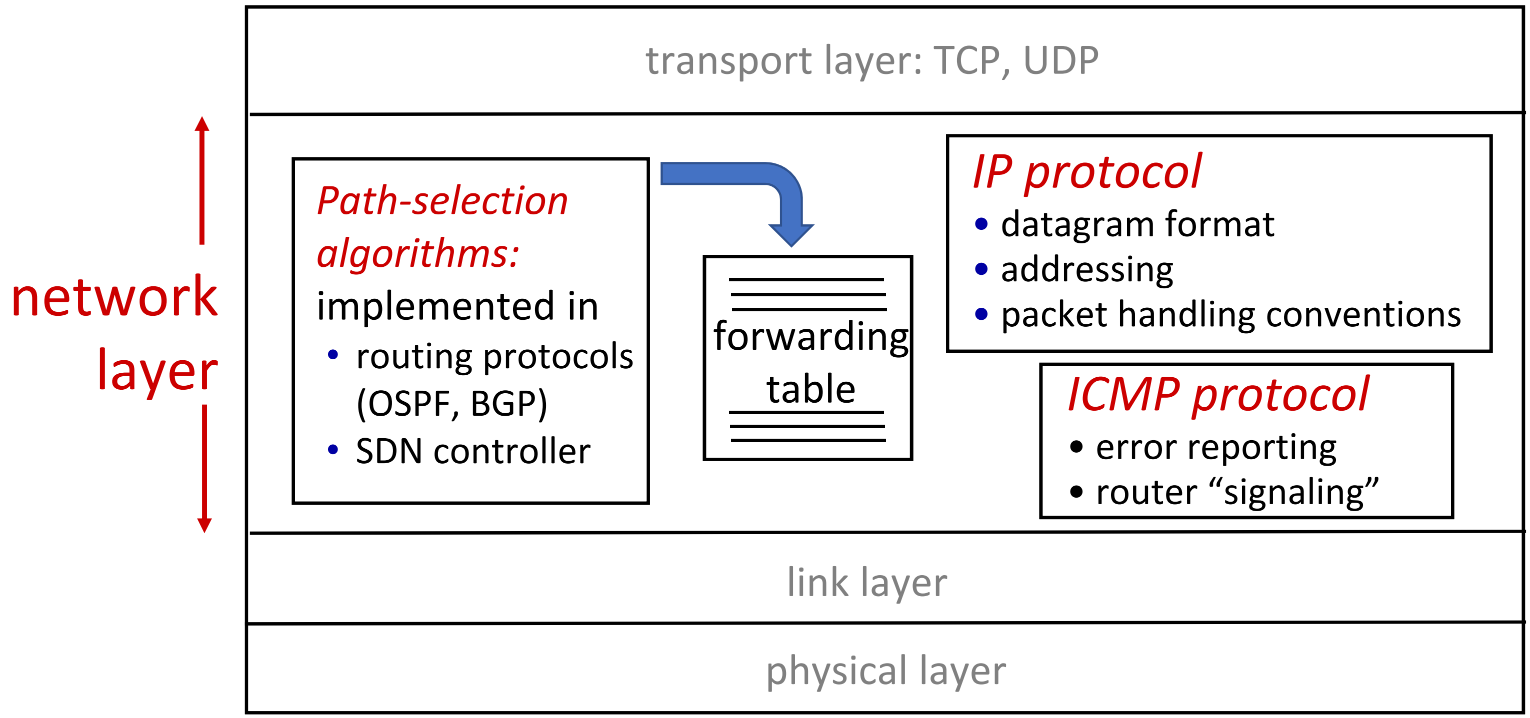
Network layer: “data plane” roadmap

- Network layer: overview
  - data plane
  - control plane
- What’s inside a router
  - input ports, switching, output ports
  - buffer management, scheduling
- IP: the Internet Protocol
  - datagram format
  - addressing
  - network address translation
  - IPv6

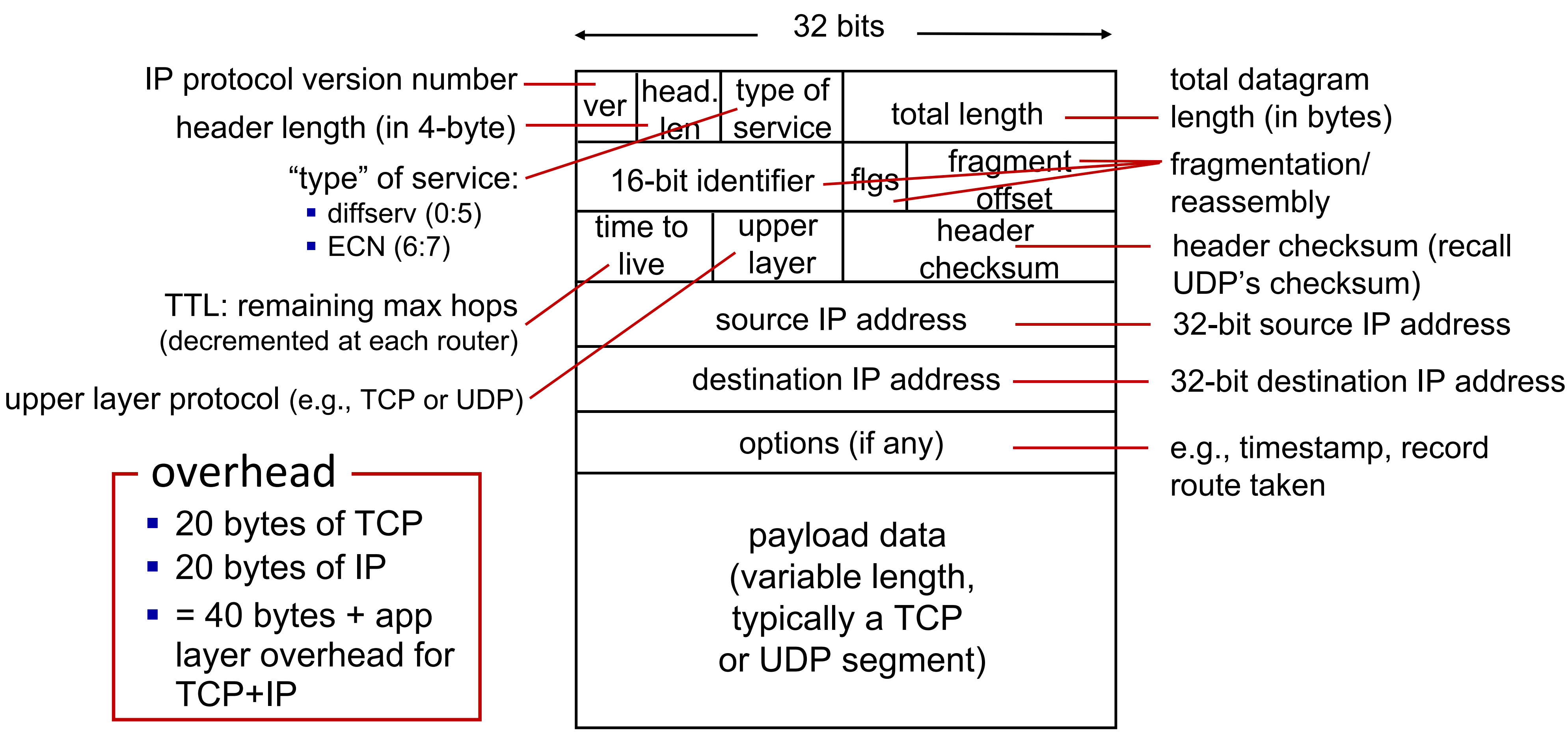
- Generalized Forwarding, SDN
  - match+action
  - OpenFlow: match+action in action
- Middleboxes

Network Layer: Internet

host, router network layer functions:



IP Datagram format: IPv4



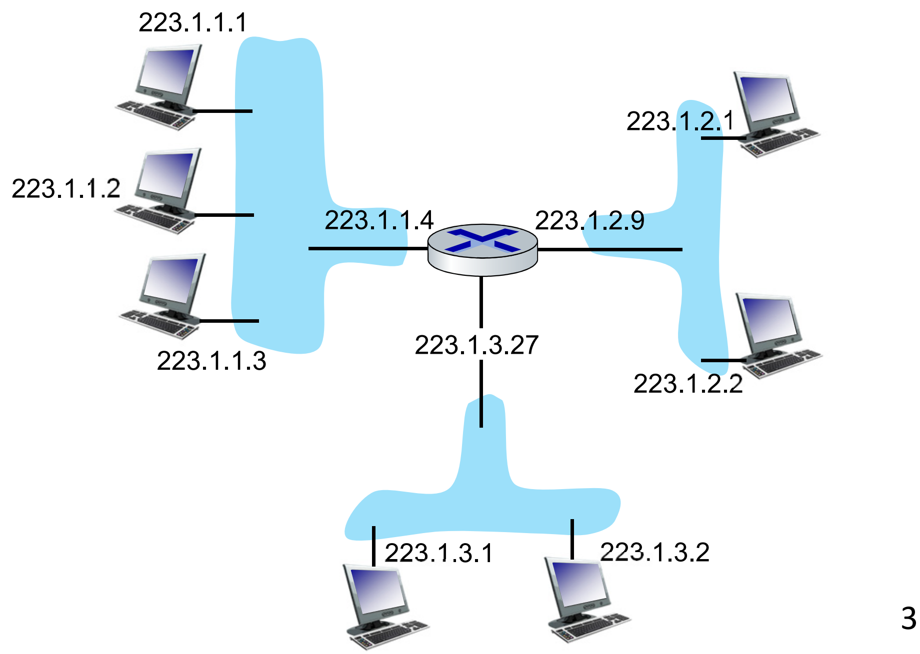
overhead
- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead for TCP+IP

IP addressing: introduction of IPv4 address

- IP address:
  - identifier in the network layer
    - is unique with some exceptions
  - IP address is 32-bit (for IPv4)
    - dotted-decimal notation
  - associated with each host or router *interface*
- interface: connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)

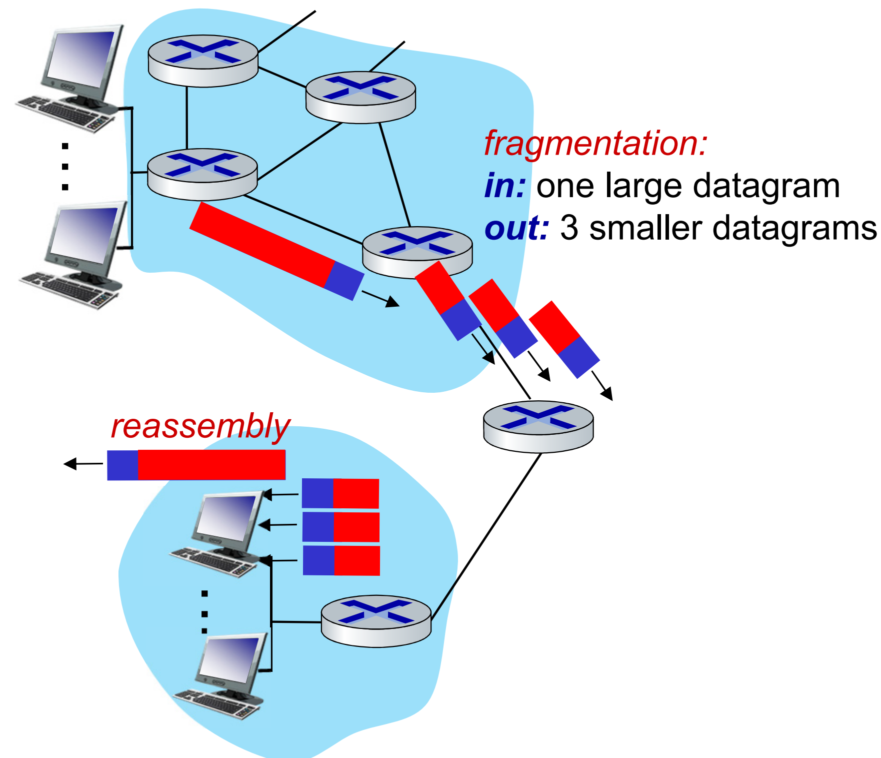
dotted-decimal IP address notation:

$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$



IP fragmentation/reassembly

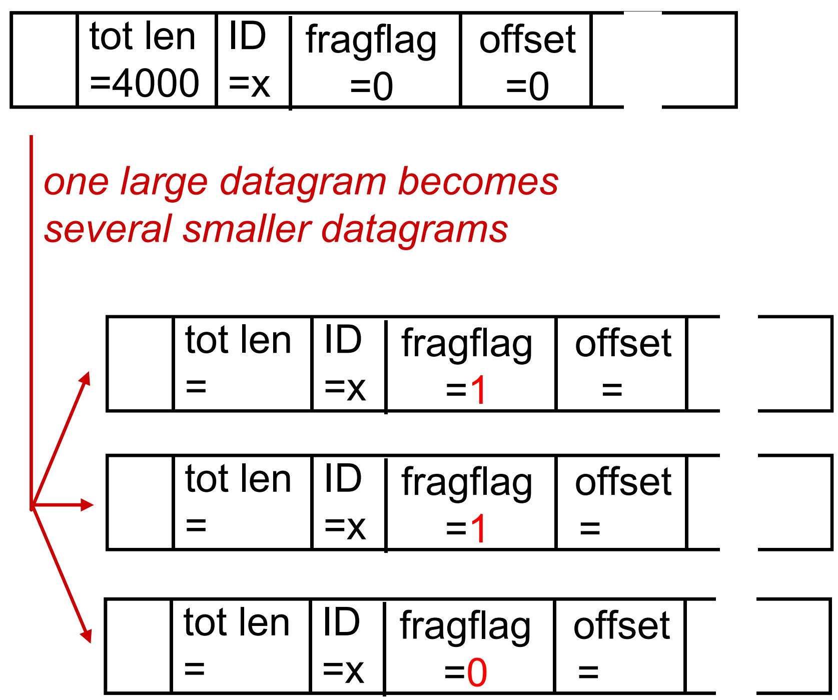
- link-layer frame have payload length limit, called MTU (max. transmission unit)
  - different link types have different MTUs
    - Ethernet: 1500, WiFi: 2304 (bytes)
- oversized IP datagram is divided ("fragmented") by router
  - one datagram becomes several smaller datagrams (fragments)
  - "reassembled" only at *destination*
  - IP header bits used to identify and order these fragments



IP fragmentation/reassembly

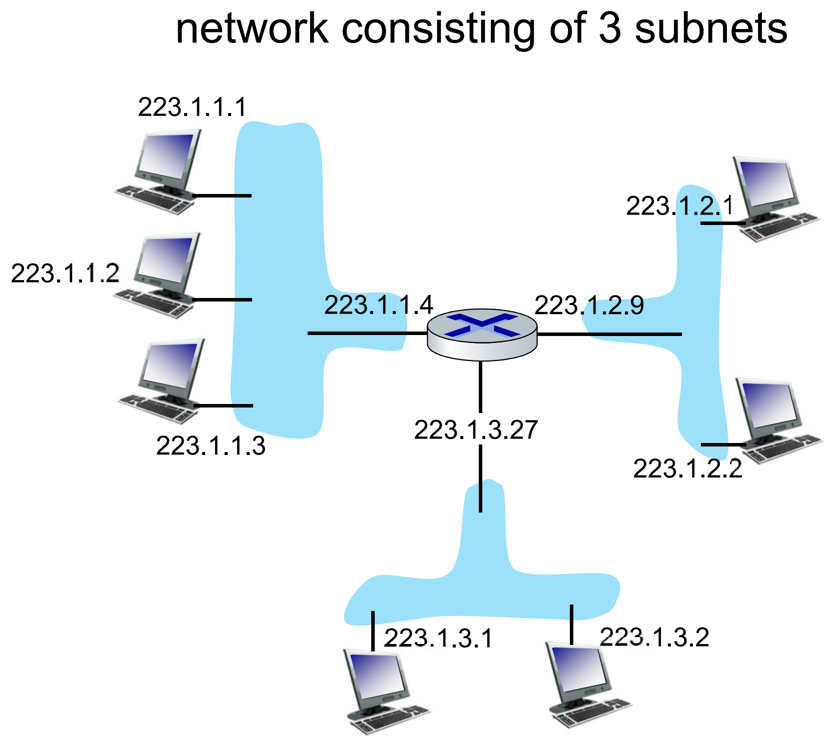
example:

- Given a 4000-byte datagram before being fragmented
  - 20-byte IP header
  - 3980-byte data
- Given MTU = 1500 bytes
  - each fragment contains
    - 20-byte IP header
    - at most 1480-byte data
- Q: How many fragments needed?
  - $\left\lceil \frac{3980}{1480} \right\rceil = 3$
- Q: offset of each fragment?
  - For 2st fragment: $\frac{1480}{8} = 185$



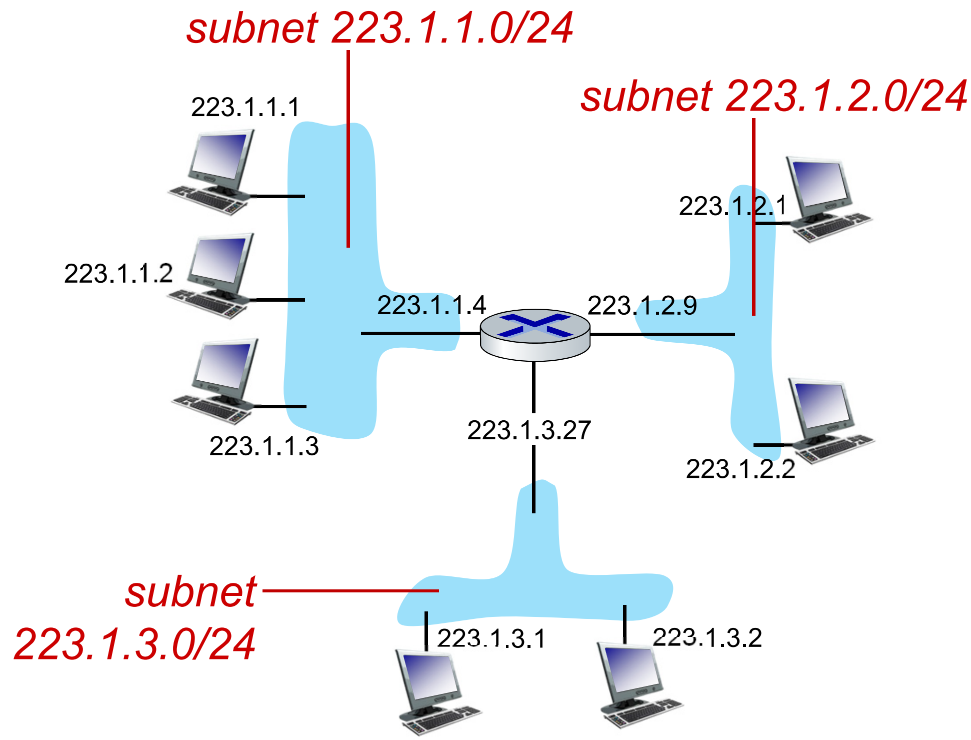
Subnet

- *What's a subnet ?*
  - device interfaces that can reach each other without passing through an intervening router
- *Recipe for defining subnets:*
  - detach each interface from its host or router, creating "islands" of isolated networks
  - each isolated network is called a subnet



Subnet

- structure of IP addresses within a subnet:
  - subnet part: devices in same subnet have common high order bits
  - host part: remaining low order bits
  - dotted-decimal notation of subnet:




```
11011111 00000001 00000001 xxxxxxxx = 223.1.1.0/24
11011111 00000001 0000001x xxxxxxxx =
11011111 00000001 00000000 1xxxxxxx =
```

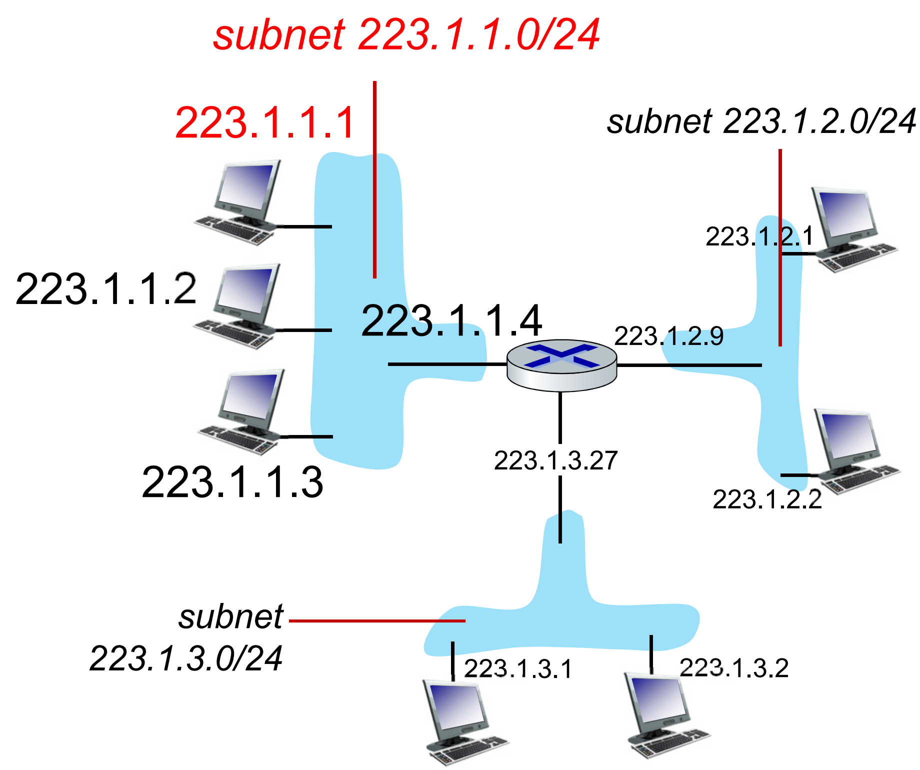
Subnet

- For the host **223.1.1.1** in the subnet 223.1.1.0/24, its setting is as follows
  - IP address: **223.1.1.1**
  - subnet mask: /24 or 255.255.255.0

    223.1.1.0/24 =

    11011111 00000001 00000001 xxxxxxxx

    11111111 11111111 11111111 00000000
  - gateway:



IP address (assignment and routing): CIDR

CIDR: Classless InterDomain Routing (pronounced “cider”)

- subnet portion of address of arbitrary length
- address format: a.b.c.d/x, where x is # of bits in subnet portion of address

subnet part | host part

11001000 00010111 00010000 00000000

200.23.16.0/23

IP address: how does a host get an IP address?

- set up manually by system administrator in config file
  - e.g. /etc/rc.config in UNIX
- DHCP: **D**ynamic **H**ost **C**onfiguration **P**rotocol
  - dynamically get address from as server
    - "plug-and-play"

DHCP: Dynamic Host Configuration Protocol

goal: host *dynamically* obtains IP address from network server when it "joins" network

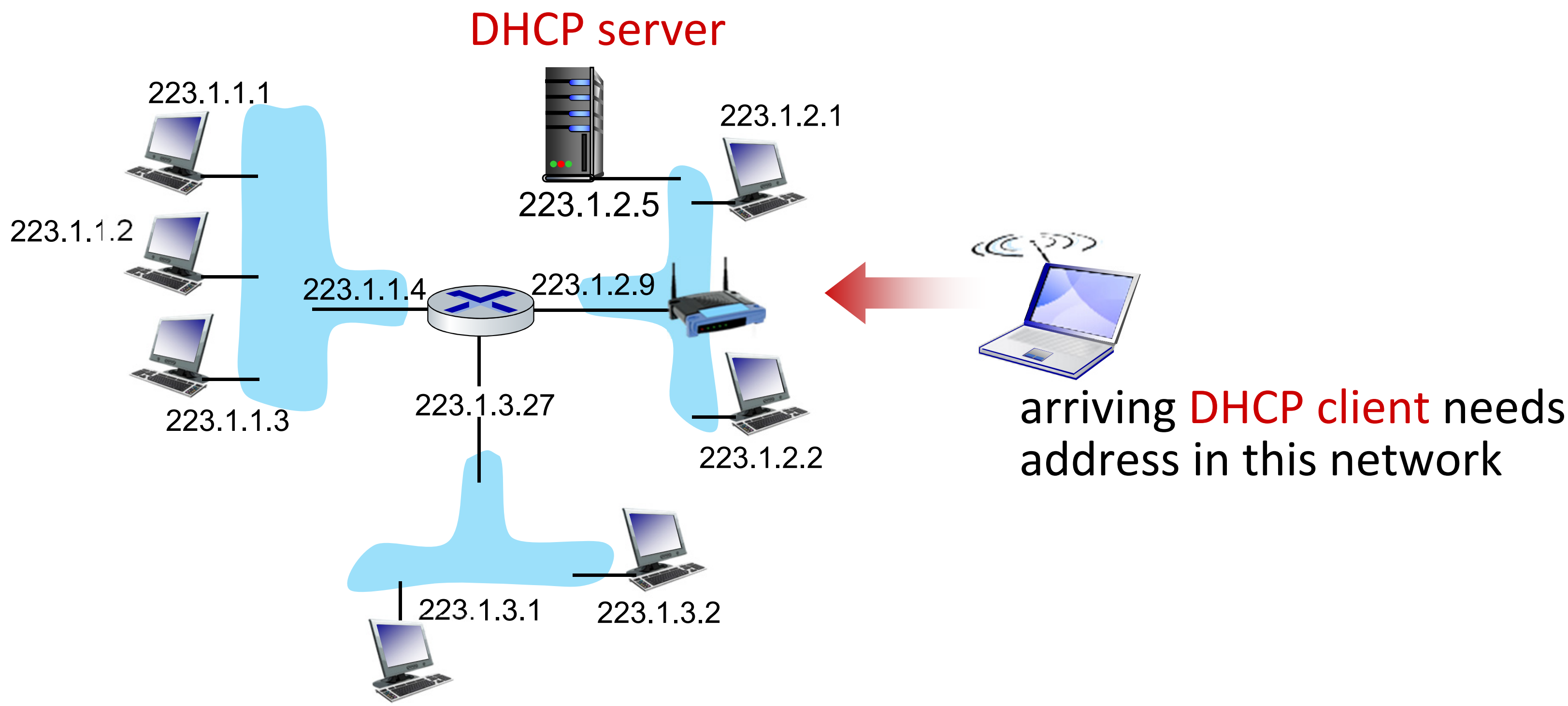
- lease is valid only for a period of time
- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/on)
- support for mobile users who join/leave network

DHCP overview:

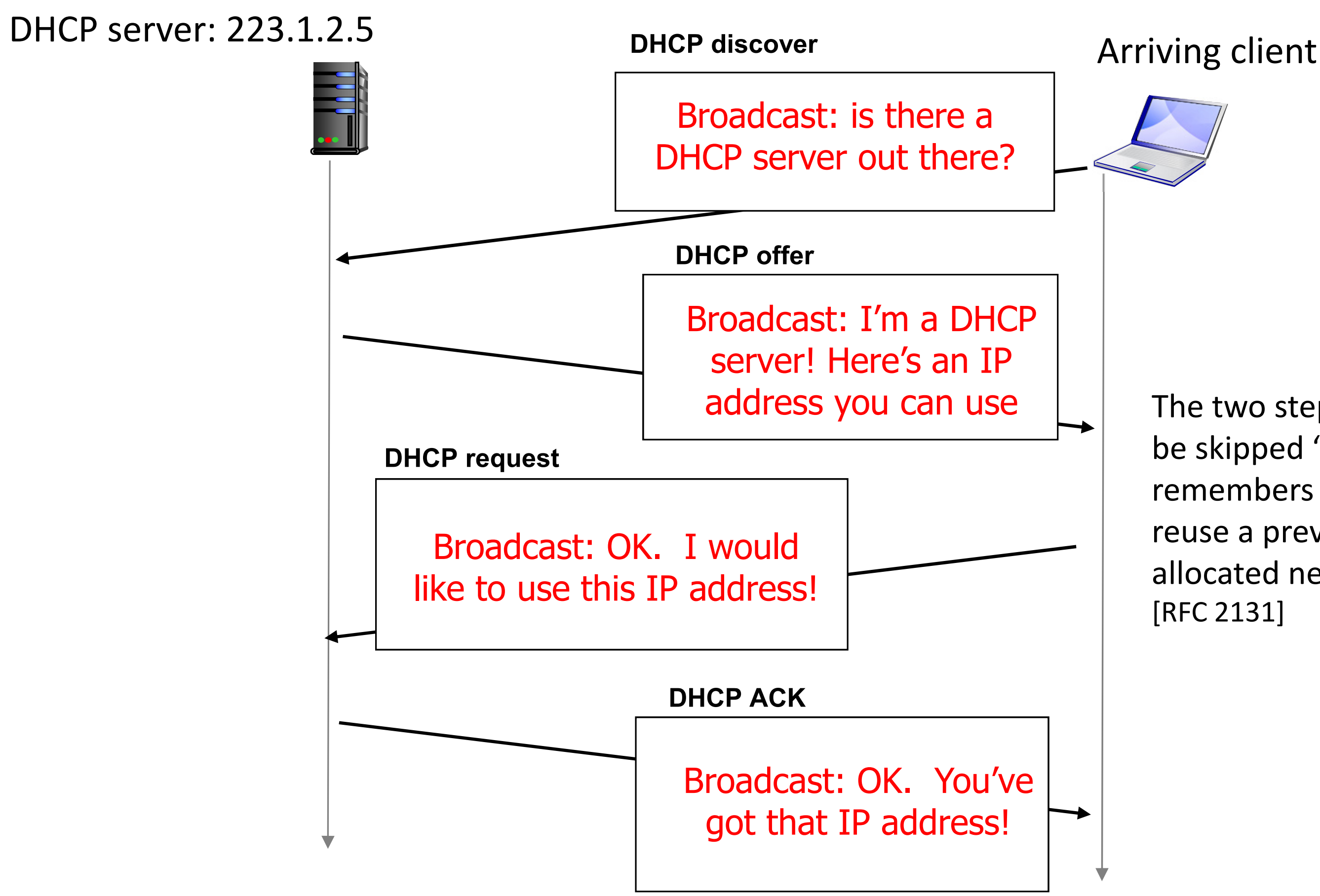
- host broadcasts DHCP discover msg [optional]
- DHCP server responds with DHCP offer msg [optional]
- host requests IP address: DHCP request msg
- DHCP server sends address: DHCP ack msg

DHCP client-server scenario

Typically, DHCP server will be co-located in router, serving all subnets to which router is attached

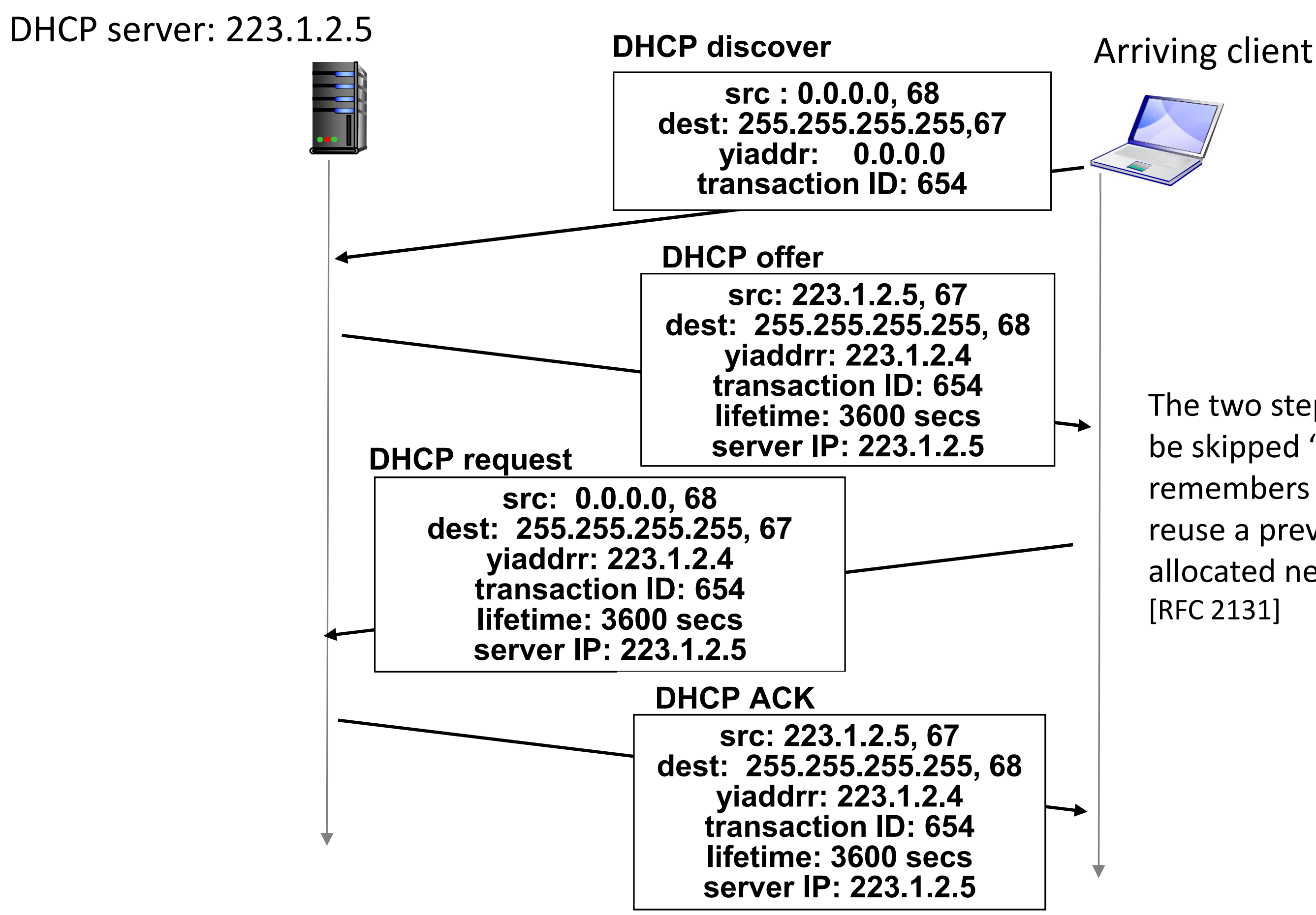


DHCP client-server scenario



The two steps above can be skipped "if a client remembers and wishes to reuse a previously allocated network address" [RFC 2131]

DHCP client-server scenario



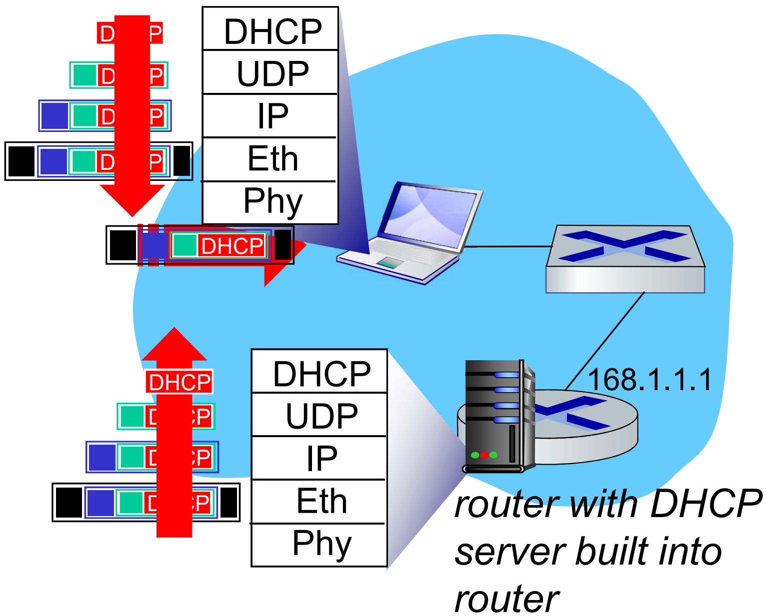
The two steps above can be skipped "if a client remembers and wishes to reuse a previously allocated network address" [RFC 2131]

DHCP: more than IP addresses

DHCP can (typically) return more than just allocated IP address on subnet:

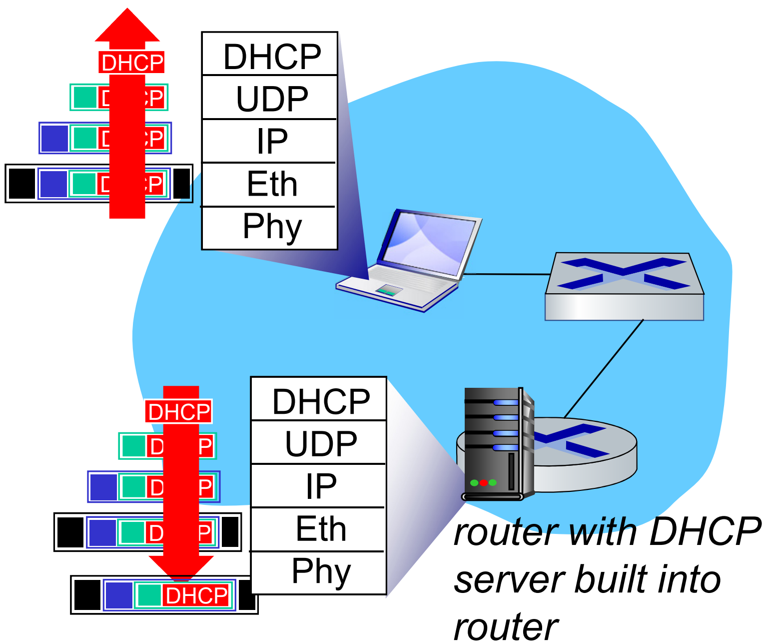
- address of first-hop router (gateway) for client
- network mask (indicating network versus host portion of address)
- name and IP address of DNS sever

DHCP: example



- Connecting laptop will use DHCP to get IP address, address of first-hop router, address of DNS server.
- DHCP REQUEST message encapsulated in UDP, encapsulated in IP, encapsulated in Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demux'ed to IP demux'ed, UDP demux'ed to DHCP

DHCP: example



- DHCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulated DHCP server reply forwarded to client, demuxing up to DHCP at client
- client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router

Hierarchical addressing: allocation and routing

Suppose ISP has the following IP addresses

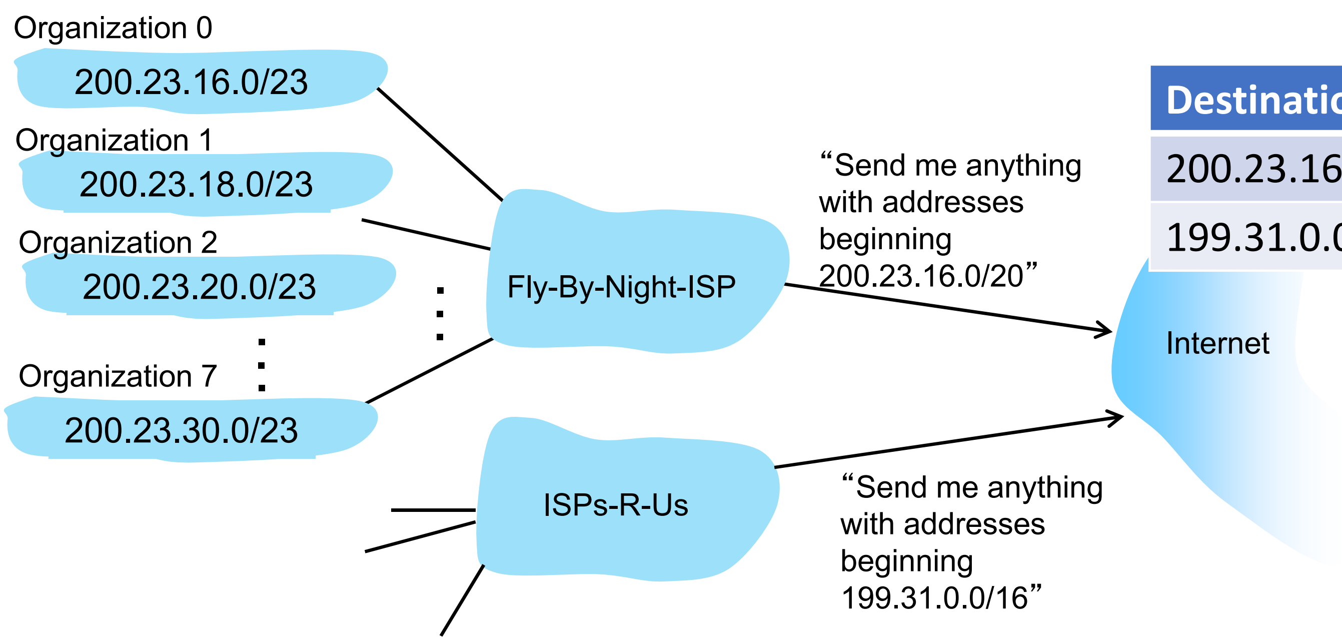
ISP's block	<u>11001000 00010111 0001</u>0000 00000000	200.23.16.0/20

ISP can then allocate out its address space in 8 blocks:

Organization 0	<u>11001000 00010111 0001000</u>0 00000000	200.23.16.0/23
Organization 1	<u>11001000 00010111 0001001</u>0 00000000	200.23.18.0/23
Organization 2	<u>11001000 00010111 0001010</u>0 00000000	200.23.20.0/23
...	....	....
Organization 7	<u>11001000 00010111 0001111</u>0 00000000	200.23.30.0/23

Hierarchical addressing: route aggregation

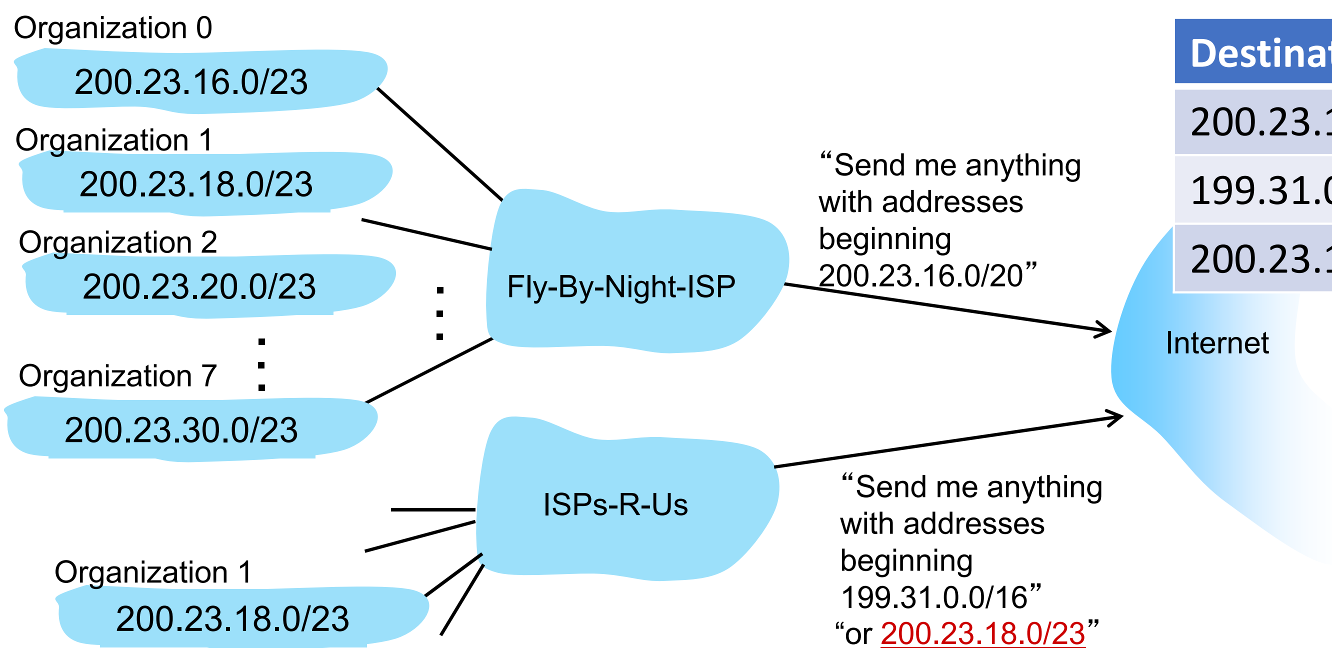
hierarchical addressing allows efficient advertisement of routing information:



Destination IP address	port
200.23.16.0/20	↖
199.31.0.0/16	↙

Hierarchical addressing: more specific routes

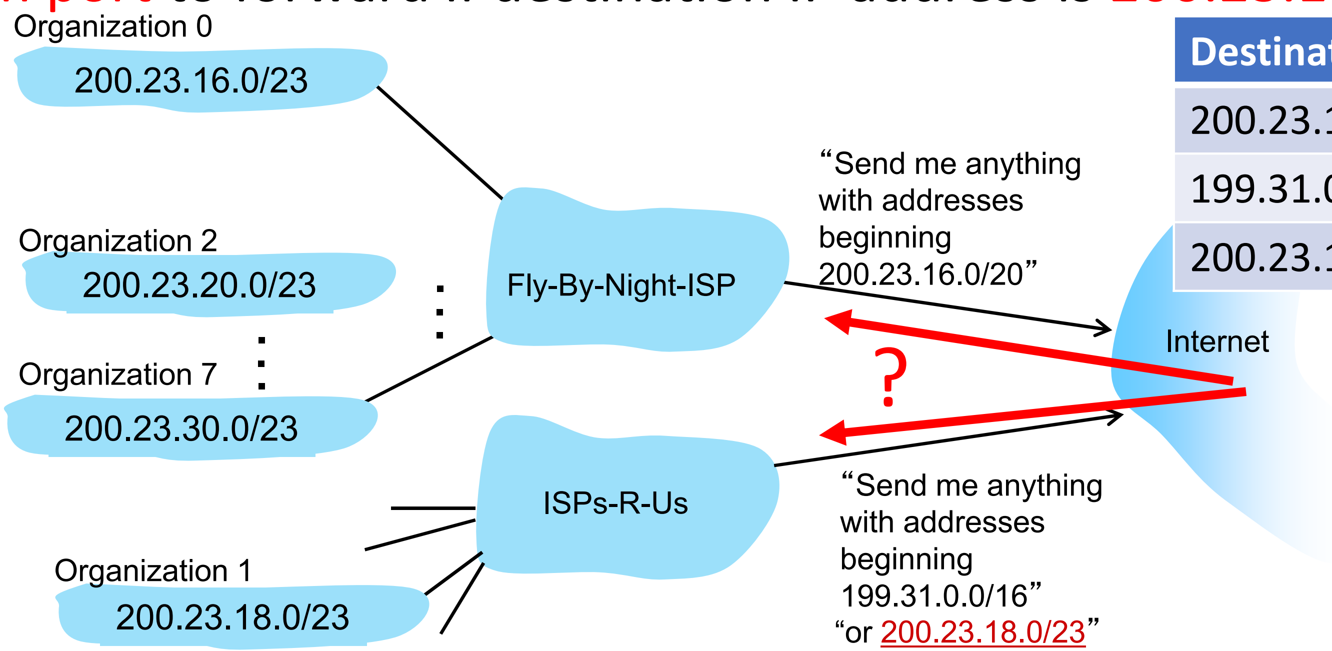
- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



Destination IP address	port
200.23.16.0/20	↖
199.31.0.0/16	↙
200.23.18.0/23	↙

Hierarchical addressing: more specific routes

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1
- which port to forward if destination IP address is 200.23.18.1?



Destination IP address	port
200.23.16.0/20	↖
199.31.0.0/16	↙
200.23.18.0/23	↙

IP addressing: last words ...

Q: how does an ISP get block of addresses?

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

- allocates IP addresses, through 5 regional registries (RRs) (who may then allocate to local registries)
- manages DNS root zone, including delegation of individual top-level domain (.com, .edu , …) management

Q: are there enough 32-bit IP addresses?

- ICANN allocated last chunk of IPv4 addresses to RRs in 2011
- NAT (next) helps IPv4 address space exhaustion
- IPv6 has 128-bit address space

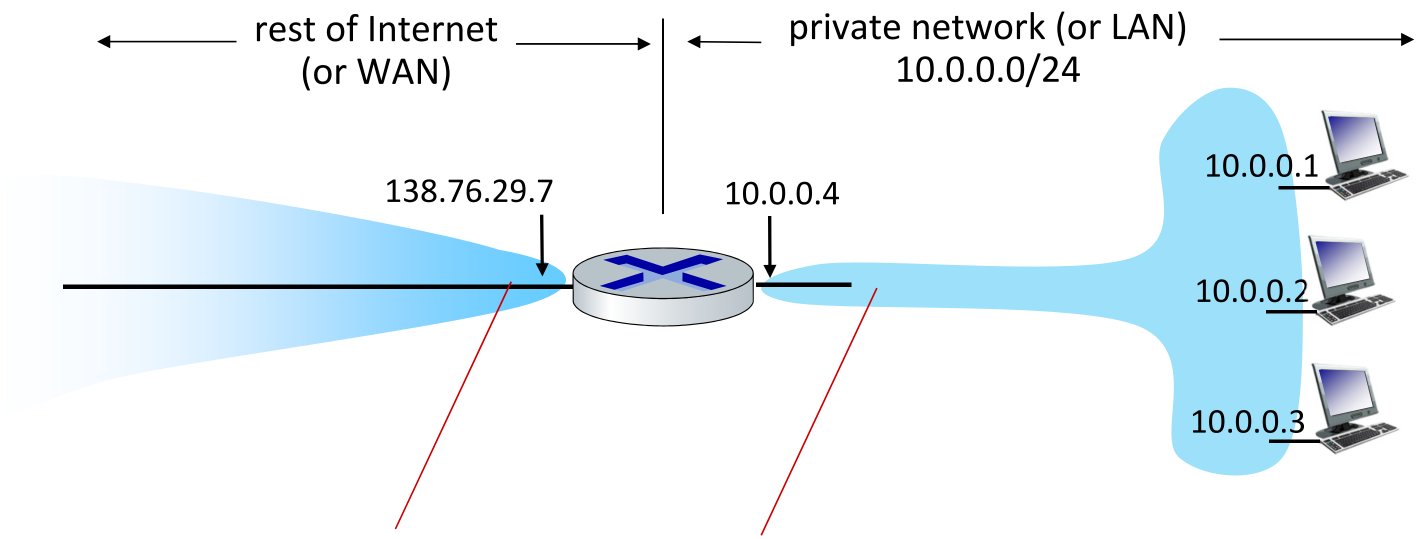
Network layer: “data plane” roadmap

- Network layer: overview
  - data plane
  - control plane
- What’s inside a router
  - input ports, switching, output ports
  - buffer management, scheduling
- IP: the Internet Protocol
  - datagram format
  - addressing
  - network address translation
  - IPv6

- Generalized Forwarding, SDN
  - match+action
  - OpenFlow: match+action in action
- Middleboxes

NAT: network address translation

NAT: all devices in private network share just one IPv4 address as far as outside world is concerned

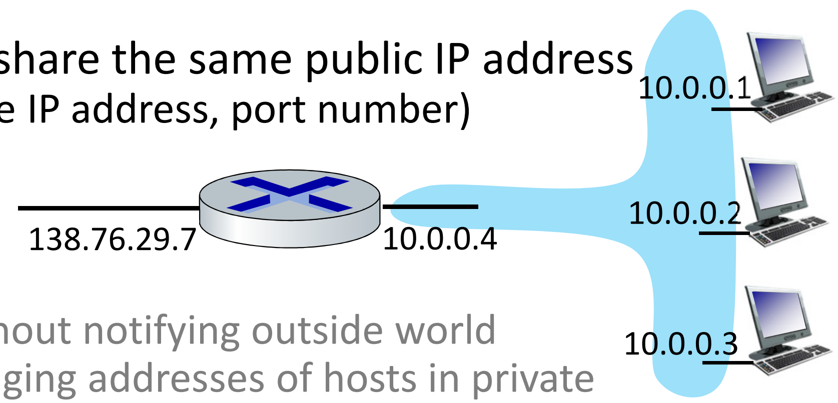


all datagrams *leaving* private network have *same* source NAT IP address: 138.76.29.7, but *different* source port numbers

datagrams with source or destination in this network have 10.0.0.0/24 address for source, destination (as usual)

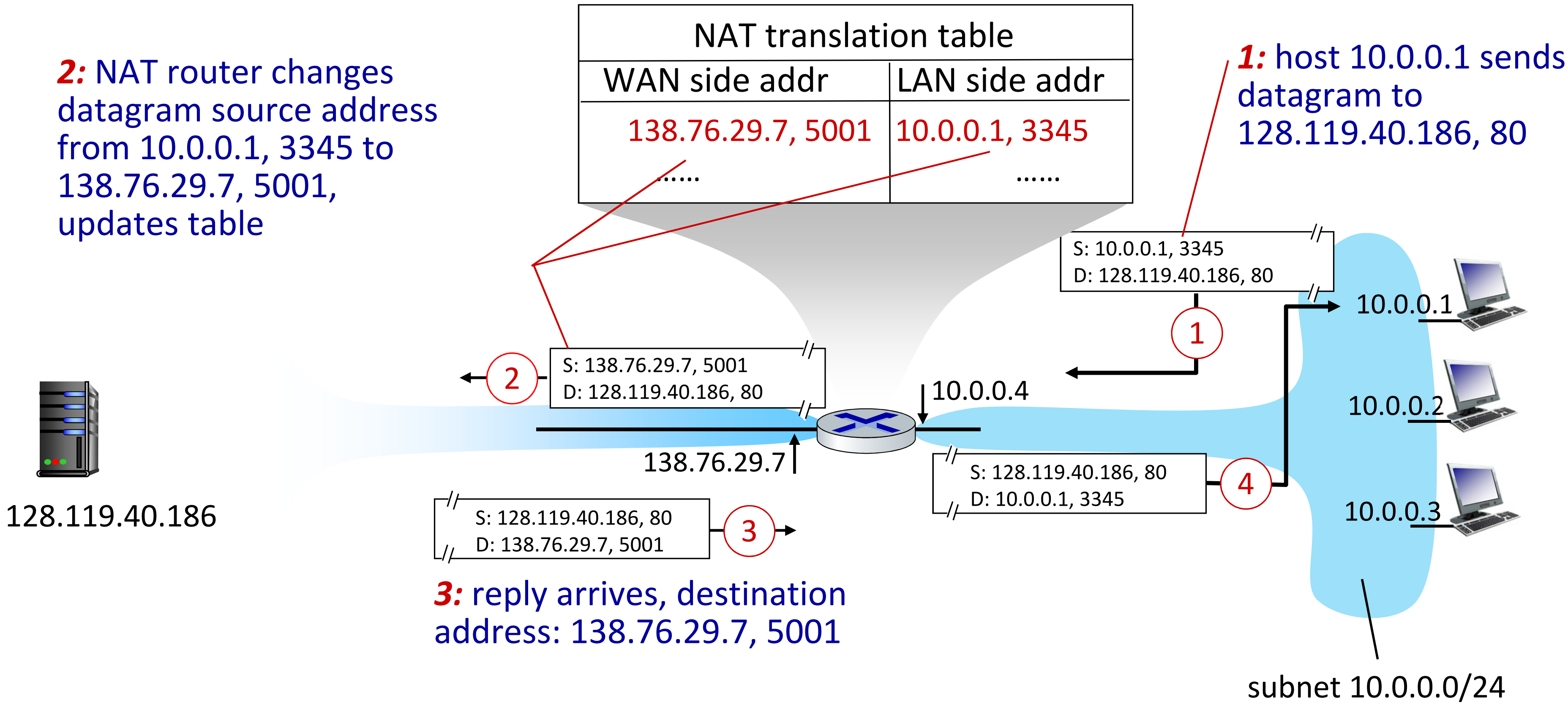
NAT: network address translation

- private IP address
  - unlike a public IP address, a private IP address is **not unique** in the Internet
    - hosts in different private networks can reuse the same “private” IP address
    - hosts in the same private network can’t reuse the same “private” IP address
  - “private” IP address space includes
    - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
- NAT uses port number to distinguish the hosts that share the same public IP address
  - (shared public IP address, port number) ←→ (private IP address, port number)



- advantages:
  - just one IP address needed for all devices
  - can change addresses of hosts in private network without notifying outside world
  - can change the shared public IP address without changing addresses of hosts in private network
  - security: devices inside private network not directly addressable, not visible by outside world

NAT: network address translation



NAT: network address translation

implementation: NAT router must (transparently):

- outgoing datagrams: replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
  - remote clients/servers will respond using (NAT IP address, new port #) as destination address
- remember (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair
- incoming datagrams: replace (NAT IP address, new port #) in destination fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation

- NAT has been controversial:
  - routers “should” only process up to layer 3
  - address “shortage” should be solved by IPv6
  - NAT traversal: what if client wants to connect to server behind NAT?
- but NAT is here to stay:
  - extensively used in home and institutional nets, 4G/5G mobile networks

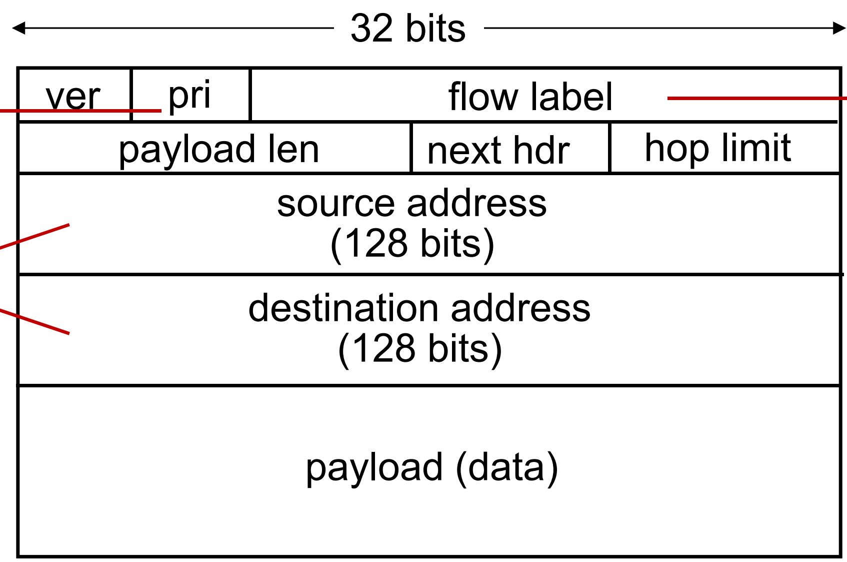
IPv6: motivation

- initial motivation:
  - run out of 32-bit IPv4 address space
- additional motivation:
  - speed up processing/forwarding
    - 40-byte fixed length header
    - …
  - enable different network-layer treatment of “flows”

IPv6 datagram format

traffic class / priority: identify priority among datagrams in flow. TOS.

128-bit IPv6 addresses



flow label: identify datagrams in same "flow" (concept of "flow" not well defined).

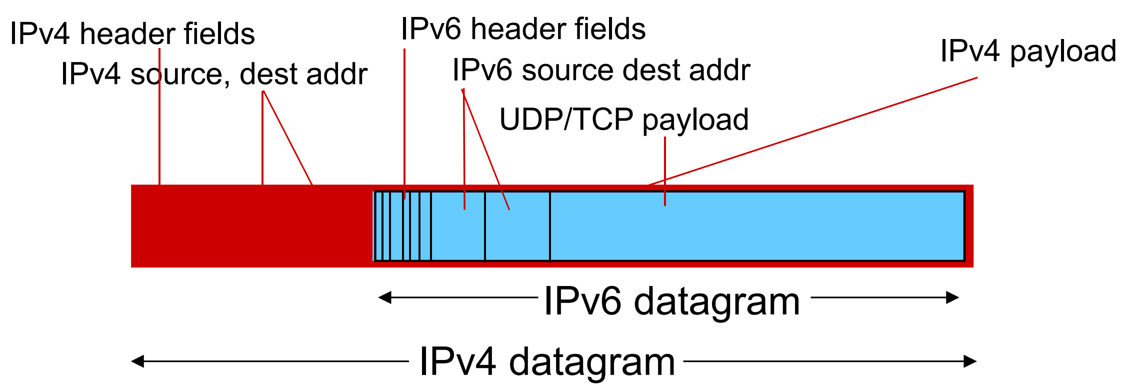
What's missing (compared with IPv4):

- no checksum (to speed processing at routers)
- no fragmentation/reassembly
- no options (available as upper-layer, next-header protocol at router)

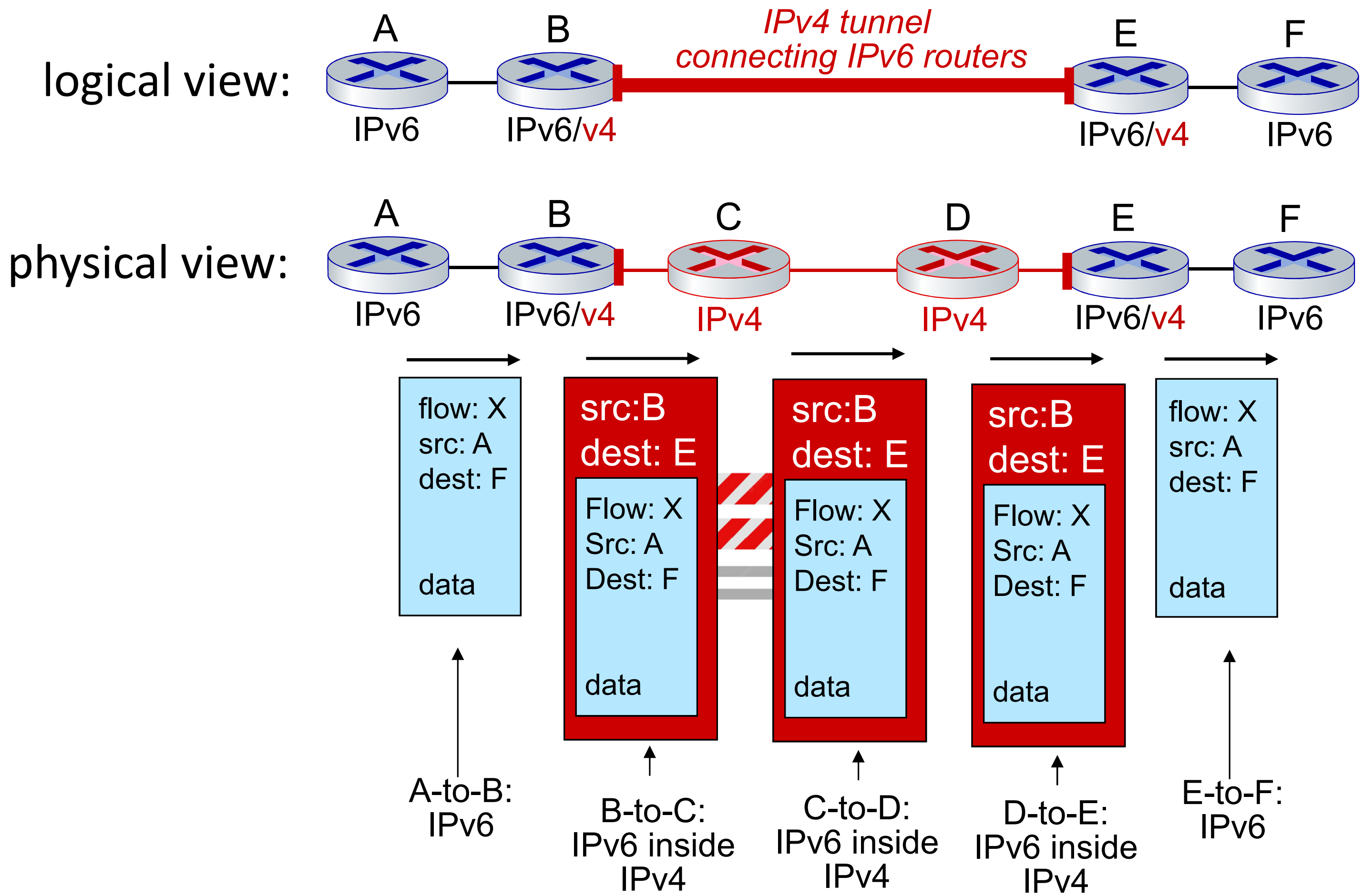
Transition from IPv4 to IPv6

- not all routers can be upgraded simultaneously
  - how will network operate with mixed IPv4 and IPv6 routers?

- tunneling: IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers ("packet within a packet")
  - tunneling used extensively in other contexts (4G/5G)



Tunneling



logical view:
A
B
IPv4 tunnel connecting IPv6 routers
E
F
IPv6
IPv6/v4
IPv6/v4
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6/v4
IPv4
IPv4
IPv6/v4
IPv6
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B: IPv6
B-to-C: IPv6 inside IPv4
C-to-D: IPv6 inside IPv4
D-to-E: IPv6 inside IPv4
E-to-F: IPv6