

Destination Address Range	Link interface
`11001000 00010111 00010*** *********`	0
`11001000 00010111 00011000 *********`	1
`11001000 00010111 00011*** *********`	2
otherwise	3

Question

DA: 11001000 00010111 00010110 10100001 which interface?

DA: 11001000 00010111 00011000 10101010 which interface?

How big is a routing table for a core router?

From http://www.cidr-report.org/as2.0/

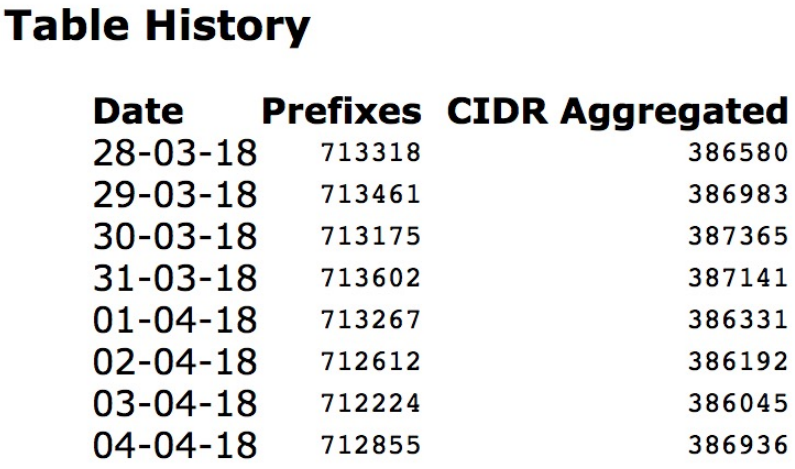
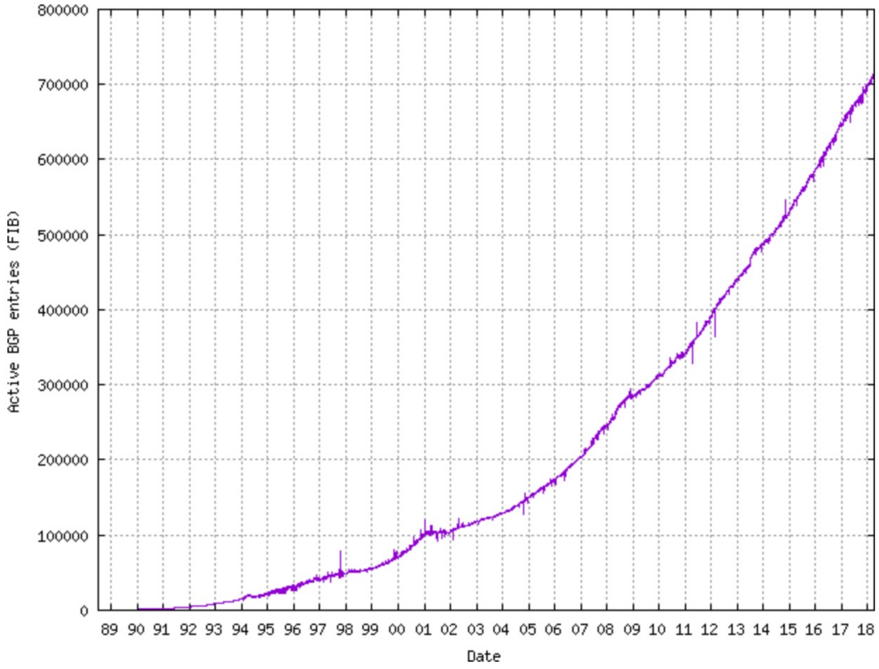
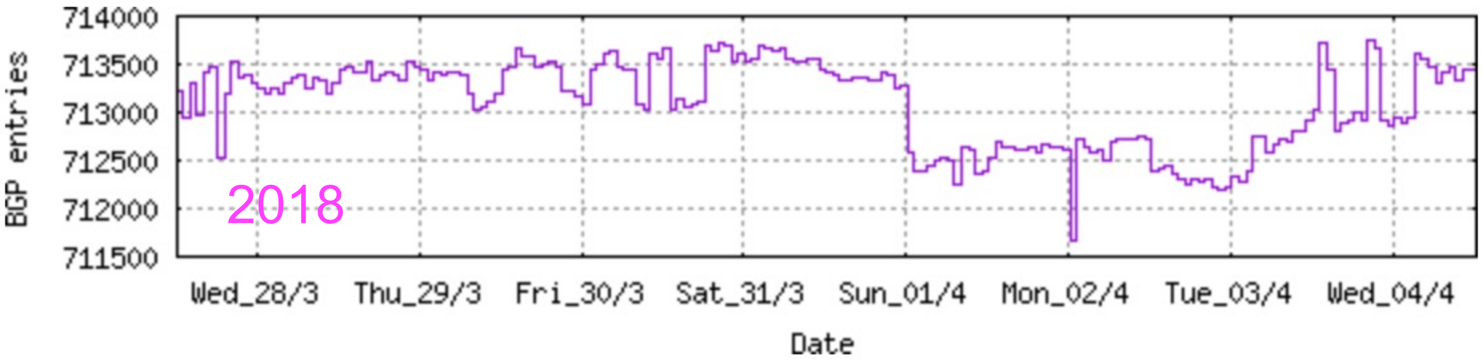


Table History

Date	Prefixes	CIDR Aggregated
28-03-18	713318	386580
29-03-18	713461	386983
30-03-18	713175	387365
31-03-18	713602	387141
01-04-18	713267	386331
02-04-18	712612	386192
03-04-18	712224	386045
04-04-18	712855	386936





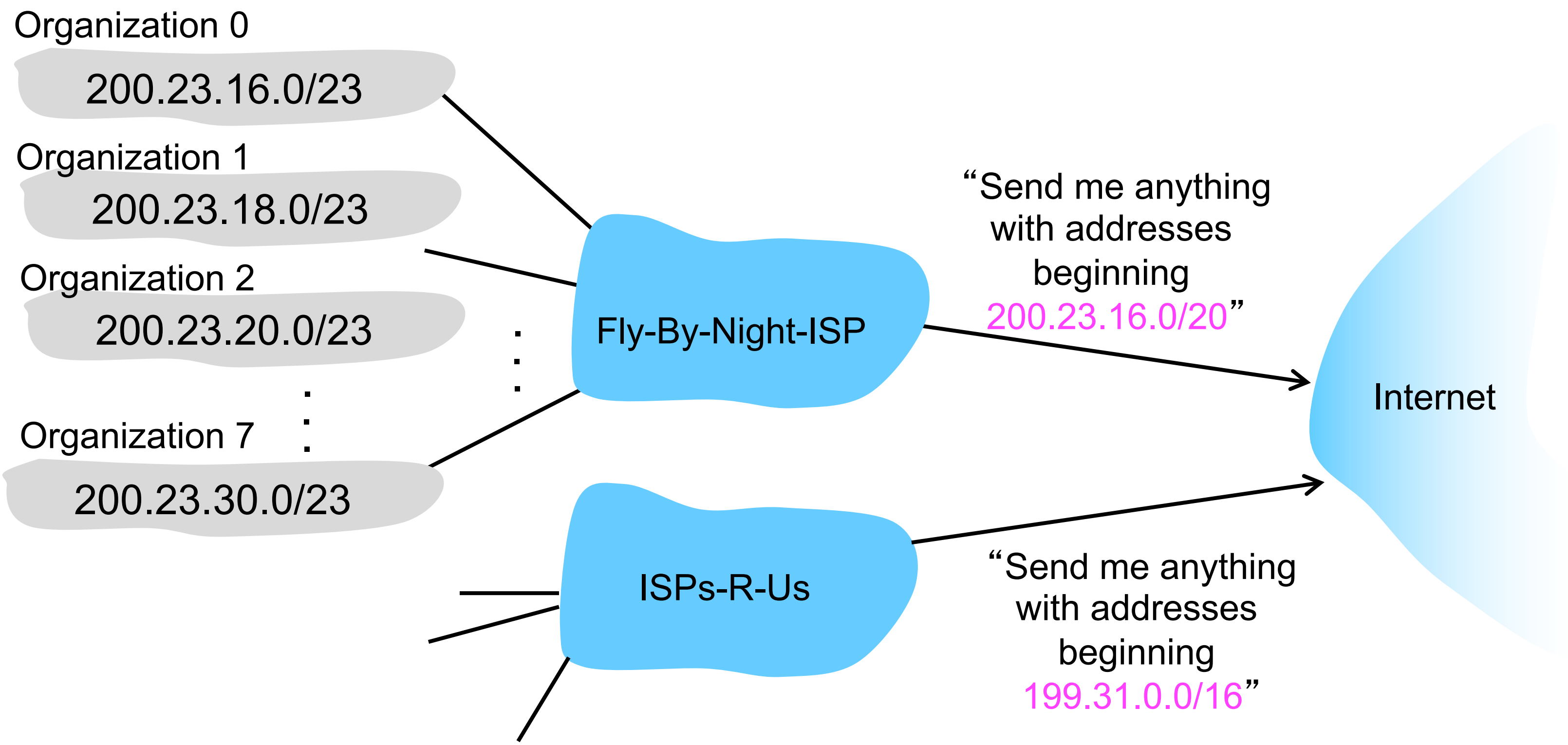
2018

Q: If a core router processes 1million pkts+ per second, how fast does it need to be able to search table?

Hierarchical addressing

Route aggregation

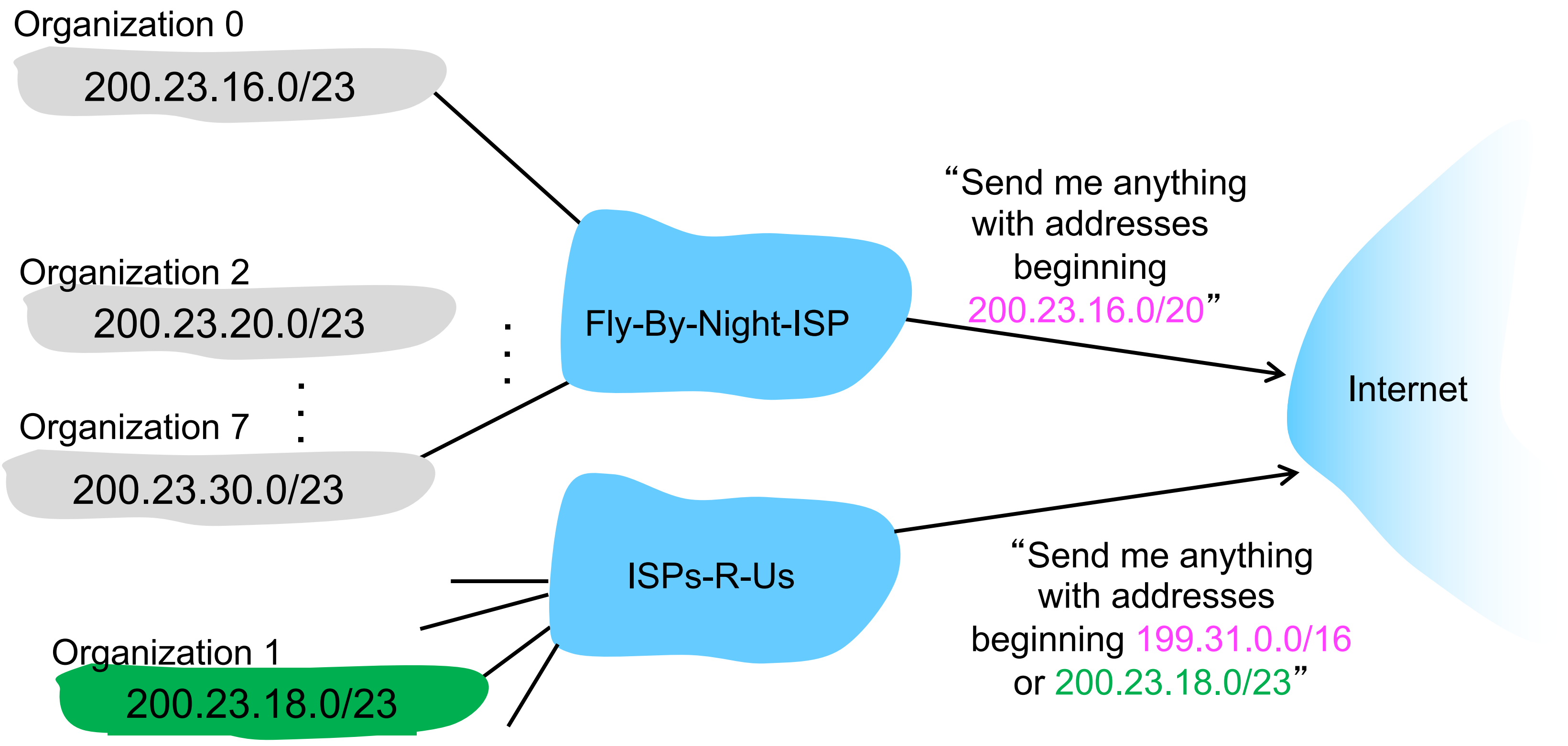
- combine multiple small prefixes into a single larger prefix
- allows efficient advertisement of routing information



Longest prefix matching

More specific routes

- ISPs-R-Us has a more specific route to Organization 1



Addressing

HOW TO GET AN IP ADDRESS?

How does ISP get block of addresses?

ICANN

- Internet Corporation for Assigned Names and Numbers
- http://www.icann.org/

ICANN functions

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes
- …

How does network get net part of IP address?

Allocated portion of its provider ISP’s address space

ISP's block	11001000 00010111 00010000	00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000	00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010	00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100	00000000	200.23.20.0/23
...	…..	….	….
Organization 7	11001000 00010111 00011110	00000000	200.23.30.0/23

How does host get an IP address?

Option 1

- hard-coded by system admin in a file on your host

Option 2:

- dynamically get address from a server
  - DHCP: Dynamic Host Configuration Protocol

We’re running out of IPv4 addresses

Why?

- inefficient use of address space
  - from pre-CIDR use of address classes (A: /8, B: /16, C: /24)
- too many networks (and devices)
  - Internet comprises 100,000+ networks
  - routing tables and route propagation protocols do not scale

Q: how many IPv4 addresses are there?

- 2^{32}

Solutions

- IPv6 addresses
- DHCP: Dynamic Host Configuration Protocol
- NAT: Network Address Translation

Network Programming

RAW SOCKETS

Raw sockets

Take bytes put into socket and push out of network interface

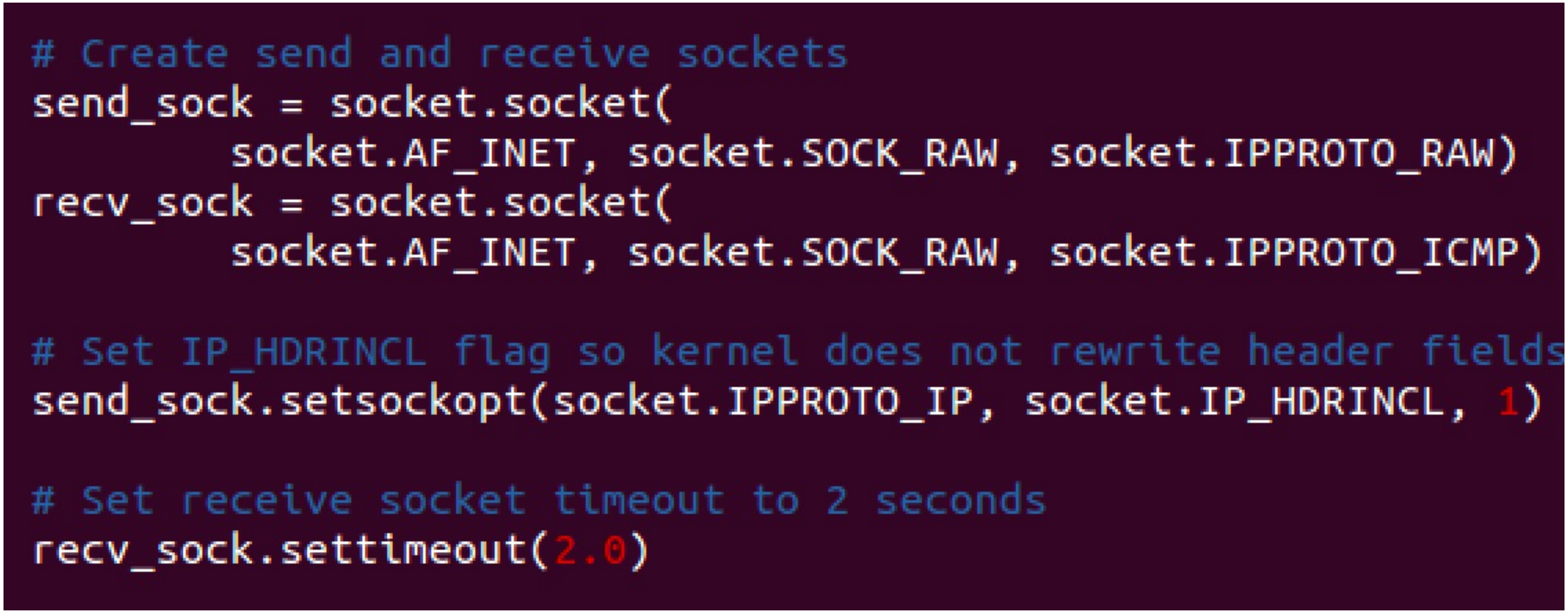
- no IP or transport layer headers added by operating system!

Q: why have raw sockets? Why are stream/datagram not enough?

Lets you create your own transport and network layer headers

- set field values as you choose
  - e.g., time-to-live fields

Homework 7/8: raw sockets



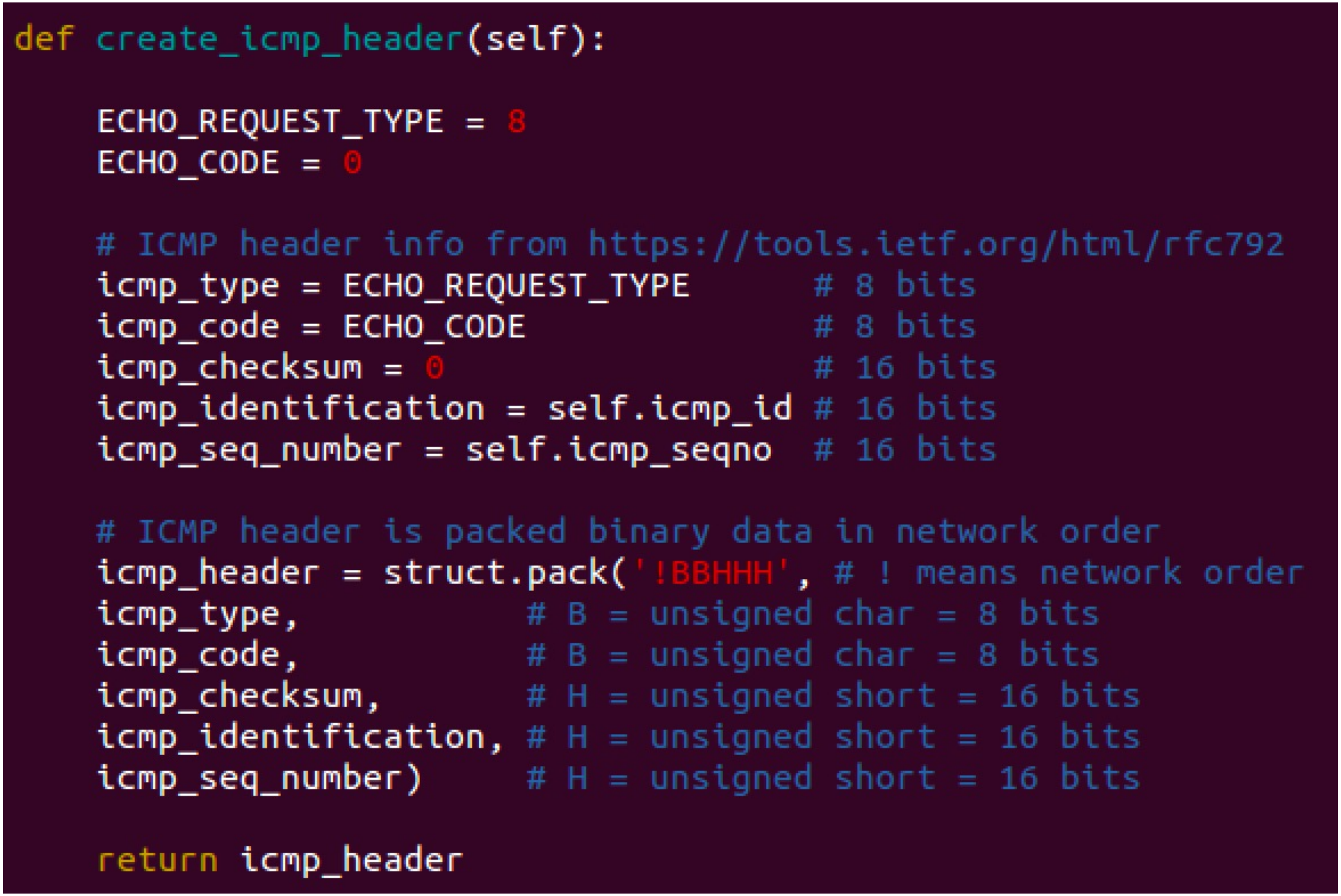
```
# Create send and receive sockets
send_sock = socket.socket(
        socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_RAW)
recv_sock = socket.socket(
        socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_ICMP)

# Set IP_HDRINCL flag so kernel does not rewrite header fields
send_sock.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)

# Set receive socket timeout to 2 seconds
recv_sock.settimeout(2.0)
```

https://docs.python.org/3/library/socket.html

Q: why set a timeout?

Byte packing and structs

How do you create a (packet) header?



```
def create_icmp_header(self):

    ECHO_REQUEST_TYPE = 8
    ECHO_CODE = 0

    # ICMP header info from https://tools.ietf.org/html/rfc792
    icmp_type = ECHO_REQUEST_TYPE      # 8 bits
    icmp_code = ECHO_CODE              # 8 bits
    icmp_checksum = 0                  # 16 bits
    icmp_identification = self.icmp_id # 16 bits
    icmp_seq_number = self.icmp_seqno  # 16 bits

    # ICMP header is packed binary data in network order
    icmp_header = struct.pack('!BBHHH', # ! means network order
    icmp_type,           # B = unsigned char = 8 bits
    icmp_code,           # B = unsigned char = 8 bits
    icmp_checksum,       # H = unsigned short = 16 bits
    icmp_identification, # H = unsigned short = 16 bits
    icmp_seq_number)     # H = unsigned short = 16 bits

    return icmp_header
```

https://docs.python.org/3/library/struct.html