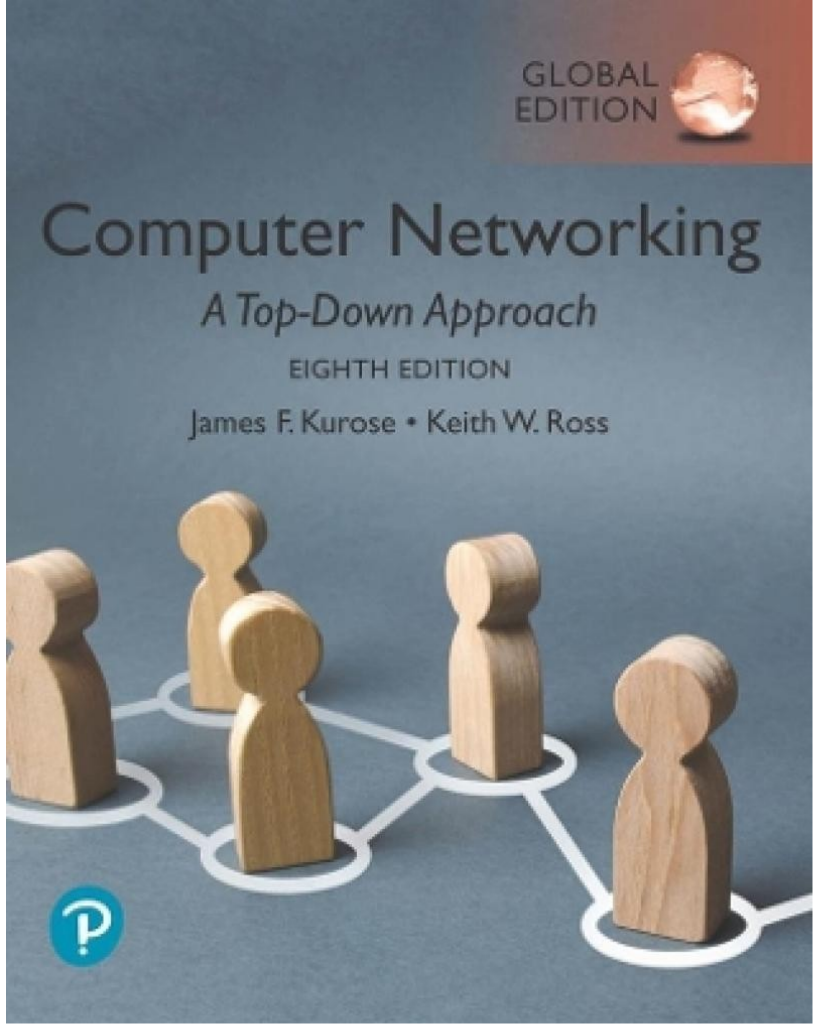
Chapter 4 Network Layer: Data Plane

A note on the use of these PowerPoint slides:



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

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
  - generalized forwarding
- IP protocol
- Network address translation (NAT)

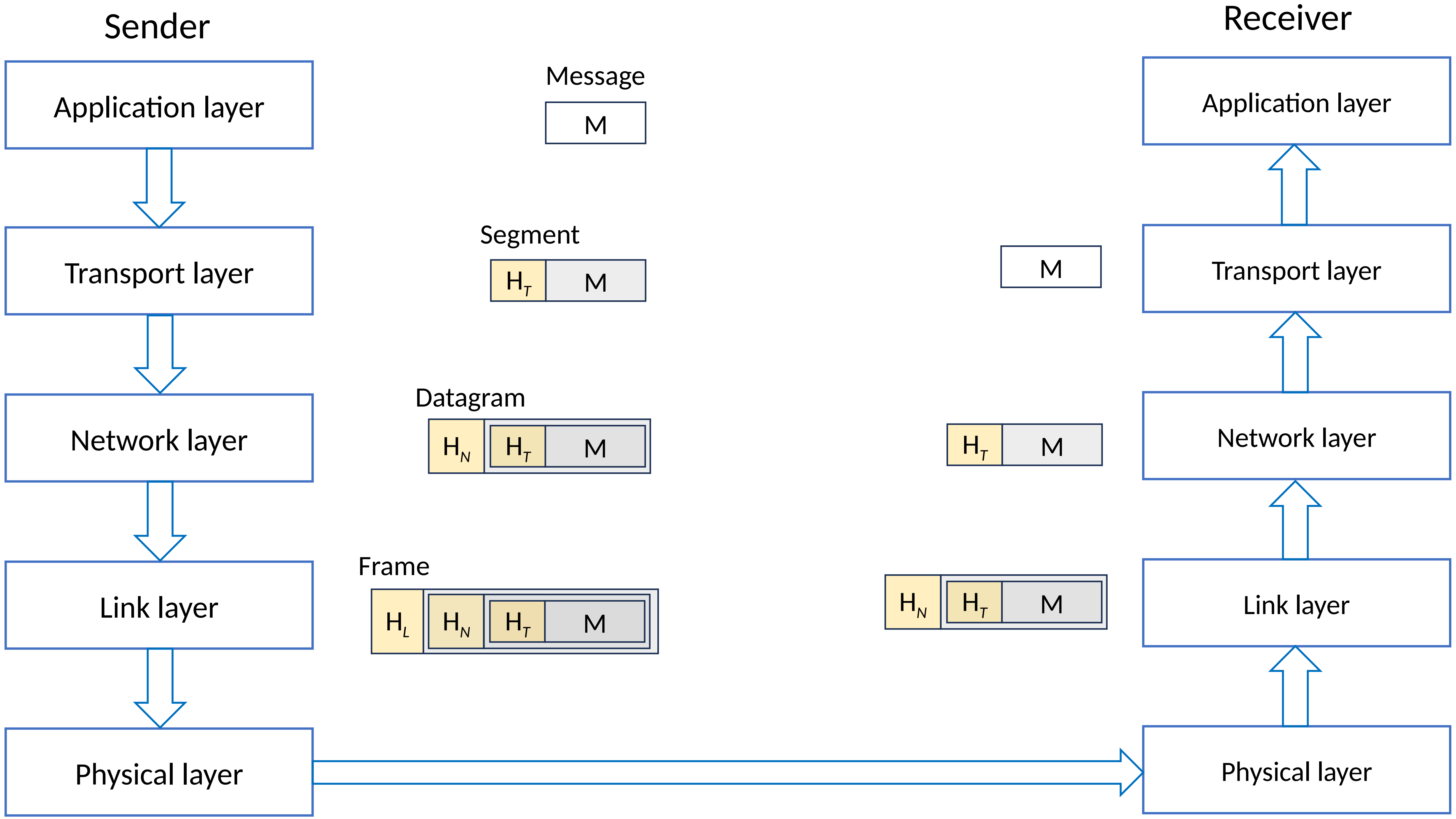
Network layer: "data plane" roadmap

4.1 Network layer: overview
- forwarding plane (data plane)
- routing (control plane)

4.2 What's inside a router
- input ports, forwarding,
- switching, output ports, scheduling

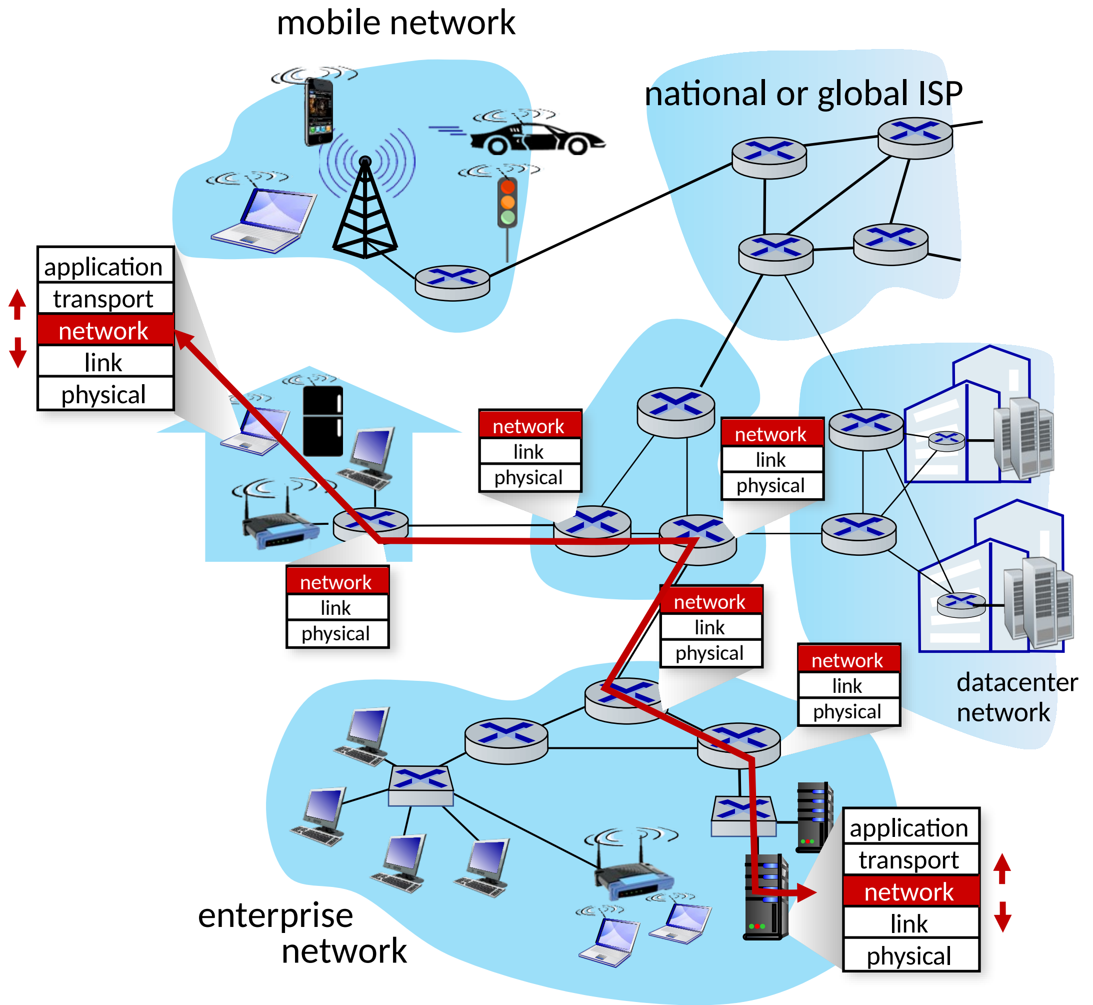
4.3 IP: the Internet Protocol
- datagram format
- addressing
- network address translation
- IPv6

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


Sender
Receiver
Application layer
Transport layer
Network layer
Link layer
Physical layer
Message
M
Segment
H_T M
Datagram
H_N H_T M
Frame
H_L H_N H_T M
Application layer
Transport layer
Network layer
Link layer
Physical layer
M
H_T M
H_N H_T M

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:

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

analogy: taking a trip

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

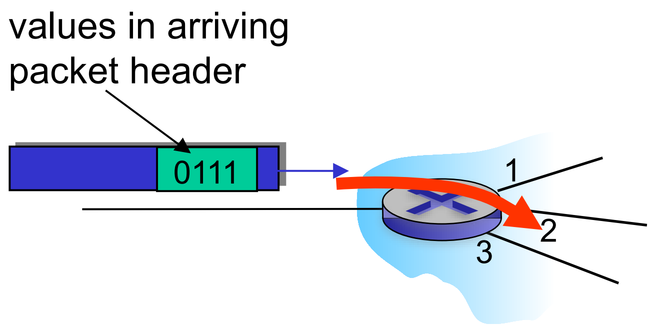
routing

- *forwarding:* process of getting through single interchange

Network layer: forwarding plane, control plane

Forwarding ("data plane"):

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

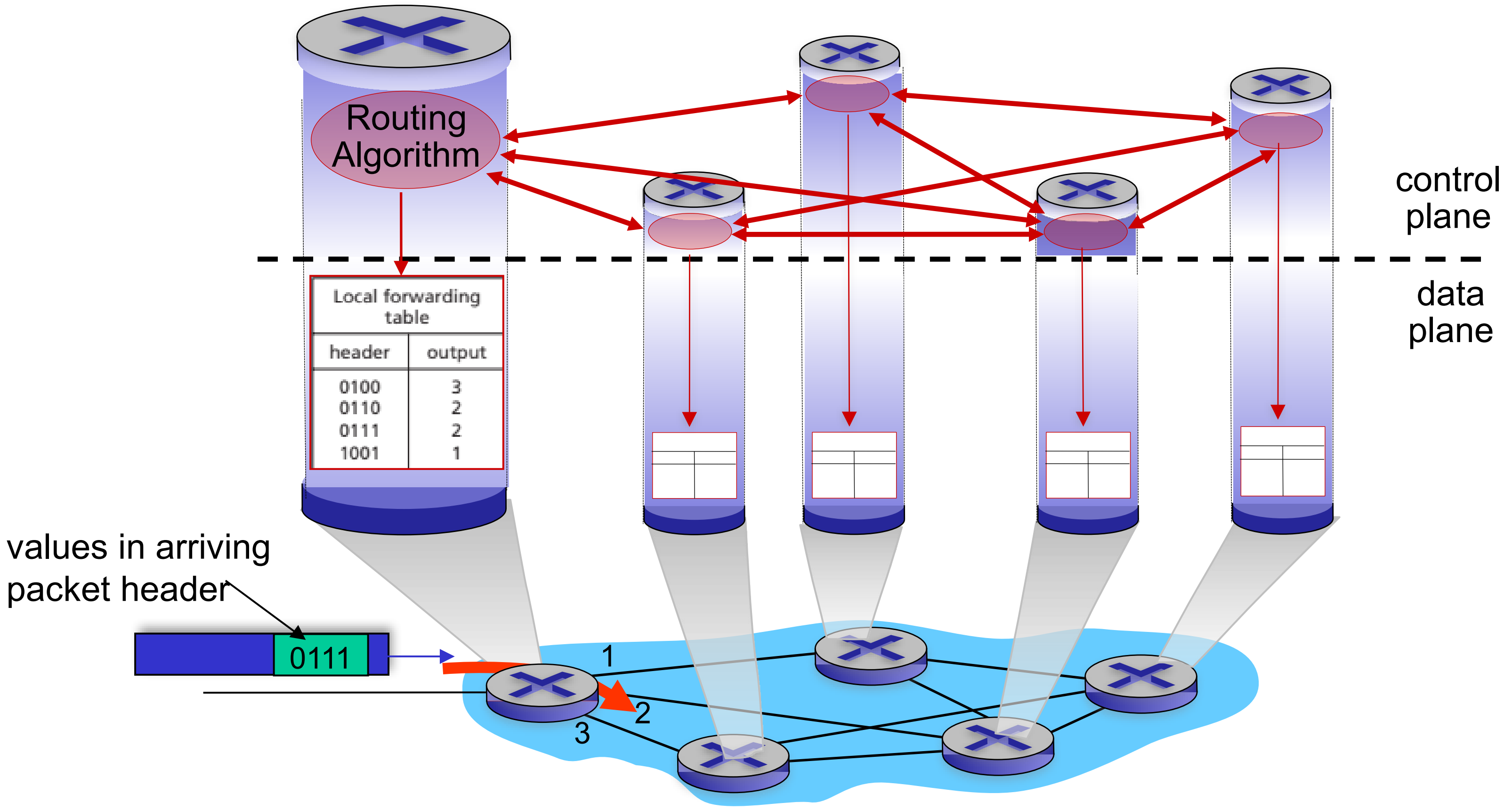


Routing ("control plane"):

- *network-wide* logic
- determines how datagram is routed among routers along end-end path from source host to destination host
- two control-plane approaches:
  - *Distributed routing*: implemented in routers
  - *Centralized routing (aka. software-defined networking)*: implemented in remote servers

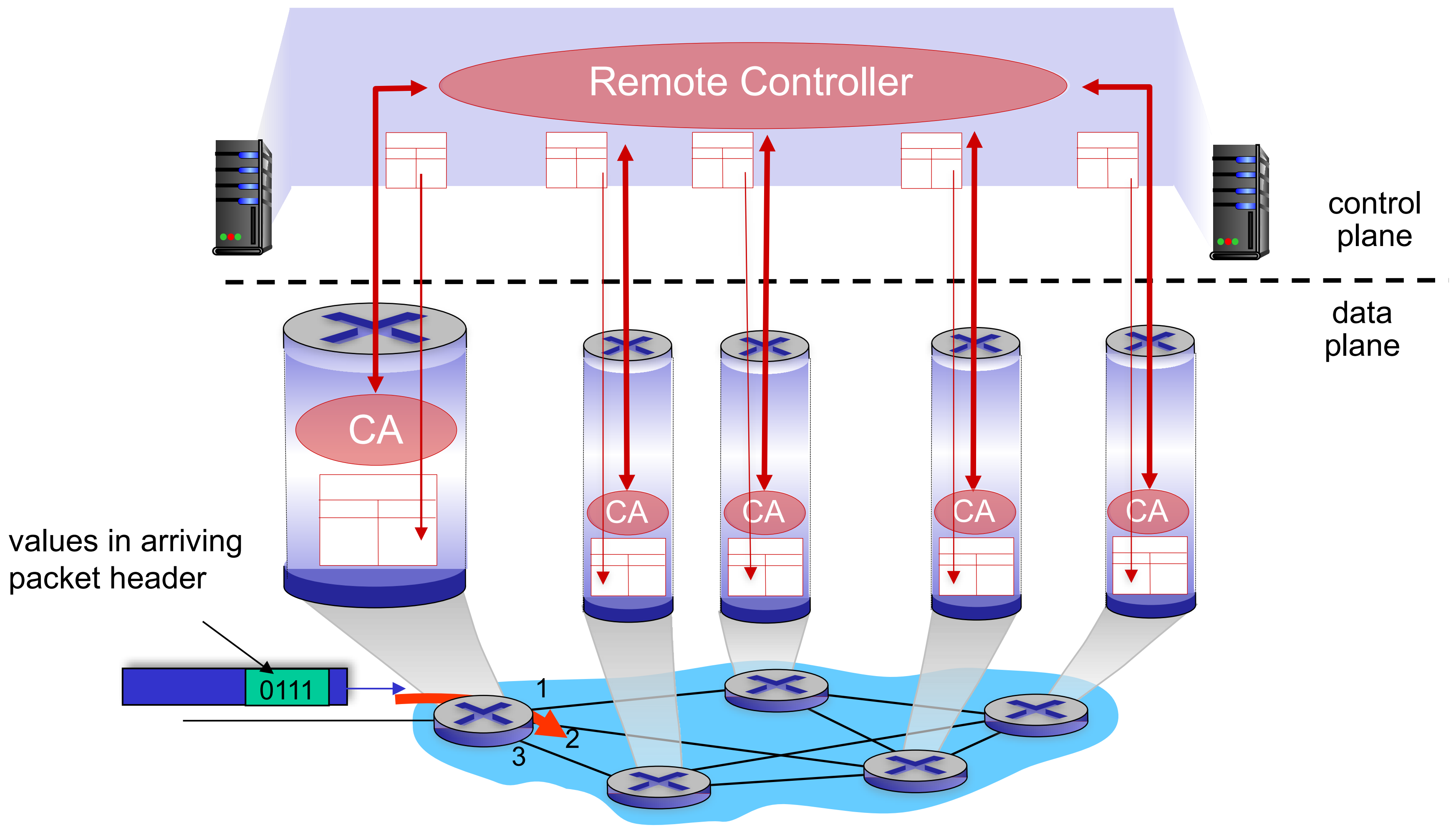
Distributed routing

Individual routing algorithms *in each and every router*



Centralized routing: Software-Defined Networking (SDN)

Remote controller computes forwarding tables in routers



IP 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 (r.d.t.)
ii. order of delivery (r.d.t.)
iii. bandwidth available to end-end flow

Network layer: "forwarding plane" roadmap

4.1 Network layer: overview
- forwarding plane
- control plane

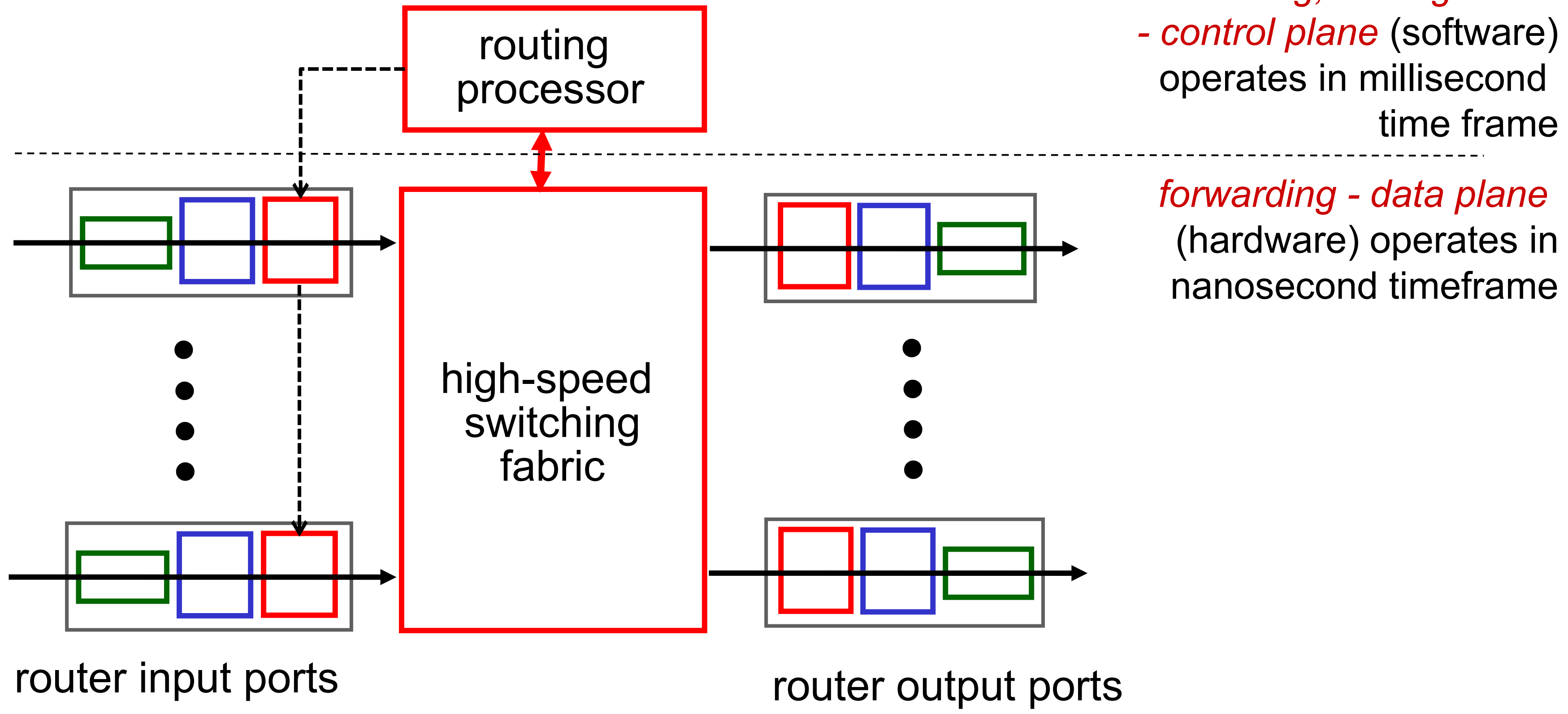
4.2 What's inside a router
- input ports, forwarding,
- switching, output ports, scheduling

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

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

Router architecture overview

high-level view of generic router architecture:



routing processor
routing, management - control plane (software) operates in millisecond time frame
forwarding - data plane (hardware) operates in nanosecond timeframe
high-speed switching fabric
router input ports
router output ports

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?

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!

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.

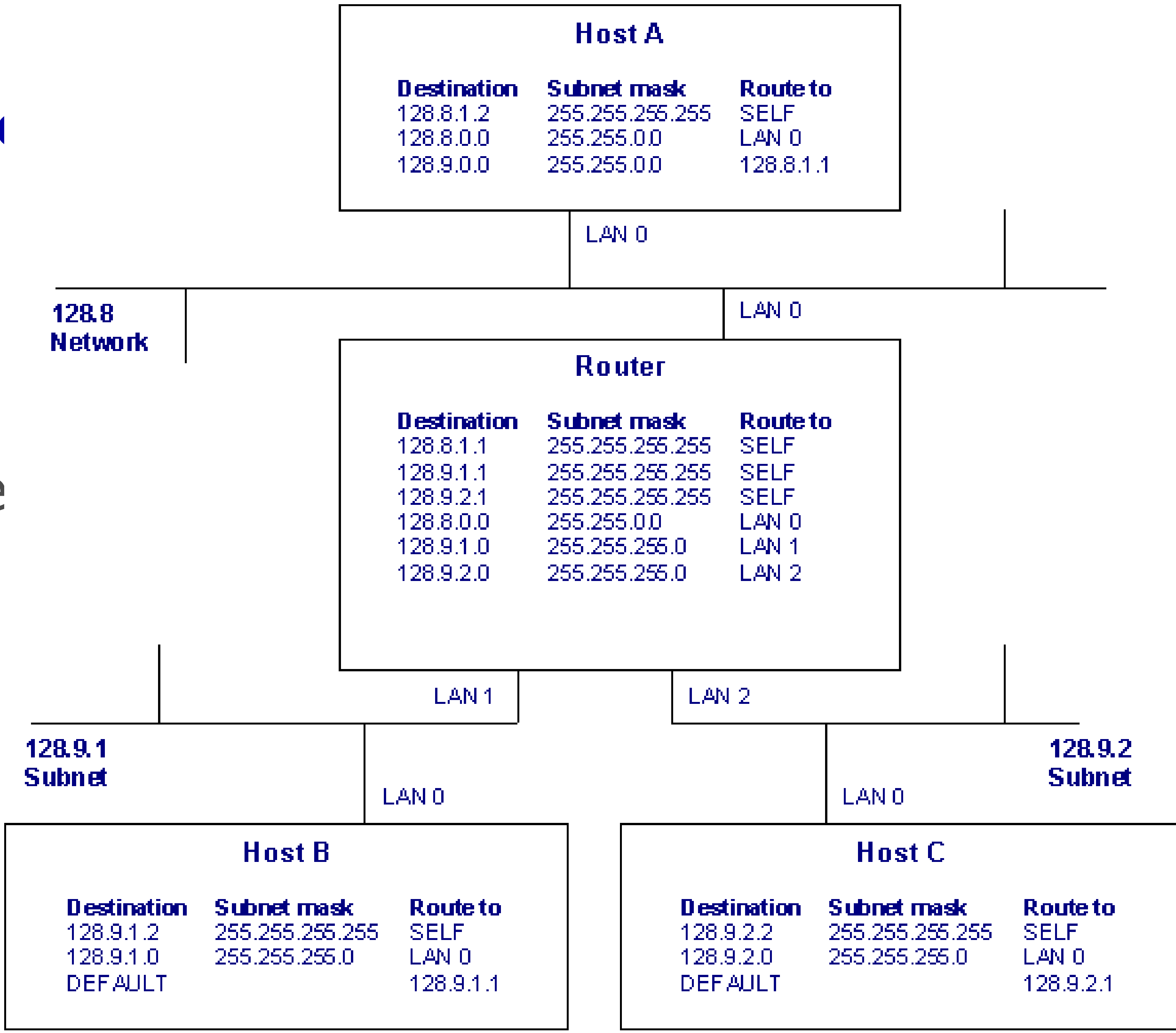
Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
11001000 00010111 00010000 000001**	**3**
otherwise	3

new entry

Destination-based forwarding exampl

- Host A sends a packet to 128.9.2.2 (Host C)

- The router has three addresses
  - one IP address per interface



Host A

Destination	Subnet mask	Route to
128.8.1.2	255.255.255.255	SELF
128.8.0.0	255.255.0.0	LAN 0
128.9.0.0	255.255.0.0	128.8.1.1

Router

Destination	Subnet mask	Route to
128.8.1.1	255.255.255.255	SELF
128.9.1.1	255.255.255.255	SELF
128.9.2.1	255.255.255.255	SELF
128.8.0.0	255.255.0.0	LAN 0
128.9.1.0	255.255.255.0	LAN 1
128.9.2.0	255.255.255.0	LAN 2

Host B

Destination	Subnet mask	Route to
128.9.1.2	255.255.255.255	SELF
128.9.1.0	255.255.255.0	LAN 0
DEFAULT		128.9.1.1

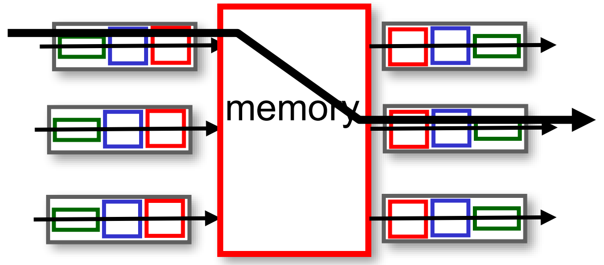
Host C

Destination	Subnet mask	Route to
128.9.2.2	255.255.255.255	SELF
128.9.2.0	255.255.255.0	LAN 0
DEFAULT		128.9.2.1

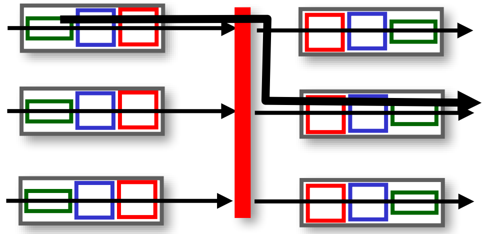
https://www.eventhelix.com/networking/ip-routing/

Switching fabrics

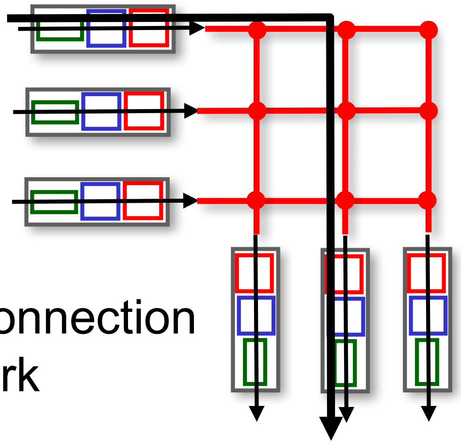
- transfer packet from input link to appropriate output link
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three major types of switching fabrics:



memory

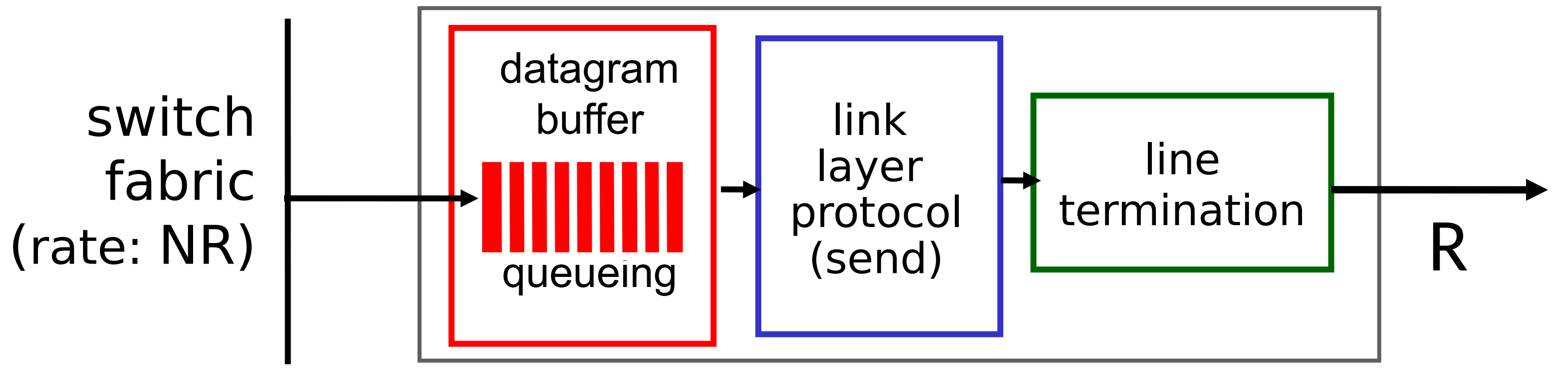


bus



interconnection network

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

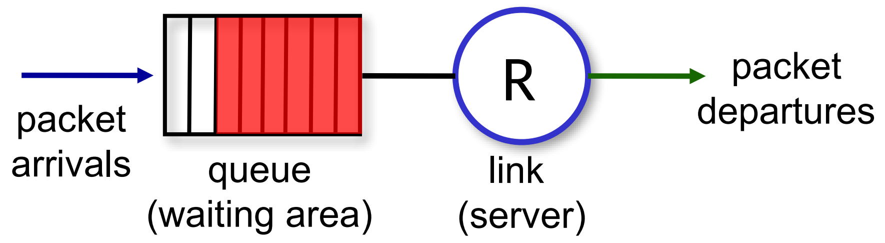
→ Priority scheduling – who gets best performance, network neutrality

Packet scheduling

packet scheduling: deciding which packet to send next on link:

1. First come, first served (FCFS)
2. Priority
3. Round robin
4. Weighted fair queueing

Abstraction: queue

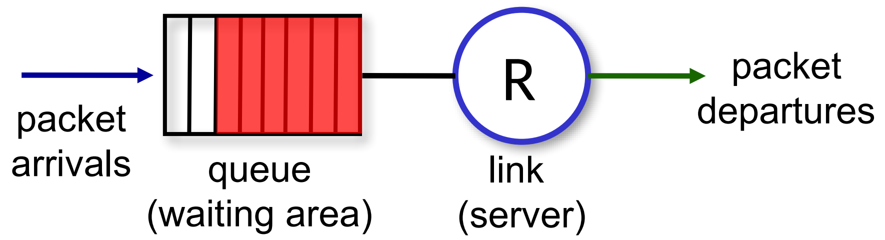


Packet Scheduling: First-come-first-served (FCFS)

FCFS:

- packets transmitted in the same order they arrive the queue
  - also known as: First-in-first-out (FIFO)
  - real world examples?

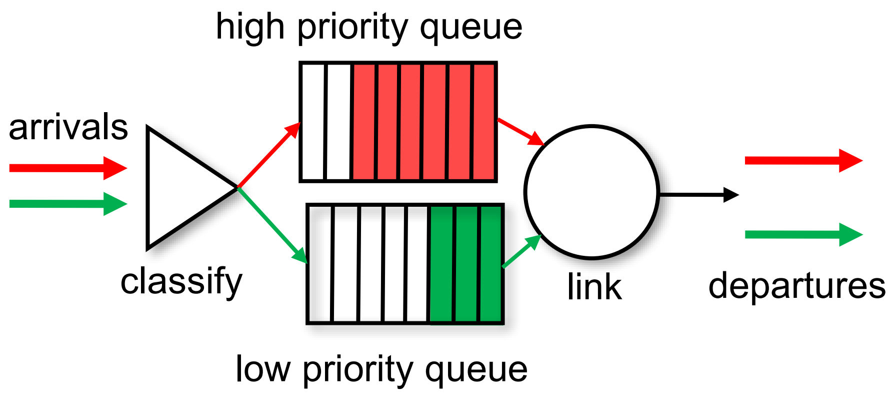
Abstraction: queue

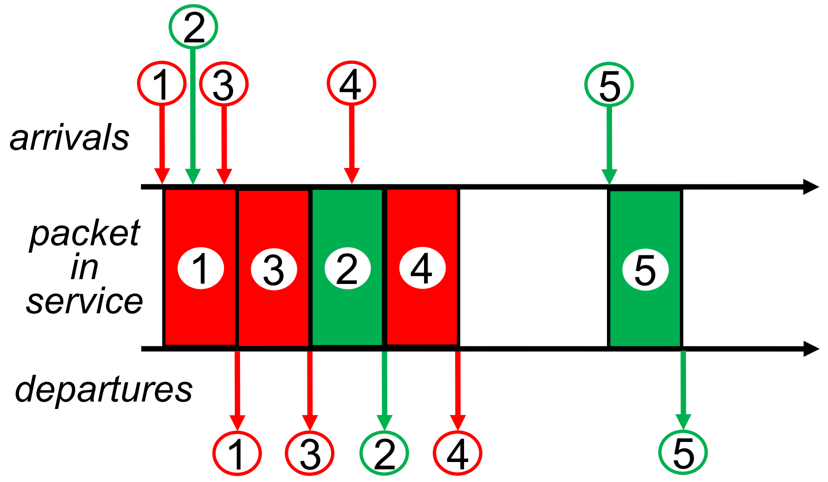


Scheduling policies: Priority scheduling

Priority scheduling:

- arriving traffic classified, queued by class
  - any header fields can be used for classification
- send packet from highest priority queue that has buffered packets
  - FCFS within priority class

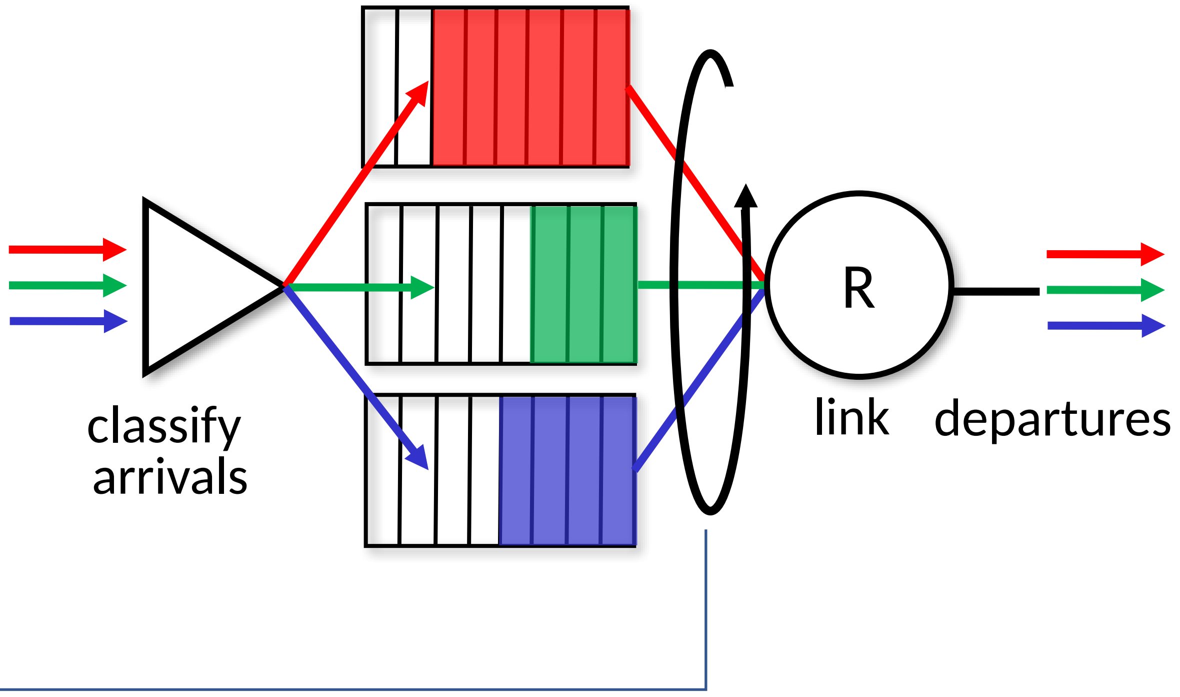




Scheduling policies: Round-robin scheduling

Round Robin (RR) scheduling:

- arriving traffic classified, queued by class
  - any header fields can be used for classification
- server cyclically, repeatedly scans class queues, sending one complete packet from each class (if available) in turn



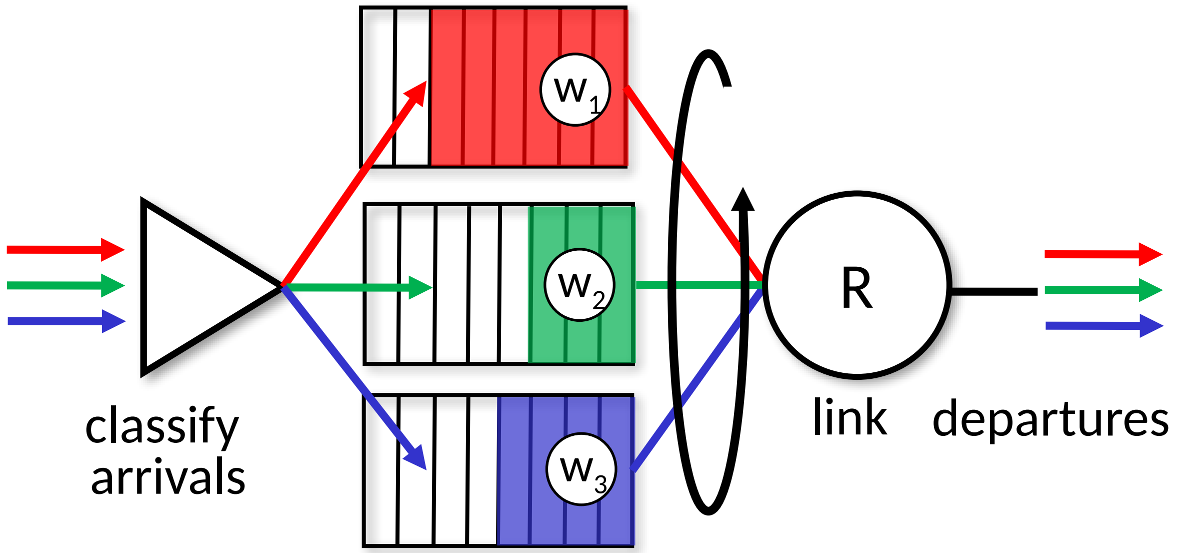
Scheduling policies: weighted fair queueing

Weighted Fair Queuing (WFQ):

- generalized Round Robin
- each class, *i*, has weight, w_i, and gets weighted amount of service in each cycle:

$$\frac{w_i}{\Sigma_j w_j}$$

- minimum bandwidth guarantee (per-traffic-class)



Network layer: “data plane” roadmap

4.1 Network layer: overview
- data plane
- control plane

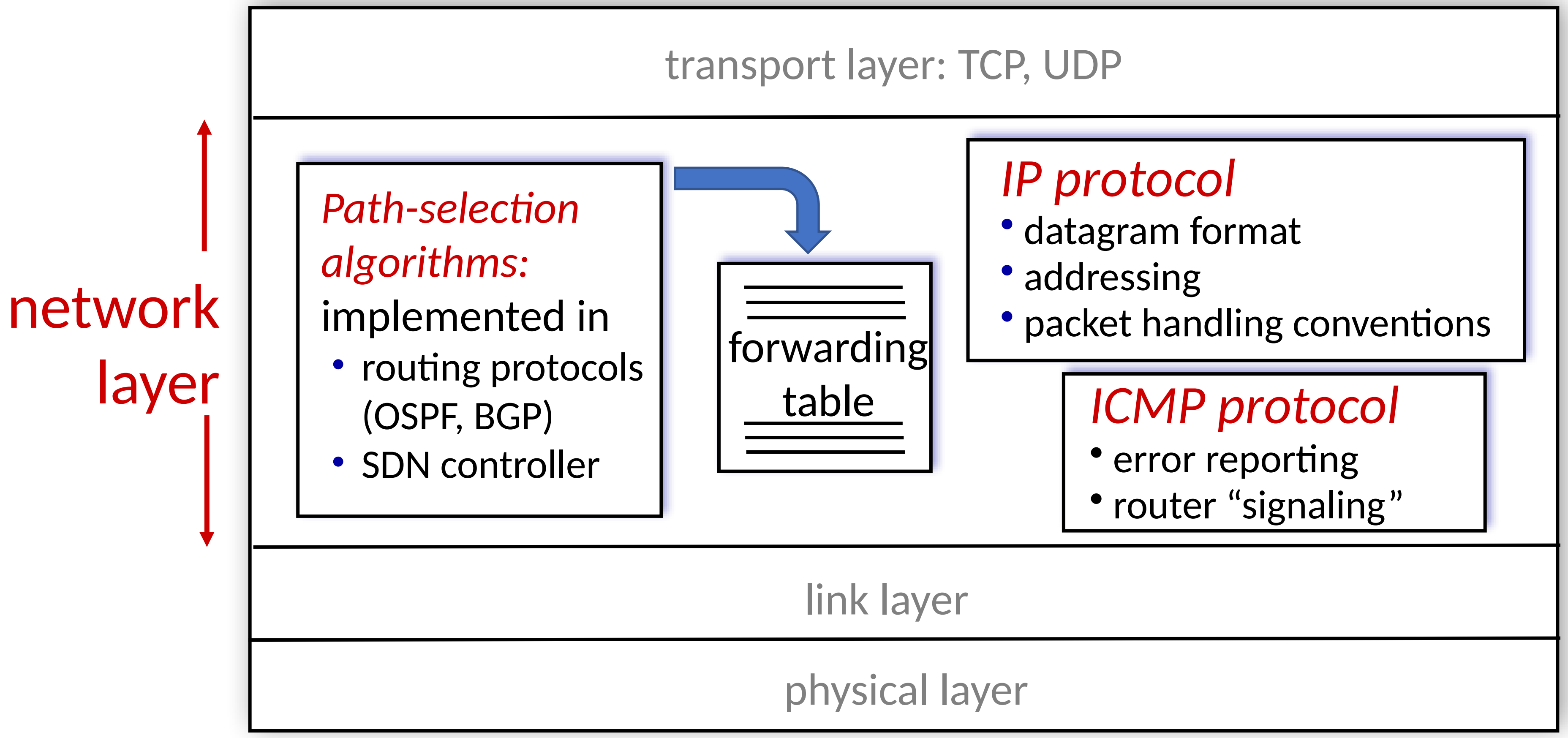
4.2 What’s inside a router
- input ports, forwarding,
- switching, output ports, scheduling

4.3 The Internet Protocol (IP)
- datagram format
- addressing
- network address translation
- IPv6

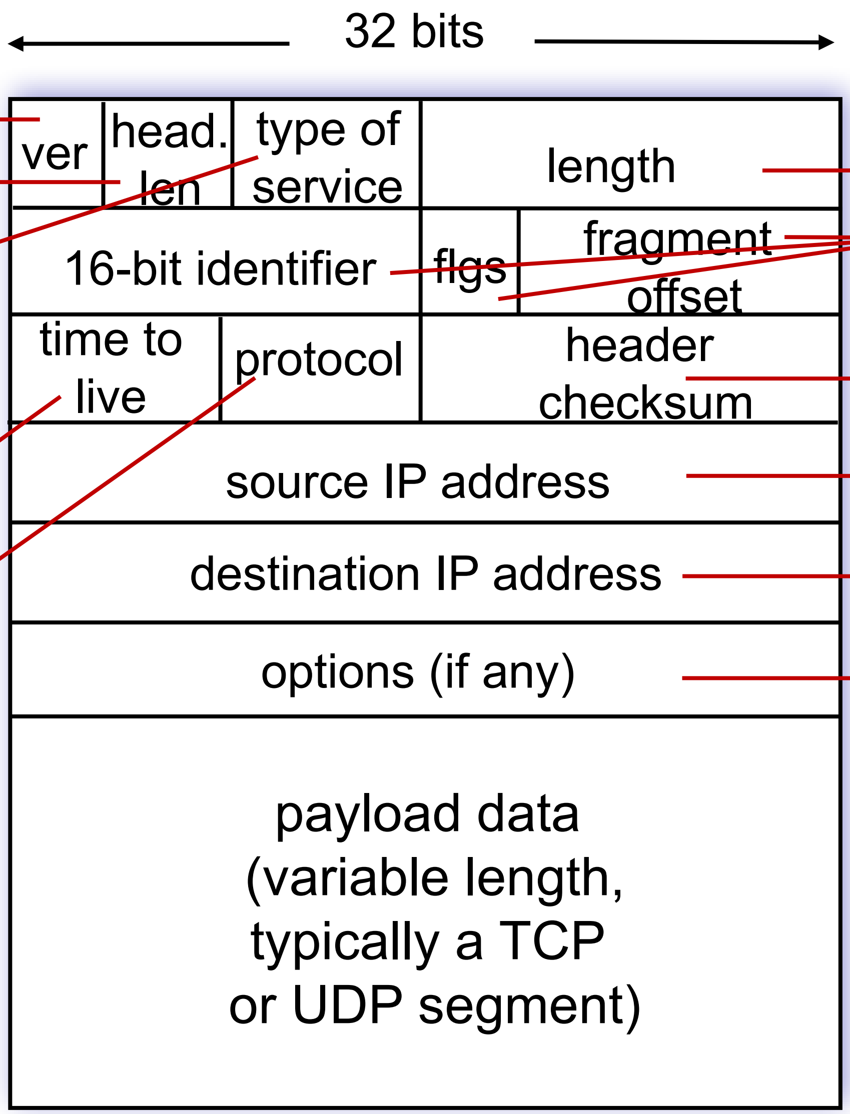
4.4 Generalized Forwarding, SDN
- match + action
- OpenFlow: match + action in action

Network Layer: Internet

host, router network layer functions:



IPv4 Datagram format



IP protocol version number

header length(bytes)

“type” of service:
- diffserv (0:5)
- ECN (6:7)

TTL: remaining max hops (decremented at each router)

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

total datagram length (bytes)

fragmentation/ reassembly

header checksum

32-bit source IP address

3

Maximum length: 64K bytes
Typically: 1500 bytes or less

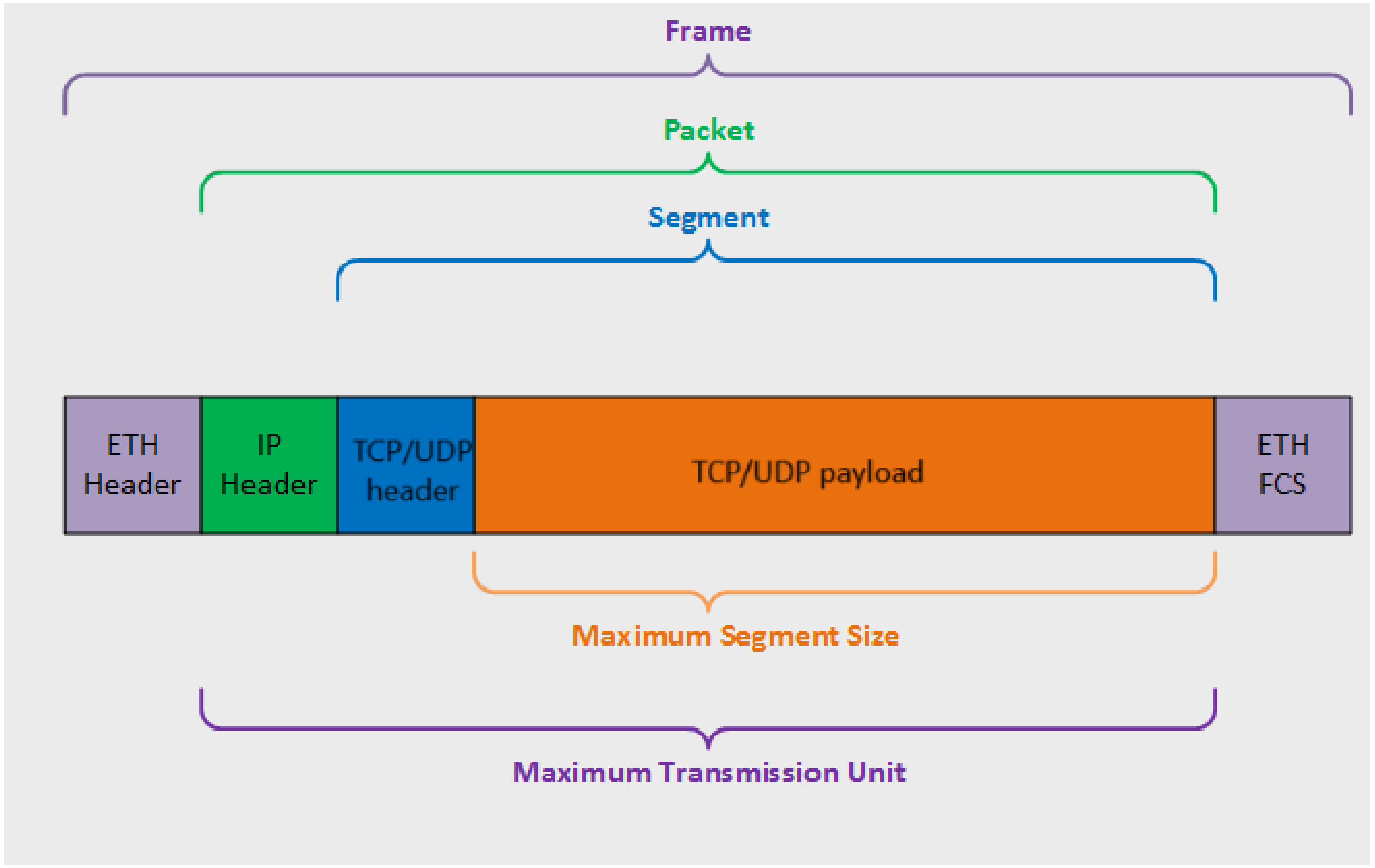
e.g., timestamp, record route taken

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

Maximum payload size: MTU vs. MSS

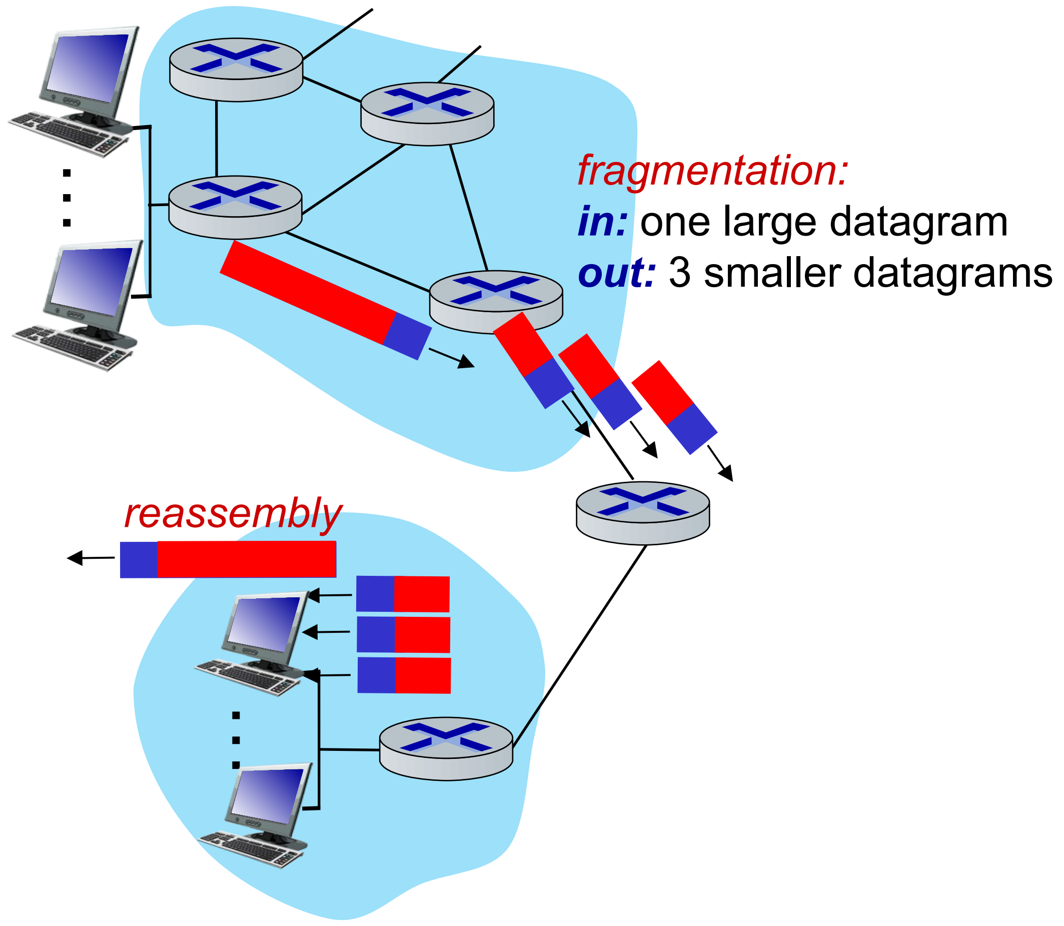
Maximum payload size:

- **Frames**:
  Maximum transfer unit (MTU)
  - e.g. Ethernet: 1500 bytes
- **Segments**:
  Maximum segment size (MSS)
  - MSS = MTU – 40
    = 1500 – 40 =1460
  - 40 = IP header size + TCP header size

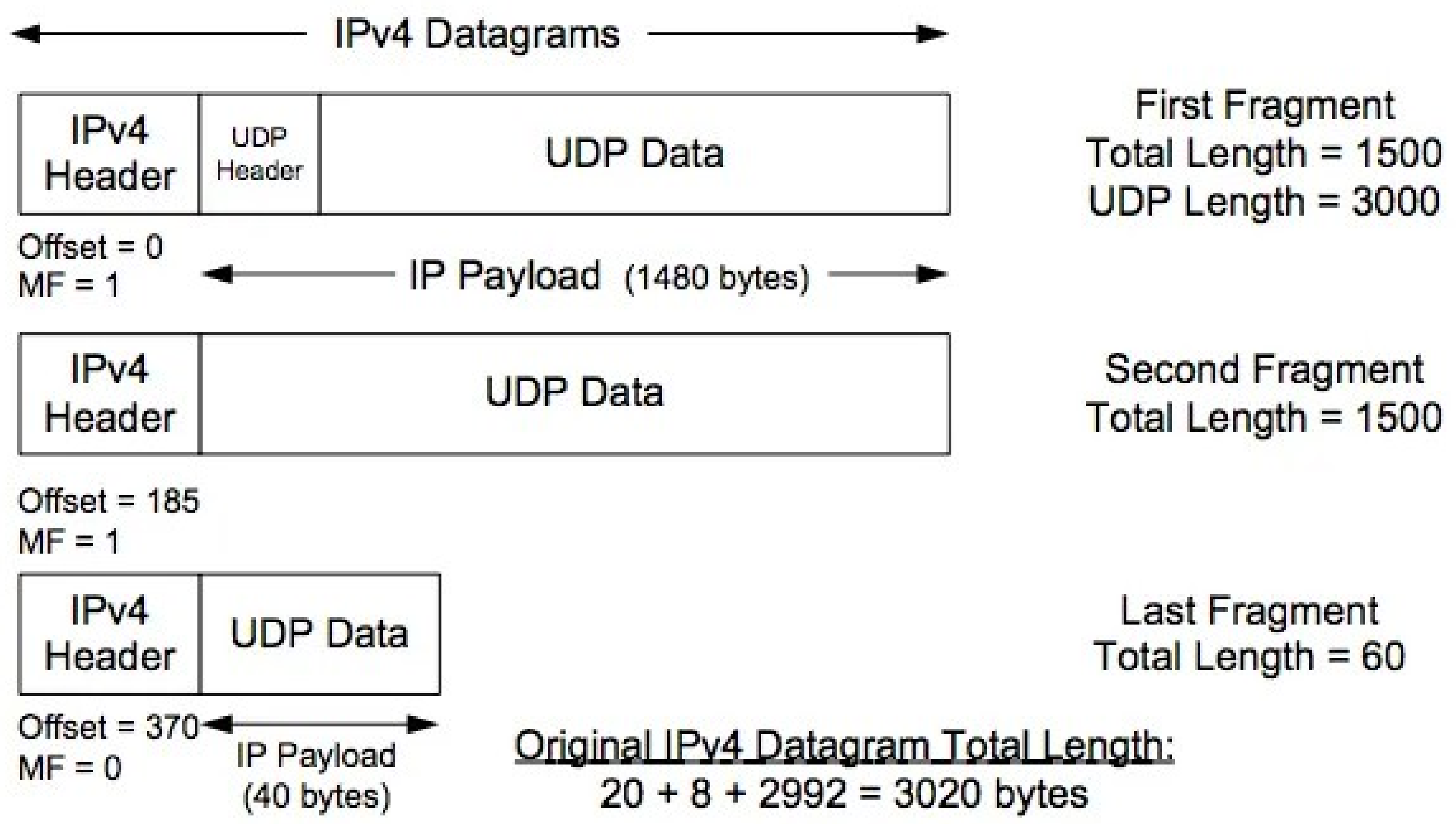


IP fragmentation

- network links have MTU (maximum transfer unit) - 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 *destination*
  - IP header bits used to identify, order related fragments



IP fragmentation



Network layer: "data plane" roadmap

4.1 Network layer: overview
- data plane
- control plane

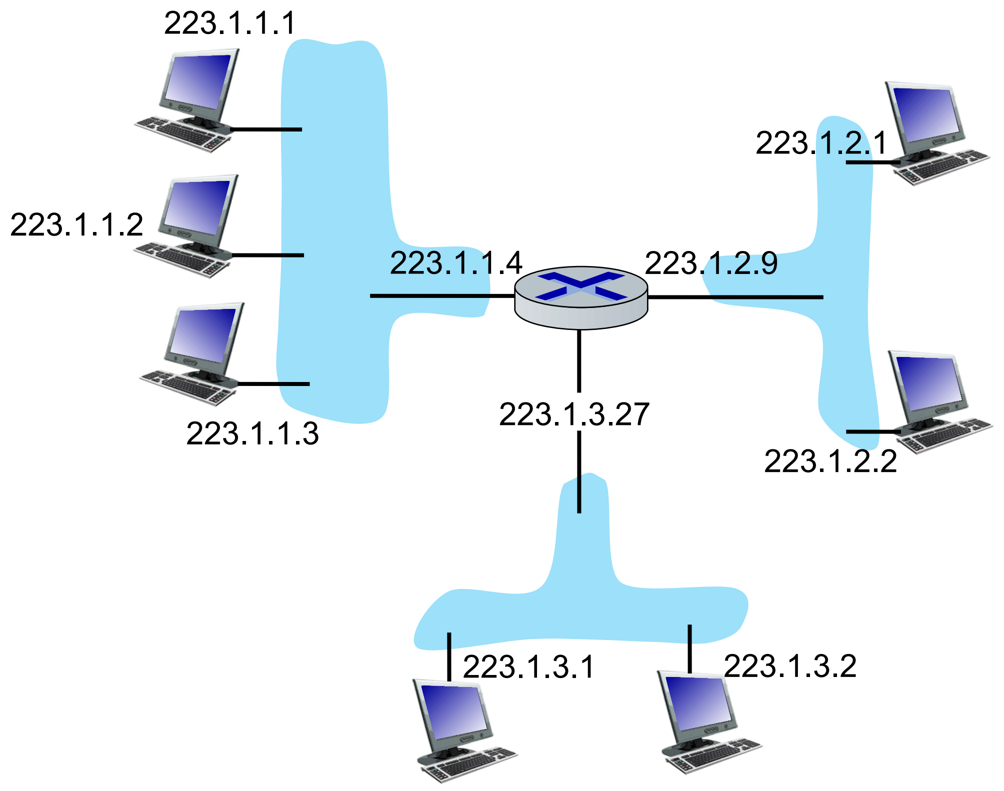
4.2 What's inside a router
- input ports, forwarding,
- switching, output ports, scheduling

4.3 The Internet Protocol (IP)
- datagram format
- addressing and subnets
- network address translation
- IPv6

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

IP addressing: introduction

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



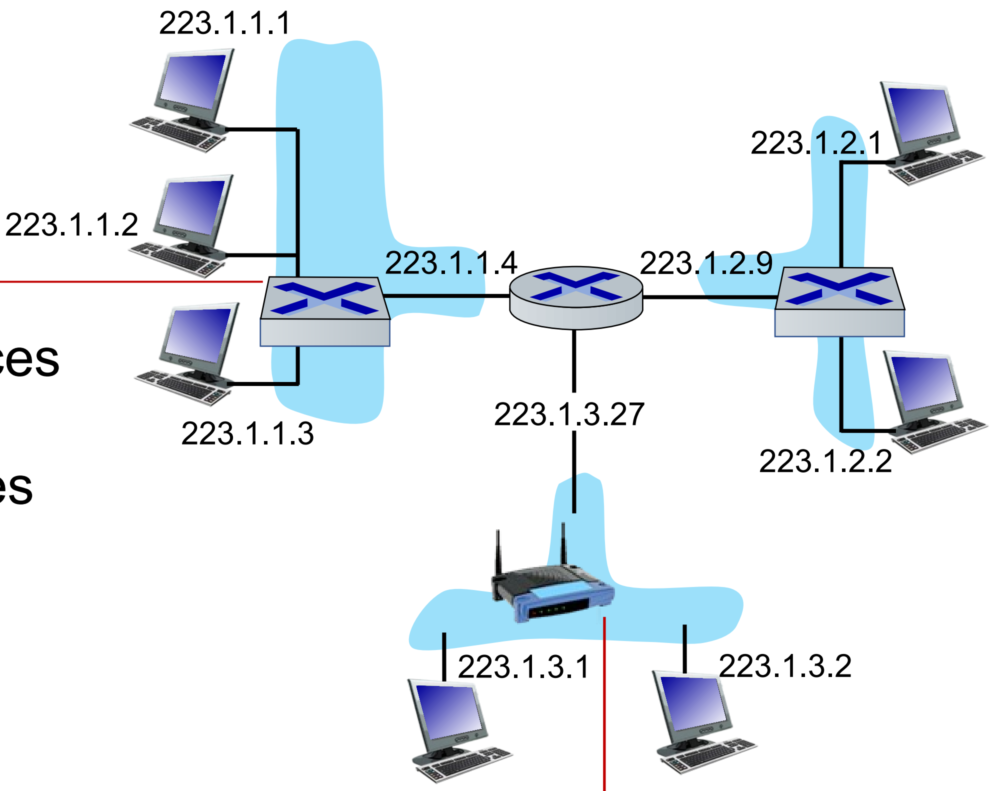
dotted-decimal IP address notation:

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

IP addressing: introduction

Q: how are interfaces actually connected?

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



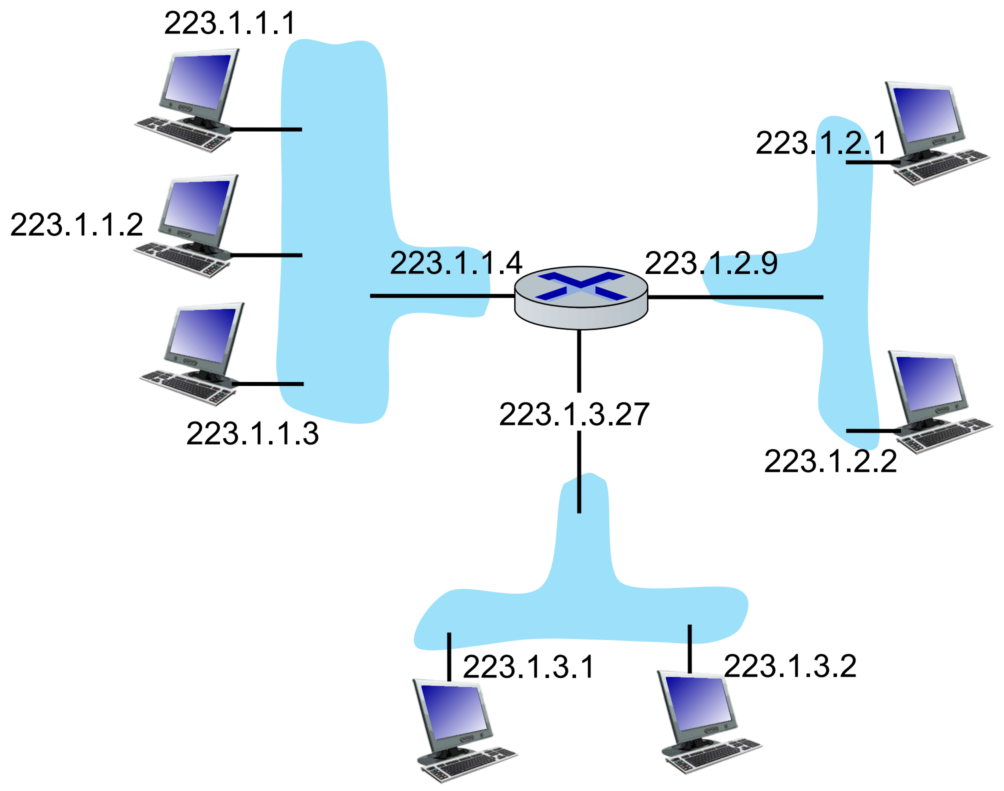
A: wired Ethernet interfaces connected by Ethernet switches

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

A: wireless WiFi interfaces connected by WiFi base station

Subnets

- *What's a subnet ?*
  - device interfaces that can physically reach each other without passing through an intervening router



network consisting of 3 subnets

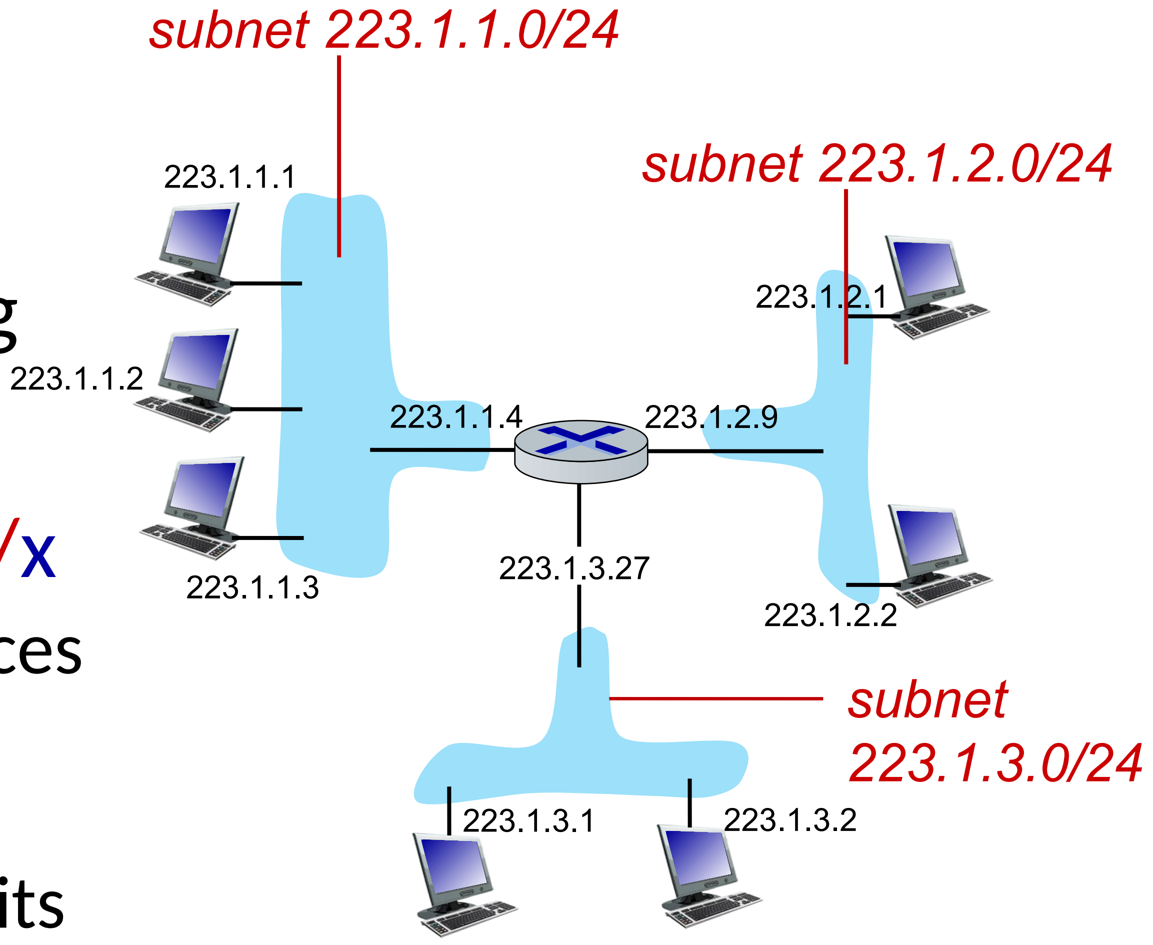
Subnets

- CIDR: Classless InterDomain Routing
- IP addresses have structure: a.b.c.d/x
  - prefix: subnet part/network part: devices in subnet have same x most significant bits (msb.)
  - host part: remaining least significant bits

← prefix (x bits) (subnet part) → ← host part →

11001000 00010111 00010000 00000000

200.23.16.0/**23**



network consisting of 3 subnets

Subnet mask vs. prefix a.b.c.d/x

- A subnet mask is a bitmask when applied by a bitwise AND operation to any IP address in the network, yields the routing prefix / subnet address

prefix = IP address && bitmask

- 198.51.100.20/24 has the subnet mask 255.255.255.0 (24 msb. are 1’s)

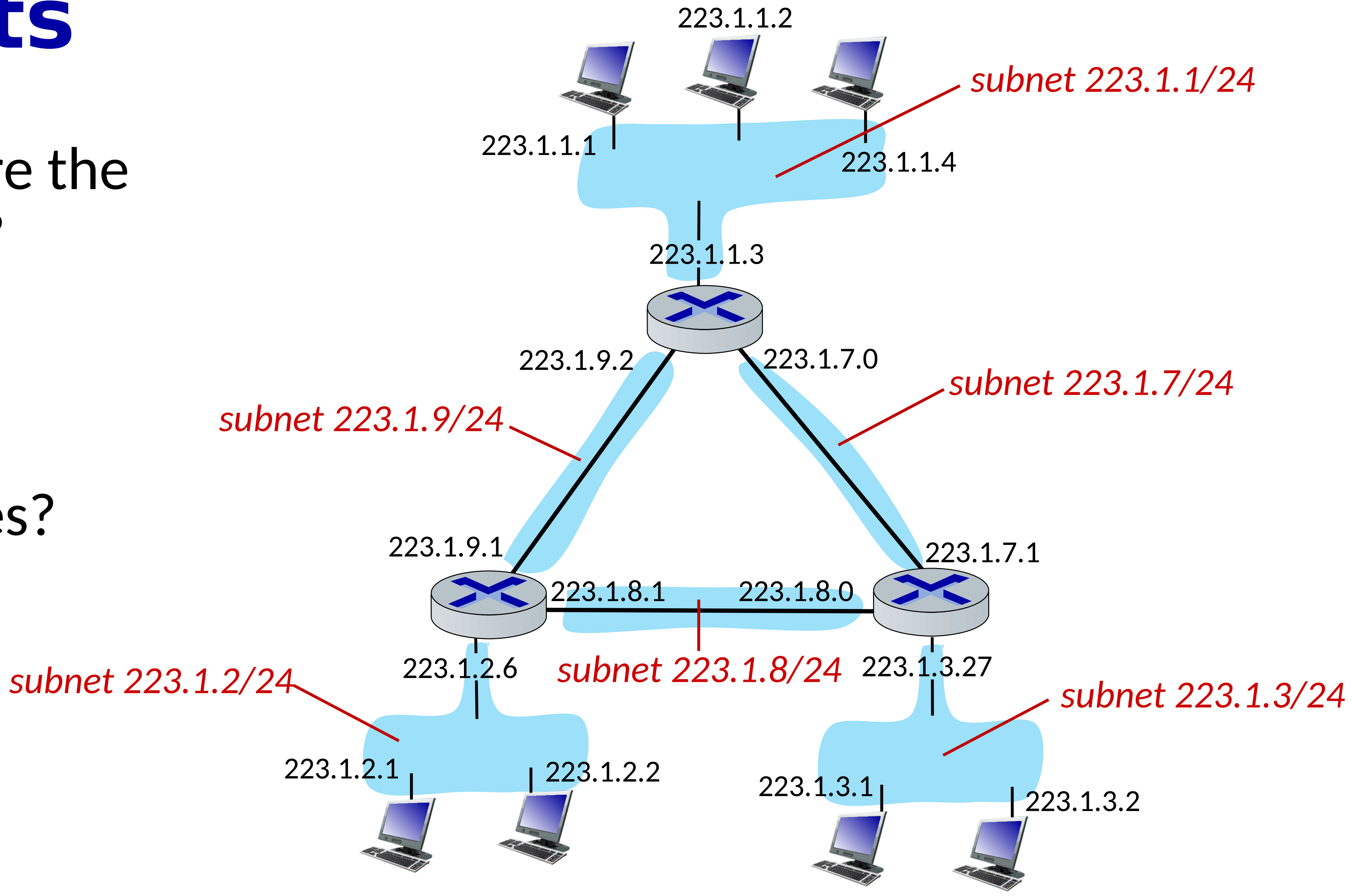
11000110 00110011 01100100 00010100 ☾ 198.51.100.20 (IP address

bitwise AND **11111111 11111111 11111111** 00000000 ☾ 255.255.255.0 (subnet ma

11000110 00110011 01100100 00000000 ☾ 198.51.100.20 (subnet ad

Subnets

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



How to get IP addresses?

That's two questions:

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

1. How gets an organization IP addresses?

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 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

2. How gets a host an IP address?

1. Manually by system admin in config file (e.g., /etc/rc.config in UNIX)

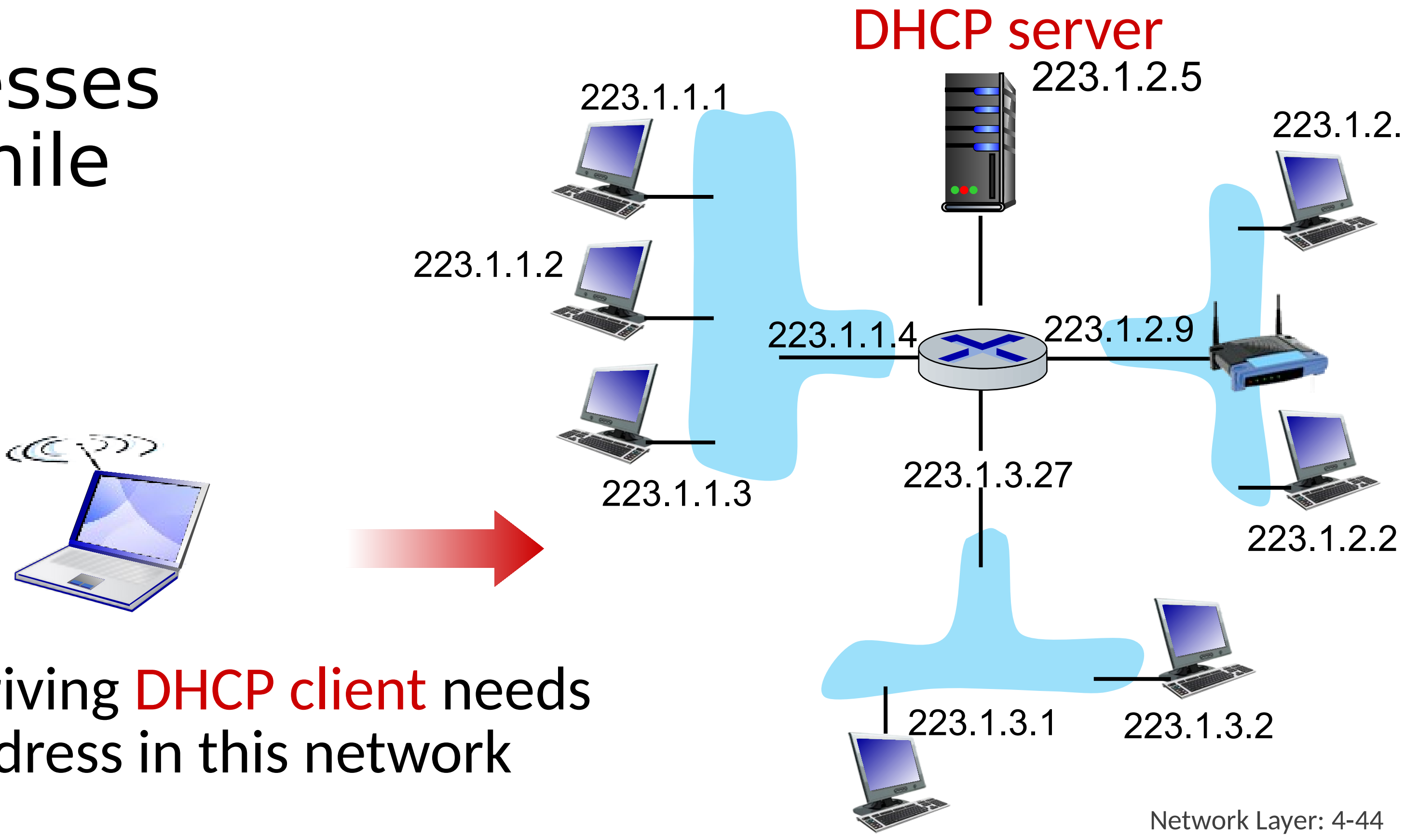
2. Dynamically from a server: DHCP: Dynamic Host Configuration Protocol
   - “plug-and-play”

DHCP client-server scenario

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

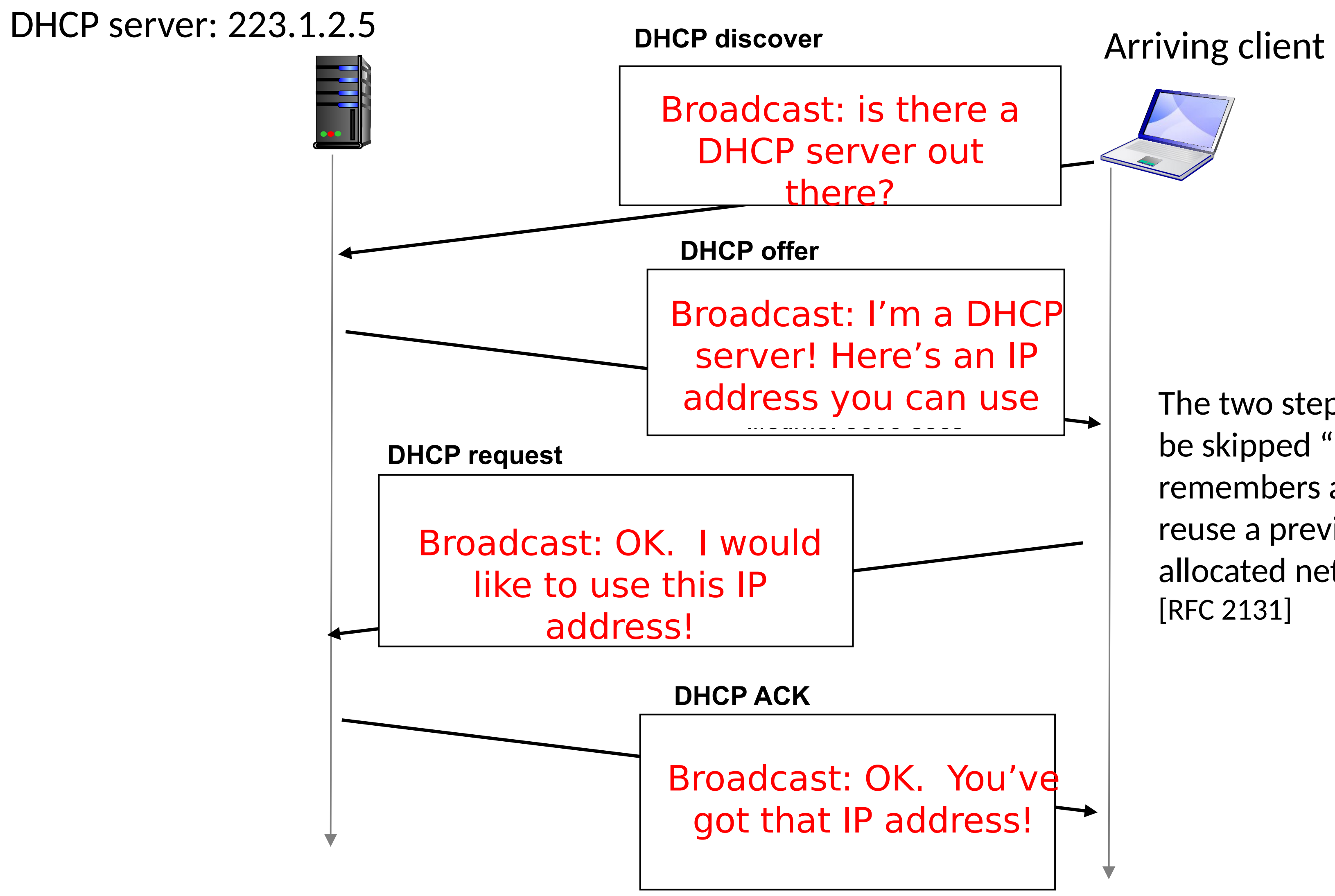
goal: host *dynamically* obtains IP address from network server when it "joins" network

- allows reuse of addresses (only hold address while connected/on)



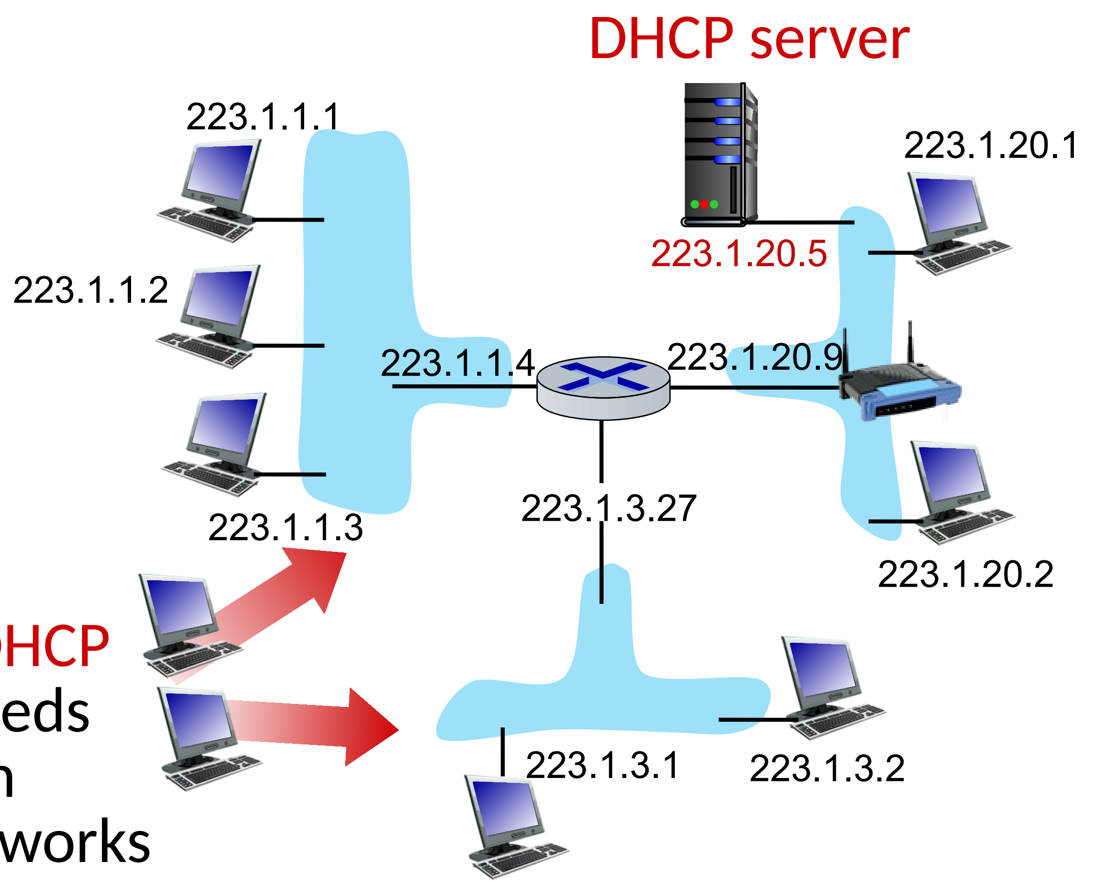
arriving DHCP client needs address in this network

DHCP client-server scenario


DHCP server: 223.1.2.5
Arriving client
DHCP discover
Broadcast: is there a DHCP server out there?
DHCP offer
Broadcast: I'm a DHCP server! Here's an IP address you can use
DHCP request
Broadcast: OK. I would like to use this IP address!
DHCP ACK
Broadcast: OK. You've got that IP address!

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

DHCP router relay agent



arriving DHCP clients needs address in these networks

DHCP server located outside router serving all subnets to which router is attached:

- Router configured as a DHCP relay agent (ip helper-address 223.1.20.5)
- Router forward DHCP requests and replies between client and DHCP server
- Router sets the gateway IP address (223.1.1.4) in giaddr field of the DHCP packet
- This allows DHCP server to identify which subnet the request originated

https://www.netmanias.com/en/post/techdocs/6000/dhcp-network-protocol/understanding-dhcp-relay-agents

DHCP: more than IP addresses

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

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

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?

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

"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

4.1 Network layer: overview
- data plane
- control plane

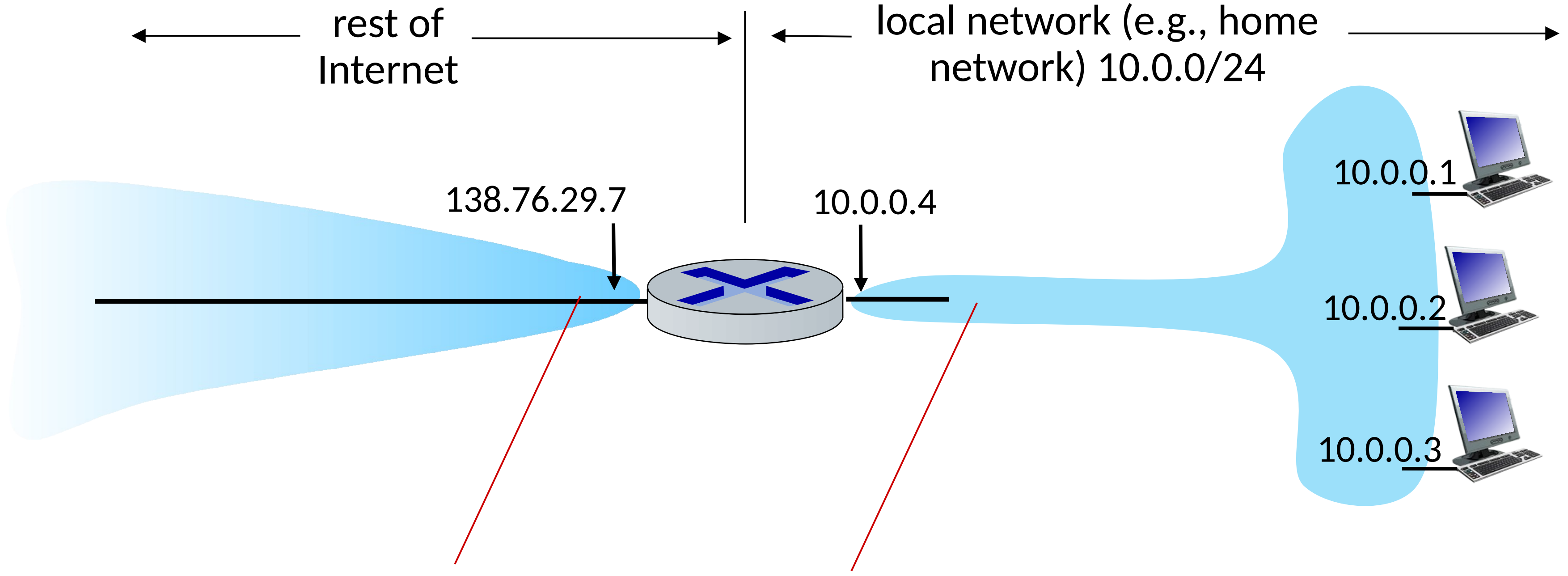
4.2 What’s inside a router
- input ports, forwarding,
- switching, output ports, scheduling

4.3 The Internet Protocol (IP)
- datagram format
- addressing
- NAT: network address translation
- IPv6

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

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

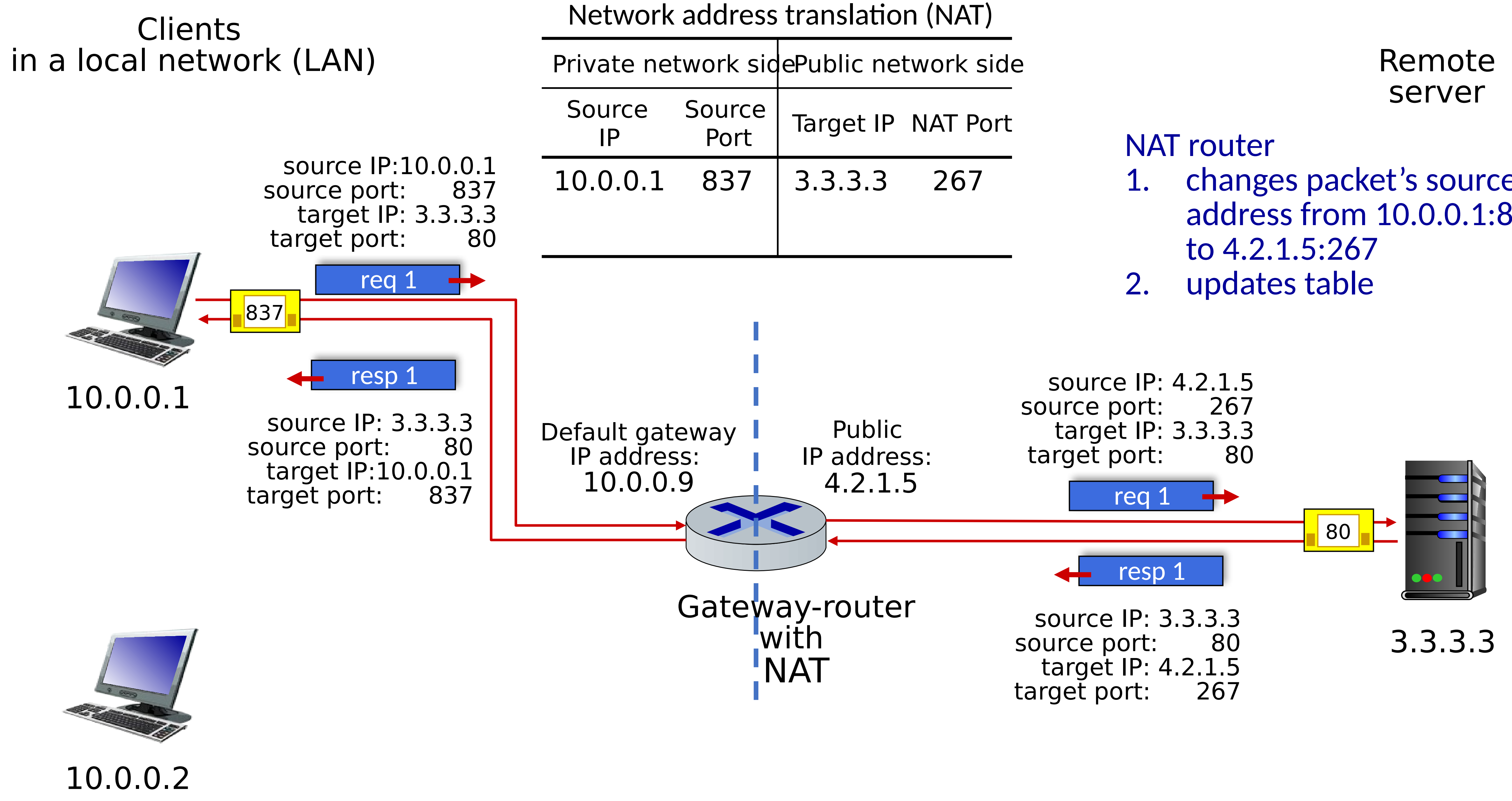
implementation: NAT router must (transparently):

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

Network address translation (NAT)

Private network side		Public network side	
Source IP	Source Port	Target IP	NAT Port
10.0.0.1	837	3.3.3.3	267

NAT router
1. changes packet’s source address from 10.0.0.1:837 to 4.2.1.5:267
2. updates table



Clients
in a local network (LAN)

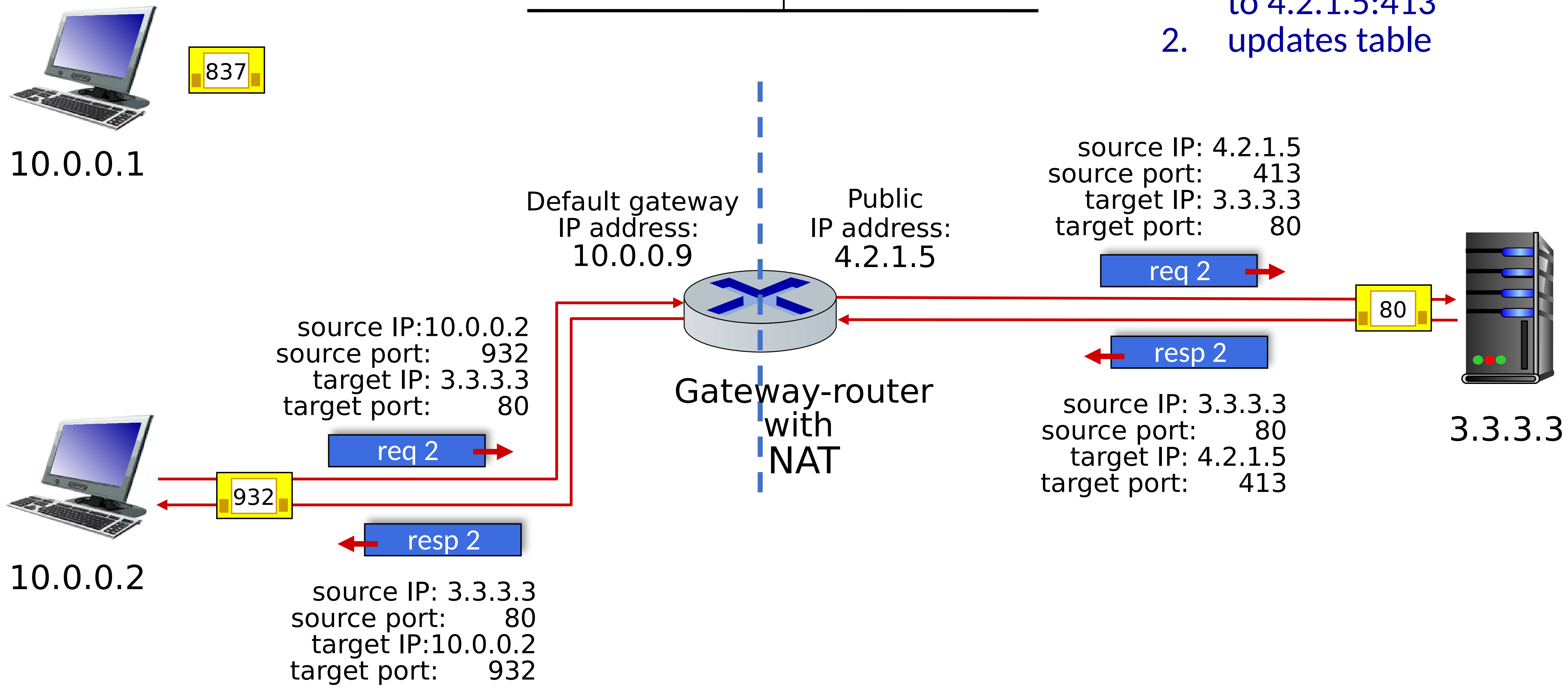
Network address translation (NAT)

Private network side		Public network side	
Source IP	Source Port	Target IP	NAT Port
10.0.0.1	837	3.3.3.3	267
10.0.0.2	932	3.3.3.3	413

Remote
server

NAT router

1. changes packet’s source address from 10.0.0.2:932 to 4.2.1.5:413
2. updates table



NAT: network address translation

- all devices in local network have 32-bit addresses in a “private” IP address space (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16 prefixes) that can only be used in local network
- advantages:
  - just one IP address needed from provider ISP for *all* devices
  - can change addresses of host in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - security: devices inside local net not directly addressable, visible by outside world

Network layer: "data plane" roadmap

4.1 Network layer: overview
- data plane
- control plane

4.2 What's inside a router
- input ports, forwarding,
- switching, output ports, scheduling

4.3 The Internet Protocol (IP)
- datagram format
- addressing
- NAT: network address translation
- IPv6

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

IPv6: motivation

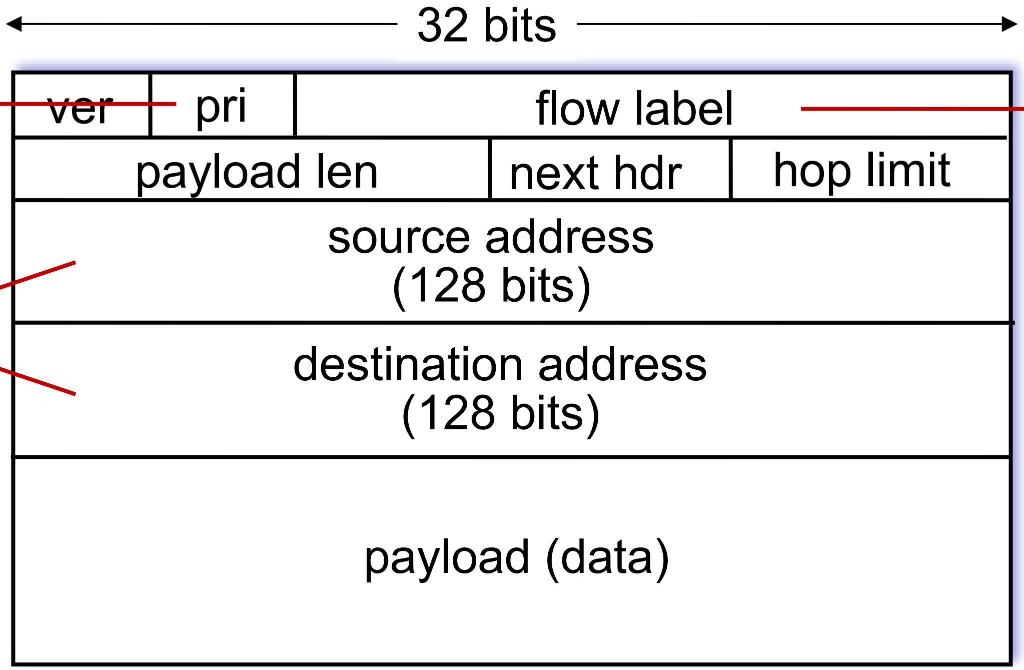
- initial motivation: 32-bit IPv4 address space would be completely allocated
- additional motivation:
  - speed processing/forwarding: 40-byte fixed length header
  - enable different network-layer treatment of “flows”

IPv6 datagram format

priority: identify priority among datagrams in flow

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

128-bit IPv6 addresses

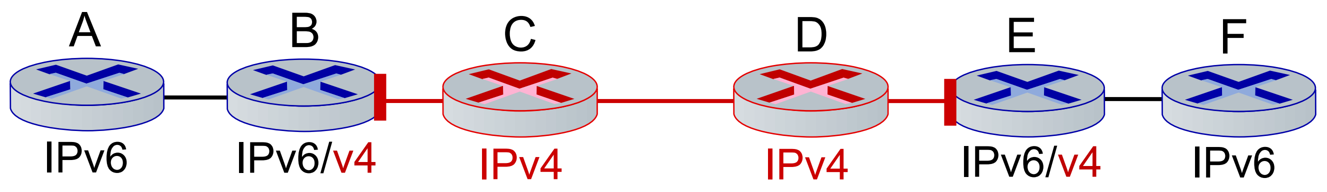


What's missing (compared with IPv4):

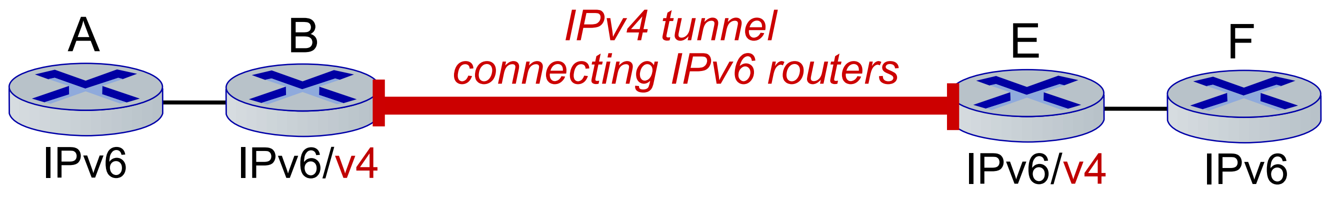
- no checksum (to speed processing at routers)
- no fragmentation/reassembly
- no options (available as upper-layer, next-header protocol at router)

Tunneling

physical view:

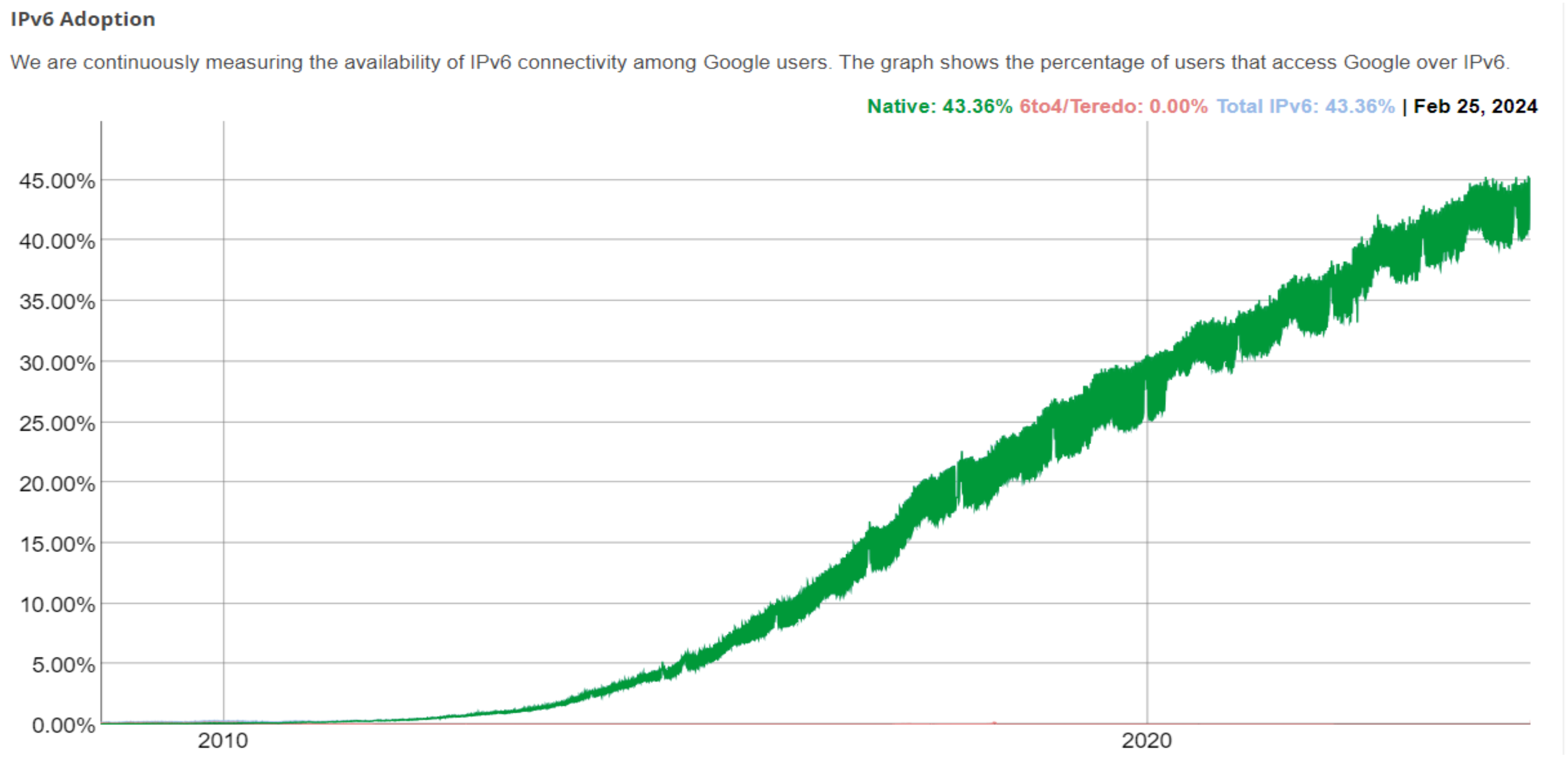


logical view:



IPv6: adoption

- Google[1]: ~ 40% of clients access services via IPv6 (2023)
- NIST: 1/3 of all US government domains are IPv6 capable



IPv6 Adoption

We are continuously measuring the availability of IPv6 connectivity among Google users. The graph shows the percentage of users that access Google over IPv6.

Native: 43.36% 6to4/Teredo: 0.00% Total IPv6: 43.36% | Feb 25, 2024

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

Network layer: “data plane” roadmap

4.1 Network layer: overview
- data plane
- control plane

4.2 What’s inside a router
- input ports, forwarding,
- switching, output ports, scheduling

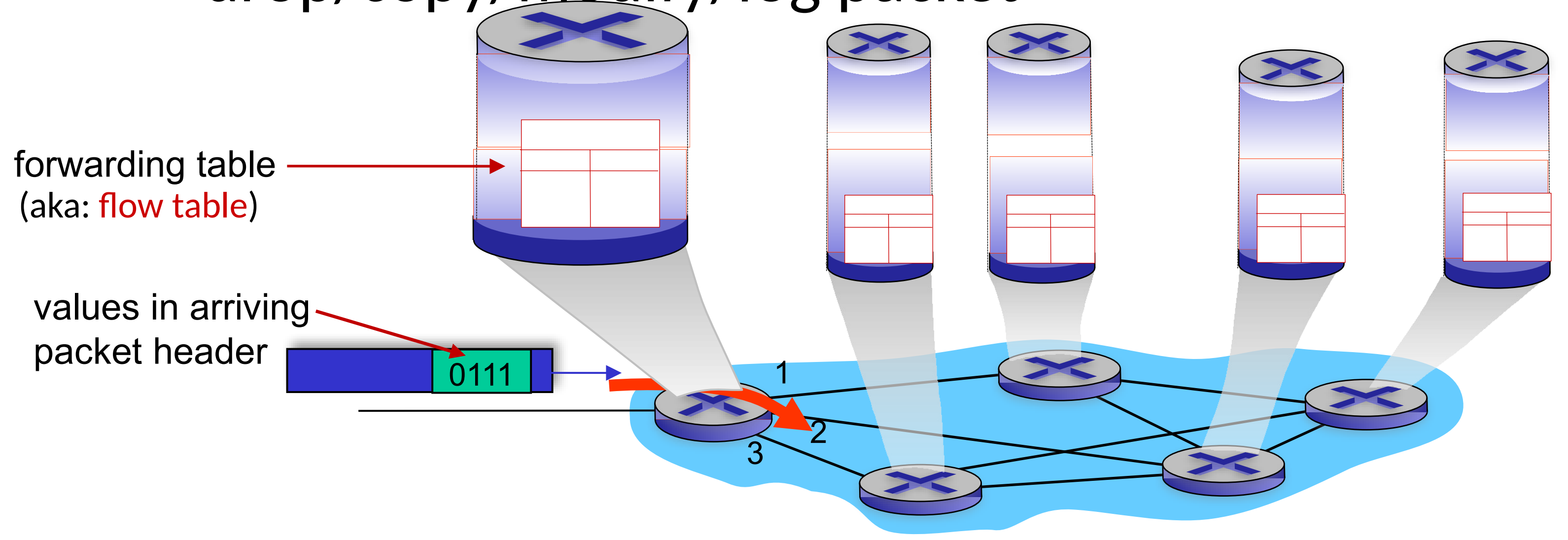
4.3 IP: the Internet Protocol
- datagram format
- addressing
- network address translation
- IPv6

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

Generalized forwarding: match plus action

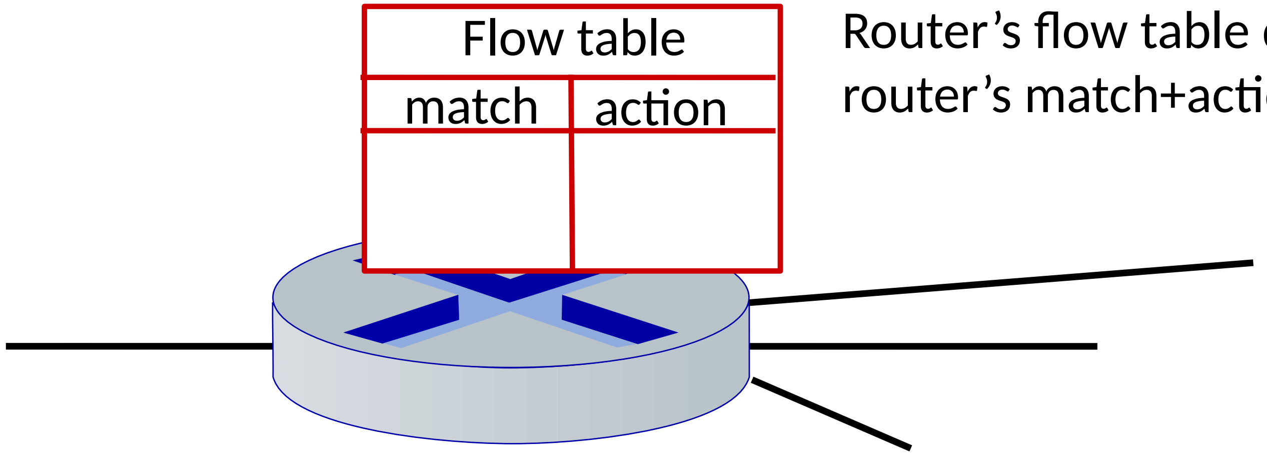
Review: each router contains a forwarding table (aka: flow table)

- "match plus action" abstraction: match bits in arriving packet, take action
  - *destination-based forwarding:* forward based on dest. IP address
  - *generalized forwarding*:
    - many header fields can determine action
    - many action possible: drop/copy/modify/log packet



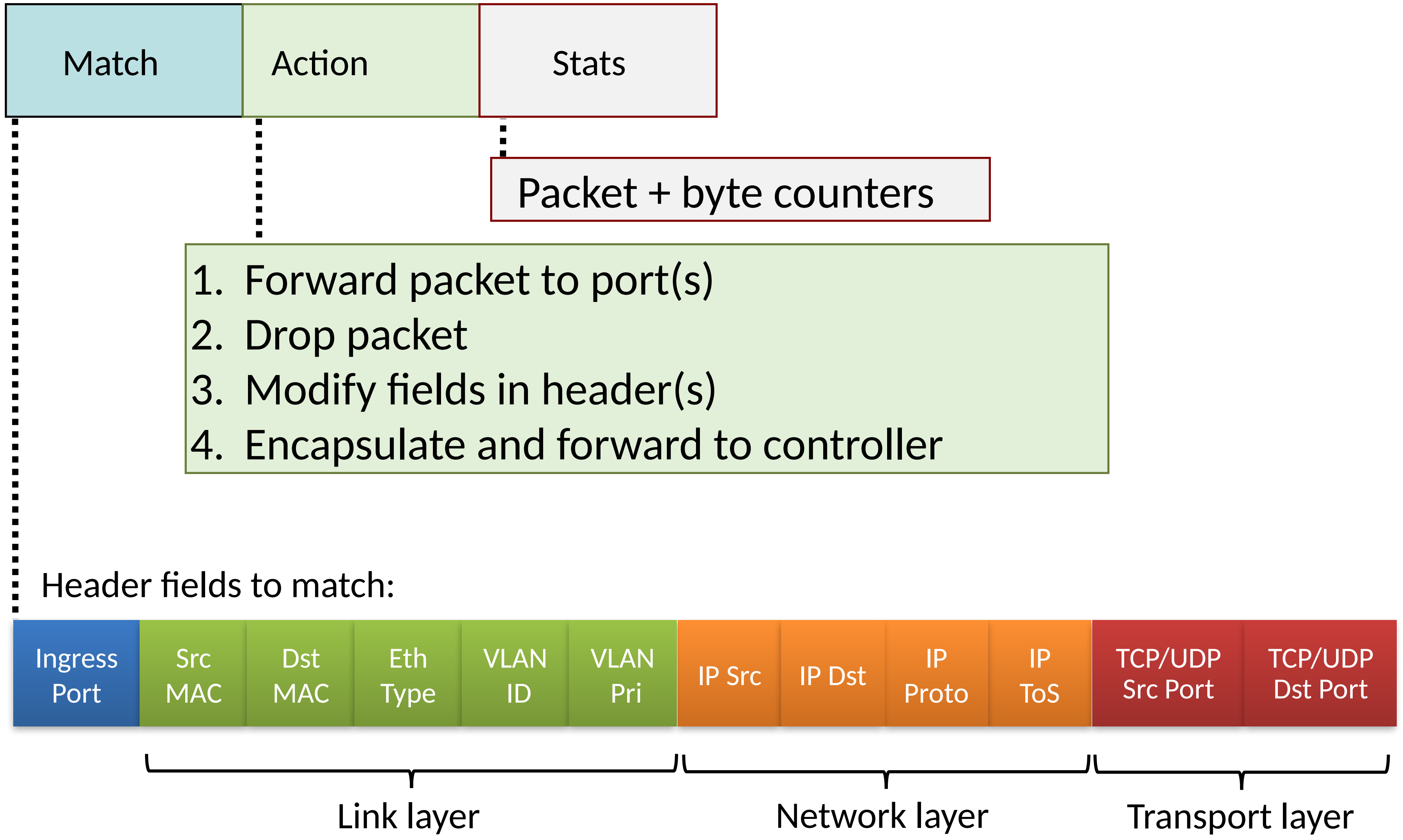
Flow table abstraction

- flow: defined by header field values (in link-, network-, transport-layer fields)
- generalized forwarding: simple packet-handling rules
  - match: pattern values in packet header fields
  - actions: for matched packet: drop, forward, modify, matched packet or send matched packet to controller
  - priority: disambiguate overlapping patterns
  - counters: #bytes and #packets



Router's flow table define router's match+action rules

OpenFlow: flow table entries



1. Forward packet to port(s)
2. Drop packet
3. Modify fields in header(s)
4. Encapsulate and forward to controller

OpenFlow: examples

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	51.6.0.8	*	*	*	*	port6

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	*	*	*	*	22	drop

Block (do not forward) all datagrams destined to TCP port 22 (ssh port #)

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	128.119.1.1	*	*	*	*	*	drop

Block (do not forward) all datagrams sent by host 128.119.1.1

OpenFlow: examples

Layer 2 destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	22:A7:23: 11:E1:02	*	*	*	*	*	*	*	*	*	port3

layer 2 frames with destination MAC address 22:A7:23:11:E1:02 should be forwarded to output port 3

OpenFlow abstraction

- match+action: abstraction unifies different kinds of devices

Router
- *match:* longest destination IP prefix
- *action:* forward out a link

Switch
- *match:* destination MAC address
- *action:* forward or flood

Firewall
- *match*: IP addresses and TCP/UDP port numbers
- *action:* permit or deny

NAT
- *match:* IP address and port
- *action:* rewrite address and port

Chapter 4: done!

- Network layer: overview
- What's inside a router
- IP: the Internet Protocol
- Generalized Forwarding, SDN

Question: how are forwarding tables (destination-based forwarding) or flow tables (generalized forwarding) computed?

Answer: by the control plane (next chapter)