CCCN 312 Computer Networks

Instructor: YOUR NAME

1st Trimester 2022/23

جامعة جدة
University of Jeddah

Chapter 4
Network Layer: Data Plane

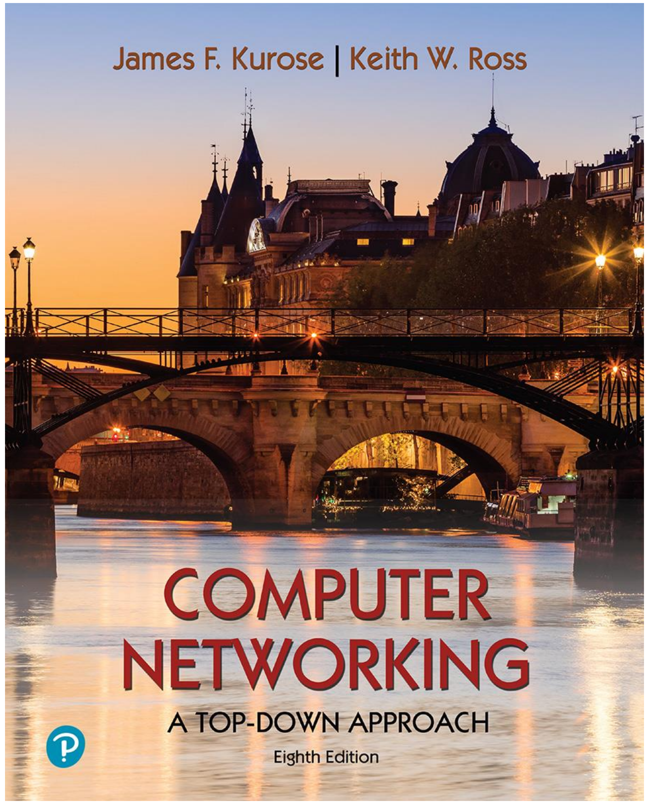
A note on the use of these PowerPoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Outline

1. Introduction ✓
2. Application layer ✓
3. Transport layer ✓
4. **Network layer: Data Plane**
5. Network layer: Control Plane
6. Link layer

Network layer: our goals

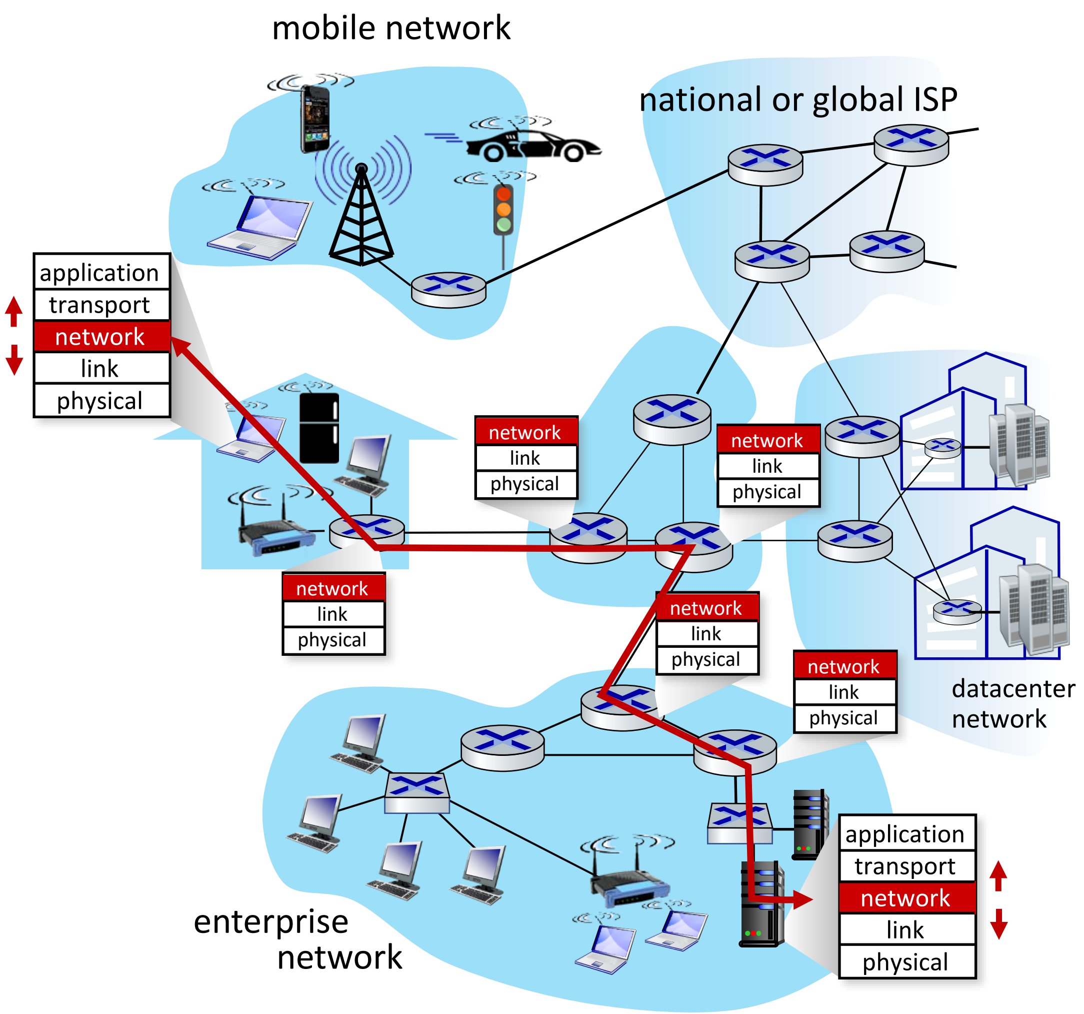
- understand principles behind network layer services, focusing on data plane:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - addressing
  - forwarding
  - Internet architecture
- instantiation, implementation in the Internet
  - IP protocol
  - NAT, DHCP

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

Network-layer services and protocols

- transport segment from sending to receiving host
  - sender: encapsulates segments into datagrams, passes to link layer
  - receiver: delivers segments to transport layer protocol
- network layer protocols in *every Internet device*: hosts, routers
- routers:
  - examines header fields in all IP datagrams passing through it
  - moves datagrams from input ports to output ports to transfer datagrams along end-end path



Two key network-layer functions

network-layer functions:

1 ▪ *forwarding:* move packets from a router's input link to appropriate router output link

2 ▪ *routing:* determine route taken by packets from source to destination

- *routing algorithms*

analogy: taking a trip

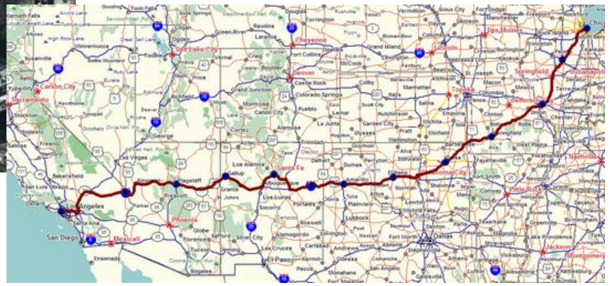
تنفيذ الخطة

▪ *forwarding:* process of getting through single interchange

▪ *routing:* process of planning trip from source to destination

الخطة

forwarding

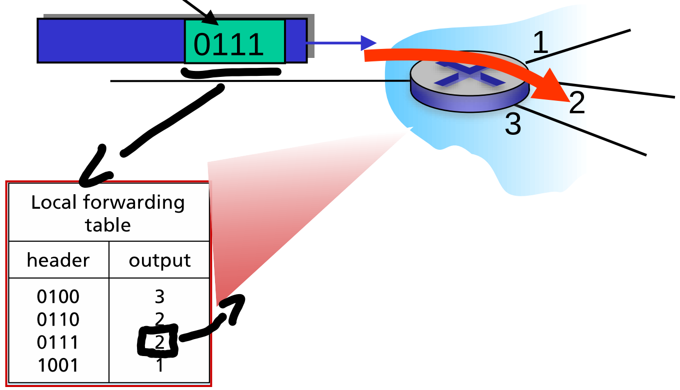


routing

Network layer: data plane, control plane

Data plane:

- *local*, per-router function
- determines how datagram arriving on router input port is *forwarded* to router output port



Control plane

- *network-wide* logic global
- determines how datagram is *routed* among routers along end-end path from source host to destination host
- two control-plane approaches:
  1. • *traditional routing algorithms:* implemented in routers
  2. • *software-defined networking (SDN)*: implemented in (remote) servers

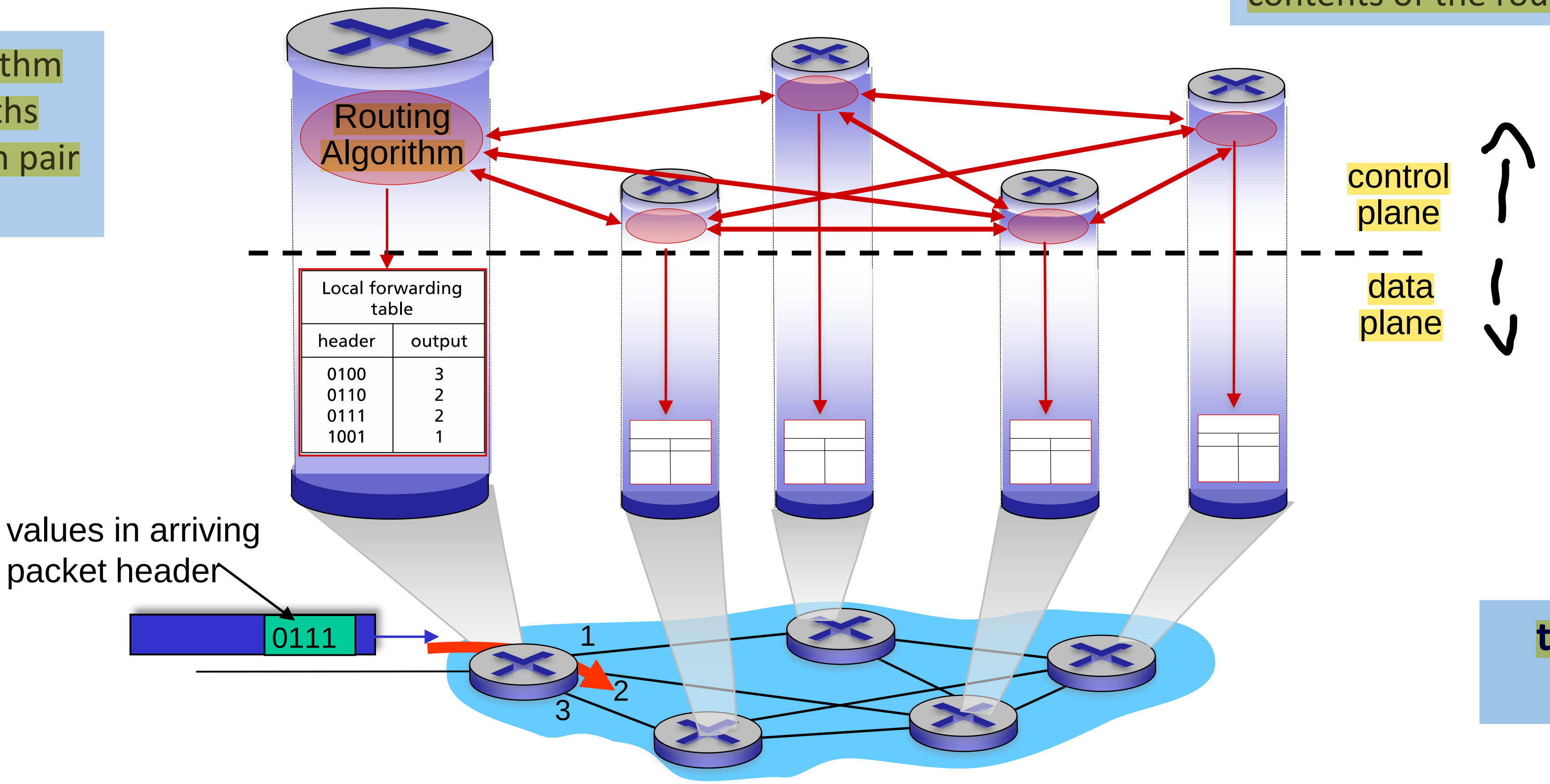
Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane

(interact by exchanging routing messages)

the routing algorithm determines the contents of the routers' forwarding tables

routing algorithm computes paths between each pair of routers

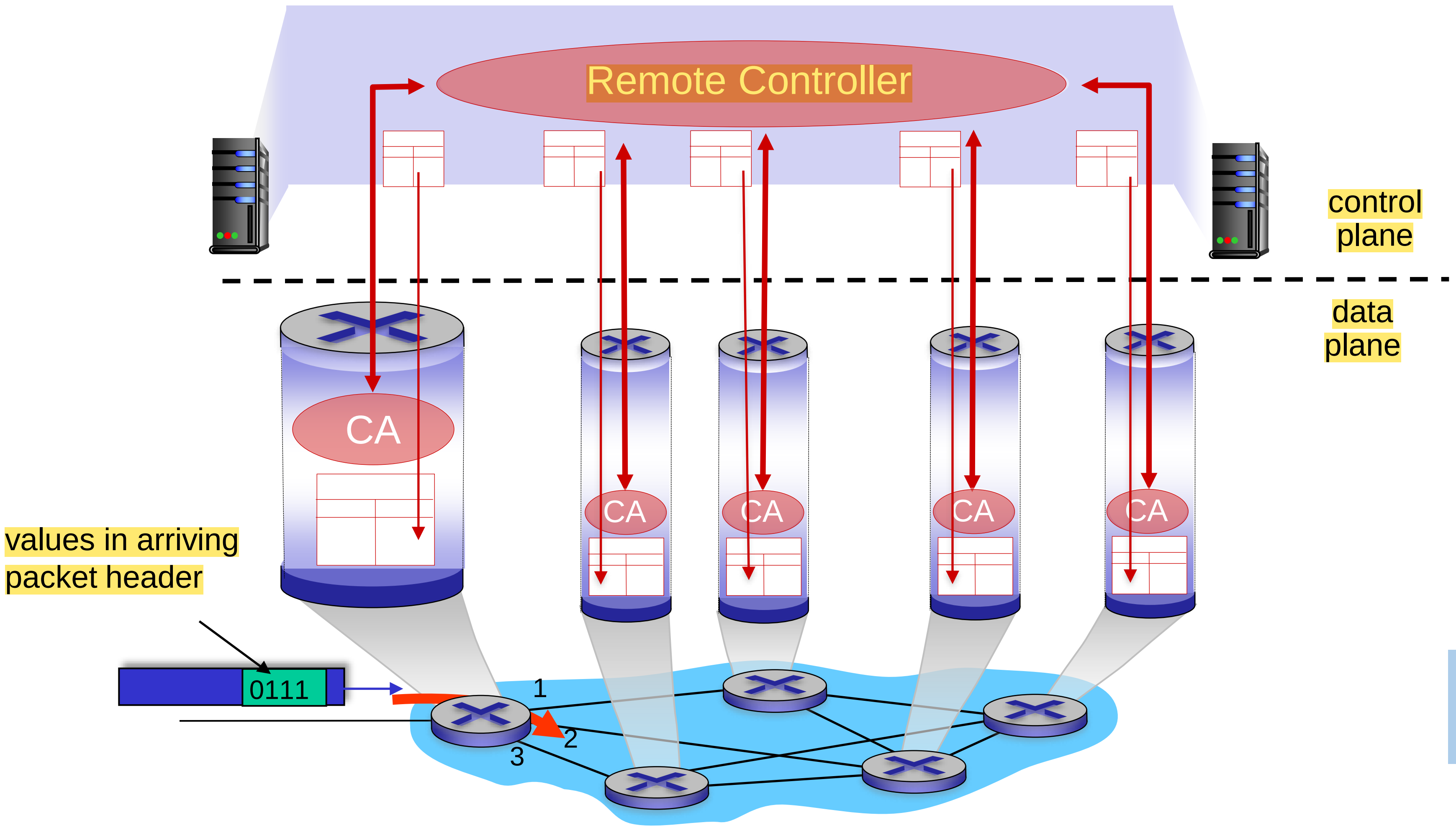


1

traditional routing algorithms

Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers



2

This is the concept of SDN

Network service model

Q: What *service model* for "channel" transporting datagrams from sender to receiver?

example services for *individual* datagrams:

- guaranteed delivery
- guaranteed delivery with less than 40 msec delay

example services for a *flow* of datagrams:

- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing

Network-layer service model

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no

Internet "best effort" service model

No guarantees on*:*

i. successful datagram delivery to destination
ii. timing or order of delivery
iii. bandwidth available to end-end flow

Network-layer service model

FYI

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no
ATM	Constant Bit Rate	Constant rate	yes	yes	yes
ATM	Available Bit Rate	Guaranteed min	no	yes	no
Internet	Intserv Guaranteed (RFC 1633)	yes	yes	yes	yes
Internet	Diffserv (RFC 2475)	possible	possibly	possibly	no

There are other service models with better QoS, but none of them succeeded

Reflections on best-effort service:

FYI

- simplicity of mechanism has allowed Internet to be widely deployed adopted
- sufficient provisioning of bandwidth allows performance of real-time applications (e.g., interactive voice, video) to be “good enough” for “most of the time”
- replicated, application-layer distributed services (datacenters, content distribution networks) connecting close to clients’ networks, allow services to be provided from multiple locations
- congestion control of “elastic” services helps

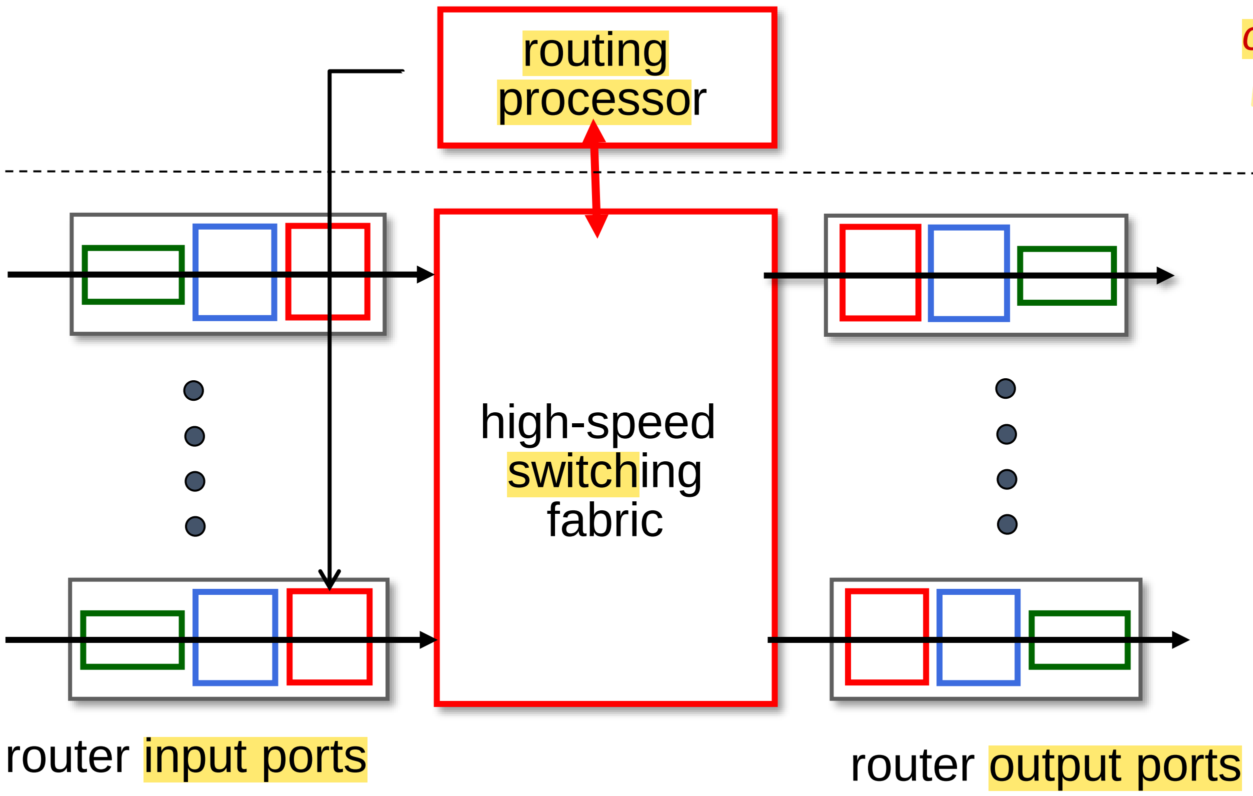
It’s hard to argue with success of best-effort service model

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

Router architecture overview

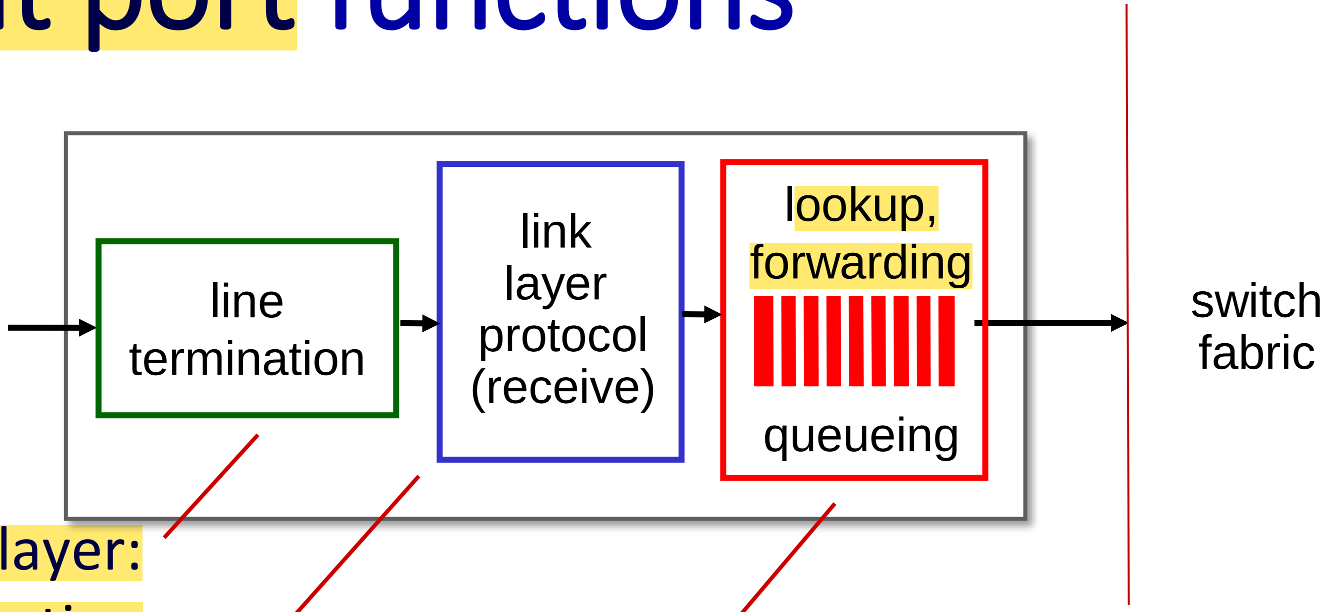
high-level view of generic router architecture:



routing, management
control plane (software)
operates in millisecond
time frame

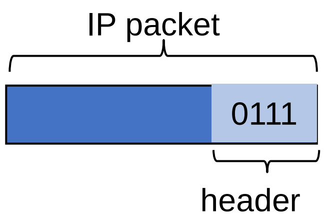
forwarding data plane
(hardware) operates
in nanosecond
timeframe

Input port functions



physical layer:
bit-level reception

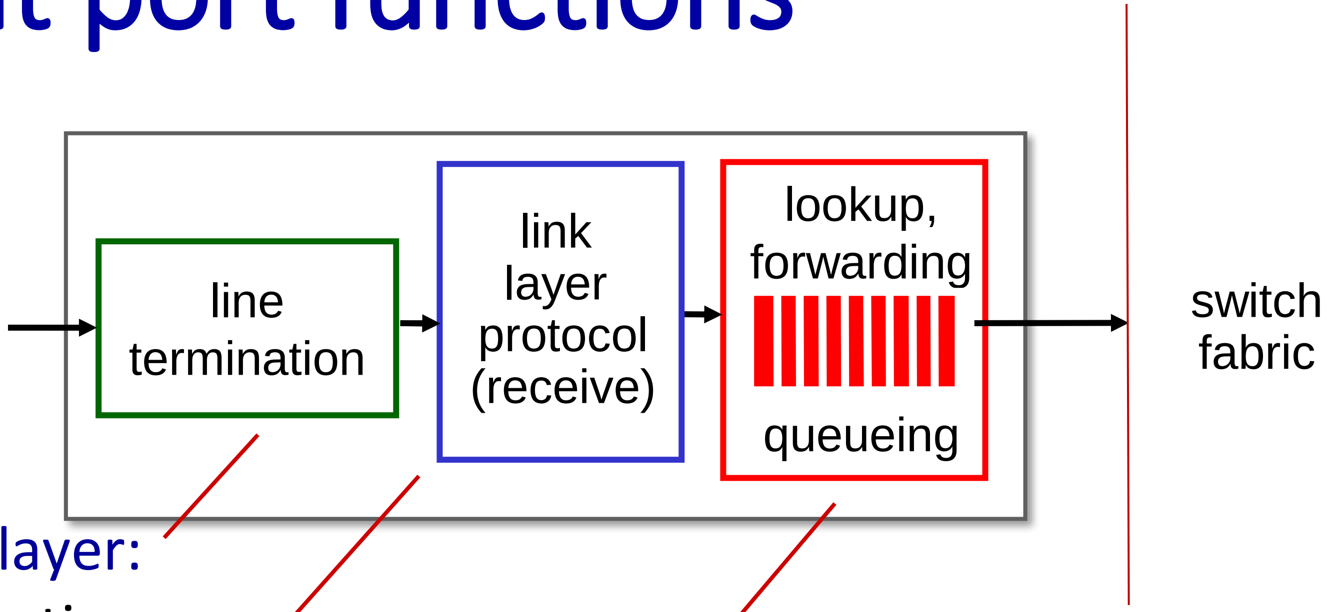
link layer:
e.g., Ethernet (chapter 6)



decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
- goal: complete input port processing at 'line speed'
- input port queuing: if datagrams arrive faster than forwarding rate into switch fabric

Input port functions



physical layer:
bit-level reception

link layer:
e.g., Ethernet
(chapter 6)

decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
1 - destination-based forwarding: forward based only on destination IP address (traditional)
2 - generalized forwarding: forward based on any set of header field values

Destination-based forwarding

forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through	0
11001000 00010111 00010000 00000100 through 11001000 00010111 00010000 00000111	3
11001000 00010111 00011000 11111111	
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Q: but what happens if ranges don't divide up so nicely?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, 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

examples:

11001000 00010111 00010110 10100001 which interface? 0

11001000 00010111 00011000 10101010 which interface? 1

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range				Link interface
11001000	00010111	00010***	********	0
11001000	000[illegible]0111	00011000	********	1
11001000	[illegible]1	00011***	********	2
otherwise				3

match!

examples:

11001000	00010111	00010110	10100001	which interface?
11001000	00010111	00011000	10101010	which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, 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

match!

examples:

11001000	00010111	00010110	10100001	which interface?
11001000	00010111	00011000	10101010	which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, 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

match!

✓ longest

examples:

11001000	00010111	00010110	10100001	which interface?
11001000	00010111	00011000	10101010	which interface?

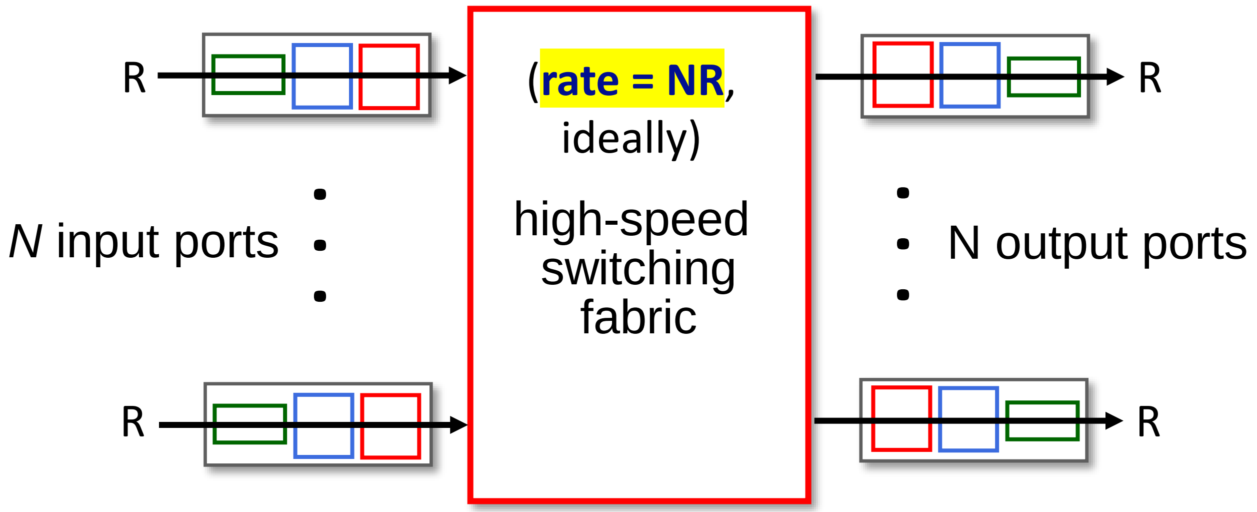
Longest prefix matching

FYI

- we'll see *why* longest prefix matching is used shortly, when we study addressing
- longest prefix matching: often performed using ternary content addressable memories (TCAMs)
  - *content addressable:* present address to TCAM: retrieve address in one clock cycle, regardless of table size
  - Cisco Catalyst: ~1M routing table entries in TCAM

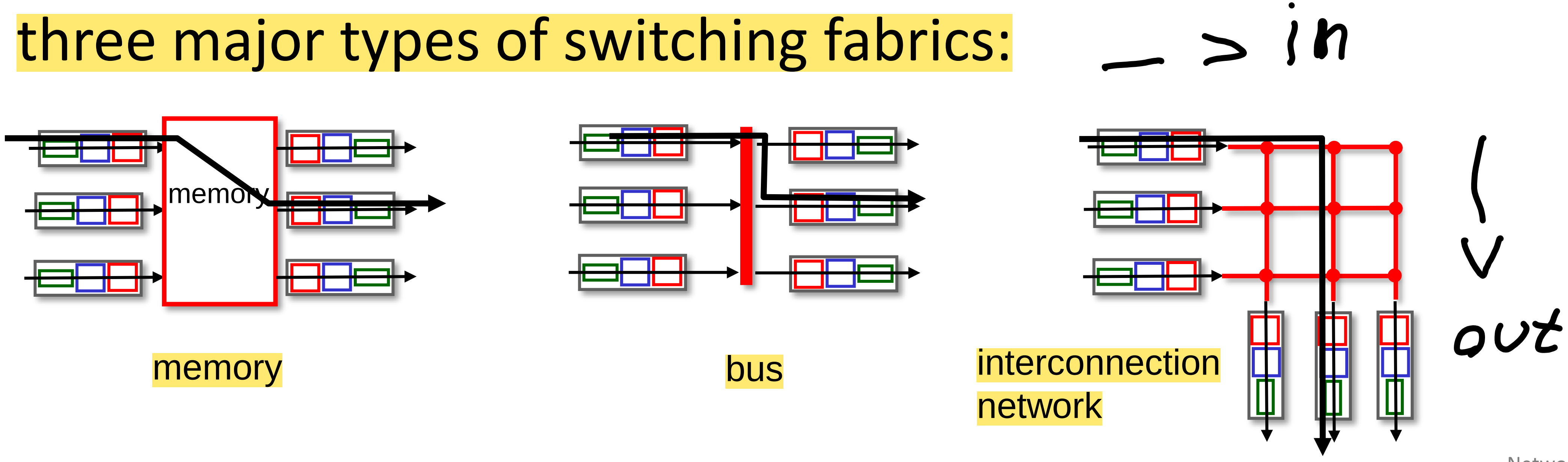
Switching fabrics

- transfer packet from input link to appropriate output link
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - For *N* inputs: switching rate *N* times line rate R is desirable



Switching fabrics

- transfer packet from input link to appropriate output link
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - For *N* inputs: switching rate *N* times line rate R is desirable
- three major types of switching fabrics:



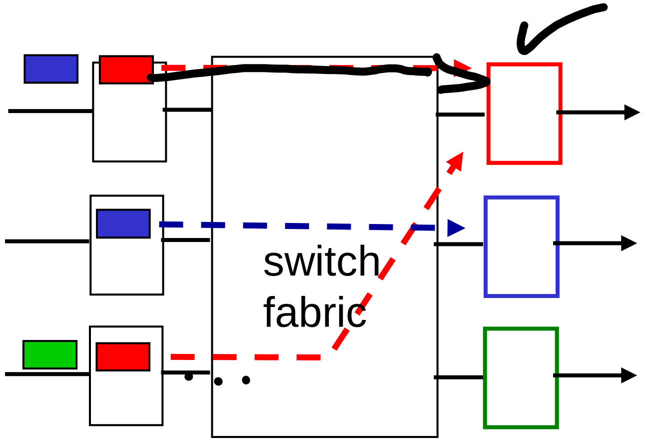
memory

bus

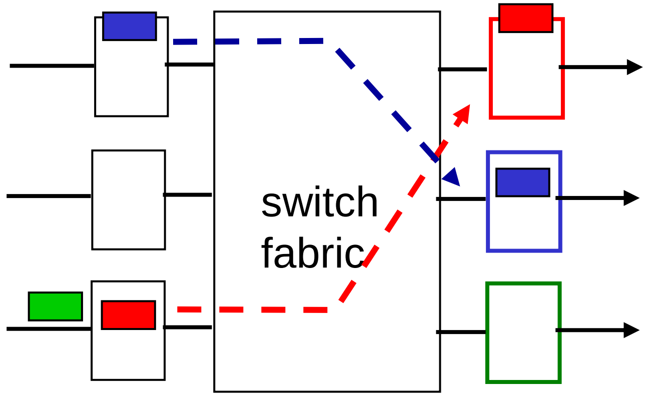
interconnection network

Input port queuing

- If switch fabric slower than input ports combined -> queueing may occur at input queues
  - queueing delay and loss due to input buffer overflow!
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward



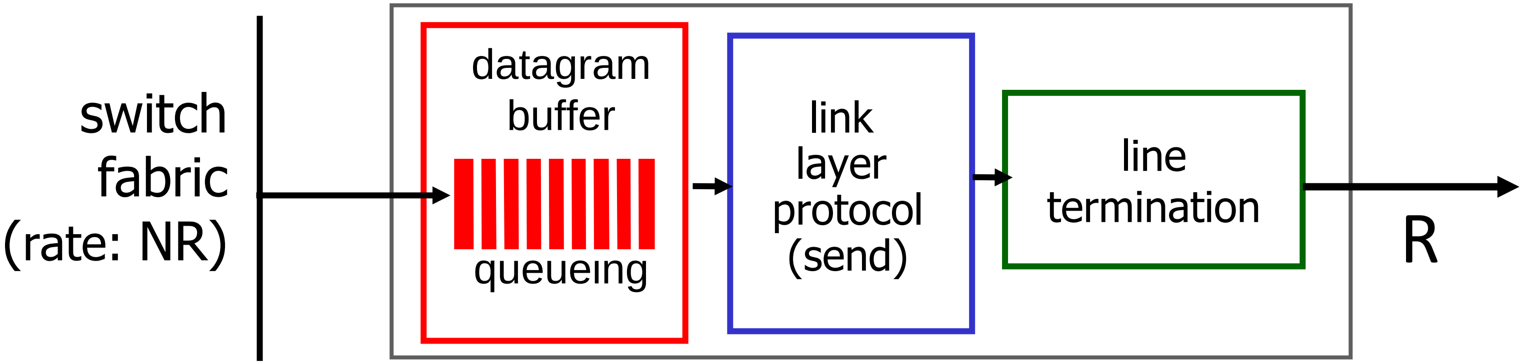
output port contention: only one red datagram can be transferred. lower red packet is *blocked*



one packet time later: green packet experiences HOL blocking

Output port queuing

This is a really important slide



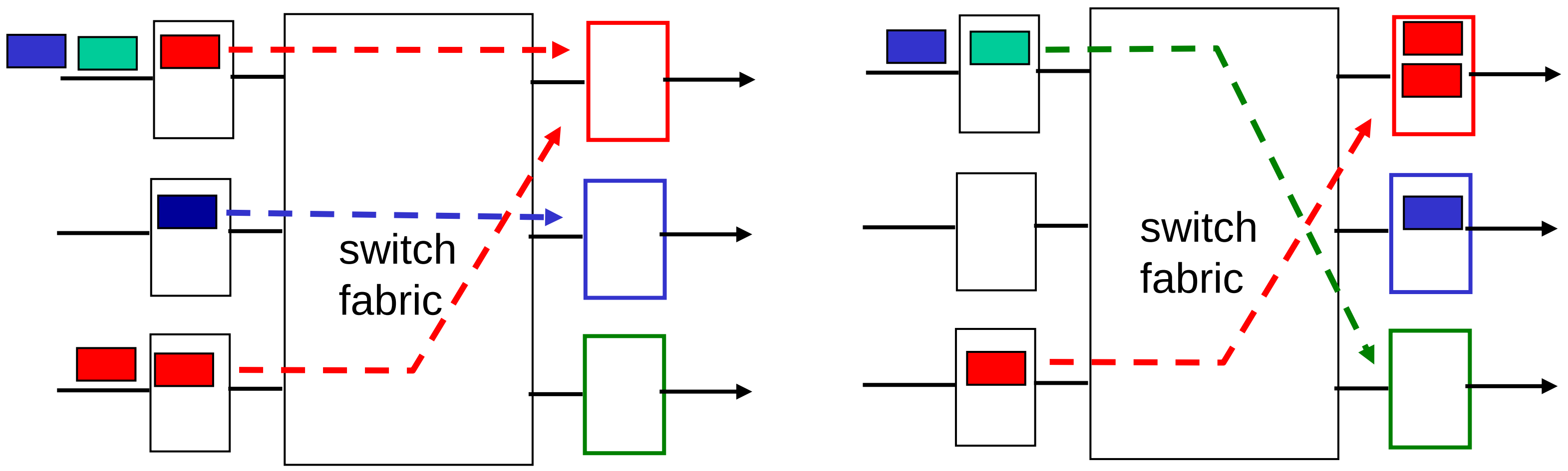
- *Buffering* required when datagrams arrive from fabric faster than link transmission rate. *Drop policy:* which datagrams to drop if no free buffers?

Datagrams can be lost due to congestion, lack of buffers

- *Scheduling discipline* chooses among queued datagrams for transmission

Priority scheduling – who gets best performance, network neutrality

Output port queuing



at *t*, packets move from input to output

one packet time later

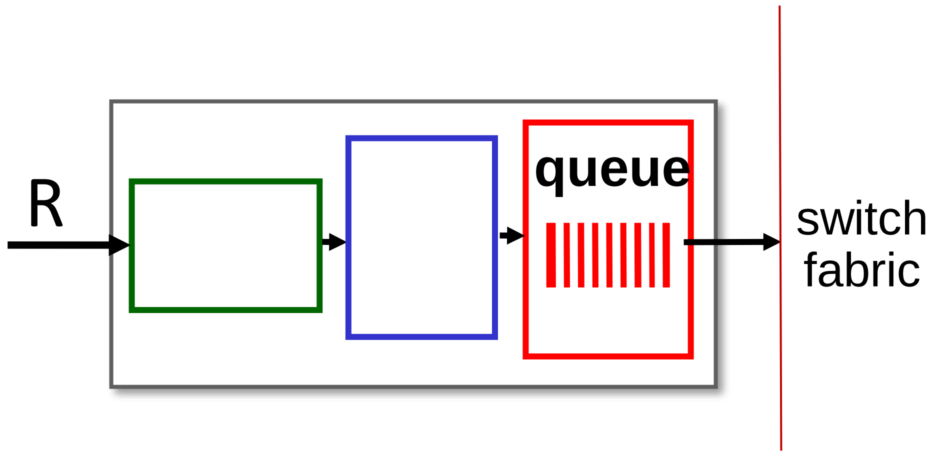
- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

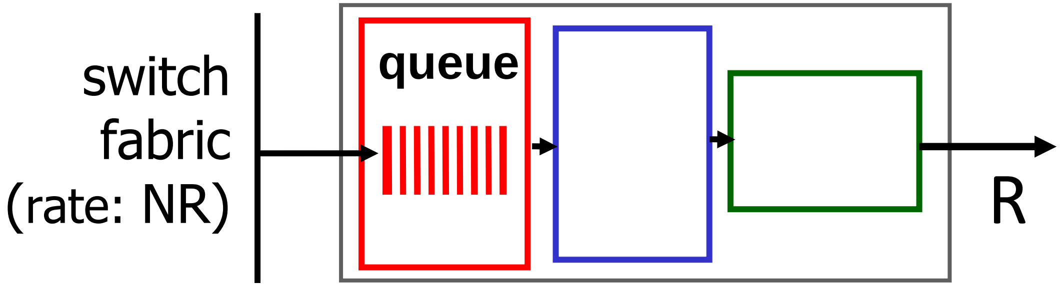
Queuing

REMEMBER we said that

packets can be "lost within the network" or "dropped at a router."

- It is *here*, at these *queues* within a router, where such packets are actually dropped and lost.
- The location and extent of queueing (either at the input port queues or the output port queues) will depend on:
  1 ▪ the traffic load,
  2 ▪ the speed of the switching fabric, and
  3 ▪ the line speed.



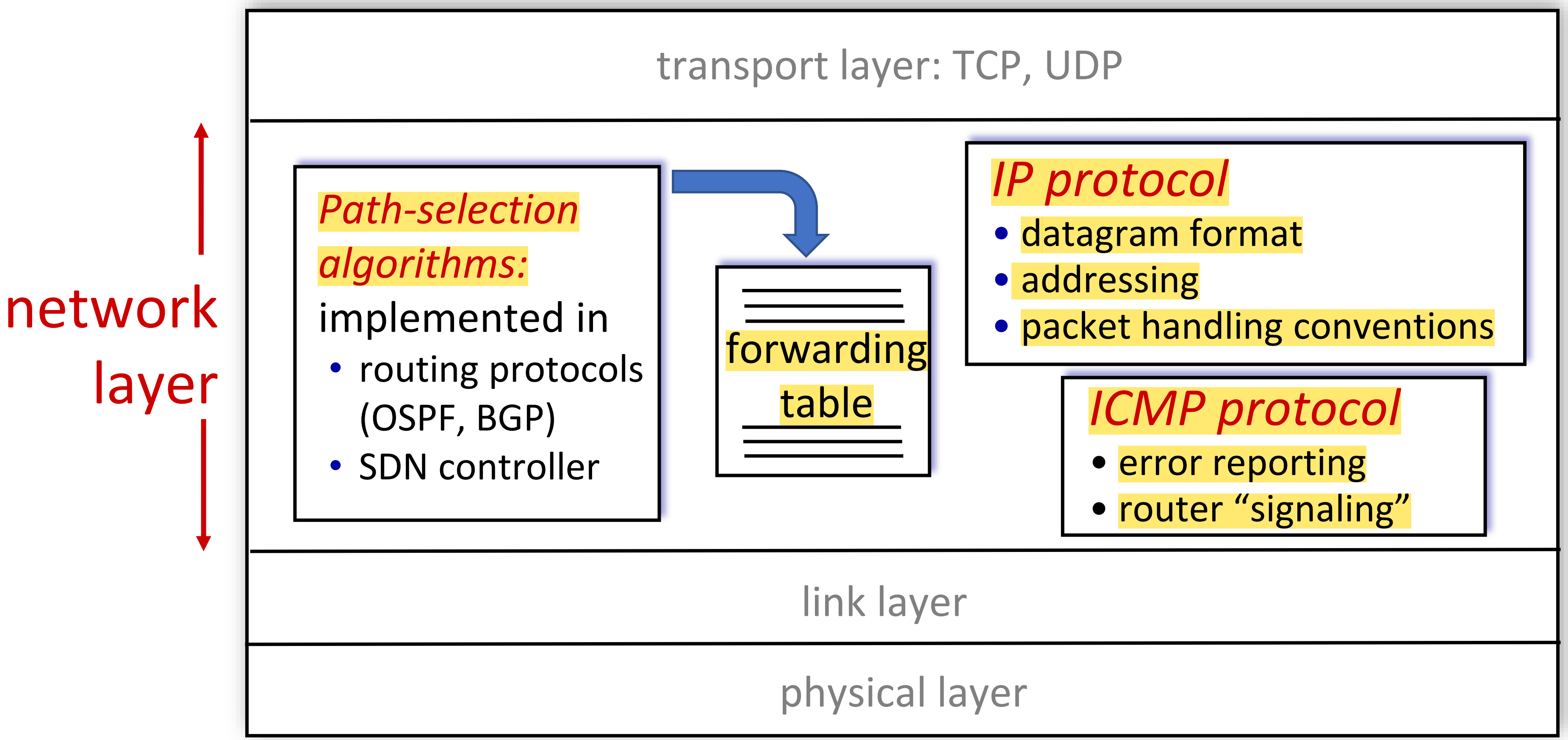


Network layer: "data plane" roadmap

- 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

Network Layer: Internet

host, router network layer functions:



IP Datagram format

IP protocol version number

header length(bytes)

To distinguish different types of IP datagrams.

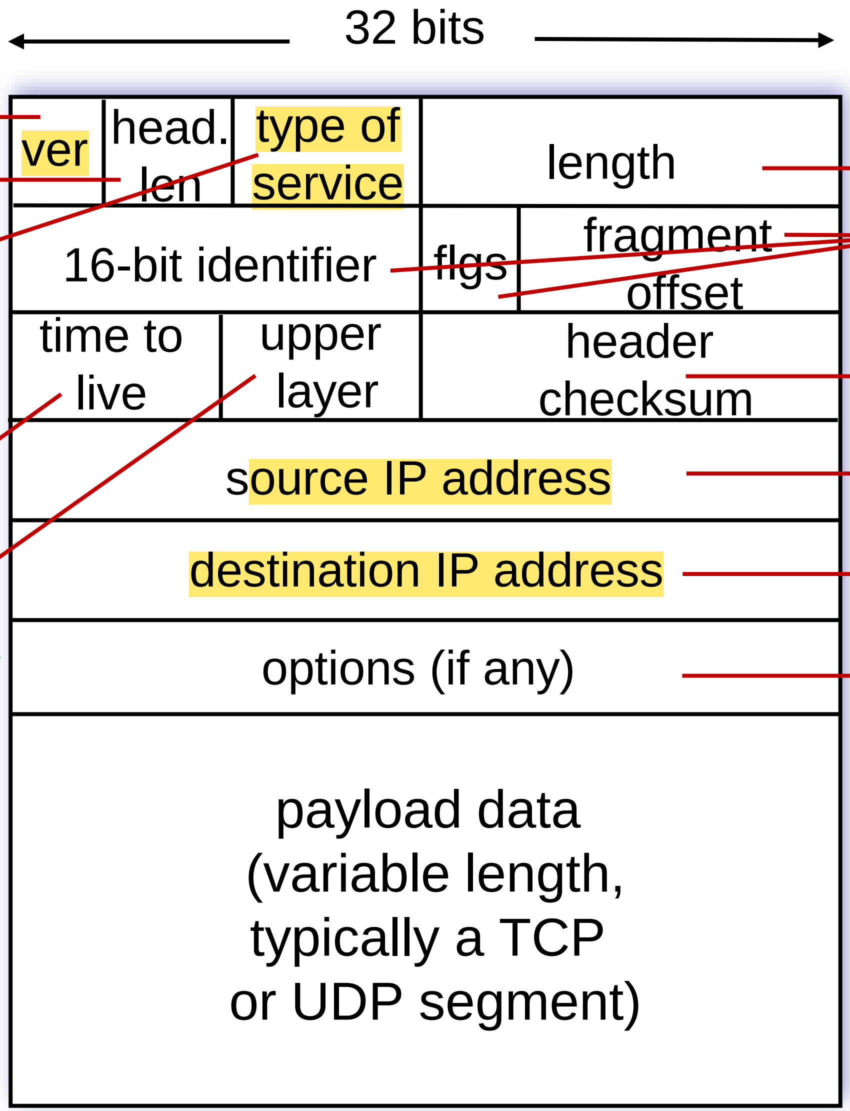
"type" of service:

- diffserv (0:5)
- ECN (6:7)

TTL: remaining max hops (decremented at each router)

upper layer protocol (e.g., TCP 6 or UDP 17)

https://en.wikipedia.org/wiki/List_of_IP_protocol_numbers



total datagram length (bytes)

fragmentation/ reassembly

header checksum

32-bit source IP address

Maximum length: 64K bytes

Typically: 1500 bytes or less

e.g., timestamp, record route taken

overhead

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

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

IP addressing: introduction

- IP address: 32-bit = 4 bytes separated by "." dots (also called 4 octs)
- Each bit can be 0 or 1 (2^1 possibilities)
- 32 bits total possibilities 2^{32} (> 4 billion)
- Each oct has a value between 0 and 255

IP address format

8 . 8 . 8 . 8

xxxxxxxx . xxxxxxxx . xxxxxxxx . xxxxxxxx

Ex: 10100000 . 00110100 . 00001111 . 11110000

160 . 52 . 15 . 240

Oct format

X X X X X X X X

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

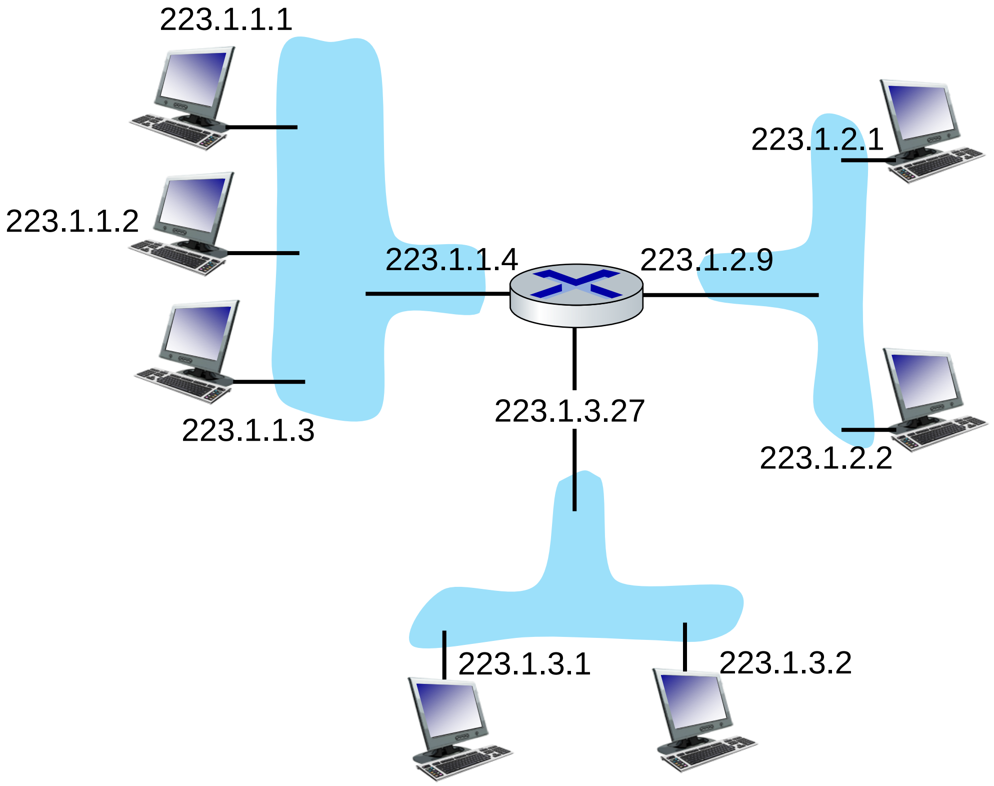
128 64 32 16 8 4 2 1 =255

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

160 . **52** . **15** . **240**

IP addressing: introduction

- IP address: 32-bit identifier associated with each host or router *interface*
- interface: connection between host or router and a physical link
  - router's typically have multiple interfaces
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)



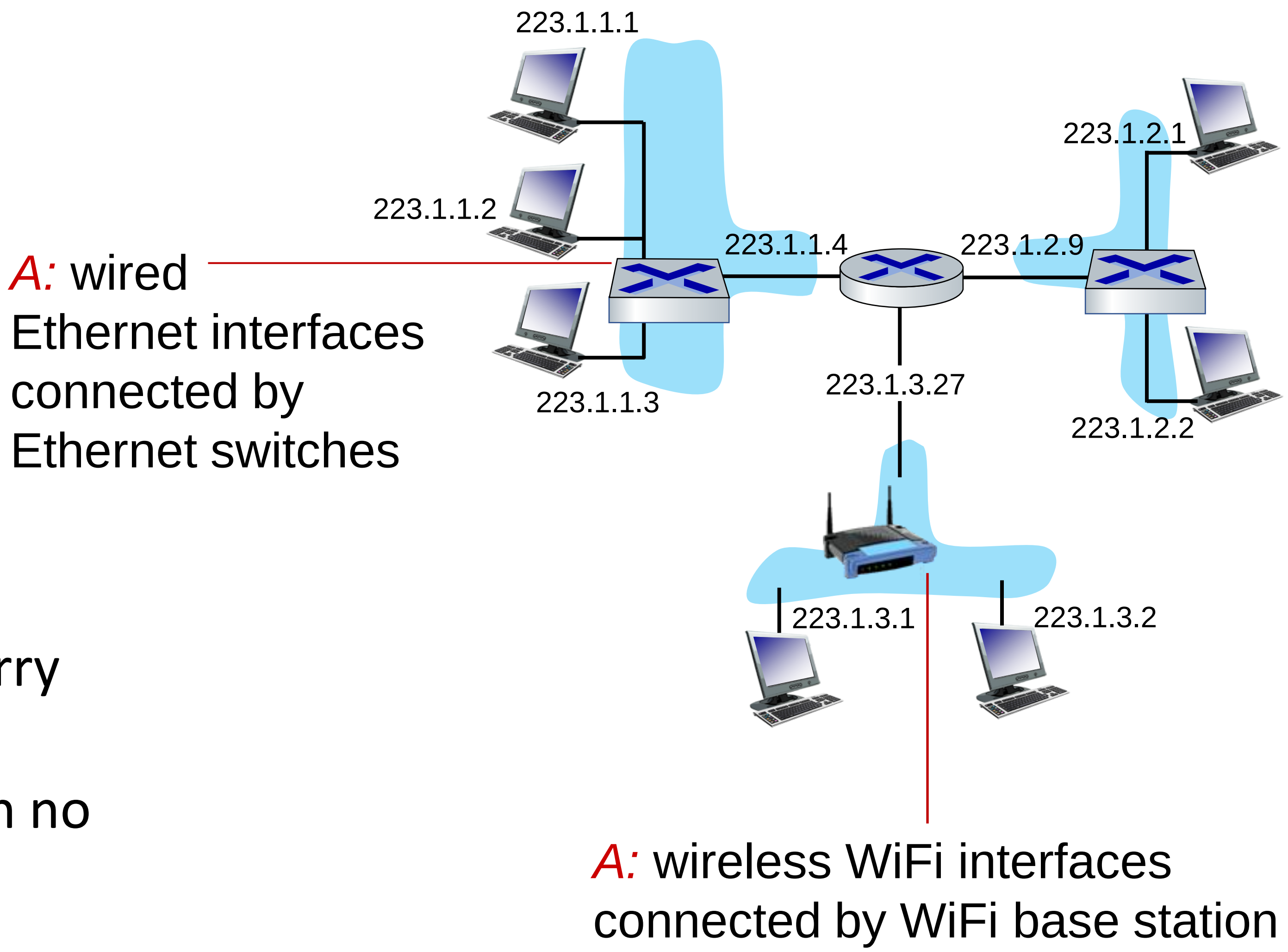
dotted-decimal IP address notation:

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

IP addressing: introduction

Q: how are interfaces actually connected?

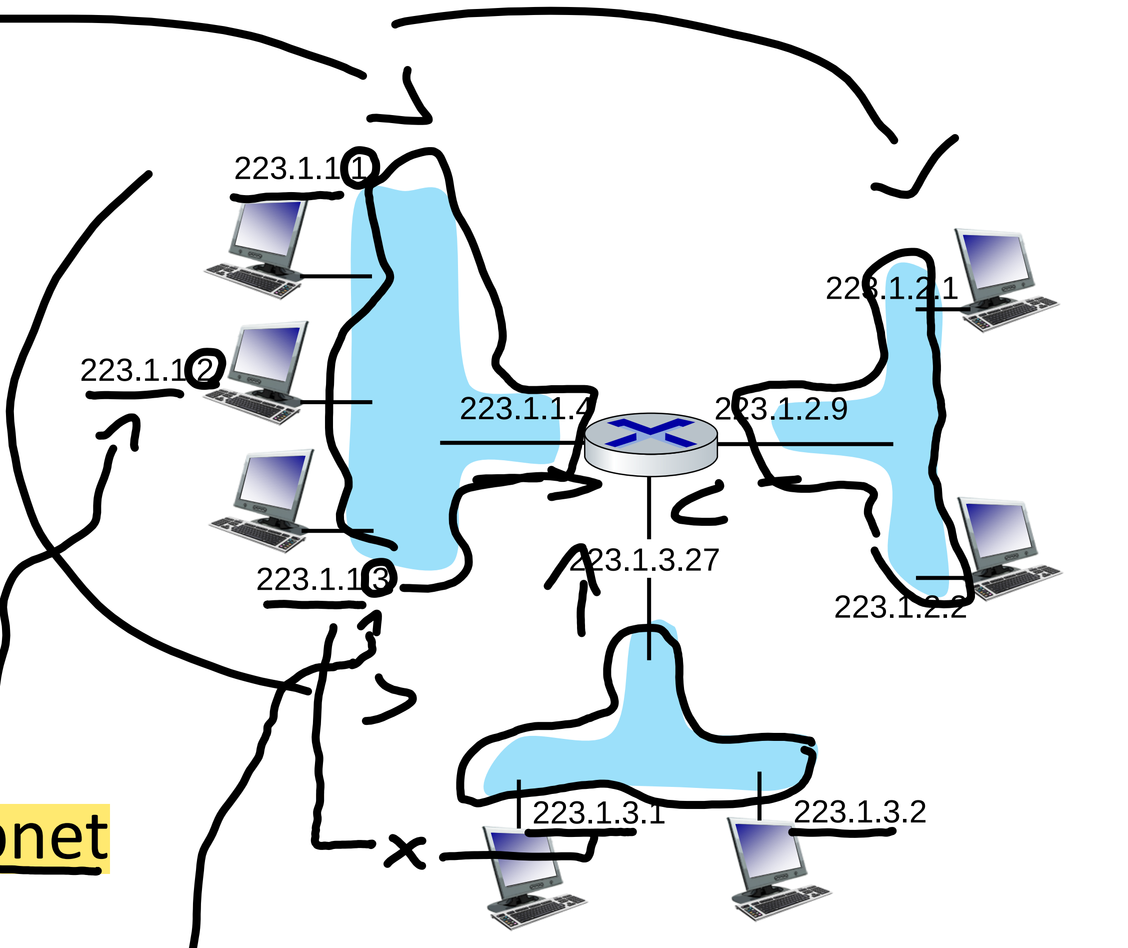
A: we'll learn about that in chapters 6, 7



For now: don't need to worry about how one interface is connected to another (with no intervening router)

Subnets

- *What's a subnet ?*
  - device interfaces that can physically reach each other without passing through an intervening router
- IP addresses have structure:
  1. • subnet part: devices in same subnet have common high order bits
  2. • host part: remaining low order bits



network consisting of 3 subnets

8 8 8 = 24

223.1.1.1 = 11011111 00000001 00000001 00000001
223.1.1.2 = 11011111 00000001 00000001 00000010
223.1.1.3 = 11011111 00000001 00000001 00000011
223.1.1.4 = 11011111 00000001 00000001 00000100

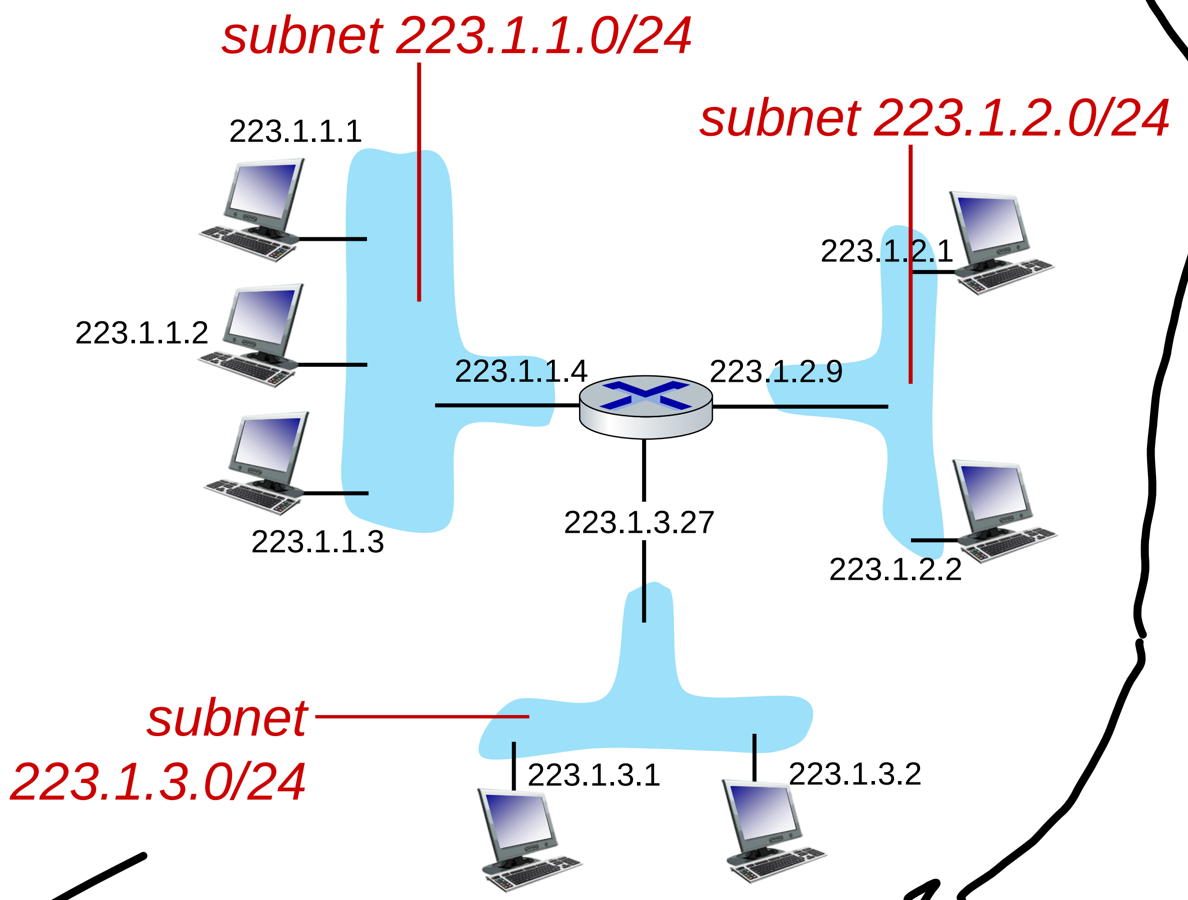
Sub host

A portion of an interface's IP address will be determined by the subnet to which it is connected.

Subnets

Recipe for defining subnets:

- detach each interface from its host or router, creating "islands" of isolated networks
- each isolated network is called a *subnet*

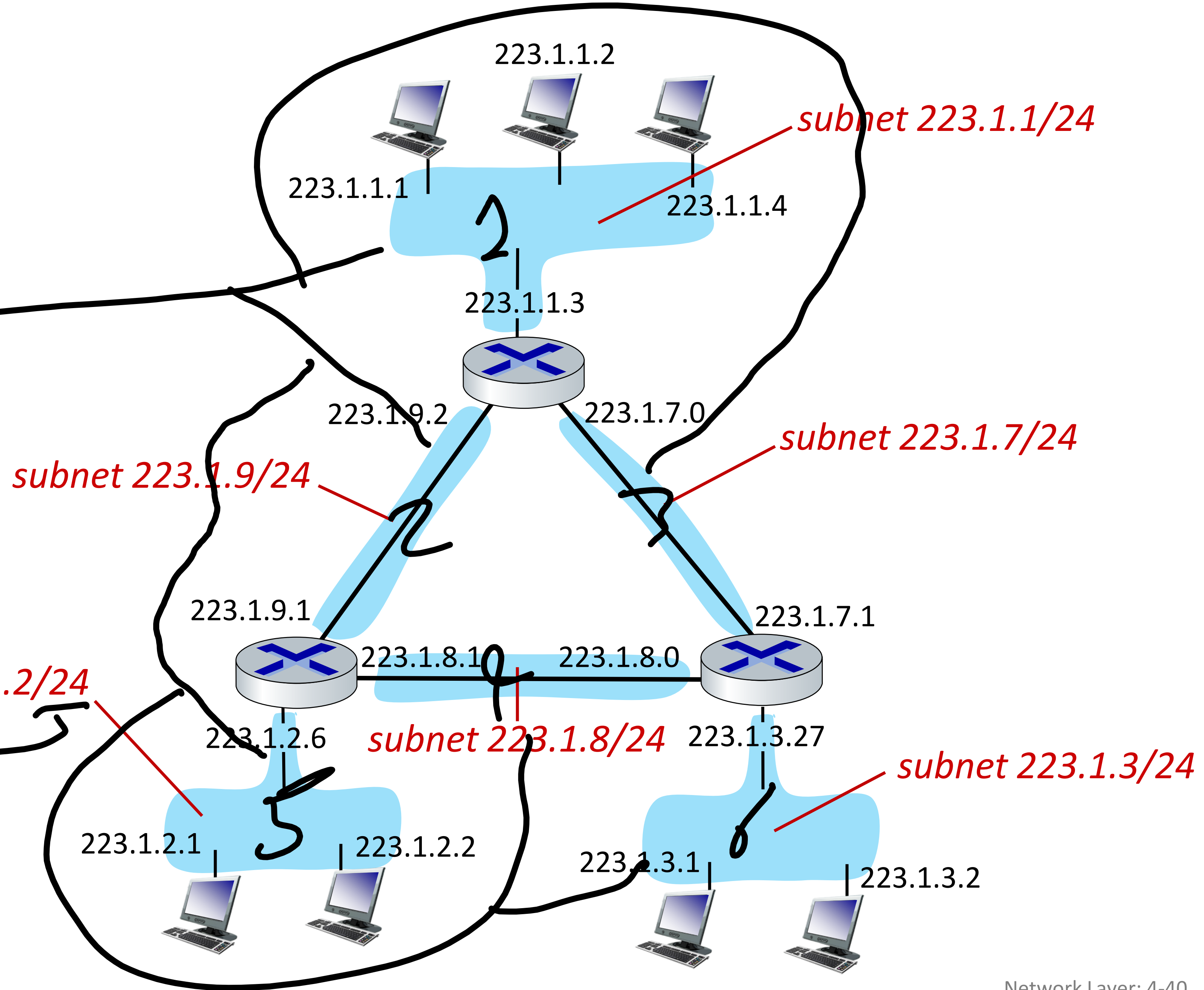


subnet mask: /24

(high-order 24 bits: subnet part of IP address)

Subnets

- where are the subnets?
- what are the /24 subnet addresses?



223.1.1 = 11011111 00000001 00000001 xxxxxxxx
223.1.2 = 11011111 00000001 00000010 xxxxxxxx
223.1.3 = 11011111 00000001 00000011 xxxxxxxx
223.1.4 = 11011111 00000001 00000100 xxxxxxxx

IP Classful addressing

In the past, when an organization requests a range of IP addresses, they receive a from one of these classes:

1 ▪ **Class A**: from 0.0.0.1 to 126.0.0.0
- 126 networks and 2^{24} = 16,777,216 hosts.
- 1 byte for the network & 3 bytes for the host.
- Mask: 255.0.0.0 /8

2 ▪ **Class B**: from 128.0.0.0 to 191.255.0.0
- 16,384 networks and 2^{16}= 65,534 hosts.
- 2 bytes for the network & two for the host.
- Mask: 255.255.0.0 /16

3 ▪ **Class C:** from 192.0.0.0 to 223.255.255.0
- 2,097,152 networks and 2^{8}= 254 hosts.
- 3 bytes for the network and the 4th byte for the host.
- Mask: 255.255.255.0 /24

Organization requires

20 addresses? C

2000 addresses? B

Which class they get?

4 ▪ **Class D**: from 224.0.0.0 to 239.255.255.255.
- for multicasting.

5 ▪ **Class E**: from 240.0.0.0 to 255.255.255.255.
- used for experimentation.

IP addressing: CIDR

CIDR: Classless InterDomain Routing (pronounced "cider")

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

if <—

subnet part ——— host + part

11001000 00010111 00010000 00000000

if
——— >

200.23.16.0/23

host —

IP addressing

$\longleftarrow$ x $\longrightarrow$ $\longleftarrow$ 32-x $\longrightarrow$

aaaaaaaa bbbbbbbb cccccccc dddddddd

a.b.c.d/x Number of IP addresses = 2^{32-x}

a.b.c.d/20	4096	a.b.c.d/26	64 −2
a.b.c.d/21	2048	a.b.c.d/27	32
a.b.c.d/22	1024	a.b.c.d/28	16
a.b.c.d/23	512	a.b.c.d/29	8
a.b.c.d/24	256	a.b.c.d/30	4
a.b.c.d/25	128	a.b.c.d/31	2 −2 = 0 ?

- 1st IP address = Network address
- Last IP address = broadcast address
- Number of IP addresses for hosts = $2^{32-x}-2$
  - a.b.c.d/x → 4096 IP address & 4094 address for hosts

Example

223 1 1 0

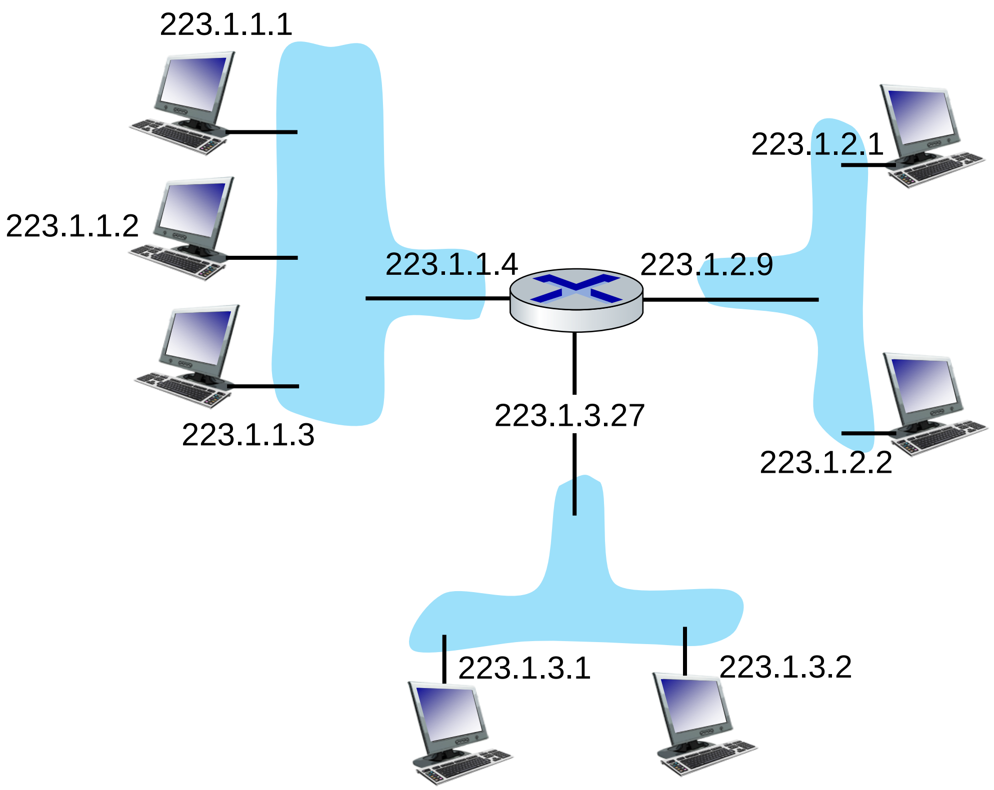
223.1.1.0 = 11011111 00000001 00000001 00000000

128 64 16 8 4 2 1

Leftmost bits = 24

223.1.1.0/24

→ *Address Range* *223.1.1.0* to *223.1.1.255*



subnet mask: /24

Mask Notation

- Values: *Network = 1*   &   *Host = 0*
- Classful example (Class B address)
  - 128.35.17.25/ 16   2   8
  - Binary: 11111111 . 11111111 . 00000000 . 00000000
  - Decimal: 255 . 255 . 0 . 0
- Classless IP example
  - 128.35.17.25/ 17   8   8   1
  - Binary: 11111111 . 11111111 . 10000000 . 00000000
  - Decimal: 255 . 255 . 128 . 0

IP addresses: how to get one?

That's actually two questions:

1. Q: How does a *host* get IP address within its network (host part of address)?
2. Q: How does a *network* get IP address for itself (network part of address)

How does ***host*** get IP address?

- hard-coded by sys. admin in config file (e.g., /etc/rc.config in UNIX)
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
    - "plug-and-play"

DHCP: Dynamic Host Configuration Protocol

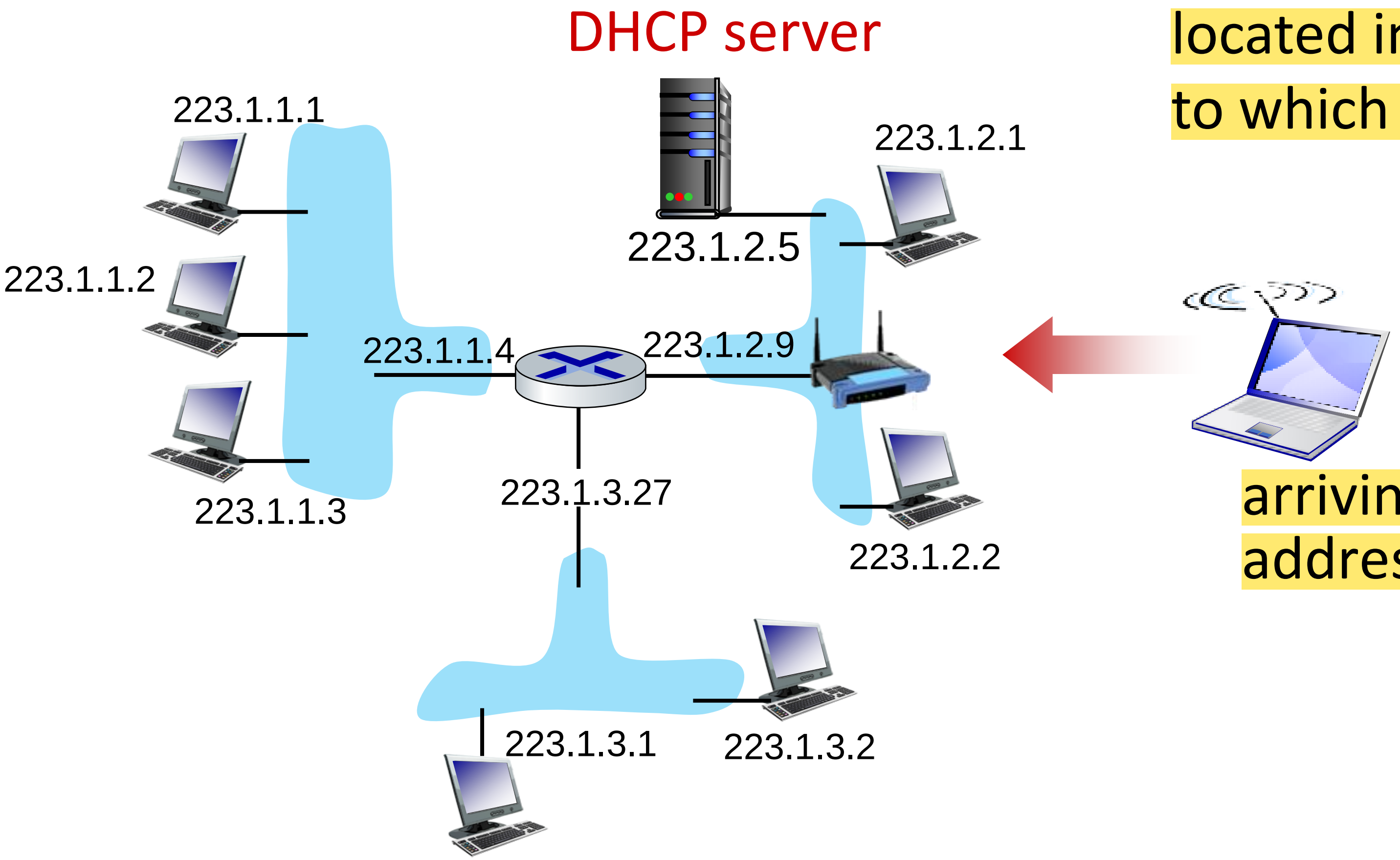
goal: host *dynamically* obtains IP address from network server when it "joins" the network

- 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

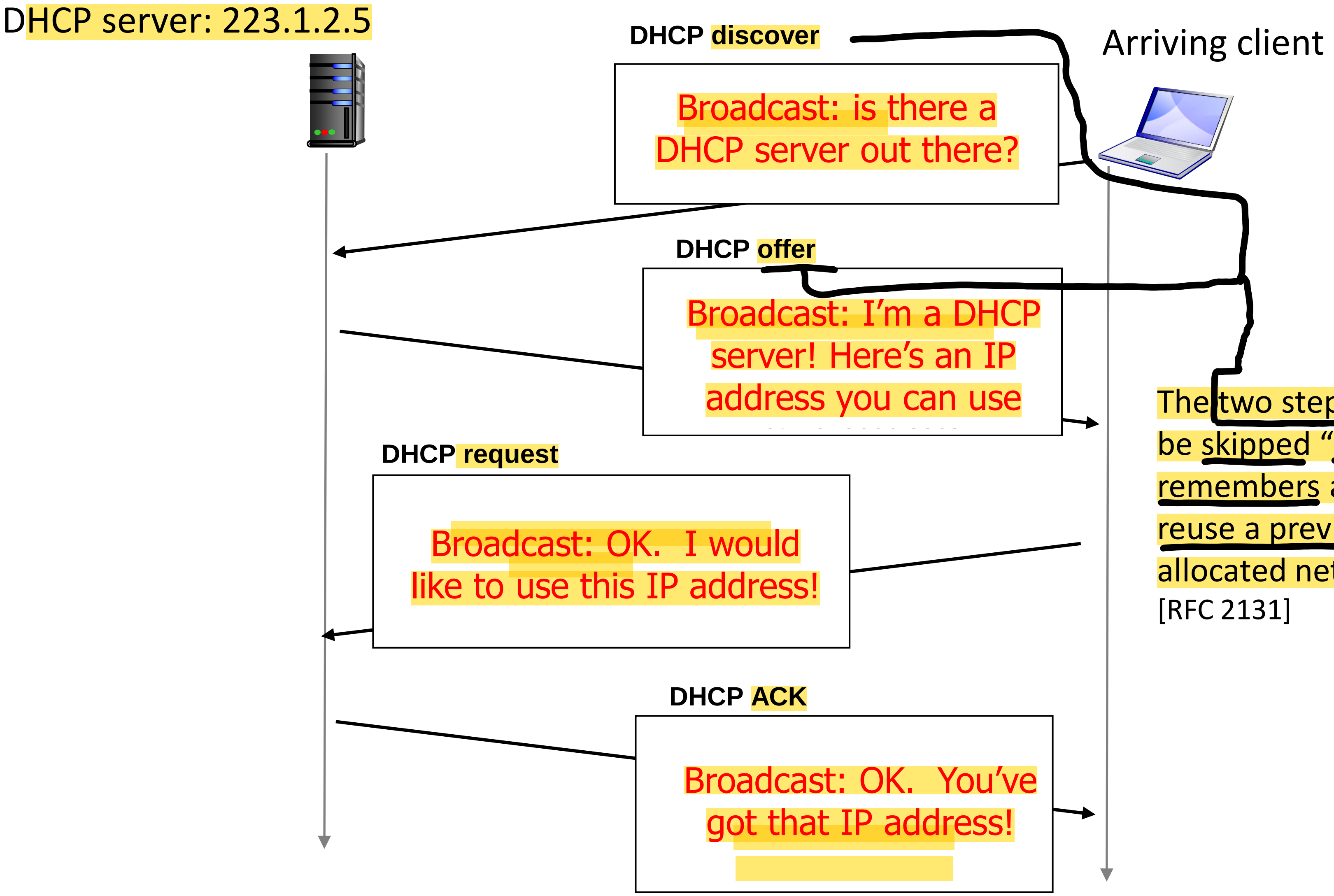
arriving DHCP client needs address in this network

DHCP client-server scenario

Initially, client does not have IP address and does not know server IP address → use 0.0.0.0

DHCP uses (**UDP**), server port# 67

IP 255.255.255.255 is used for *broadcast*

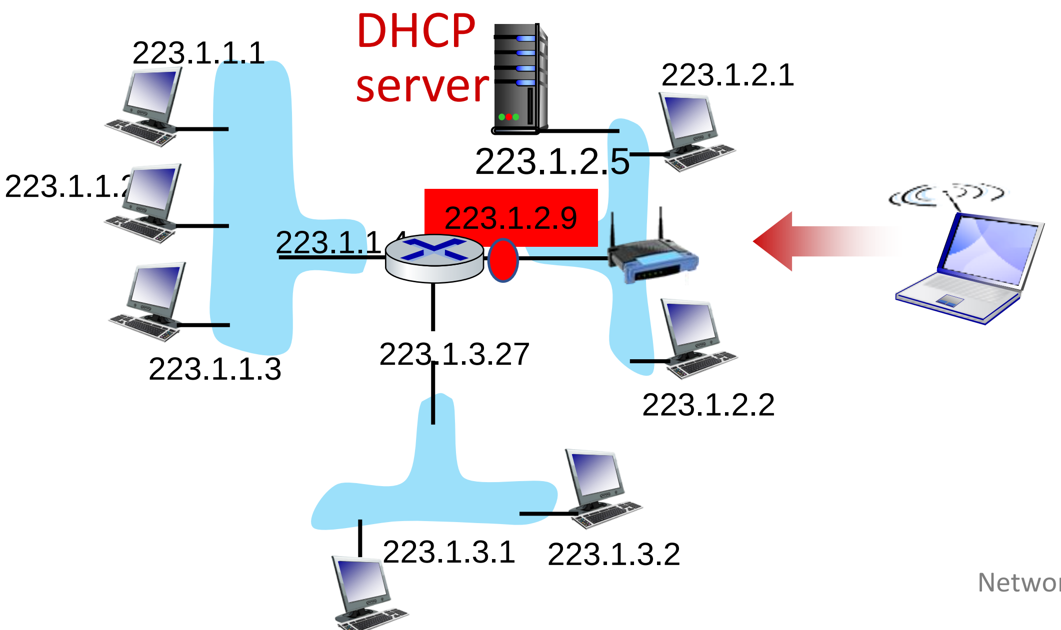


The two steps above can be skipped "if a client remembers and wishes to reuse a previously allocated network address" [RFC 2131]

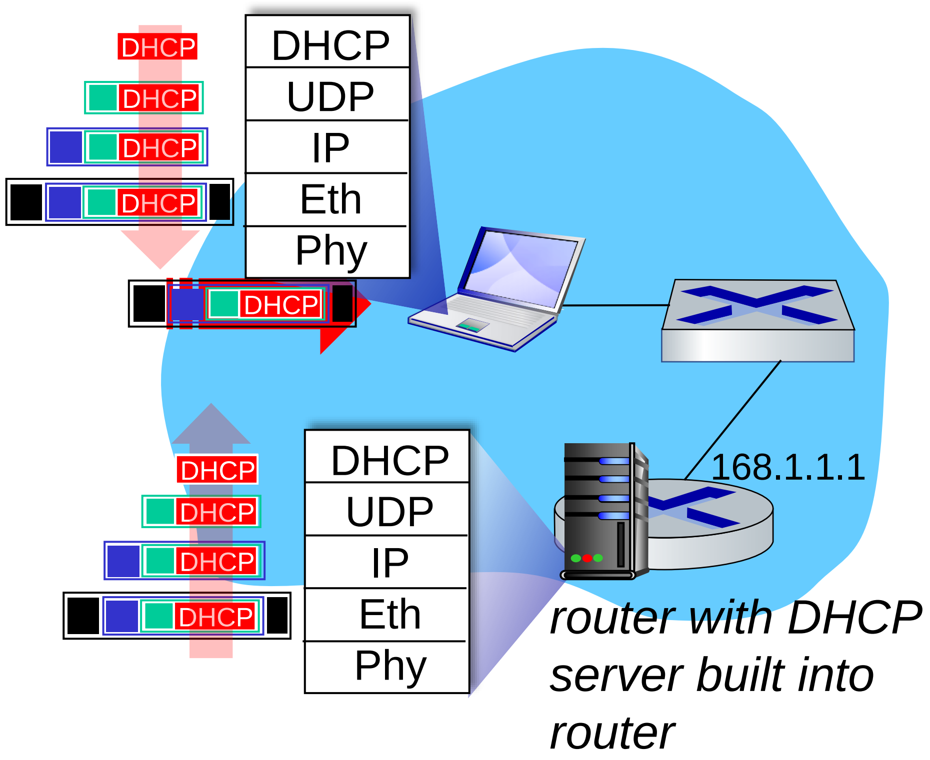
DHCP: more than IP addresses

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

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

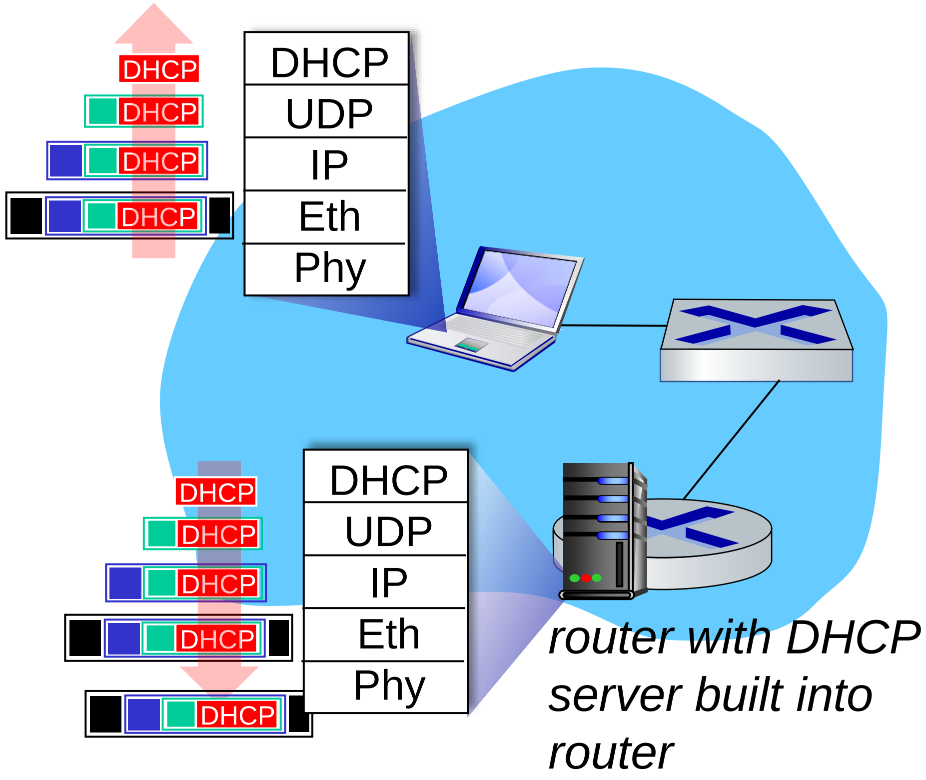


DHCP: example



- 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



- DCP 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

IP addresses: how to get one?

Q: how does *network* get subnet part of IP address?

A: gets allocated portion of its provider ISP's address space

8 2 4

ISP's block 11001000 00010111 00010000 00000000 200.23.16.0/20

8+8=16+4=20

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

8 8 4+3=7

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

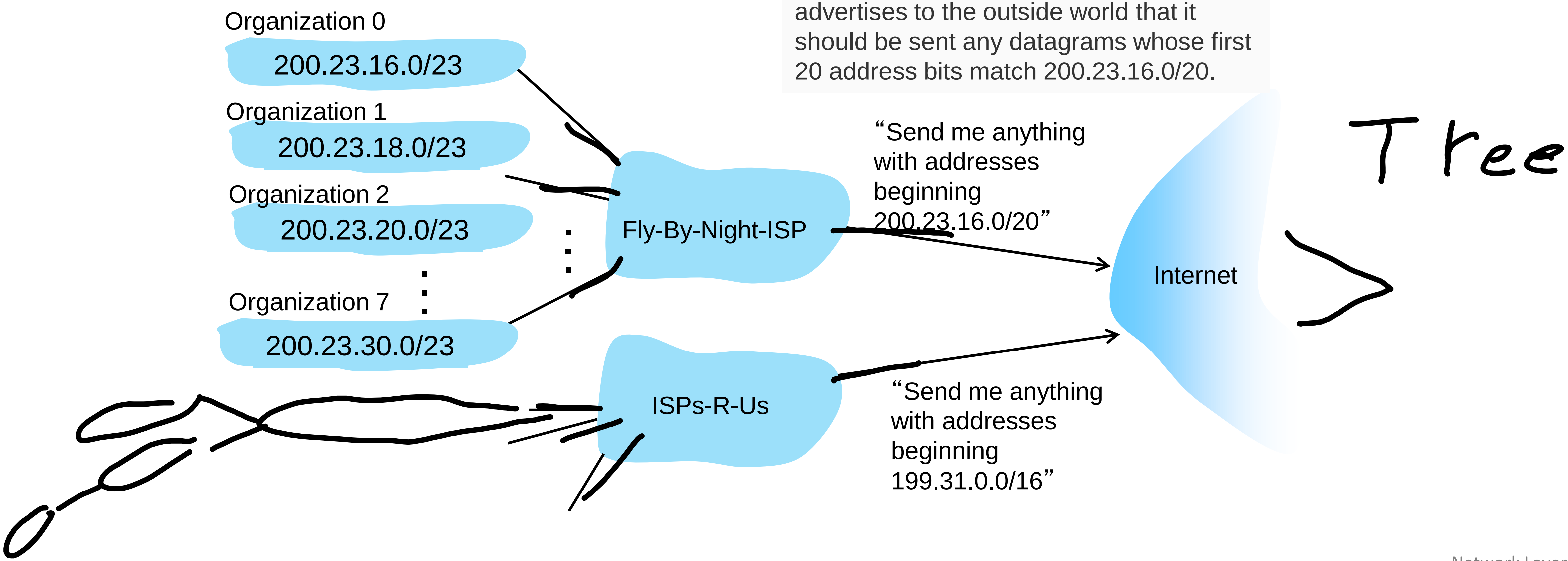
8+8+7 = 23

Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:

Route aggregation facilitate routing

advertises to the outside world that it should be sent any datagrams whose first 20 address bits match 200.23.16.0/20.



Hierarchical addressing: route aggregation

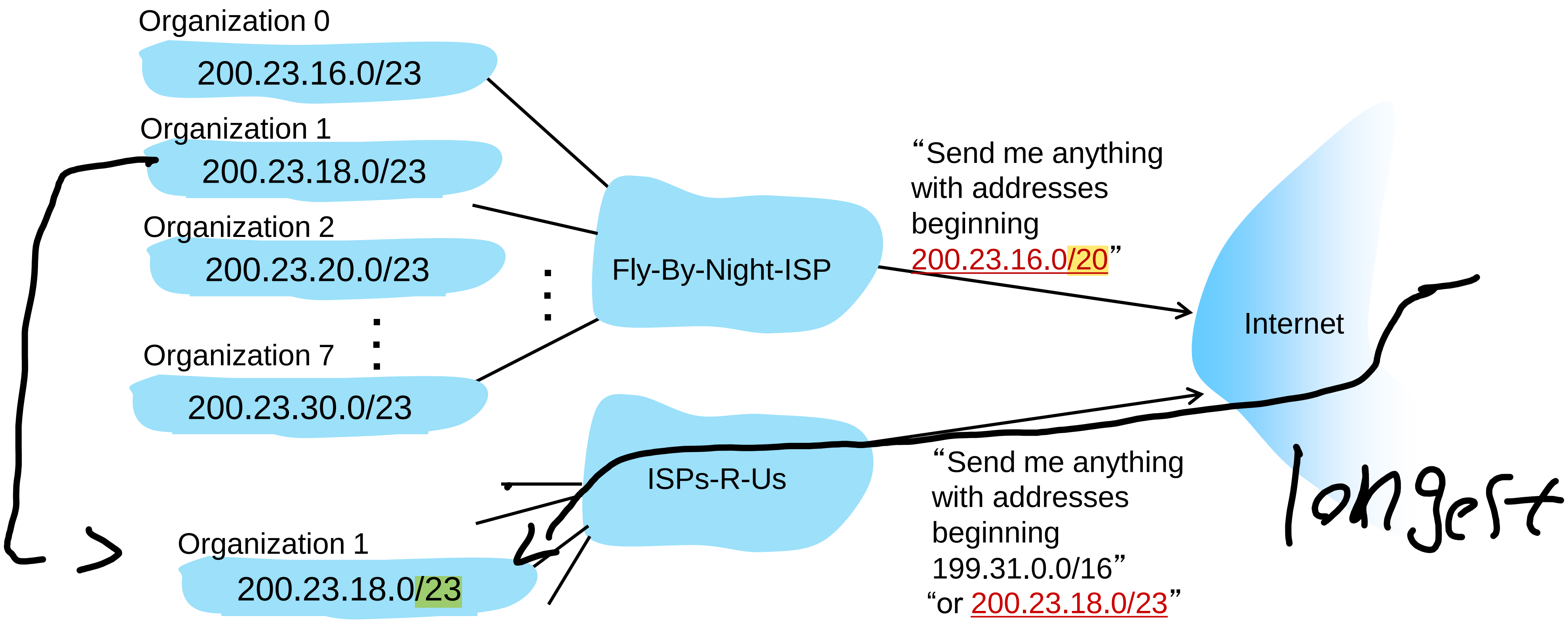
hierarchical addressing allows efficient advertisement of routing information:

- Hierarchical addressing: addresses are given to ISPs in chunks (continuous set of IP address range) ex. All addresses in a /20 block
  - Then, ISP breaks the big chunk into multiple chunks ex. say /23
  - These chunks can be further broken into smaller chunks ex. /25 and less
  - And so on.
  - All the smaller addresses are part of the original ISP chunk.

- Route aggregation: the ability to use a single prefix to advertise multiple networks is often referred to as address aggregation or route aggregation.

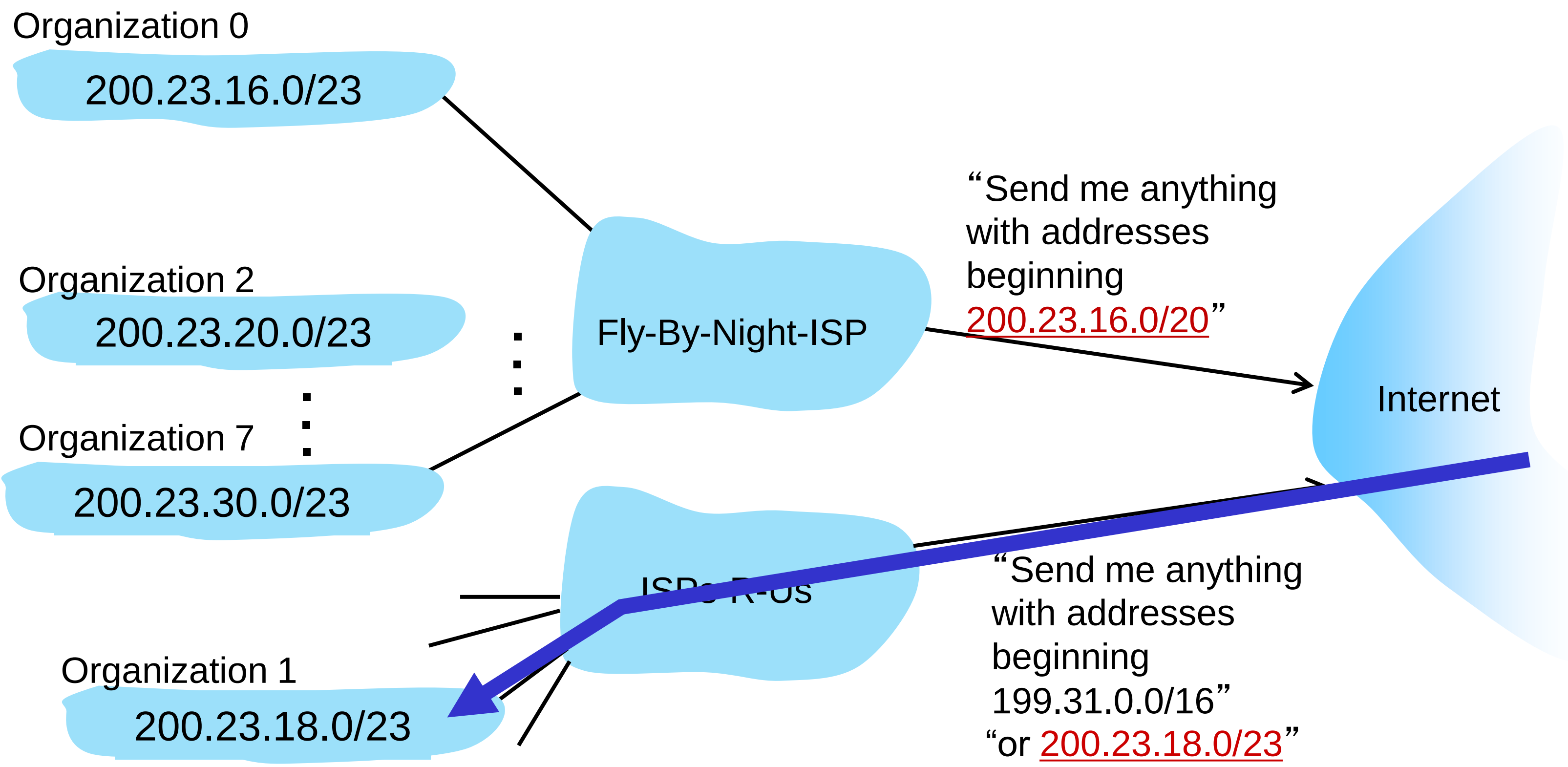
Hierarchical addressing: more specific routes

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



Hierarchical addressing: more specific routes

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



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 TLD (.com, .edu , …) management

Q: are there enough 32-bit IP addresses?

- There are ($2^{32} > 4.3$) billion address
- ICANN allocated last chunk of IPv4 addresses to RRs in 2011
- NAT (next) helps IPv4 address space exhaustion
- IPv6 has 128-bit address space

"Who the hell knew how much address space we needed?" Vint Cerf (reflecting on decision to make IPv4 address 32 bits long)

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

Private IP addresses

The private IP address ranges follow:

Hosts

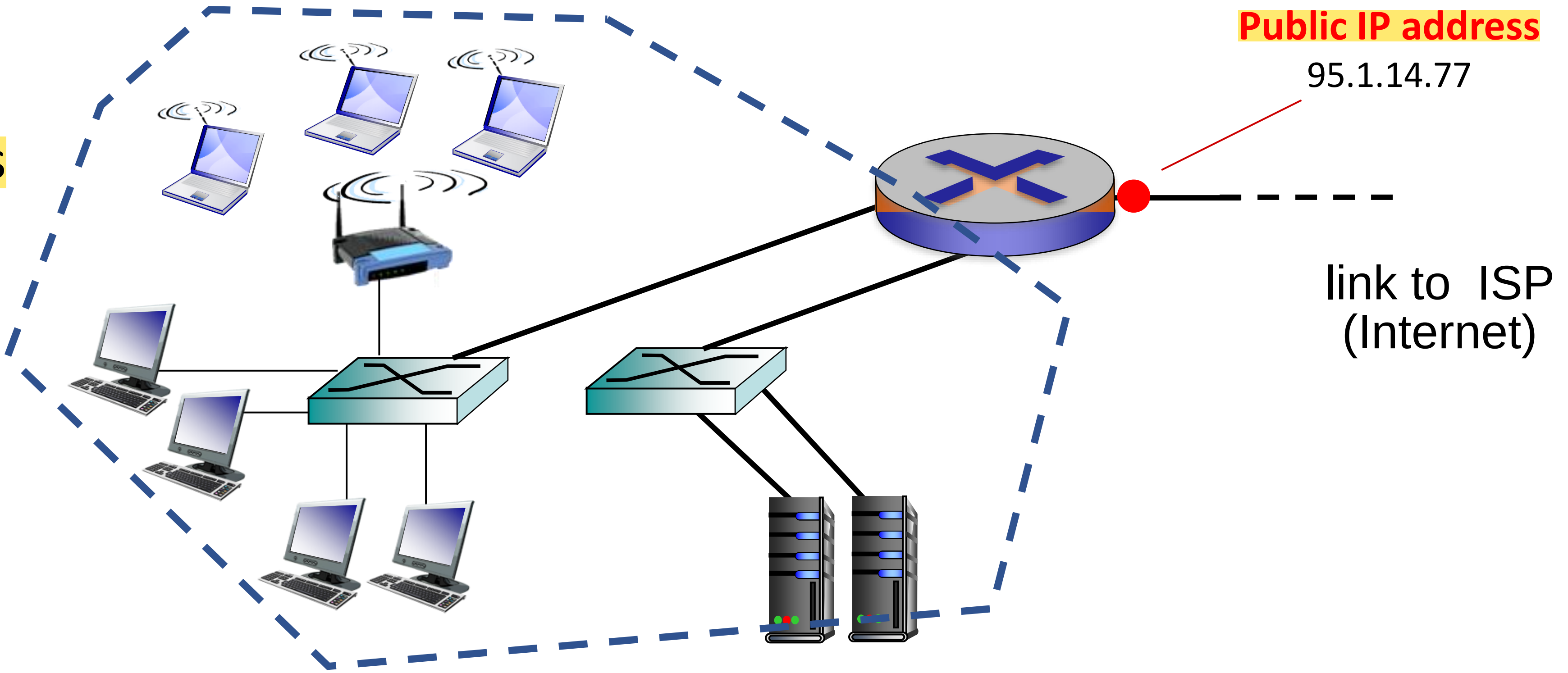
- **Class A**: 10.0.0.0 – 10.255.255.255 — $2^{32-8} \sim$ 16.8 million
  - 10.0.0.0/8, or just 10/8
- **Class B**: 172.16.0.0 – 172.31.255.255 — $2^{32-12} \sim$ 1.05 million
  - 172.16.0.0/12, or just 172.16/12
- **Class C**: 192.168.0.0 – 192.168.255.255 — 2^{32-16} =65,536
  - 192.168.0.0/16, or just 192.168/16

- Private IP addresses are not routable outside the local network (they cannot be advertised to the public Internet).
- They are widely used on almost all local networks today.
- Private addresses are usually translated with NAT at an edge router to map the private addresses used on a LAN to the public address space used by the ISP.

Local Network (Home, Company, or university, etc)

Private IP addresses

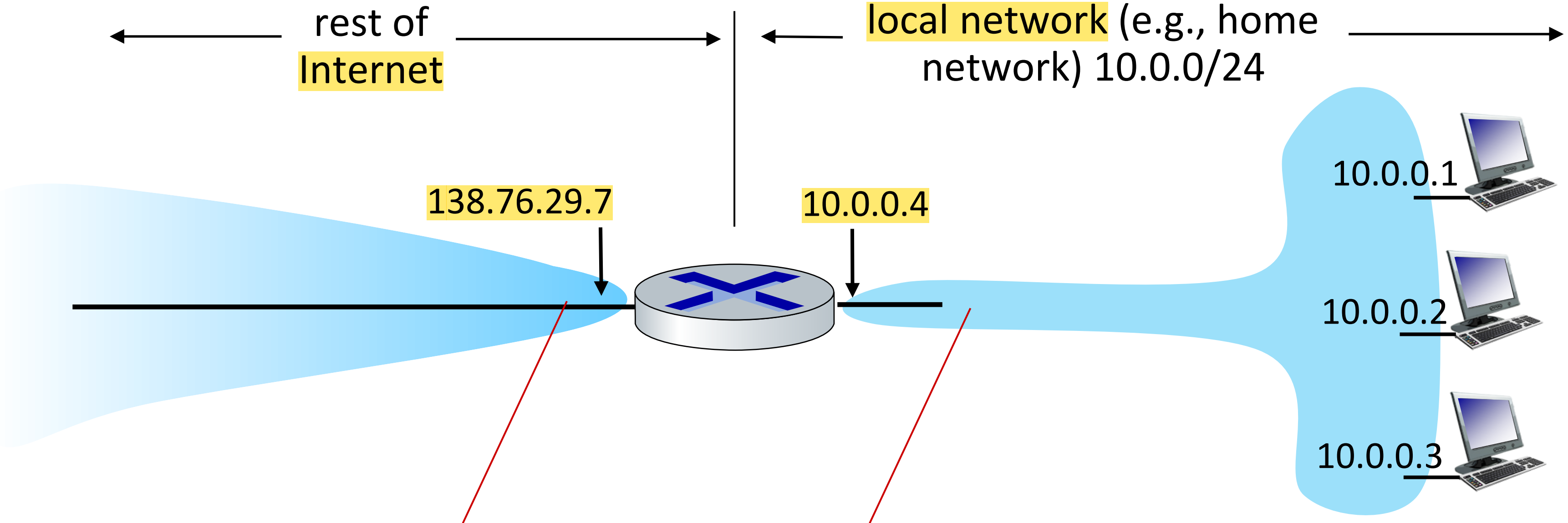
10/8,
172.16/12,
or 192.168/16



- Private IP addresses are not routable outside the local network (they cannot be advertised to the public Internet).
- They are widely used on almost all local networks today.
- Private addresses are usually translated with NAT at an edge router to map the private addresses used on a LAN to the public address space used by the ISP.

NAT: network address translation

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



all datagrams *leaving* local 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 10.0.0/24 address for source, destination (as usual)

NAT: network address translation

- all devices in local network have 32-bit addresses in a "private" IP address space (10/8, 172.16/12, 192.168/16 prefixes) that can only be used in local network
- advantages:
  1 - just one IP address needed from provider ISP for *all* devices
  2 - can change addresses of host in local network without notifying outside world
  3 - can change ISP without changing addresses of devices in local network
  4 - security: devices inside local net not directly addressable, visible by outside world

NAT: network address translation

implementation: NAT router must (transparently):

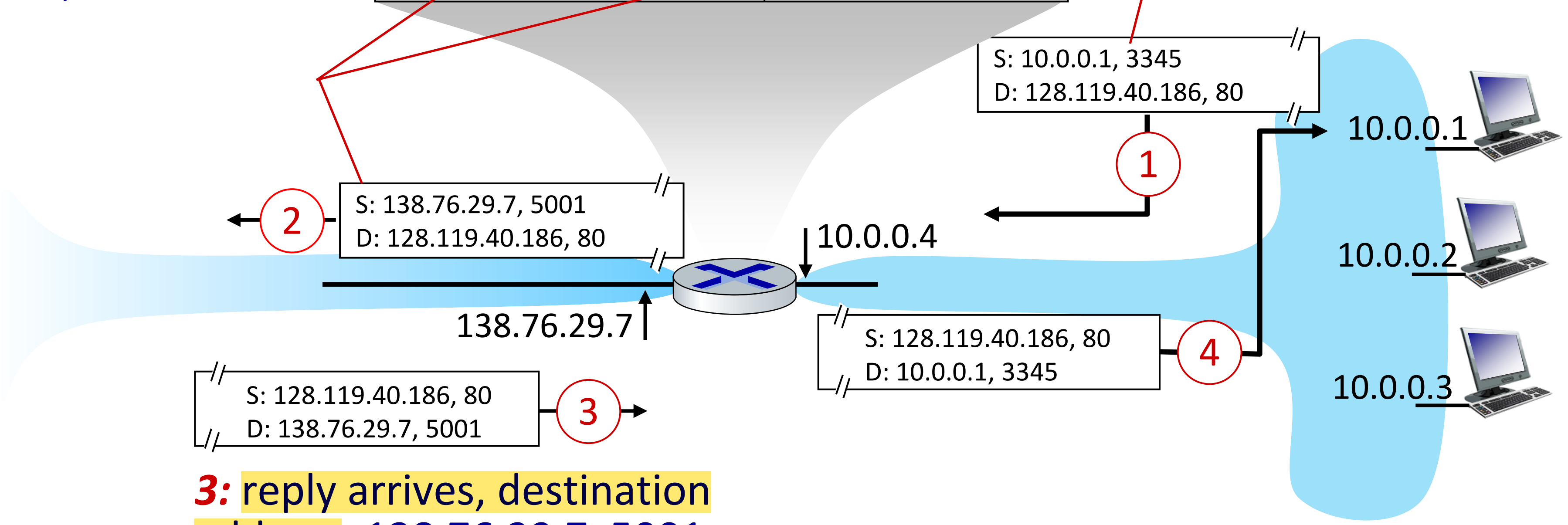
- outgoing datagrams: replace (source IP address, socket port #) of every outgoing datagram to (NAT IP address, new port #)
  - remote clients/servers will respond using (NAT IP address, new port #) as destination address
- remember (in NAT translation table) every (source IP address, port #) to (NAT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

2: NAT router changes datagram source address from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

NAT translation table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80



3: reply arrives, destination address: 138.76.29.7, 5001

NAT: network address translation

- NAT has been controversial:
  - routers "should" only process up to layer 3 — socket port # → transport layer
  - address "shortage" should be solved by IPv6 — 2^{128} → 3.4×10^{38}
  - violates end-to-end argument (port # manipulation by network-layer device)
  - 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 cellular nets

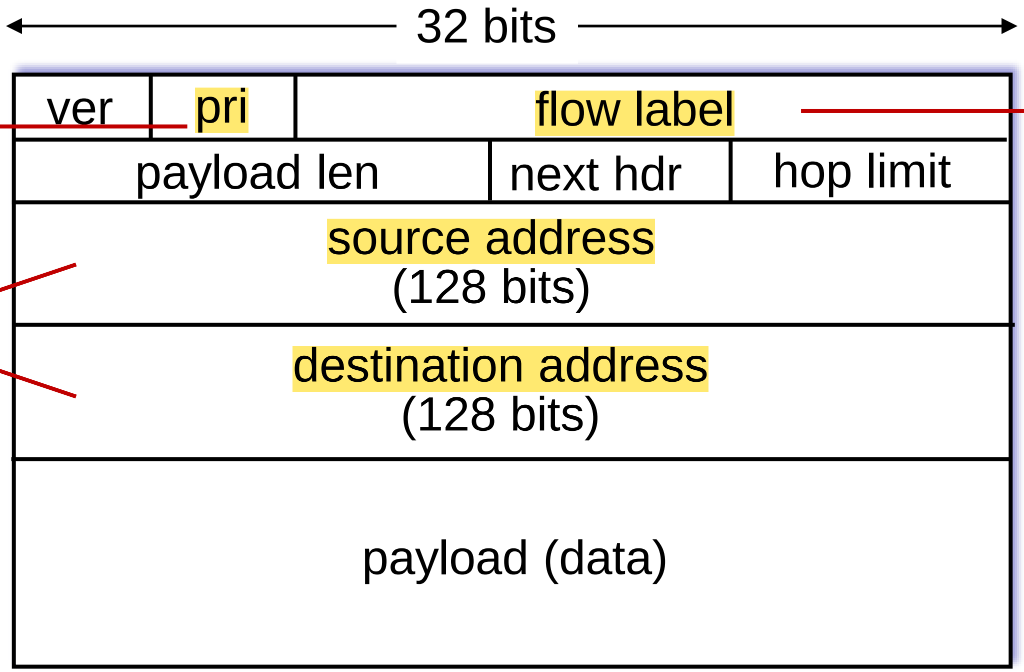
IPv6: motivation

- initial motivation: 32-bit IPv4 address space would be completely allocated
- additional motivation:
  1. • speed processing/forwarding: 40-byte fixed length header
  2. • Enable different network-layer treatment of “flows”

IPv6 datagram format

priority: identify priority among datagrams in flow

128-bit IPv6 addresses



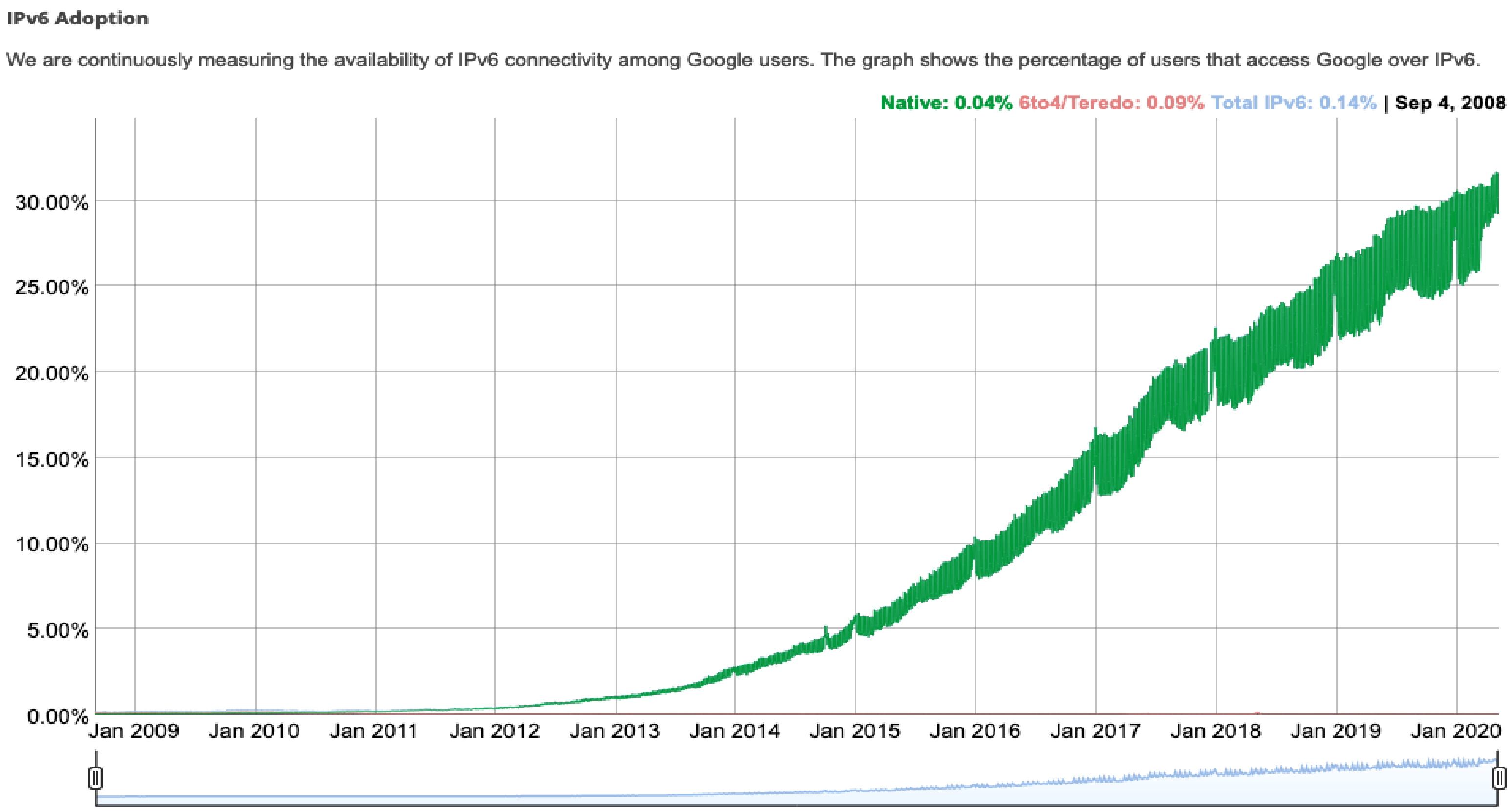
flow label: identify datagrams in same "flow." (concept of “flow” not well defined).

What’s missing (compared with IPv4):

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

IPv6: adoption

- Google[1]: ~ 30% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable



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

IPv6: adoption

- Google[1]: ~ 30% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable
- Long (long!) time for deployment, use
  - 25 years and counting!
  - think of application-level changes in last 25 years: WWW, social media, streaming media, gaming, telepresence, …
  - *Why?*

[1] https://www.google.com/intl/en/ipv6/statistics.html

Revision

Additional IP addressing problems

Additional Problems

IP addressing

IP addressing

x ⟷ 32-x

aaaaaaaa bbbbbbbb cccccccc dddddddd

a.b.c.d/x Number of hosts = 2^{32-x}

a.b.c.d/20	**4096**	**a.b.c.d/26**	**64**
a.b.c.d/21	**2048**	**a.b.c.d/27**	**32**
a.b.c.d/22	**1024**	**a.b.c.d/28**	**16**
a.b.c.d/23	**512**	**a.b.c.d/29**	**8**
a.b.c.d/24	**256**	**a.b.c.d/30**	**4**
a.b.c.d/25	**128**	**a.b.c.d/31**	**2**

Oct format

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

=255

Usually, first and last IP addresses are reserved for network address (or subnet) and broadcast address

IP addressing

- **Network address** - the address by which we refer to the network.
  - the first IP address in it which all host part bits = 0
  - Network Addresses have all 0's in the host portion.
- **Broadcast address** - A special address used to send data to all hosts in the network
  - the last IP address in the network which all host part bits = 1
  - Broadcast Addresses have all 1's in the host portion.
- **Host addresses** - The addresses assigned to the end devices in the network
  - Host Addresses can not have all 0's or all 1's in the host portion.

example:

223.1.1.0/24

11011111 00000001 00000001 00000000

Number of IP addresses: $2^{32-24} = 2^8 = 256$ IP address

Network address = 223.1.1.0

broadcast = 223.1.1.255

Hosts take the remaining addresses from 223.1.1.1 to 223.1.1.254 → 256-2 = 254 hosts

IP addressing

- Which IP address should be assigned to PC B ?

a) 192.168.$\boxed{15}$.5 ✗ 3^{ed} byte “15”

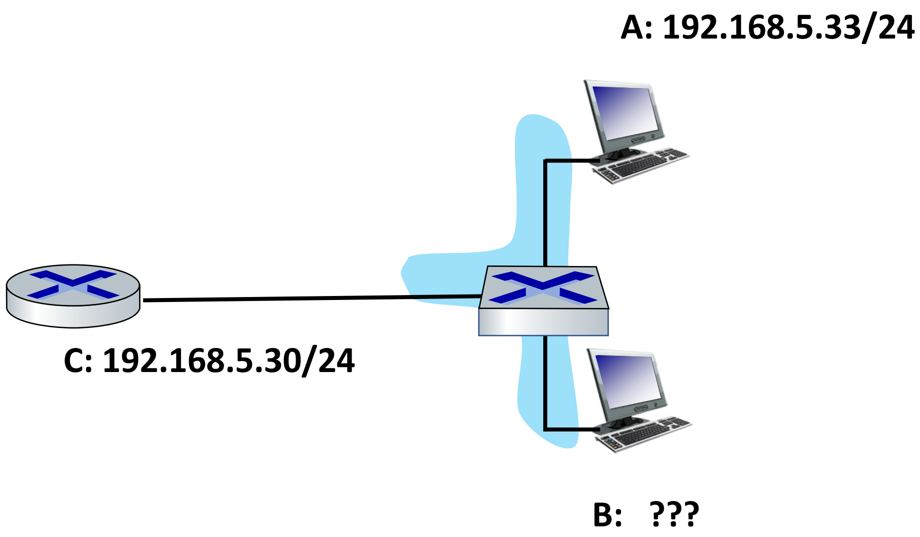
b) 192.168.5.0 ✗ subnet address

c) 192.168.5.40 ✓

d) 192.168.5.255 ✗ broadcast

e) 192.168.5.256 ✗ error

?



IP addressing

- Which IP address should be assigned to PC B ?

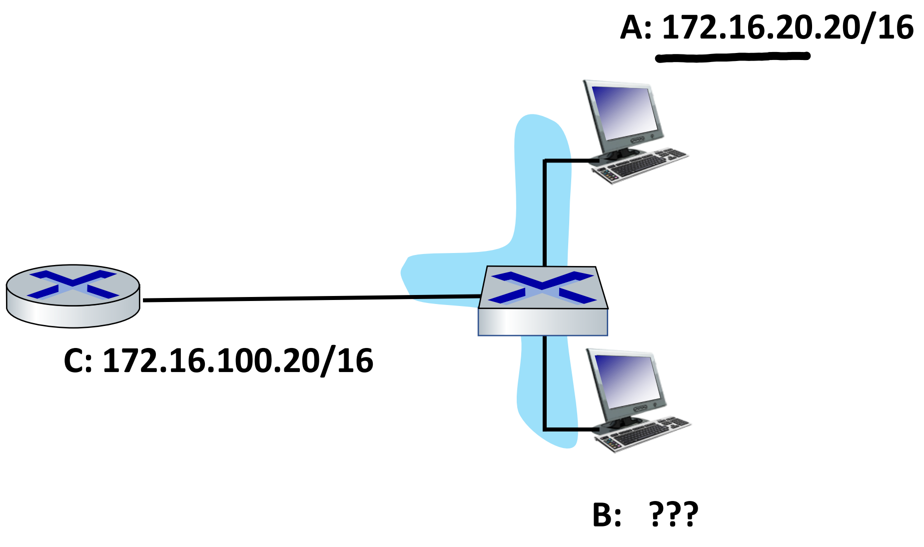
a) 172.16.255.255 ✗ broadcast

b) 172.16.0.255 ✓

c) 172.168.20.21 ✗ another subnet

d) 172.16.254.255 ✓

e) 172.16.0.0 ✗ subnet address



IP addressing

- Which IP address should be assigned to PC B ?

a) 192.168.5.5 ×

b) 192.168.5.0 × sub

c) 192.168.5.66 ×

d) 192.168.5.255 ×

e) 192.168.5.50

We can't tell !!

Need to convert to binary

192. 168. 5. 60 /28

11000000 . 10101000 . 00000101 . 00111100

Network

11000000 . 10101000 . 00000101 . 0011 0000

192.168.5.48/28

Broadcast

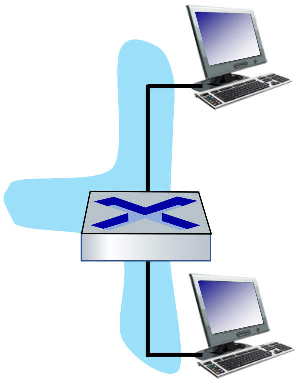
11000000 . 10101000 . 00000101 . 0011 1111

192.168.5.63/28

Host addresses = any IP between these: 192.168.5.48 to 192.168.5.63

8 8 8 4

A: 192.168.5.60/28



B: ???

IP addressing

- Which IP address should be assigned to PC B ?

a) 192.168.5.5 ×

b) 192.168.5.0 sub

c) 192.168.5.62

d) 192.168.5.255

e) 192.168.5.50 ×

We can't tell !!

Need to convert to binary

192. 168. 5. 61 /30

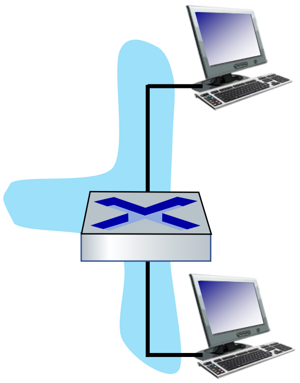
11000000 . 10101000 . 00000101 . 00111101

Network
11000000 . 10101000 . 00000101 . 00111100
192.168.5.60/30

Broadcast
11000000 . 10101000 . 00000101 . 001111 11
192.168.5.63/30

Host addresses = any IP between these: 192.168.5.60 to 192.168.5.63

A: 192.168.5.61/30



B: ???

IP addresses: how to get one?

Q: how does *network* get subnet part of IP address?

A: gets allocated portion of its provider ISP's address space

ISP's block	11001000 00010111 00010000 00000000		200.23.16.0/20

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

Organization 0	11001000 00010111 0001**000**0	00000000	200.23.16.0/23
Organization 1	11001000 00010111 0001**001**0	00000000	200.23.18.0/23
Organization 2	11001000 00010111 0001**010**0	00000000	200.23.20.0/23
...	....	....	
Organization 7	11001000 00010111 0001**111**0	00000000	200.23.30.0/23

IP addresses: how to get one?

ISP's block	**11001000 00010111 00010000 00000000**	**200.23.16.0/20**

What if ISP want to divide its address space into 4 blocks:

Organization 0	11001000 00010111 0001**00**00 00000000	200.23.16.0/22
Organization 1	11001000 00010111 0001**01**00 00000000	200.23.20.0/22
Organization 2	11001000 00010111 0001**10**00 00000000	200.23.24.0/22
Organization 3	11001000 00010111 0001**11**00 00000000	200.23.28.0/22

IP addresses: how to get one?

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

What if ISP want to divide its address space into 16 blocks:

Organization 0	<u>11001000 00010111 0001</u>	**0000**	00000000	200.23.16.0/24
Organization 1	<u>11001000 00010111 0001</u>	**0001**	00000000	200.23.17.0/24
Organization 2	<u>11001000 00010111 0001</u>	**0010**	00000000	200.23.18.0/24
…	…			….
…	…			….
Organization16	<u>11001000 00010111 0001</u>	**1111**	00000000	200.23.31.0/24

Consider a router that interconnects three subnets: Subnet 1, Subnet 2, and Subnet 3.

Suppose all the three subnets are required to have the prefix **223.1.17/24**.

Suppose that Subnet 1 is required to support at least 60 interfaces,

Subnet 2 is to support at least 90 interfaces, and

Subnet 3 is to support at least 12 interfaces.

Provide three network addresses (of the form a.b.c.d/x)

that satisfy these constraints.

/mask	# host
a.b.c.d/20	4096
a.b.c.d/21	2048
a.b.c.d/22	1024
a.b.c.d/23	512
a.b.c.d/24	256
a.b.c.d/25	128
a.b.c.d/26	64
a.b.c.d/27	32
a.b.c.d/28	16
a.b.c.d/29	8
a.b.c.d/30	4
a.b.c.d/31	2

223.1.17/24 → **256**

Subnet 1 > 60 interfaces → **Closest is 64 → mask /26** 2^{32-26}

Subnet 2 > 90 interfaces → **Closest is 128 → mask /25** 2^{32-25}

Subnet 3 > 12 interfaces → **Closest is 16 → mask /28** 2^{32-28}

Continue….

223.1.17/24

11011111 00000001 00010001 xxxxxxxx

Oct format

x x x x x x x x

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

128 64 32 16 8 4 2 1

Subnet 1 (/26) 11011111 00000001 00010001 00 xxxxxx — 223.1.17.0 /26

00 000000 — 223.1.17.0 নেট

00 111111 — 223.1.17.63 ব্রড

Subnet 2 (/25) 11011111 00000001 00010001 1 xxxxxxx — 223.1.17.128 /25

1 0000000 — 223.1.17.128 নেট

1 1111111 — 223.1.17.255 ব্রড

Subnet 3 (/28) 11011111 00000001 00010001 0100 xxxx — 223.1.17.64 /28

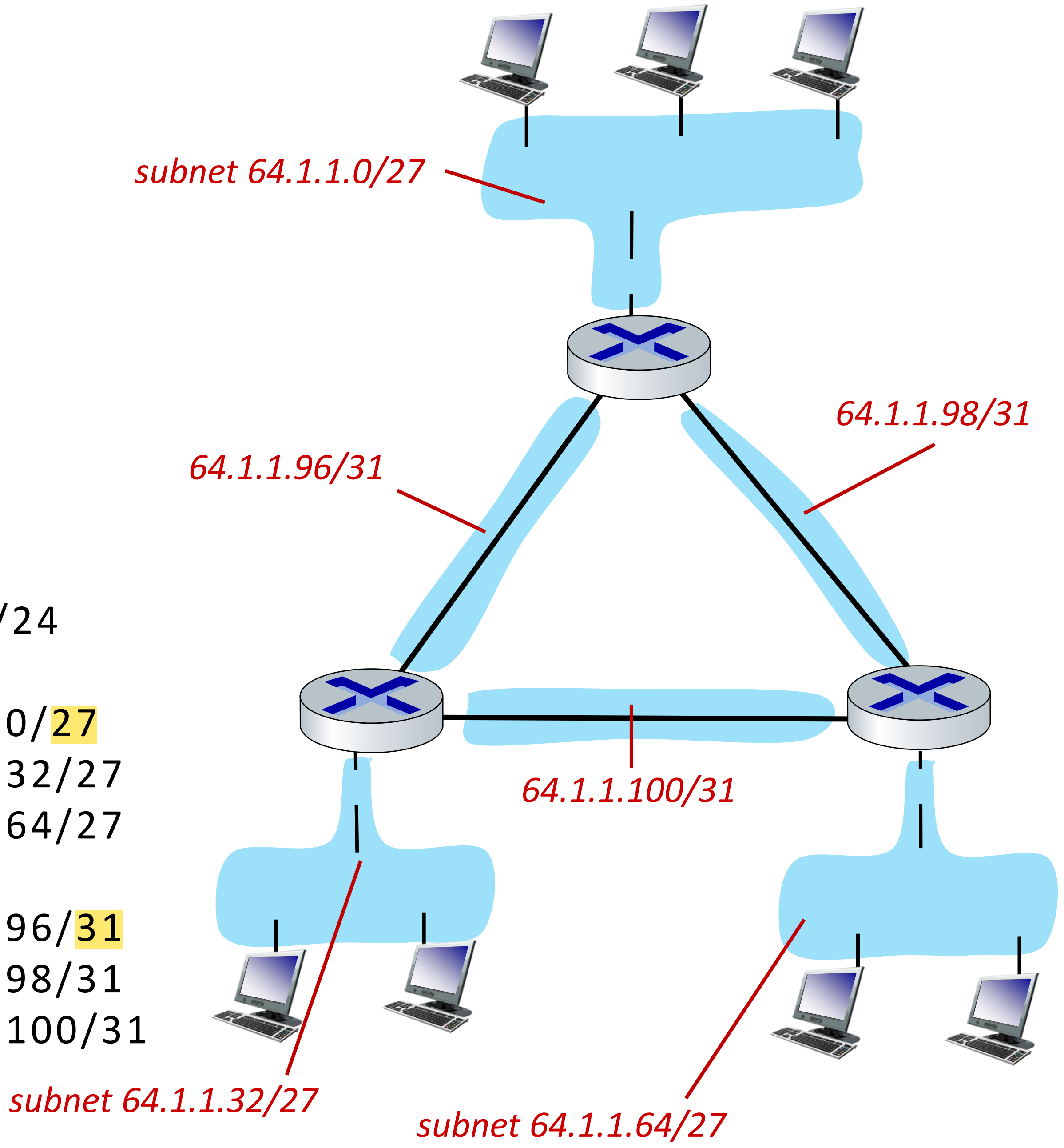
or, 0110 xxxx — 223.1.17.96 /28

or, 0111 xxxx — 223.1.17. /28
.112

Subnetting

- Use this block: 64.1.1/24
- Each subnet should have 32 IP addresses
- Subnets between routers → 2 IP addresses
  - *$2^{32-27} = 2^5$ = 32 host addresses*
  - *2^{32-31} = 2 host addresses*

01000000 00000001 00000001 xxxxxxxx	64.1.1/24
01000000 00000001 00000001 **000**xxxxx	64.1.1.0/27
01000000 00000001 00000001 **001**xxxxx	64.1.1.32/27
01000000 00000001 00000001 **010**xxxxx	64.1.1.64/27
01000000 00000001 00000001 **0110000**x	64.1.1.96/31
01000000 00000001 00000001 **0110001**x	64.1.1.98/31
01000000 00000001 00000001 **0110010**x	64.1.1.100/31



Forwarding decision

Q: Given this forwarding table, computer the forwarding decision for the following:

16.16.20.28	00010000 00010000 00010100 00011100	1
16.16.4.28	00010000 00010000 0000 0100 00011100	2
16.100.100.234	00010000 01100100 01100100 11101010	5

Forwarding Table

Dest. Address range	Output interface
16.16.0.0/16	1
16.16.0.0/20	2
64.1.0.0/20	3
64.1.1.0/24	4
16 /8	5

Preparation for answer	**1st step: convert to binary (easy way)**			
128 64 32 16 8 4 2 1	128 64 32 16 8 4 2 1	128 64 32 16 8 4 2 1		
00010000	00010000			1
00010000	00010000	0000 xxxx		2
01000000	00000001	0000 xxxx		3
01000000	00000001	0000 0001		4
00010000				5

Alternative methods to subnetting

If you like it, you can use it.

Alternative approach to Subnetting

Borrow 1 bit from the host portion of the address.

Original	192.	168.	1.	0	000	0000
Mask	255.	255.	255.	0	000	0000

1 Network

The borrowed bit value is **0** for the Net 0 address.

Net 0	192.	168.	1.	0	000	0000

The borrowed bit value is **1** for the Net 1 address.

Net 1	192.	168.	1.	1	000	0000

2 Subnets

The new subnets have the **SAME** subnet mask.

Mask	255.	255.	255.	1	000	0000

Decimal Representation

Original	192.	168.	1.	0	000	0000	Network: 192.168.1.0/24
Mask	255.	255.	255.	0	000	0000	Mask: 255.255.255.0

Borrowing 1 bit creates 2 subnets with the same mask.

Net 0	192.	168.	1.	0	000	0000	Network: 192.168.1.0/25
Mask	255.	255.	255.	1	000	0000	Mask: 255.255.255.128
Net 1	192.	168.	1.	1	000	0000	Network: 192.168.1.128/25
Mask	255.	255.	255.	1	000	0000	Mask: 255.255.255.128

IP addressing

- The prefix and the subnet mask are different ways of representing the same thing - the network portion of an address.

Network and Host Portions of an IP Address

IP Address	172 .	16 .	4 .	1
	10101100	00010000	00000100	00000001
Subnet Mask	255 .	255 .	255 .	0
	111111111	1111111111	111111111	00000000
	Prefix /24 (24 high order bits)			

Alternative approach to Subnetting

Example

Divide network 192.168.1.0/24 into 4 subnets

Solution :

Old mask /24 → 11111111.11111111.11111111.00000000
255 . 255 . 255 . 0

4 subnets need 2 bits

- New subnet mask = 255 . 255 . 255 . 192
  11111111.11111111.11111111.**11**000000 **(/26)**

- interesting octet is **192**
- hop count = 256 – 192 = **64**

- **The first subnet is** → **192.168.1.0/26**
- **The second subnet is** → **192.168.1.64/26**
- **The third subnet is** → **192.168.1.128/26**
- **The fourth subnet is** → **192.168.1.192/26**

Alternative approach to Subnetting

Example

Divide network 172.168.0.0/16 into 8 subnets

Solution :

Old mask /16 → 11111111.11111111.00000000.00000000

255 . 255 . 0 . 0

- 8 subnets need 3 bits
- new subnet mask = 255 . 255 . 224 . 0

11111111.11111111.**111**00000.00000000 **(/19)**

- interesting octet is **224**
- hop count = 256 – 224 = **32**

- **The first subnet is → 172.168.0.0/19**
- **The second subnet is → 172.168.32.0/19**
- **The third subnet is → 172.168.64.0/19**

-The 8th subnet is → 172.168.224.0/19

Alternative approach to Subnetting

Example

Divide network 10.0.0.0/10 into 4 subnets

Solution :

Old mask /10 → 11111111.11000000.00000000.00000000

255 . 192 . 0 . 0

4 subnets need 2 bits

- subnet mask = 255 . 240 . 0 . 0

11111111.11**11**1111.11111111.00000000 **(/12)**

- interesting octet is **240**
- hop count = 256 – 240= **16**

- **The first subnet is → 10.0.0.0/12**
- **The second subnet is → 10.16.0.0/12**
- **The third subnet is → 10.32.0.0/12**
- **The fourth subnet is → 10.48.0.0/12**