Network Layer: The Data Plane

provide routing and forwarding service

routing service -> control plane (set 4)
- traditional routing
- SDN

forwarding service -> data plane (set 5)

Set 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

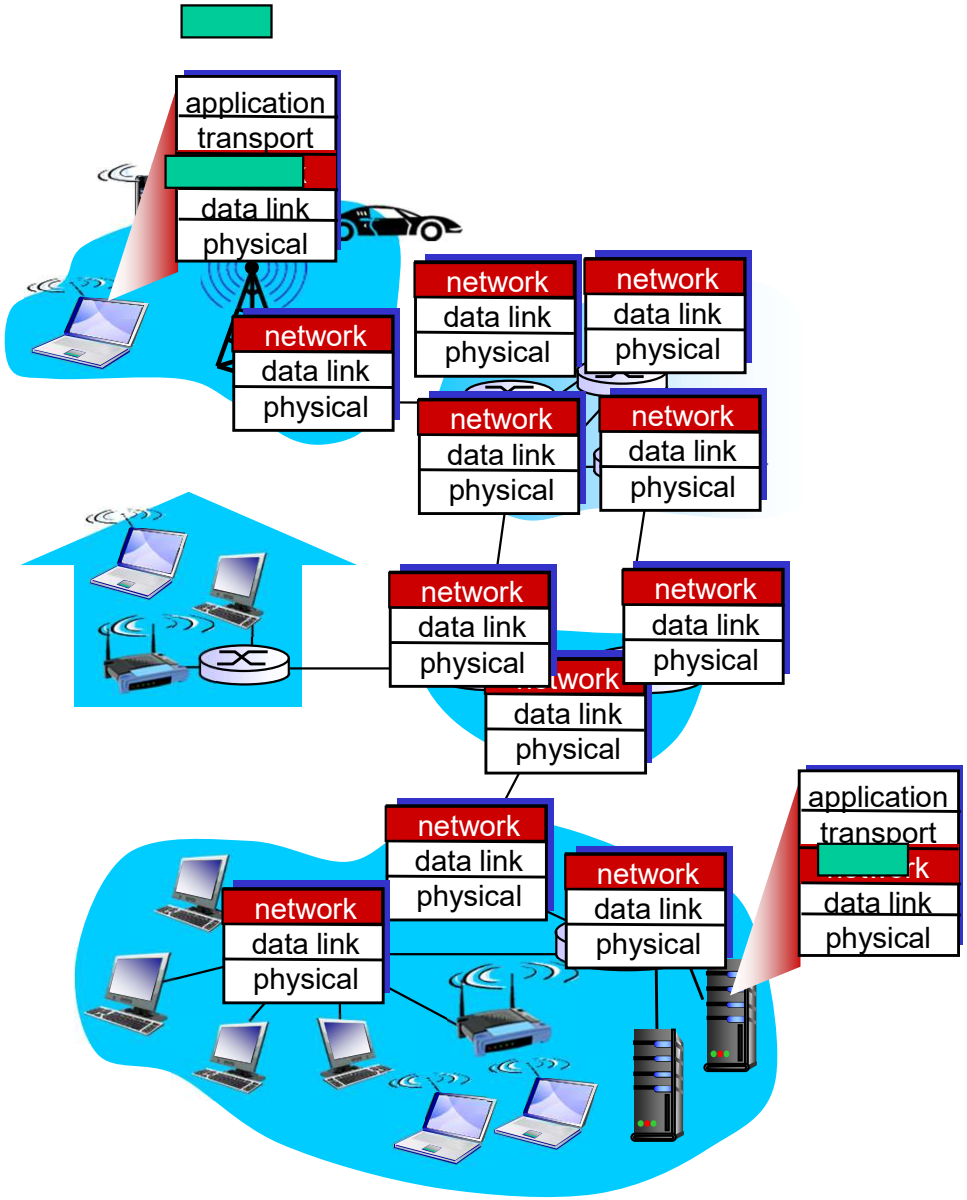
Chapter 4: network layer

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

Network layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- 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:* move packets from router's input to appropriate router output
- *routing:* determine route taken by packets from source to destination
  - *routing algorithms*

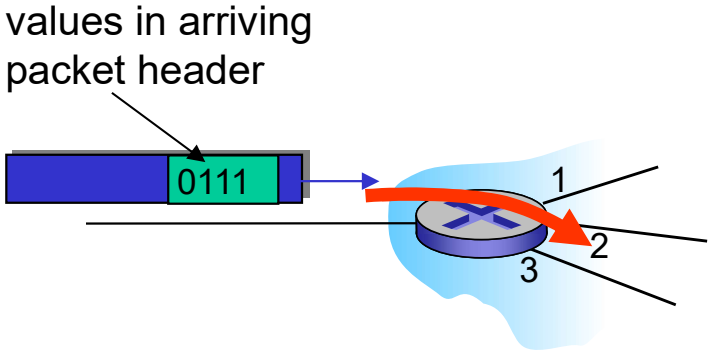
analogy: taking a trip

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

Network layer: data plane, control plane

Data plane

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

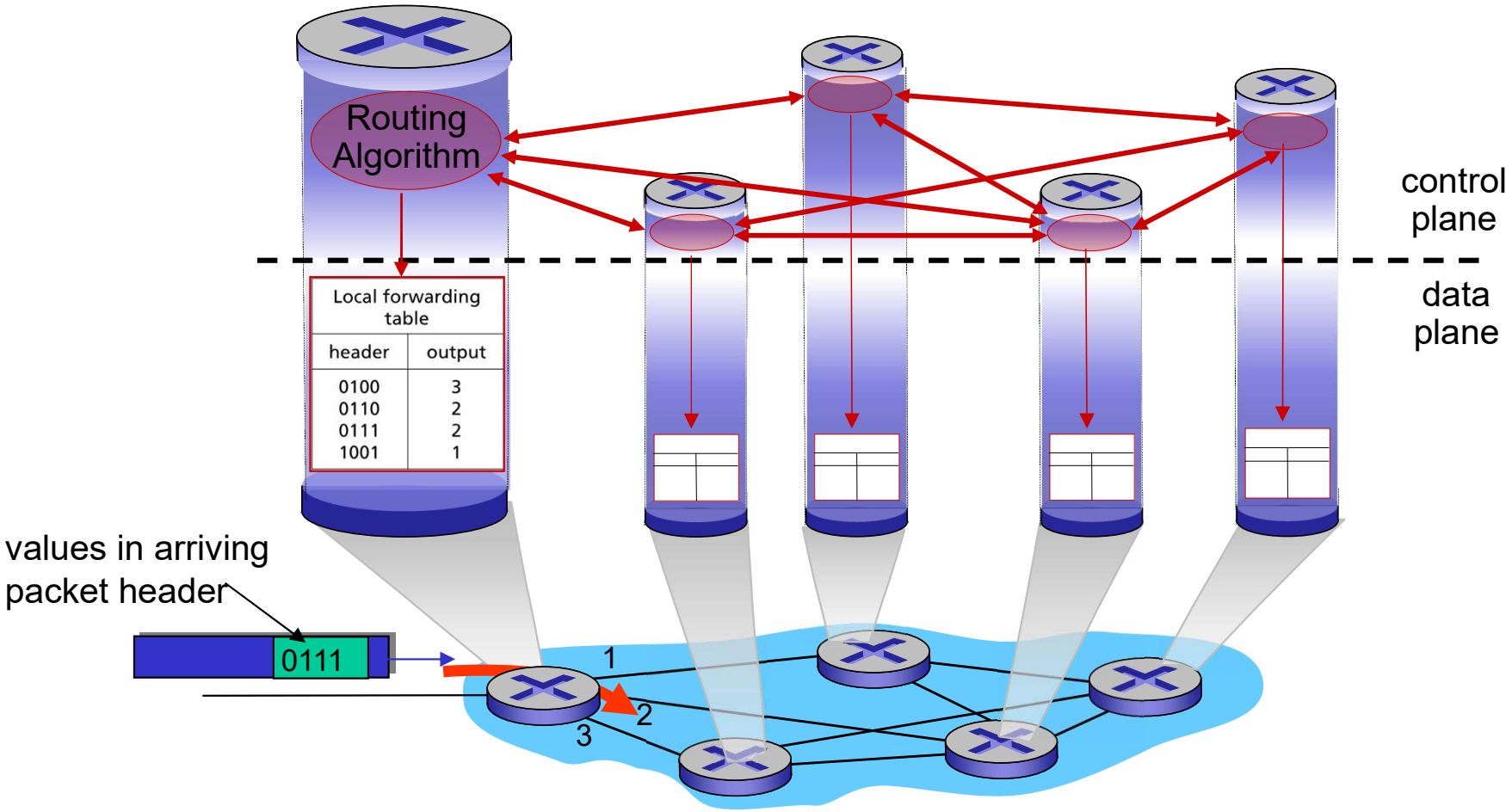


Control plane

- 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

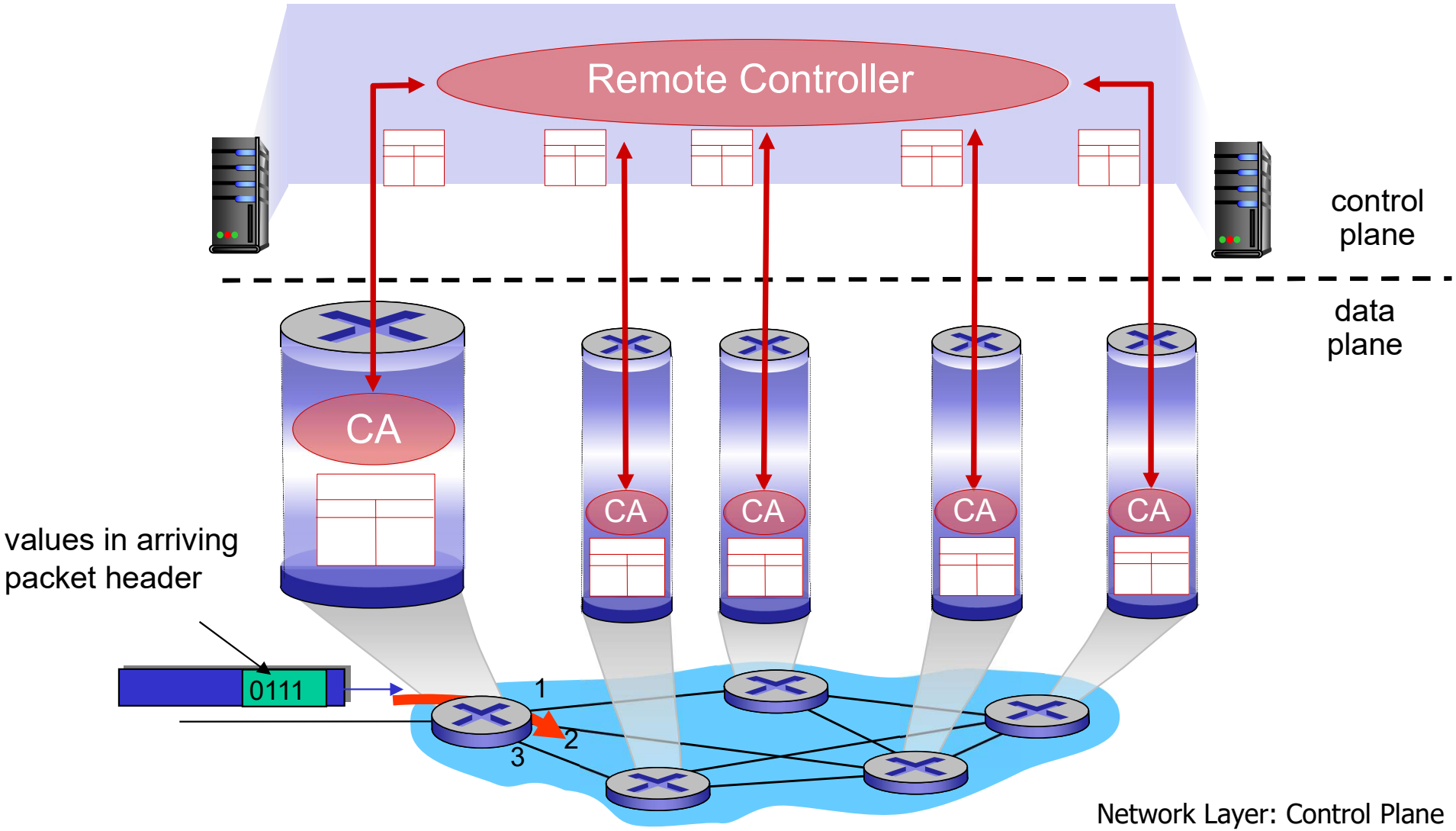
Control plane: the traditional approach

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



Control plane: the SDN approach

A distinct (typically remote) controller interacts with local control agents (CAs)



Set 4: outline

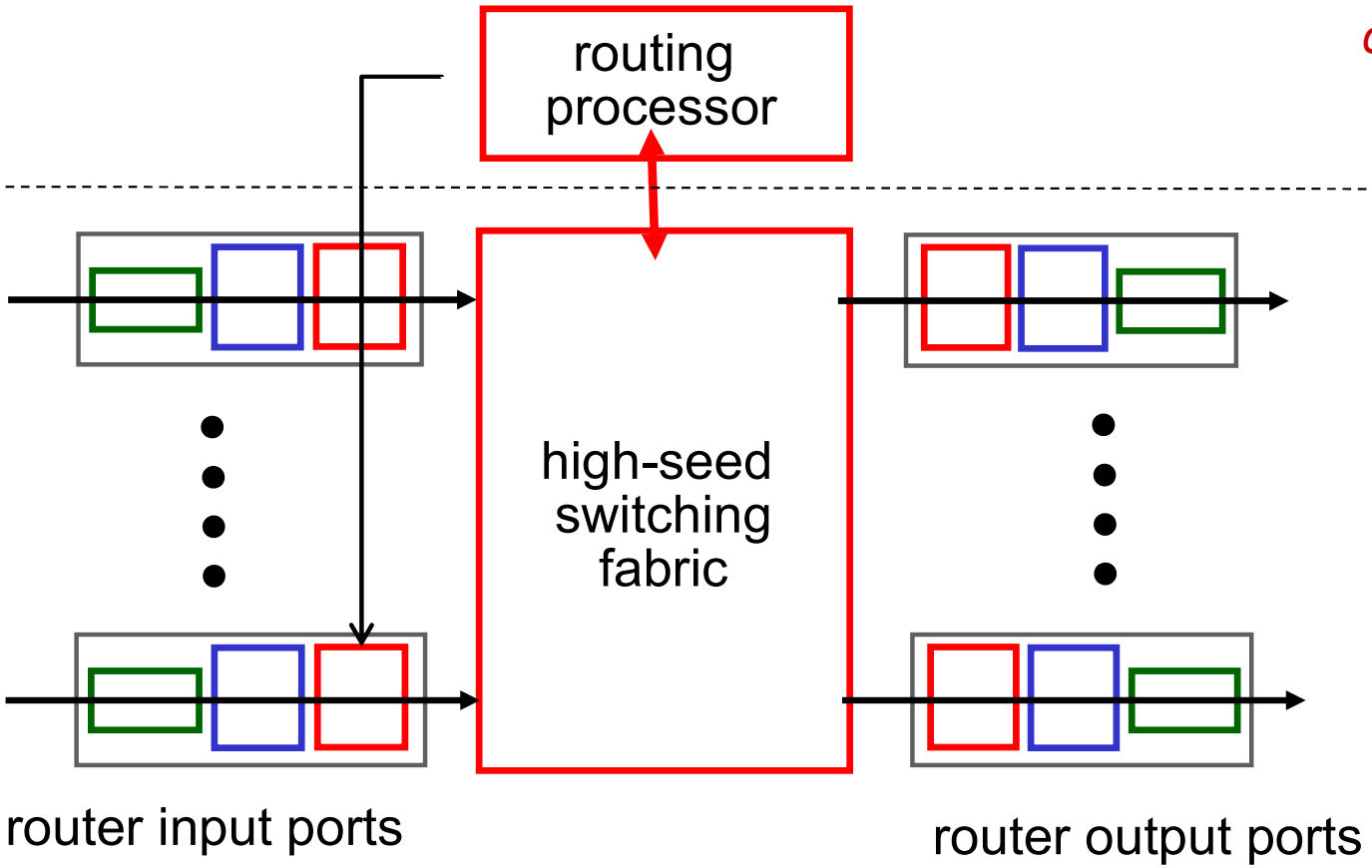
4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 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) opertes in nanosecond timeframe

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

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

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

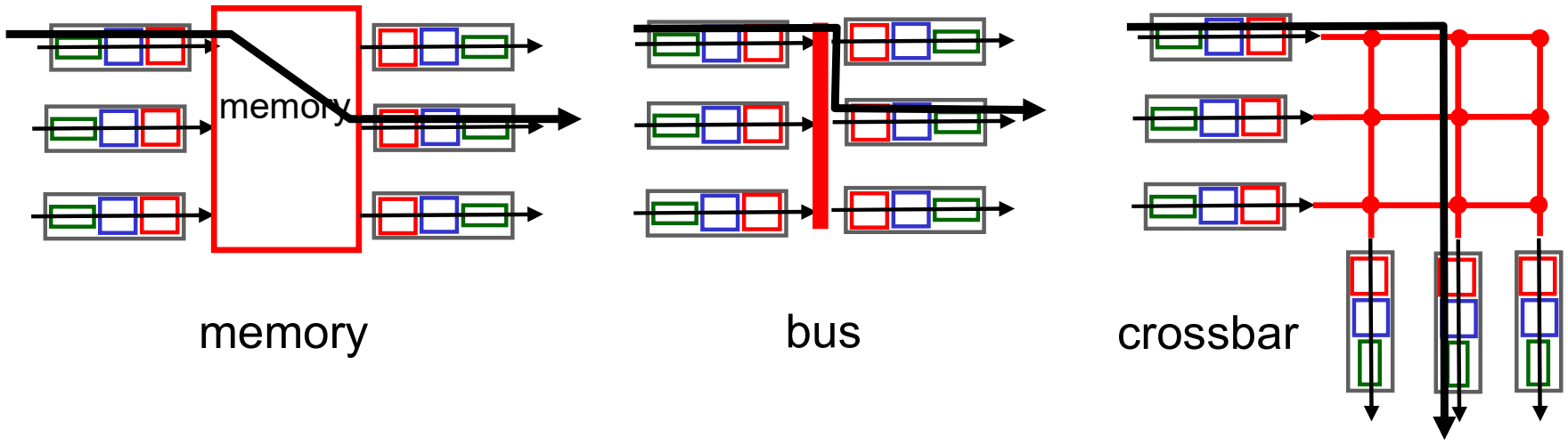
examples:

DA: 11001000 00010111 00010110 10100001 which interface?

DA: 11001000 00010111 00011000 10101010 which interface?

Switching fabrics

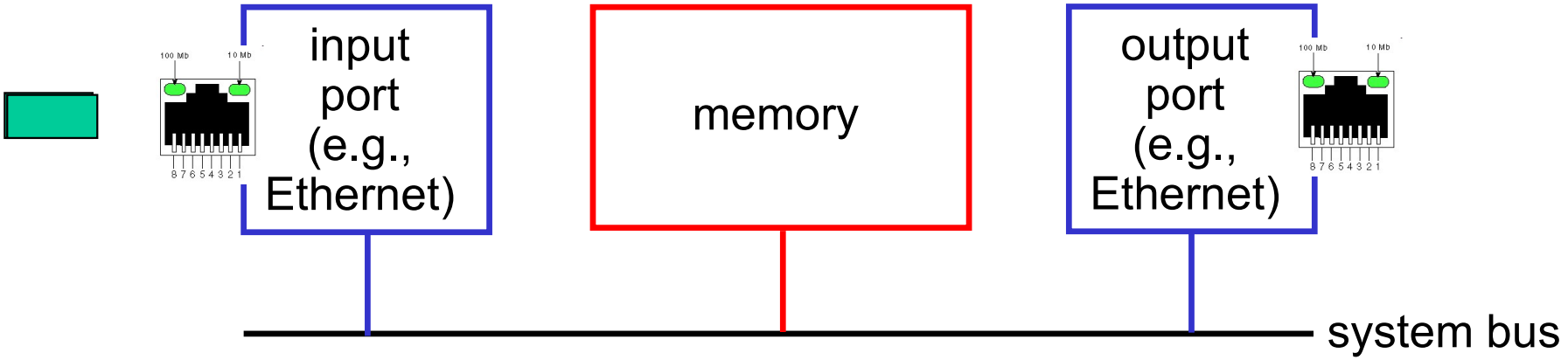
- transfer packet from input buffer to appropriate output buffer
- switching rate: rate at which packets can be transferred from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three types of switching fabrics



Switching via memory

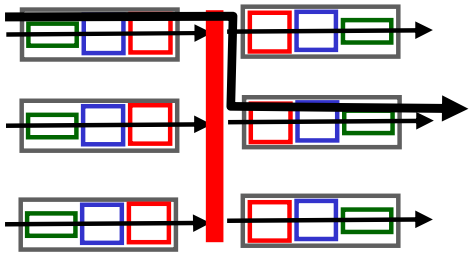
first generation routers:

- 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 a bus

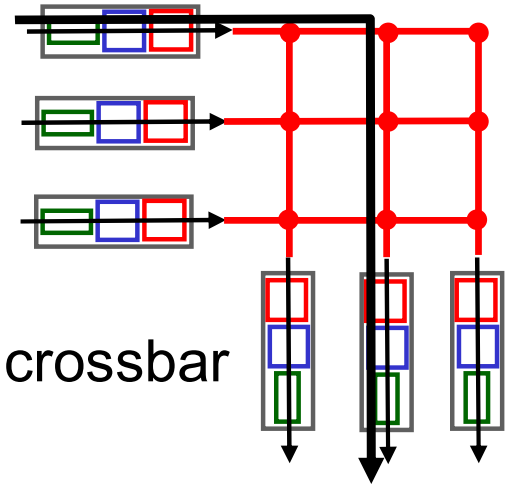
- datagram from input port memory
  to output port memory via a shared bus
- *bus contention:* switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers



bus

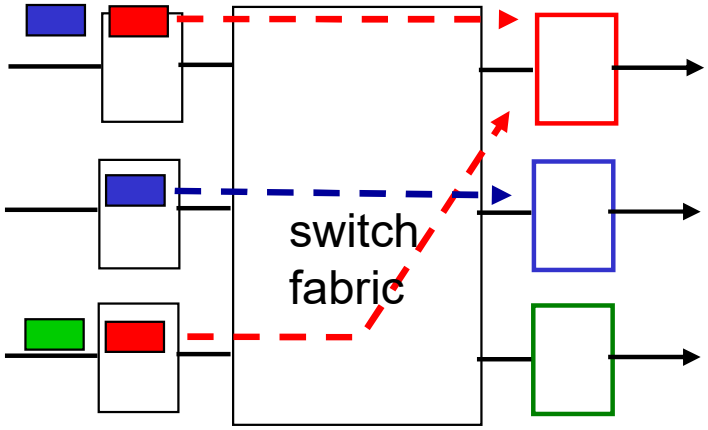
Switching via interconnection network

- overcome bus bandwidth limitations
- banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection network

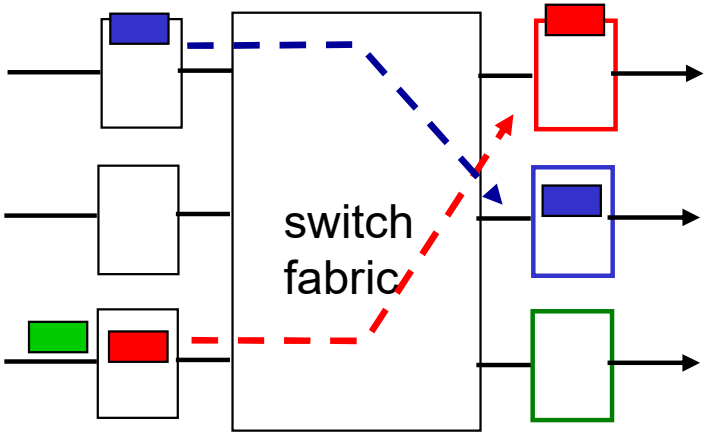


Input port queuing

- fabric is 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

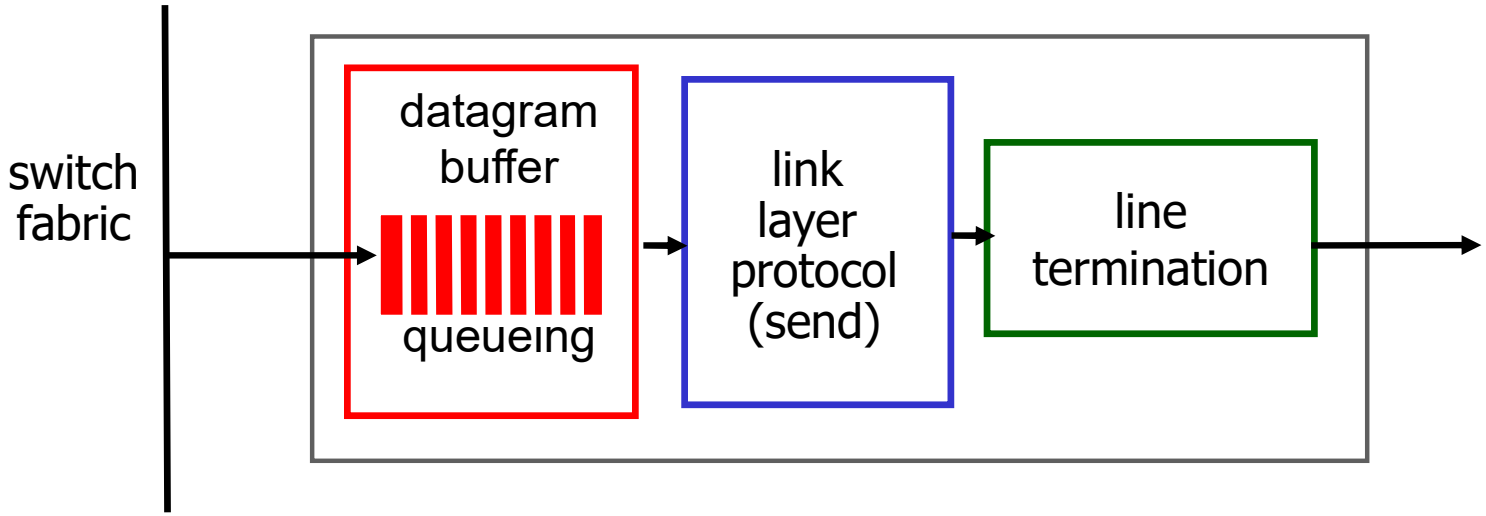


output port contention:
only one red datagram can be transferred.
lower red packet is blocked



one packet time later:
green packet
experiences HOL
blocking

Output ports



buffering required when datagrams arrive from fabric faster than the transmission rate

- *buffering* required from fabric faster rate

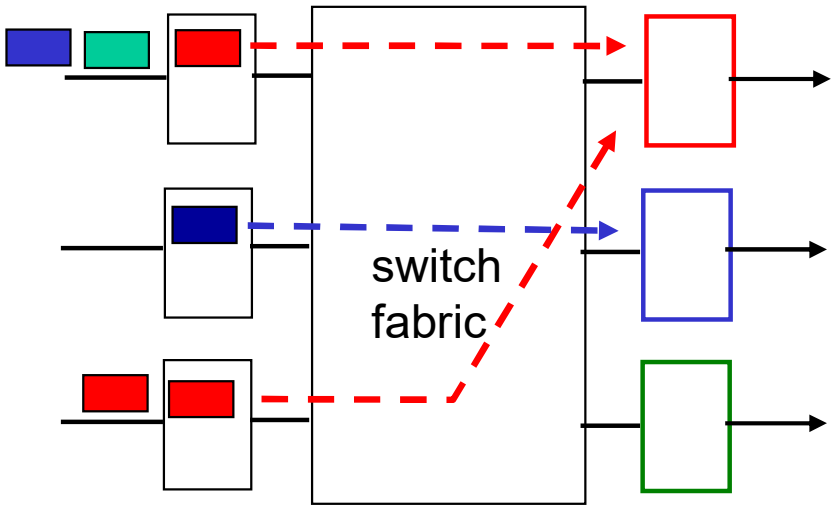
Datagram (packets) can be lost due to congestion, lack of buffers

- *scheduling* datagrams

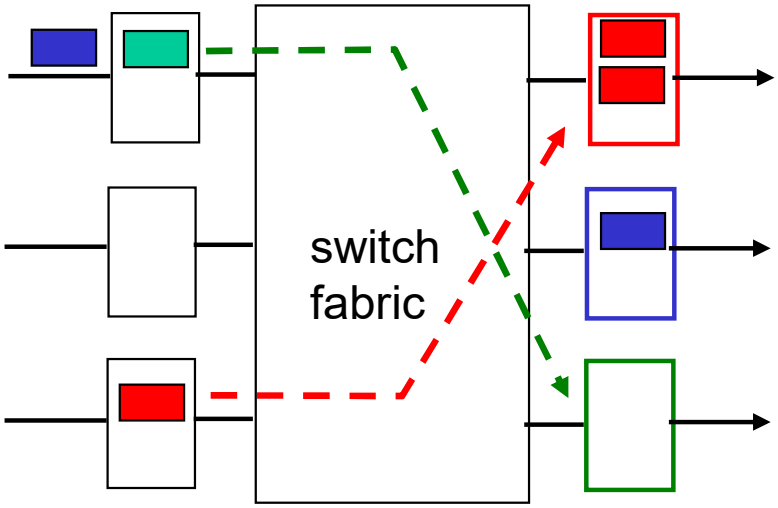
Priority scheduling – who gets best performance, network neutrality

scheduling chooses amogst queue packets for transmission/dropping

Output port queueing



at *t*, packets more from input to output

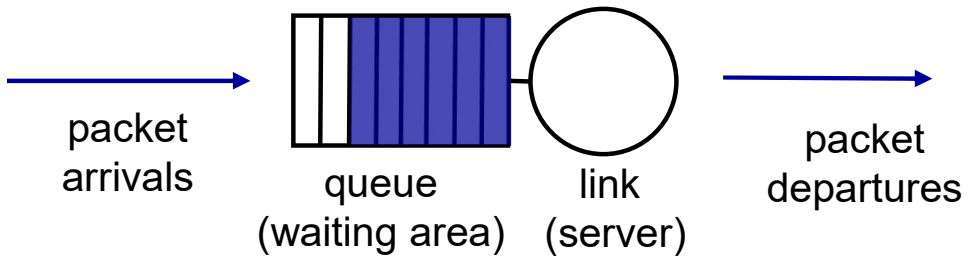


one packet time later

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

Scheduling mechanisms

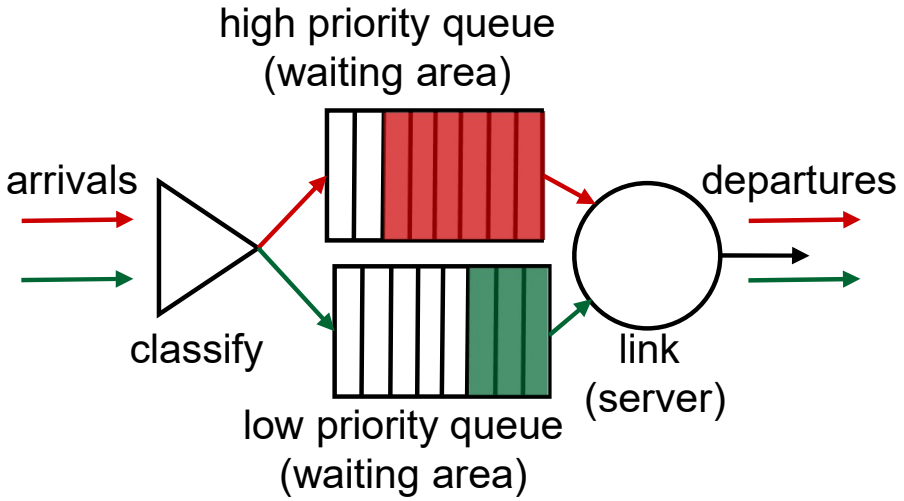
- *scheduling:* choose next packet to send on link
- *FIFO (first in first out) scheduling:* send in order of arrival to queue
  - real-world example?
  - *discard policy:* if packet arrives to full queue: who to discard?
    - *tail drop:* drop arriving packet
    - *priority:* drop/remove on priority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

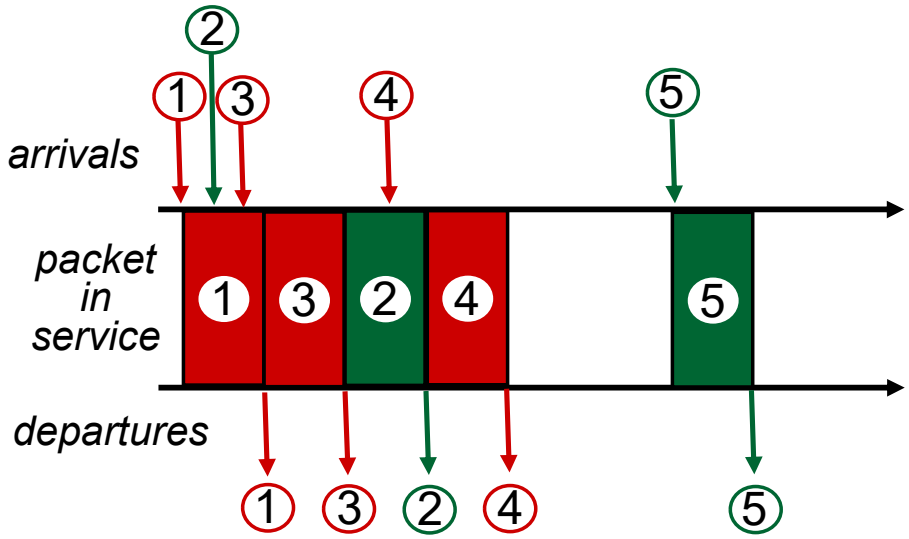


Scheduling policies: priority

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?





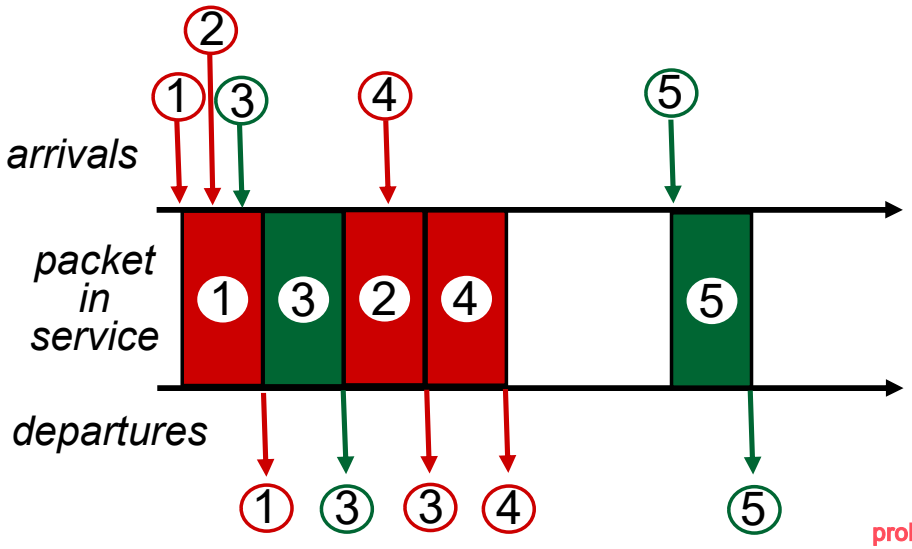
issues: can cause starvation for non hot packets

Scheduling policies: still more

Round Robin (RR) scheduling:

this solves problem of starvation in priority scheduling

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



mulitple queues

high p q1: p1, p2

less p q2: p3, p5

less pp q3: p4, p6

take one packet from each queue and put it on the link

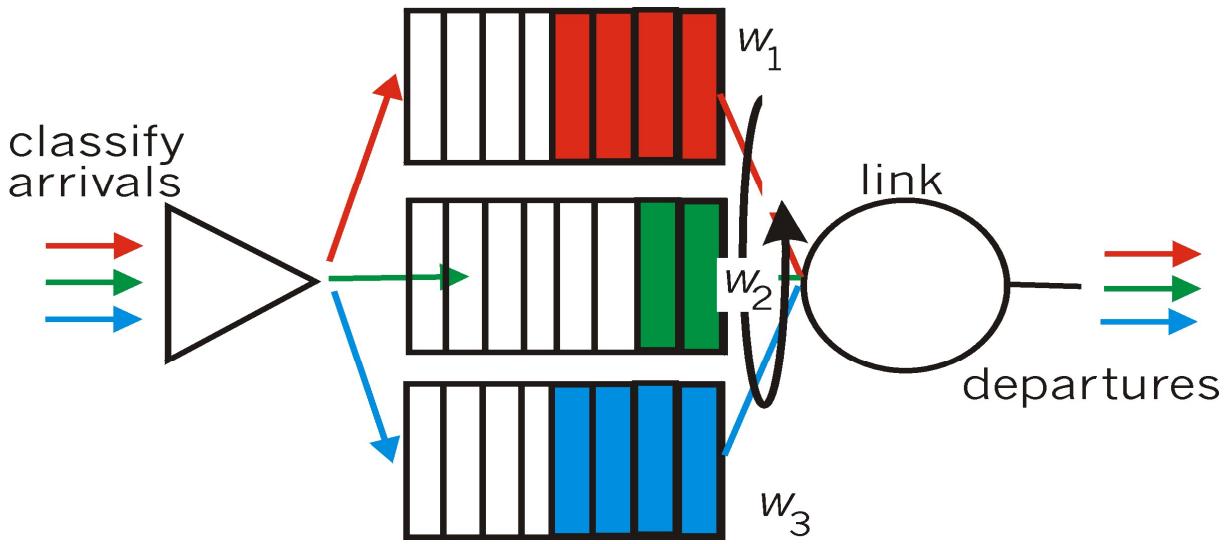
problem: this is not fair to queues that have more packets

Scheduling policies: still more

Weighted Fair Queuing (WFQ):

- generalized Round Robin
- each class gets weighted amount of service in each cycle
- real-world example?

this solves fairness problem of round robin

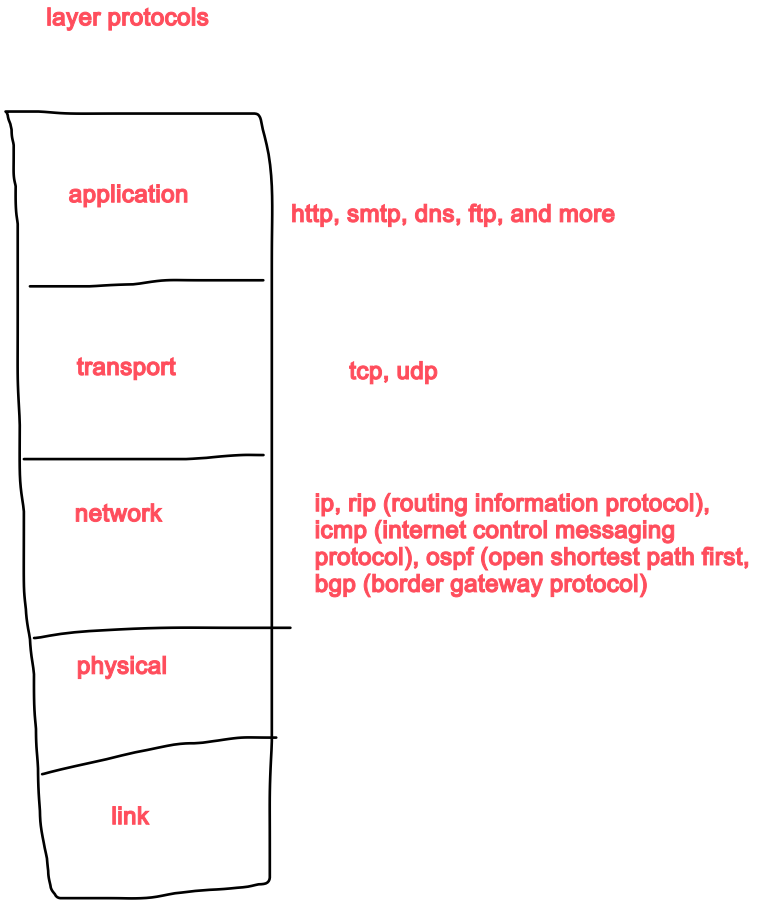


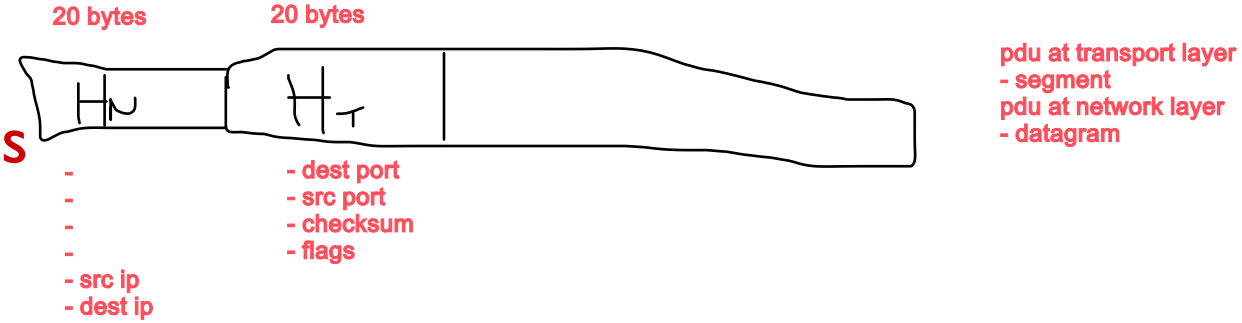
Set 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

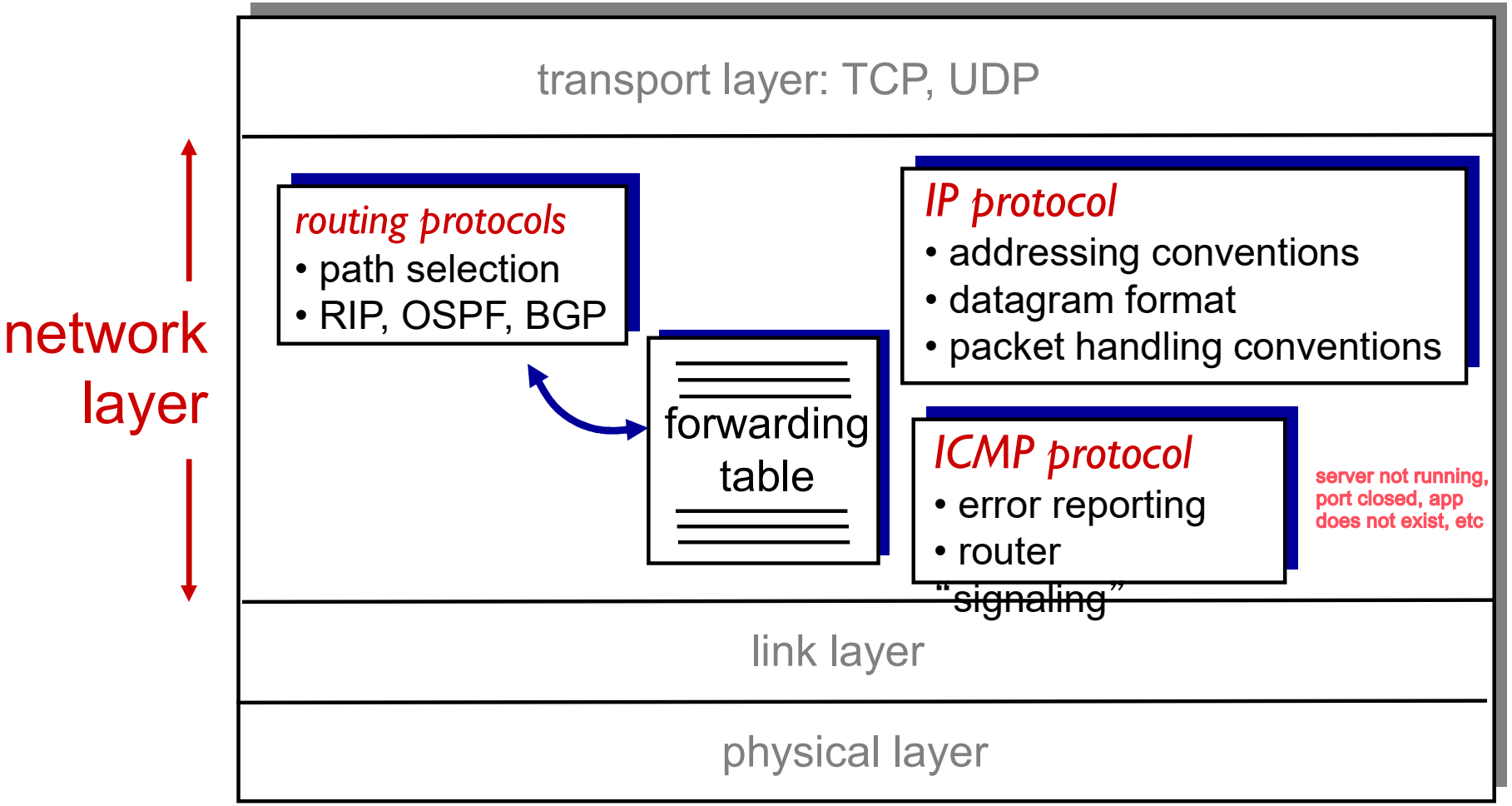
4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6



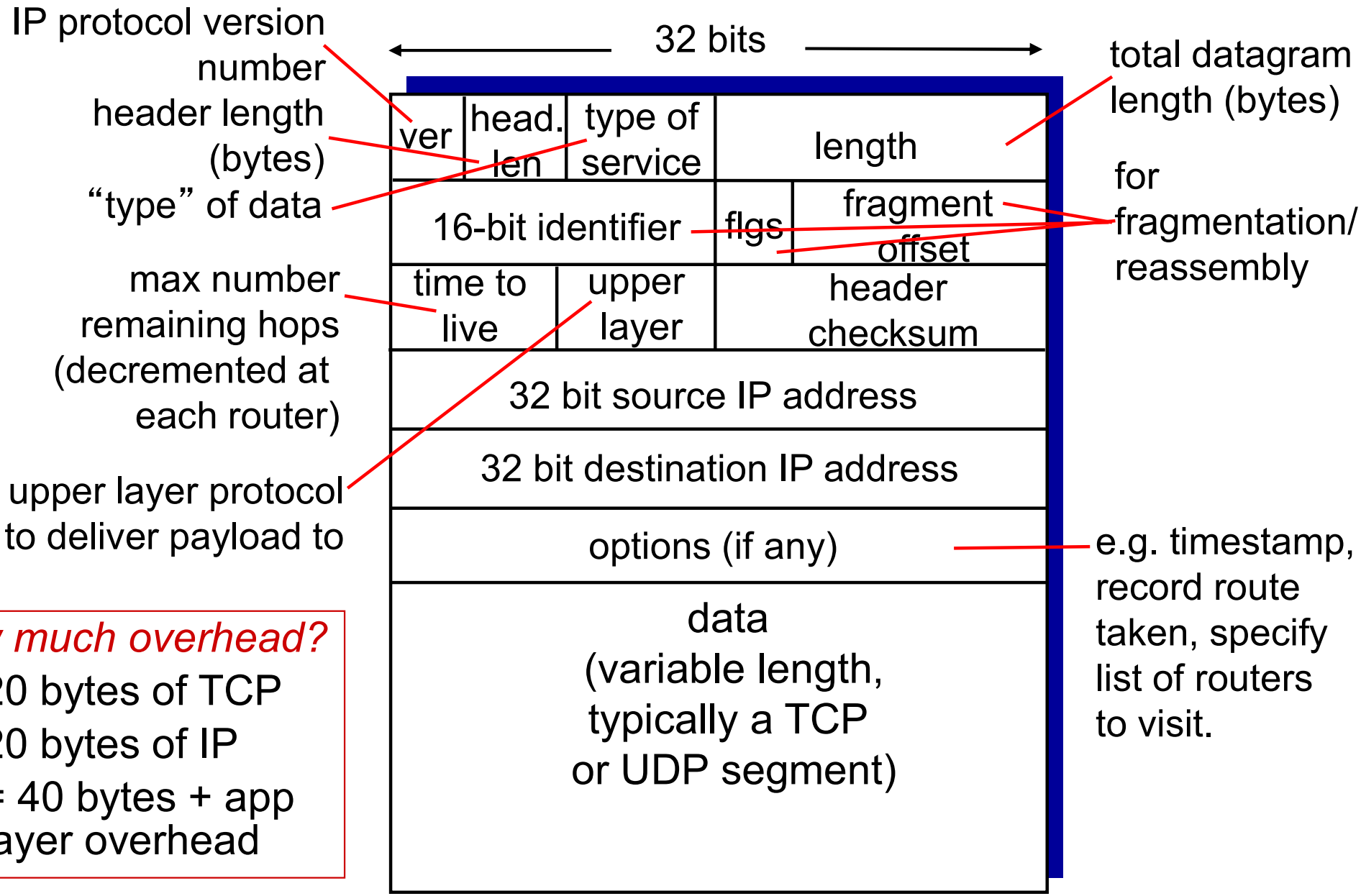


The Internet network layer

host, router network layer functions:



IP datagram format

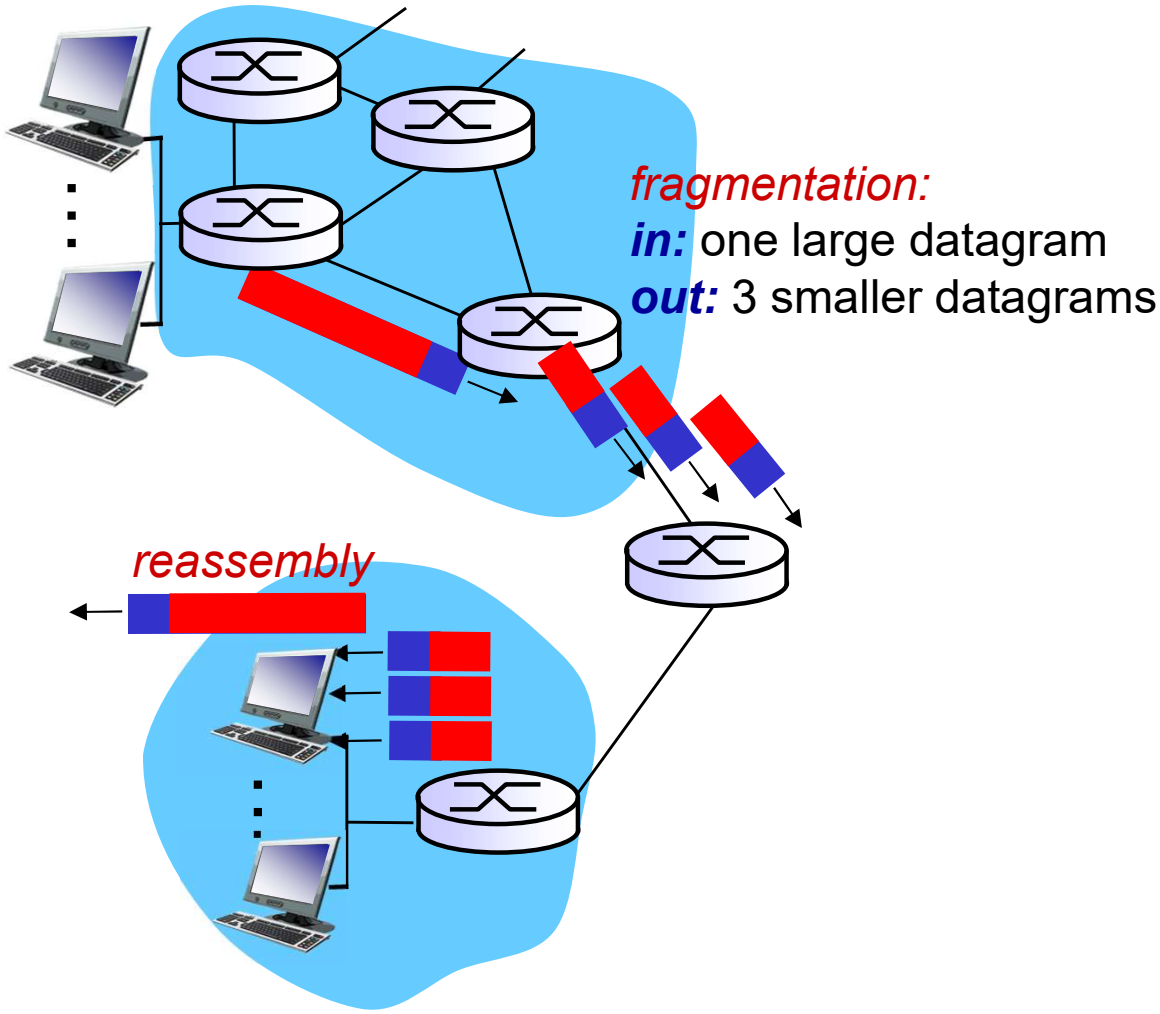


how much overhead?

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

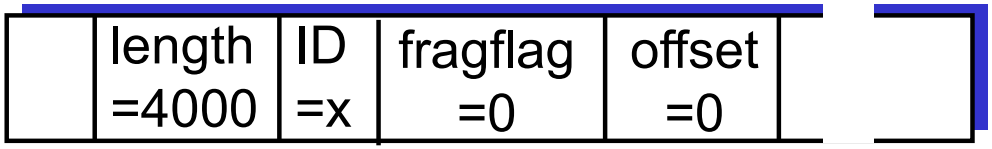
IP fragmentation, reassembly

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



IP fragmentation, reassembly

offset: how many 8 bytes in front of you (paragraphs)



example:

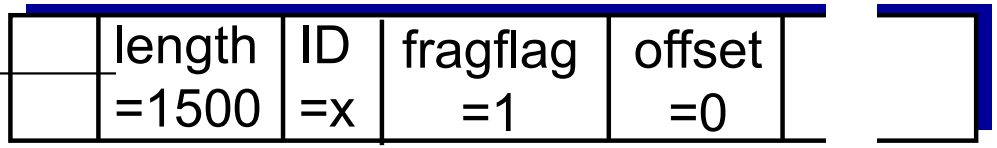
- 4000 byte datagram
- MTU = 1500 bytes

one large datagram becomes several smaller datagrams

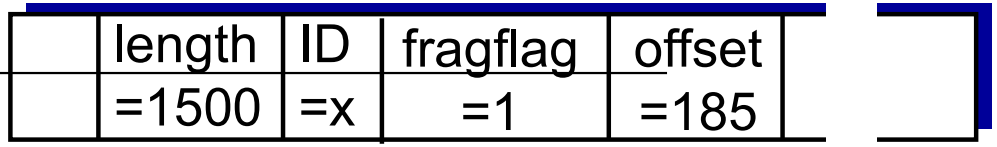
1480 bytes in data field

offset = 1480/8

header 20 bytes, data 1480 bytes



header 20 bytes, data 1480 bytes



header 20 bytes, data 1020 bytes

- IPv6 *does not* allow for fragmentation

Set 4: outline

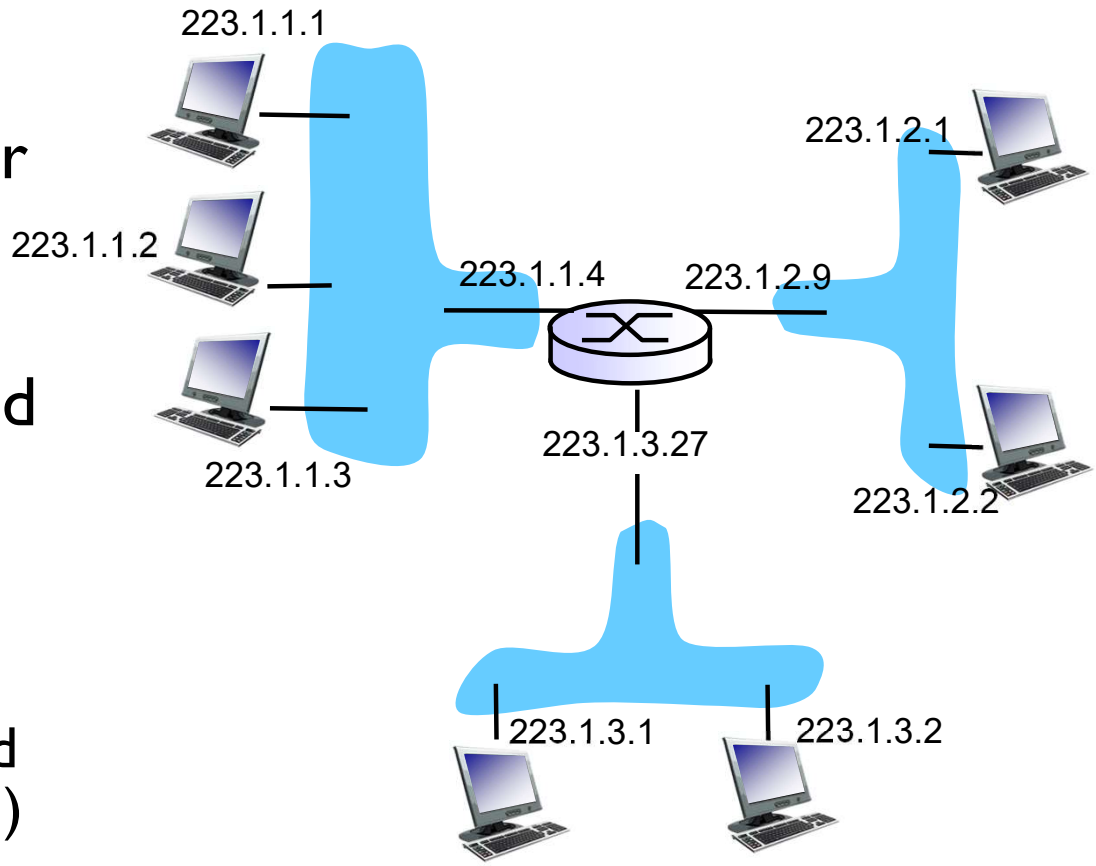
4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

IP addressing: introduction

- *IP address:* 32-bit (v4) 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*

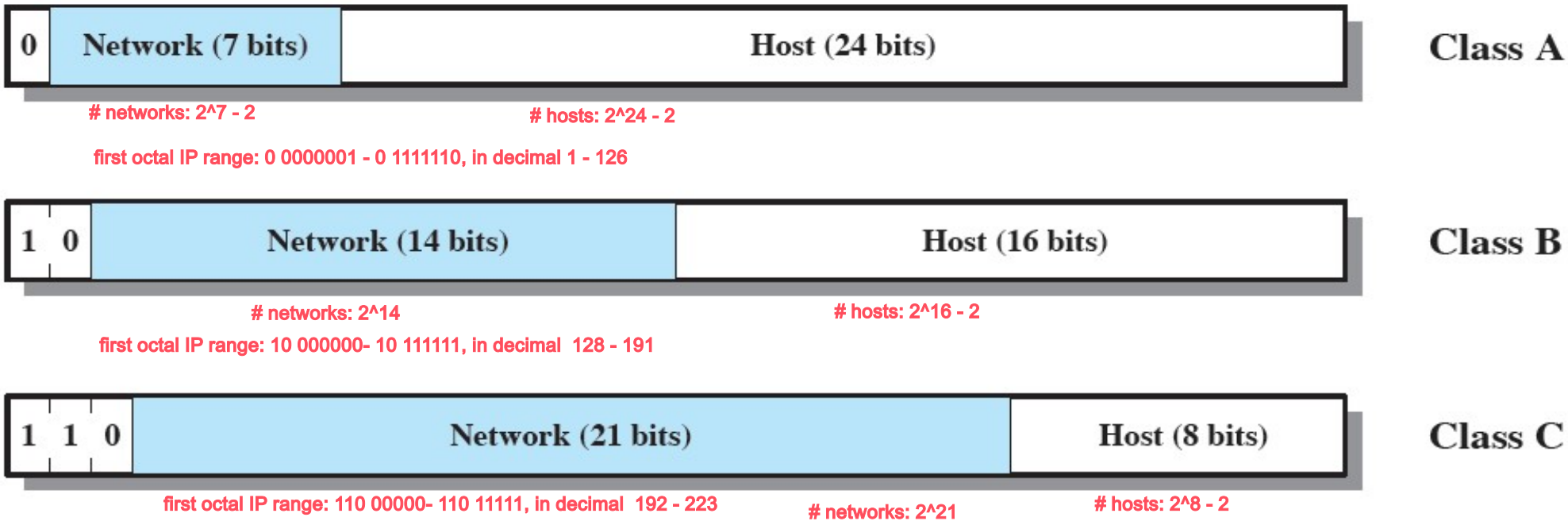


223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

IP addressing

- The source and destination address fields in the IP header each contains 32-bits global internet address, generally consisting of a *network identifier* and a *host identifier*



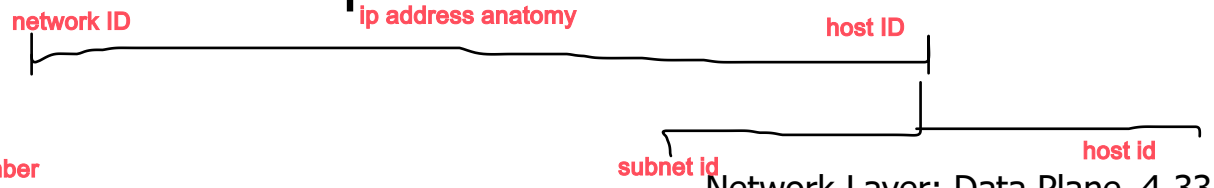
- **Class A**: few networks, each with many hosts
- **Class B**: Medium number of networks, each with a medium number of hosts
- **Class C**: Many networks, each with a few hosts

IP addressing

- IP addresses are written in **dotted decimal notation,** with a decimal number representing each of the octets of the 32-bit address. For example, the IP address 11000000 11100100 00010001 00111001 is written as 192.228.17.57
- Class **A** network addresses begin with a binary 0. Network addresses with a first octet of 0 (binary **0**0000000) and 127 (binary **0**1111111) are reserved, so there are **126** potential Class **A** network numbers, which have a first dotted decimal number in the range **1** to **126**
- Class **B** network addresses begin with a binary 10, so that the range of the **first decimal** number in a Class B address is **128** to **191** (binary 10000000 to 10111111)
- There are 2^{14} = 16,384 Class B network addresses
- For Class **C** addresses, the **first decimal** number ranges from **192** to **223** (binary 11000000 to 11011111). The total number of Class **C** network addresses is 2^{21} = 2,097,152

Subnets and Subnet Masks

- Consider an internet that includes one or more WANs and a number of sites, each of which has a number of LANs
- We would like to insulate the overall internet against explosive growth in network numbers and routing complexity
- Assign a single network number to ALL of the LANs at a site
- Each LAN is assigned a subnet number
- The *host* portion of the IP address is partitioned into a subnet number and a host number
- The local router must route on the basis of an extended network number consisting of the network portion of the IP address and the subnet number
- The address mask indicates the bit positions of this extended network number



network ID and subnet ID is the extended network number

subnet mask: the parameter that tells you how the host ID is split up

Subnets and Subnet Masks

IPv4 Addresses and Subnet Masks

(a) Dotted decimal and binary representations of IPv4 address and subnet masks

	Binary Representation	Dotted Decimal
IP address	11000000.11100100.00010001.00111001	192.228.17.57
Subnet mask	11111111.11111111.11111111.11100000	255.255.255.224
Bitwise AND of address and mask (resultant network/subnet number)	11000000.11100100.00010001.00100000	192.228.17.32
Subnet number	11000000.11100100.00010001.001	1
Host number	00000000.00000000.00000000.00011001	25

ip: 192.10.17.22

mask: 255.255.255.240

240 in binary => 1111 0000

22 in binary => 0001 0110

this means the first 4 bits of host ID are subnet ID, so,

subnet ID: 0001 = 1
host #: 0110 = 6

and extended network # = 192.10.17.1

ip: 130.157.224.240

mask: 255.255.255.240

224 in binary => 1110 0000
240 in binary => 1111 0000

240 in binary => 1111 0000

this means the first 4 bits of host ID are subnet ID, so,

subnet ID: 1111 = 1
host #: 0000 = 0

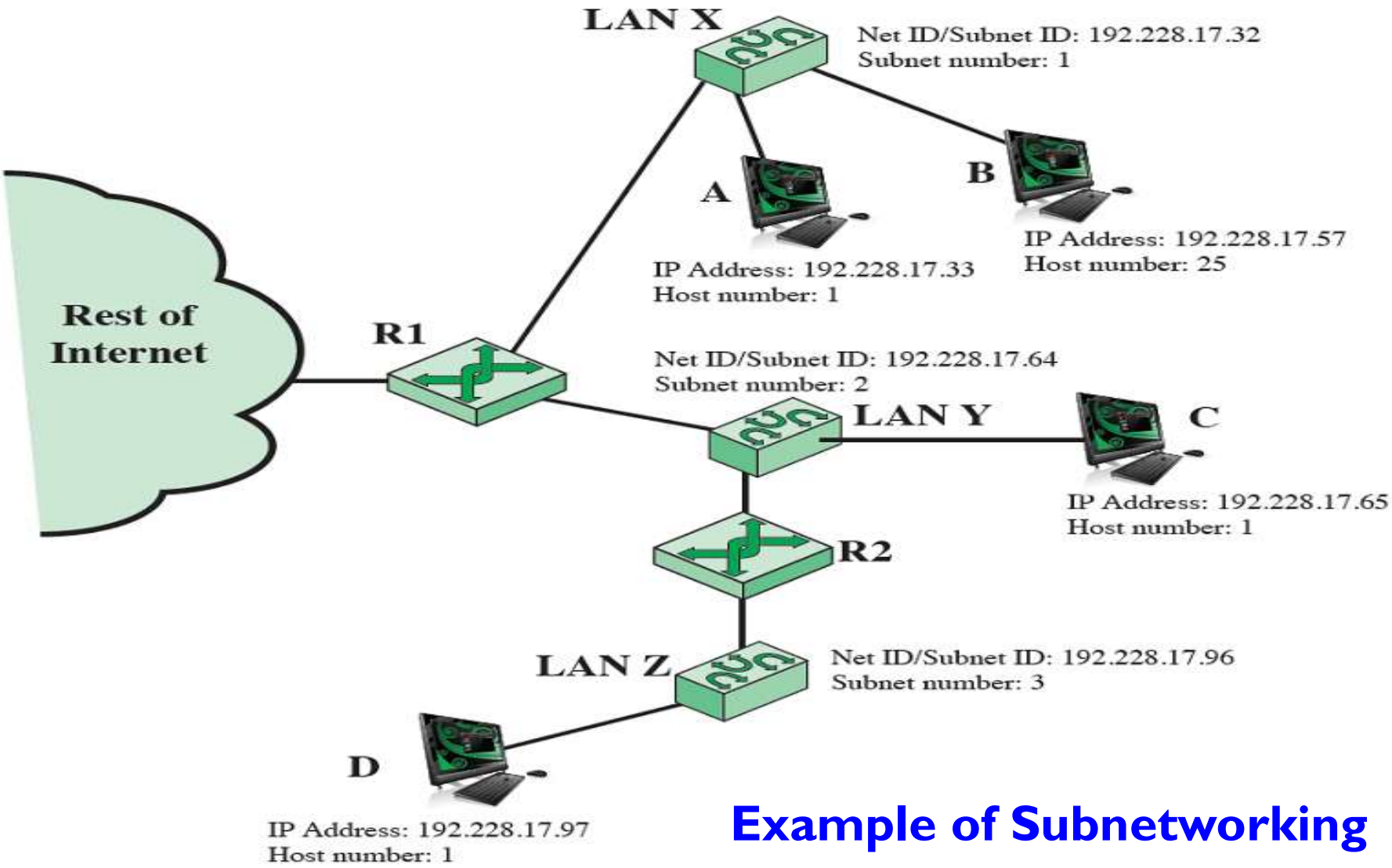
and extended network # = 130.157.224.15

Subnets and Subnet Masks

(b) Default subnet masks

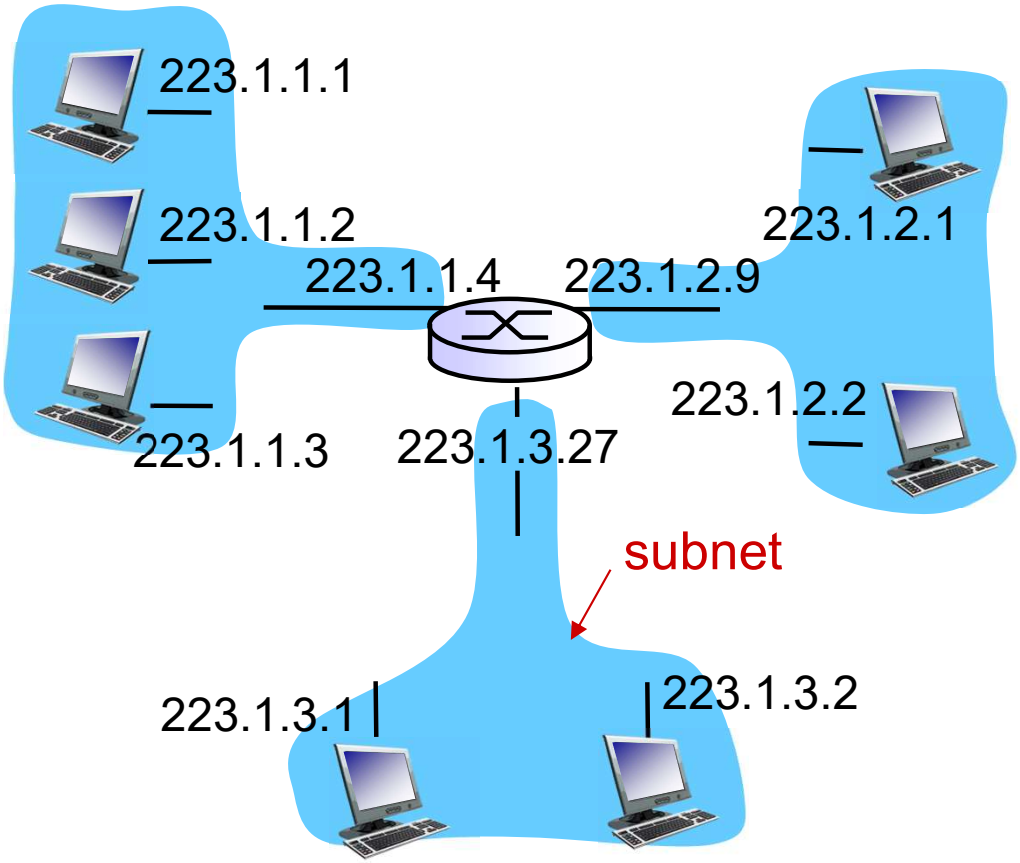
	Binary Representation	Dotted Decimal
Class A default mask	11111111.00000000.00000000.00000000	255.0.0.0
Example Class A mask	11111111.11000000.00000000.00000000	255.192.0.0
Class B default mask	11111111.11111111.00000000.00000000	255.255.0.0
Example Class B mask	11111111.11111111.11111000.00000000	255.255.248.0
Class C default mask	11111111.11111111.11111111.00000000	255. 255. 255.0
Example Class C mask	11111111.11111111.11111111.11111100	255. 255. 255.252

Subnets and Subnet Masks



Example of Subnetworking

Subnets

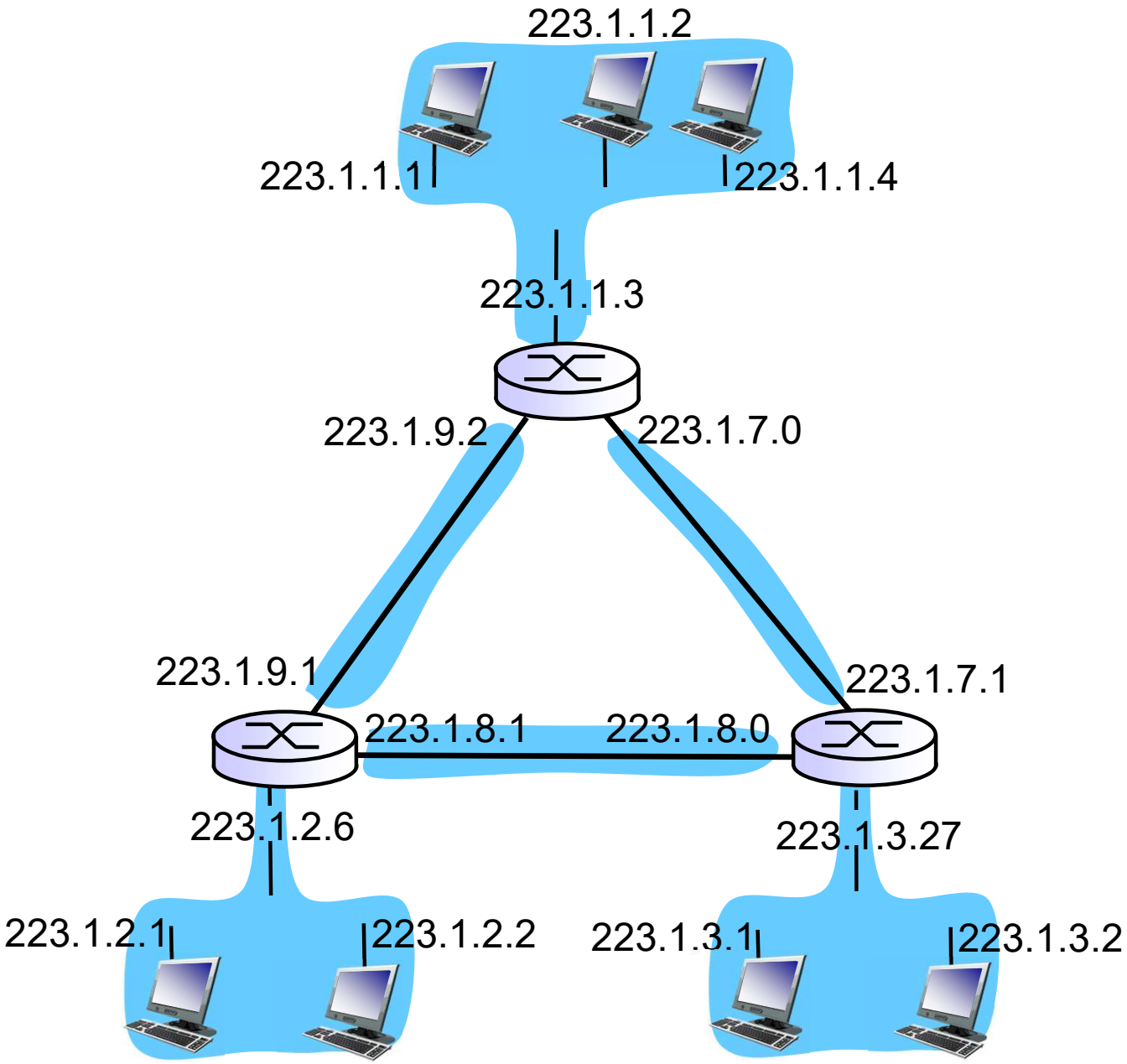


network consisting of 3 subnets

Subnets

how many?

6 subnets



Exercise

- Exercise:

  Given a network address of 192.168.100.0 and a subnet mask of 255.255.255.192,

  192 = 1100 0000

  class c

  a) How many subnets are created?

  since 2 one bits in subnet mask (1100 0000), then 4 subnets created

  b) How many hosts are there per subnet?

  since 6 zero bits in subnet mask (1100 0000), then 2^6-2 hosts created (62)

- Exercise:

  Given a company with six individual departments and each department having ten computers or networked devices, what mask could be applied to the company network to provide the subnetting necessary to divide up the network *equally*?

  six departments -> 6 subnets

  ten computers -> minimum 10 hosts

  1110 0000 -> 255.255.255.224 for class c (not equally)
  1111 0000 -> 255.255.255.240 for class c (is equally)

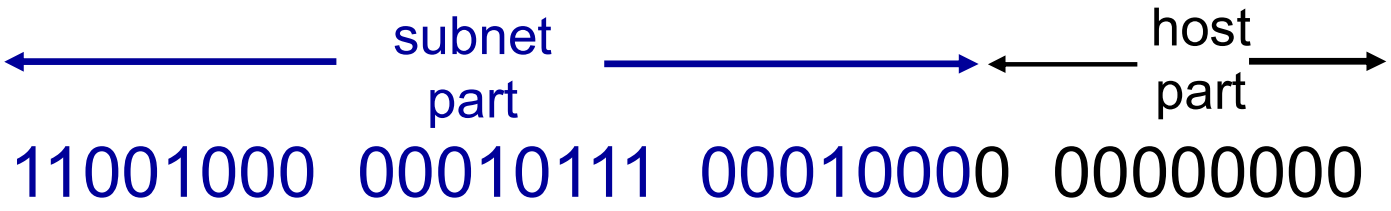
  equally -> each subnet should have max the same amount of hosts as subnets
  (e.g. 16 subnets that hold 16 hosts)

IP addressing: CIDR

do not consider class, no such class a or b or c

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



200.23.16.0/23

23 bits for the subnet, so the host will be 9 bits

so 2^9 - 2 = 510 possible hosts

IP broadcast address

- IP broadcast address 255.255.255.255
- When a host sends a datagram with destination address 255.255.255.255, the message is delivered to all hosts on the same subnet.
- Routers optionally forward the message into neighboring subnets as well (although they usually don't).

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 a server
  - "plug-and-play"

DHCP is protocol in application layer

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 is a four-step process:

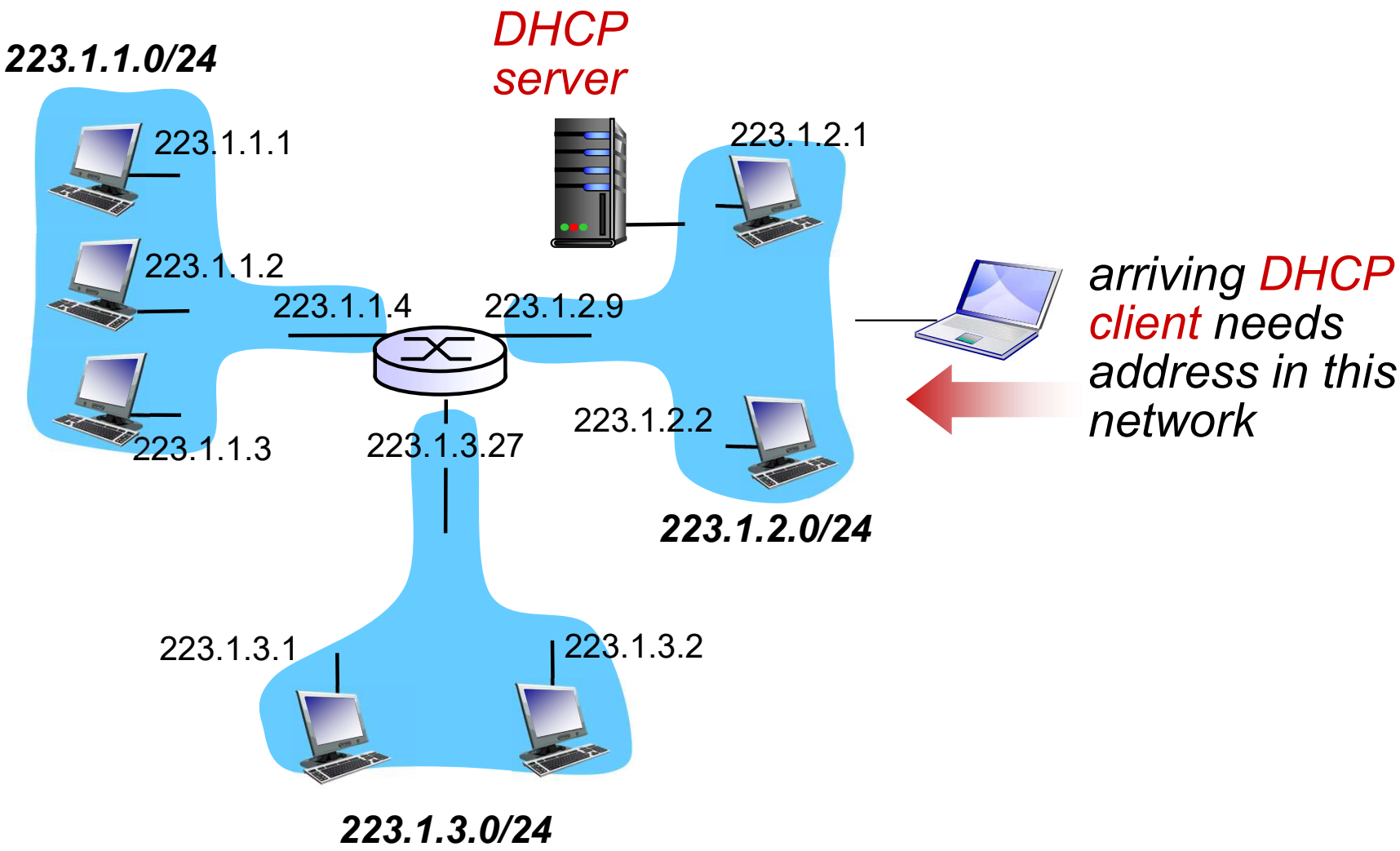
broadcast
- host broadcasts "DHCP discover" msg [optional]

broadcast
- DHCP server responds with "DHCP offer" msg [optional]

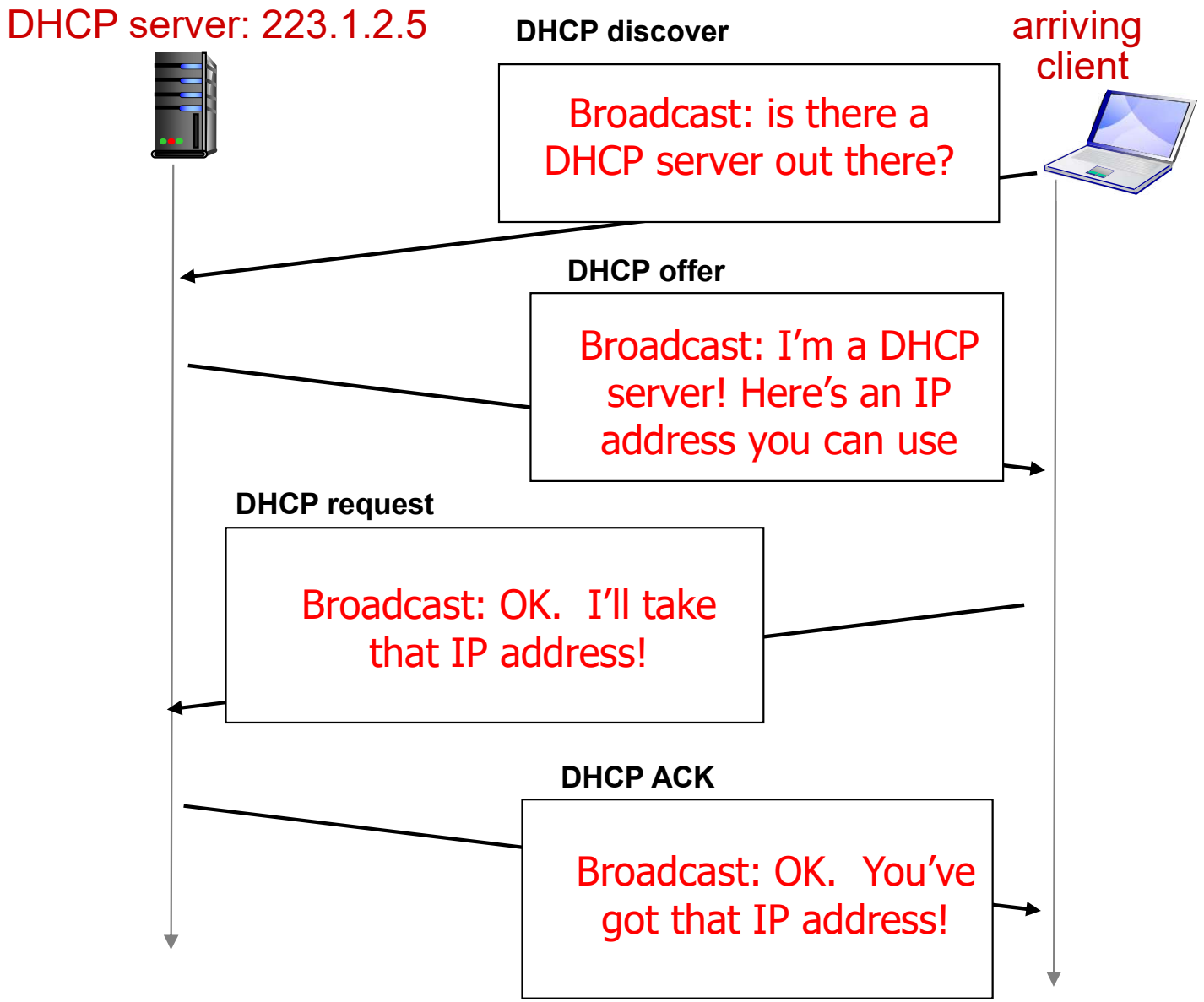
broadcast
- host requests IP address: "DHCP request" msg

broadcast
- DHCP server sends address: "DHCP Ack" msg

DHCP client-server scenario



DHCP client-server scenario

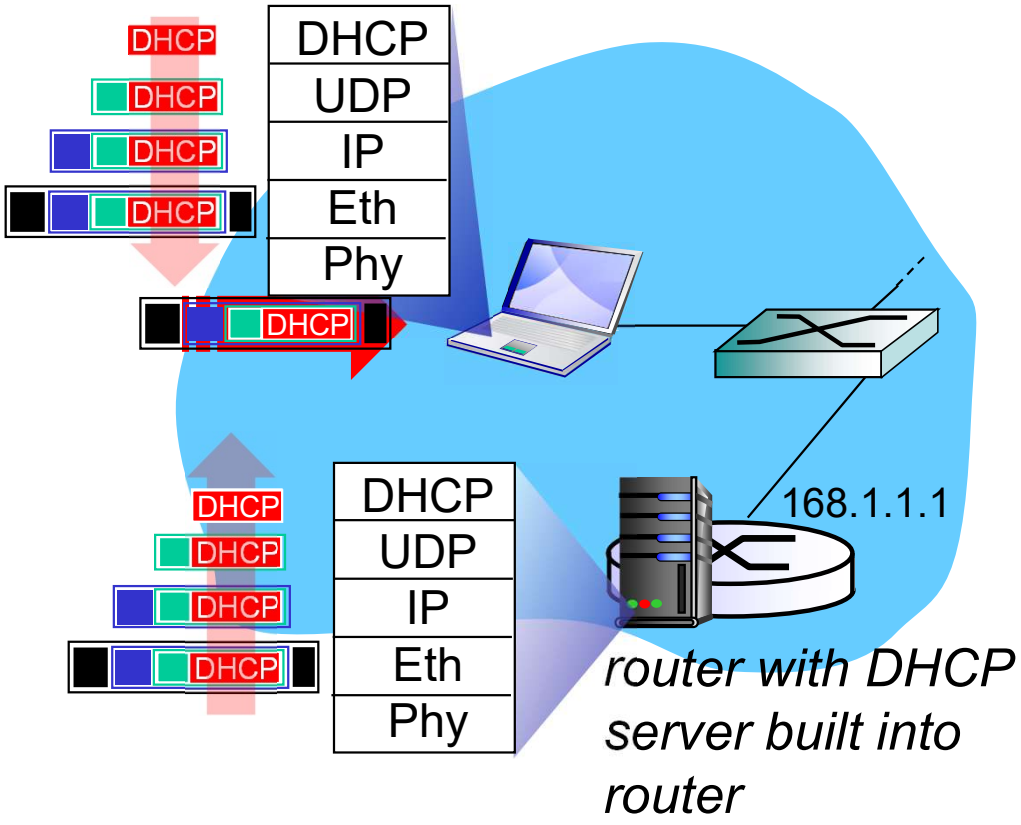


DHCP: more than IP addresses

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

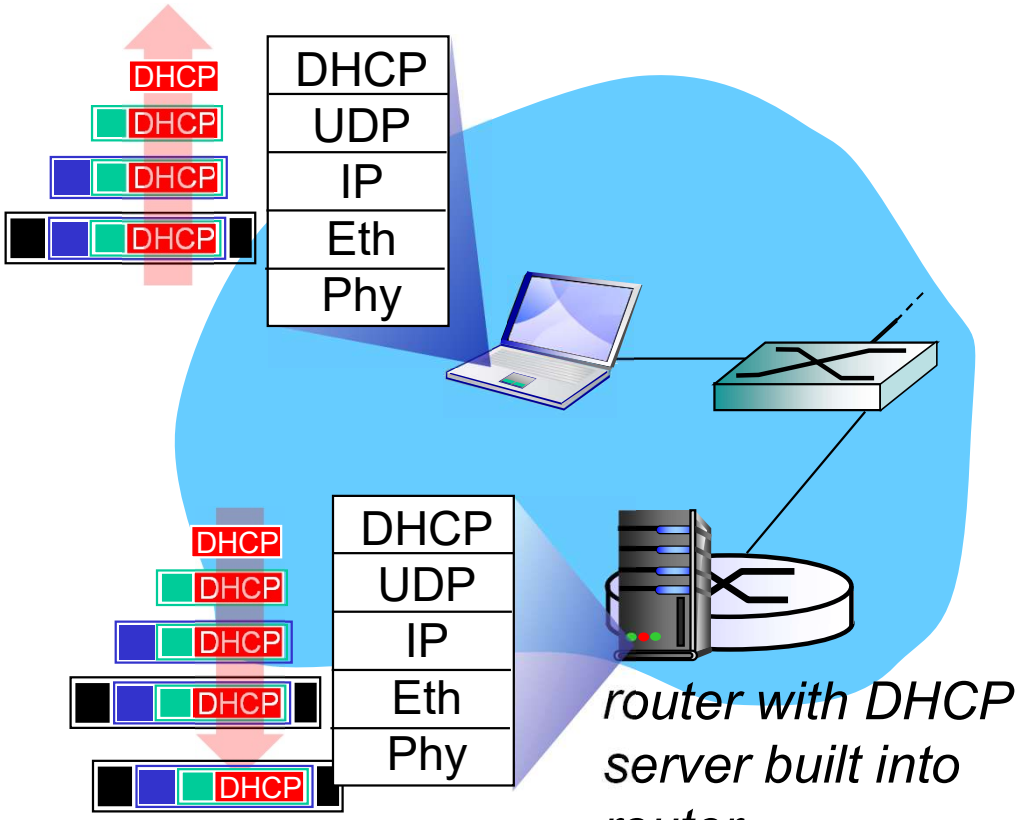
- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

DHCP: example



- connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed 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
- 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 output (home LAN)

request

Message type: **Boot Request (1)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) **DHCP Message Type = DHCP Request**
Option: (61) Client identifier
    Length: 7; Value: 010016D323688A;
    Hardware type: Ethernet
    Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Option: (t=50,l=4) Requested IP Address = 192.168.1.101
Option: (t=12,l=5) Host Name = "nomad"
Option: (55) Parameter Request List
    Length: 11; Value: 010F03062C2E2F1F21F92B
    1 = Subnet Mask; 15 = Domain Name
    3 = Router; 6 = Domain Name Server
    44 = NetBIOS over TCP/IP Name Server
    ……

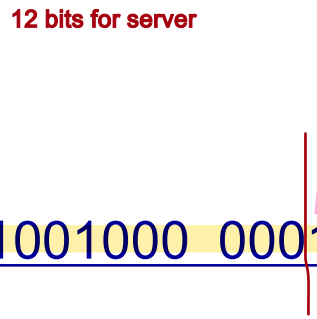
reply

Message type: **Boot Reply (2)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 192.168.1.101 (192.168.1.101)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 192.168.1.1 (192.168.1.1)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) DHCP Message Type = DHCP ACK
Option: (t=54,l=4) Server Identifier = 192.168.1.1
Option: (t=1,l=4) Subnet Mask = 255.255.255.0
Option: (t=3,l=4) Router = 192.168.1.1
Option: (6) Domain Name Server
    Length: 12; Value: 445747E2445749F244574092;
    IP Address: 68.87.71.226;
    IP Address: 68.87.73.242;
    IP Address: 68.87.64.146
Option: (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."

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 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

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

public ip pros: security, if change isp no need to change internal ip addresses, if network admin decides to change ip addresses on machine we dont need to tell outside world

how does router know which client to direct packets to when it is addressed to public ip only?

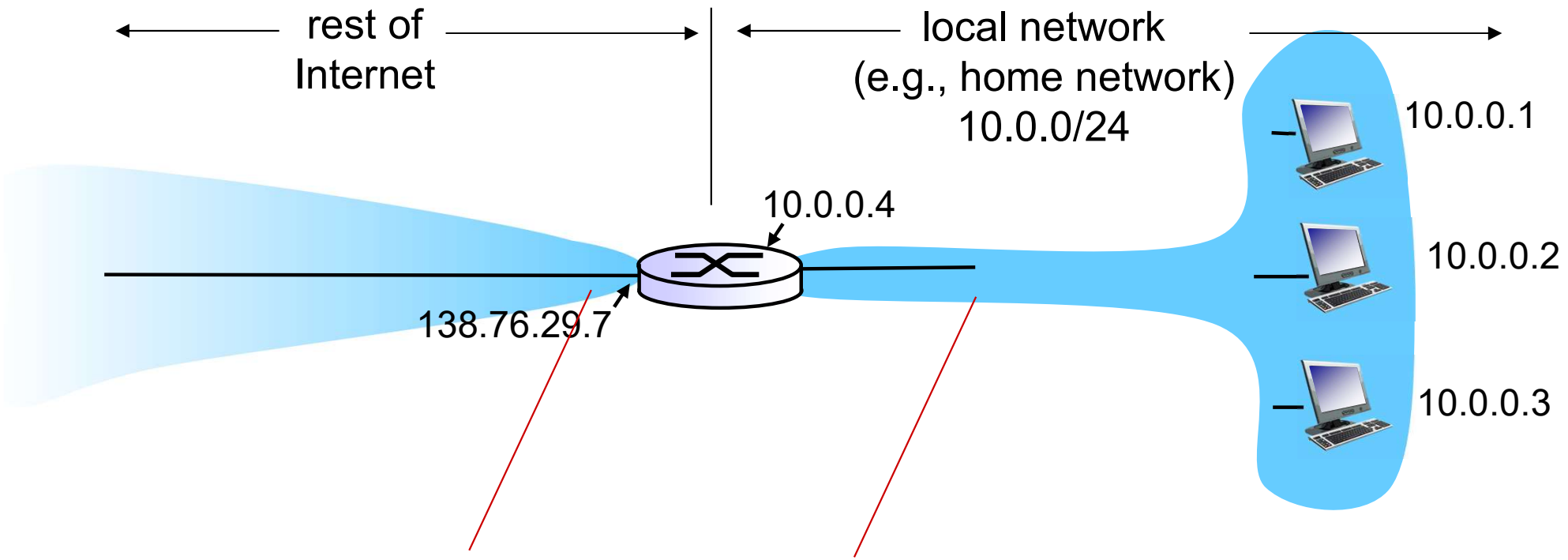
the server will assign different port for public ip/port combo so nat entries are unique

outgoing packet from client
src ip 10.102.140.12 port 12000
dest ip 130.10.10.10 port 80

outgoing packet from server
src ip 19.150.209.241 port 5000
dest ip 130.10.10.10 port 80

nat entry created: ip 19.150.209.241 port 5000 -> ip 10.102.140.120 port 12000

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

implementation: NAT router must:

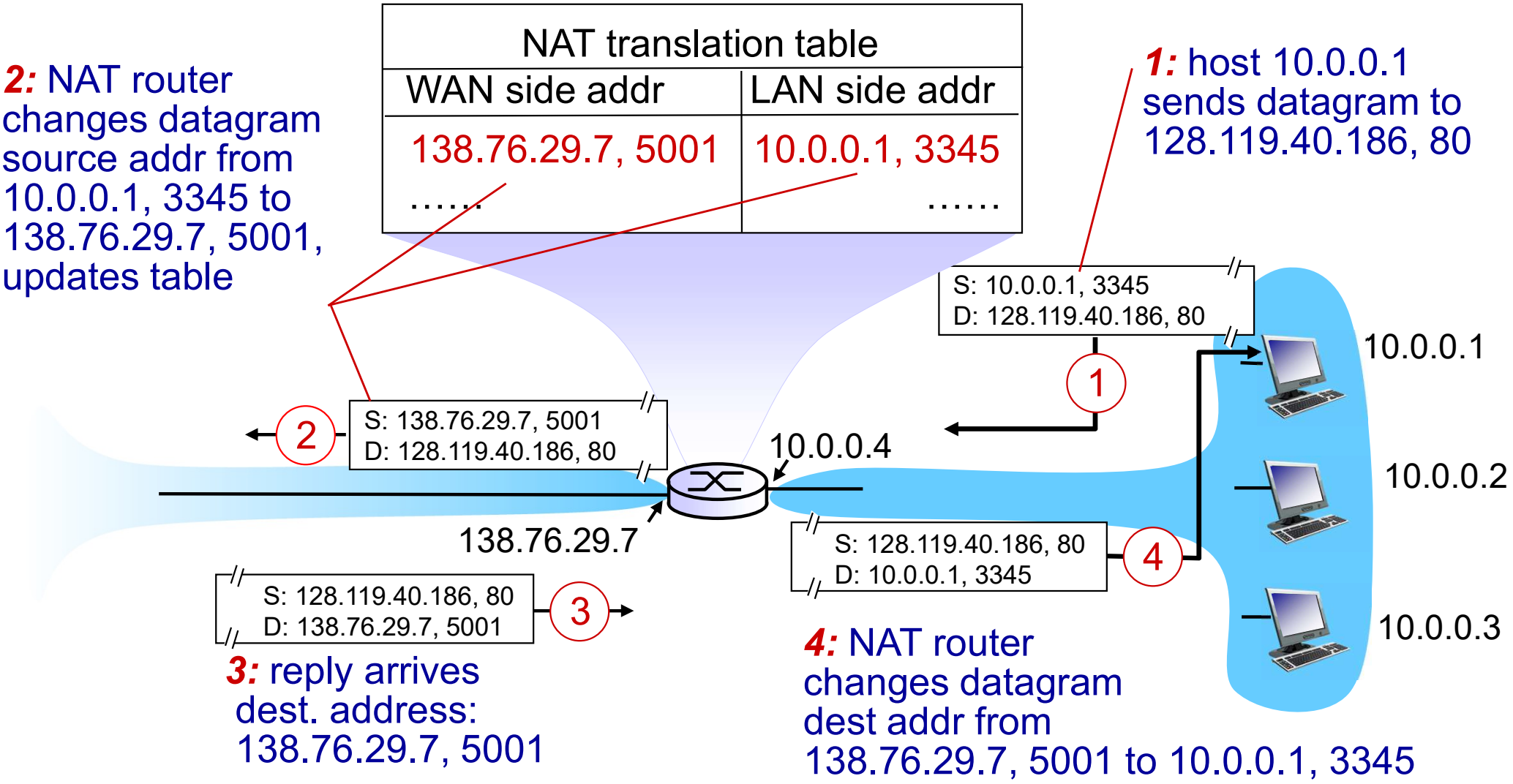
- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)

  . . . 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



* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

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?

Set 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

IPv6: motivation

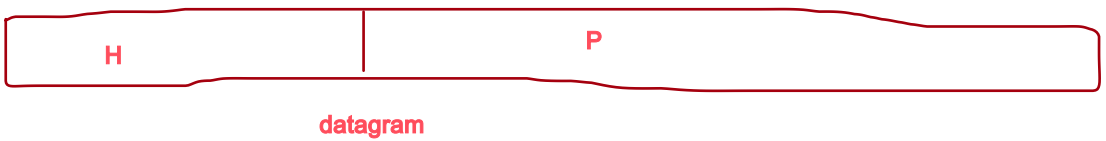
- *initial motivation:* 32-bit address space soon to be completely allocated.
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

IPv6 datagram format:

- fixed-length 40 byte header
- no fragmentation allowed

128 bits, 2^128-2 connections possible
8 sections of 4 hex bits
double colon can be used to abbreviate consecutive 0 segments

20 bytes header for ipv4
40 byte header for ipv6

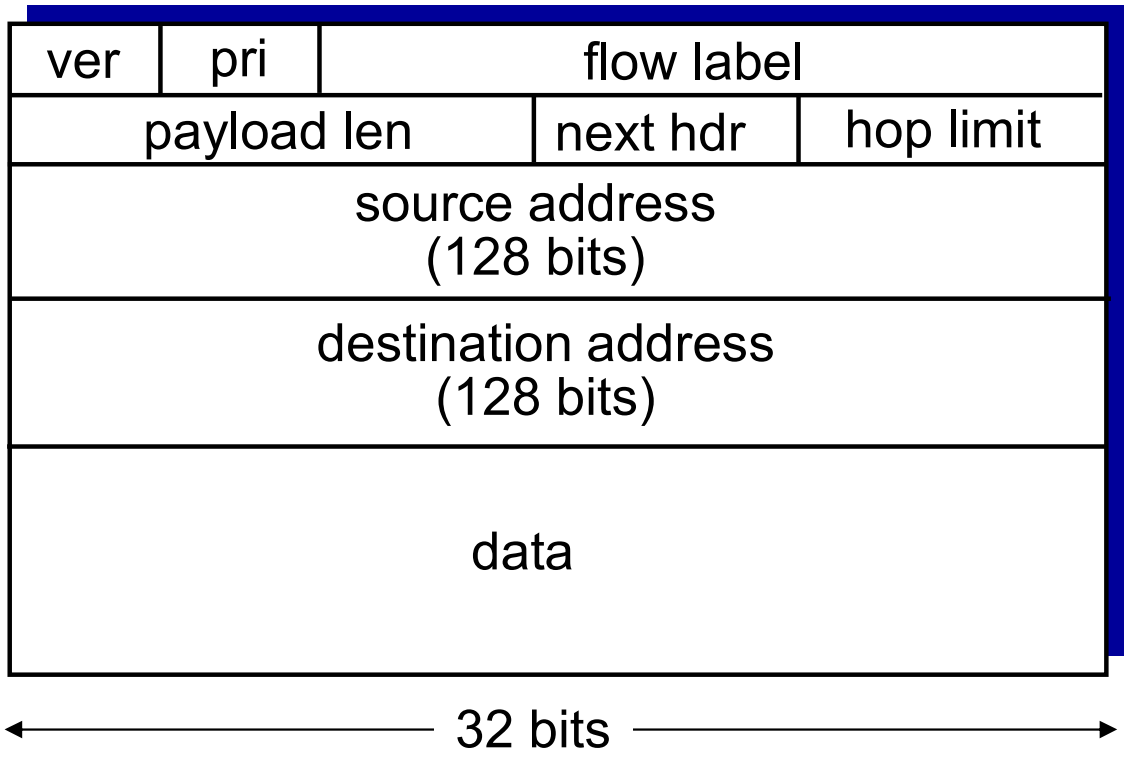


IPv6 datagram format

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



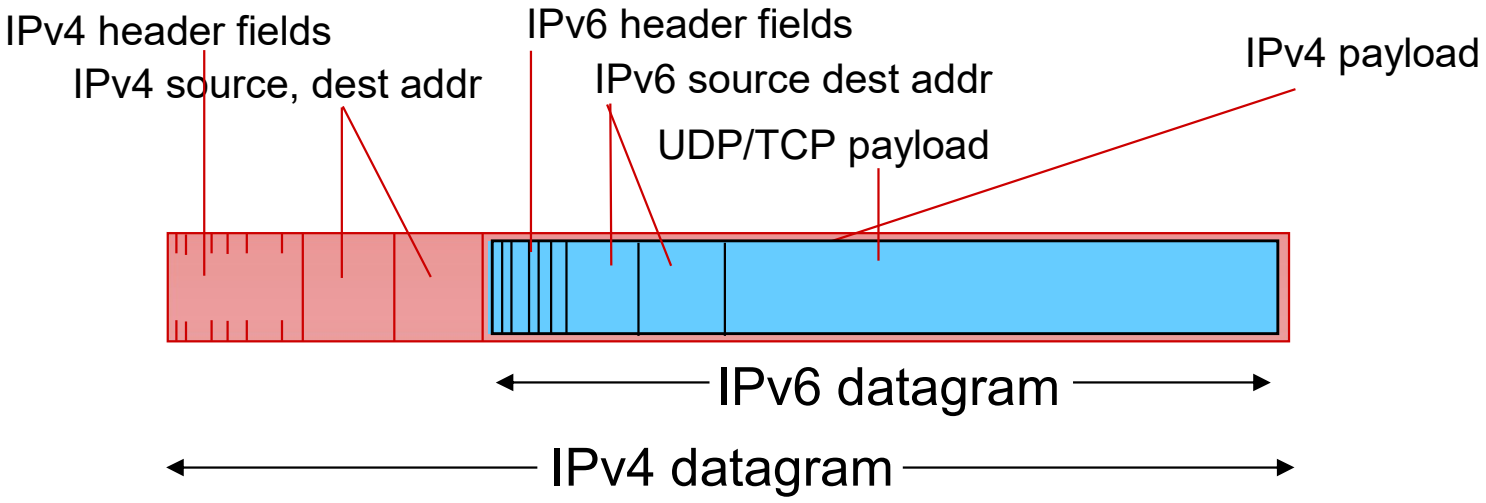
no flags, no checksum, no fragmentation. no router will fragment only the source

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

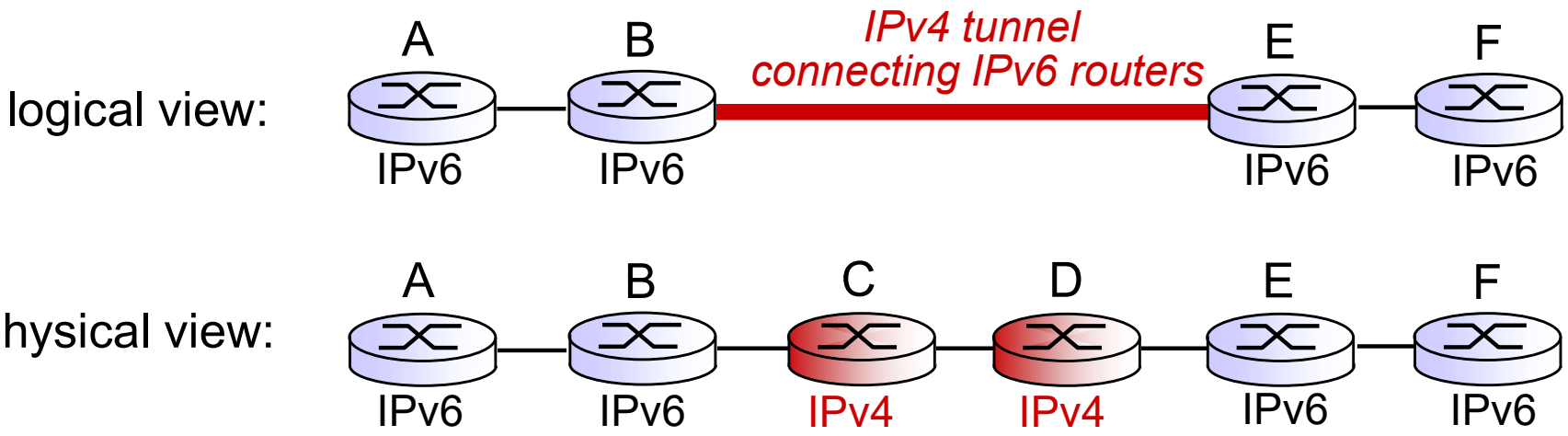
Transition from IPv4 to IPv6

- 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

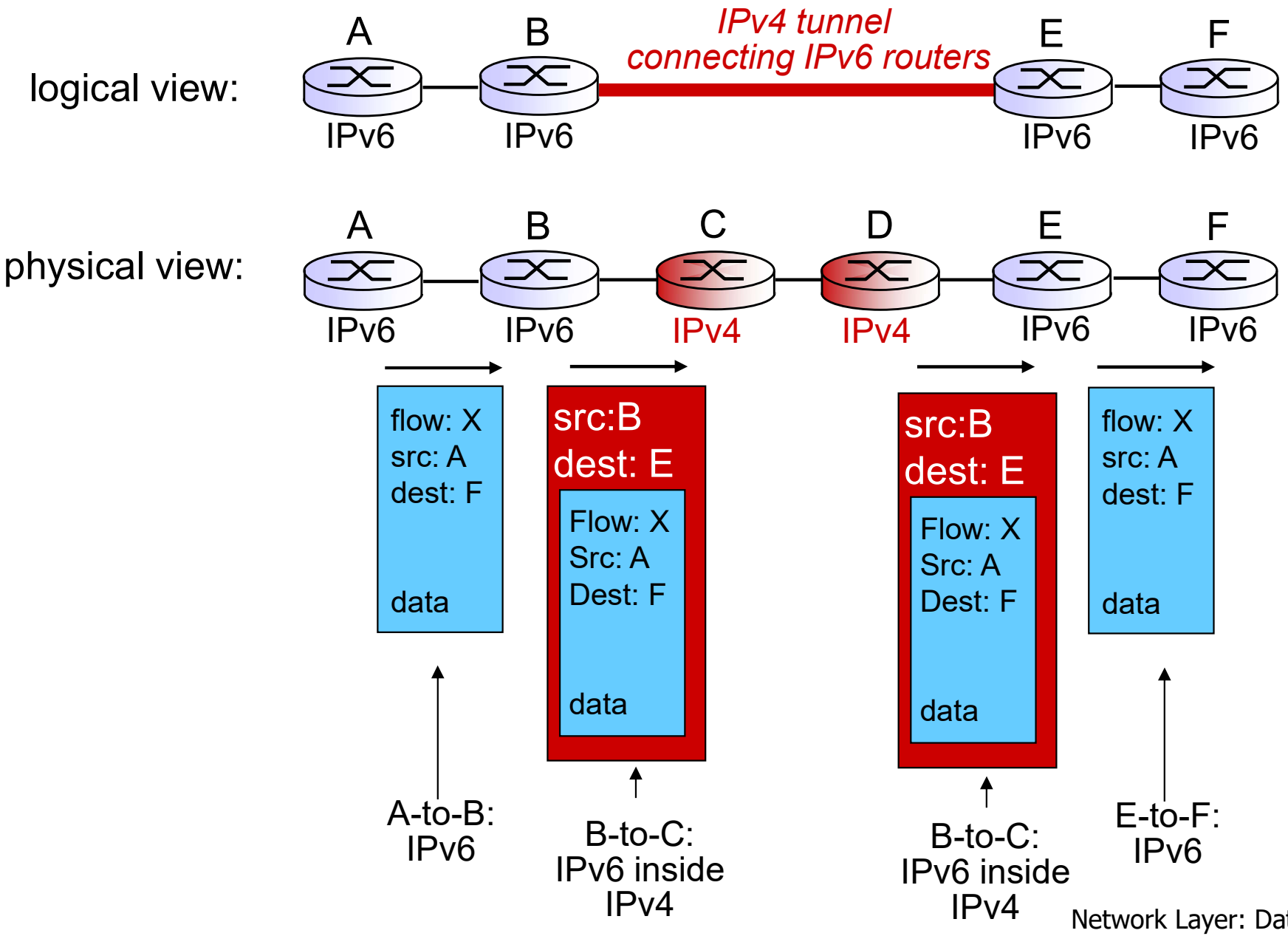


wrap ipv6 packets in ipv4 for older

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

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 Addresses

- The IPv6 addresses are represented by a sequence of 8 4-hexadecimal numbers divided by colons, for example
  - 2001:0DB8:0055:0000:CD23:0000:0000:0205
  - 2001:0DB8:55:0:CD23:0:0:0205
  - 2001:0DB8:55::CD23:0:0:0205 **OR** 2001:0DB8:55:0:CD23::0205 (consecutive groups of all zeros can be substituted by a double colon, but this may be done once in the address)

IPv6: adoption

- Google: ~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, Facebook, streaming media, Skype, …
  - *Why?*
    - it is very difficult to change network-layer protocols
    - introducing new protocols into the network layer is like replacing the foundation of a house

Set 4: *done!*

4.1 Overview of Network layer: data plane and control plane

4.2 What's inside a router

4.3 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- NAT
- IPv6

Question: how do forwarding tables computed?

Answer: by the control plane (next set)