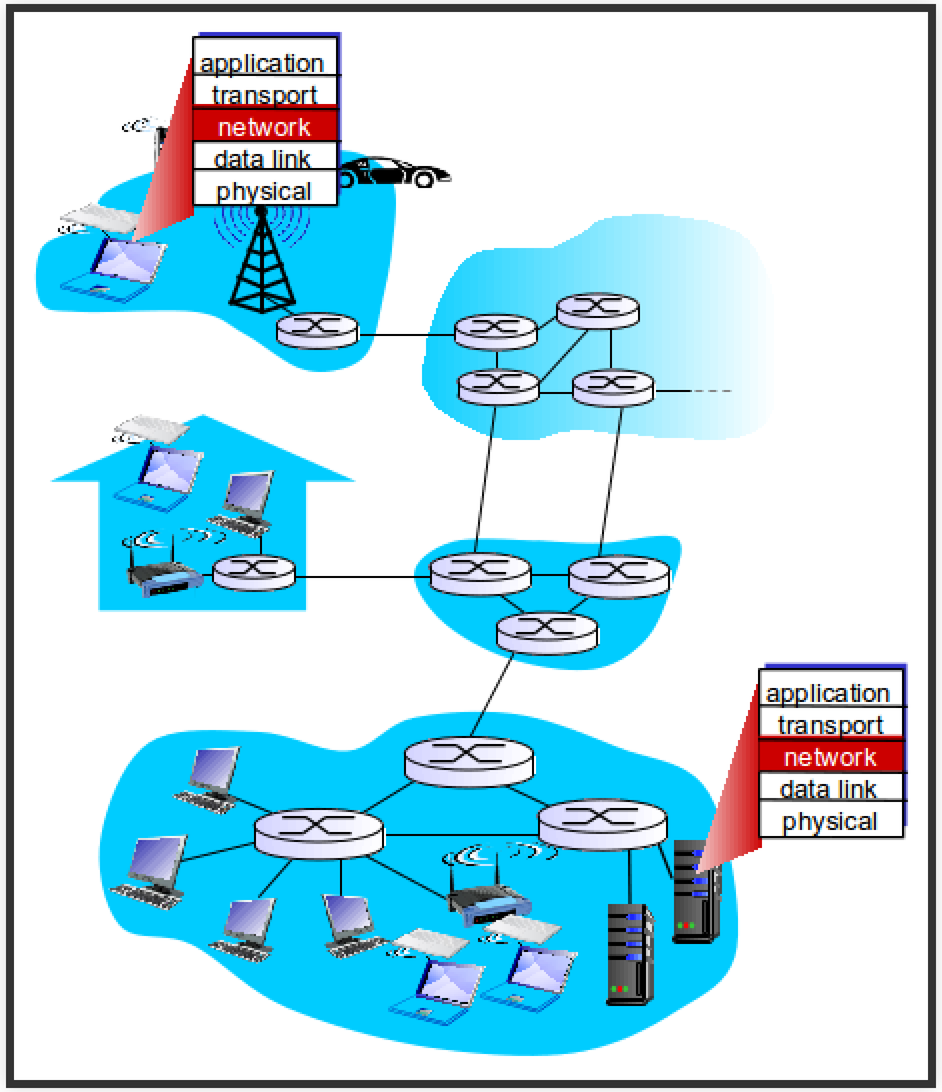
NETWORK LAYER: DATA PLANE

GOALS

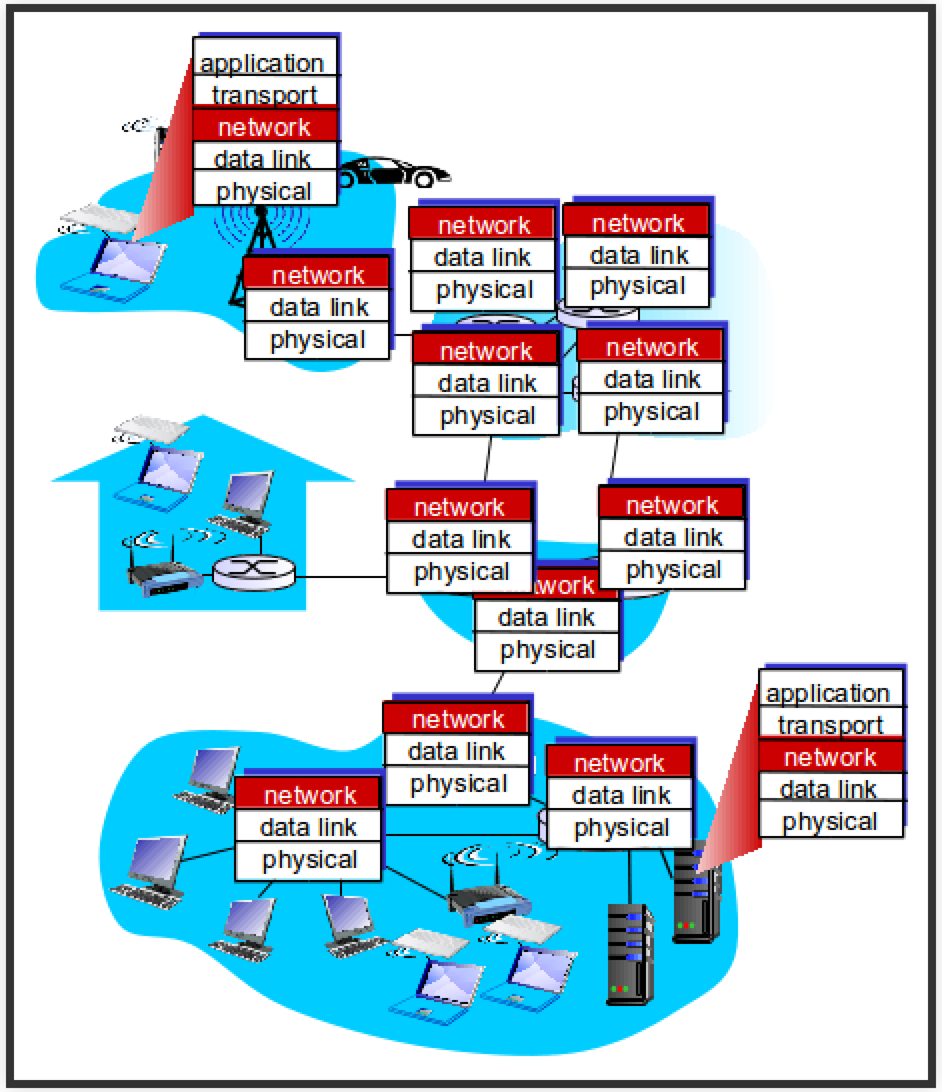
- Understand principles behind network layer services, focusing on the data plane:
    - Network layer service models
    - Forwarding versus routing
    - How a router works
    - Generalized forwarding
- Instantiation, implementation in the Internet

💡 Per router functions (Network wide is chapter 5)

INTRODUCTION



INTRODUCTION



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

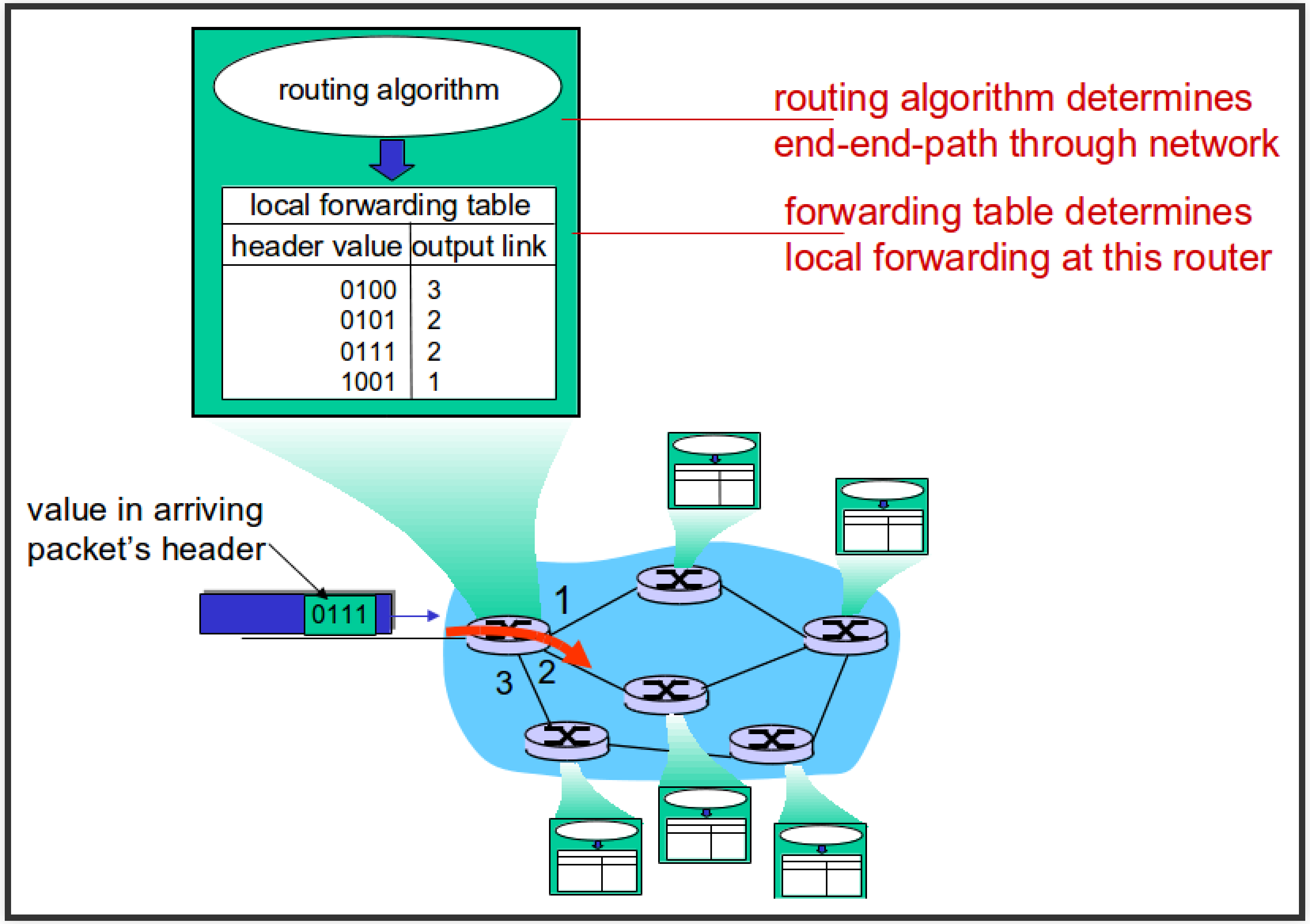
- **Forwarding:** move packets from router’s input to appropriate router output

Analogy - Traveling: process of getting through single interchange

- **Routing:** determine route taken by packets from source to dest.

Analogy - Traveling: process of planning trip from source to dest

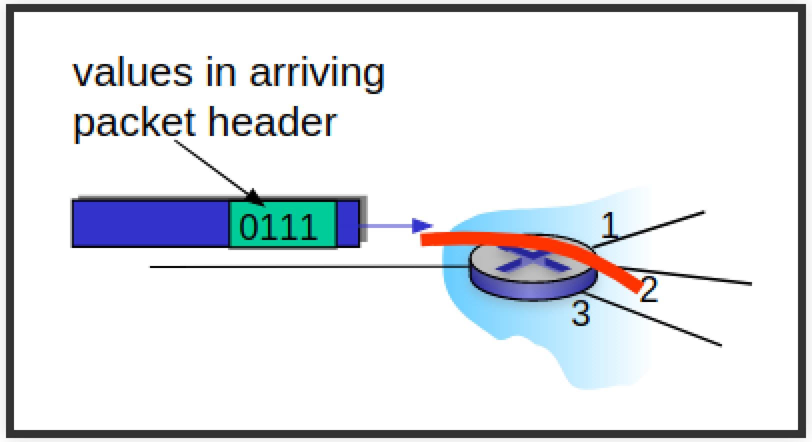
INTERPLAY: ROUTING AND FORWARDING


routing algorithm
local forwarding table
value in arriving packet's header
0111
1
2
3
routing algorithm determines end-end-path through network
forwarding table determines local forwarding at this router

header value	output link
0100	3
0101	2
0111	2
1001	1

NETWORK LAYER: DATA PLANE

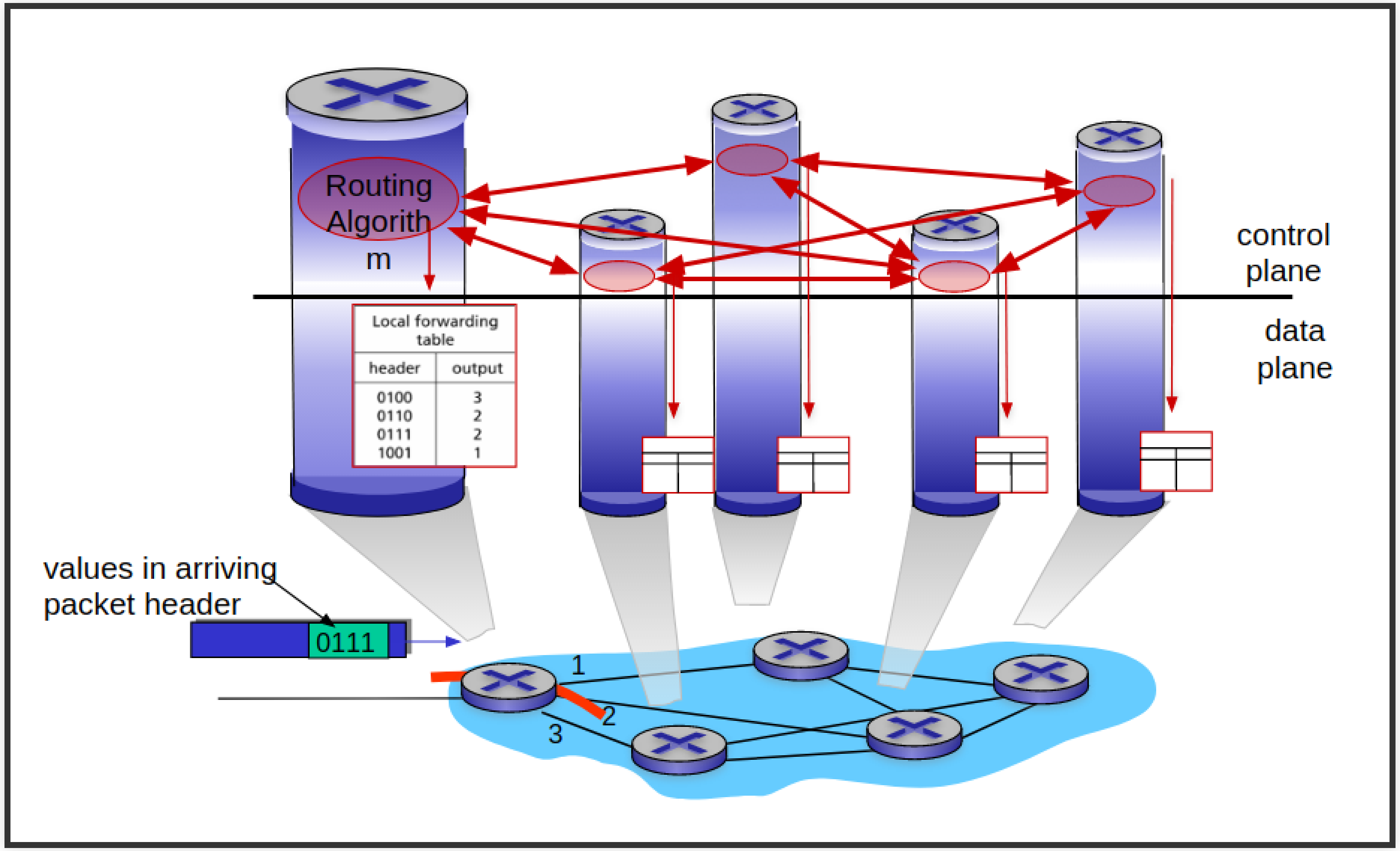
- local, per-router function
- determines how datagram arriving on router input port is forwarded to router output port
- Forwarding function



NETWORK LAYER: 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

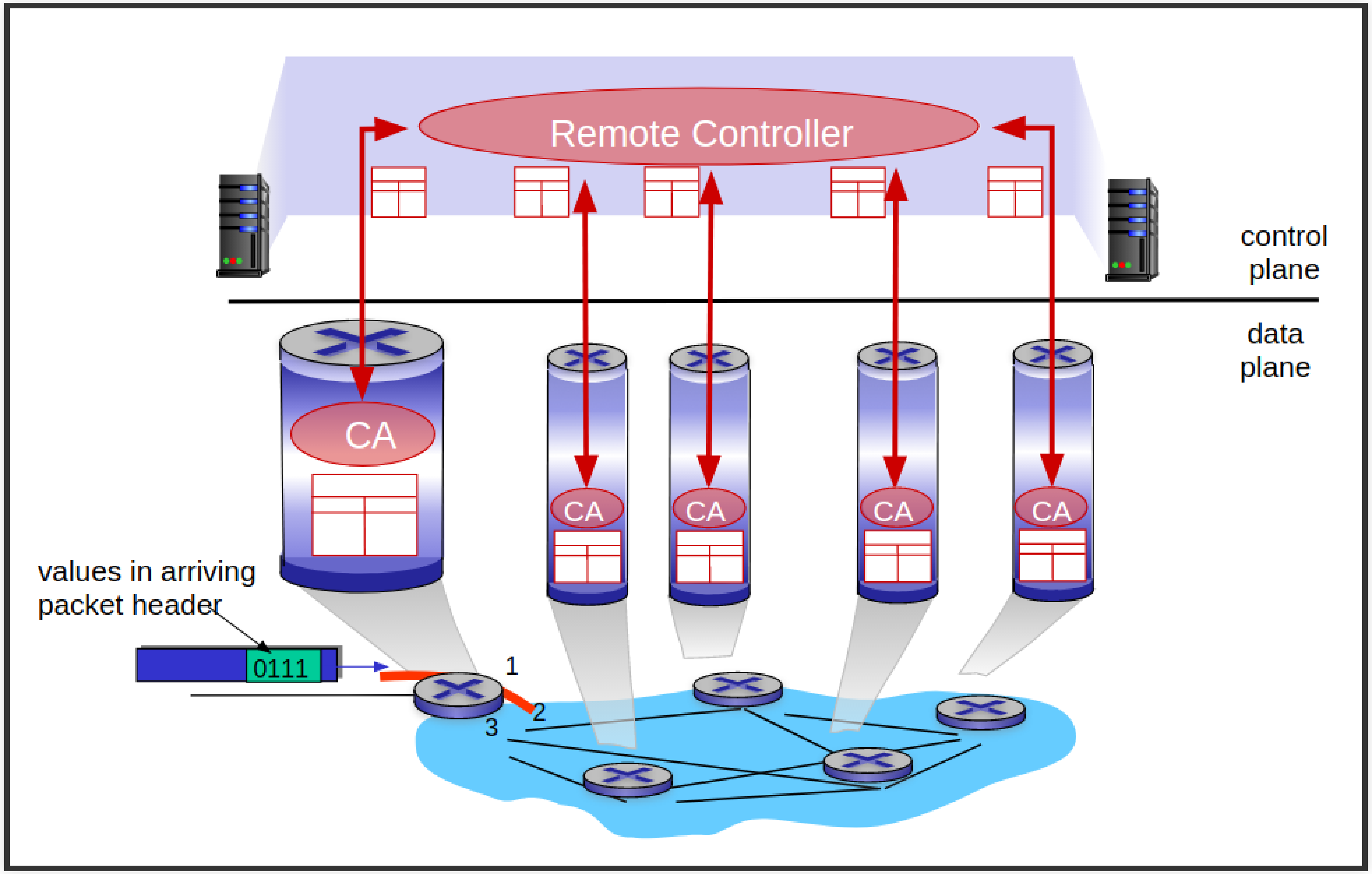
PER-ROUTER CONTROL PLANE



Local forwarding table

header	output
0100	3
0110	2
0111	2
1001	1

LOGICALLY CENTRALIZED CONTROL PLANE (SDN)



NETWORK SERVICE MODEL

Q: What service model for “channel” transporting datagrams from sender to receiver?

example services for individual datagrams:

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

example services for a flow of datagrams:

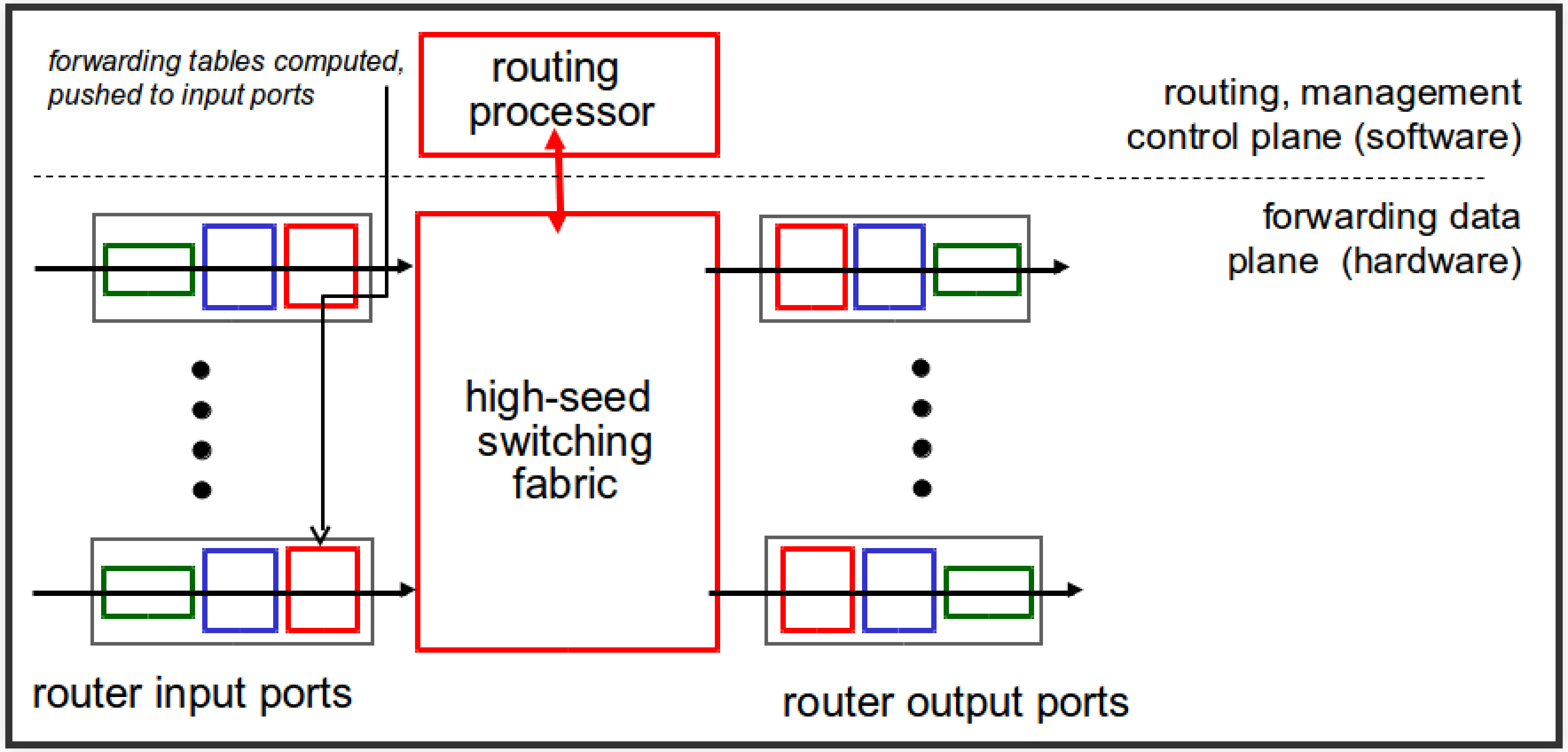
- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing

SERVICES AND GUARANTEES

Network Archtecture	Service Model	Bandwidth	Loss	Order	Timing	Congesti feedback
Internet	Best efford	None	No	No	No	No (inferred via loss)
ATM	CBR	Constant rate	Yes	Yes	Yes	No congesti

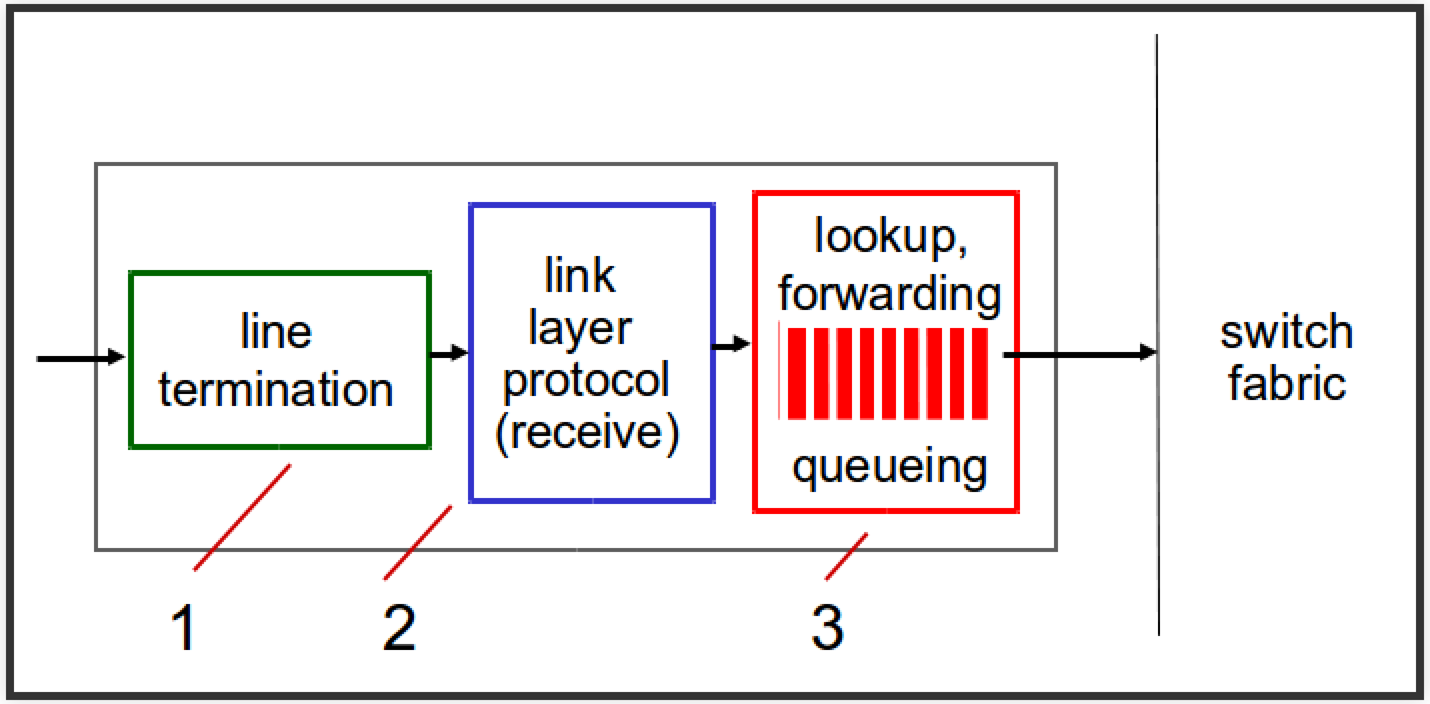
WHAT'S INSIDE A ROUTER

ROUTER ARCHITECTURE OVERVIEW



forwarding tables computed, pushed to input ports
routing processor
routing, management control plane (software)
forwarding data plane (hardware)
high-seed switching fabric
router input ports
router output ports

INPUT PORT FUNCTIONS



INPUT PORT FUNCTIONS

- **Physical layer:** bit-level reception
- **Data link layer:** e.g., Ethernet (chapter 5)
- **Decentralized switching:**
  - given datagram dest., 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

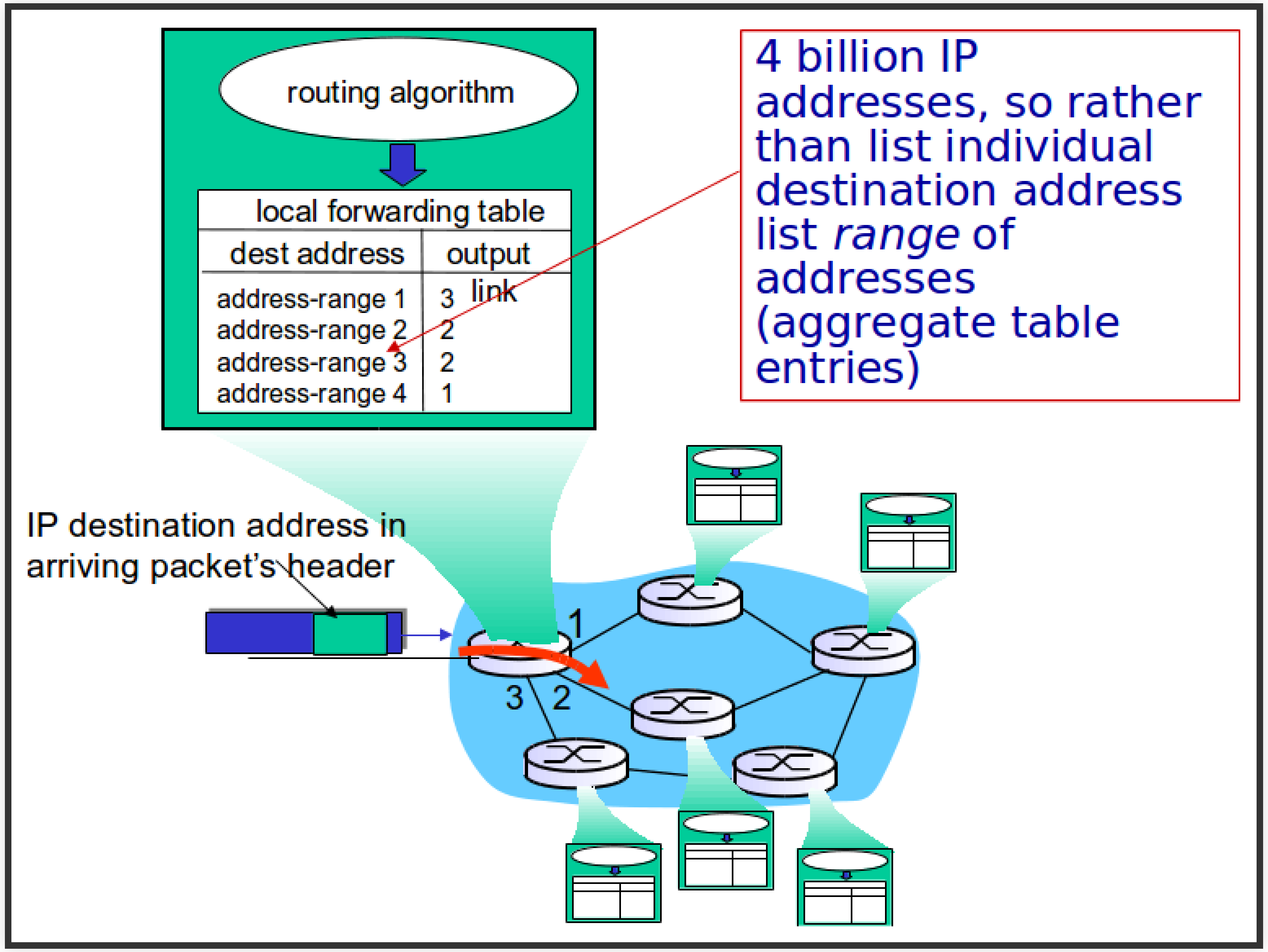
- **Destination-based forwarding:** forward based only on destination IP address (traditional)
- **Generalized forwarding:** forward based on any set of header field values

DATAGRAM FORWARDING TABLE

With 4 billion IP addresses, it would become a large table if it was a plain map!

How can we solve this?

DATAGRAM FORWARDING TABLE



4 billion IP addresses, so rather than list individual destination address list *range* of addresses (aggregate table entries)

DATAGRAM FORWARDING TABLE

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 to 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 to 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 to 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.

LONGEST PREFIX MATCHING

Destination Address Range	Link Interface
11001000 00010111 00010*** *********	0
11001000 00010111 00011000 *********	1
11001000 00010111 00011*** *********	2
otherwise	3

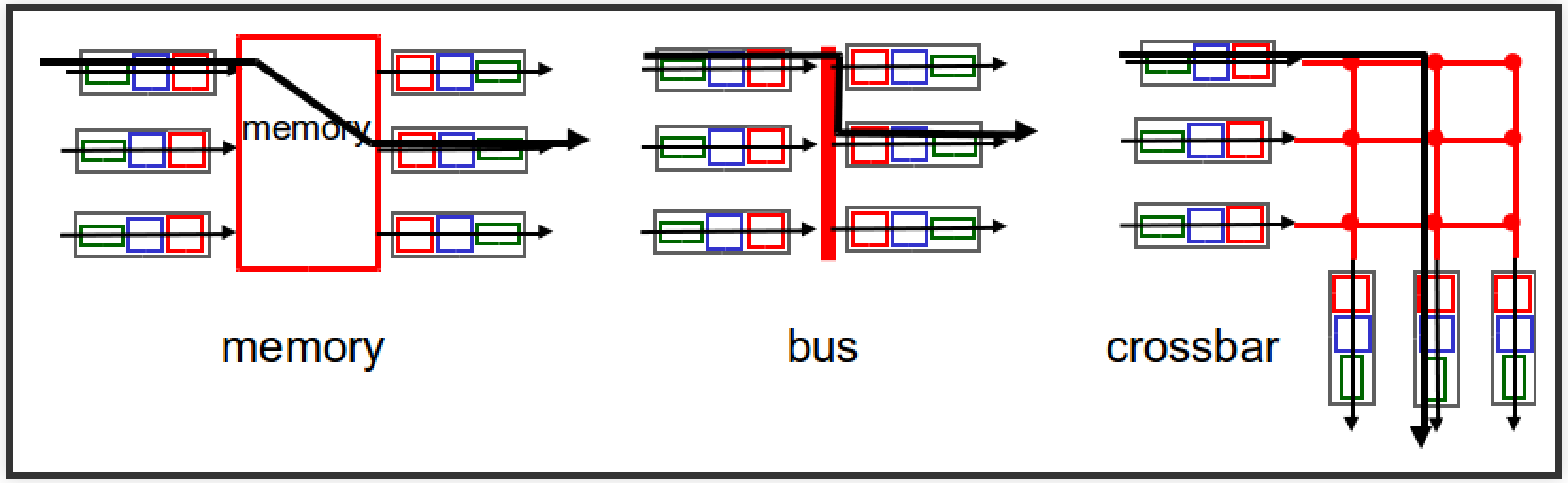
⚠ Example: which interface? DA: 11001000 00010111 00010110 10100001

LONGEST PREFIX MATCHING

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

Once output port has been determined → sent into Switching fabric

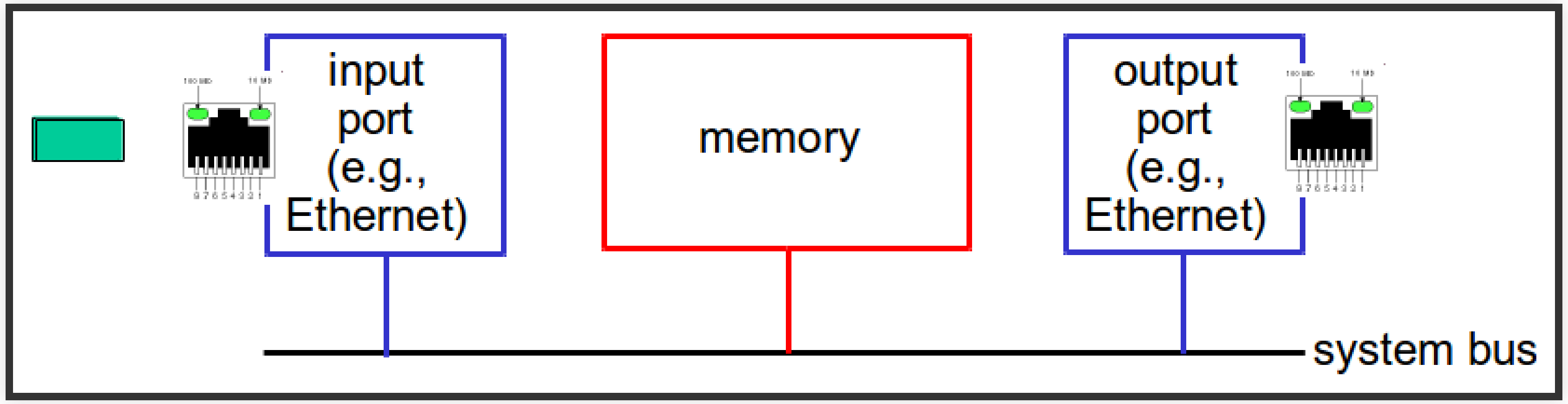
SWITCHING FABRICS



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

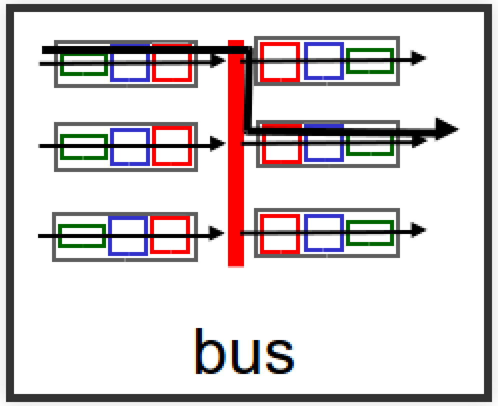


SWITCHING VIA MEMORY

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)
- Only one packet moved - even if different destination ports

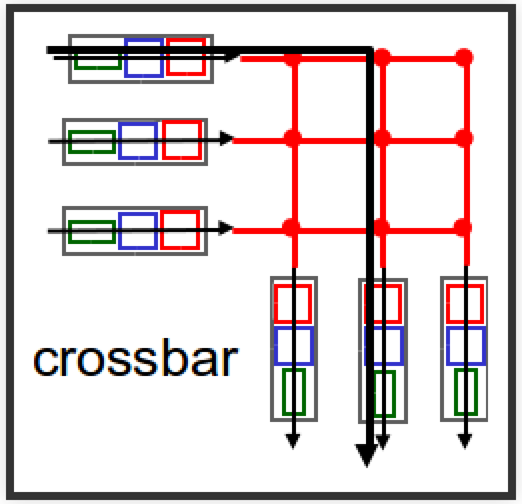
SWITCHING VIA A BUS



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

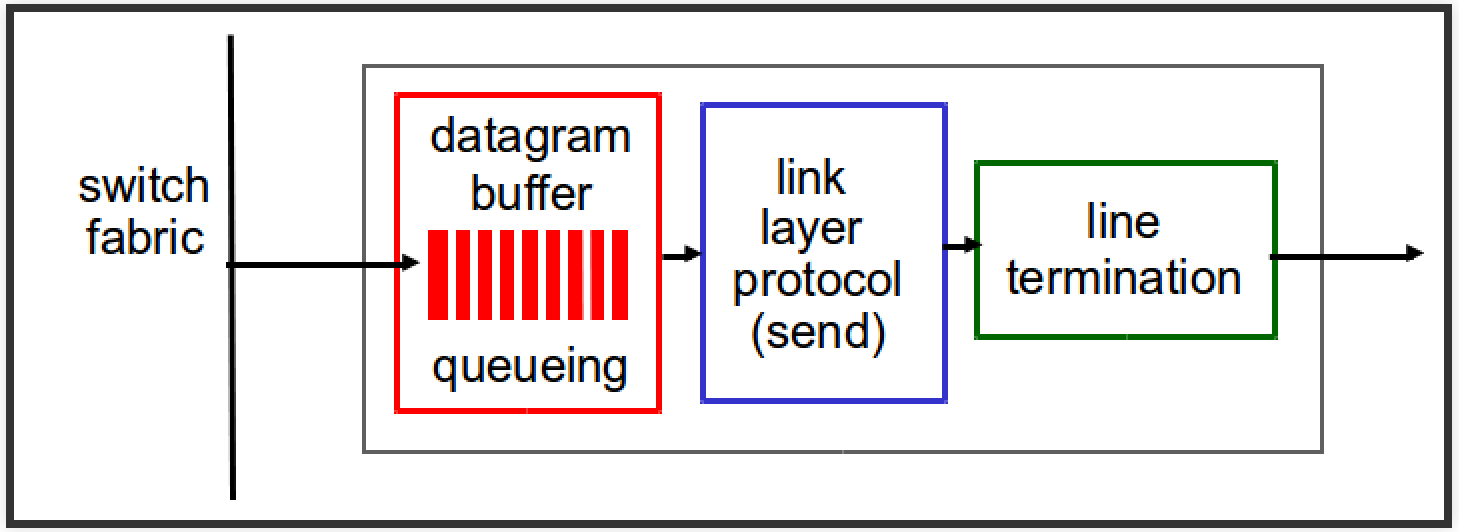
SWITCHING VIA INTERCONNECTION NETWORK



SWITCHING VIA INTERCONNECTION NETWORK

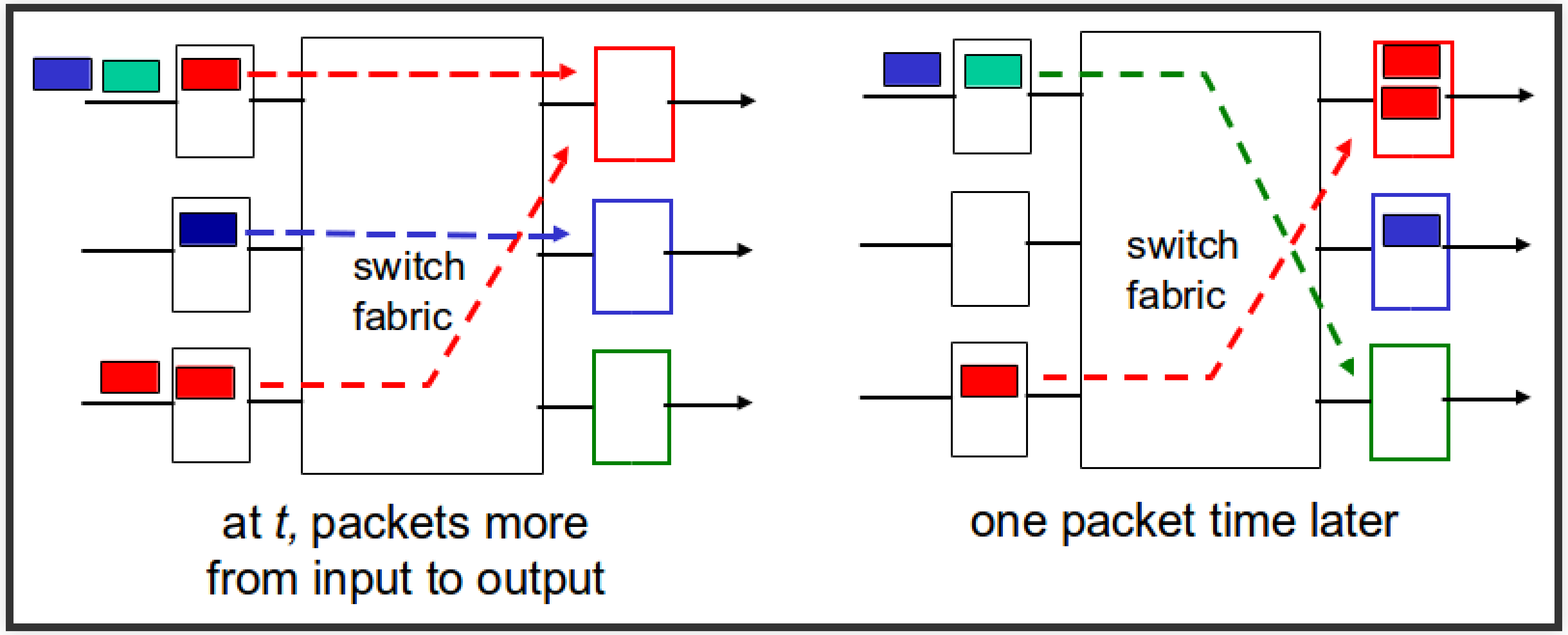
- Overcome bus bandwidth limitations
- banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection network
- Packets can be switched if they have different destination ports

OUTPUT PORTS



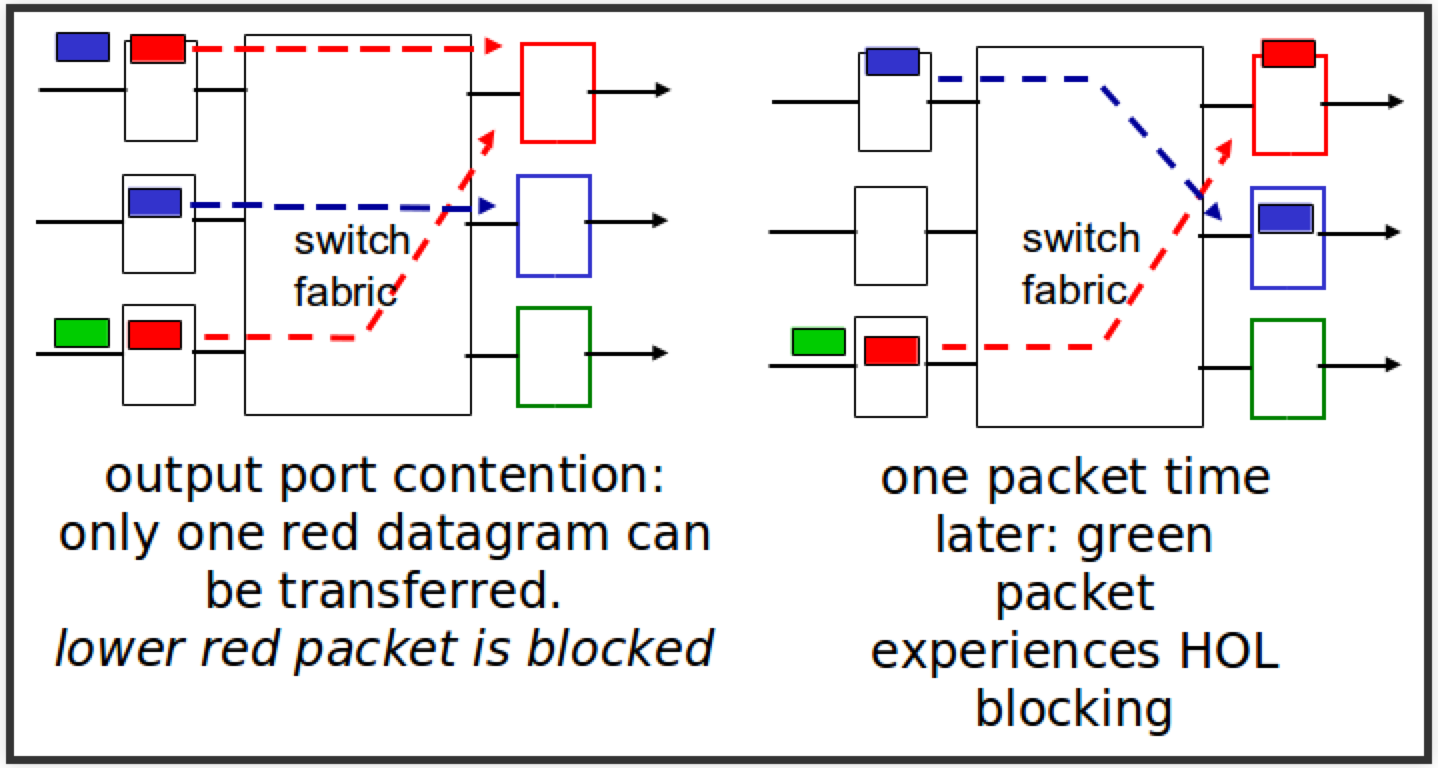
- **Buffering** required when datagrams arrive from fabric faster than the transmission rate
- **scheduling discipline** chooses among queued datagrams for transmission

OUTPUT PORT QUEUEING



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

INPUT PORT QUEUING



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

one packet time later: green packet experiences HOL blocking

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

HOW MUCH BUFFERING?

- RFC 3439 rule of thumb: average buffering equal to “typical” RTT (say 250 msec) times link capacity C
    - e.g., C = 10 Gpbs link: 2.5 Gbit buffer
- recent recommendation: with N flows, buffering equal to (C RTT)/(sqrt(N))

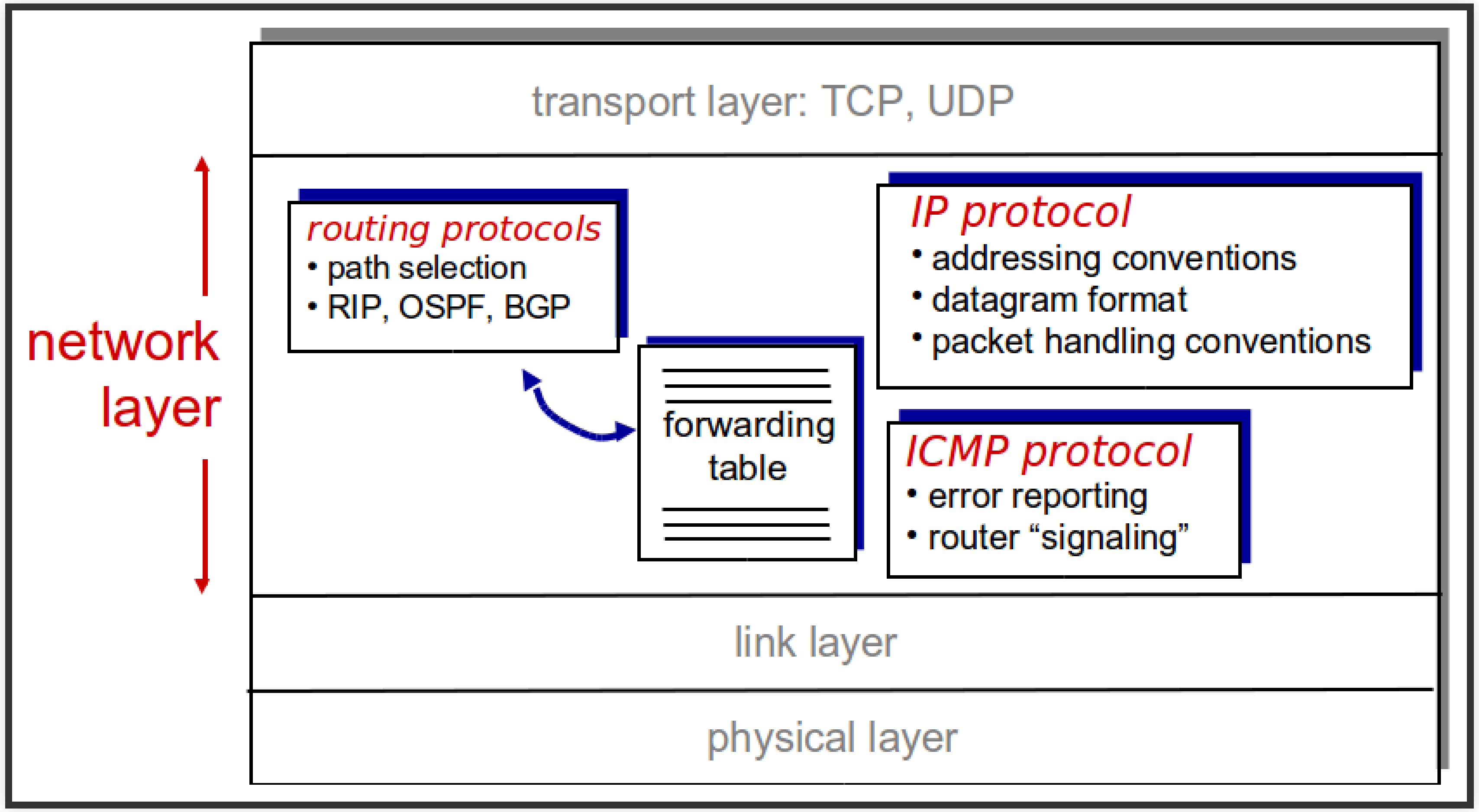
SCHEDULING MECHANISMS: FIFO

- **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

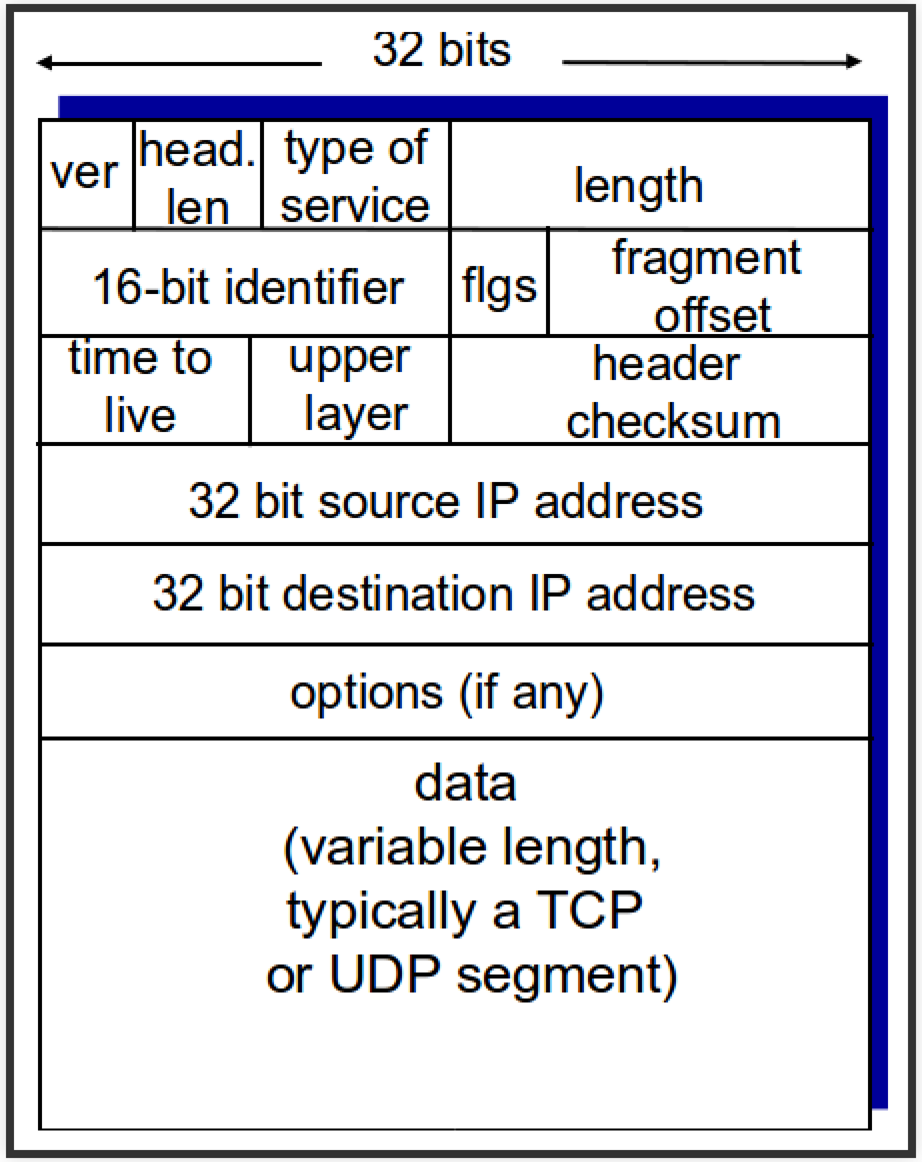
IP: INTERNET PROTOCOL

THE INTERNET NETWORK LAYER

host, router network layer functions:



IP DATAGRAM FORMAT



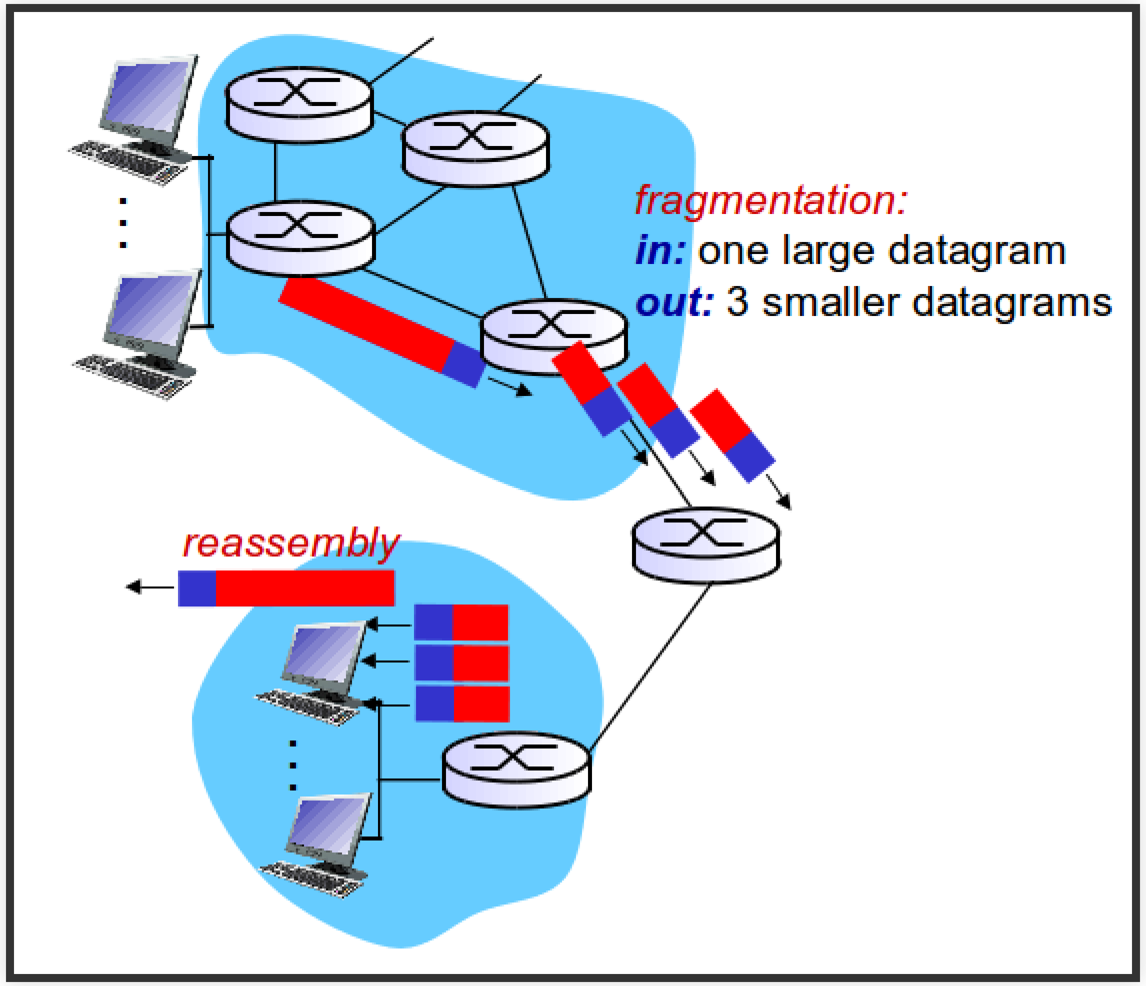
IP DATAGRAM FORMAT

❗ **How much overhead?** 20 bytes of TCP + 20 bytes of IP = 40 bytes + app layer overhead

IP FRAGMENTATION, REASSEMBLY

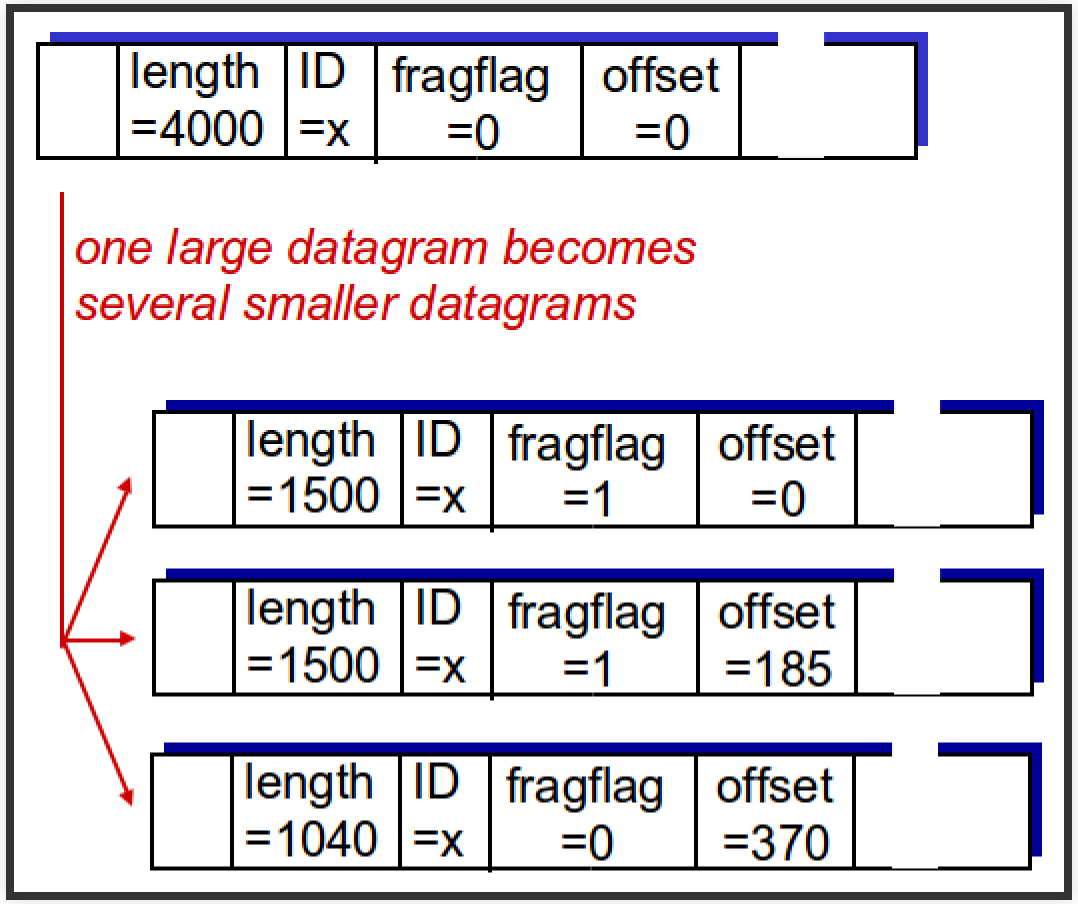
- network links have MTU (max.transfer size) - largest possible link-level frame
  - different link types, different MTUs
- large IP datagram divided (“fragmented”) within net
  - one datagram becomes several datagrams
  - “reassembled” only at final destination
  - IP header bits used to identify, order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

IP FRAGMENTATION, REASSEMBLY



IP FRAGMENTATION, REASSEMBLY

❗ **Example** 4000 byte datagram, MTU = 1500 bytes



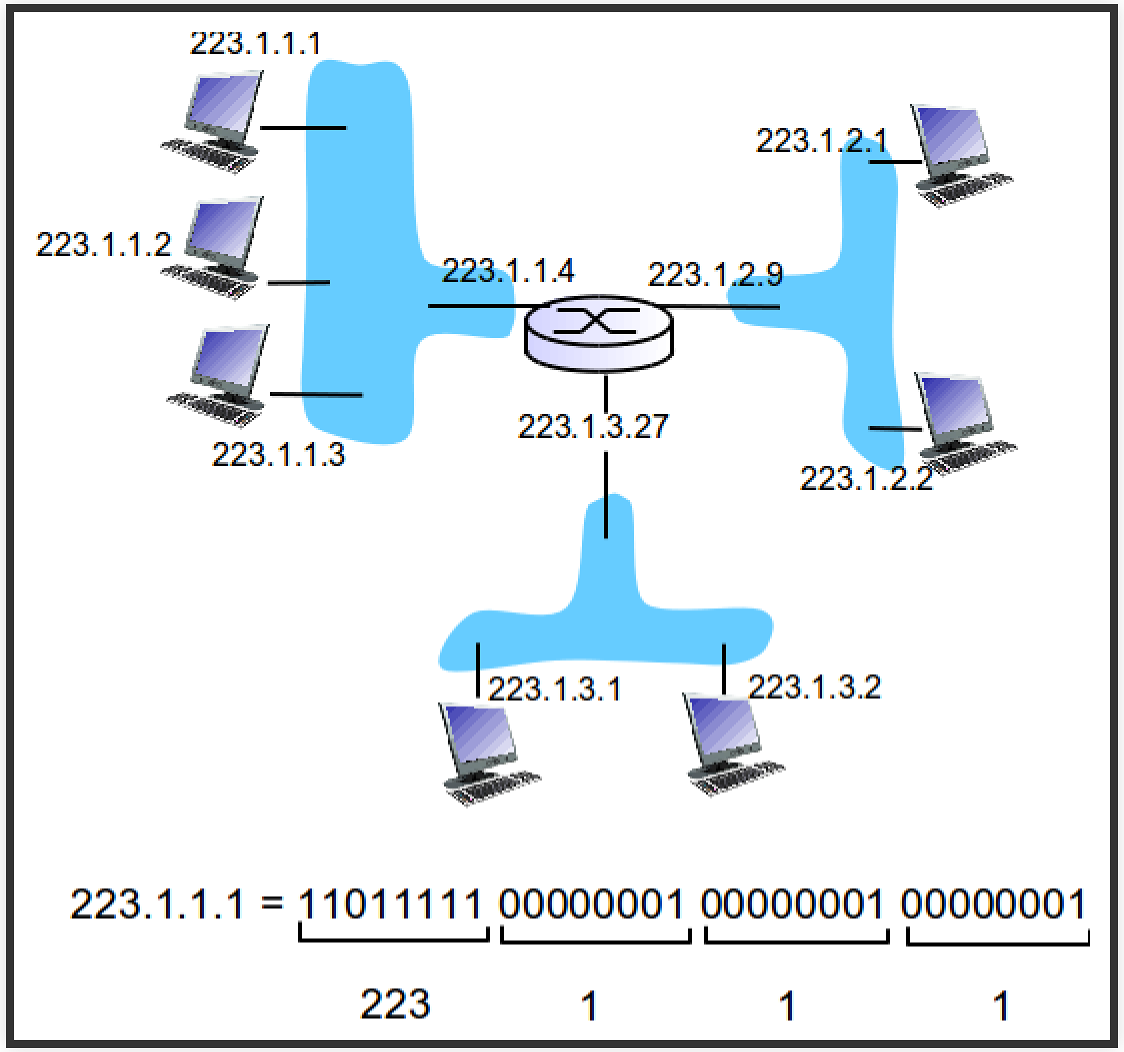
length =4000 | ID =x | fragflag =0 | offset =0
one large datagram becomes several smaller datagrams
length =1500 | ID =x | fragflag =1 | offset =0
length =1500 | ID =x | fragflag =1 | offset =185
length =1040 | ID =x | fragflag =0 | offset =370

IP ADDRESSING: INTRODUCTION

- **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

IP ADDRESSING: INTRODUCTION

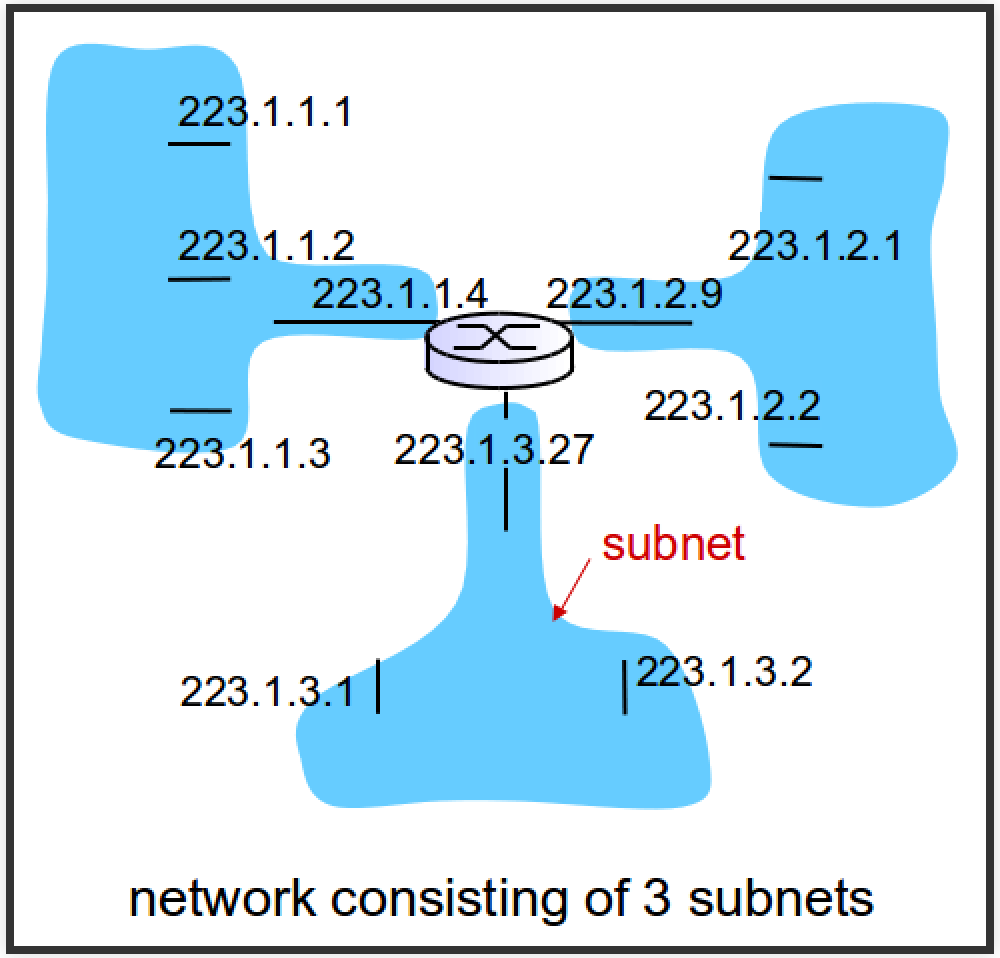


$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$

SUBNETS

- **IP address:**
  - subnet part - high order bits
  - host part - low order bits
- **what is a subnet?**
  - device interfaces with same subnet part of IP address
  - can physically reach each other **without intervening router**

SUBNETS



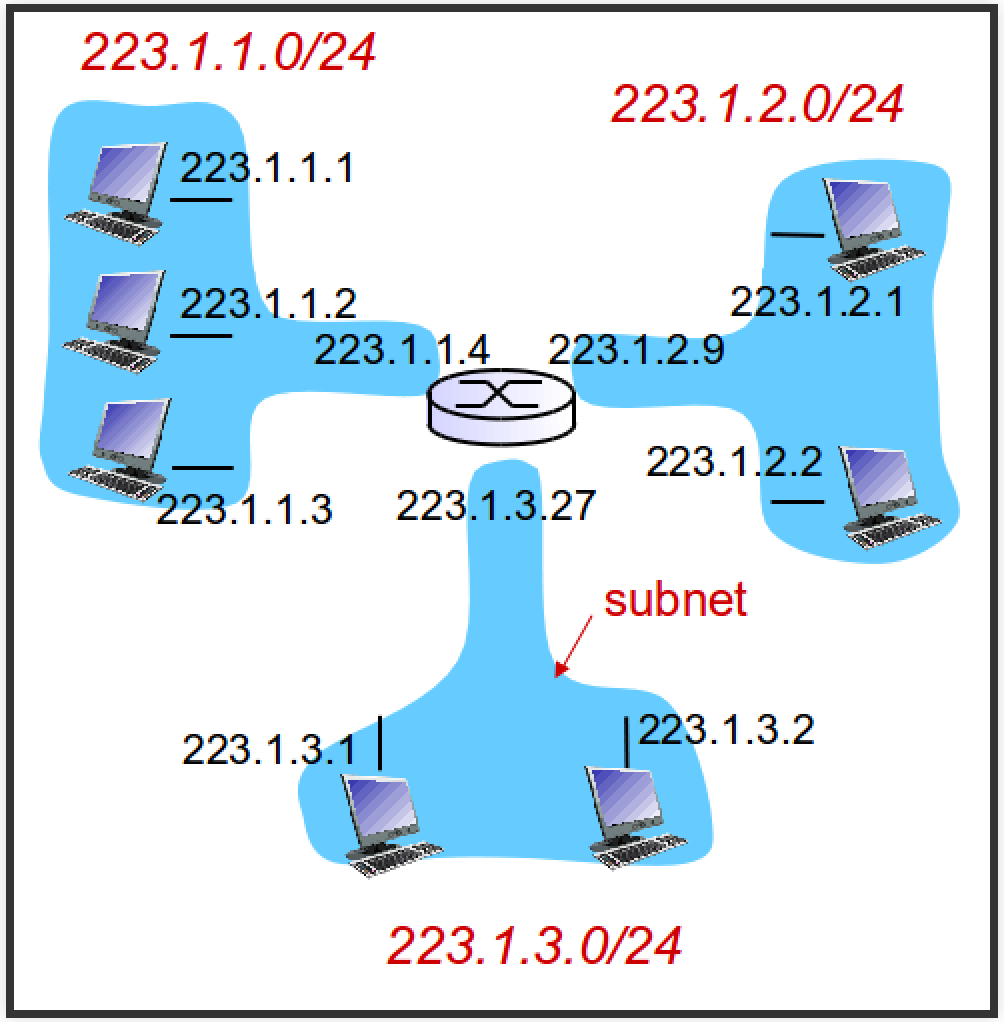
network consisting of 3 subnets

SUBNETS

Recipe

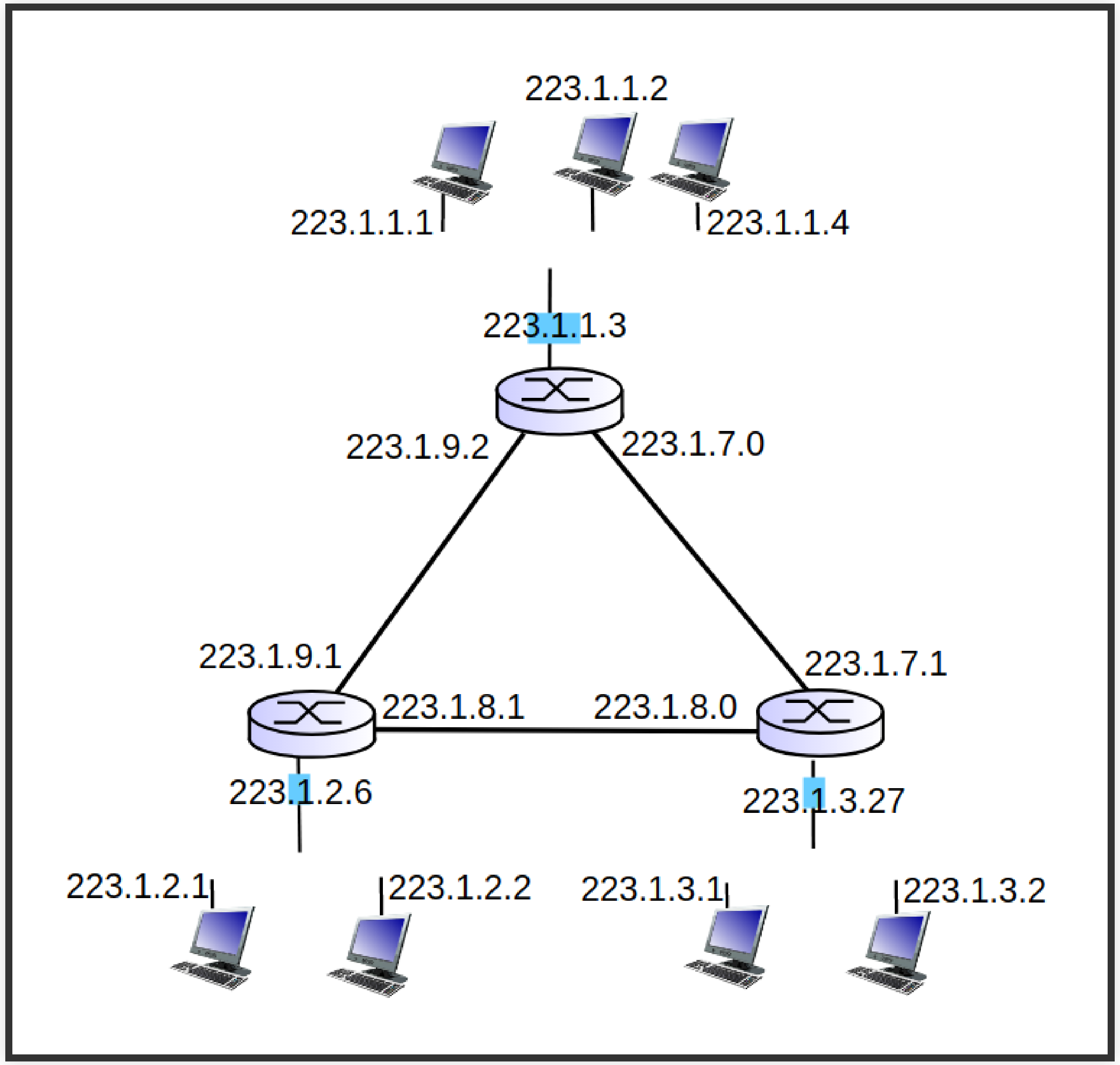
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a *subnet*

SUBNETS


223.1.1.0/24
223.1.2.0/24
223.1.1.1
223.1.1.2
223.1.1.4
223.1.2.9
223.1.2.1
223.1.2.2
223.1.1.3
223.1.3.27
subnet
223.1.3.1
223.1.3.2
223.1.3.0/24

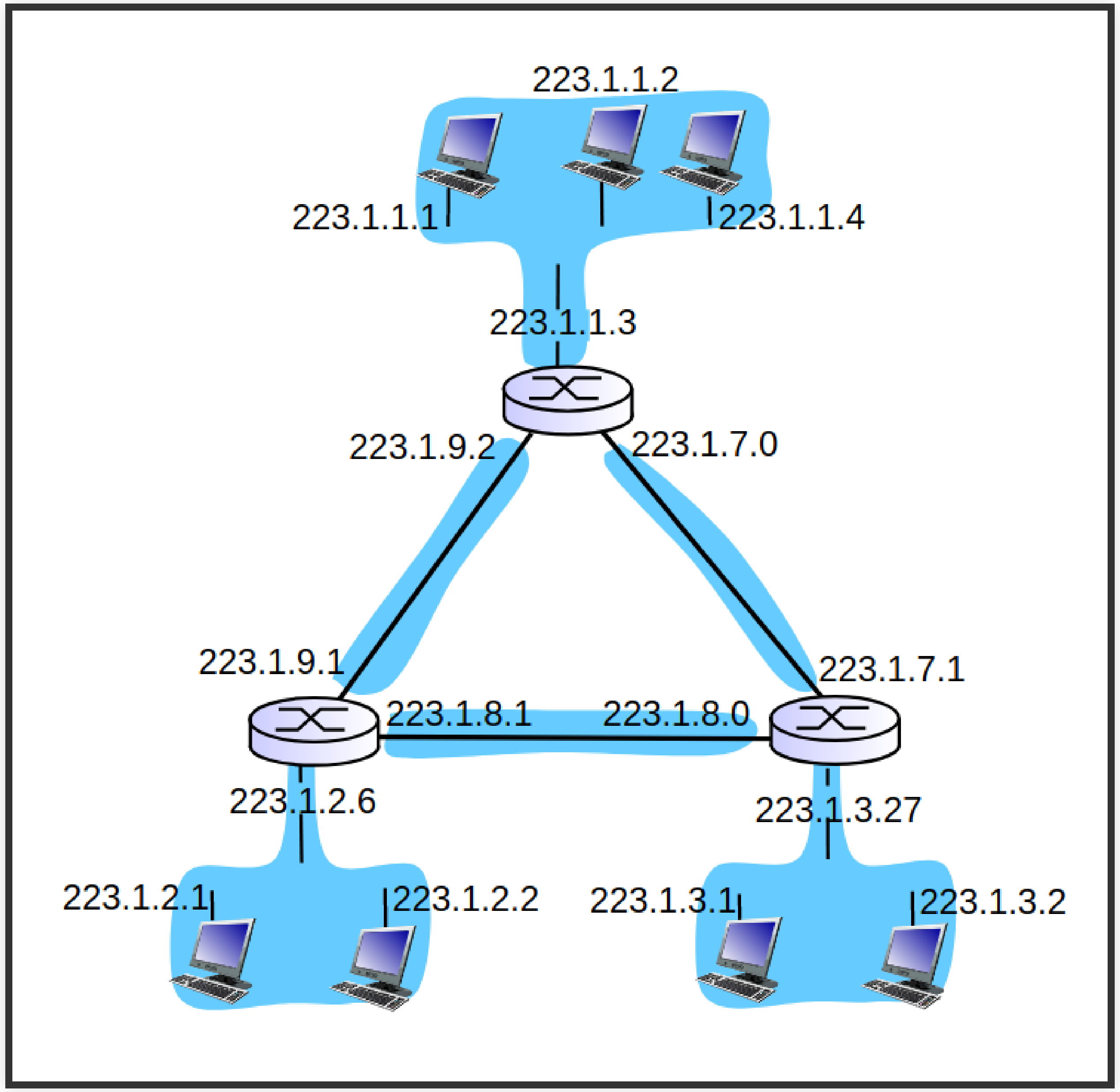
SUBNETS

How many?



SUBNETS

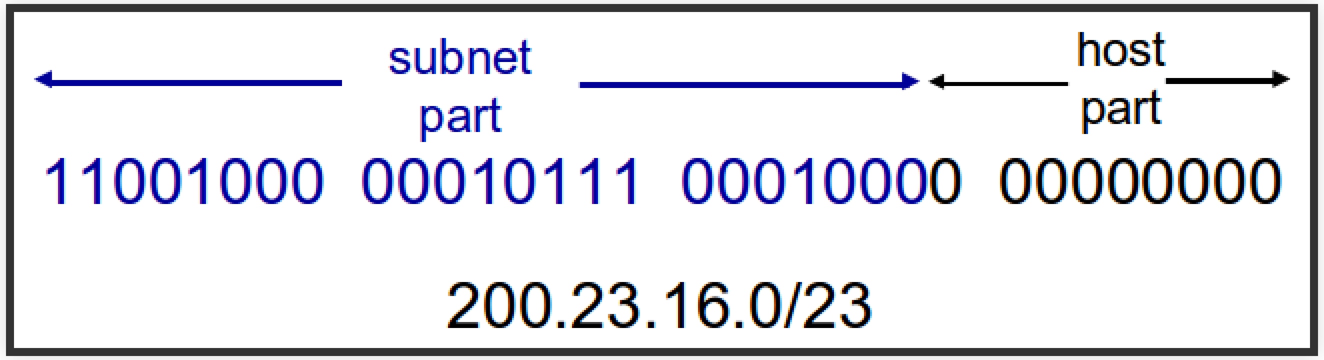
How many?


223.1.1.2
223.1.1.1
223.1.1.4
223.1.1.3
223.1.9.2
223.1.7.0
223.1.9.1
223.1.7.1
223.1.8.1
223.1.8.0
223.1.2.6
223.1.3.27
223.1.2.1
223.1.2.2
223.1.3.1
223.1.3.2

IP ADDRESSING: CIDR

Routing

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



CIDR Classless Interdomain Routing

IP ADDRESSES: HOW TO GET ONE?

❗ Q: How does a host get IP address?

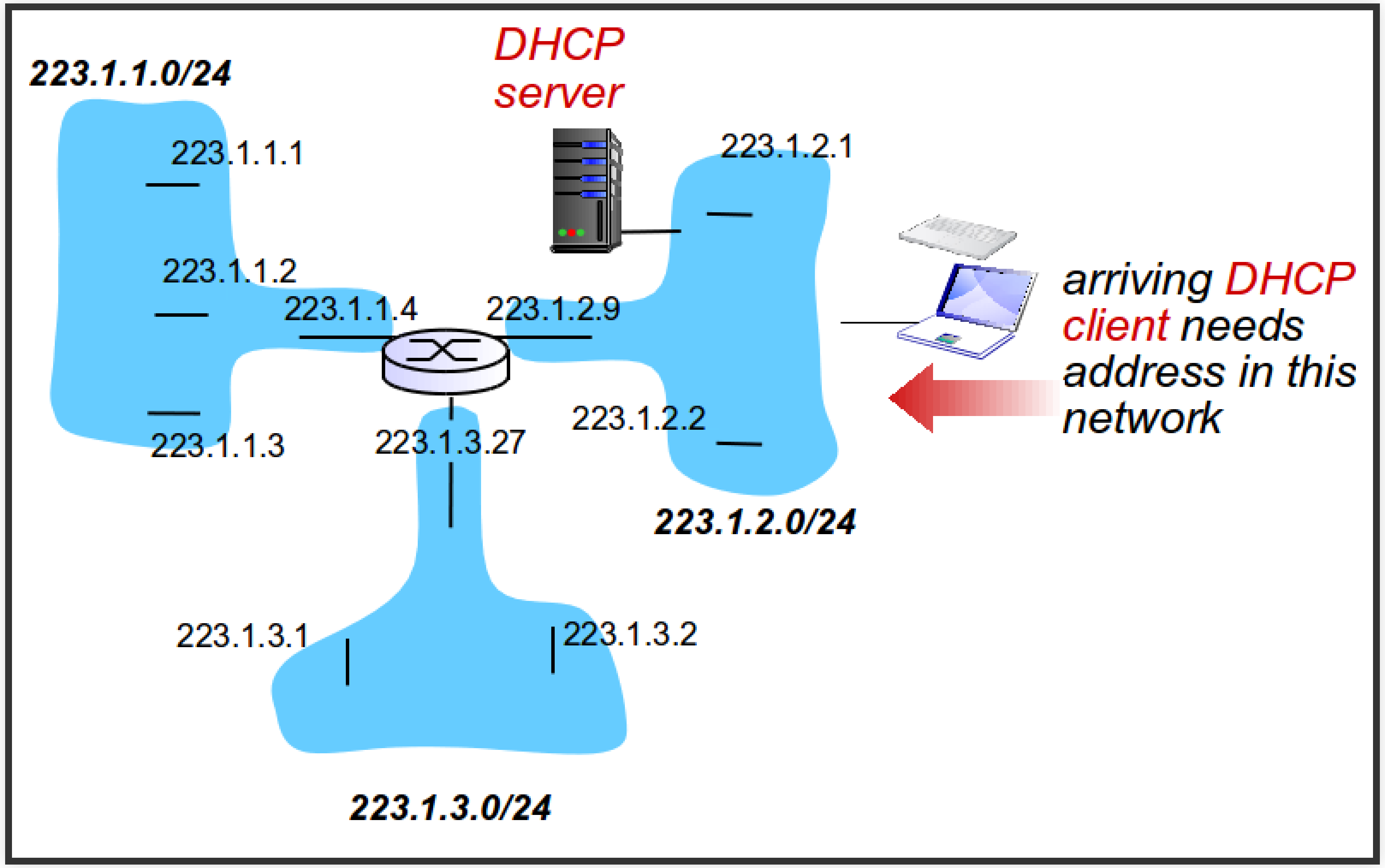
- hard-coded by system admin in a file
  - Windows: control-panel → network → configuration → tcp/ip → properties
  - UNIX: /etc/rc.config
- **DHCP: D**ynamic **H**ost **C**onfiguration **P**rotocol: dynamically get address from *AS* server
  - "plug-and-play"

DHCP: DYNAMIC HOST CONFIGURATION PROTOCOL

❗ **Goal:** 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

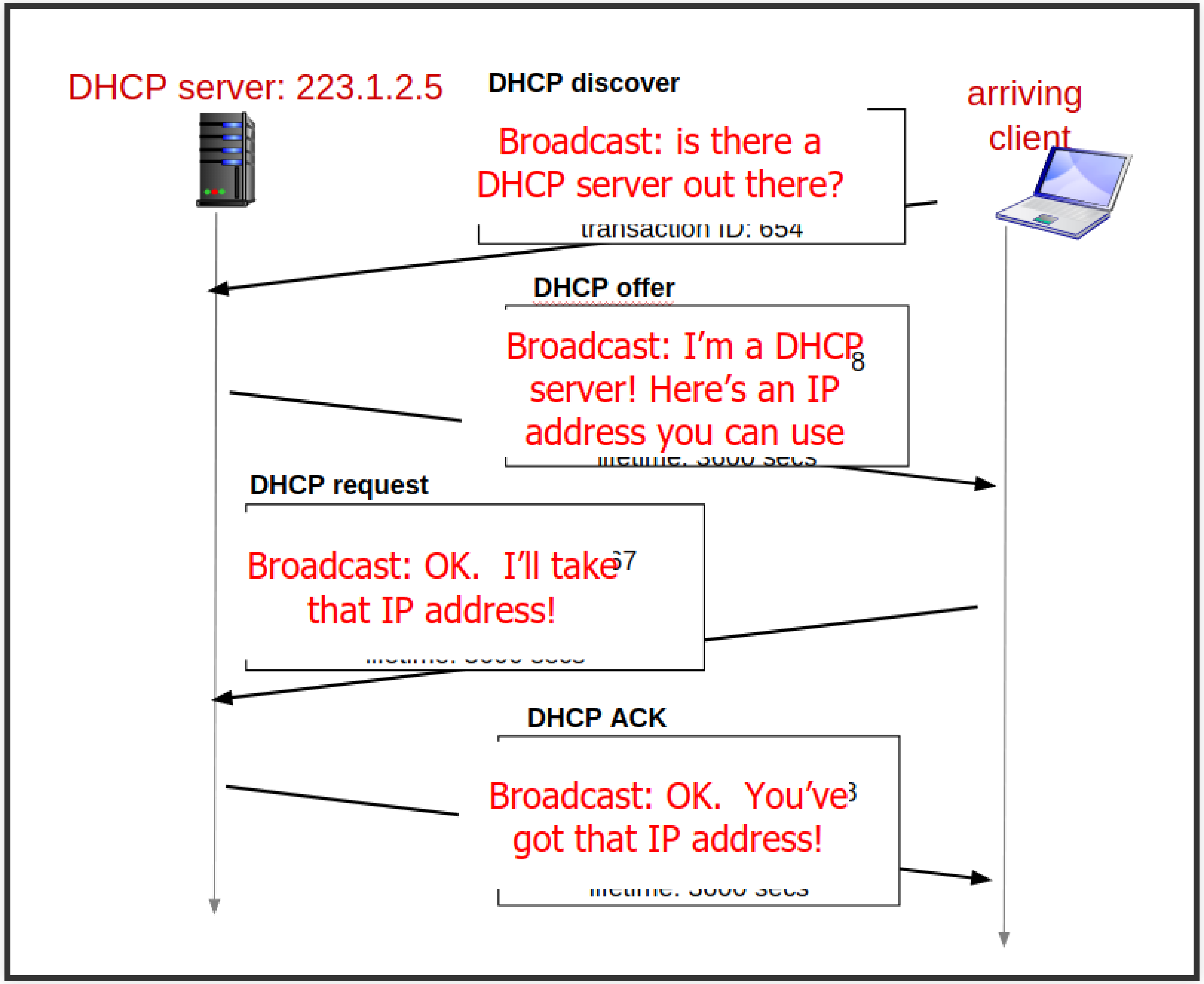


DHCP

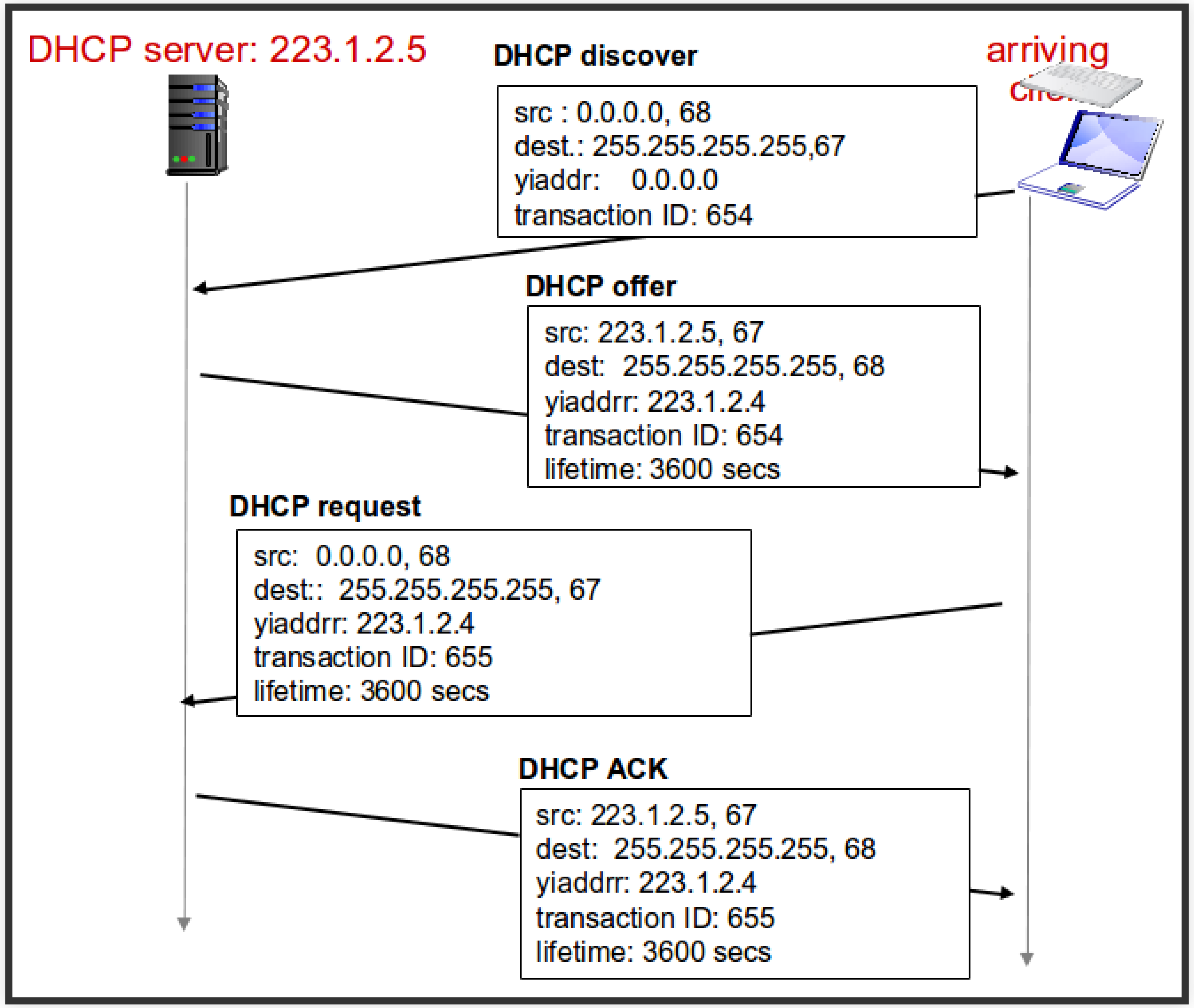
DHCP overview:

- host broadcasts “DHCP discover” msg
- DHCP server responds with “DHCP offer” msg
- host requests IP address: “DHCP request” msg
- DHCP server sends address: “DHCP ack” msg

DHCP CLIENT-SERVER SCENARIO


DHCP server: 223.1.2.5
arriving client
DHCP discover
Broadcast: is there a DHCP server out there?
transaction ID: 654
DHCP offer
Broadcast: I'm a DHCP server! Here's an IP address you can use
lifetime: 3600 secs
DHCP request
Broadcast: OK. I'll take that IP address!
lifetime: 3600 secs
DHCP ACK
Broadcast: OK. You've got that IP address!
lifetime: 3600 secs

DHCP CLIENT-SERVER SCENARIO

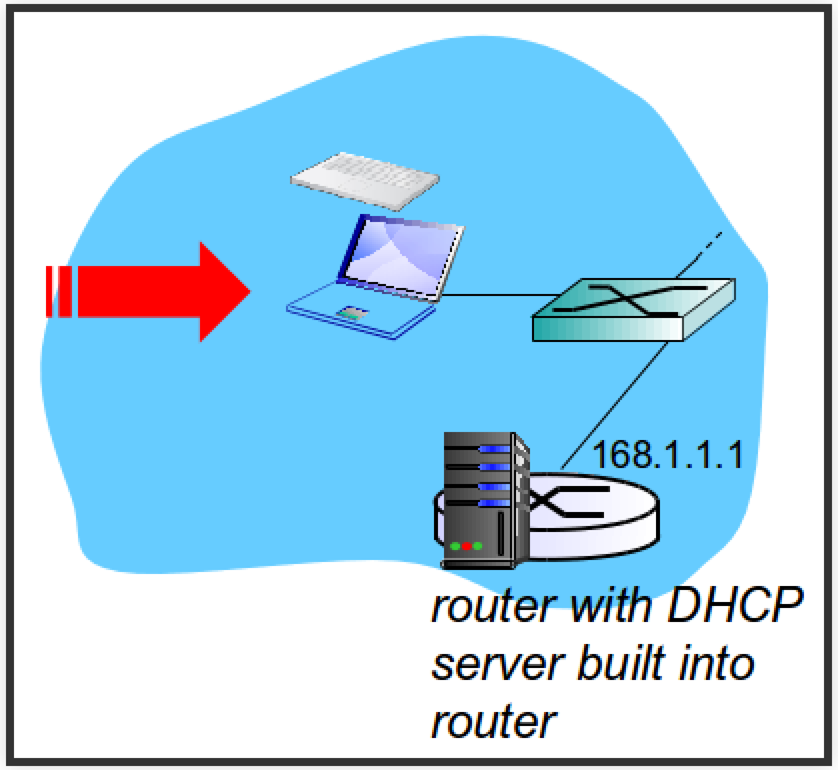


DHCP: MORE THAN IP ADDRESSES

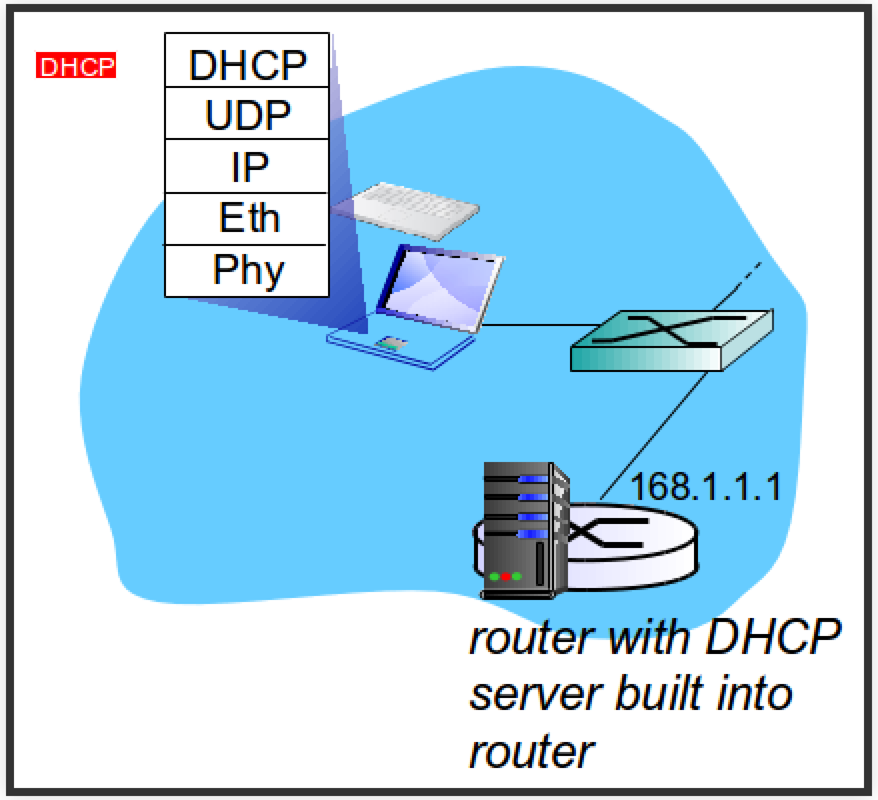
DHCP can return more than just allocated IP address on subnet:

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

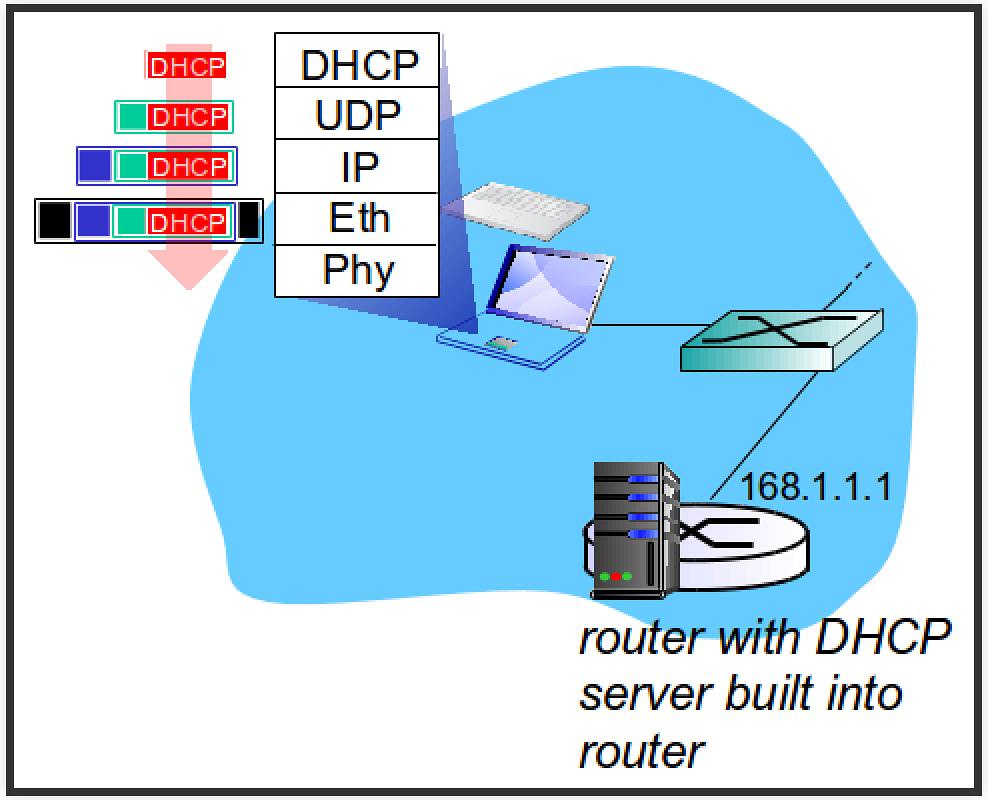
DHCP: EXAMPLE



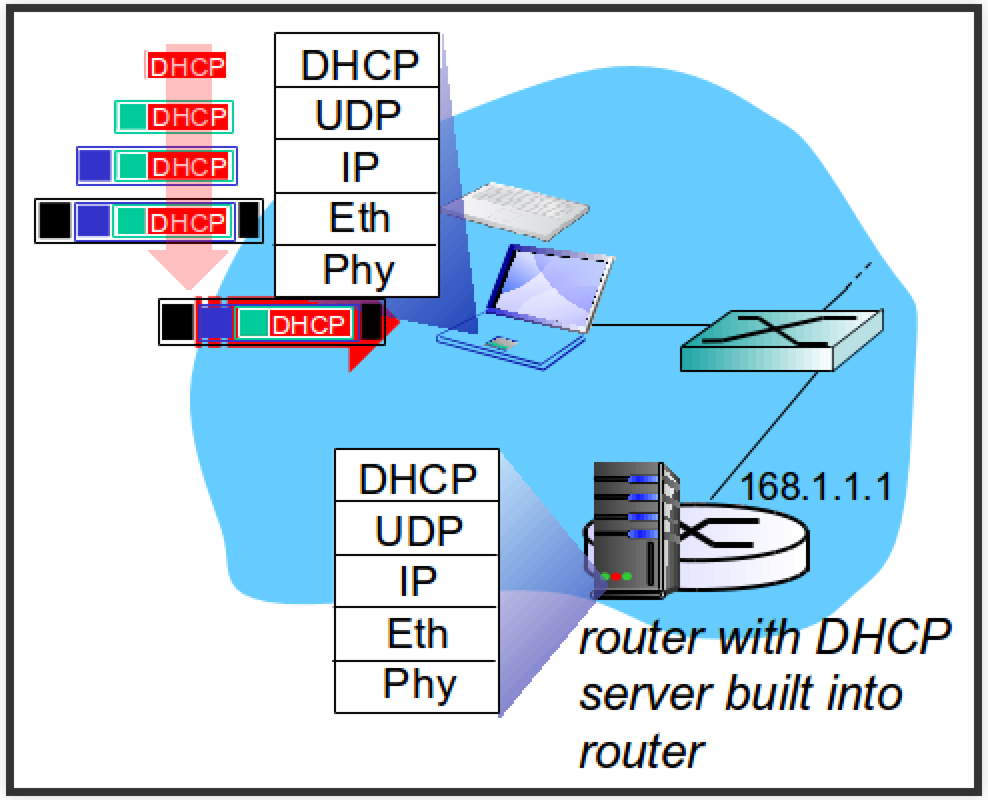
DHCP: EXAMPLE



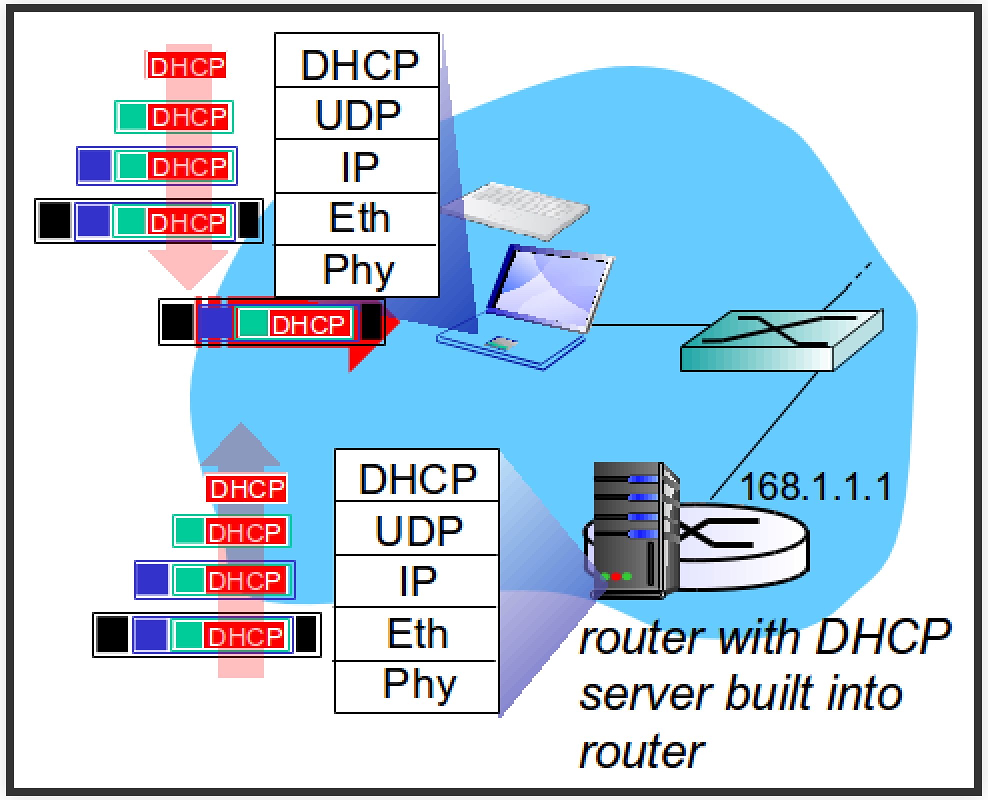
DHCP: EXAMPLE



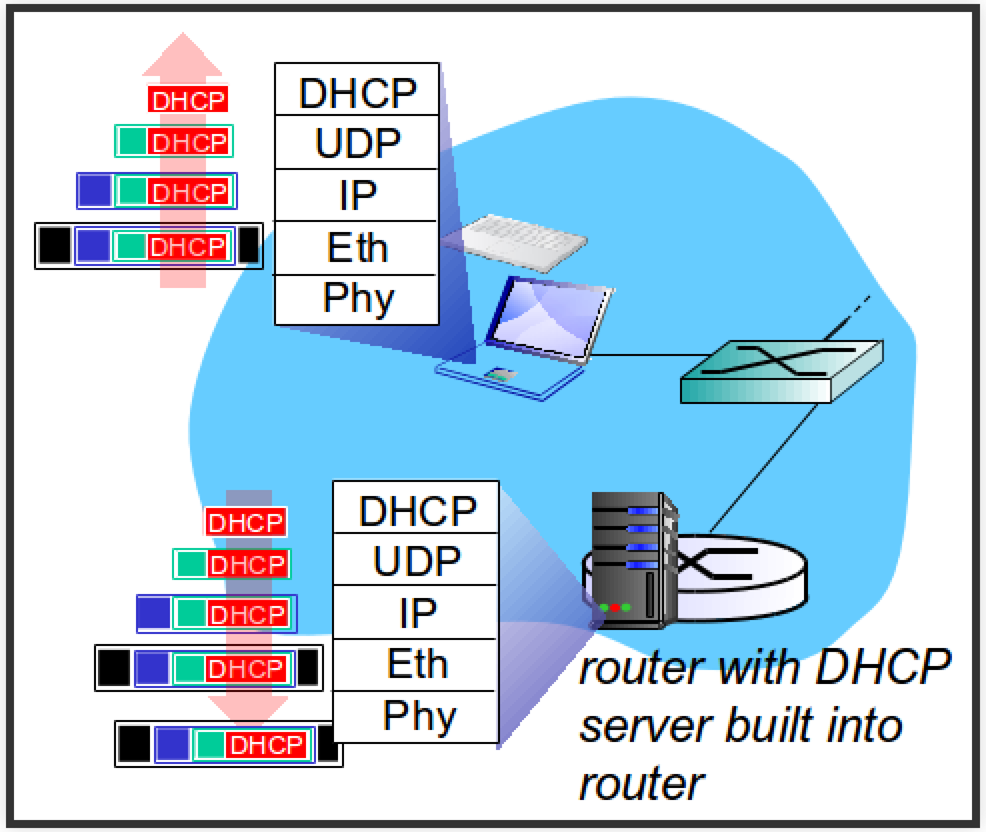
DHCP: EXAMPLE



DHCP: EXAMPLE



DHCP: EXAMPLE



IP ADDRESSES: HOW TO GET ONE?

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

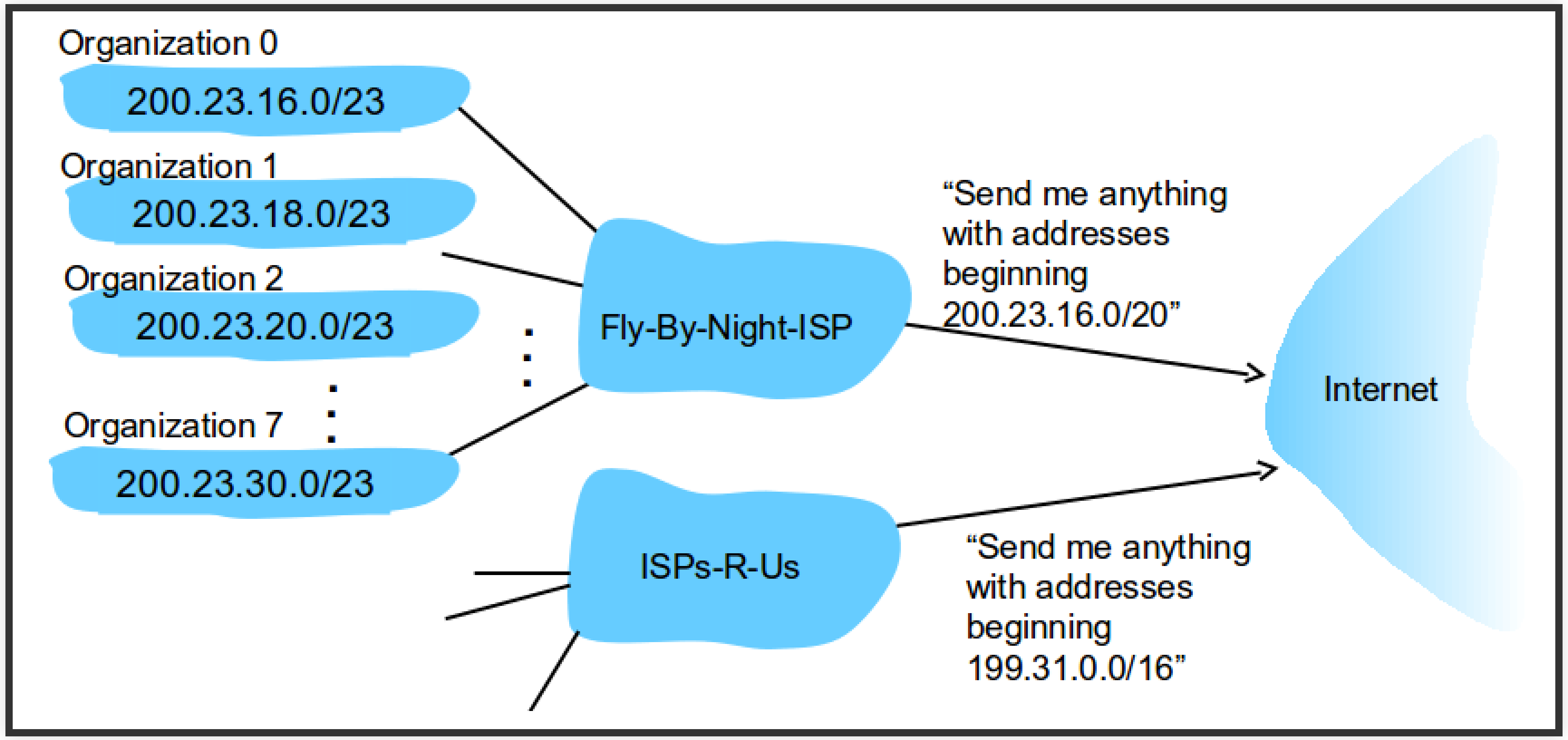
A: gets allocated portion of its provider ISP’s address space

```
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
```

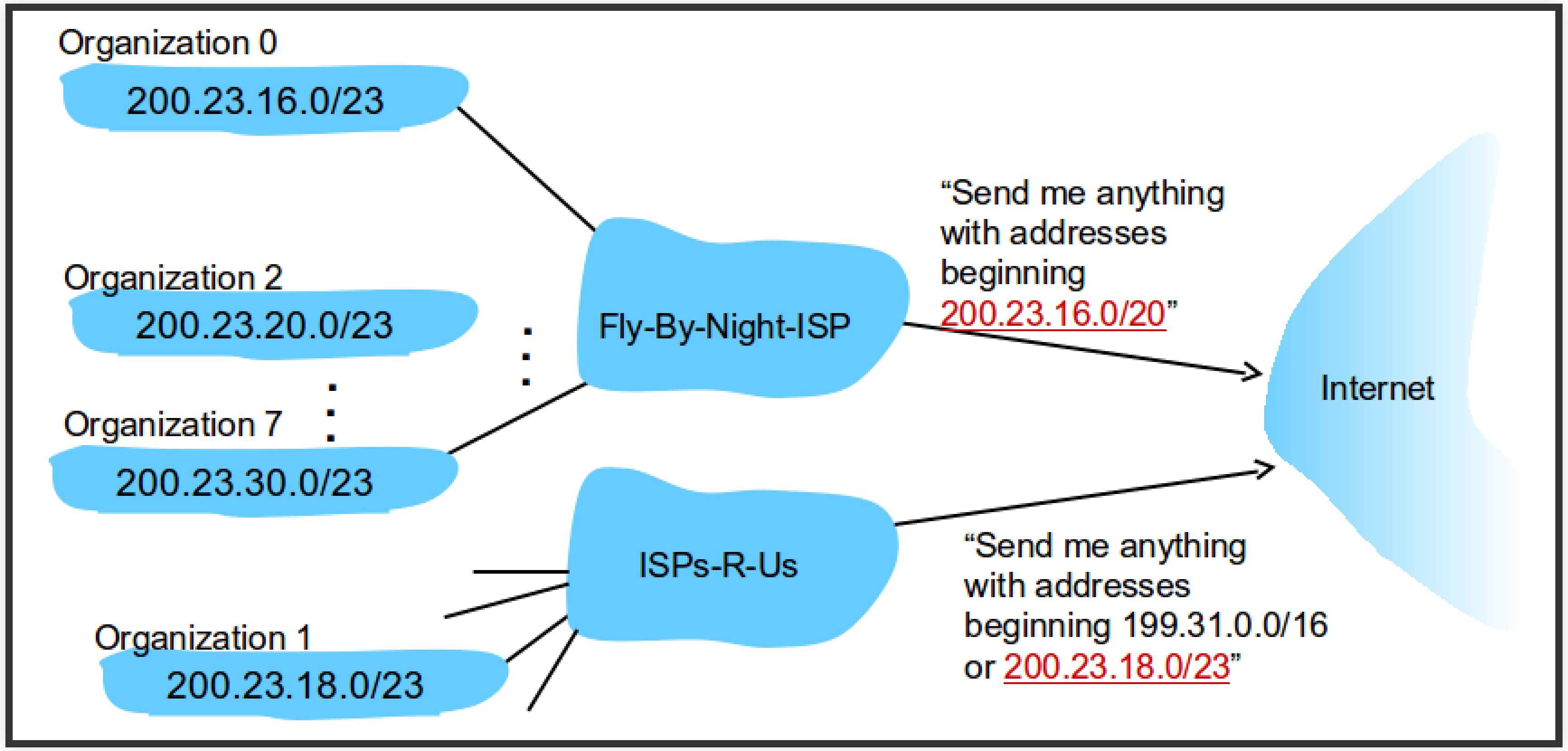
HIERARCHICAL ADDRESSING: ROUTE AGGREGATION

hierarchical addressing allows efficient advertisement of routing information:



HIERARCHICAL ADDRESSING: MORE SPECIFIC ROUTES

ISPs-R-Us has a more specific route to Organization 1



RESERVED IP ADRESSES

For private networks - RFC-1918

```
10.0.0.0        -   10.255.255.255  (10/8 prefix)
172.16.0.0      -   172.31.255.255  (172.16/12 prefix)
192.168.0.0     -   192.168.255.255 (192.168/16 prefix)
```

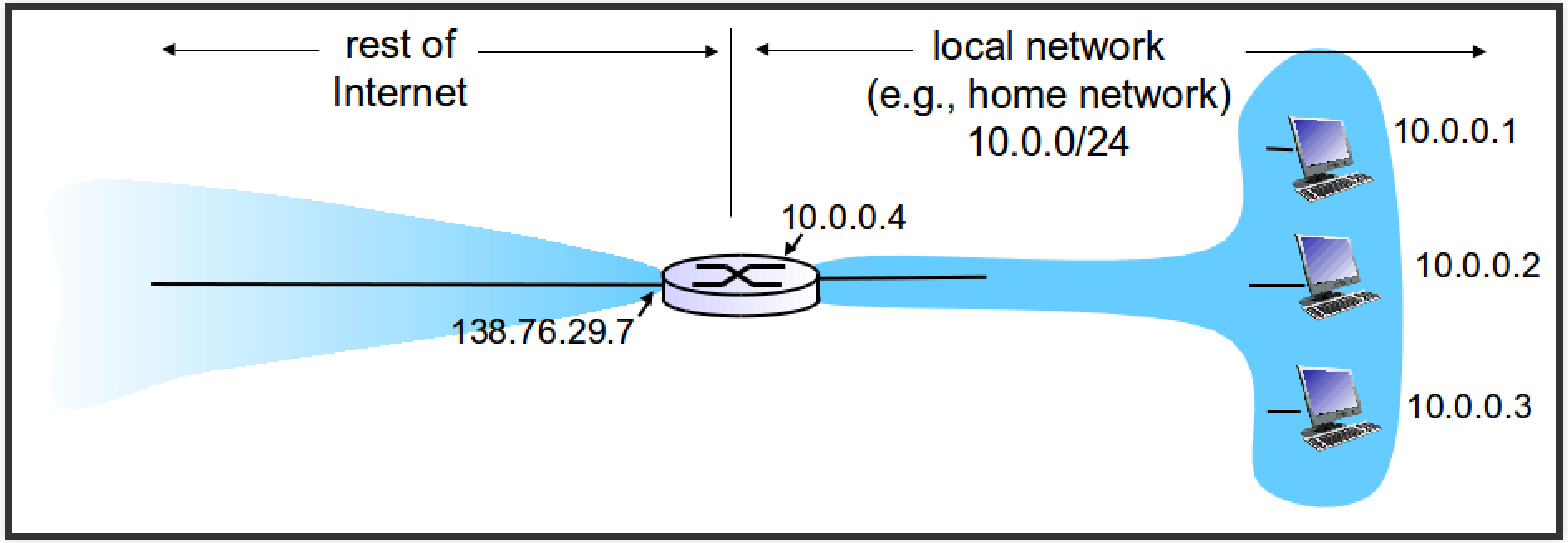
IP ADDRESSING: THE LAST WORD...

❗ **Q:** how does an ISP get block of addresses?

A: ICANN: Internet **C**orporation for **A**ssigned **N**ames and **N**umbers

- http://www.icann.org/
- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

NAT: NETWORK ADDRESS TRANSLATION



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:

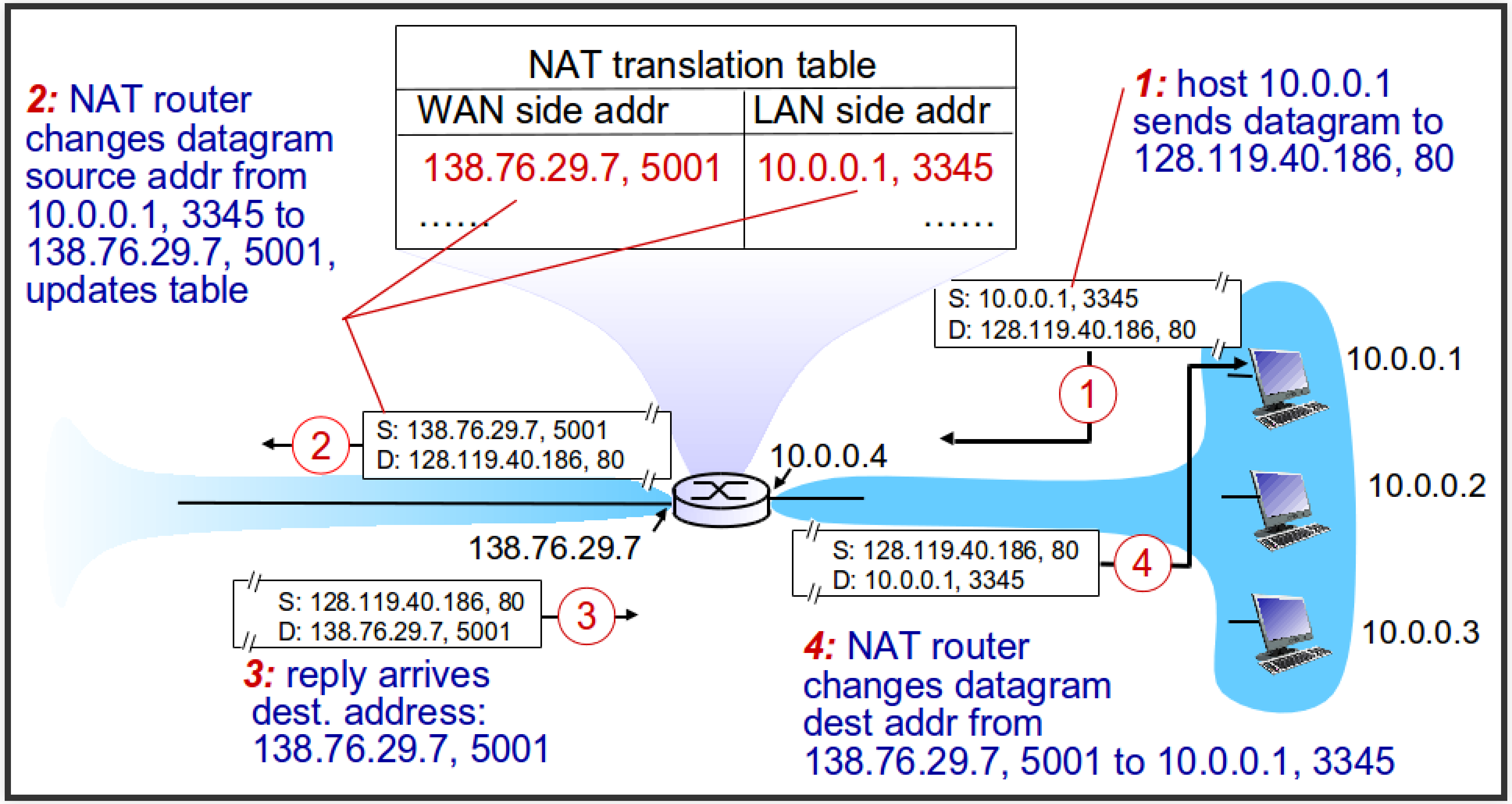
- range of addresses not needed from ISP: just one IP address for all devices
- can change addresses of devices in local network without notifying outside world
- can change ISP without changing addresses of devices in local network
- devices inside local net not explicitly addressable, visible by outside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



NAT translation table

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

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table
1: host 10.0.0.1 sends datagram to 128.119.40.186, 80
S: 10.0.0.1, 3345
D: 128.119.40.186, 80
1
10.0.0.1
10.0.0.2
10.0.0.3
10.0.0.4
2
S: 138.76.29.7, 5001
D: 128.119.40.186, 80
138.76.29.7
S: 128.119.40.186, 80
D: 10.0.0.1, 3345
4
S: 128.119.40.186, 80
D: 138.76.29.7, 5001
3
3: reply arrives dest. address: 138.76.29.7, 5001
4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

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
  - violates end-to-end argument
  - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - address shortage should instead be solved by IPv6

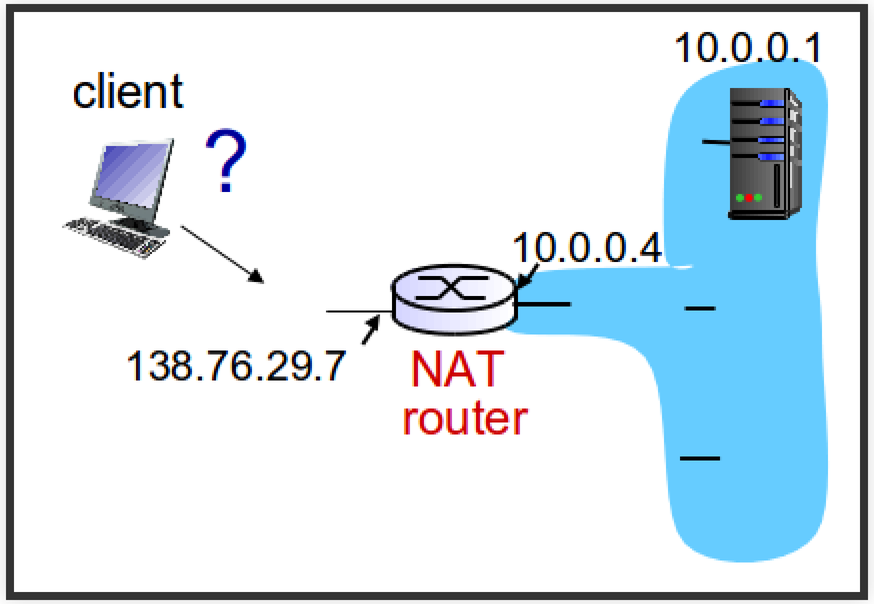
NAT TRAVERSAL PROBLEM

- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can’t use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7

SOLUTION 1

Statically configure NAT to forward incoming connection requests at given port to server

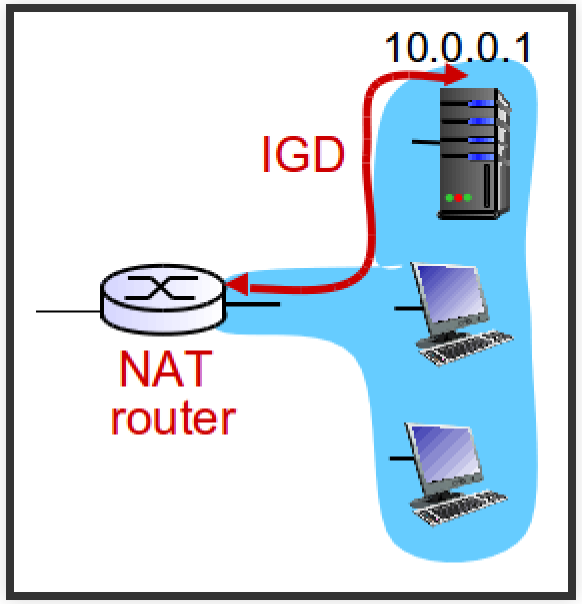
- e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000



SOLUTION 2

Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:

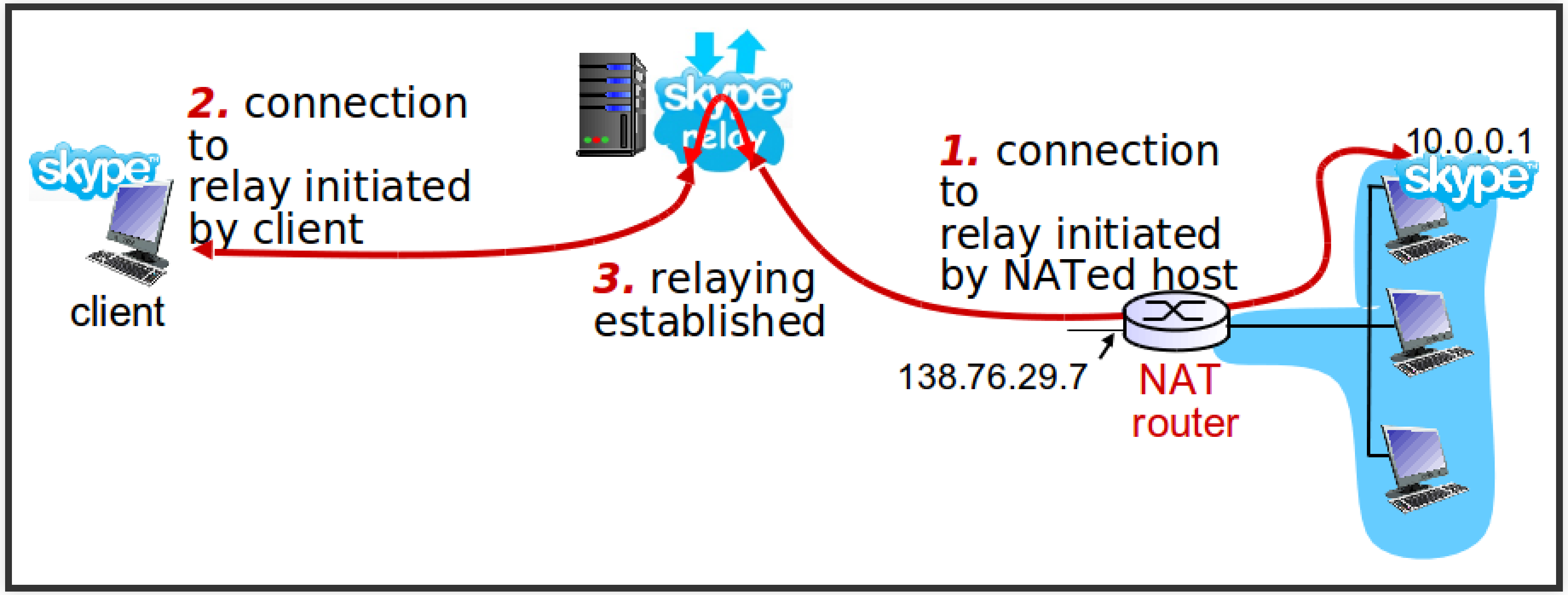
- learn public IP address (138.76.29.7)
- add/remove port mappings (with lease times) i.e., automate static NAT port map configuration



SOLUTION 3

Relaying (used in Skype)

- NATed client establishes connection to relay
- external client connects to relay
- relay bridges packets between to connections


2. connection to relay initiated by client
client
skype
skype relay
3. relaying established
1. connection to relay initiated by NATed host
10.0.0.1
138.76.29.7
NAT router

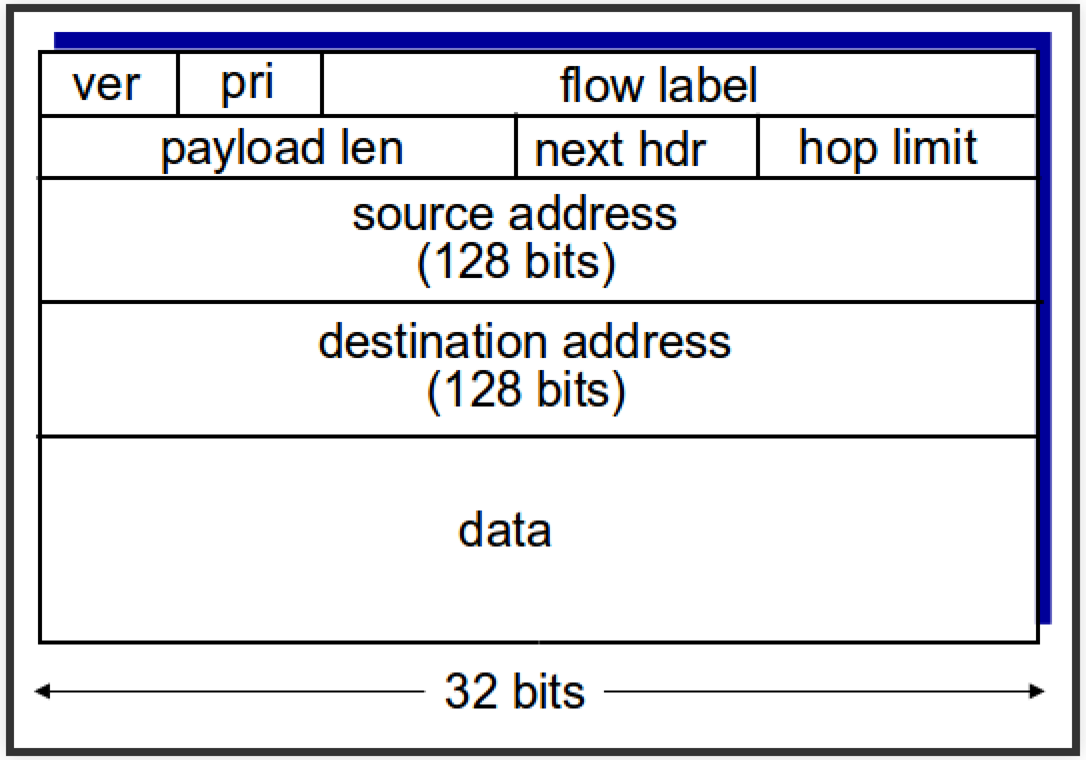
IPV6

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

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

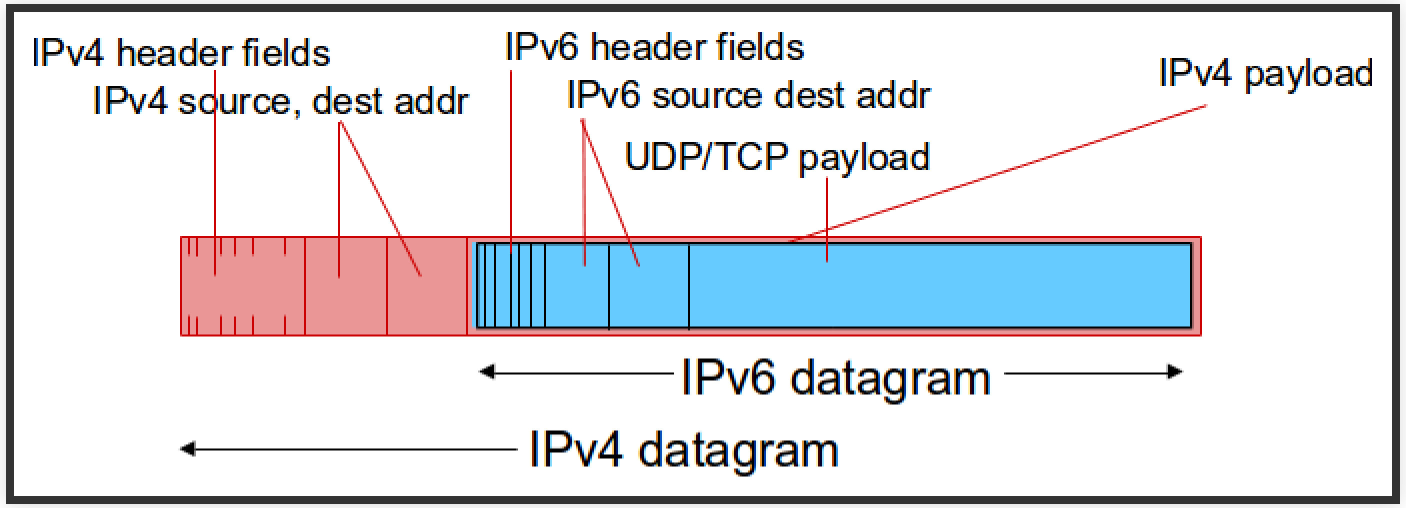


OTHER CHANGES FROM IPV4

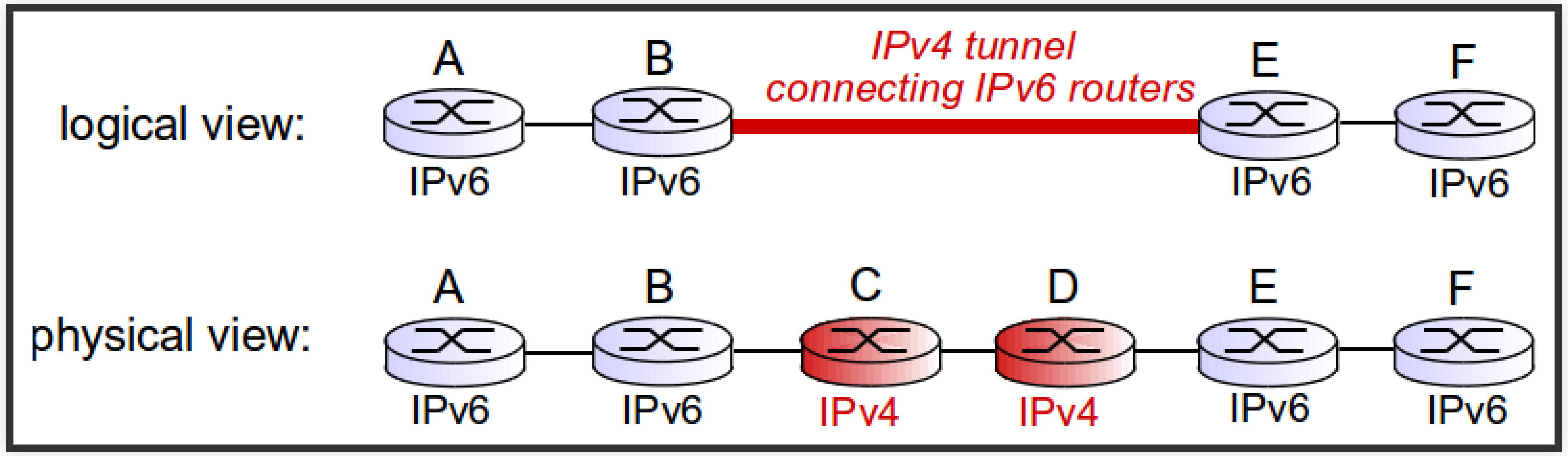
- **checksum:** removed entirely to reduce processing time at each hop
- **options:** allowed, but outside of header, indicated by “Next Header” field
- **ICMPv6:** new version of ICMP
    - additional message types, e.g. “Packet Too Big”
    - multicast group management functions

TRANSITION FROM IPV4 TO IPV6

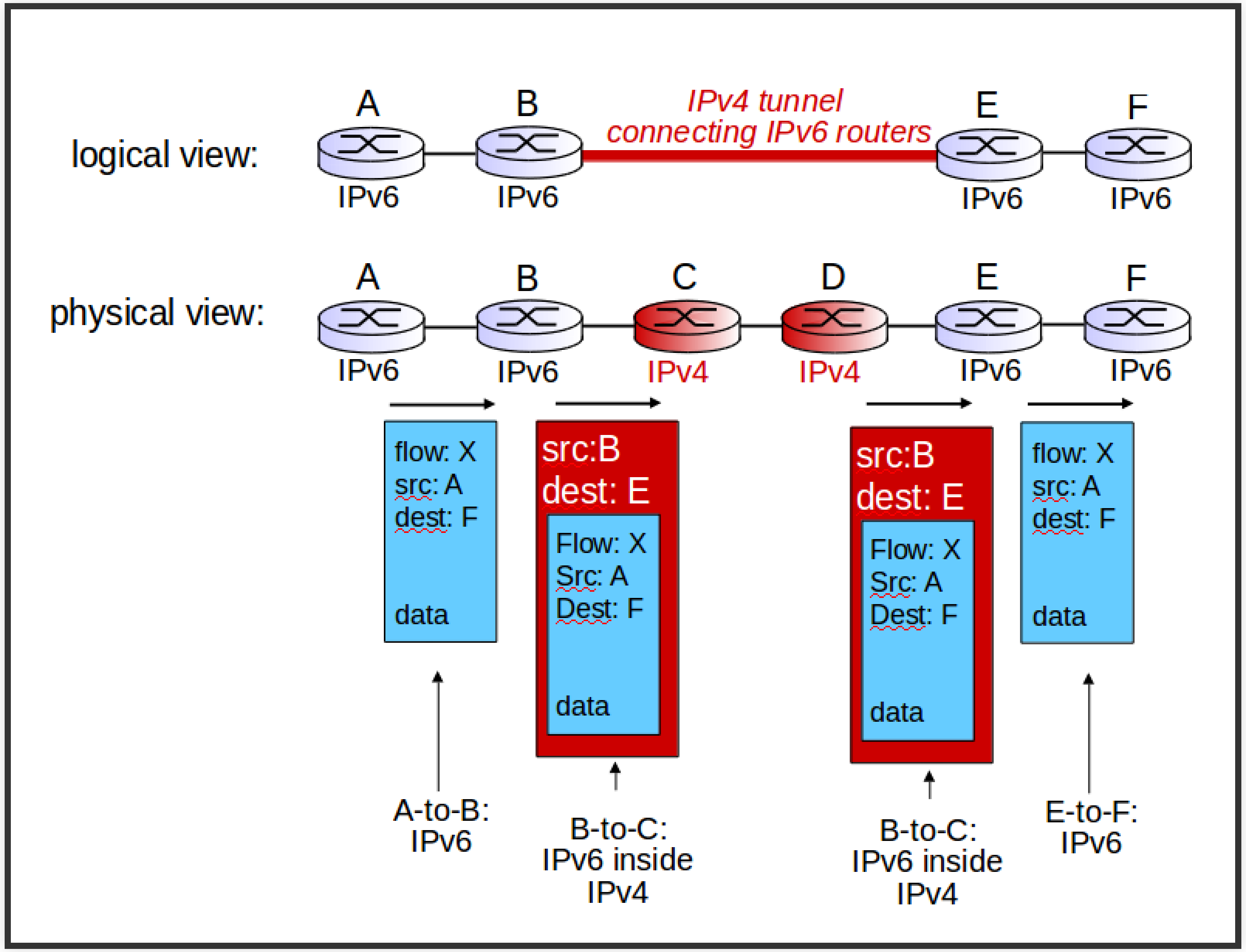
- not all routers can be upgraded simultaneously
  - no “flag days”
  - how will network operate with mixed IPv4 and IPv6 routers?
- **tunneling:** IPv6 datagram carried as payload in IPv4 datagram among IPv4 routers



TUNNELING


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

IPv6 addresses are written in eight groups of four hex digits, with a-f preferred over A-F (RFC 5952). The groups are separated by colons, and have leading 0's removed, eg

```
fedc:13:1654:310:fedc:bc37:61:3210
```

If an address contains a long run of 0's – for example, if the IPv6 address had an embedded IPv4 address – then when writing the address the string "::" should be used to represent however many blocks of 0000 as are needed to create an address of the correct length.

Also, embedded IPv4 addresses may continue to use the "." separator:

```
::ffff:147.126.65.141
```

IPV6 ADDRESS FORMAT

The IPv6 loopback address is ::1 (that is, 127 0-bits followed by a 1-bit).

Network address prefixes may be written with the “/” notation, as in IPv4:

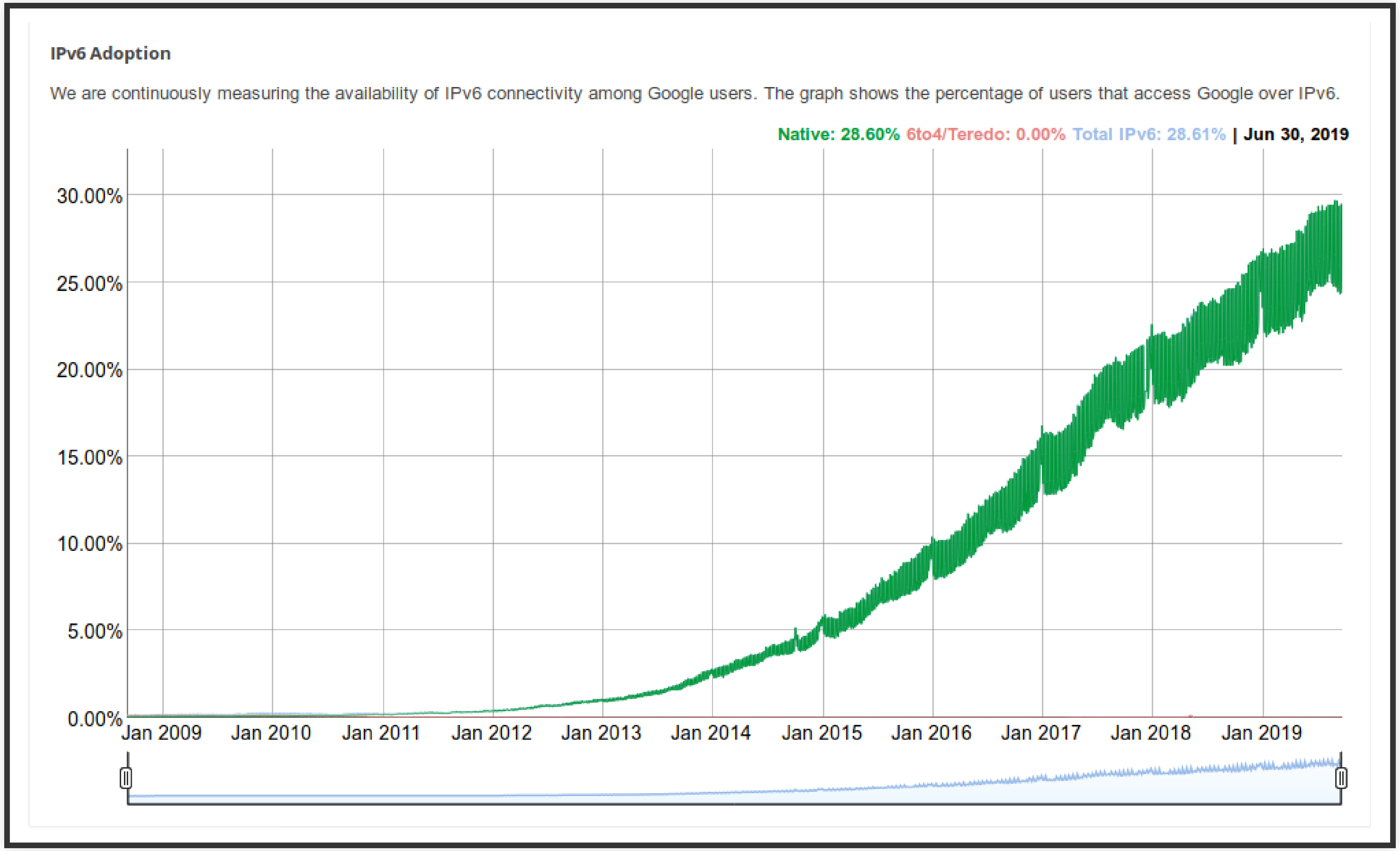
12ab:0:0:cd30::/60

Generally speaking, IPv6 addresses consist of a 64-bit network prefix (perhaps including subnet bits) followed by a 64-bit “interface identifier”.

IPV6 ADOPTION

- **Long** (long!) time for deployment, use
  - 20 years and counting!
  - think of application-level changes in last 20 years: WWW, Facebook, streaming media, Skype, …
  - **Why?**

IPV6 ADOPTION

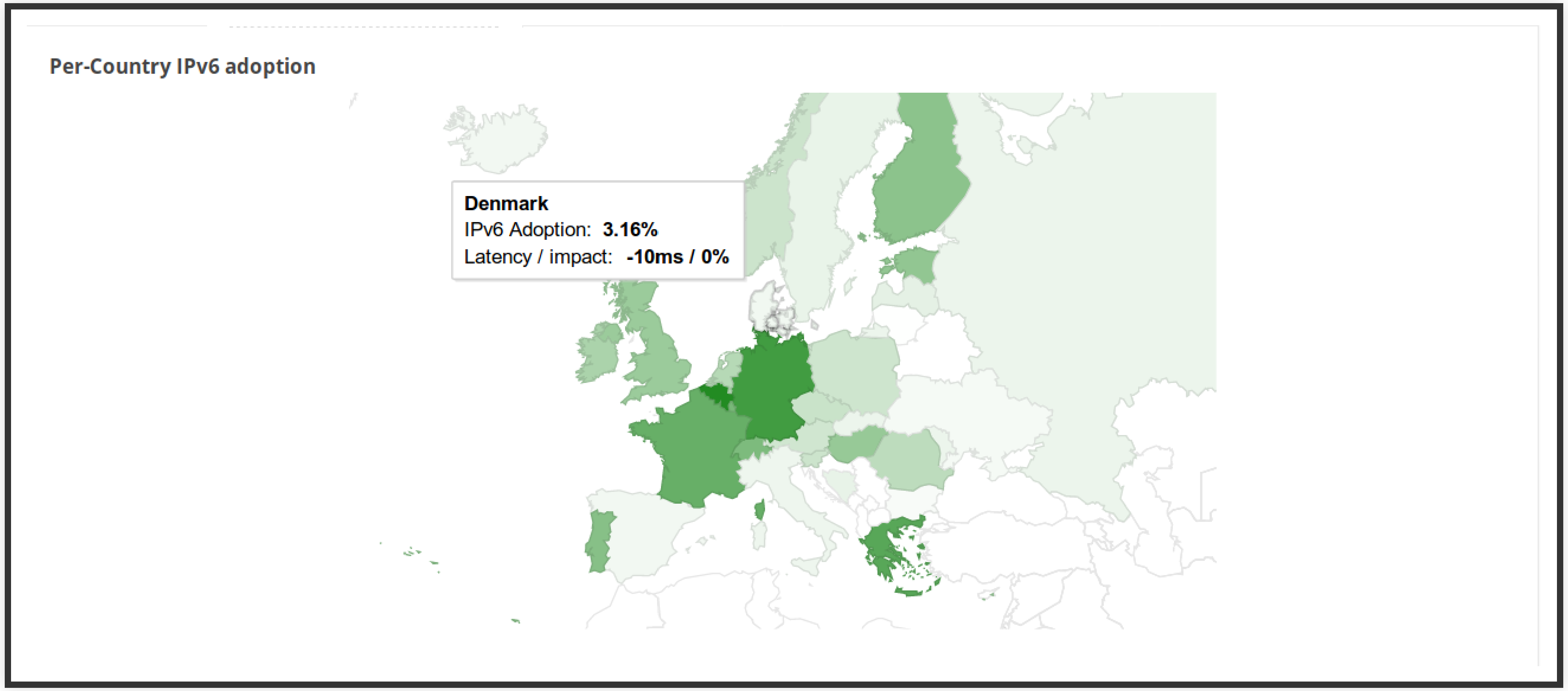


IPv6 Adoption

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

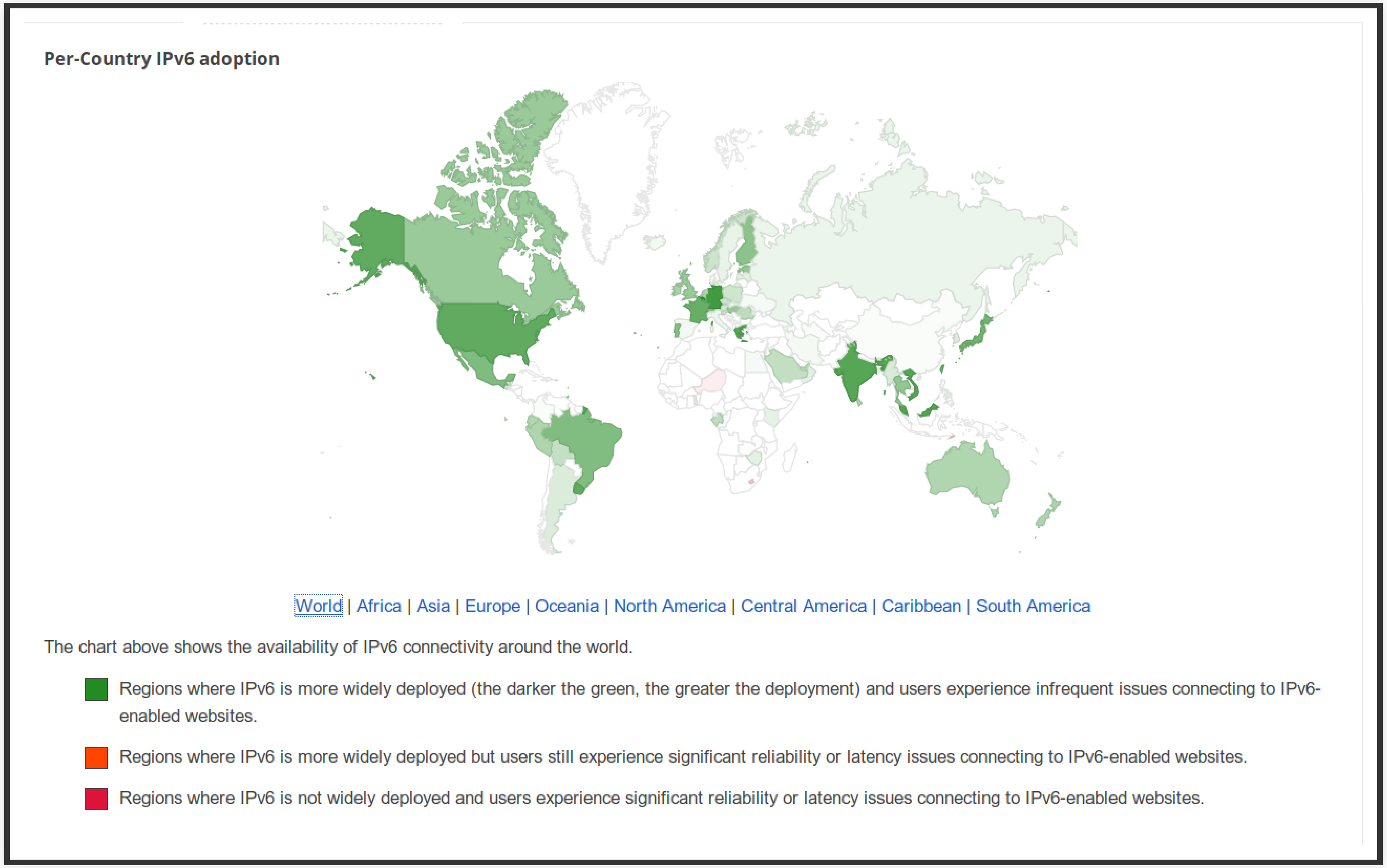
Native: 28.60% 6to4/Teredo: 0.00% Total IPv6: 28.61% | **Jun 30, 2019**

IPV6 ADOPTION

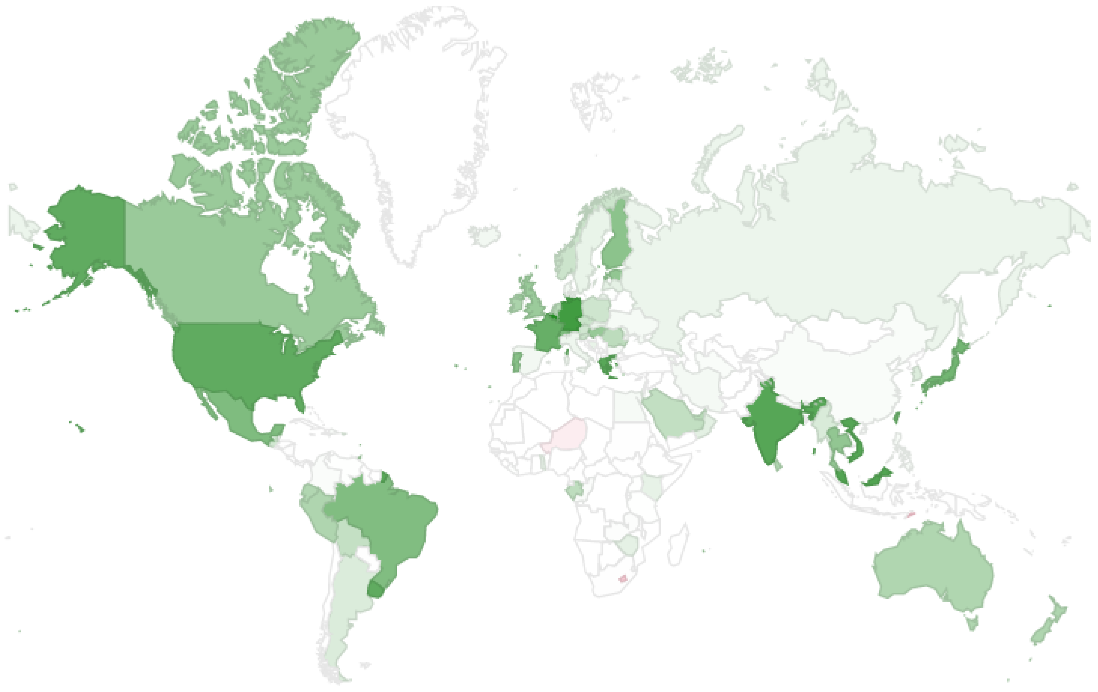

Per-Country IPv6 adoption

Denmark
IPv6 Adoption: **3.16%**
Latency / impact: **-10ms / 0%**

IPV6 ADOPTION



Per-Country IPv6 adoption



World | Africa | Asia | Europe | Oceania | North America | Central America | Caribbean | South America

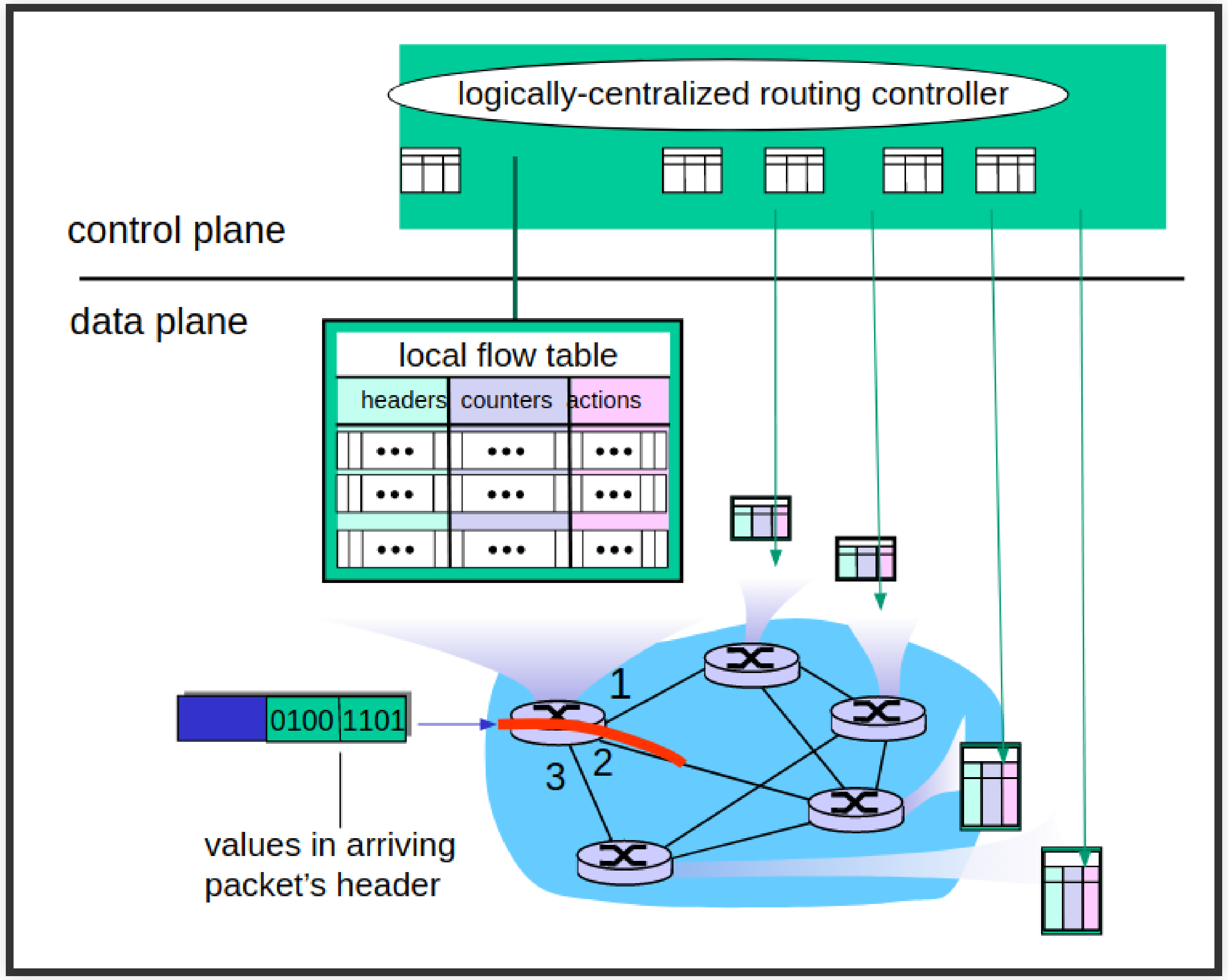
The chart above shows the availability of IPv6 connectivity around the world.

- 🟩 Regions where IPv6 is more widely deployed (the darker the green, the greater the deployment) and users experience infrequent issues connecting to IPv6-enabled websites.
- 🟧 Regions where IPv6 is more widely deployed but users still experience significant reliability or latency issues connecting to IPv6-enabled websites.
- 🟥 Regions where IPv6 is not widely deployed and users experience significant reliability or latency issues connecting to IPv6-enabled websites.

GENERALIZED FORWARD AND SDN

GENERALIZED FORWARDING AND SDN

Each router contains a flow table that is computed and distributed by a logically centralized routing controller

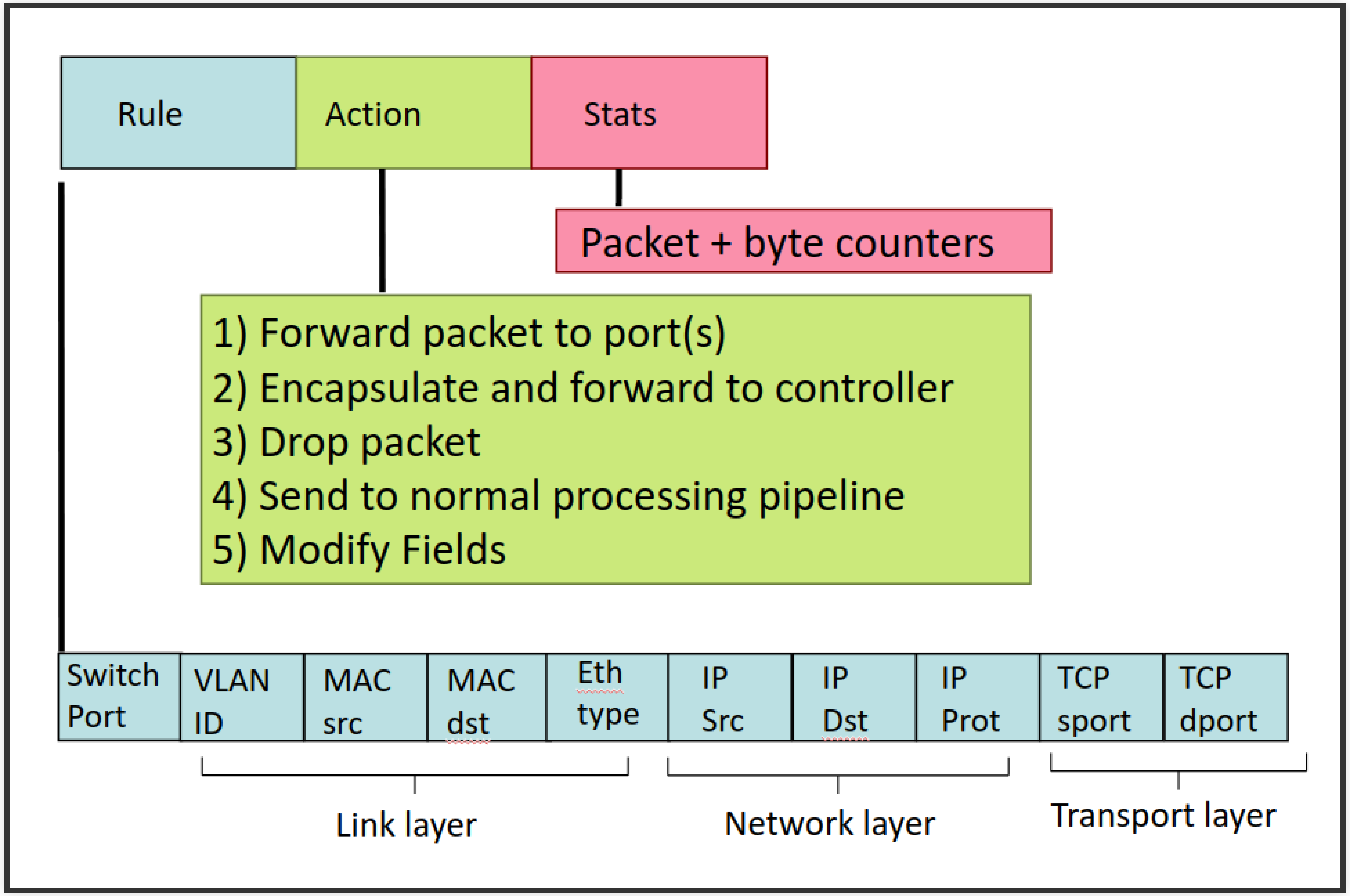

logically-centralized routing controller
control plane
data plane
local flow table
headers
counters
actions
0100
1101
1
2
3
values in arriving packet's header

OPENFLOW DATA PLANE ABSTRACTION

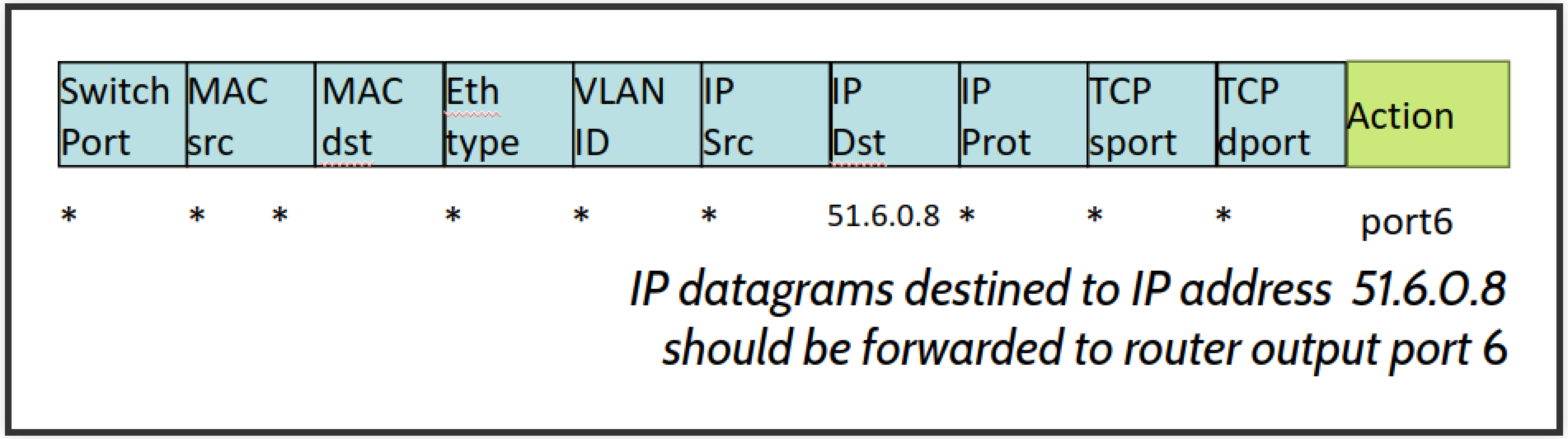
- **Flow:** defined by header fields
- Generalized forwarding: simple packet-handling rules
  - **Pattern:** match values in packet header fields
  - **Actions:** for matched packet: drop, forward, modify, matched packet or send matched packet to controller
  - **Priority:** disambiguate overlapping patterns
  - **Counters:** #bytes and #packets

Flow table in a router (computed and distributed by controller) define router's match+action rules

OPENFLOW: FLOW TABLE ENTRIES



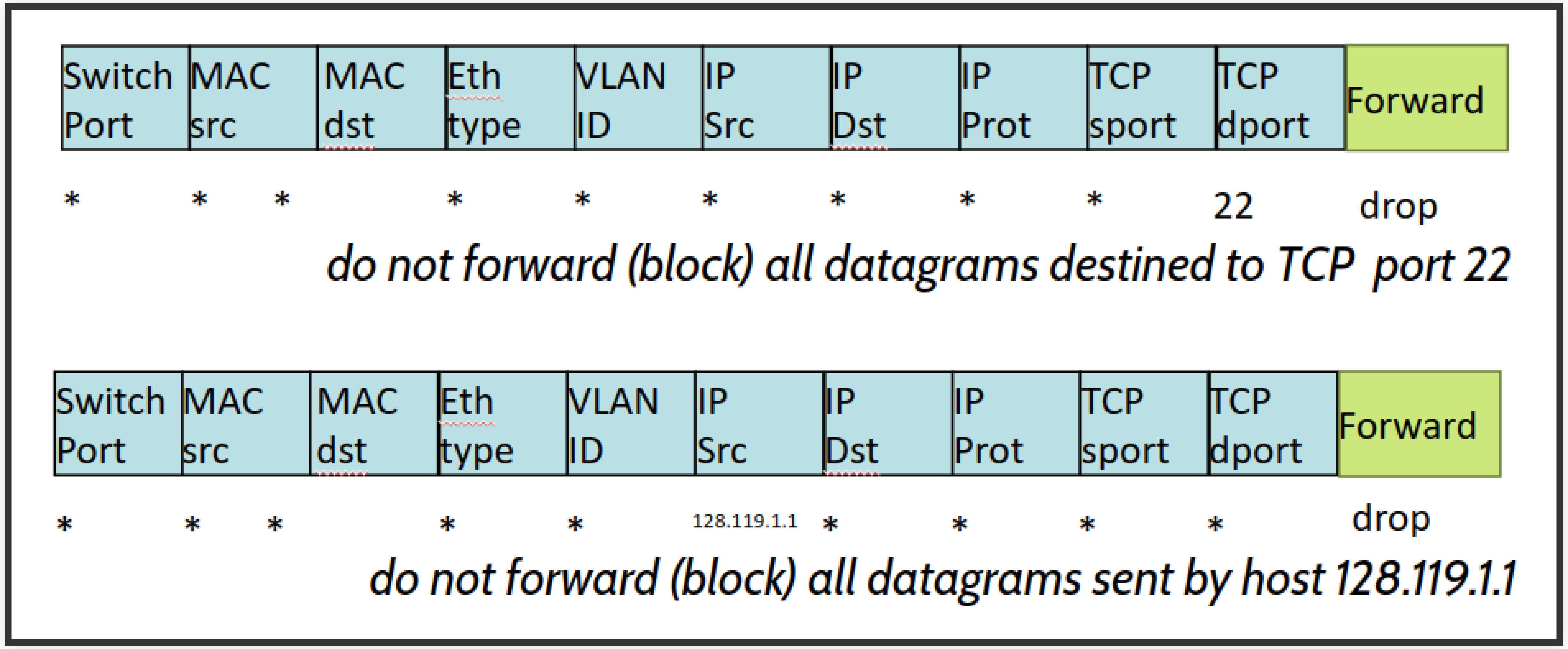
OPENFLOW: DESTINATION-BASED FORWARDING



Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	51.6.0.8	*	*	*	port6

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

OPENFLOW: FIREWALL



Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

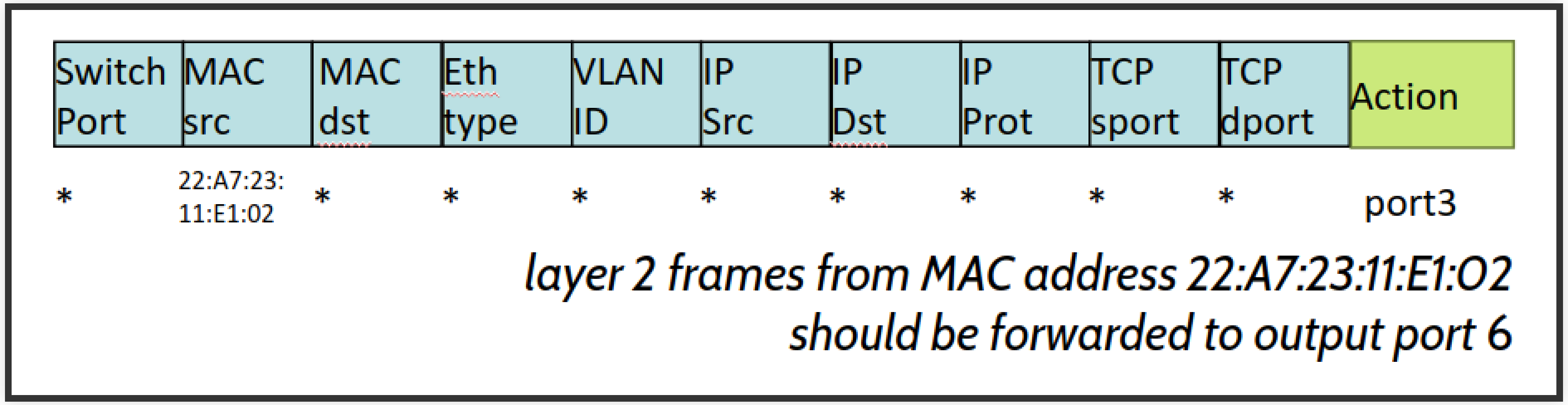
do not forward (block) all datagrams destined to TCP port 22

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	128.119.1.1	*	*	*	*	drop

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

OPENFLOW: SWITCH FORWARDING

Destination-based layer 2 (switch) forwarding:



Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	22:A7:23:11:E1:02	*	*	*	*	*	*	*	*	port3

layer 2 frames from MAC address 22:A7:23:11:E1:02 should be forwarded to output port 6

OPENFLOW ABSTRACTION

match+action: unifies different kinds of devices

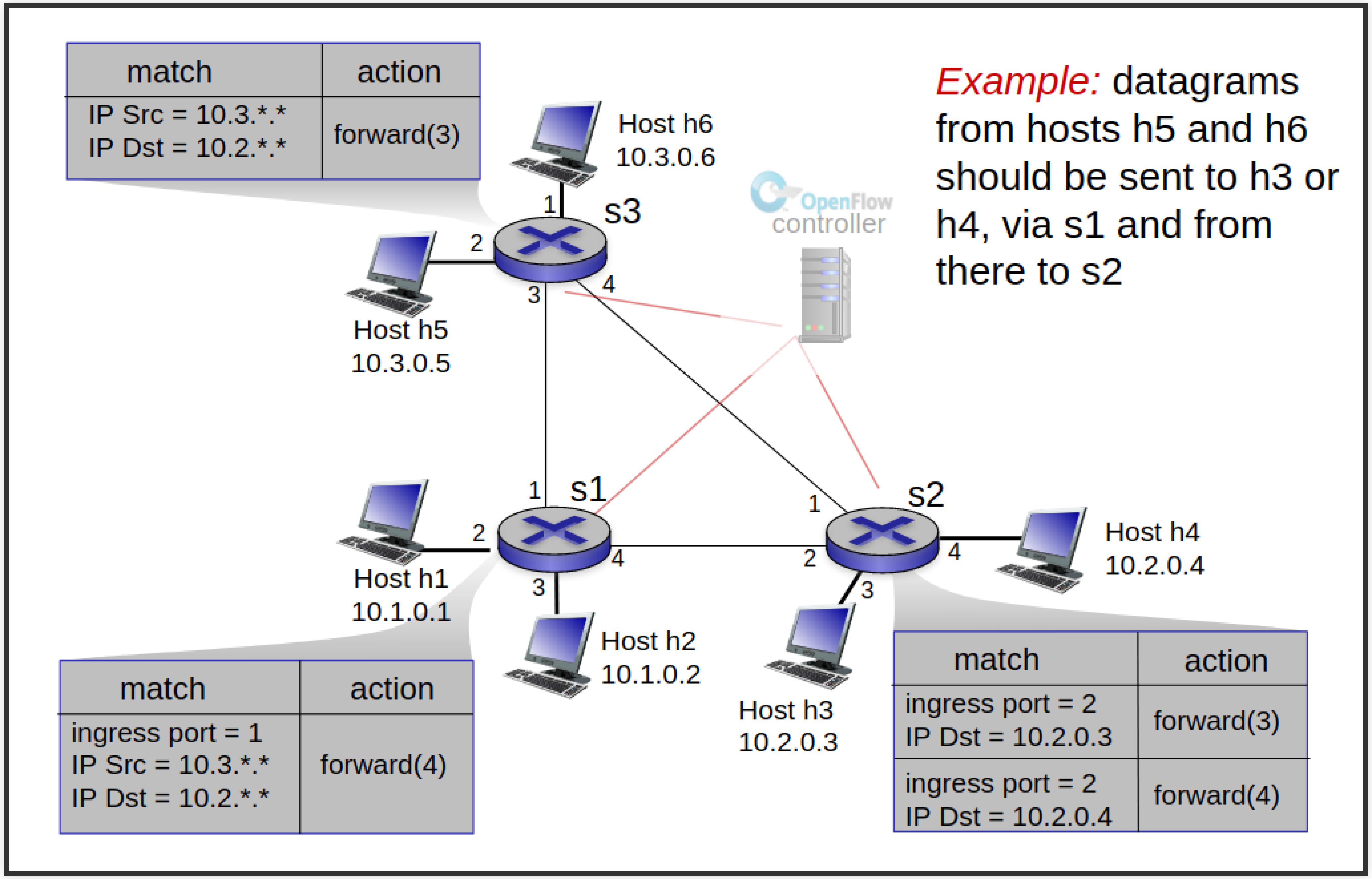
- **Router**
  - match: longest destination IP prefix
  - action: forward out a link
- **Switch**
  - match: destination MAC address
  - action: forward or flood

OPENFLOW ABSTRACTION

match+action: unifies different kinds of devices

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

OPENFLOW EXAMPLE



Example: datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2

match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)

match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 2 IP Dst = 10.2.0.3	forward(3)
ingress port = 2 IP Dst = 10.2.0.4	forward(4)

SUMMARY

- Per Router functions
- Inside a router
- IP: Internet protocol
- NAT
- IPv4 vs IPv6
- IP Forwarding and Generalized Forwarding