CS 3800: Computer Networks

Lecture 7: Network Layer

Instructor: John Korah

Acknowledgement

- The following slides include material from author resources for:
  - KR Text book
  - “Data and computer communications,” William Stallings, Tenth edition

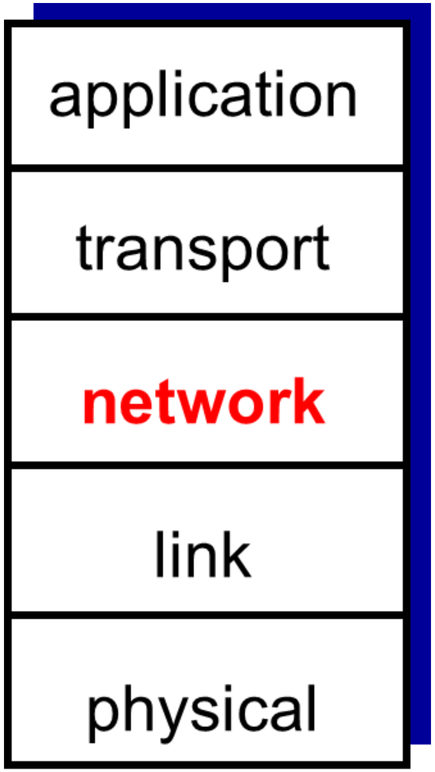
Learning Goals

- Understand principles behind network layer services, focusing on data plane:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - generalized forwarding
- instantiation, implementation in the Internet

Topics

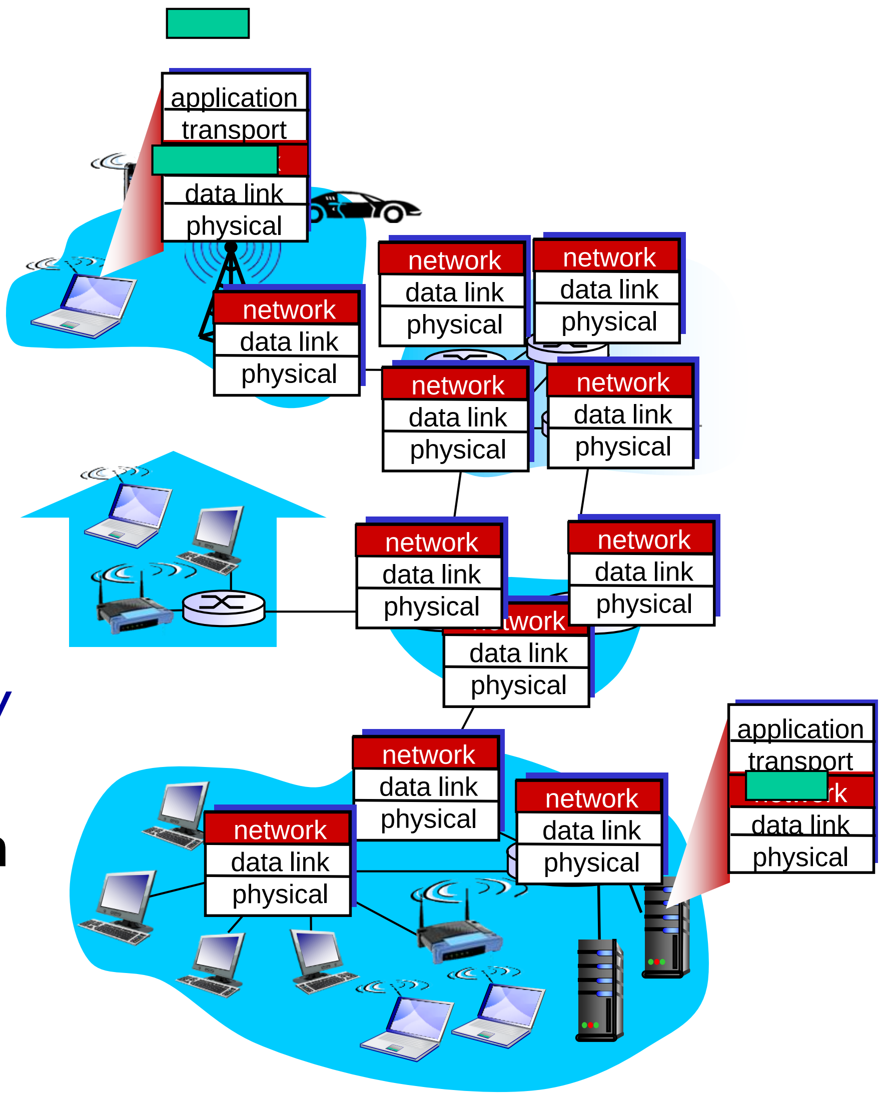
- Overview of Network layer
  - data plane
  - control plane
- What's inside a router
- IP: Internet Protocol
  - datagram format
  - fragmentation
  - IPv4 addressing
  - network address translation
  - IPv6

Recall



Network layer

- Transport segment from sending to receiving host
- Encapsulation: On sending side encapsulates segments into datagrams
- De-encapsulation: on receiving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- router examines header fields in all IP datagrams passing through it



Two key network-layer functions

Network-layer functions:

- *forwarding: ?*
- *routing:* ?

Two key network-layer functions

Network-layer functions:

- *Forwarding:* Move packets from router's input to appropriate router output
- *Routing:* Determine route taken by packets from source to destination
    - *routing algorithms*

Two key network-layer functions

Network-layer functions:

- *forwarding:* move packets from router's input to appropriate router output
- *routing:* determine route taken by packets from source to destination
  - *routing algorithms*

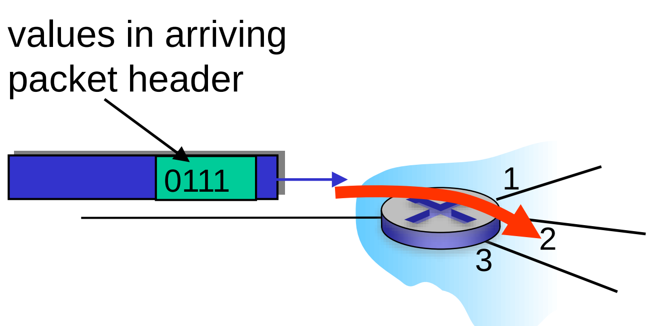
Analogy: taking a trip

- *forwarding:* process of getting through single interchange
- *routing:* process of planning trip from source to destination

Network layer: Data plane, Control plane

Data plane (forwarding)

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

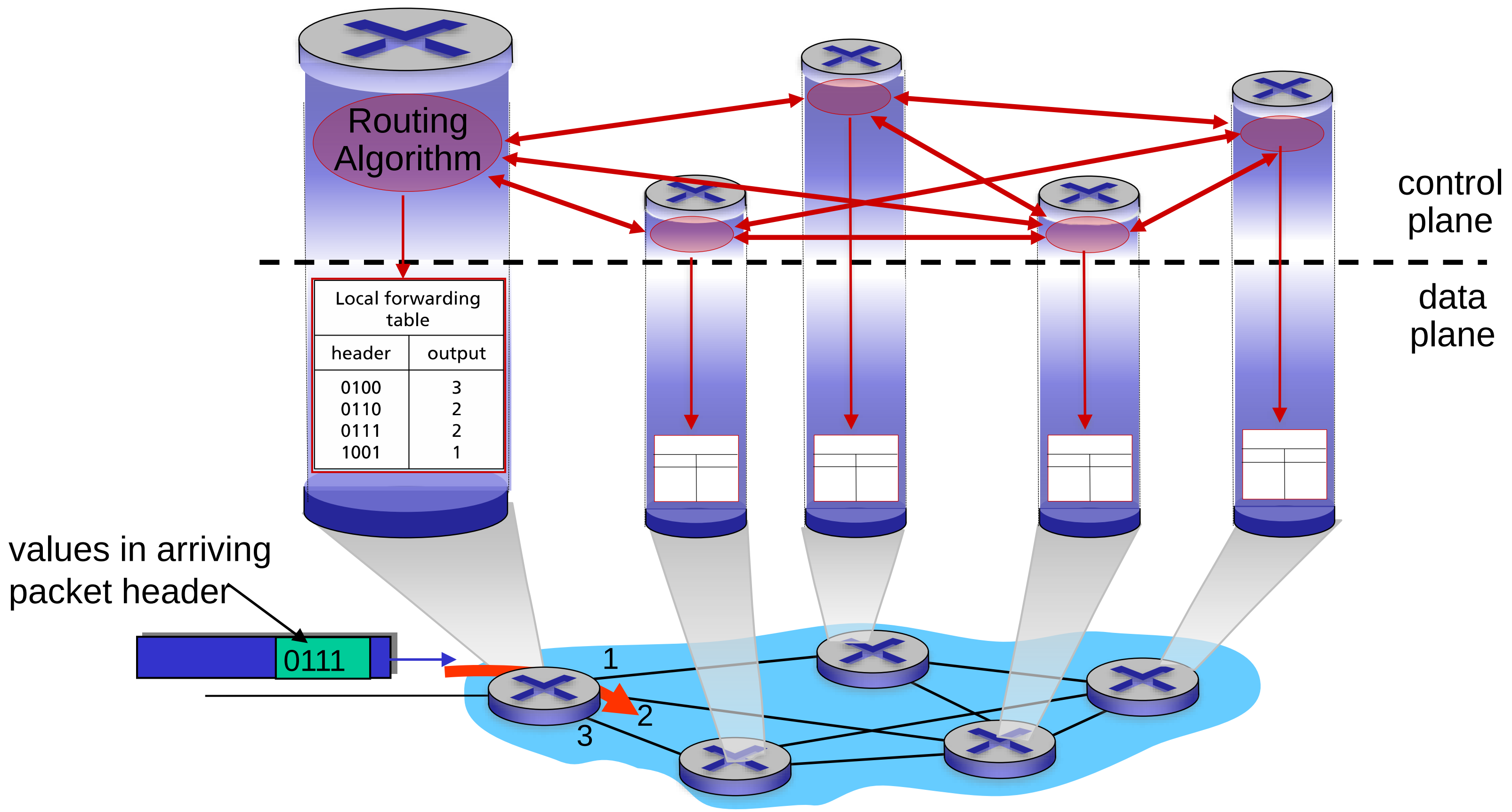


Control plane (routing)

- network-wide logic
- determines how datagram is routed among routers along end-end path from source host to destination host
- two control-plane approaches:
  - *traditional routing algorithms:* implemented in routers
  - *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

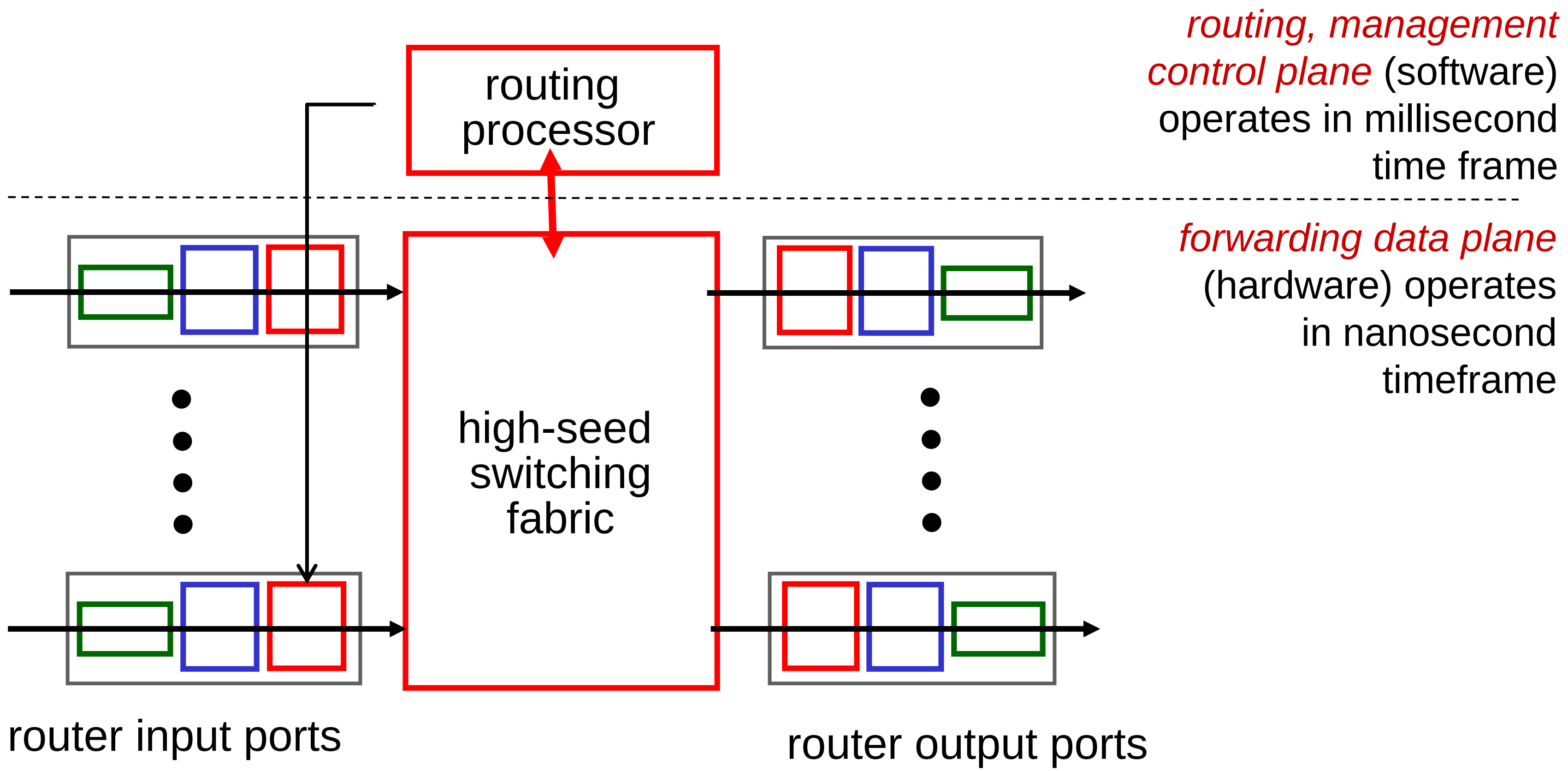


Topics

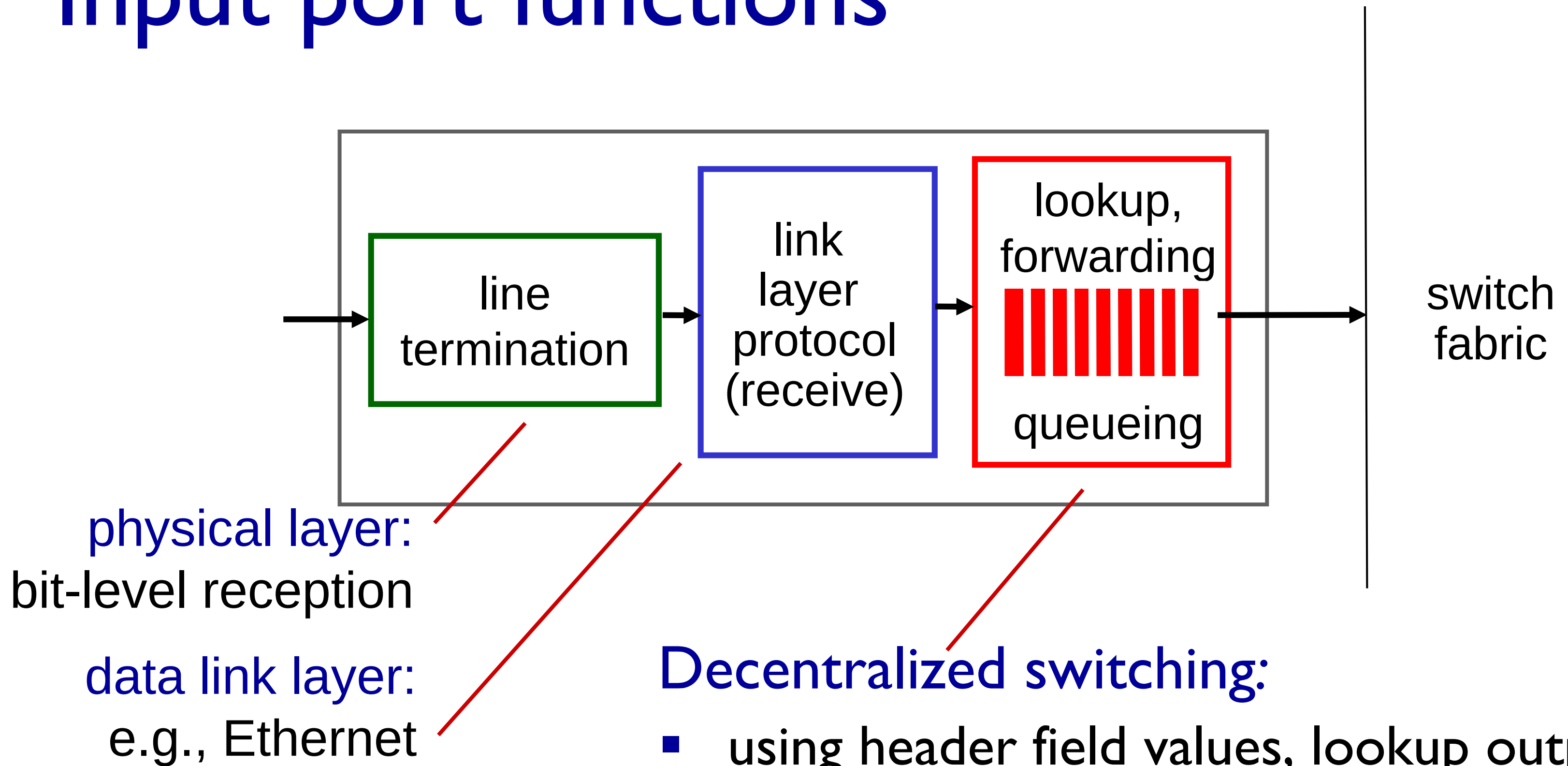
- Overview of Network layer
  - data plane
  - control plane
- What's inside a router
- IP: Internet Protocol
  - datagram format
  - fragmentation
  - IPv4 addressing
  - network address translation
  - IPv6

Router architecture overview

- high-level view of generic router architecture:



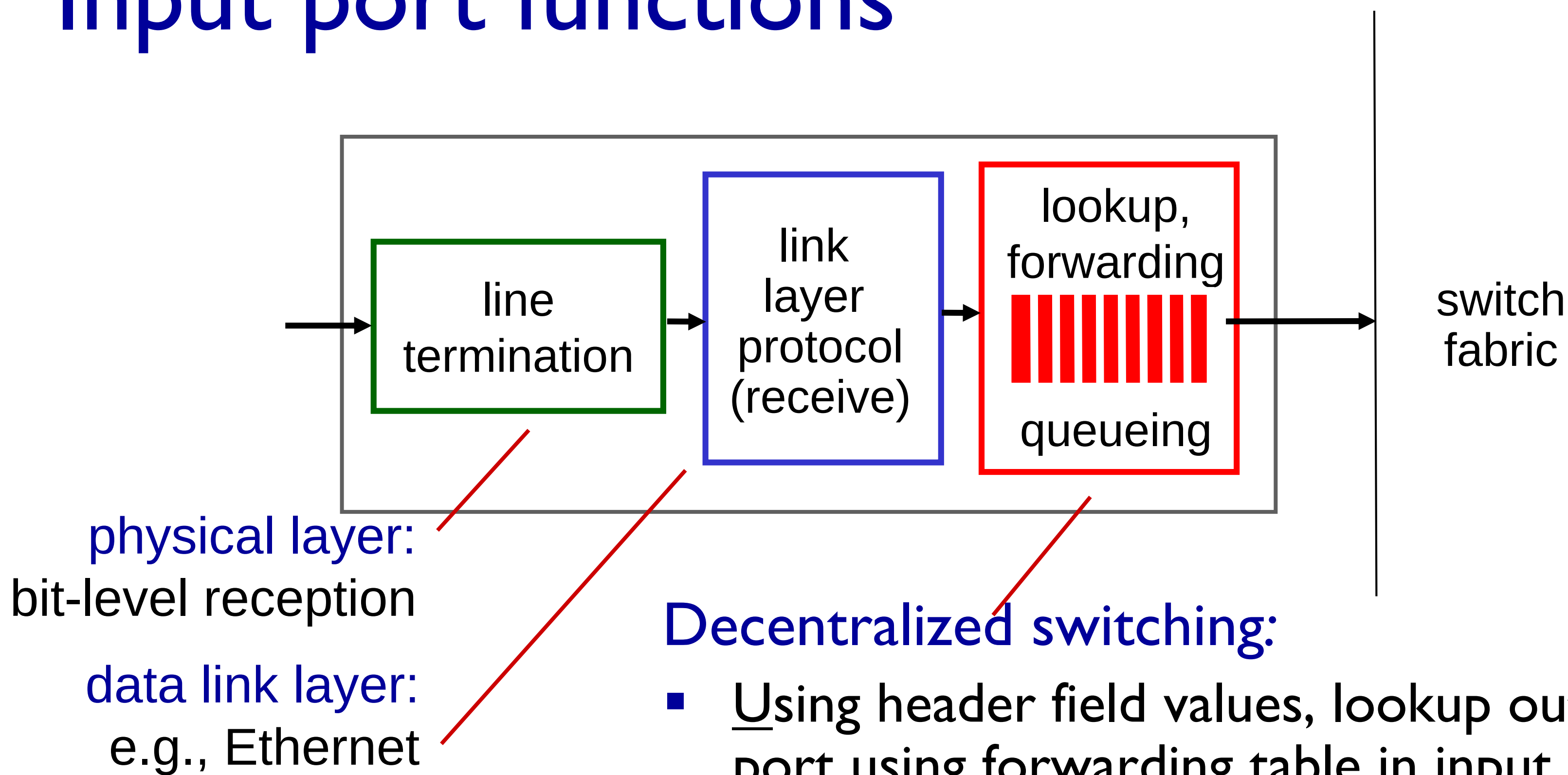
Input port functions



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'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Input port functions



Decentralized switching:

- Using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
- *Destination-based forwarding:* forward based only on destination IP address (traditional)
- *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
`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

Longest prefix matching

longest prefix matching

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

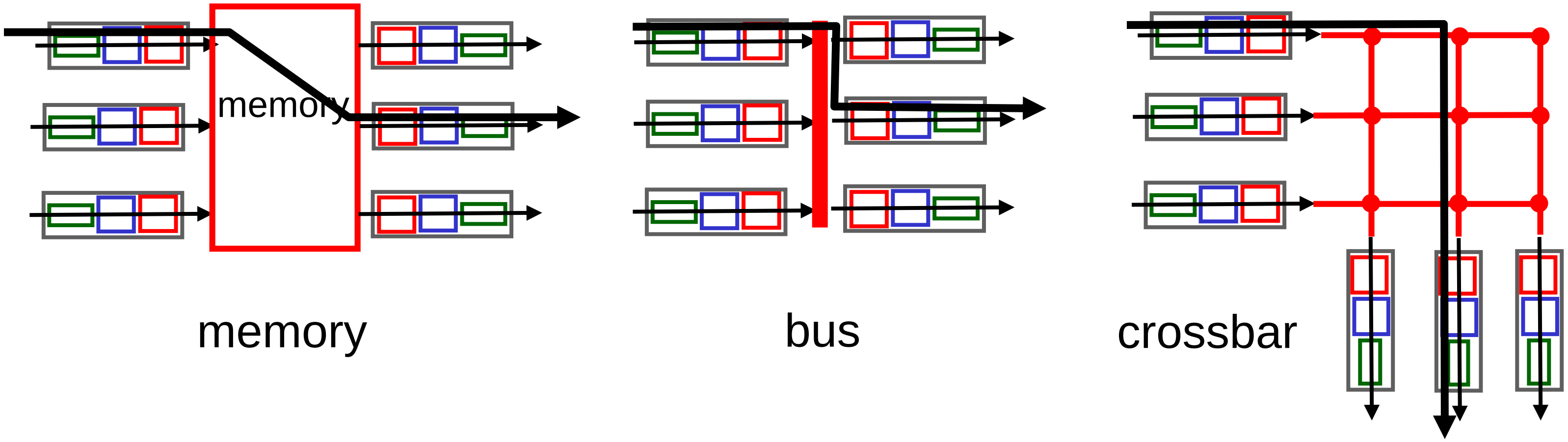
examples:

DA: 11001000 00010111 00010110 10100001 which interface?

DA: 11001000 00010111 00011000 10101010 which interface?

Switching fabrics

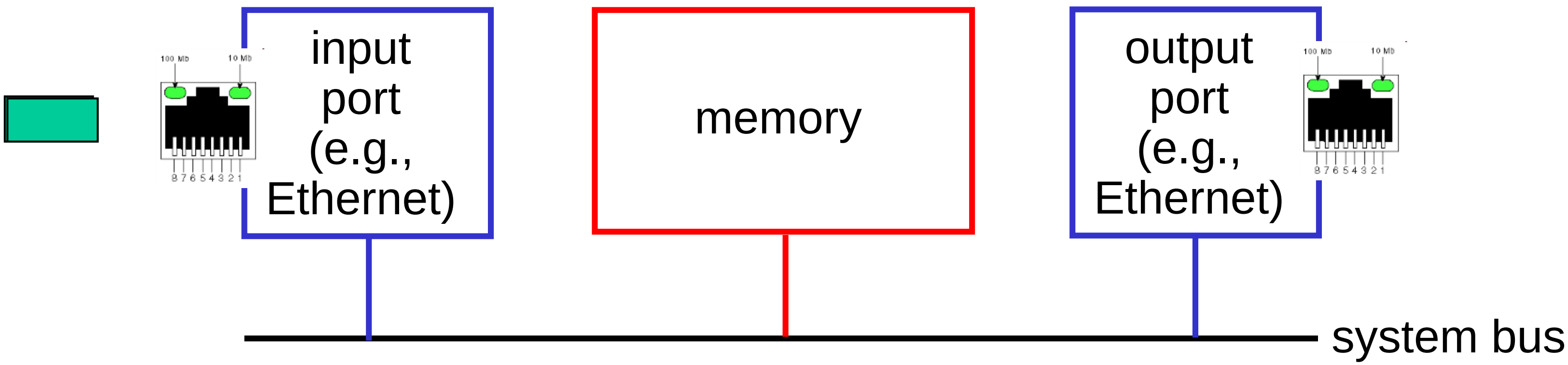
- Transfer packet from input buffer to appropriate output buffer
- Switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- Three types of switching fabrics



Switching via memory

First generation routers:

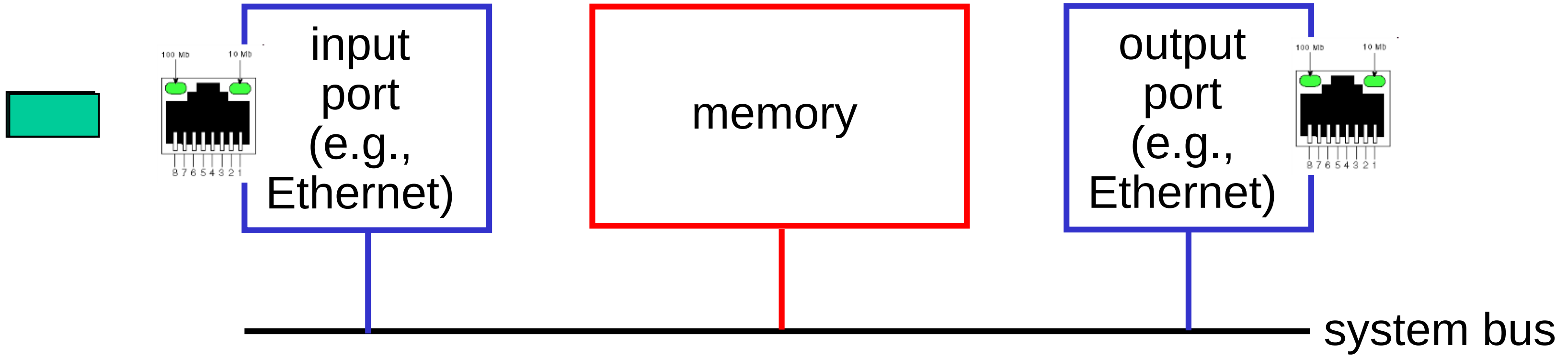
- Traditional computers with switching under direct control of CPU
- Packet copied to system's memory
- Speed limited by memory bandwidth (2 bus crossings per datagram)



Switching via memory

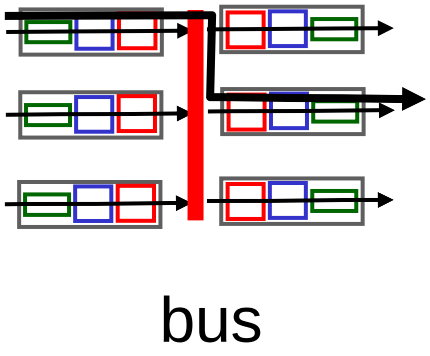
Advantages: ?

Disadvantages: ?



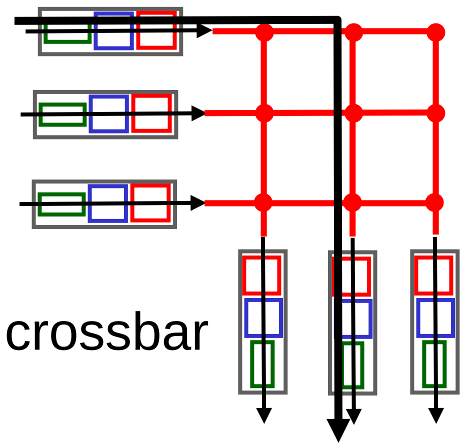
Switching via a bus

- Datagram from input port memory to output port memory via a shared bus
- *Bus contention:* switching speed limited by bus bandwidth
- Advantages:
- Disadvantages:



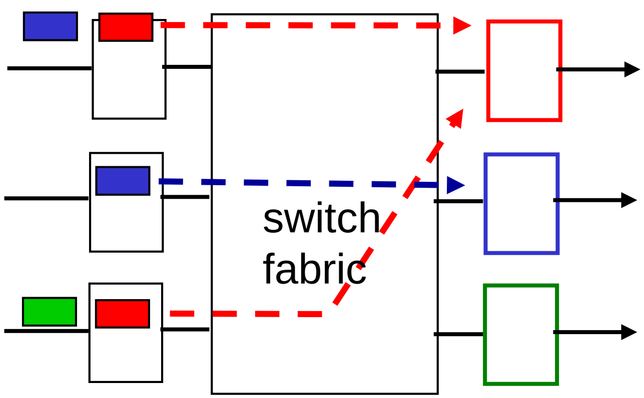
Switching via interconnection network

- Overcome bus bandwidth limitations
- Banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessors
- Advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.

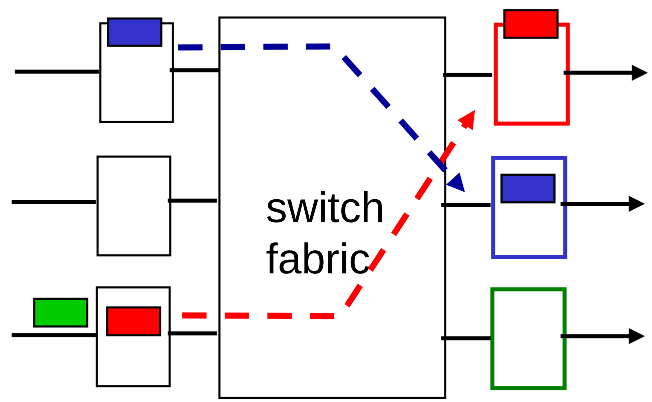


Input port queuing

- 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

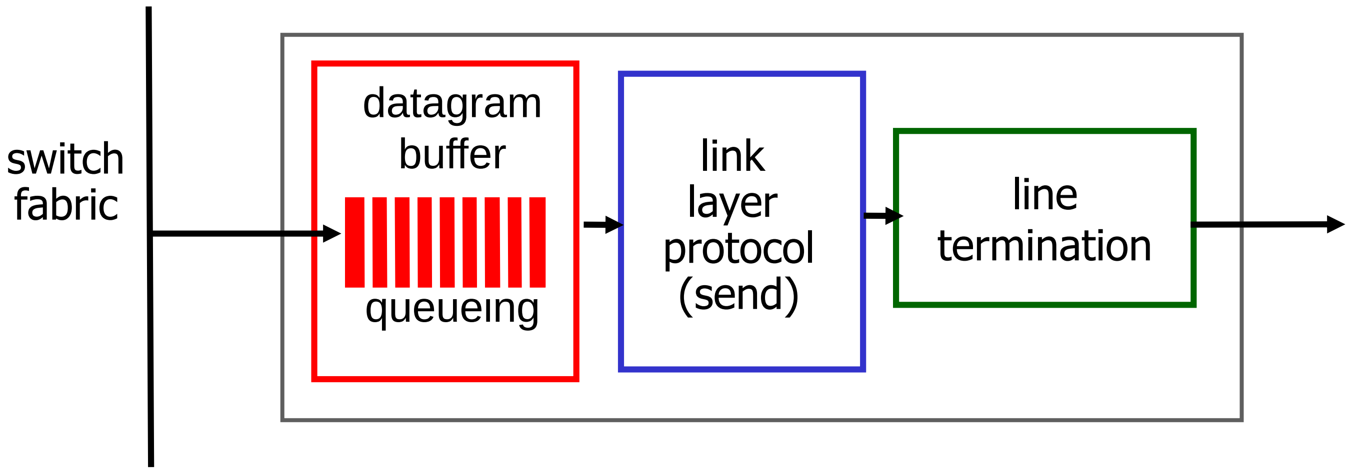


output port contention:
only one red datagram can be transferred.
lower red packet is blocked



one packet time later:
green packet experiences HOL blocking

Output ports

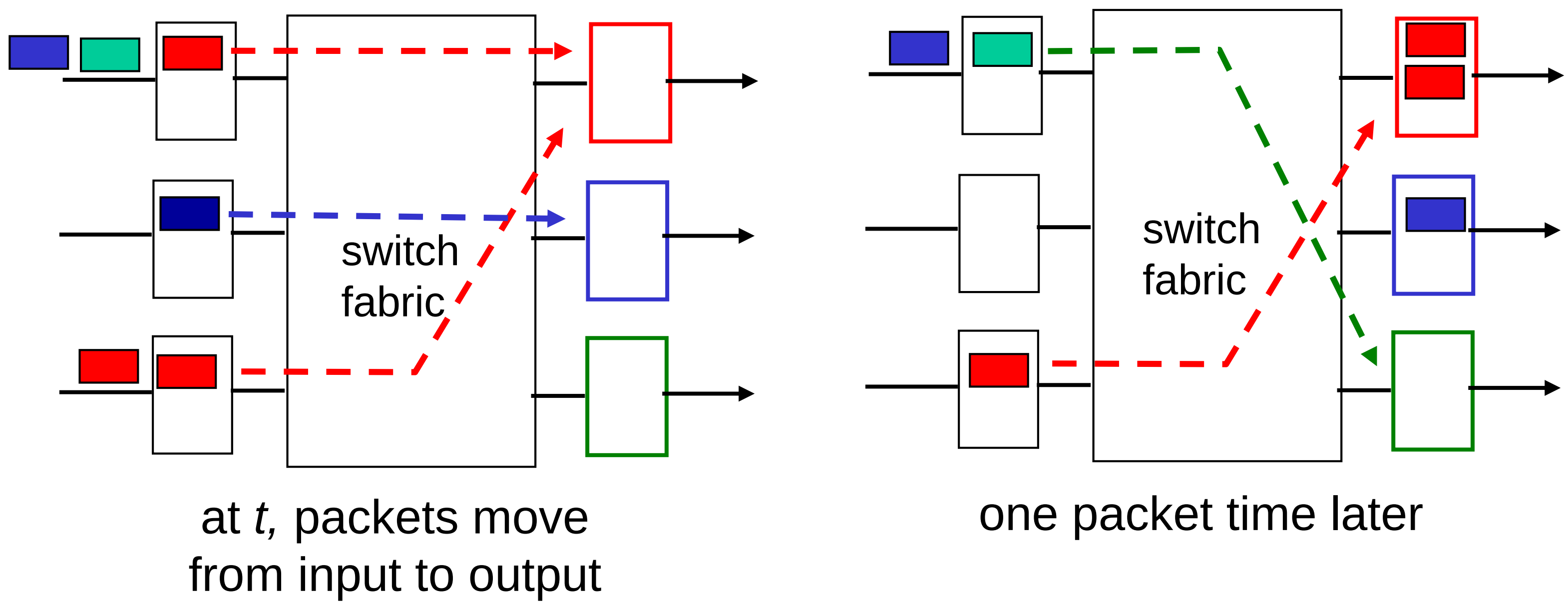


- *buffering* required wh… fabric faster than the…
- *scheduling discipline* chooses among queued datagrams for transmission

Datagram (packets) can be lost due to congestion, lack of buffers

Priority scheduling – who gets best performance, network neutrality

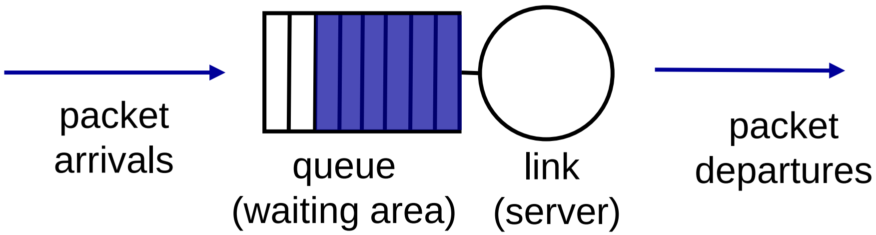
Output port queueing



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

Scheduling mechanisms

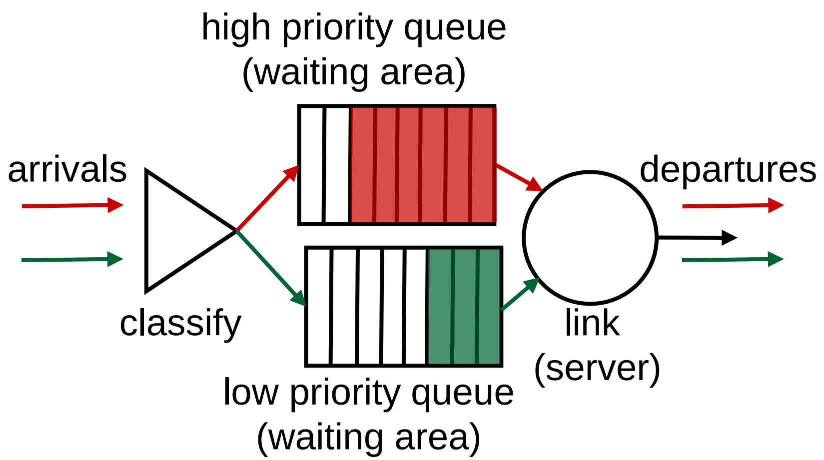
- *Scheduling:* choose next packet to send on link
- *FIFO (first in first out) scheduling:* send in order of arrival to queue
  - real-world example?
  - *discard policy:* if packet arrives to full queue: who to discard?
    - *tail drop:* drop arriving packet
    - *priority:* drop/remove on priority basis
    - *random:* drop/remove randomly

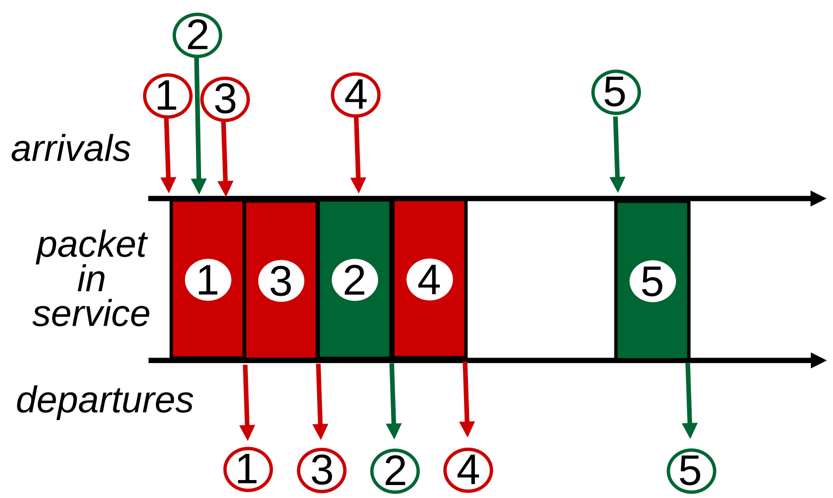


Scheduling policies: priority

priority scheduling: send highest priority queued packet

- multiple *classes*, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.
  - real world example?

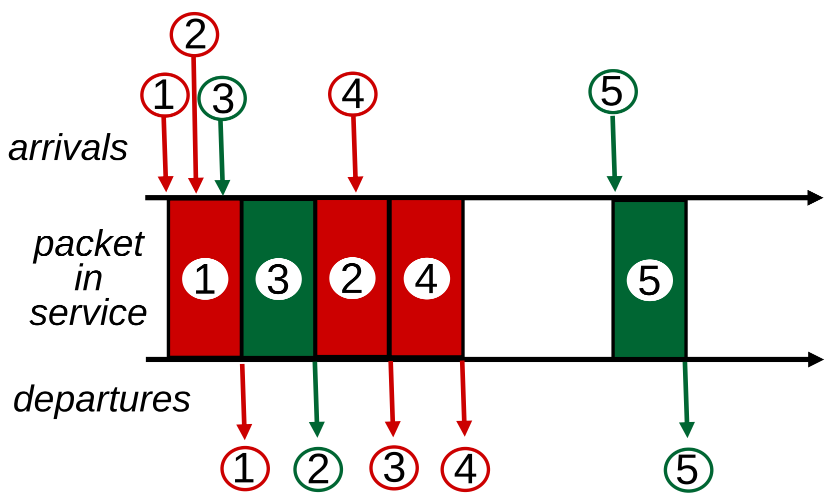




Scheduling policies: still more

Round Robin (RR) scheduling:

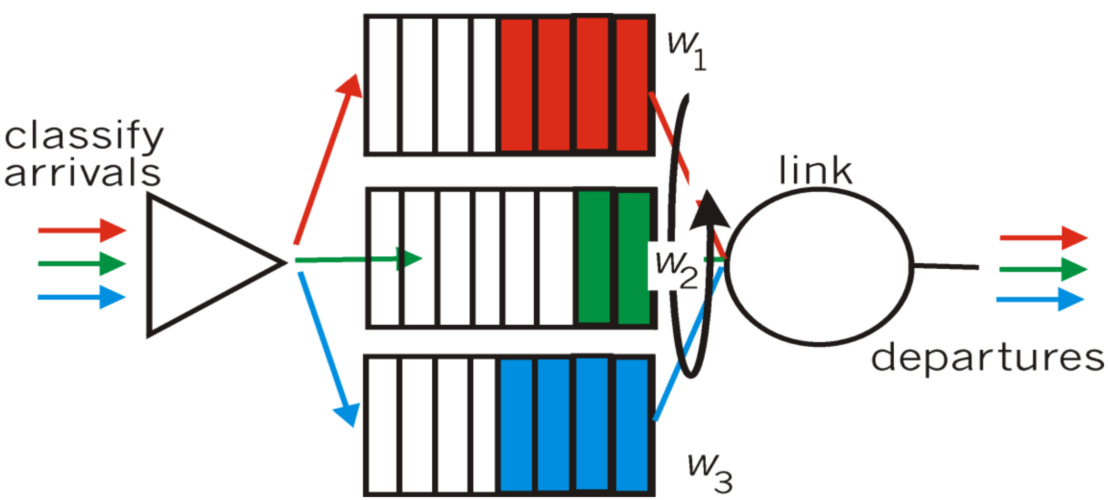
- multiple classes
- cyclically scan class queues, sending one complete packet from each class (if available)
- real world example?



Scheduling policies: still more

Weighted Fair Queuing (WFQ):

- generalized Round Robin
- each class gets weighted amount of service in each cycle

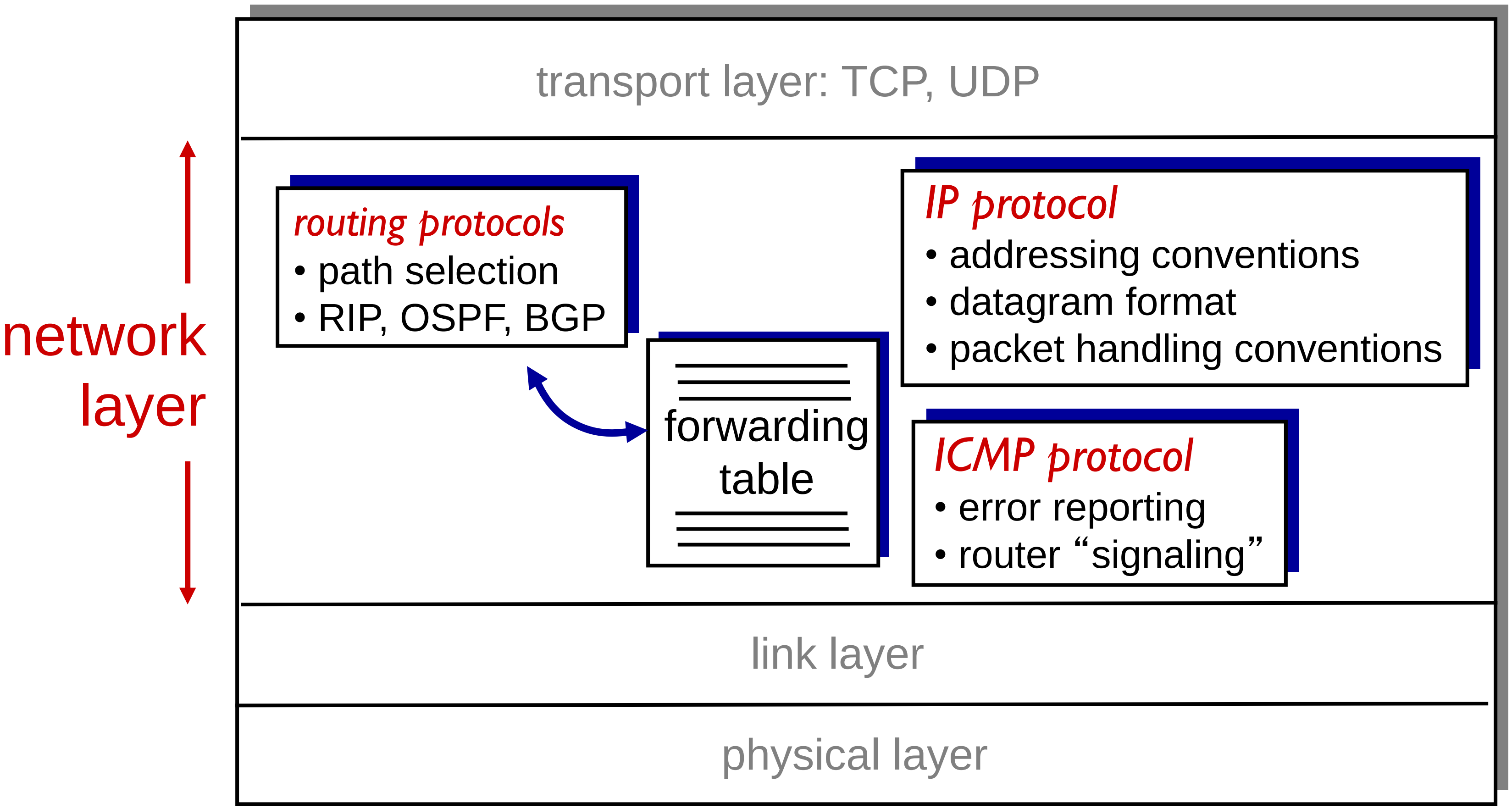


Topics

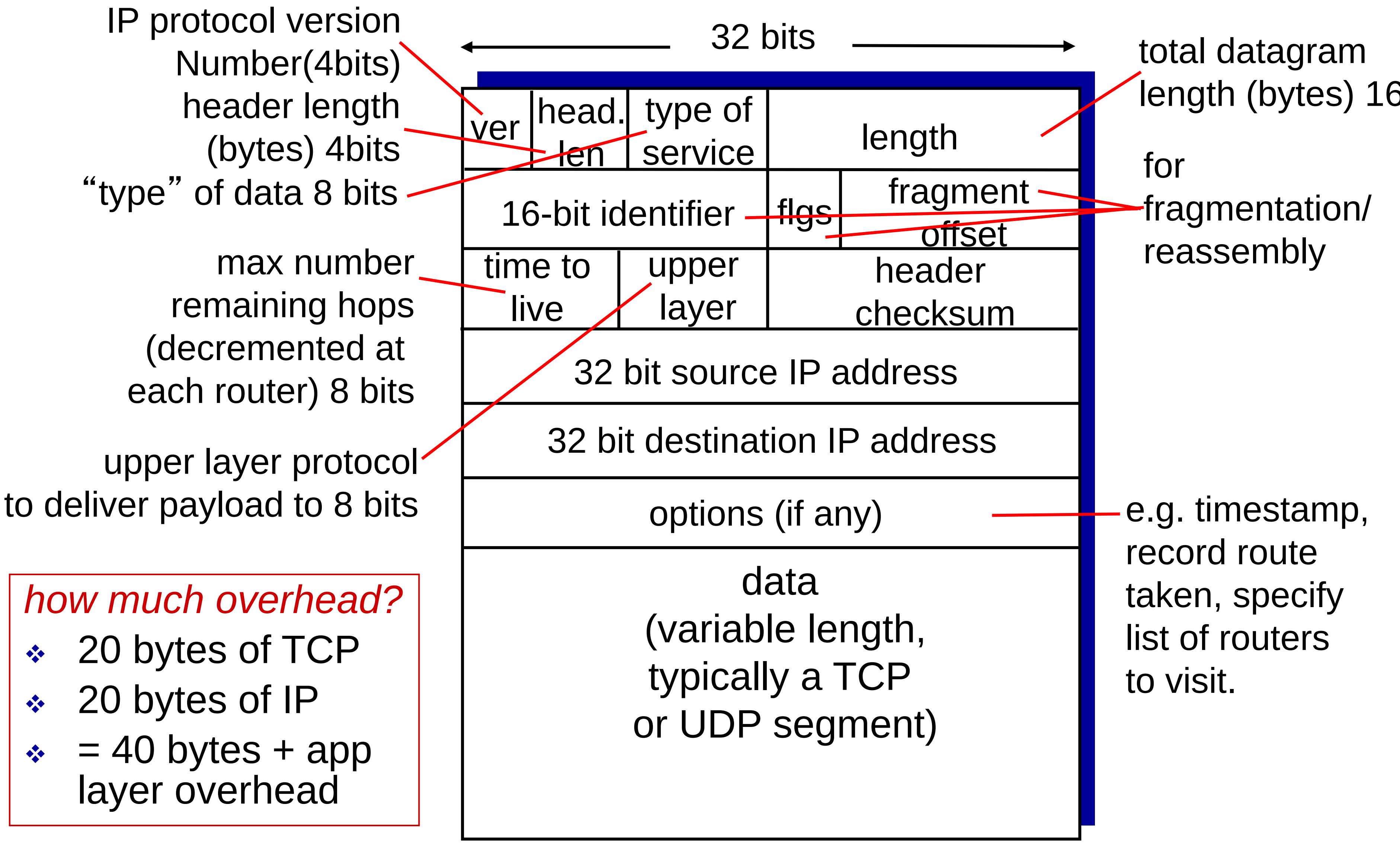
- Overview of Network layer
  - data plane
  - control plane
- What's inside a router
- IP: Internet Protocol
  - datagram format
  - fragmentation
  - IPv4 addressing
  - network address translation
  - IPv6

The Internet network layer

host, router network layer functions:



IP datagram format

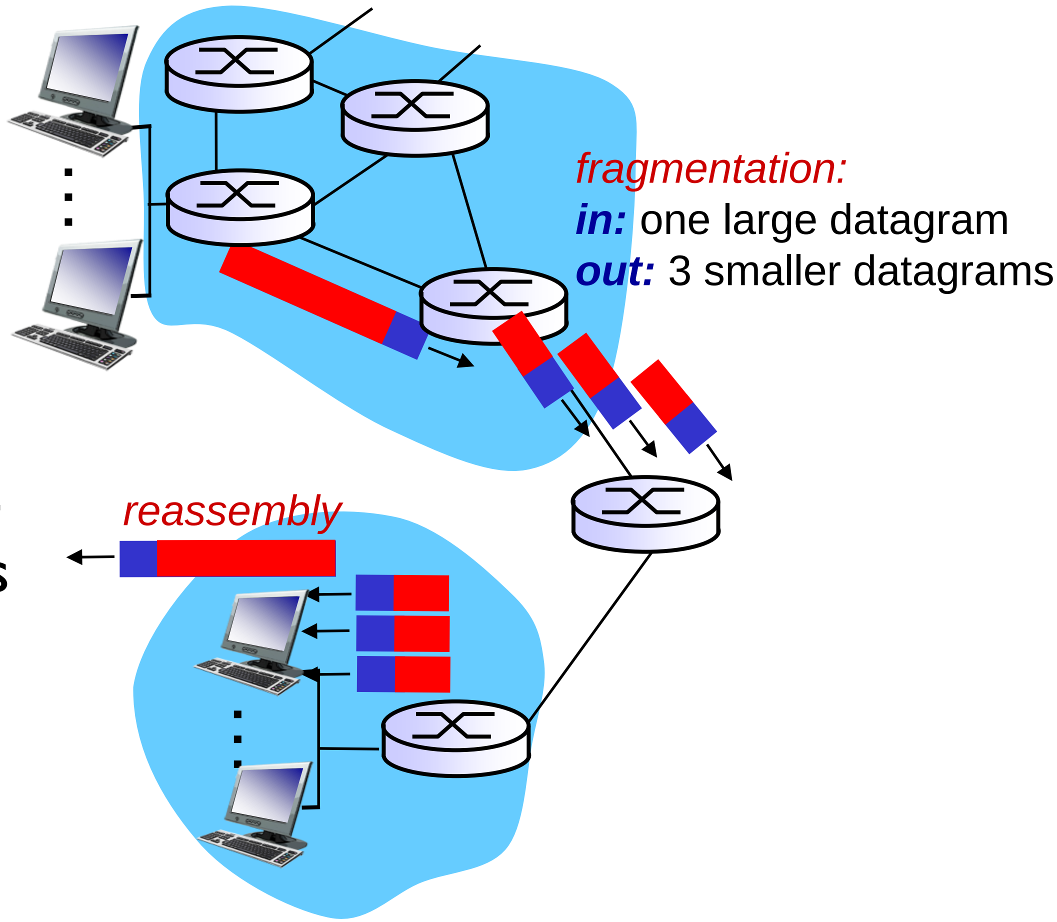


how much overhead?

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

IP fragmentation, reassembly

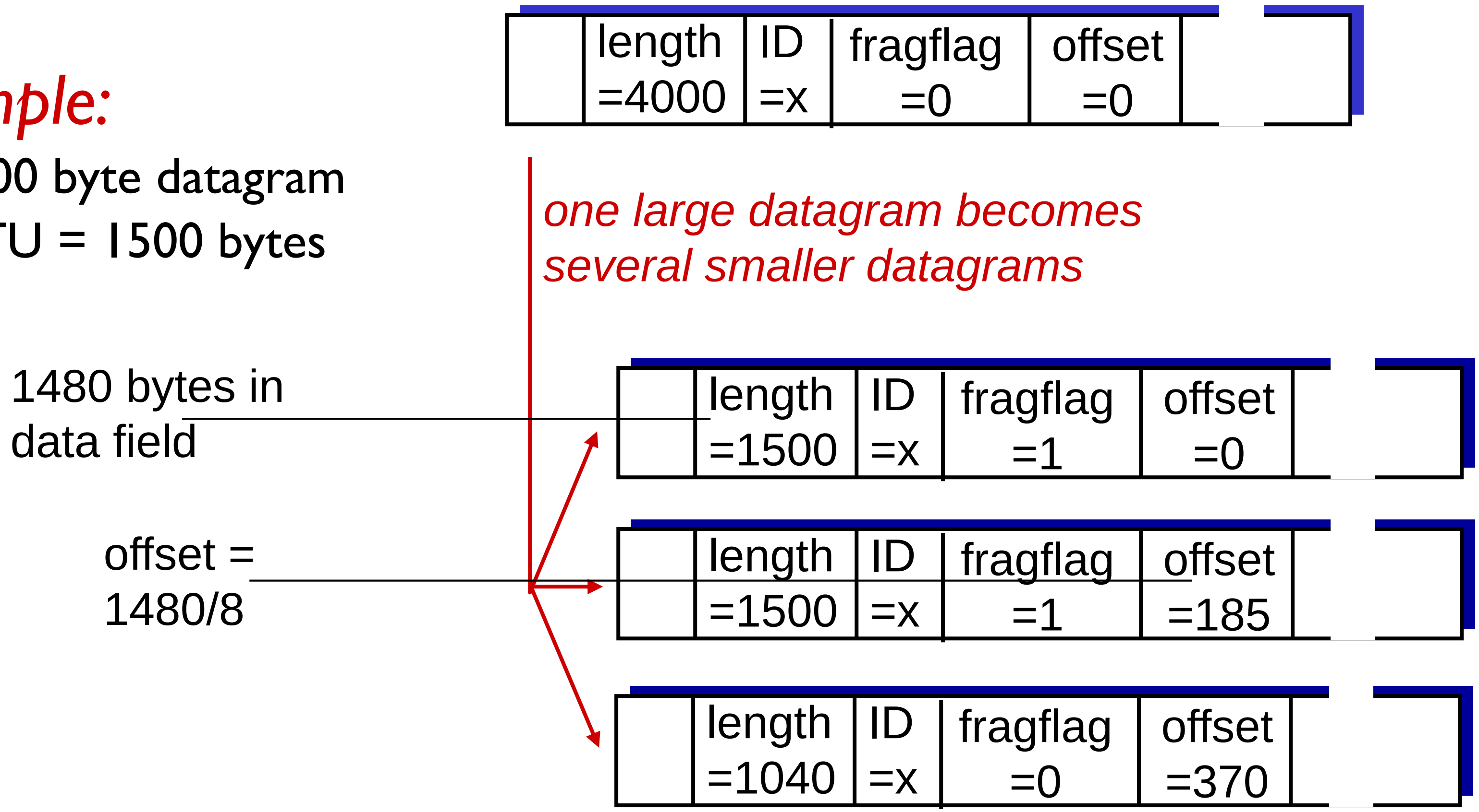
- network links have MTU (max.transfer size) - largest possible link-level frame
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - **"reassembled" only at final destination**
  - IP header bits used to identify, order related fragments



IP fragmentation, reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes

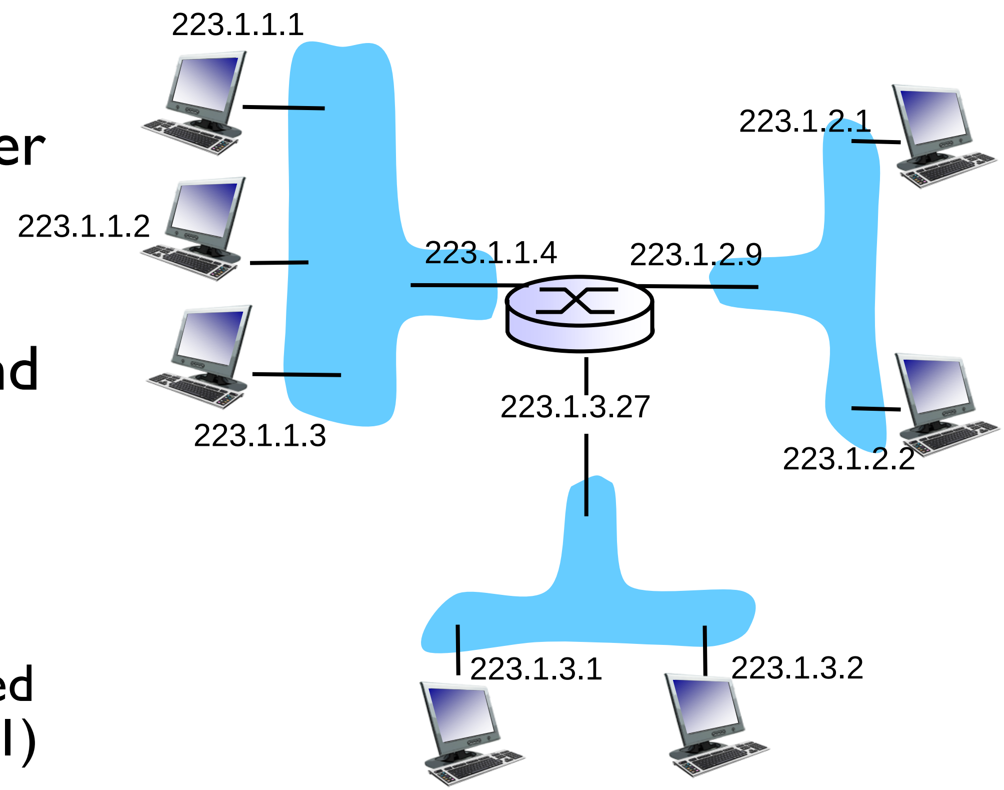


Topics

- Overview of Network layer
  - data plane
  - control plane
- What's inside a router
- IP: Internet Protocol
  - datagram format
  - fragmentation
  - IPv4 addressing
  - network address translation
  - IPv6

Recap: IP addressing

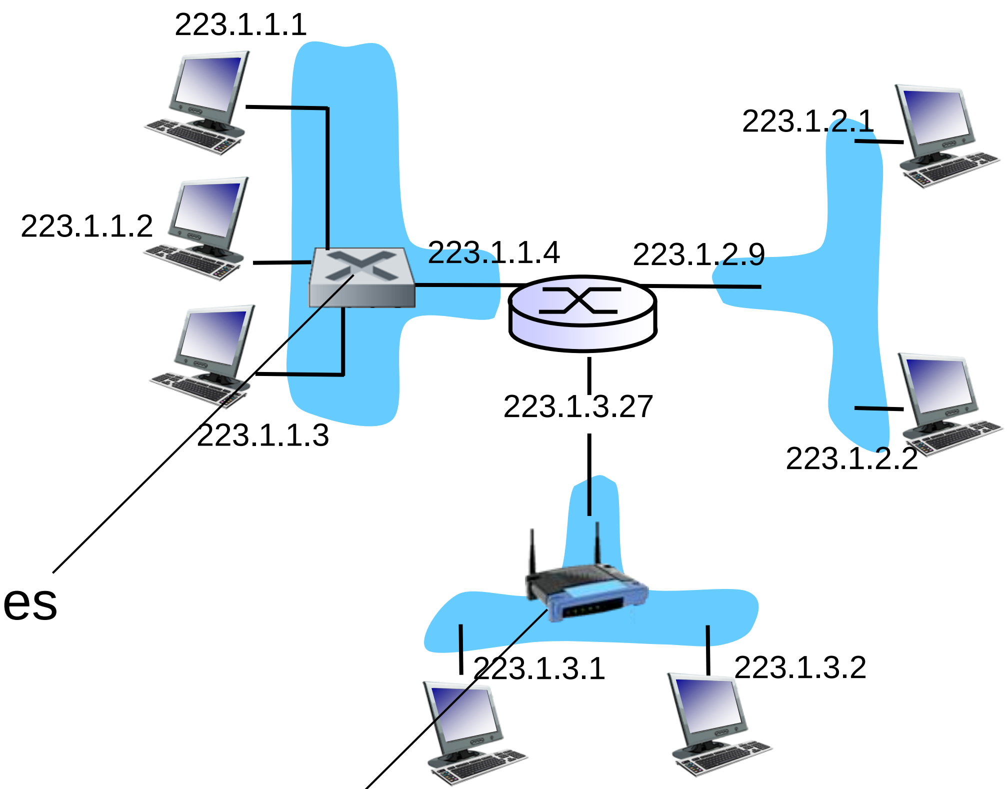
- *IP address:* 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
- *IP addresses associated with each interface*



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

IP addressing: Introduction

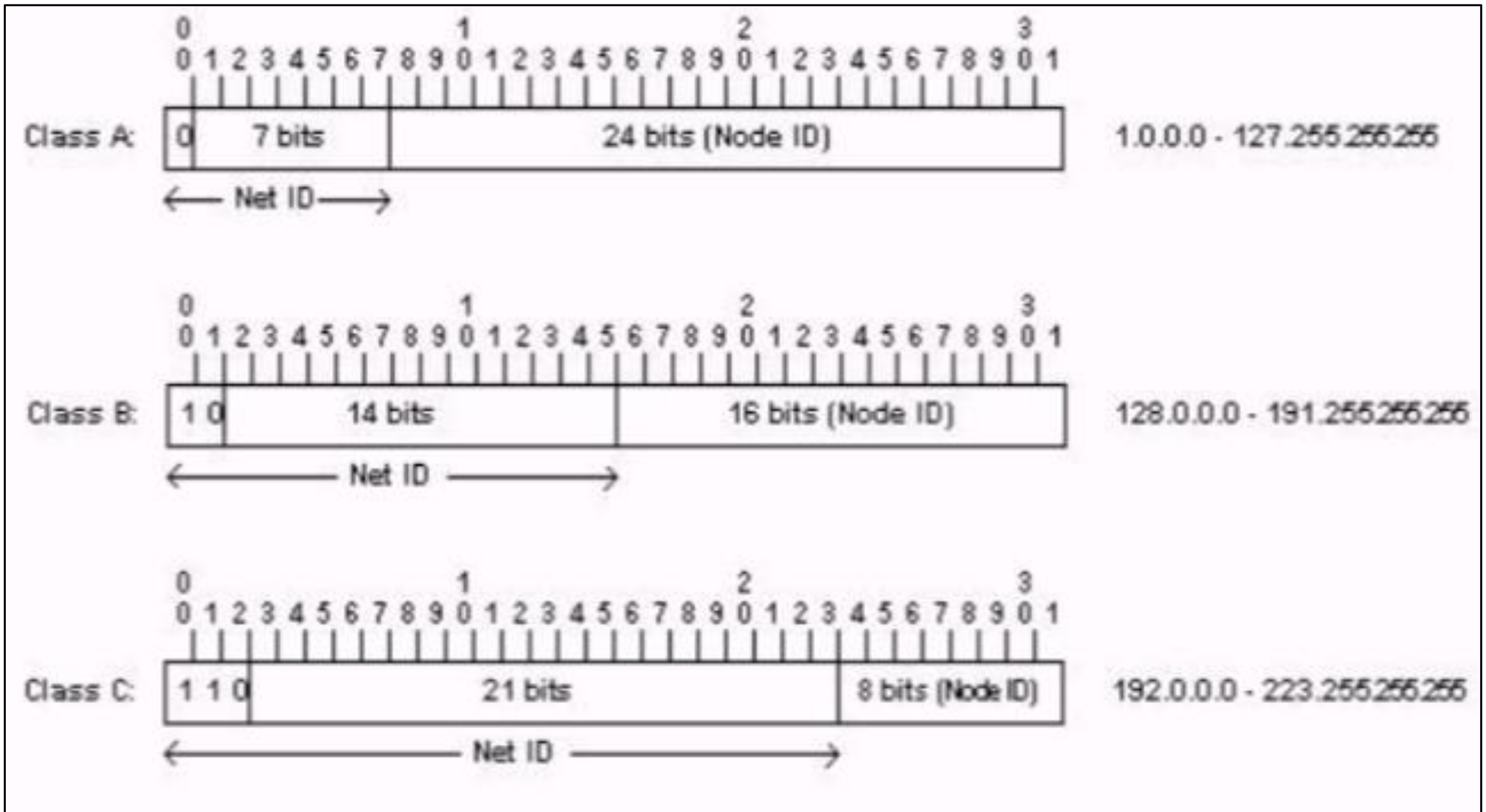
Q: how are interfaces actually connected?



A: wired Ethernet interfaces connected by Ethernet switches

A: wireless WiFi interfaces connected by WiFi base station

IP Address Classes



IP Address Classes

- Class A: The first octet is the network portion.
  - Octets 2, 3, and 4 are for subnets/hosts
- Class B: The first two octets are the network portion.
  - Octets 3 and 4 are for subnets/hosts
- Class C: The first three octets are the network portion.
  - Octet 4 is for subnets/hosts

Private Address Range

Address Class	Reserved Address Space
Class A	10.0.0.0 - 10.255.255.255
Class B	172.16.0.0 - 172.31.255.255
Class C	192.168.0.0 - 192.168.255.255

Problem with IP Address Classes

- Inefficient usage of IP addresses!
- Solution: Subnets
  - Creates multiple logical networks that exist within a single Class A, B, or C network.
  - If you do not subnet, you will only be able to use one network from your Class A, B, or C network, which is unrealistic
  - Each LAN must have a unique network ID, with every node on that link being a member of the same network

IP addressing: CIDR

CIDR: Classless InterDomain Routing

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

subnet part

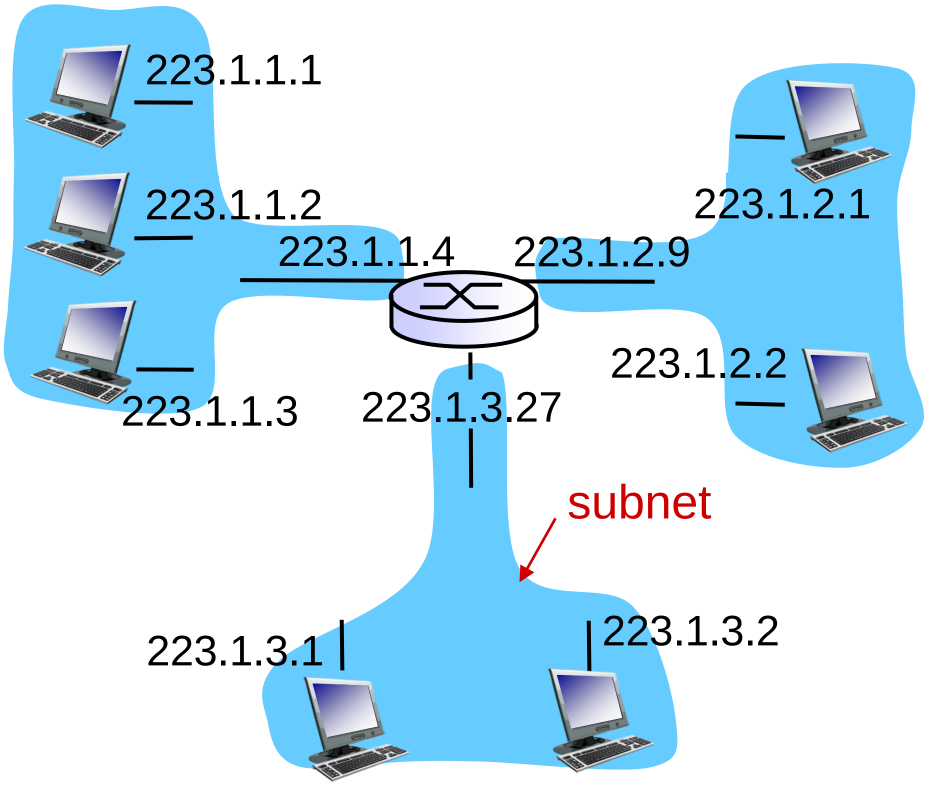
host part

11001000 00010111 00010000 00000000

200.23.16.0/23

Subnets

- *What's a subnet ?*
  - device interfaces with same subnet part of IP address
  - can physically reach each other *without intervening router*



network consisting of 3 subnets

Subnetting Your Network

- Every IP address has a Subnet Mask
  - 172.16.25.2 255.255.0.0
    - subnet part - high order bits
    - host part - low order bits
- Classless Interdomain Routing(CIDR)
  - 172.16.25.2 /16

Subnet Mask

- Determines the way an IP address is split into network and hosts portions

- **Class A** - 0nnnnnnn.hhhhhhhh.hhhhhhhh.hhhhhhhh

Subnet Mask = 255.0.0.0

Subnet Mask

- Determines the way an IP address is split into network and hosts portions

- **Class A** - 0nnnnnnn.hhhhhhhh.hhhhhhhh.hhhhhhhh

Subnet Mask = 255.0.0.0 $\longrightarrow$ IP Address /8

Subnet Mask

- Determines the way an IP address is split into network and hosts portions

- **Class A** - 0nnnnnnn.hhhhhhhh.hhhhhhhh.hhhhhhhh

Subnet Mask = 255.0.0.0 → IP Address /8

- **Class B** - 10nnnnnn.nnnnnnnn.hhhhhhhh.hhhhhhhh

Subnet Mask = 255.255.0.0 → IP Address /?

Subnet Mask

- Determines the way an IP address is split into network and hosts portions

- **Class A** - 0nnnnnnn.hhhhhhhh.hhhhhhhh.hhhhhhhh

Subnet Mask = 255.0.0.0 → IP Address /8

- **Class B** - 10nnnnnn.nnnnnnnn.hhhhhhhh.hhhhhhhh

Subnet Mask = 255.255.0.0 → IP Address /16

Subnet Mask

- Determines the way an IP address is split into network and hosts portions

- **Class A** - 0nnnnnnn.hhhhhhhh.hhhhhhhh.hhhhhhhh

Subnet Mask = 255.0.0.0 → IP Address /8

- **Class B** - 10nnnnnn.nnnnnnnn.hhhhhhhh.hhhhhhhh

Subnet Mask = 255.255.0.0 → IP Address /16

- **Class C** - 100nnnnn.nnnnnnnn.nnnnnnnn.hhhhhhhh

Subnet Mask = 255.255.255.0 IP Address /?

Subnet Mask

- Determines the way an IP address is split into network and hosts portions

- **Class A** - 0nnnnnnn.hhhhhhhh.hhhhhhhh.hhhhhhhh

Subnet Mask = 255.0.0.0 → IP Address /8

- **Class B** - 10nnnnnn.nnnnnnnn.hhhhhhhh.hhhhhhhh

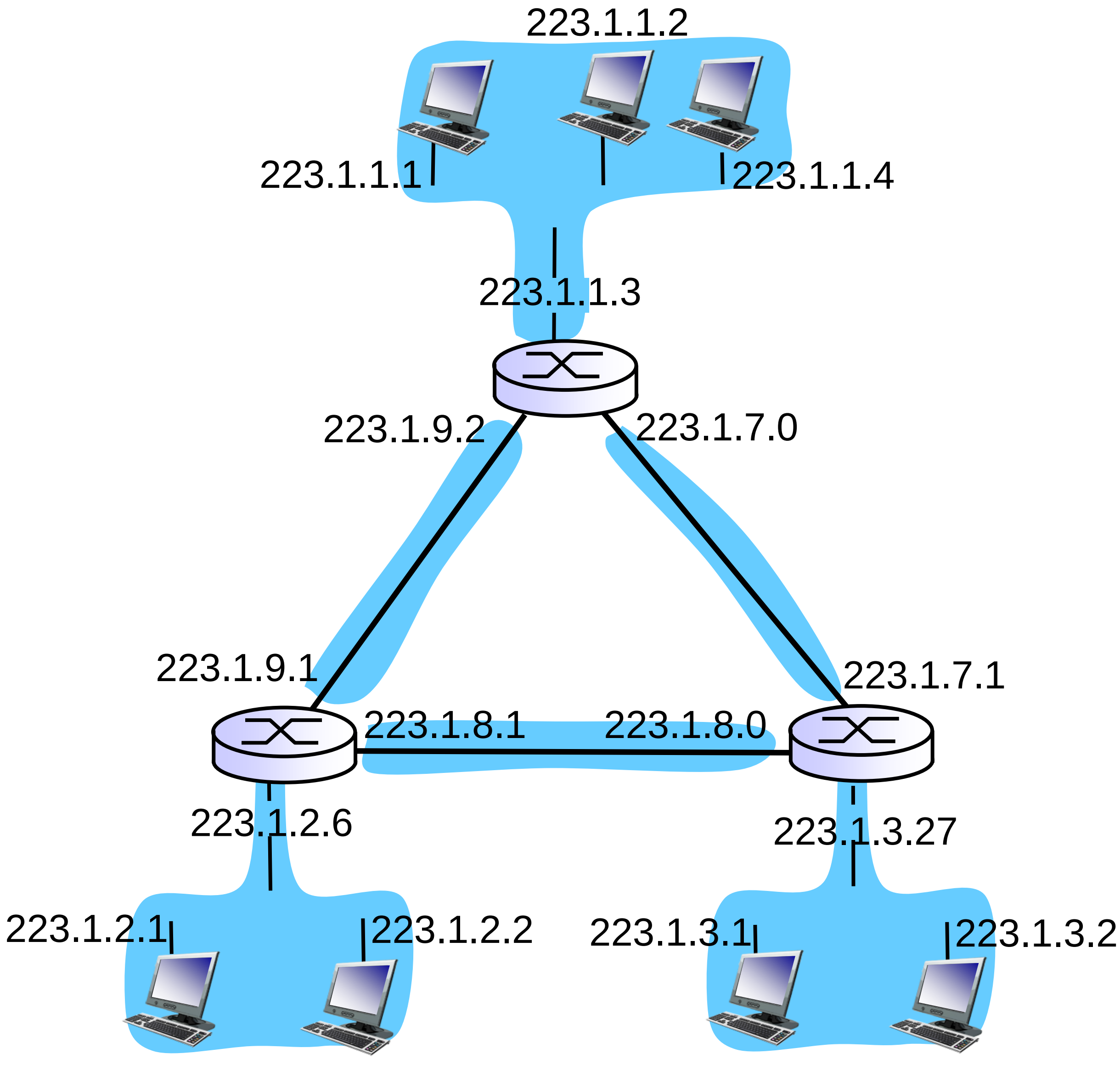
Subnet Mask = 255.255.0.0 → IP Address /16

- **Class C** - 100nnnnn.nnnnnnnn.nnnnnnnn.hhhhhhhh

Subnet Mask = 255.255.255.0 IP Address /24

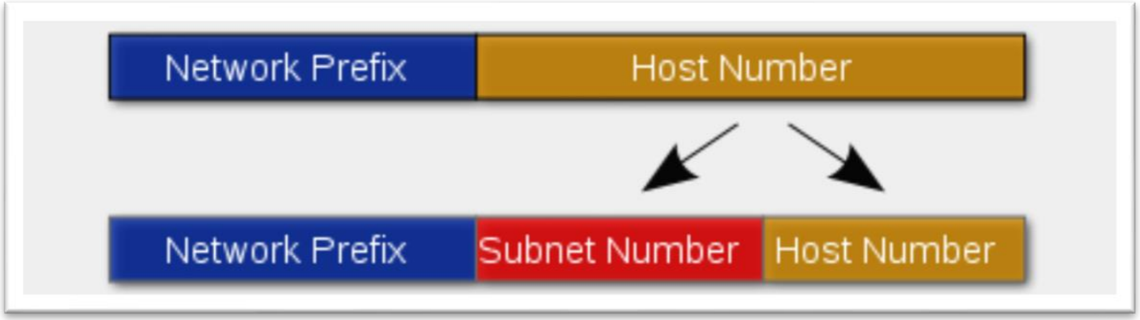
Subnets

how many?



Subnetted Networks

- The network portion of the address is **<u>extended</u>** by splitting up the host number



- Borrowing 1 or more bits from the host bit portion

Example:

Dividing a network into 2 subnets requires to borrow 1 bit

Class C:

11111111.11111111.11111111.10000000(255.255.255.128)

CIDR = IP address /25

This would allow 126 hosts per subnet

All 1's are reserved for broadcast ID

All 0's are reserved for network ID

Class C Subnetting

# of Subnets	# of Hosts/Subnet	NetMask	4th Octet	CIDR Notation
2				
4				
8				
16				
32				
64				

Class C Subnetting

# of Subnets	# of Hosts/Subnet	NetMask	4^{th} Octet	CIDR Notation
2	126	255.255.255.128	10000000	/25
4				
8				
16				
32				
64				

Class C Subnetting

# of Subnets	# of Hosts/Subnet	NetMask	4th Octet	CIDR Notation
2	126	255.255.255.128	10000000	/25
4	62	255.255.255.192	11000000	/26
8				
16				
32				
64				

Class C Subnetting

# of Subnets	# of Hosts/Subnet	NetMask	4^{th} Octet	CIDR Notation
2	126	255.255.255.128	10000000	/25
4	62	255.255.255.192	11000000	/26
8	30	255.255.255.224	11100000	/27
16				
32				
64				

Class C Subnetting

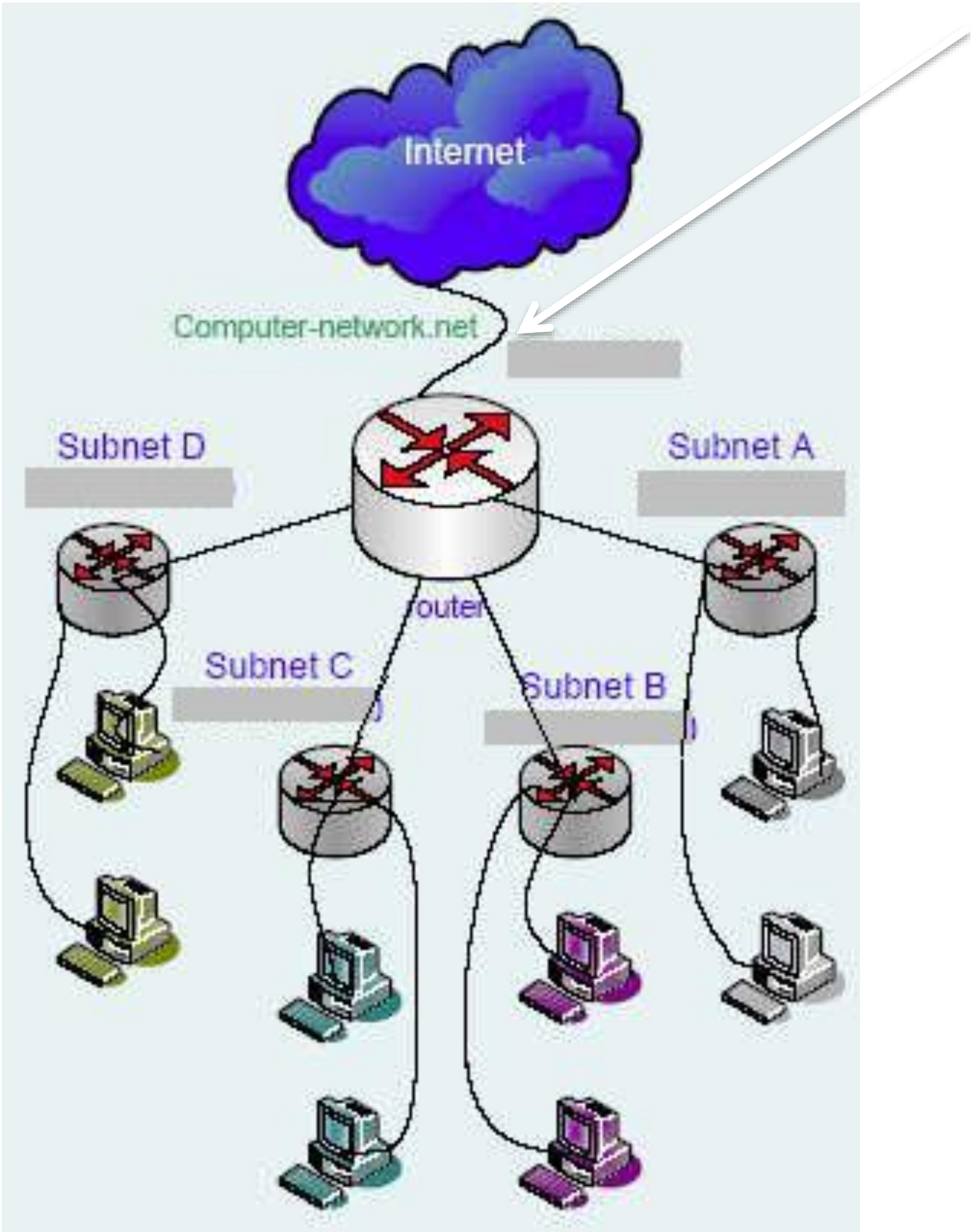
# of Subnets	# of Hosts/Subnet	NetMask	4th Octet	CIDR Notation
2	126	255.255.255.128	10000000	/25
4	62	255.255.255.192	11000000	/26
8	30	255.255.255.224	11100000	/27
16	14	255.255.255.240	11110000	/28
32				
64				

Class C Subnetting

# of Subnets	# of Hosts/Subnet	NetMask	4th Octet	CIDR Notation
2	126	255.255.255.128	10000000	/25
4	62	255.255.255.192	11000000	/26
8	30	255.255.255.224	11100000	/27
16	14	255.255.255.240	11110000	/28
32	6	255.255.255.248	11111000	/29
64				

Class C Subnetting

# of Subnets	# of Hosts/Subnet	NetMask	4th Octet	CIDR Notation
2	126	255.255.255.128	10000000	/25
4	62	255.255.255.192	11000000	/26
8	30	255.255.255.224	11100000	/27
16	14	255.255.255.240	11110000	/28
32	6	255.255.255.248	11111000	/29
64	2	255.255.255.252	11111100	/30

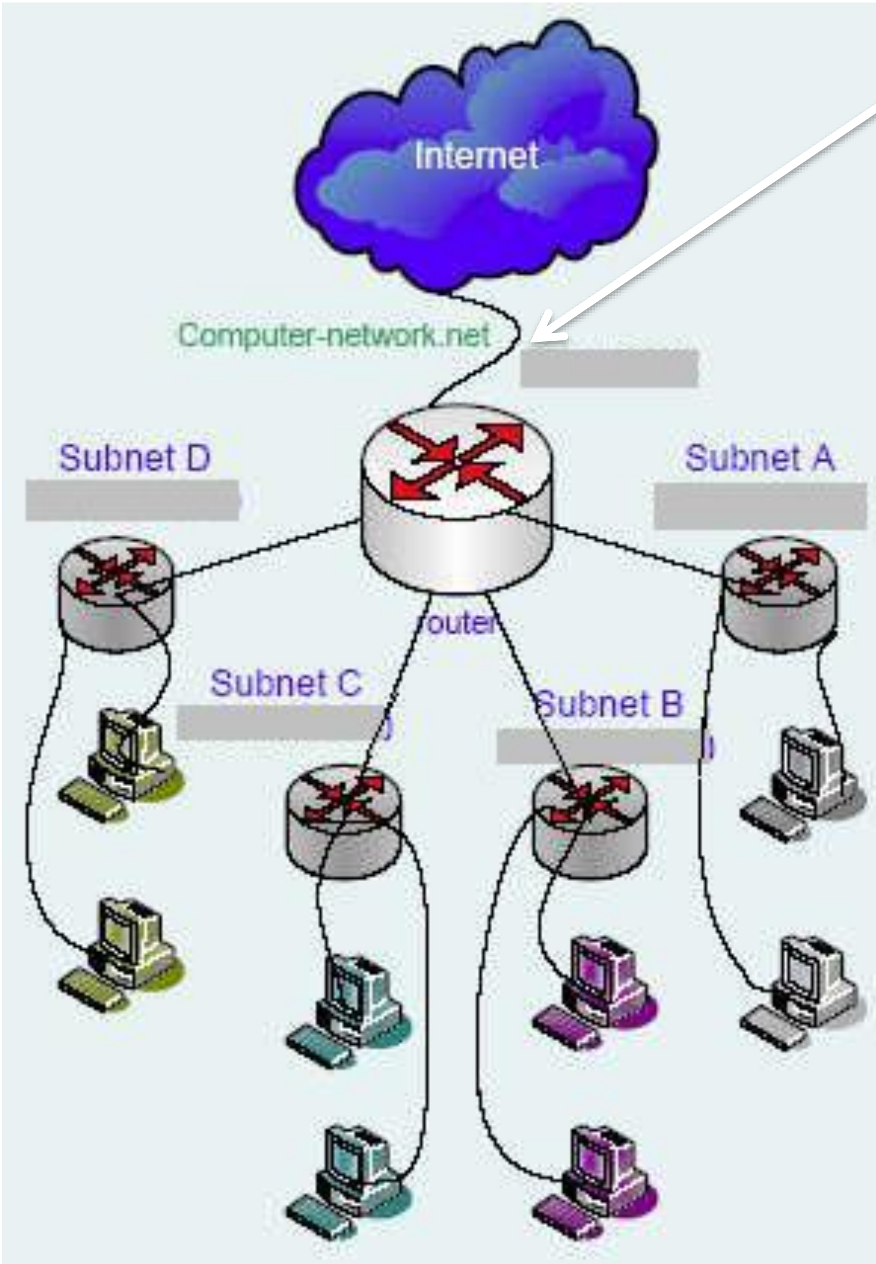


192.168.5.130 /24

Subnet mask = 255.255.255.0

192.168.5.0 = Network ID

Create 4 Subnets. How?



192.168.5.130 /24

Subnet mask = 255.255.255.0

192.168.5.0 = Network ID

Create 4 Subnets. How?

192.168.5.0 =>

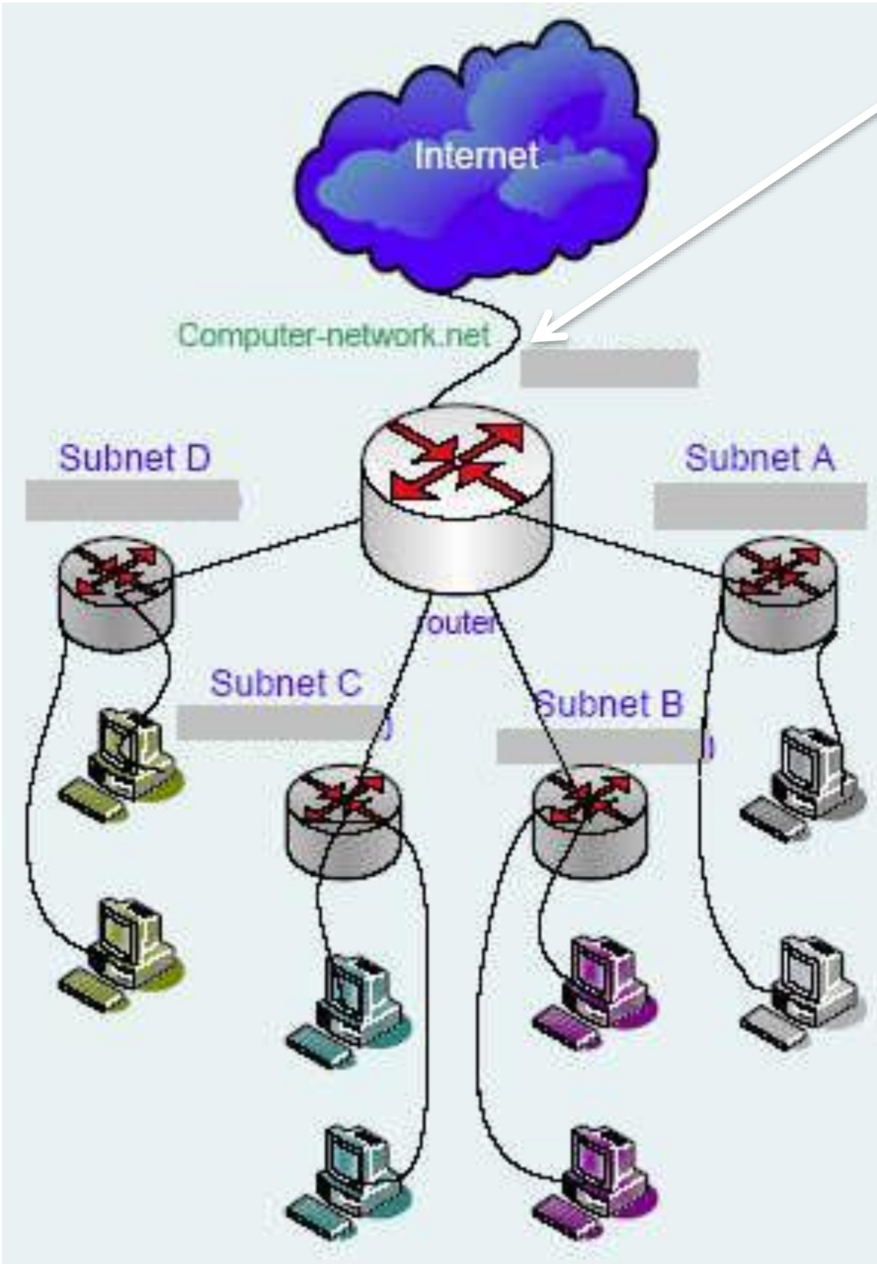
11000000.10101000.00000101.00000000

Borrow 2 bits from host byte

11000000.10101000.00000101.**11**000000

Subnet Mask = 255.255.255.192

= /26



192.168.5.130 /24

Subnet mask = 255.255.255.0

192.168.5.0 = Network ID

Create 4 Subnets. How?

192.168.5.0 =>

11000000.10101000.00000101.**00**000000

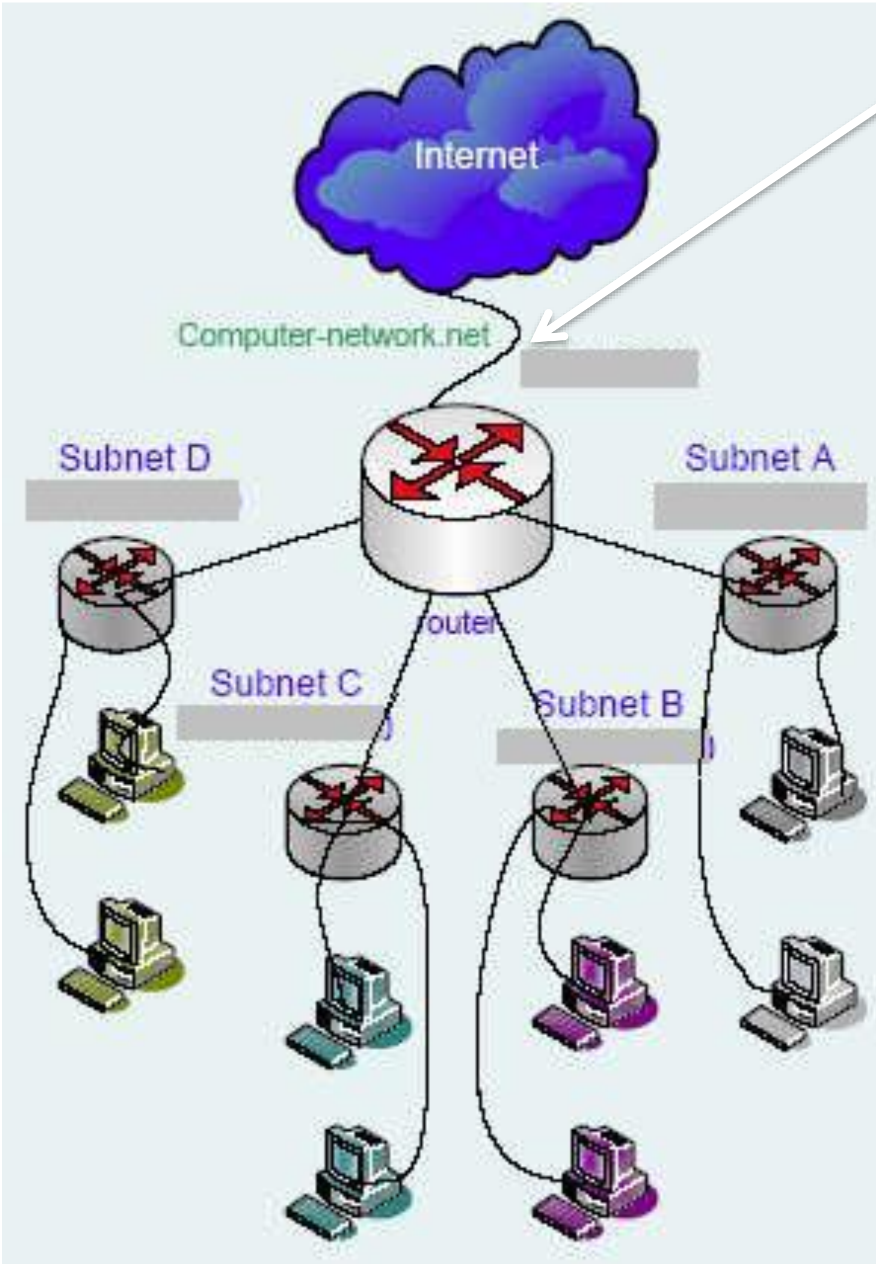
Borrow 2 bits from host byte

11000000.10101000.00000101.**11**000000

Subnet Mask = 255.255.255.192

= /26

What are my new subnets?



192.168.5.130 /24

Subnet mask = 255.255.255.0

192.168.5.0 = Network ID

Create 4 Subnets. How?

192.168.5.0 =>

11000000.10101000.00000101.**00**000000

Borrow 2 bits from host byte

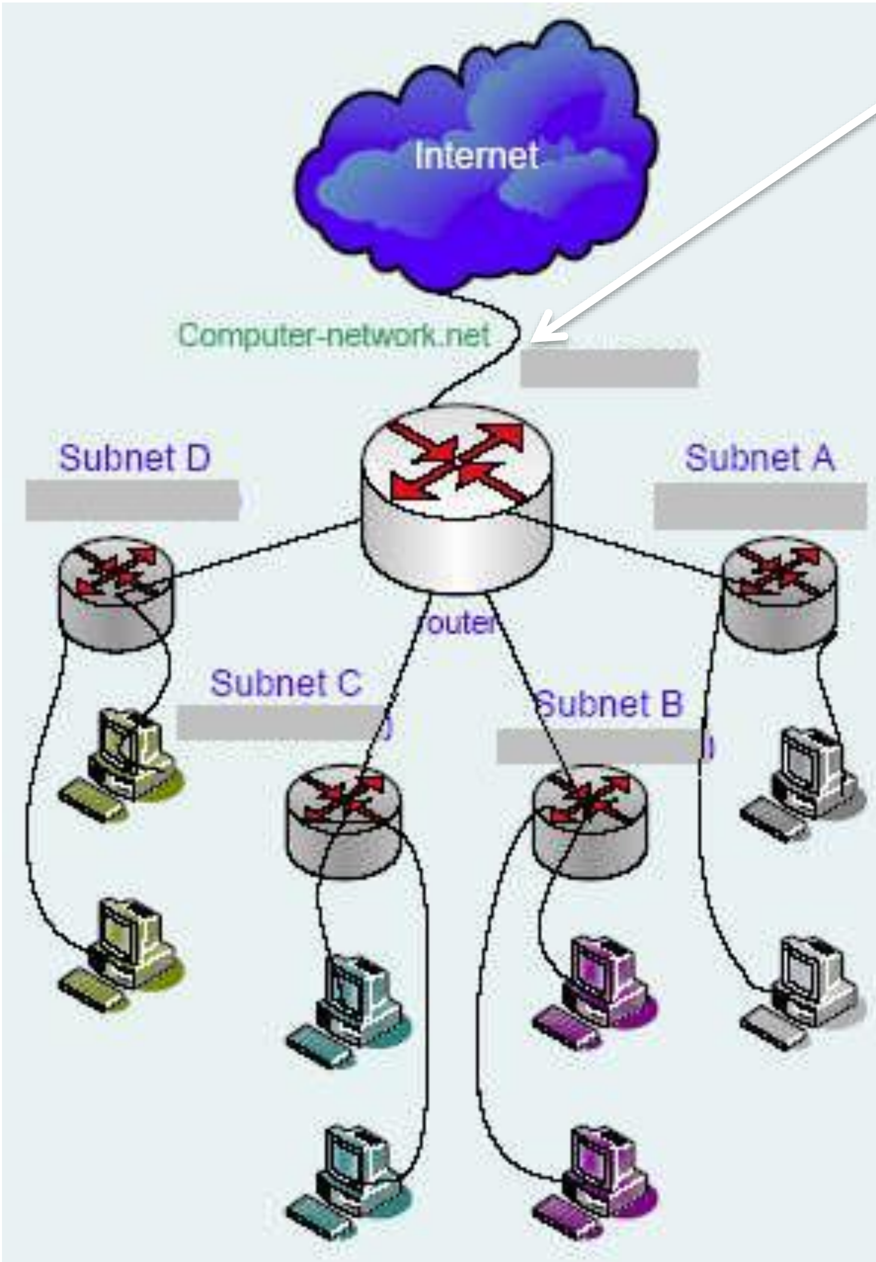
11000000.10101000.00000101.**11**000000

Subnet Mask = 255.255.255.192

= /26

Subnet A -> 192.168.5.1/26

to 192.168.5.62/26



192.168.5.130 /24

Subnet mask = 255.255.255.0

192.168.5.0 = Network ID

Create 4 Subnets. How?

192.168.5.0 =>

11000000.10101000.00000101.**00**000000

Borrow 2 bits from host byte

11000000.10101000.00000101.**11**000000

Subnet Mask = 255.255.255.192

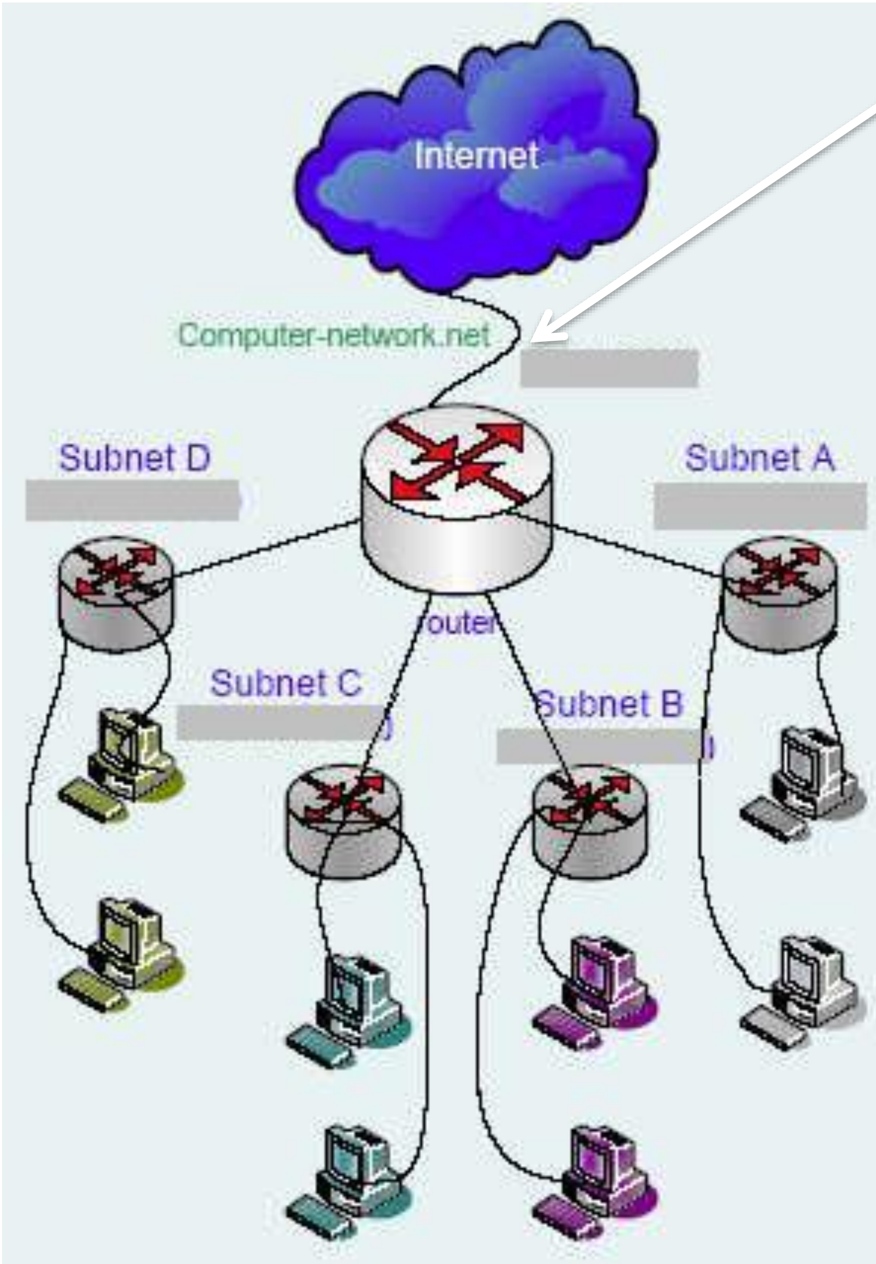
= /26

Subnet A -> 192.168.5.1/26

to 192.168.5.62/26

Subnet B -> 192.168.5.65/26

to 192.168.5.126/26



192.168.5.130 /24

Subnet mask = 255.255.255.0

192.168.5.0 = Network ID

Create 4 Subnets. How?

192.168.5.0 =>

11000000.10101000.00000101.**00**000000

Borrow 2 bits from host byte

11000000.10101000.00000101.**11**000000

Subnet Mask = 255.255.255.192

= /26

Subnet A -> 192.168.5.1/26

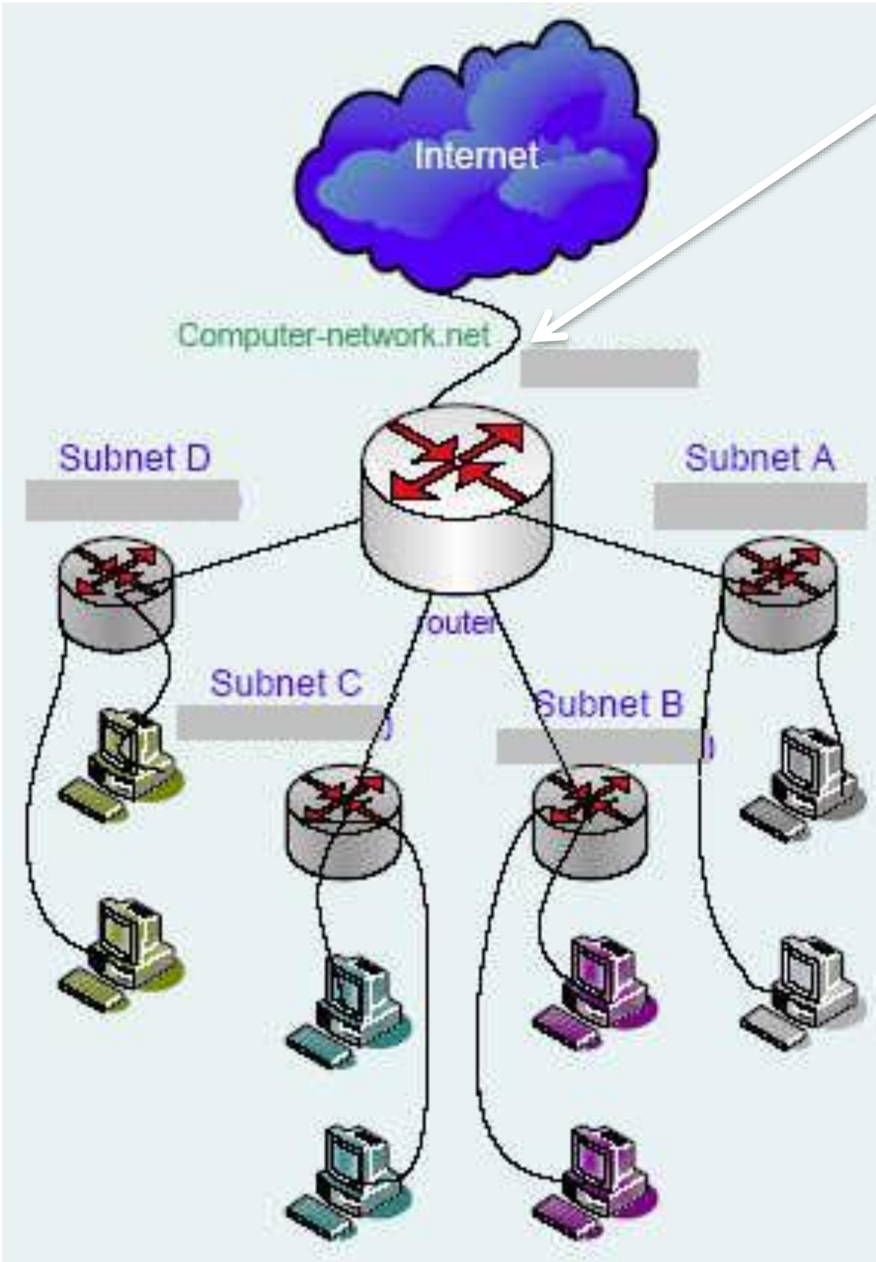
to 192.168.5.62/26

Subnet B -> 192.168.5.65/26

to 192.168.5.126/26

Subnet C -> 192.168.5.129/26

to 192.168.5.190/26



192.168.5.130 /24

Subnet mask = 255.255.255.0

192.168.5.0 = Network ID

Create 4 Subnets. How?

192.168.5.0 =>

11000000.10101000.00000101.**00**000000

Borrow 2 bits from host byte

11000000.10101000.00000101.**11**000000

Subnet Mask = 255.255.255.192

= /26

Subnet A -> 192.168.5.1/26

to 192.168.5.62/26

Subnet B -> 192.168.5.65/26

to 192.168.5.126/26

Subnet C -> 192.168.5.129/26

to 192.168.5.190/26

Subnet D -> 192.168.5.193/26

to 192.168.5.254/26

How to create subnets

- Determine the number of required network IDs:
  - One for each subnet
  - One for each wide area network connection
- Determine the number of required host IDs per subnet:
  - One for each TCP/IP host
  - One for each router interface
- Based on the above requirements, create the following:
  - One subnet mask for your entire network
  - A unique subnet ID for each physical segment
  - A range of host IDs for each subnet

Advantages

- Allows a single shared network address to split it up into many smaller networks.
- Without subnets, organizations would require many network addresses
  - Limited number of Network addresses available
- Alleviates traffic
  - Smaller routing tables
  - Alleviates excessive packet collision and congestion
- Easier to manage and solve problems
- Better Security
  - Separating departments with highly sensitive material
    - *Accounting and Administration*

Disadvantages

- Doesn't allocate IP address proportionately per subnet
- Limited by the number of IP address
- Need to buy hardware such as routers

IP addresses: how to get one?

Q: How does a *host* get IP address?

IP addresses: how to get one?

Q: How does a *host* get IP address?

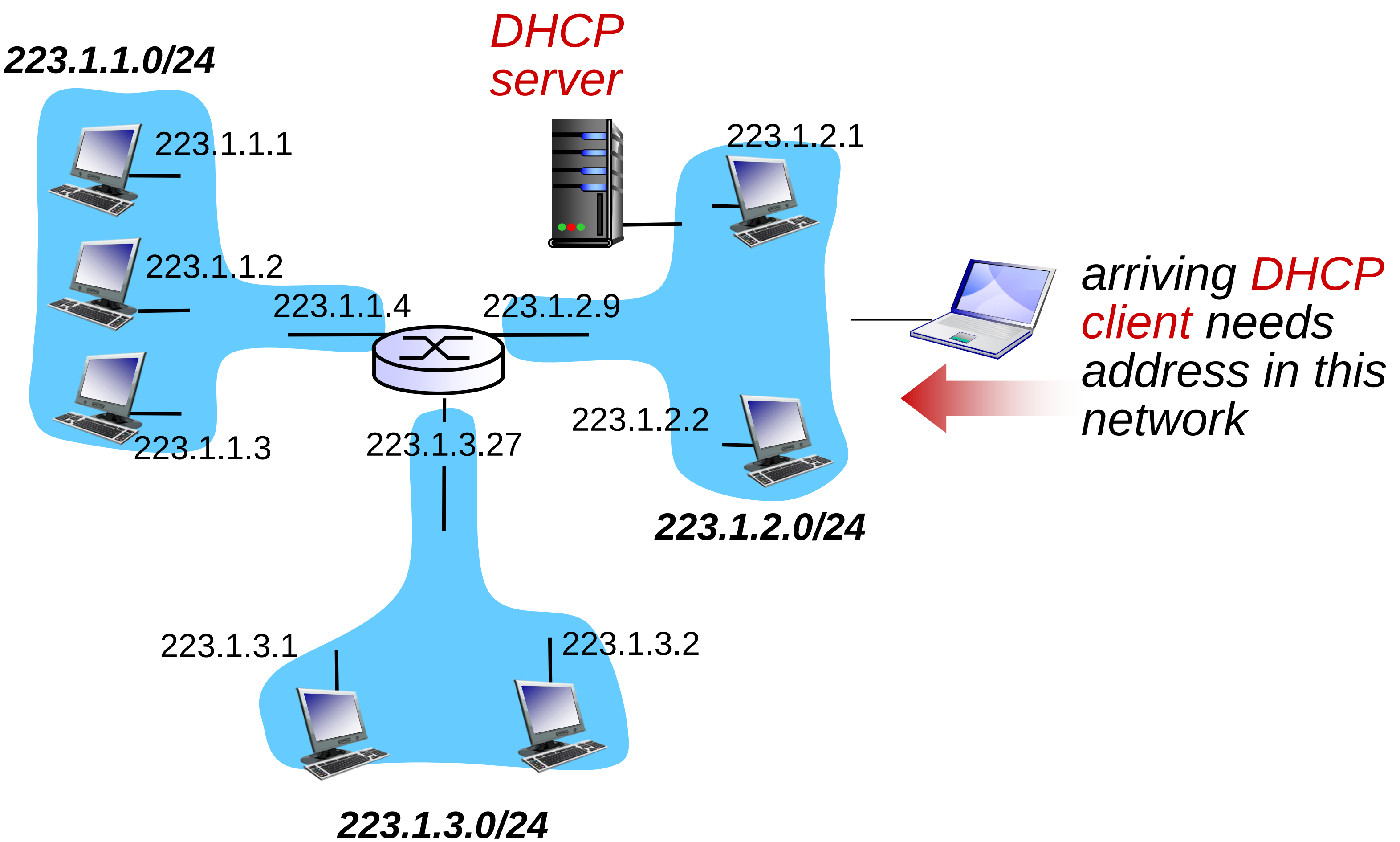
- hard-coded by system admin in a file
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

DHCP: Dynamic Host Configuration Protocol

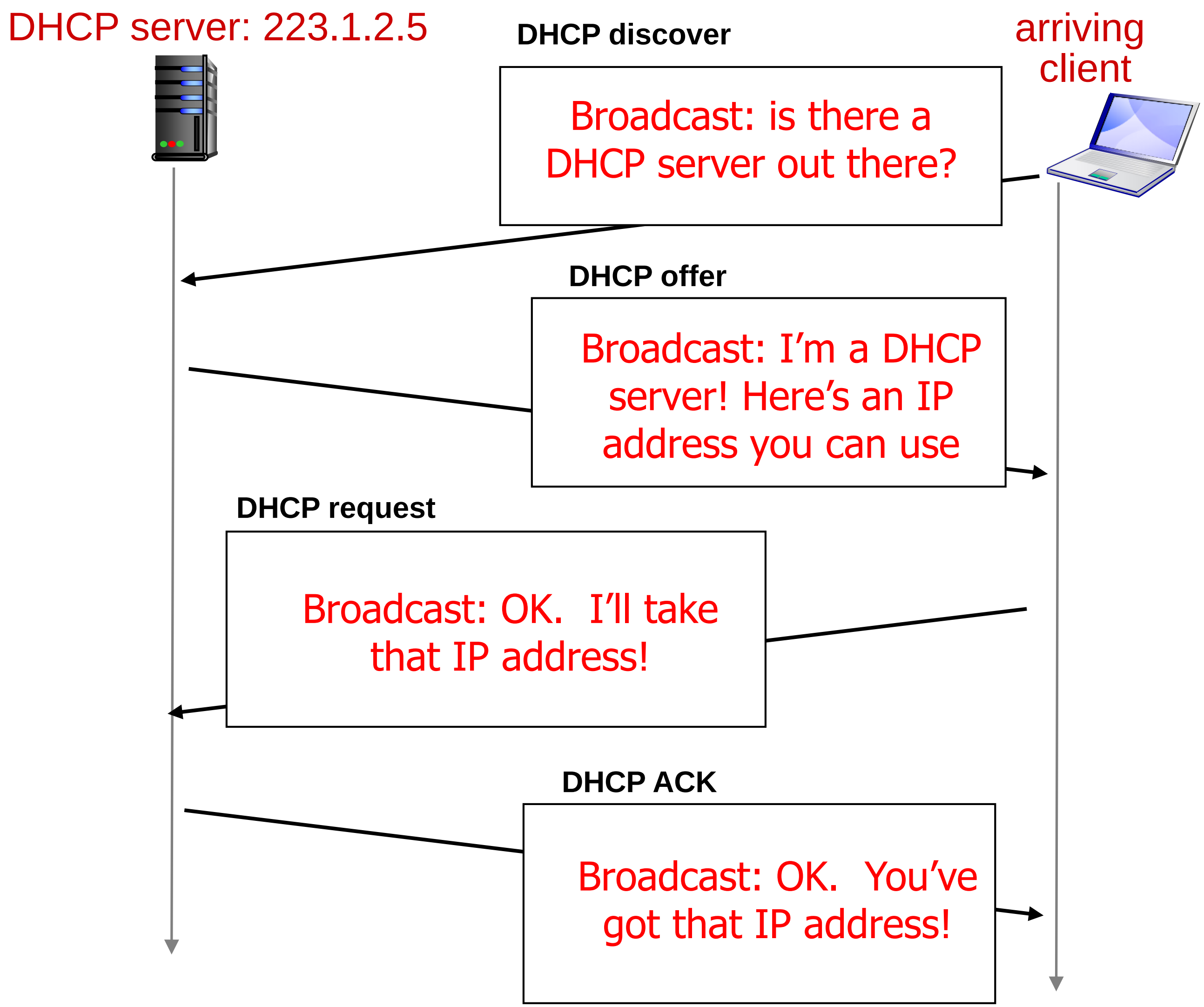
Goal: allow host to *dynamically* obtain its IP address from network server when it joins network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/"on")
- support for mobile users who want to join network (more shortly)

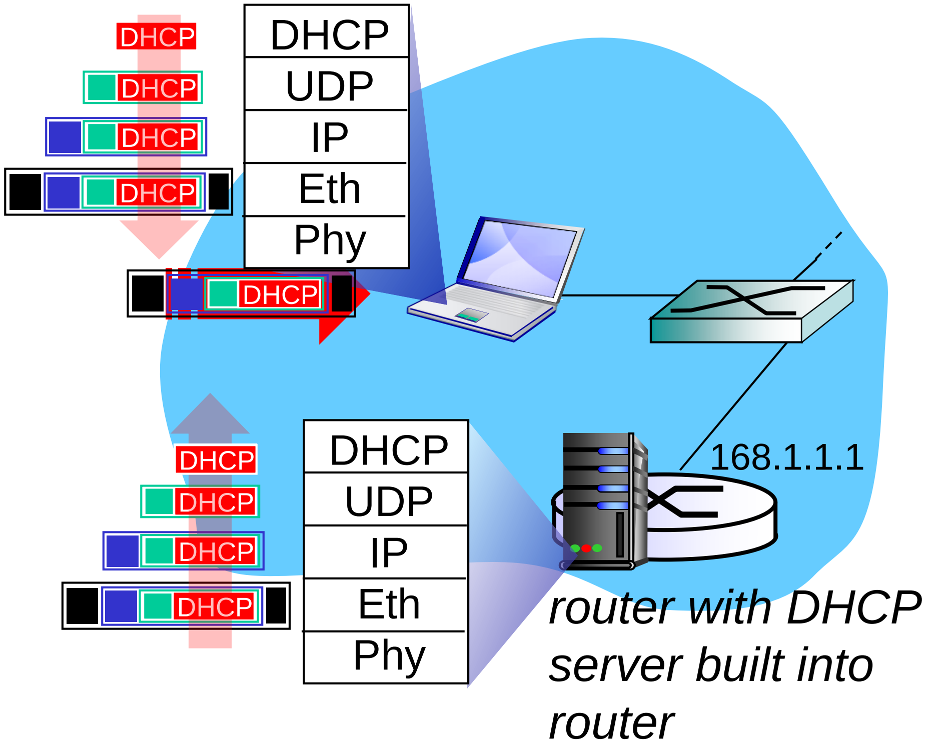
DHCP client-server scenario



DHCP client-server scenario

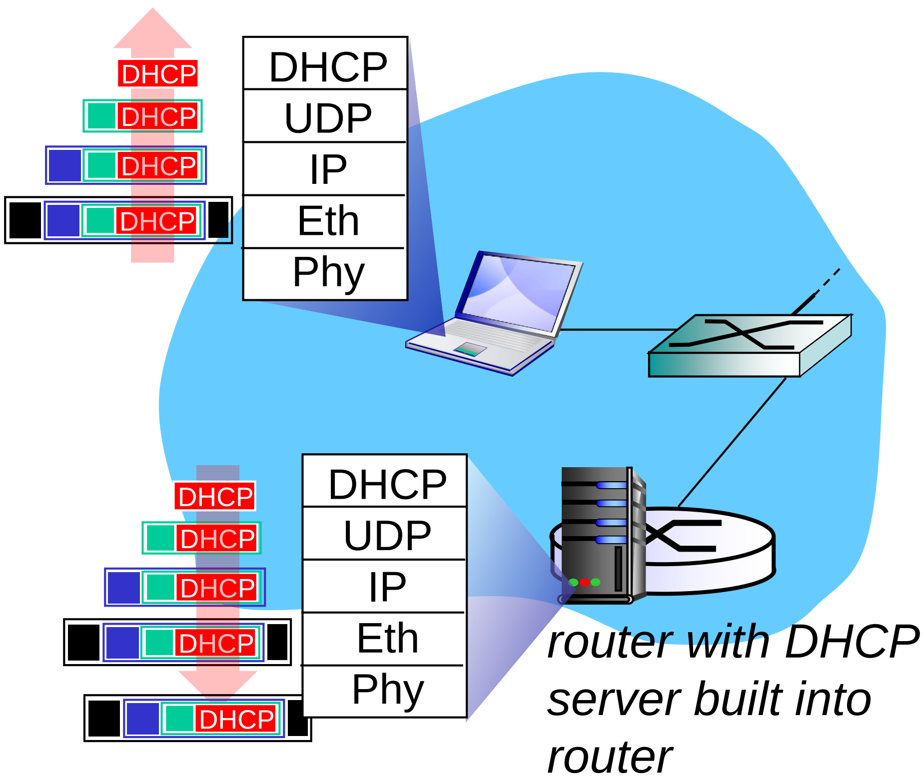


DHCP: example



- connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP.
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server

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
- encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client
- client now knows its IP address, name and IP address of DSN server, IP address of its first-hop router

DHCP Wireshark exercise

- Refer to the document on Blackboard

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)

IP addresses: how to get one?

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

A: gets 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 0001000	0 00000000	200.23.16.0/23
Organization 1	11001000 00010111 0001001	0 00000000	200.23.18.0/23
Organization 2	11001000 00010111 0001010	0 00000000	200.23.20.0/23
...	…..	….	….
Organization 7	11001000 00010111 0001111	0 00000000	200.23.30.0/23

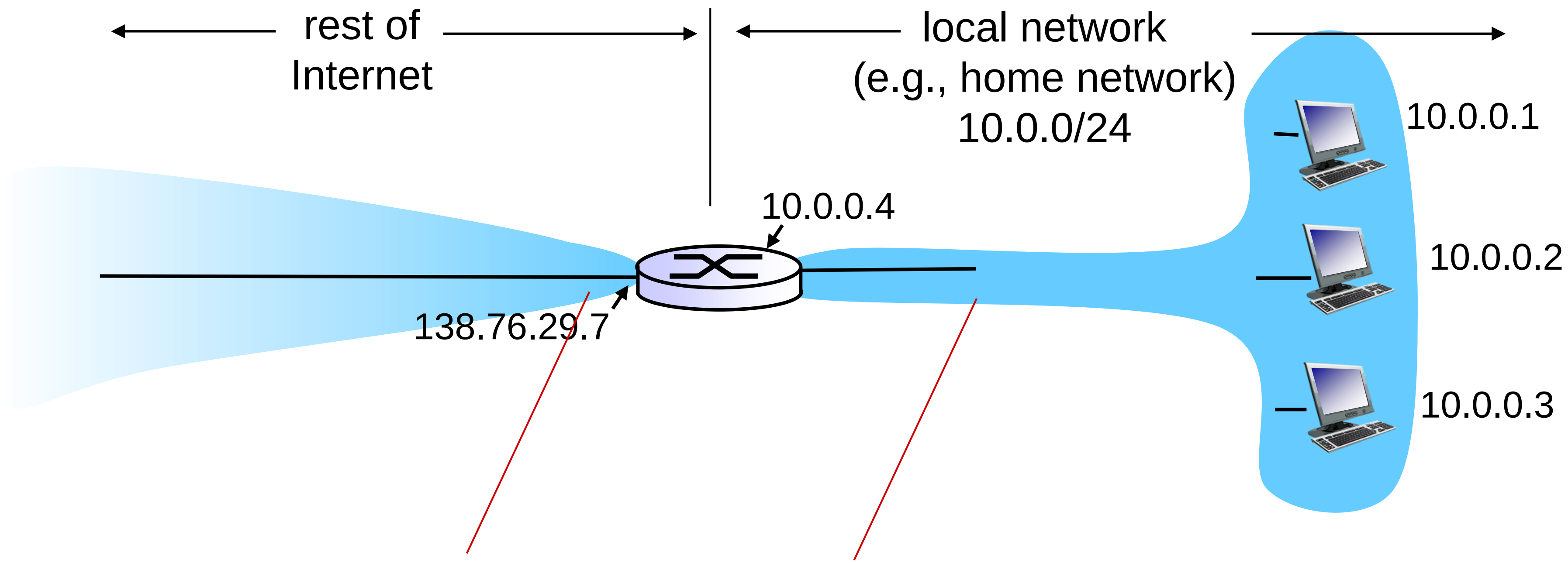
IP addressing: the last word...

Q: how does an ISP get block of addresses?

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

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

NAT: network address translation



all datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7,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

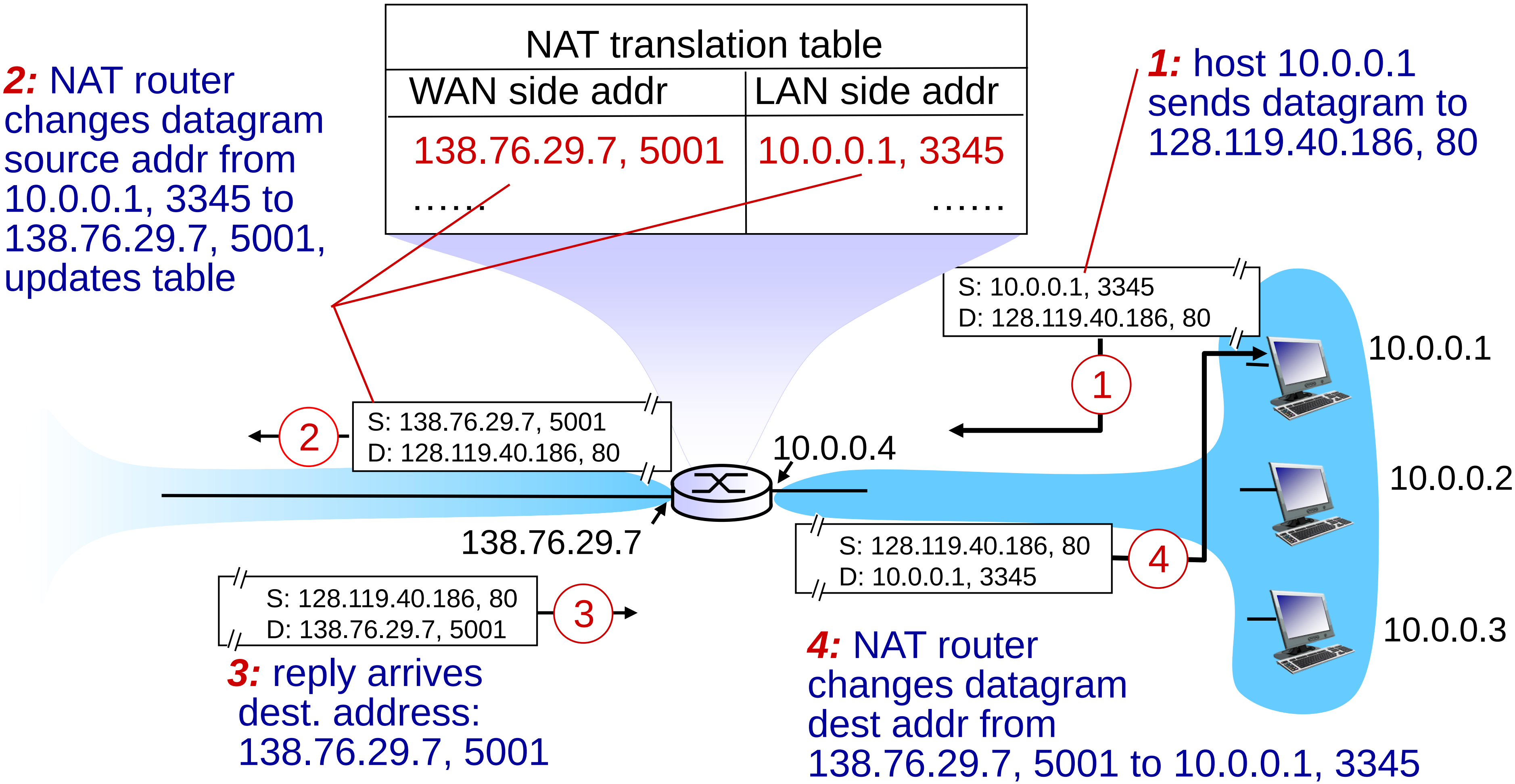
Motivation: local network uses just one IP address as far as outside world is concerned:

- range of addresses not needed from ISP: just one IP address for all devices
- can change addresses of devices in local network without notifying outside world
- can change ISP without changing addresses of devices in local network
- devices inside local net not explicitly addressable, visible by outside world (a security plus)

NAT: network address translation

NAT translation table

WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……



NAT: network address translation

Implementation: NAT router must:

- *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 addr

- *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 dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
  - routers should only process up to layer 3
  - address shortage should be solved by IPv6
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - NAT traversal: what if client wants to connect to server behind NAT?

Topics

- Overview of Network layer
  - data plane
  - control plane
- What's inside a router
- IP: Internet Protocol
  - datagram format
  - fragmentation
  - IPv4 addressing
  - network address translation
  - IPv6

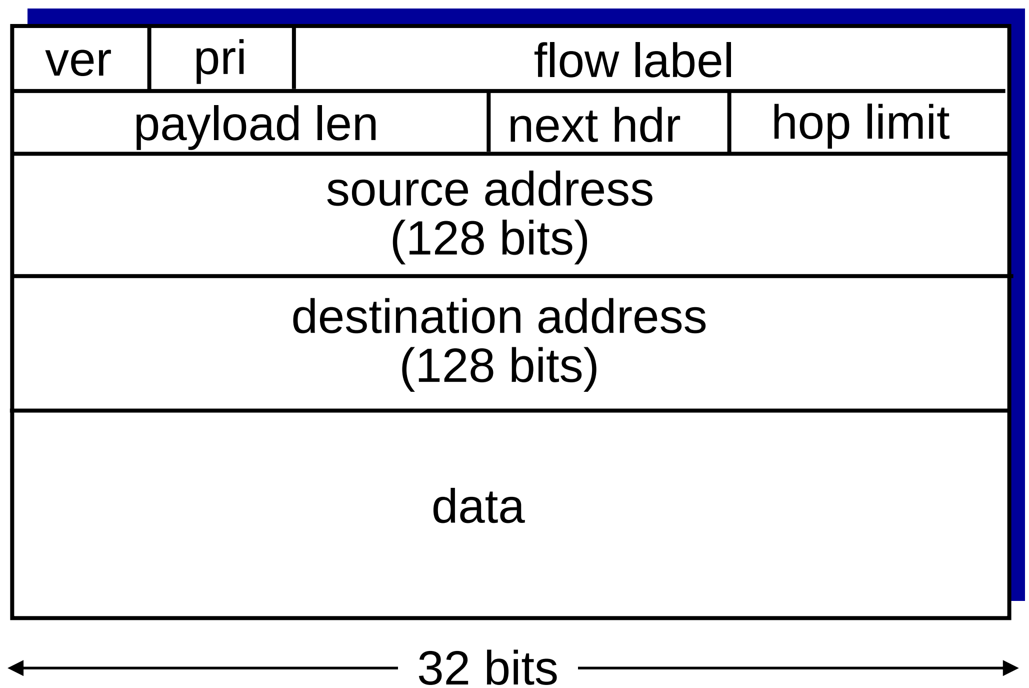
IPv6: motivation

- *Initial motivation:* 32-bit address space soon to be completely allocated.
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

- *IPv6 datagram format:*
  - fixed-length 40 byte header
  - **<u>no fragmentation allowed</u>**

IPv6 datagram format

priority: identify priority among datagrams in flow
flow Label: identify datagrams in same "flow."
(concept of"flow" not well defined).
next header: identify upper layer protocol for data

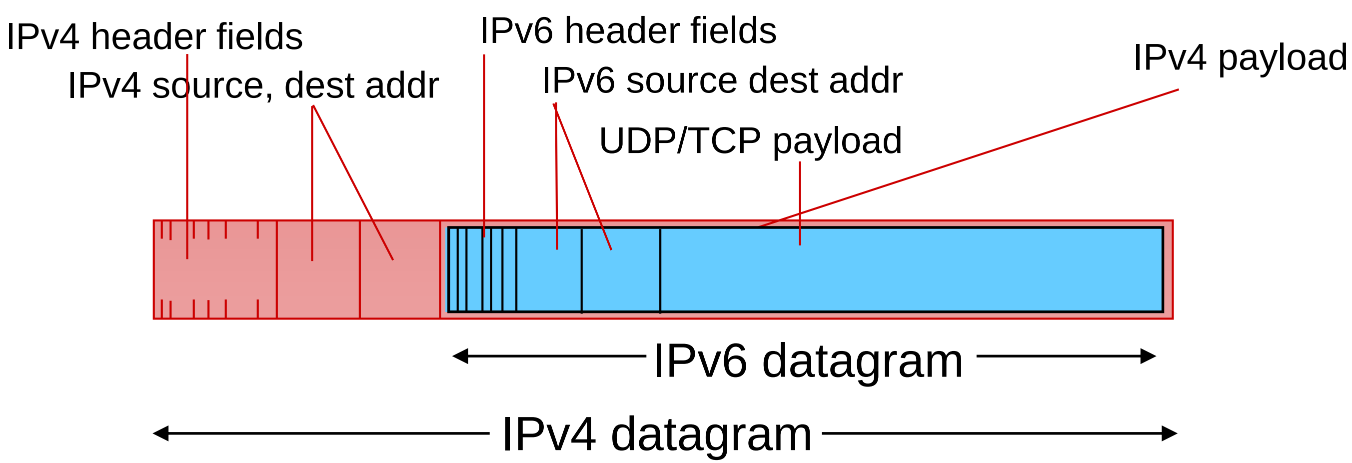


Other changes from IPv4

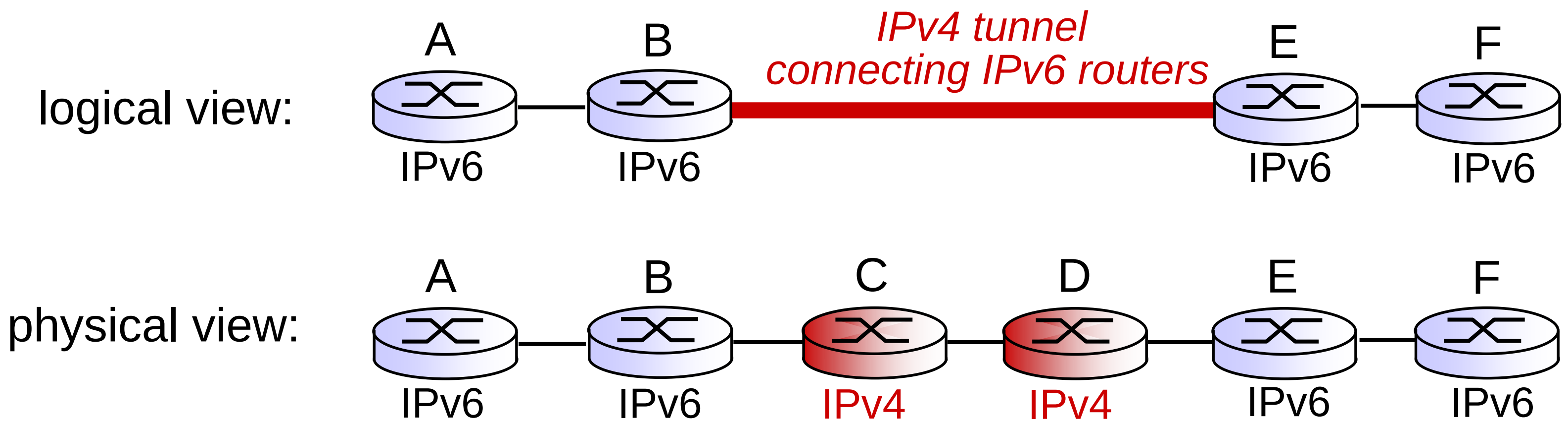
- *checksum:* removed entirely to reduce processing time at each hop
- *options:* allowed, but outside of header, indicated by "Next Header" field
- *ICMPv6:* new version of ICMP
  - additional message types, e.g. "Packet Too Big"
  - multicast group management functions

Transition from IPv4 to IPv6

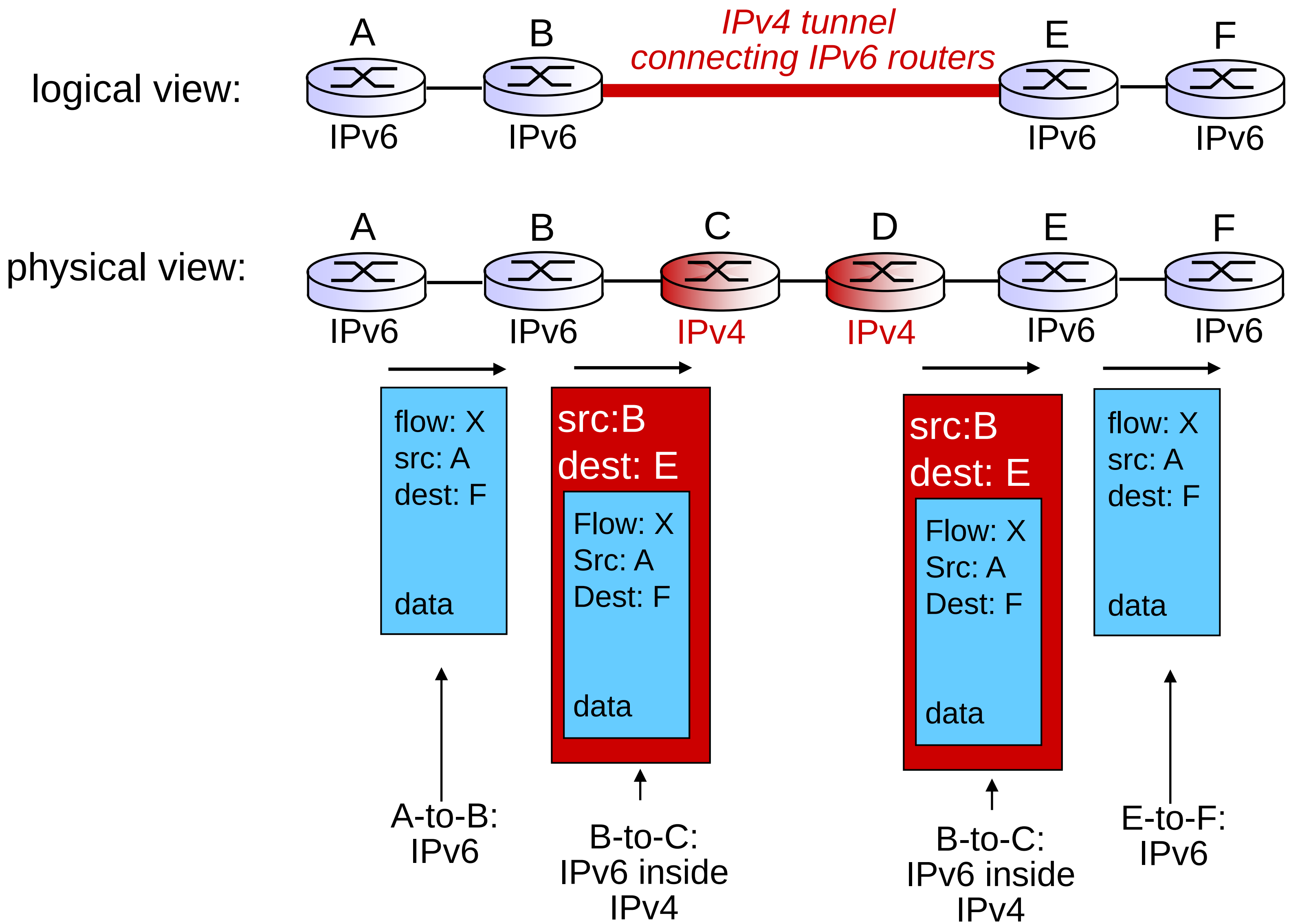
- not all routers can be upgraded simultaneously
  - no "flag days"
  - how will network operate with mixed IPv4 and IPv6 routers?
- *tunneling:* IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers



Tunneling


A
B
IPv4 tunnel
connecting IPv6 routers
E
F
logical view:
IPv6
IPv6
IPv6
IPv6
A
B
C
D
E
F
physical view:
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6

Tunneling



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

IPv6: adoption

- Carrier networks and ISPs with mobile networks leading the charge.
  - T-Mobile USA has more than 90% of its traffic going over IPv6
  - Verizon Wireless close behind at 82.25%.
  - Comcast and AT&T have its networks at 63% and 65%, respectively
- *Long (long!) time for deployment, use*
  - 20 years and counting!
  - *Why?*