Chapter 4. Network Layer: The Data Plane

chapter goals:

- understand principles behind network layer services, focusing on data plane:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - generalized forwarding

How the Internet Works?

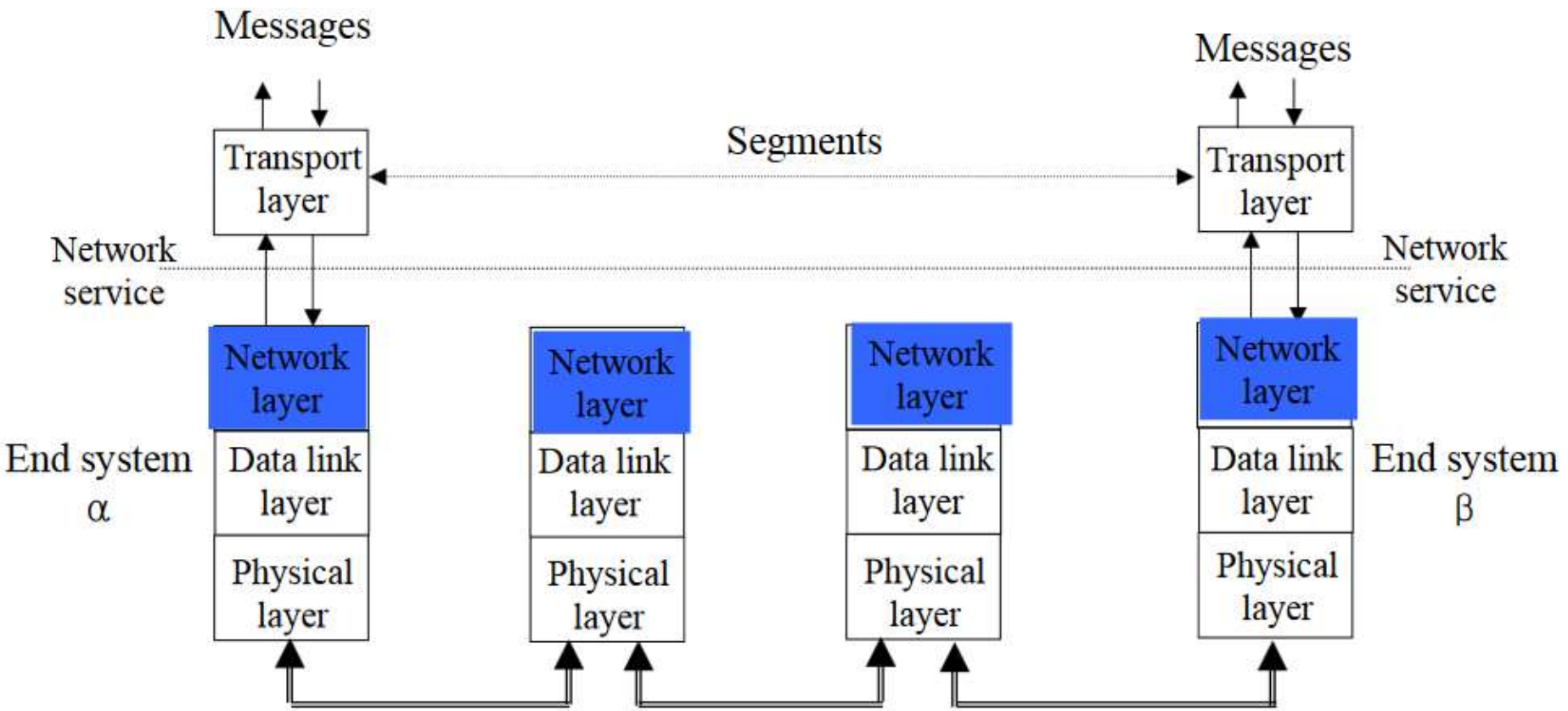
4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

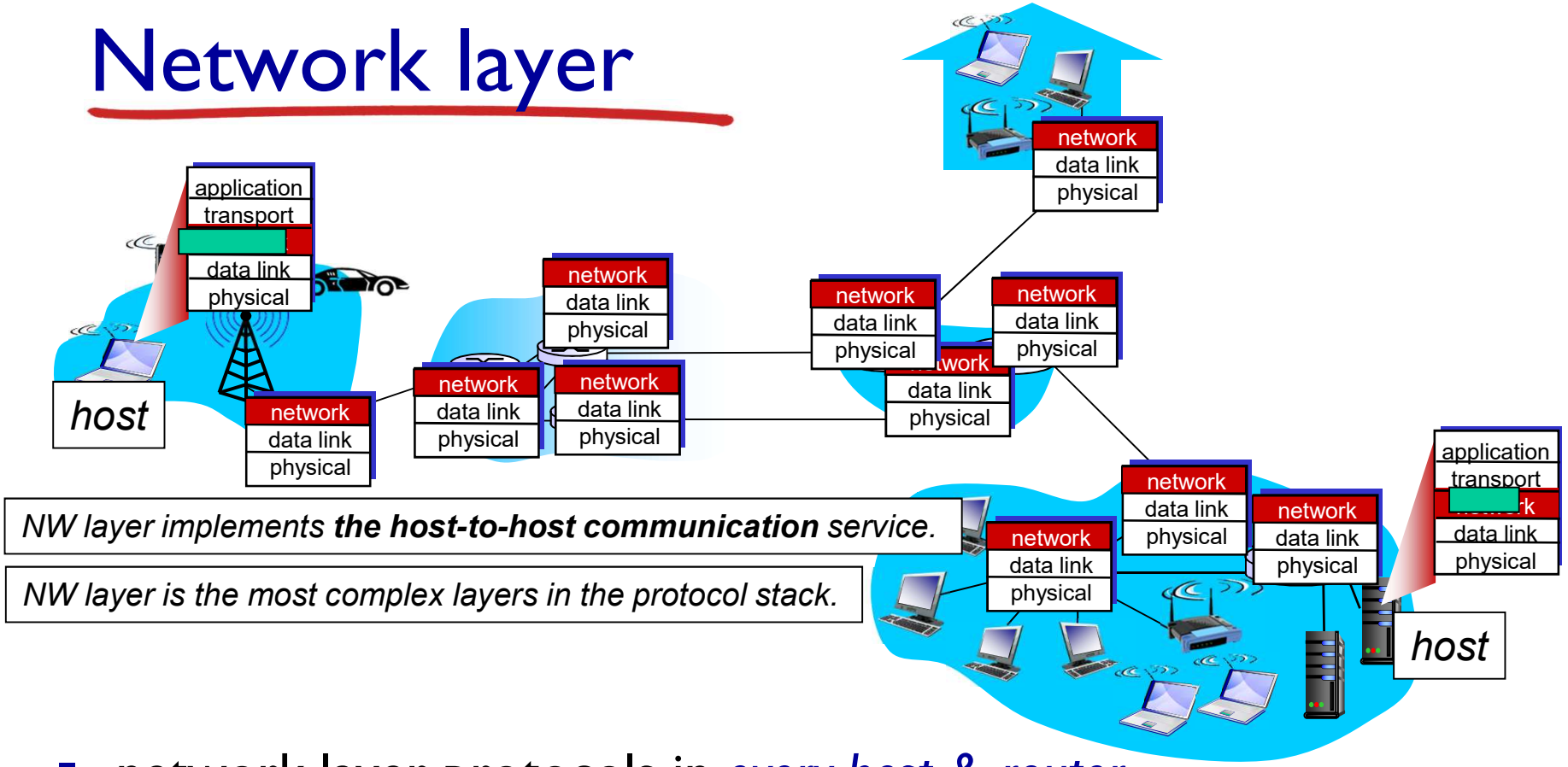
4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

4.1 Overview of Network layer
- data plane & control plane



Network layer



*NW layer implements **the host-to-host communication** service.*

NW layer is the most complex layers in the protocol stack.

- network layer protocols in *every host & router*
- *delivers* segment from *sending* to *receiving* host
  - on sending host) encapsulates segments into datagrams
  - on intermediate routers) forwards datagram; ***router examines header fields in all IP datagrams***
  - on receiving shost) decapsulates datagram into segments

Network service model

Q) What service does the network layer provide to the transport layer?

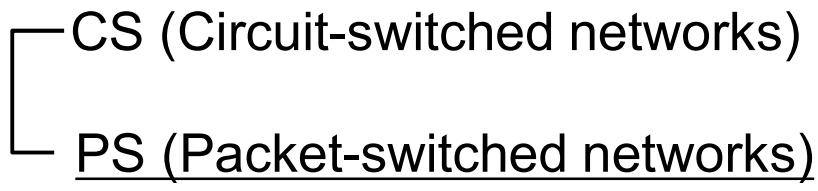
Network service model defines *the characteristics of e2e transport of packets between sending & receiving hosts*

example services for individual datagrams:

- guaranteed delivery
- guaranteed delivery with bounded delay
  - a specified host-to-host delay bound (less than 40 msec)

example services for a flow of datagrams:

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



* VC NW (ATM)

- evolved from telephony
- human conversation:
  - ✓ strict timing, reliability requirements
  - ✓ need for guaranteed service
- "dumb" end systems, e.g., telephones
  - ✓ ***complexity inside network***

* Datagram NW (Internet)

- "smart" end systems, e.g., computers
  - ✓ can adapt, perform control, error recovery
  - ✓ ***simple inside network, complexity at "edge"***
- data exchange among computers
  - ✓ "elastic" service, no strict timing req.
- Minimal (no) service guarantees
  - ✓ makes it *easier to interconnect networks that use very heterogeneous link-layer technologies*

Network layer service models:

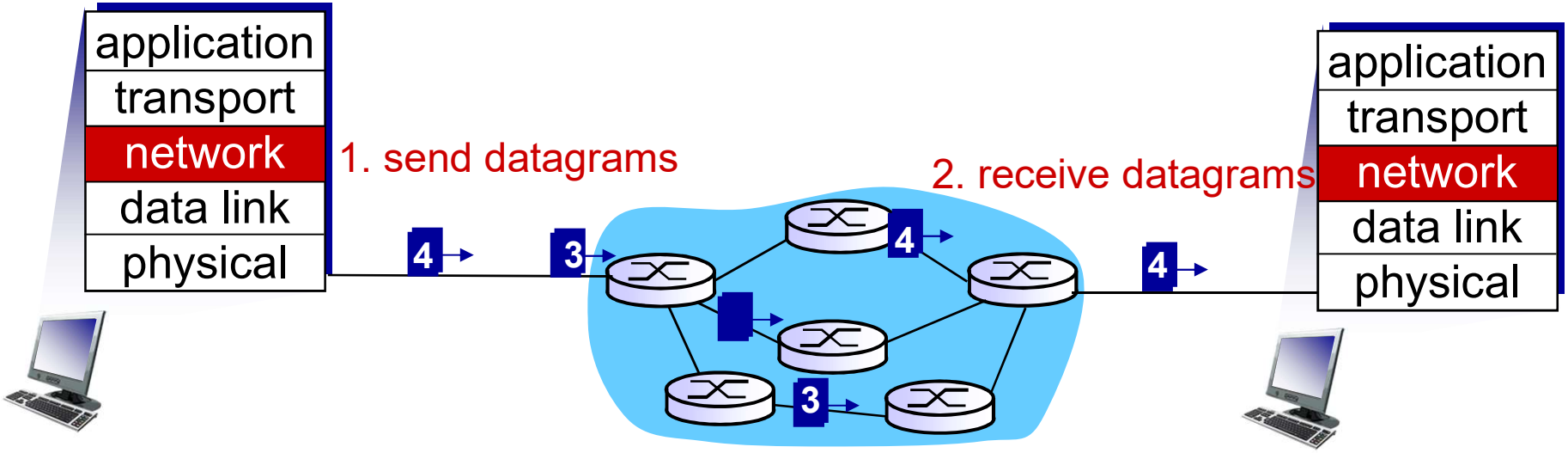
Datagram* network** provides network-layer ***connectionless service.

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

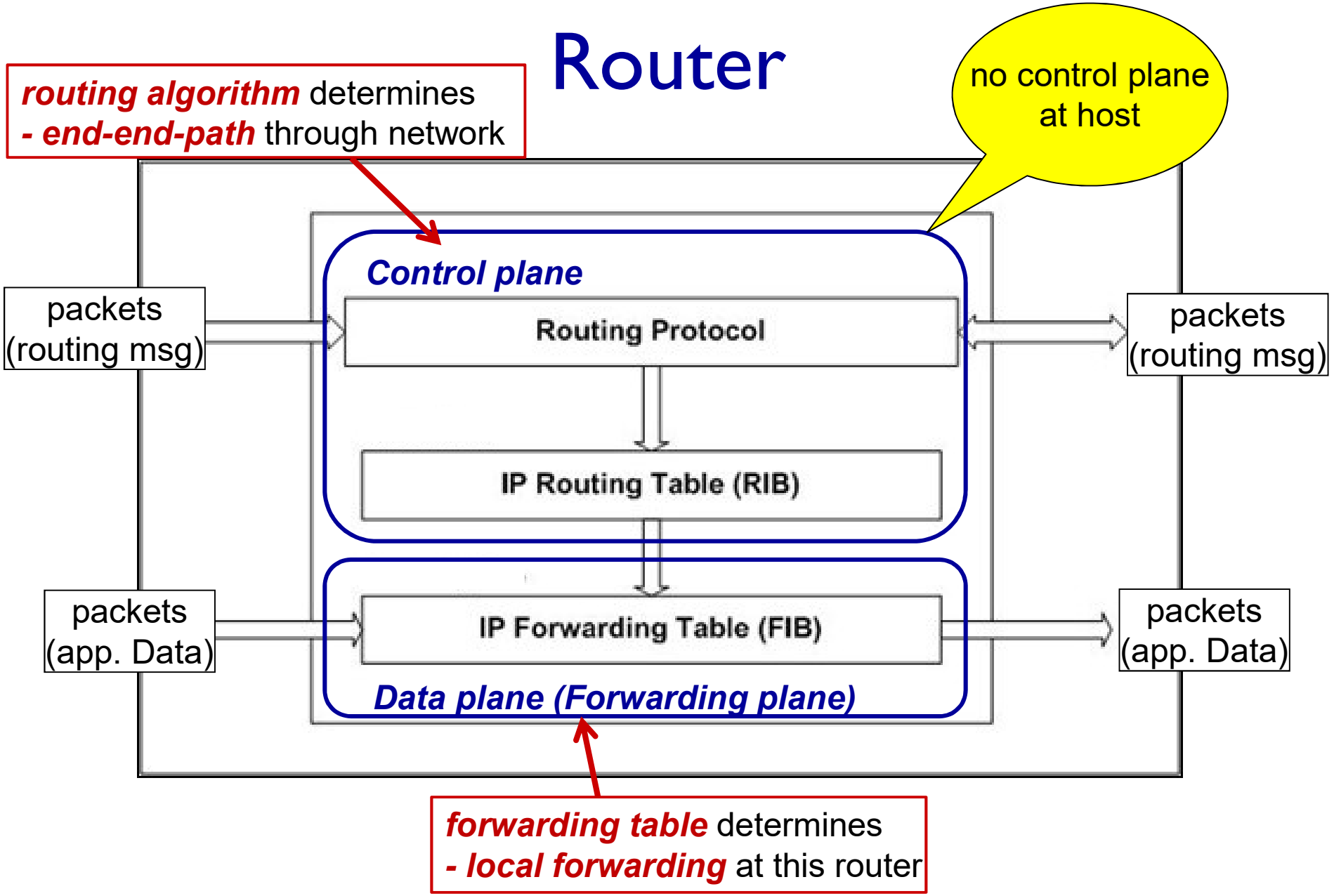
Virtual-circuit (VC) network provides network-layer ***connection*** service

Datagram networks

- *no call setup* at network layer
- routers: *no state* about end-to-end connections
  - no network-level concept of "connection"
- packets forwarded *using destination host address in datagram header and forwarding table(FIB) at each router*



Router

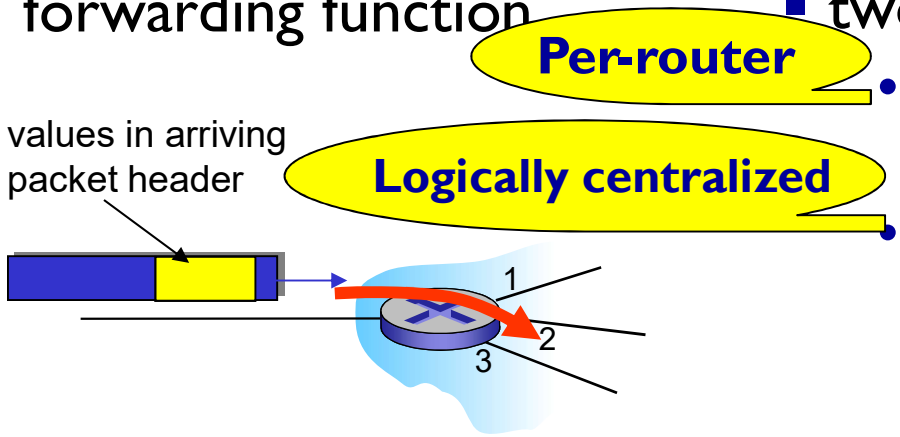


http://aftabsiddiqui.com/index.php/ip-routing-table-rib-and-forwarding-table-fib/

Network layer: data plane, control plane

Data plane *forwarding*

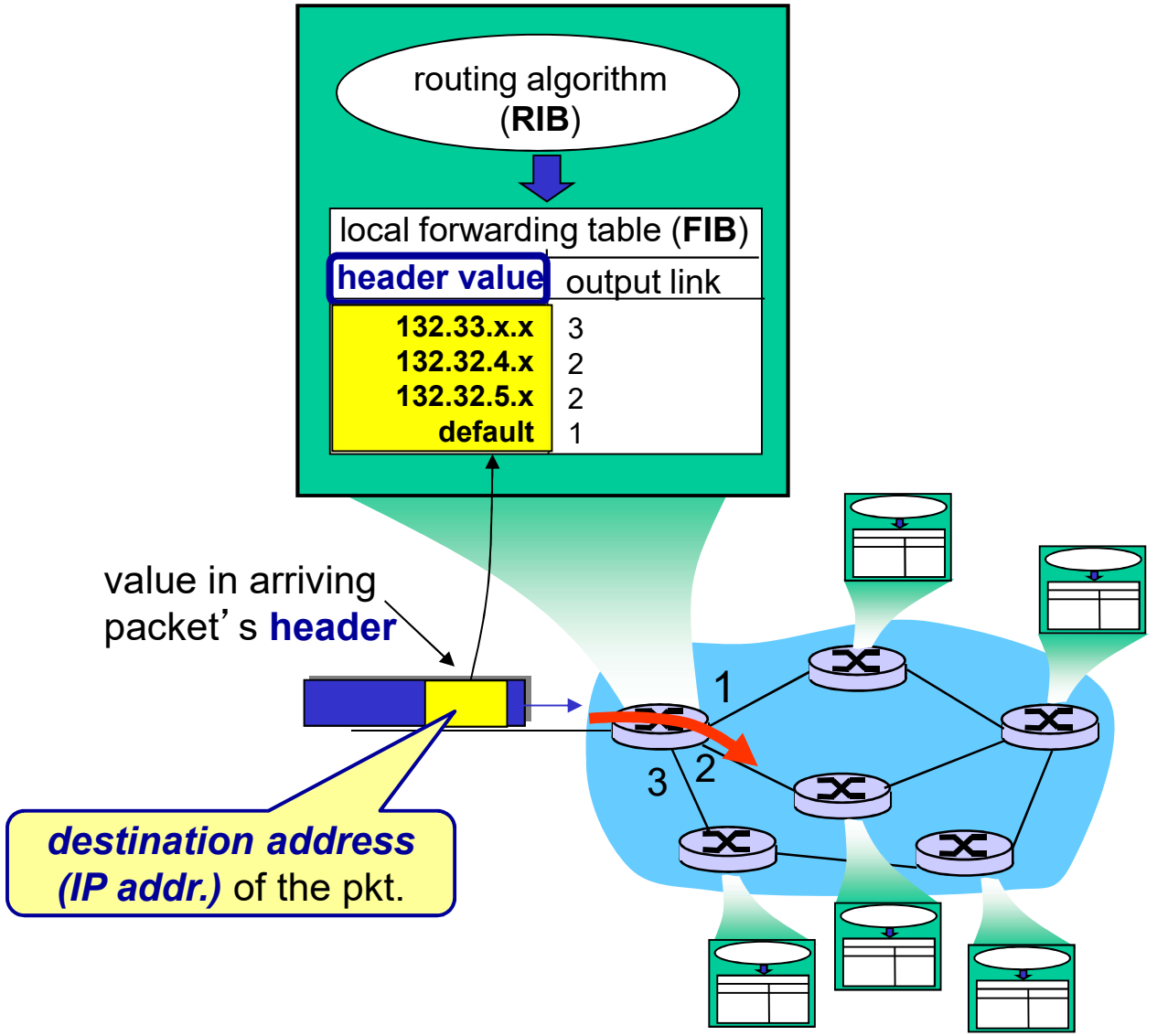
- local, per-router function
- determines how datagram arriving on router input port is forwarded to router output port
- forwarding function



Control plane *routing*

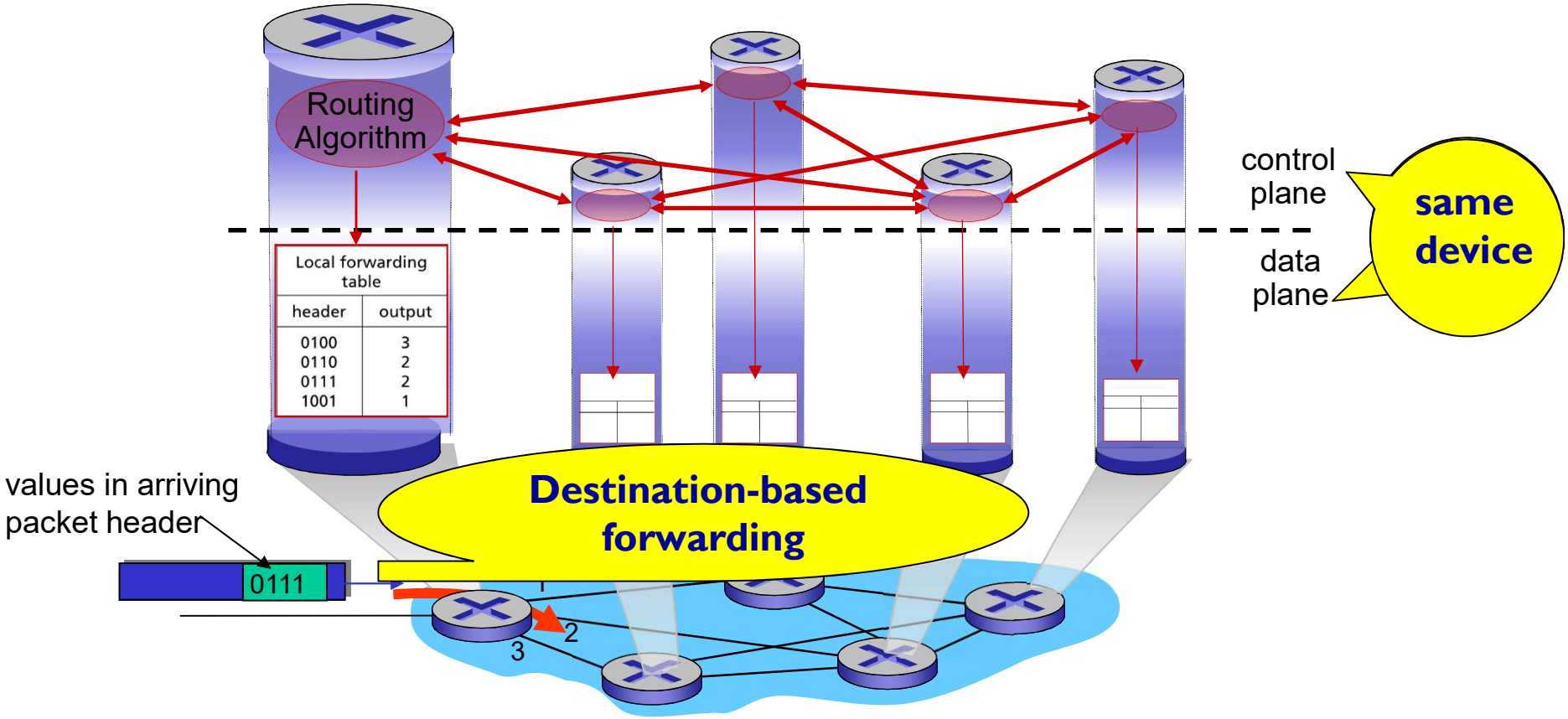
- network-wide logic
- determines how datagram is routed among routers along end-end path from source host to destination host
- two control-plane approaches:
  - *traditional routing algorithms:* implemented in routers
  - *software-defined networking (SDN)*: implemented in (remote) servers

Interplay between routing and forwarding



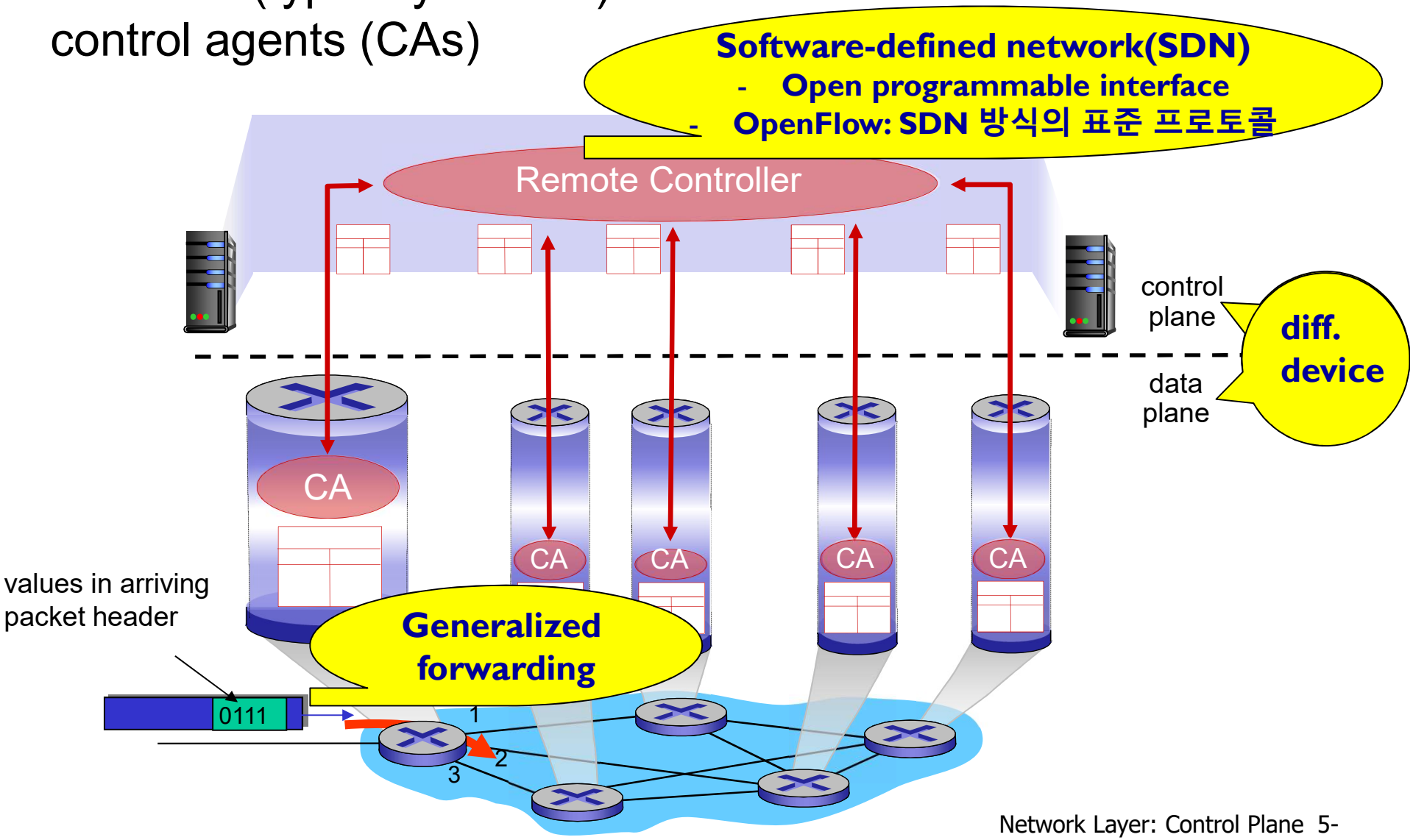
Per-router control plane

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

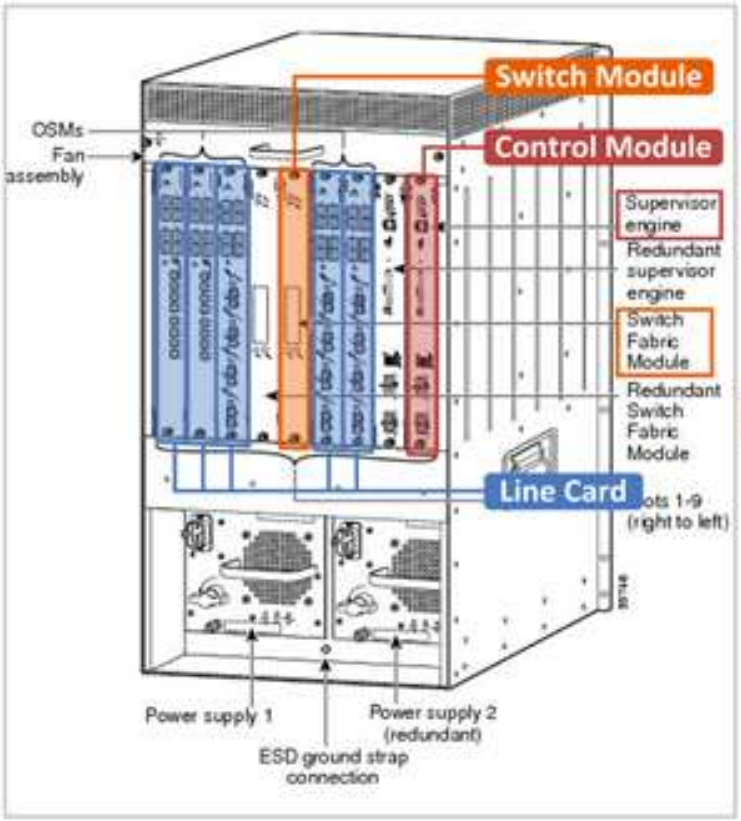


Logically centralized control plane

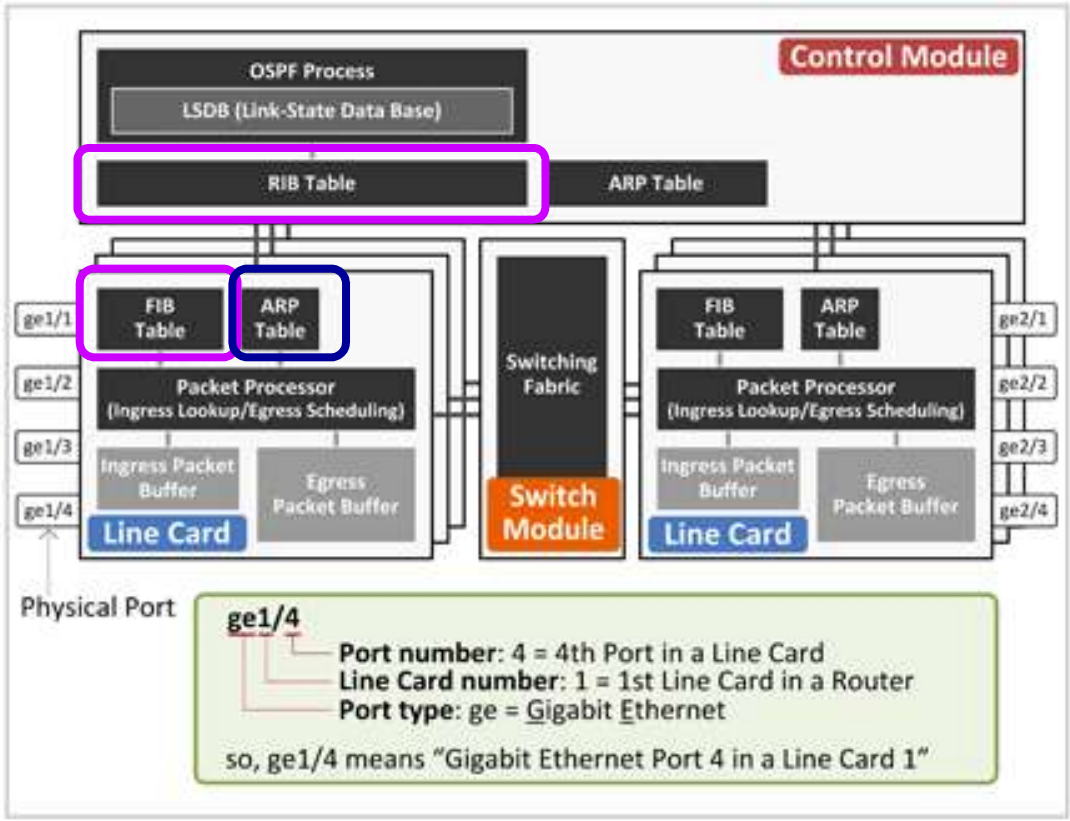
A distinct (typically remote) controller interacts with local control agents (CAs)



4.2 What's inside a router



Cisco 7600 Router

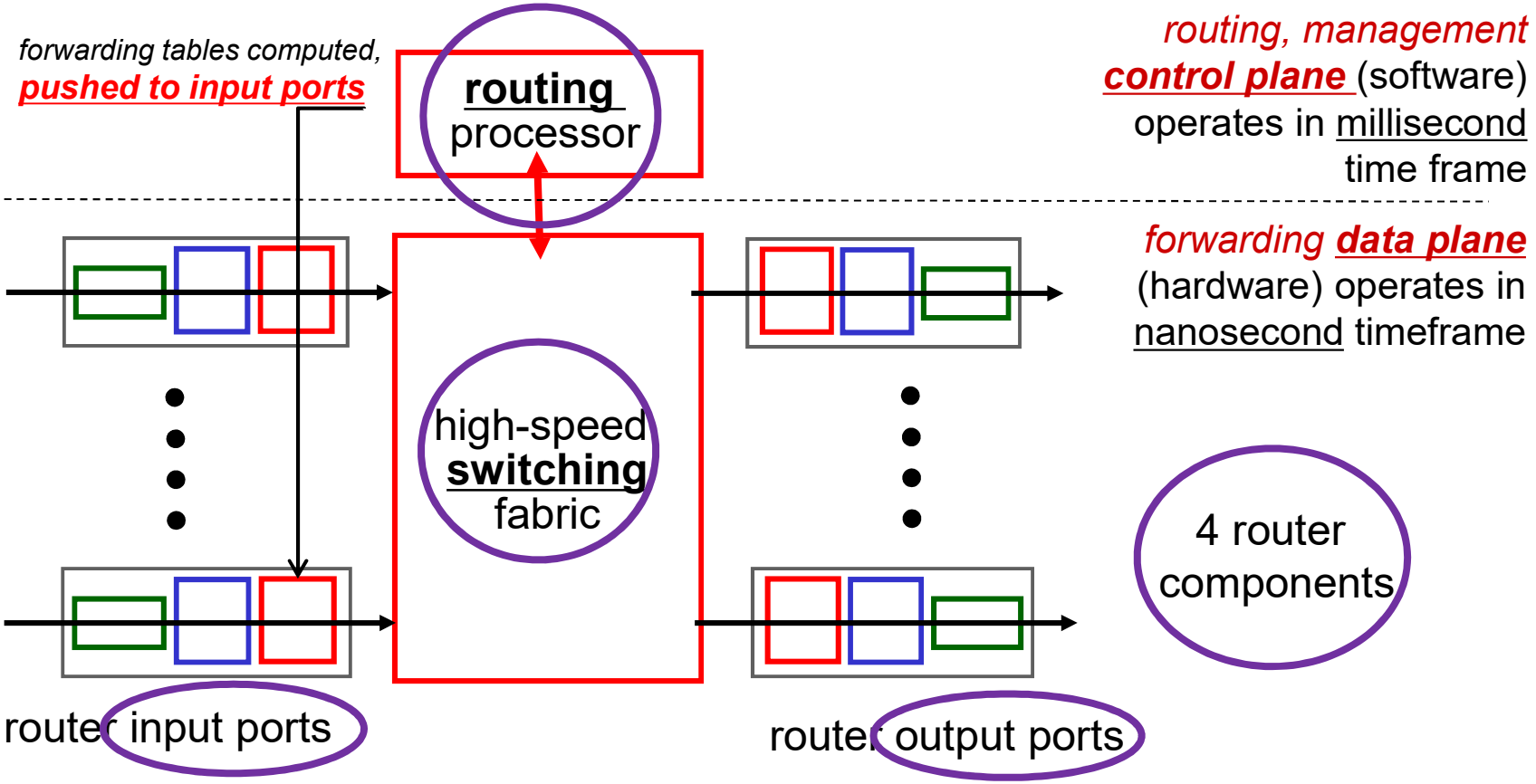


General Router Architecture

http://www.netmanias.com/en/post/blog/6338/ip-routing-network-protocol-switching/switching-and-routing-part-1-router-architecture

Router architecture overview

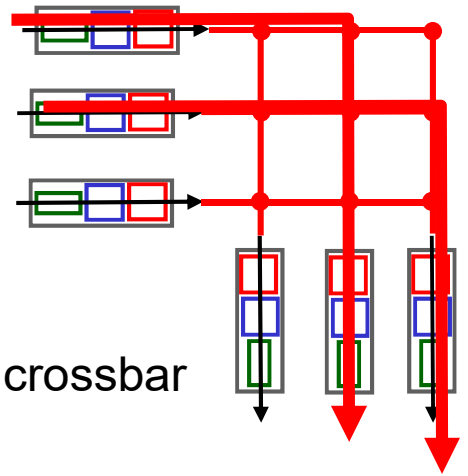
- high-level view of generic router architecture:



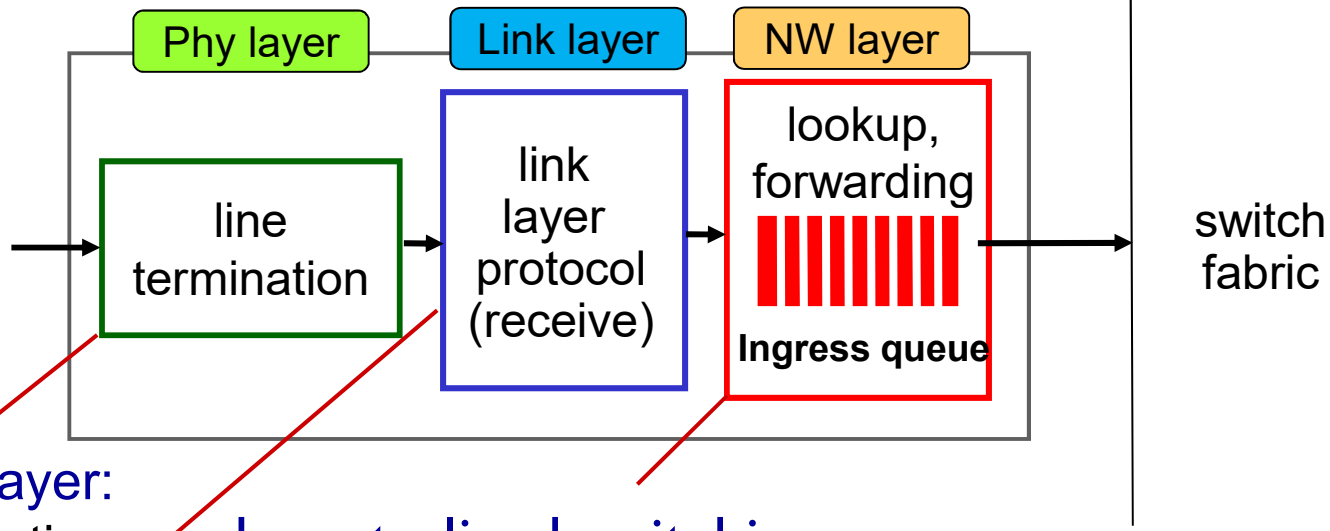
Port=NIC(H/W), different from a socket port(S/W)

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
- interconnection network
  - Crossbar networks are capable of *forwarding multiple packets in parallel*, which have *different input ports and different output ports*.
  - However, *if two packets* from two different input ports *are destined to the same output port*, then *one will have to wait at the **input** buffer* since only one packet can be sent at a time
  - Cisco 12000: switches 60 Gbps through the interconnection network



Input port functions



physical layer:
bit-level reception

data link layer:
e.g., Ethernet
see chapter 6

decentralized switching:

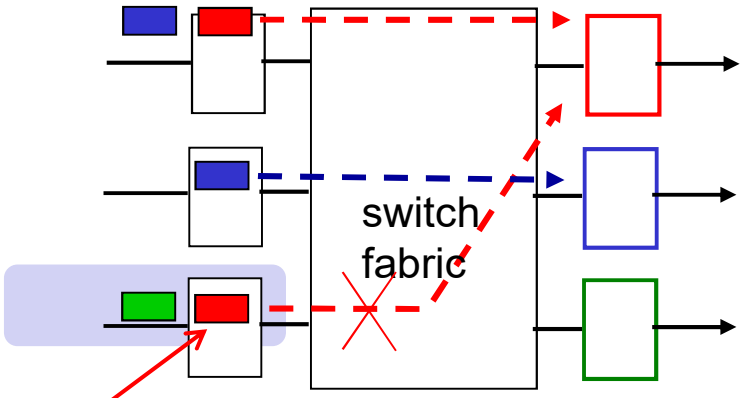
- Given datagram, using header field(**dest. IP addr.**) values, **lookup output port** using **FIB in input port** memory (*"**match plus action**"*)
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than "forwarding rate into switch fabric", i.e., *the switch fabric is not fast enough, then input queueing delay or loss due to buffer overflow*

Head-of-the-Line (HOL) blocking

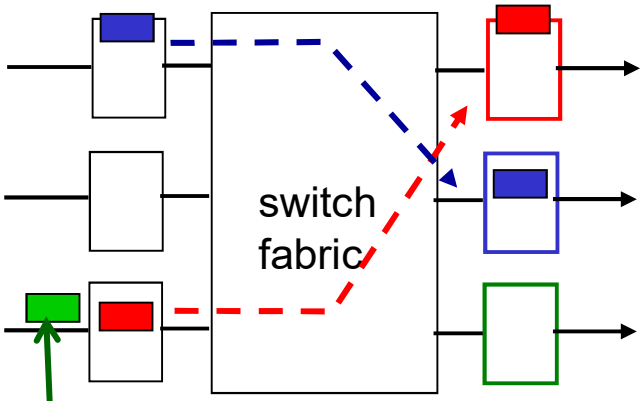
- A *queued pkt is blocked* by another packet at the head of the line *in an input queue*.

→ The queued pkt *must wait* for transfer through the fabric.

→ An input queue will grow and *pkt loss may occur in the input queue*.



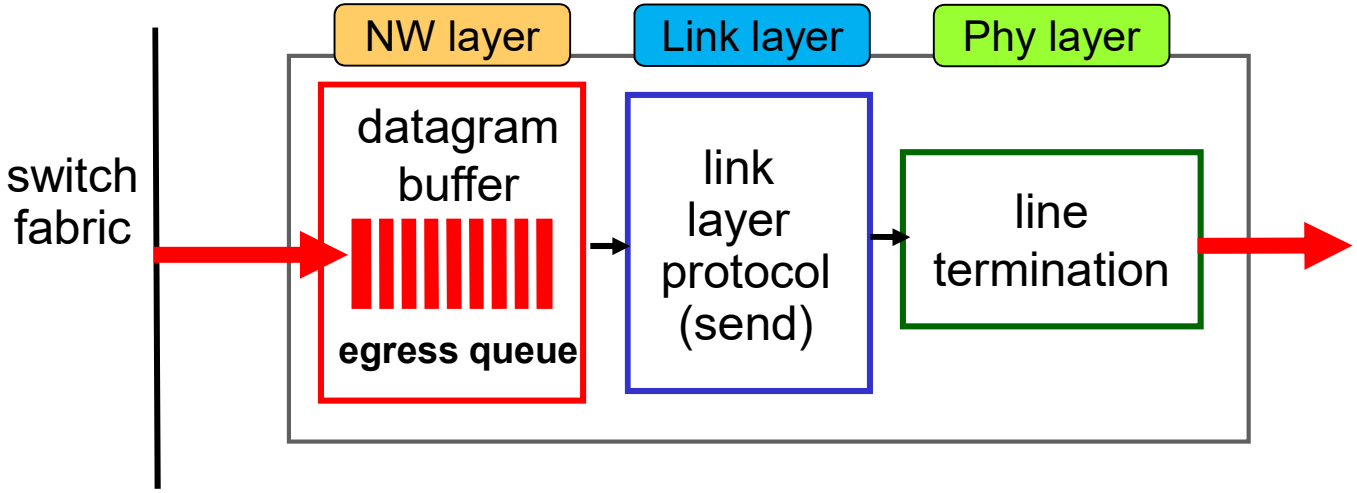
output port contention:
only one (upper) red datagram can be transferred since a crossbar switching fabric can transfer only one packet to a given output port at a time.
lower red packet is blocked



one packet time later:
green packet experiences HOL blocking

Results) even though the green output port is available, the green pkt must wait for *one packet time*!

Output ports



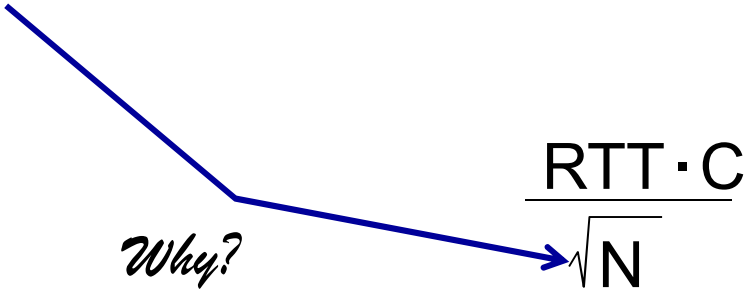
- *Buffering/queueing* required when *datagrams arrive from fabric faster than output line speed (transmission rate)*
  - ✓ *queueing (delay) and loss due to output port buffer overflow!*
- *Packet scheduler* chooses among queued datagrams for transmission and *provides quality-of-service guarantees*.
  - Internet : FCFS
  - Other network : premium service first, e.g., weighted fair queuing (WFQ)

Priority scheduling – who gets best performance
➔ Violate ***network neutrality*** in the Internet

How much buffering at routers?

- RFC 3439 rule of thumb: *average buffering equal to "typical" RTT* (say 250 msec=0.25sec) *times link capacity C*
  - e.g., C = 10 Gbps link: 2.5 Gbit buffer

- recent recommendation: with a large number of TCP flows (*N*), buffering equal to

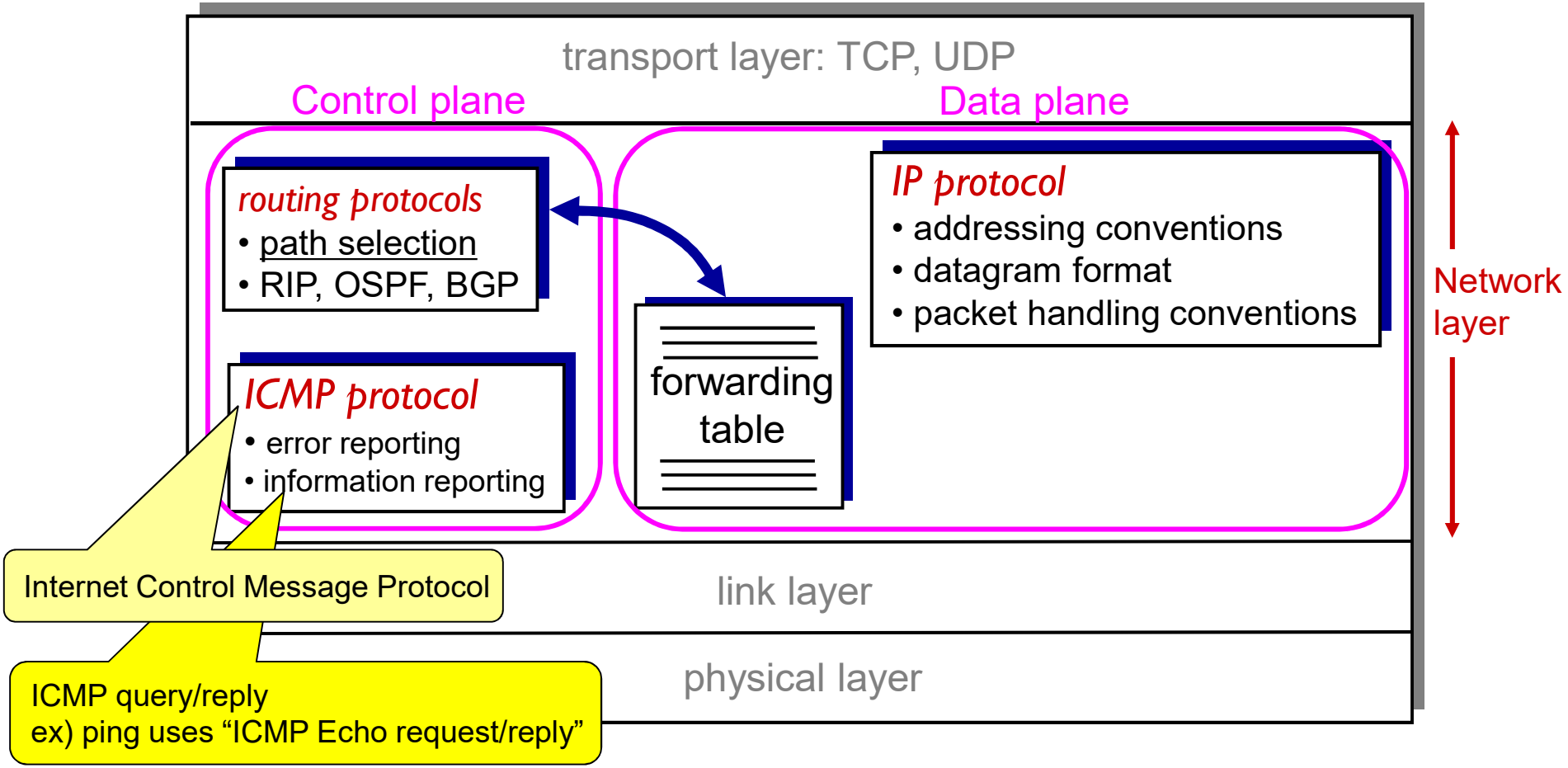


Chapter 4.3
Internet Protocol (IP)

- datagram format & fragmentation
- IPv4 addressing
- network address translation (NAT)
- IPv6

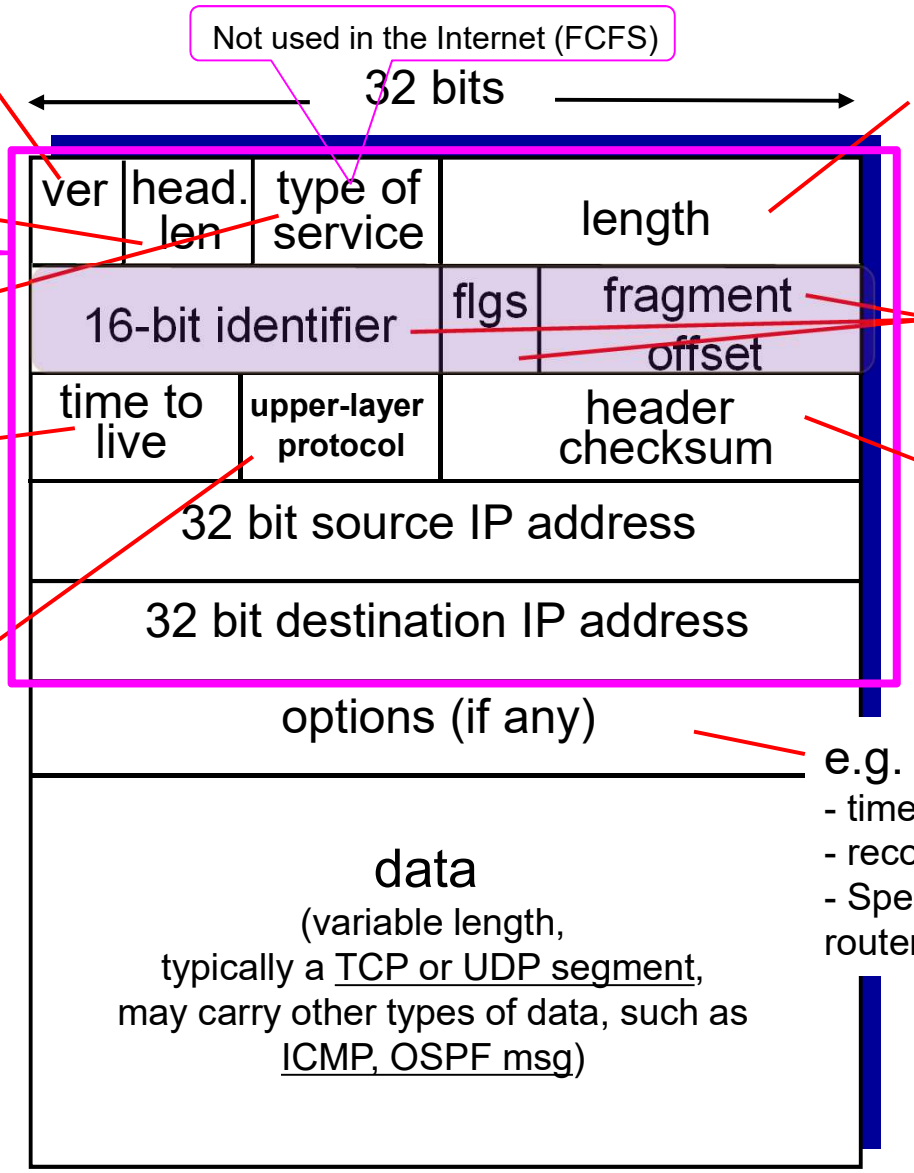
The Internet network layer

host, router network layer functions:



IPv4 datagram format

Datagram plays a central role in the Internet!!



- max number of remaining hops
- decremented by 1 at each router
- If it reaches 0 → drop the datagram!

It binds network and transport layer. Analogous to the role of the port # field in the transport layer segment

ex) 6=TCP, 17=UDP, 1=ICMP, 89=OSPF

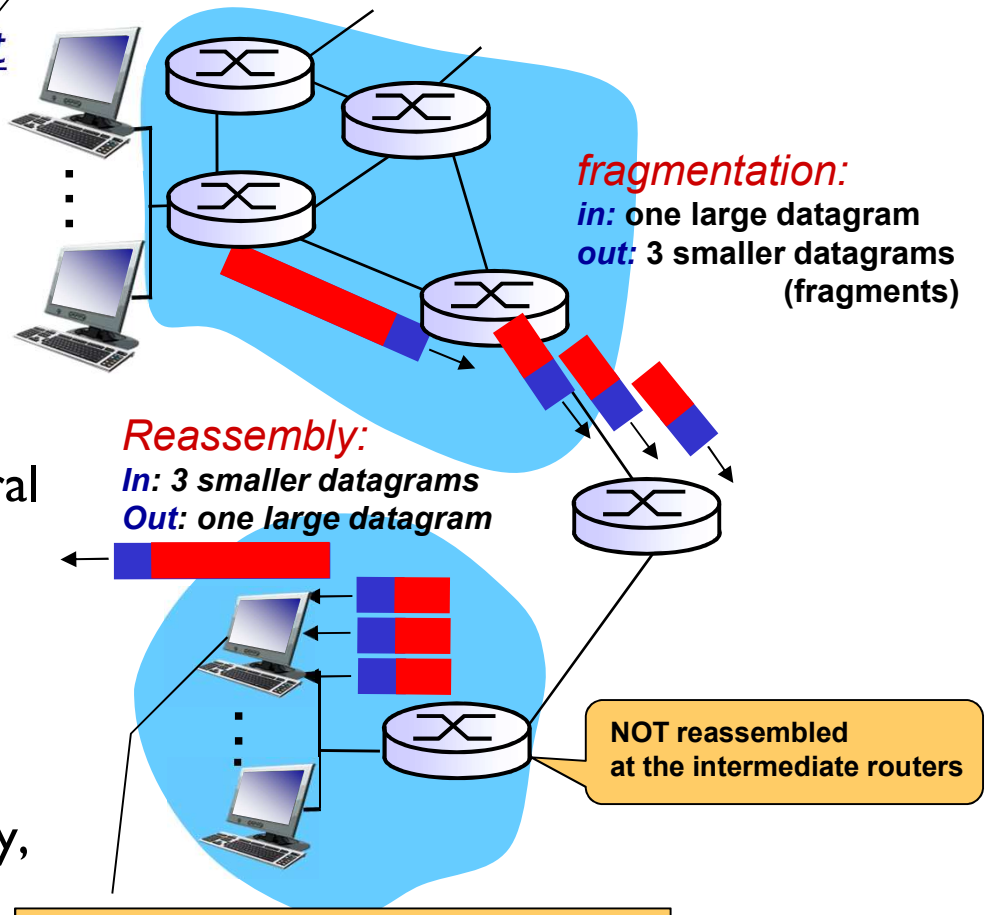
how much overhead?

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

IPv4 fragmentation, reassembly

- network links have *MTU* (max. transmission unit) – *max. amount of data that a link-level frame can carry*
  - different link types along the path, different MTUs
- large IP datagram divided ("*fragmented*") at routers
  - one datagram becomes several datagrams (*fragment*)
  - "*reassembled*" *only at final destination "host" not router before they reach the transport layer using "IP header".*
  - IP header bits used to identify, order related fragments

Ethernet frames carry up to 1,500 bytes,
Some wide-area links carry no more than 576 bytes

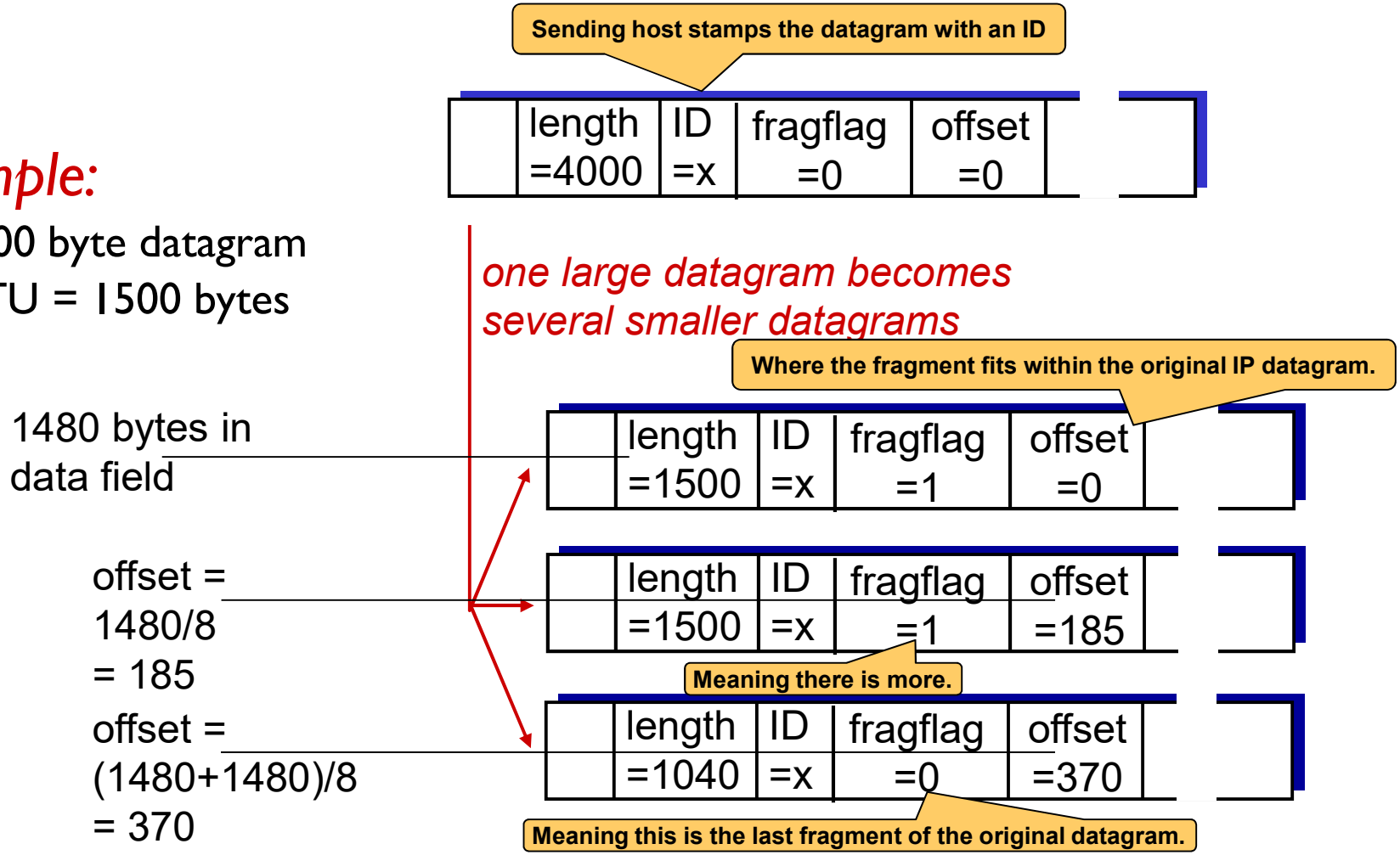


If one or more of fragments does not arrive,
→ Discard the incomplete datagram,
(→ TCP will recover from this loss.)

IPv4 fragmentation at router

example:

- 4000 byte datagram
- MTU = 1500 bytes



P19. Consider sending a 2,400-byte datagram into a link that has an MTU of 700bytes. Suppose the original datagram is stamped with the identification number 422. How many fragments area generated? What are the value in the various fields in the IP datagram(s) generated related to fragmentation?

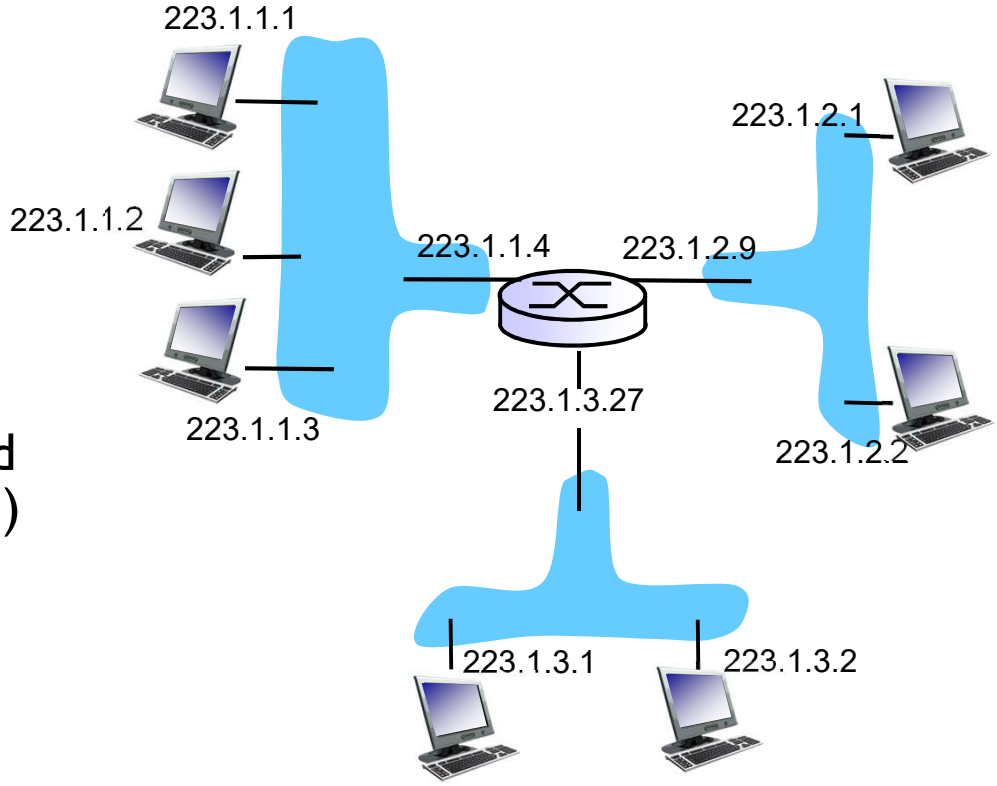
Total datagram length	ID	Frag. Flag	offset

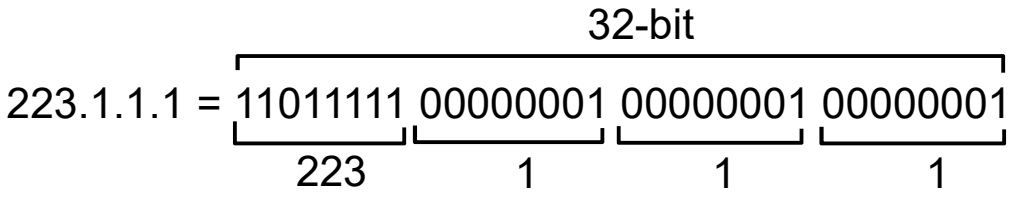
Chapter 4.3
Internet Protocol (IP)

- datagram format & fragmentation
- IPv4 addressing
- network address translation (NAT)
- IPv6

IPv4 addressing: introduction

- ***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)
- ***IPv4 address:*** 32-bit *identifier for host, router interface*
- *IP addresses associated with each interface*

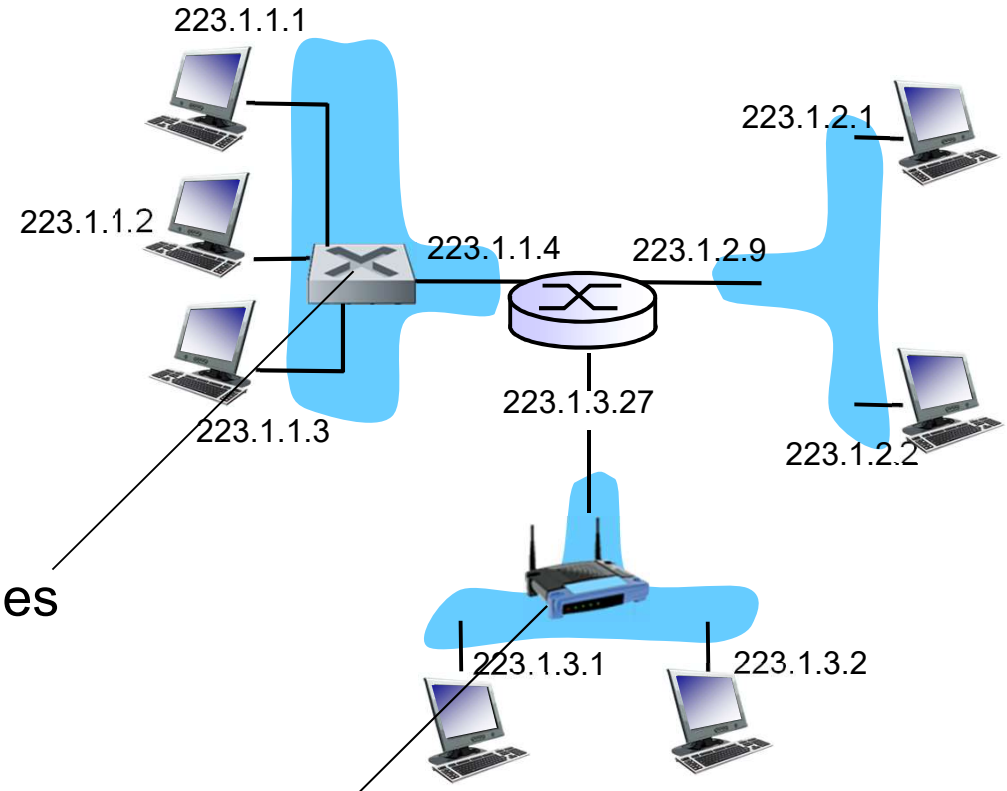




IP addressing: introduction

Q: how are interfaces actually connected?

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



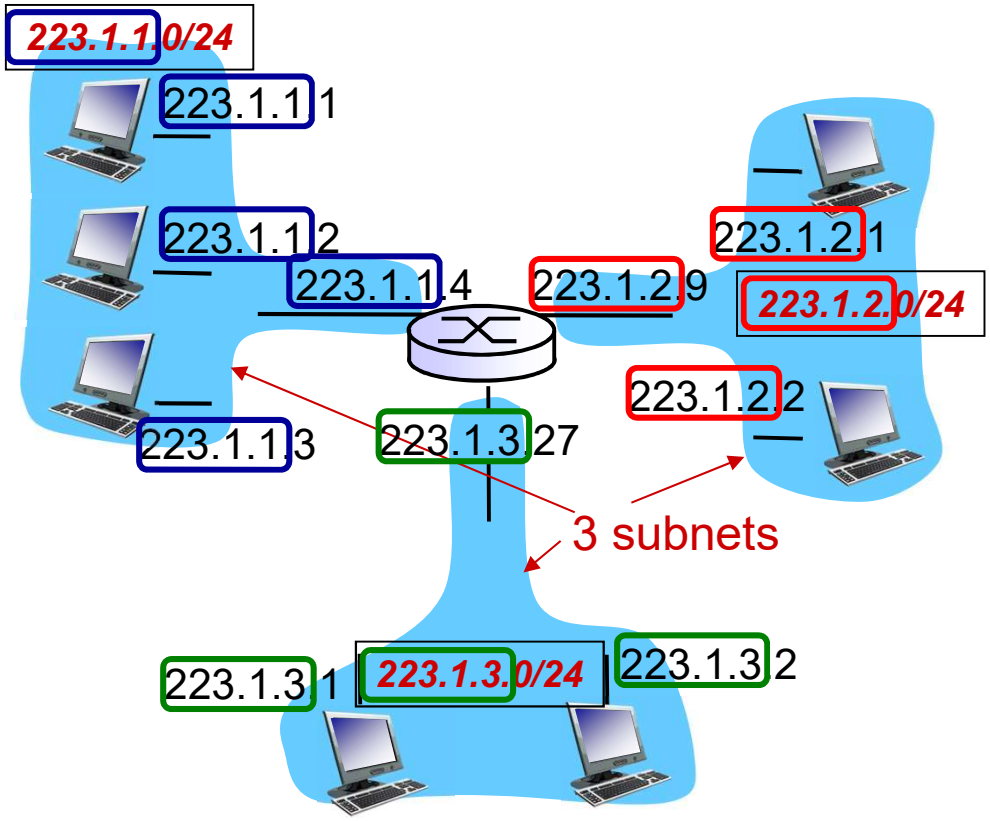
A: wired Ethernet interfaces connected by Ethernet switches

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

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

network consisting of 3 subnets



subnet part: leftmost 24-bit

subnet mask : /24

223.1.1.1 = 11011111 00000001 00000001 00000001

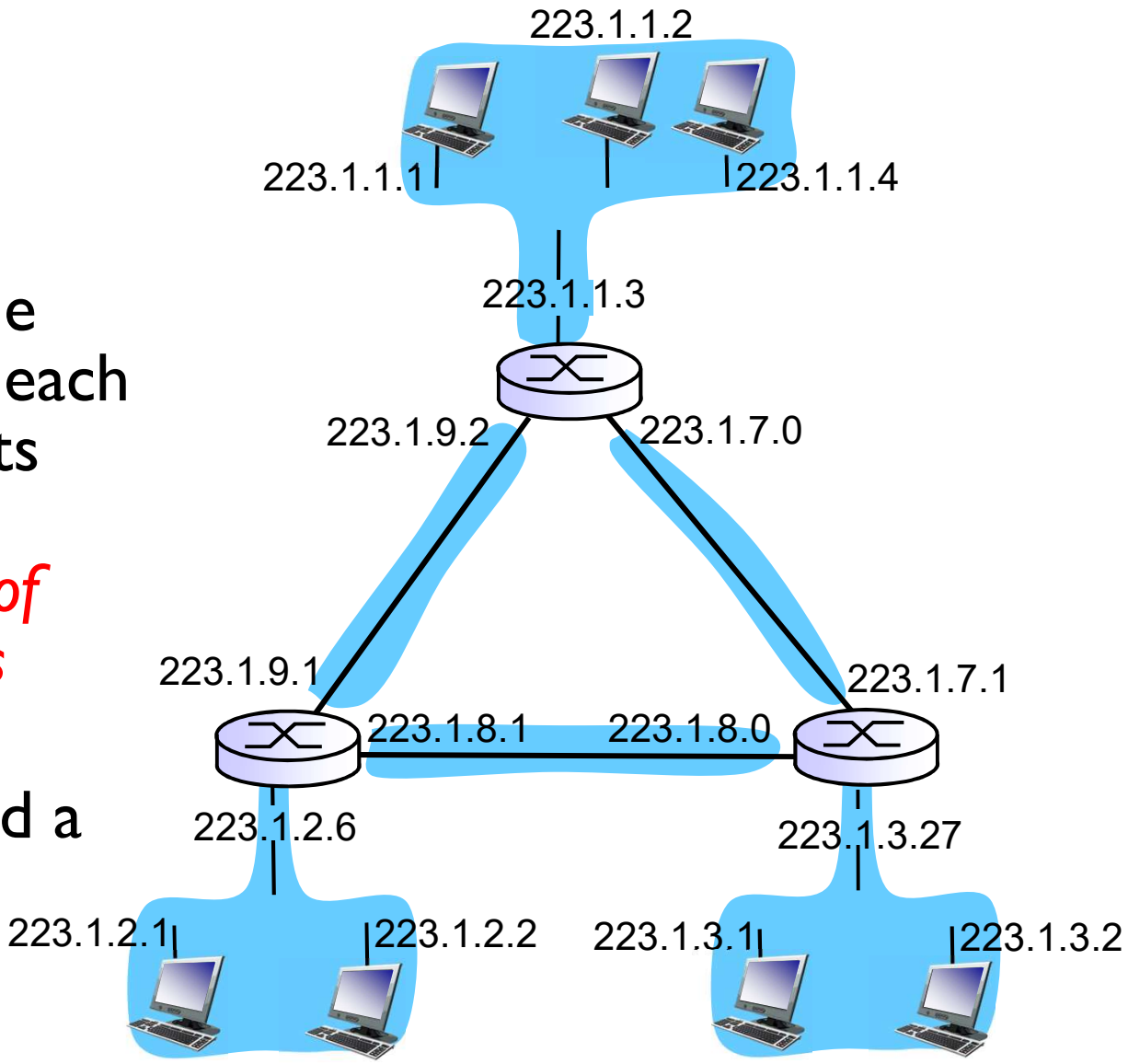
host part: rightmost (32-24=) 8-bit

Subnets

how many?

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*



IPv4 addressing: classful addressing

Class	1st Octet Decimal Range	1st Octet High Order Bits	Network/Host ID (N=Network, H=Host)	Default Subnet Mask	Number of Networks	Hosts per Network (Usable Addresses)
A	1 – 126*	0	N.H.H.H **/8**	255.0.0.0	126 ($2^7 - 2$)	16,777,214 ($2^{24} - 2$)
B	128 – 191	10	N.N.H.H **/16**	255.255.0.0	16,382 ($2^{14} - 2$)	65,534 ($2^{16} - 2$)
C	192 – 223	110	N.N.N.H **/24**	255.255.255.0	2,097,150 ($2^{21} - 2$)	254 ($2^8 - 2$)
D	224 – 239	1110	Reserved for Multicasting			
E	240 – 254	1111	Experimental; used for research			

Broadcast addr.: **x.x.255.255**
Network identifier : **x.x.0.0**

Note: Class A addresses 127.0.0.0 to 127.255.255.255 cannot be used and is reserved for loopback and diagnostic functions.

※ Drawback of classful addressing

- Class C provides too small hosts (up to 254 hosts).
- Class B provides too many hosts (up to 65534 hosts).

  ➔ ***Rapid depletion*** of the class B address spaces & ***poor utilization*** of the assigned address space.

  ➔ ***CIDR (Classless InterDomain Routing)***

Private IP Addresses

Class	Private Networks	Subnet Mask	Address Range
A	10.0.0.0	255.0.0.0	10.0.0.0 - 10.255.255.255
B	172.16.0.0 - 172.31.0.0	255.240.0.0	172.16.0.0 - 172.31.255.255
C	192.168.0.0	255.255.0.0	192.168.0.0 - 192.168.255.255

IPv4 addressing: CIDR

CIDR: Classless InterDomain Routing

- ***subnet*** *portion* of address of ***arbitrary*** *length*
- address format: a.b.c.d/***x***, where *x is # leftmost bits in subnet portion* of address
- ***x*** is called, "***subnet mask***", "***network prefix***", "***prefix***"

11001000 00010111 00010000 00000000

200.23.16.0/23

Quiz) Consider a subnet with prefix 128.119.40.128/26. Give an example of one IP address (of form xxx.xxx.xxx.xxx) that can be assigned to this network.

min. h where $2^h - 2 >= 4$

(1) 4개의 IPv4 주소가 필요한 ISP 에서 사용하는 주소의 host part는 _____ bits 이다.

➔ 그렇다면 해당 ISP 에서 사용하는 IPv4 주소의 subnet mask는 _____ bits?

min. h where $2^h - 2 >= 35$

(2) 35개의 IPv4 주소가 필요한 ISP에서 사용하는 주소의 host part는 _____ bits 이다.

➔ 그렇다면 해당 ISP 에서 사용하는 IPv4 주소의 subnet mask는 _____ bits?

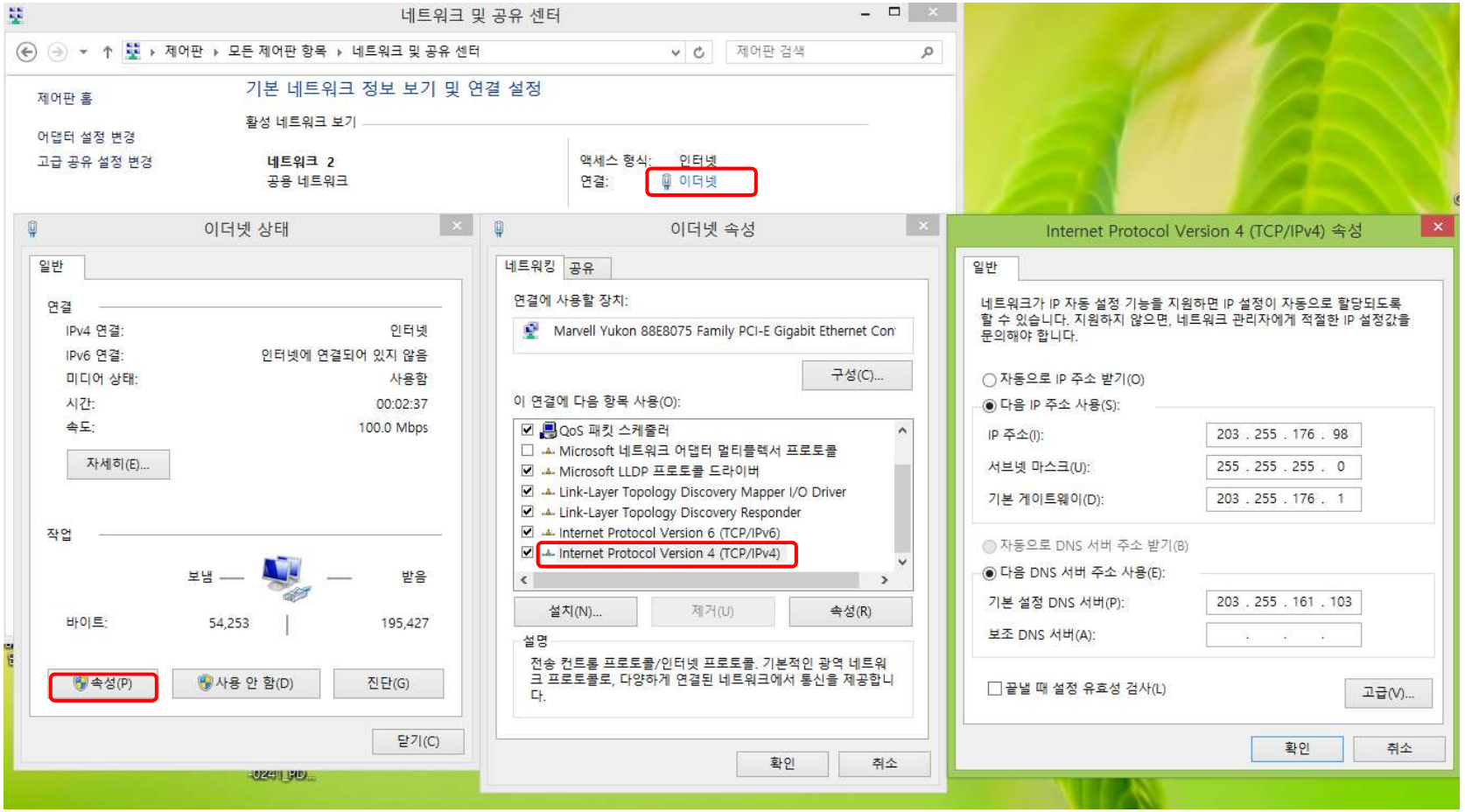
(3) Subnet mask 가 22인 네트워크에서는 _____ 개의 host IP 설정이 가능하다.

$2^{32-22} - 2$

IP addresses: how to get one?

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

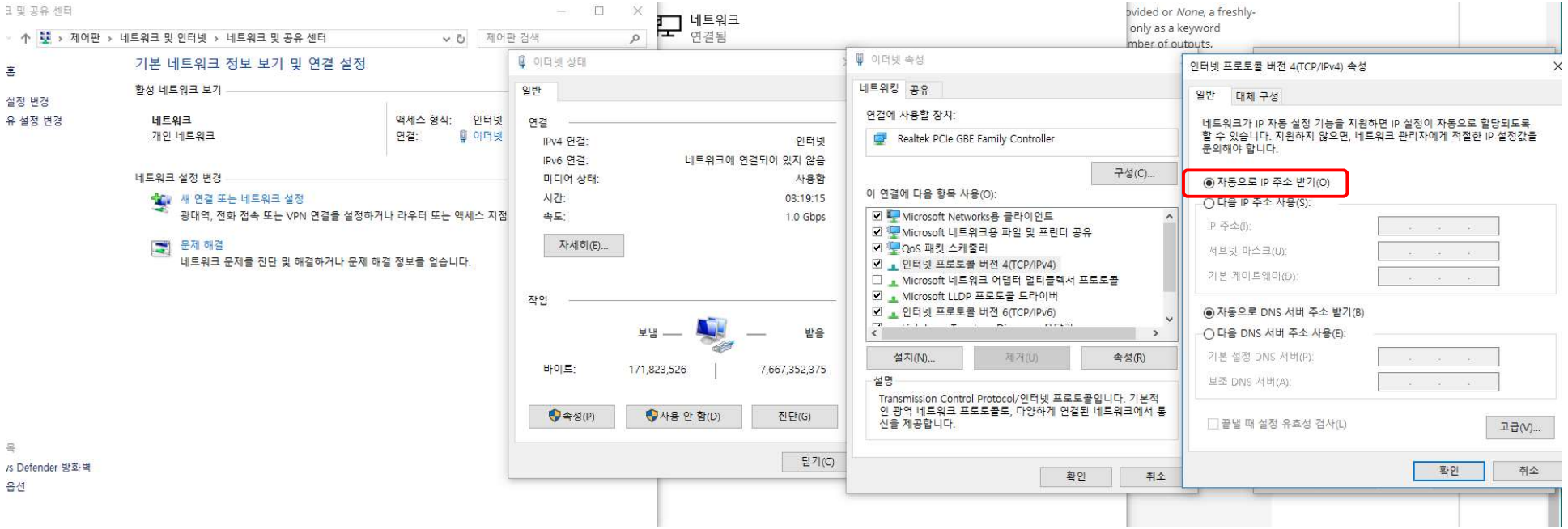
- hard-coded by system admin in a file
  - Windows: control-panel → network → configuration → tcp/ip → properties
  - UNIX: /etc/rc.config



- DHCP: Dynamic Host Configuration Protocol: dynamically get *temporary address* from as server
  - Client-Server protocol
  - "plug-and-play"
  - Service/deamon Runs over UDP (server port : 67)

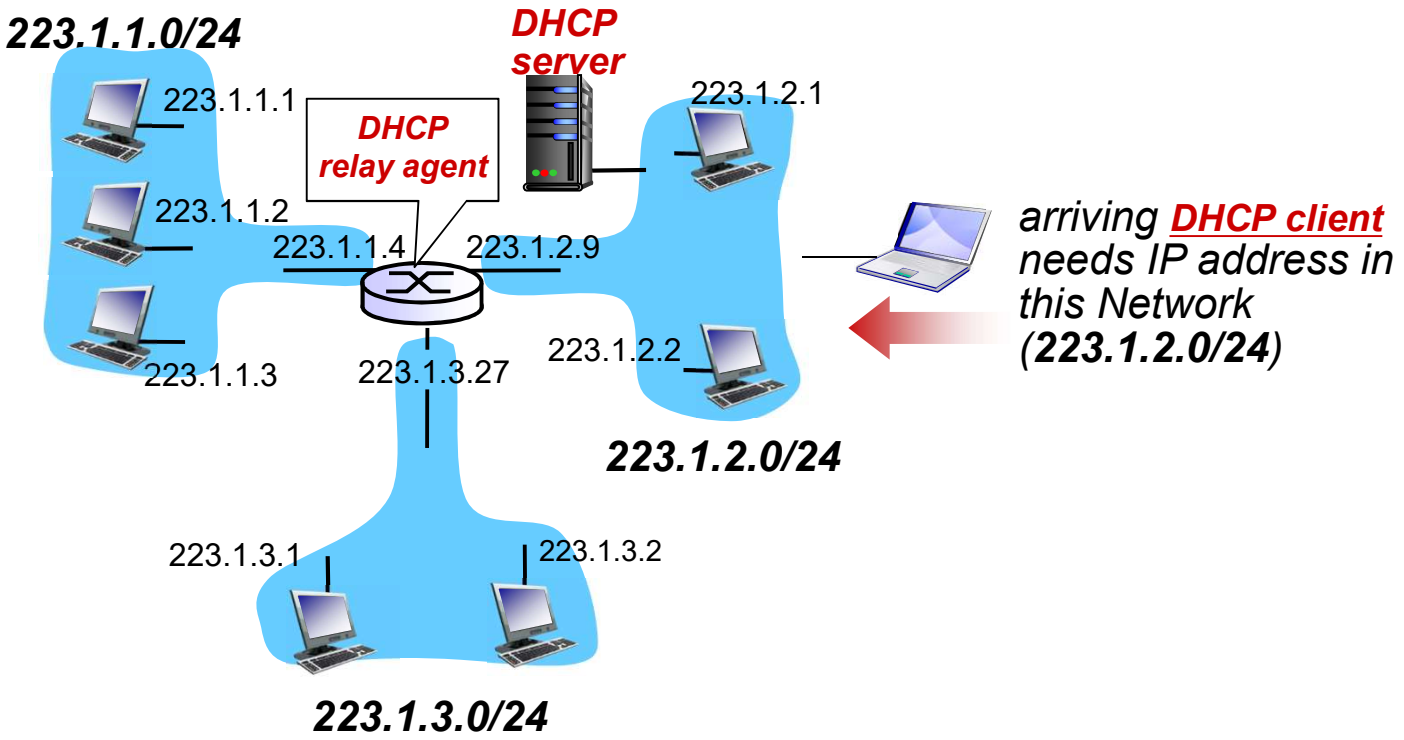
Client also uses a well-known port (68). Why?

Due to "no client IP address & Broadcast"

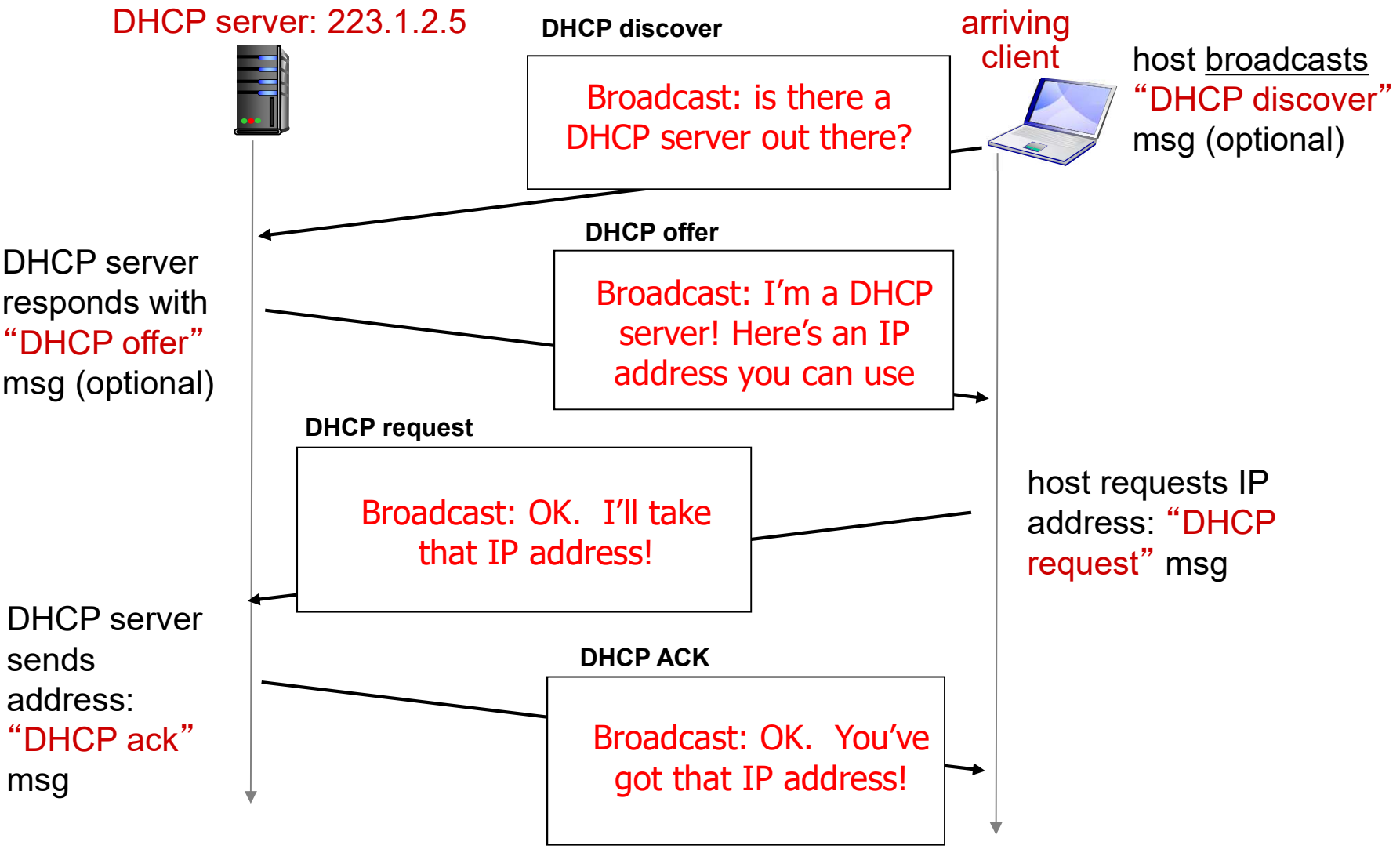


DHCP: Dynamic Host Configuration Protocol

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



DHCP client-server scenario



DHCP: more

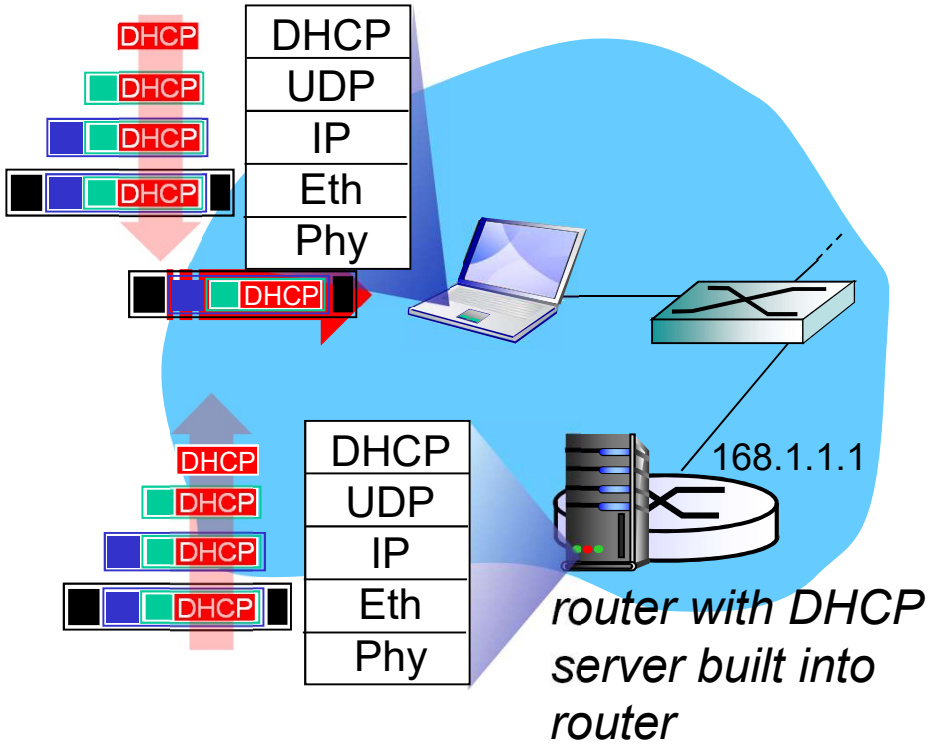
DHCP server ..

- 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 can *return more than just allocated IP address* on subnet:

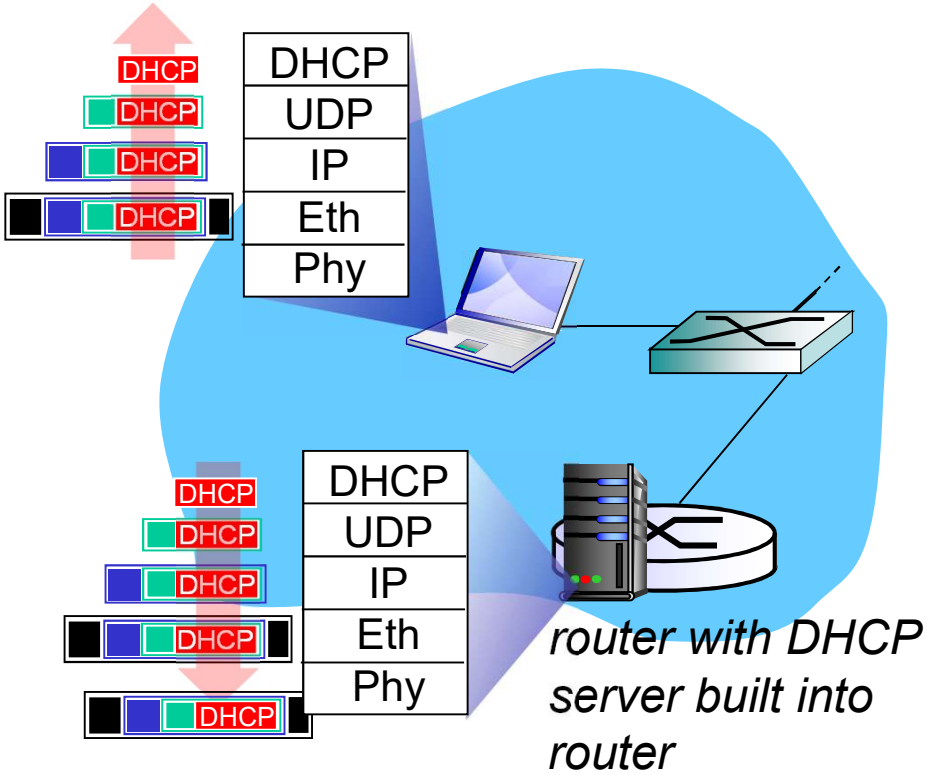
- *network mask* (indicating network versus host portion of address)
- address of *first-hop router (default gateway)* for client
- name and IP address of *DNS sever*

DHCP: example



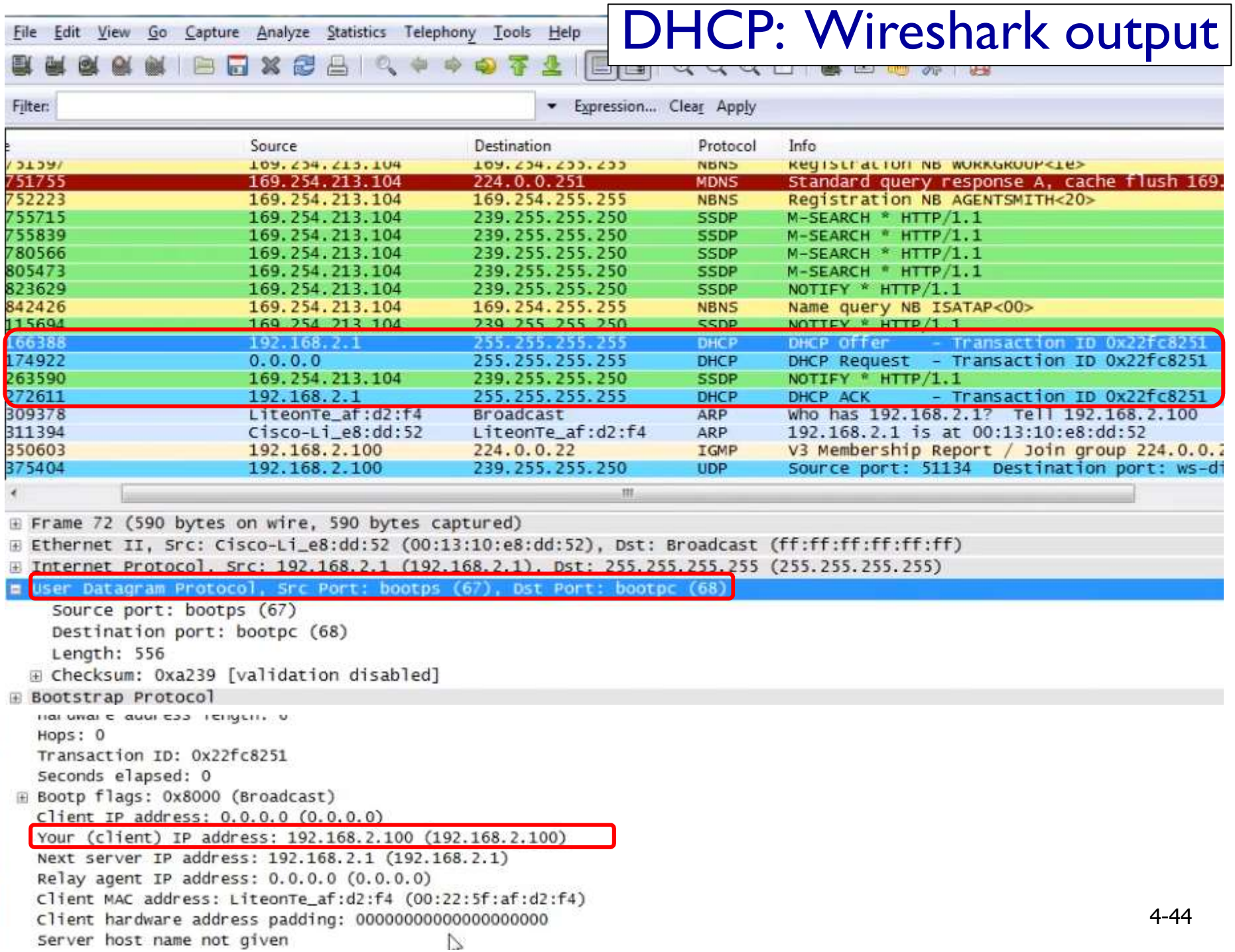
- connecting laptop needs its *IP address*, *addr of first-hop router*, *addr of DNS server*: use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed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

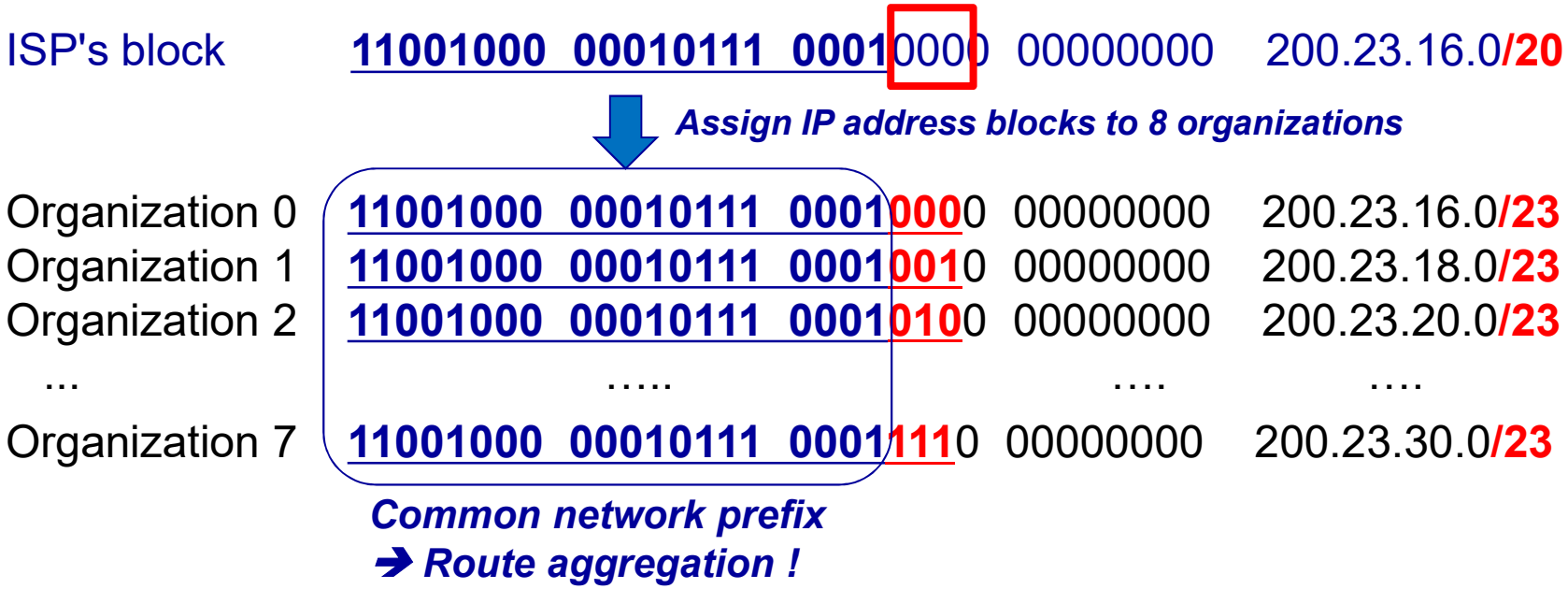
DHCP: Wireshark output



IP addresses: how to get one *subnet part*?

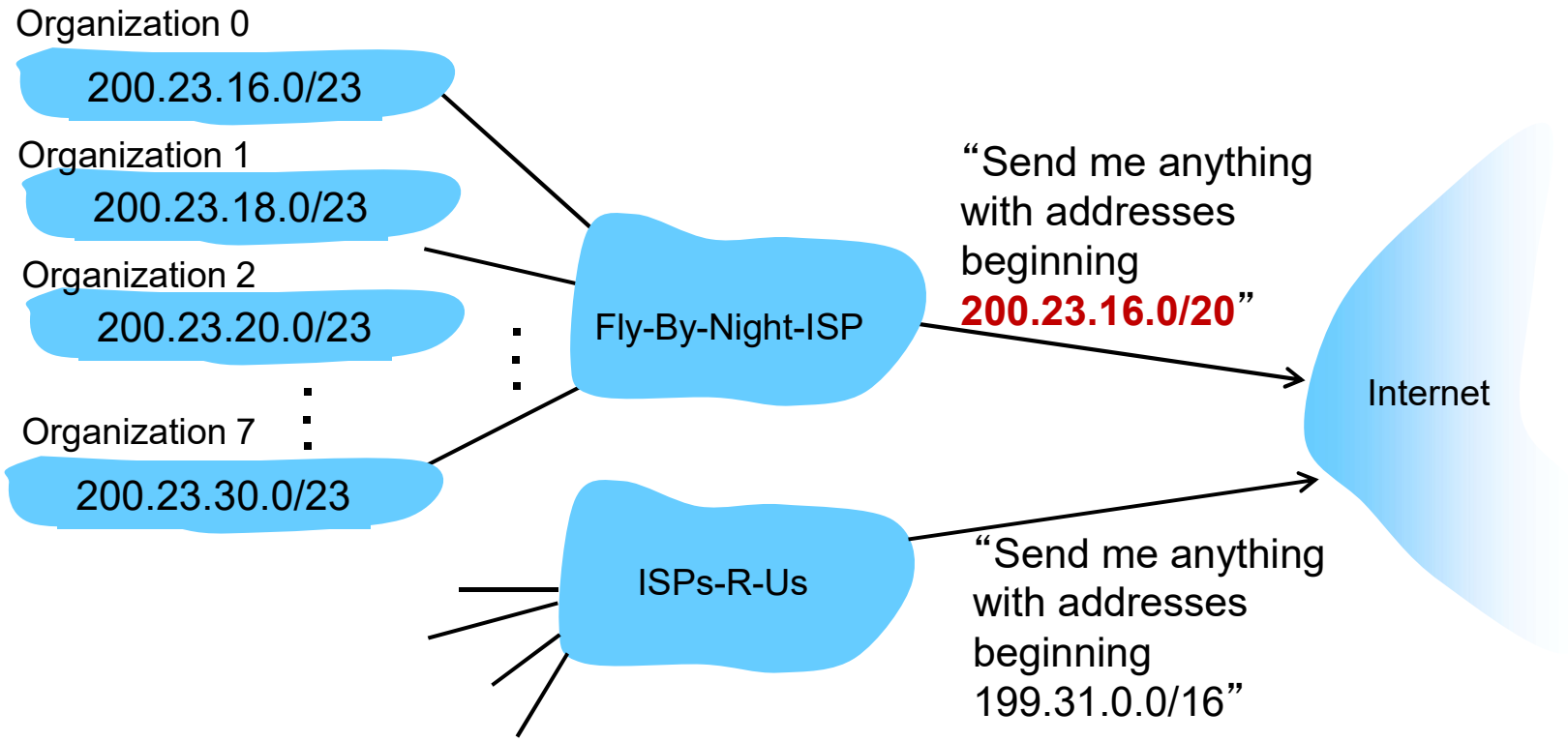
Q: how does *network* get subnet part of IP addr?

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



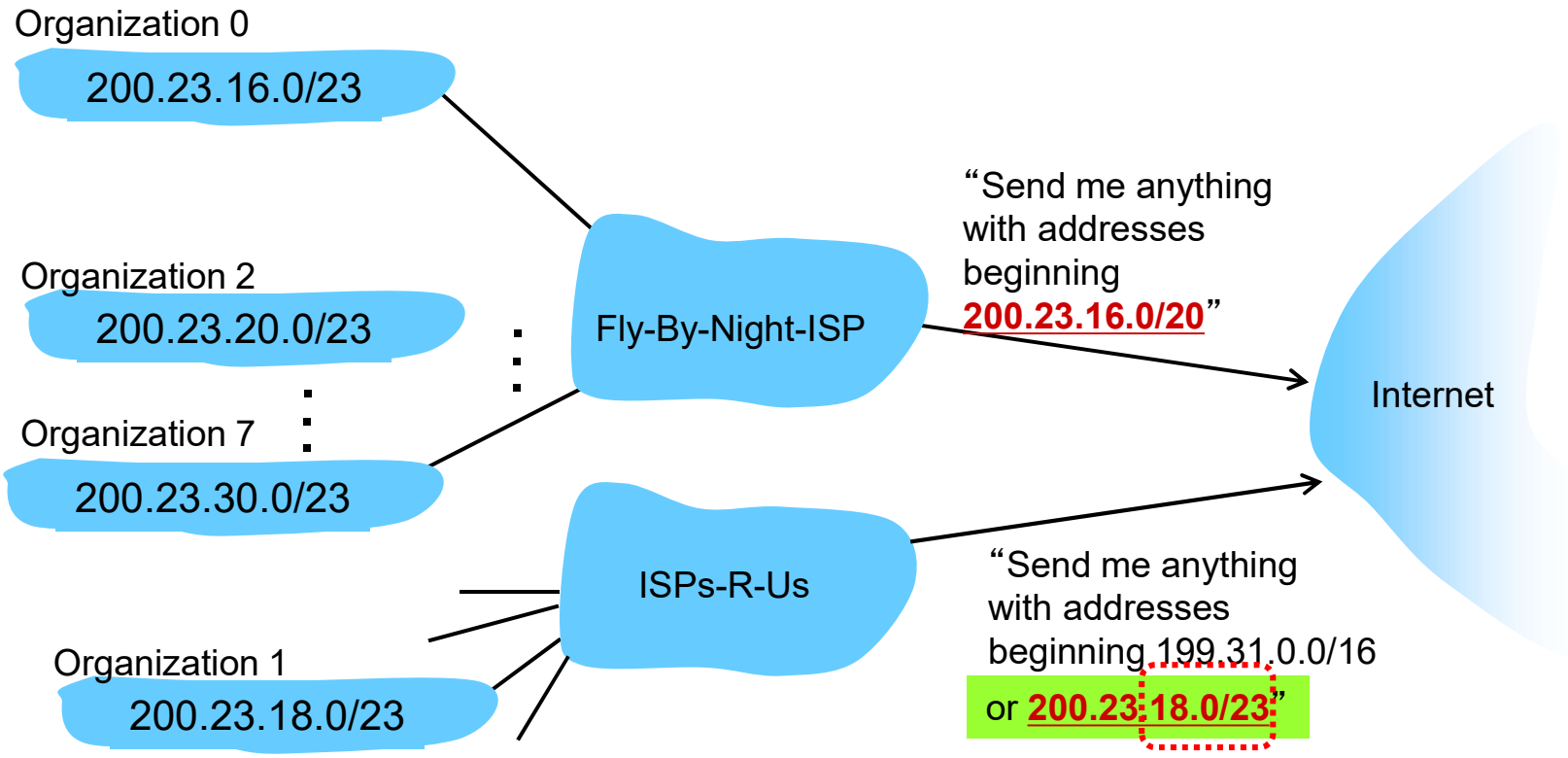
Hierarchical addressing: *route aggregation*

hierarchical addressing allows efficient advertisement of routing information:



Hierarchical addressing: more specific routes

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



Destination-based forwarding

forwarding table (FIB)

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? _____

(3) 20-bit subnet mask 주소 블락(block) 을 할당 받은 ISP가 있다.

이 ISP가 8개의 회사에 주소를 공평하게 나누어 주려고 한다.

각 회사의 subnet mask는 _____ bits 가 되야하는가?

힌트) 8개의 각 다른 네트워크를 표현하려면 몇 bits가 필요한가?

➔ 그렇다면 각 회사는 몇 개의 IPv4 주소를 할당할 수 있는가?

Quiz) Suppose an ISP owns the block of addresses of the form 128.119.40.64/26. Suppose it wants to create four subnets from this block, with each block having the same number of IP addresses. What are the prefixes (of from a.b.c.d/x) for the four subnets?

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* [RFC 2050] *to regional Internet registries*
- *manages DNS root servers*
- assigns domain names, resolves disputes

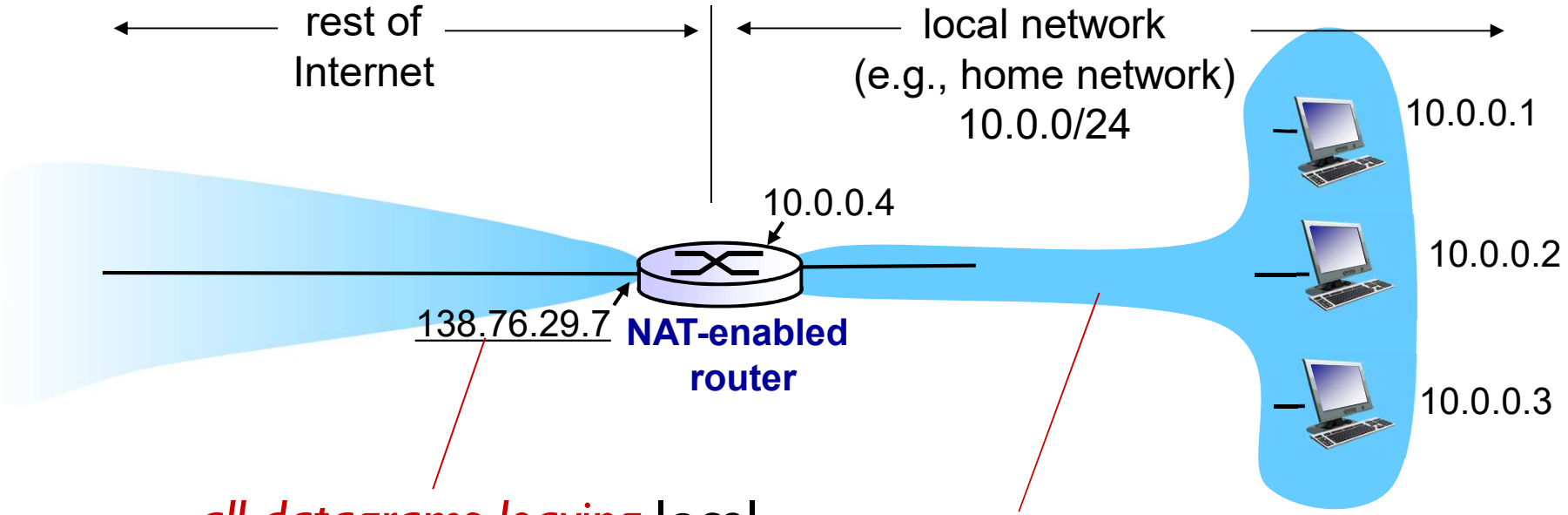
A: 한국인터넷정보센터 (**KRNIC**)

Registry	Area Covered
AFRINIC	Africa Region
APNIC	Asia/Pacific Region
ARIN	Canada, USA, and some Caribbean Islands
LACNIC	Latin America and some Caribbean Islands
RIPE NCC	Europe, the Middle East, and Central Asia

Chapter 4.3
Internet Protocol (IP)

- datagram format & fragmentation
- IPv4 addressing
- Network Address Translation (NAT)
- IPv6

NAT: network address translation



all datagrams leaving local network have *same single source NAT IP address*, 138.76.29.7, and different source port numbers

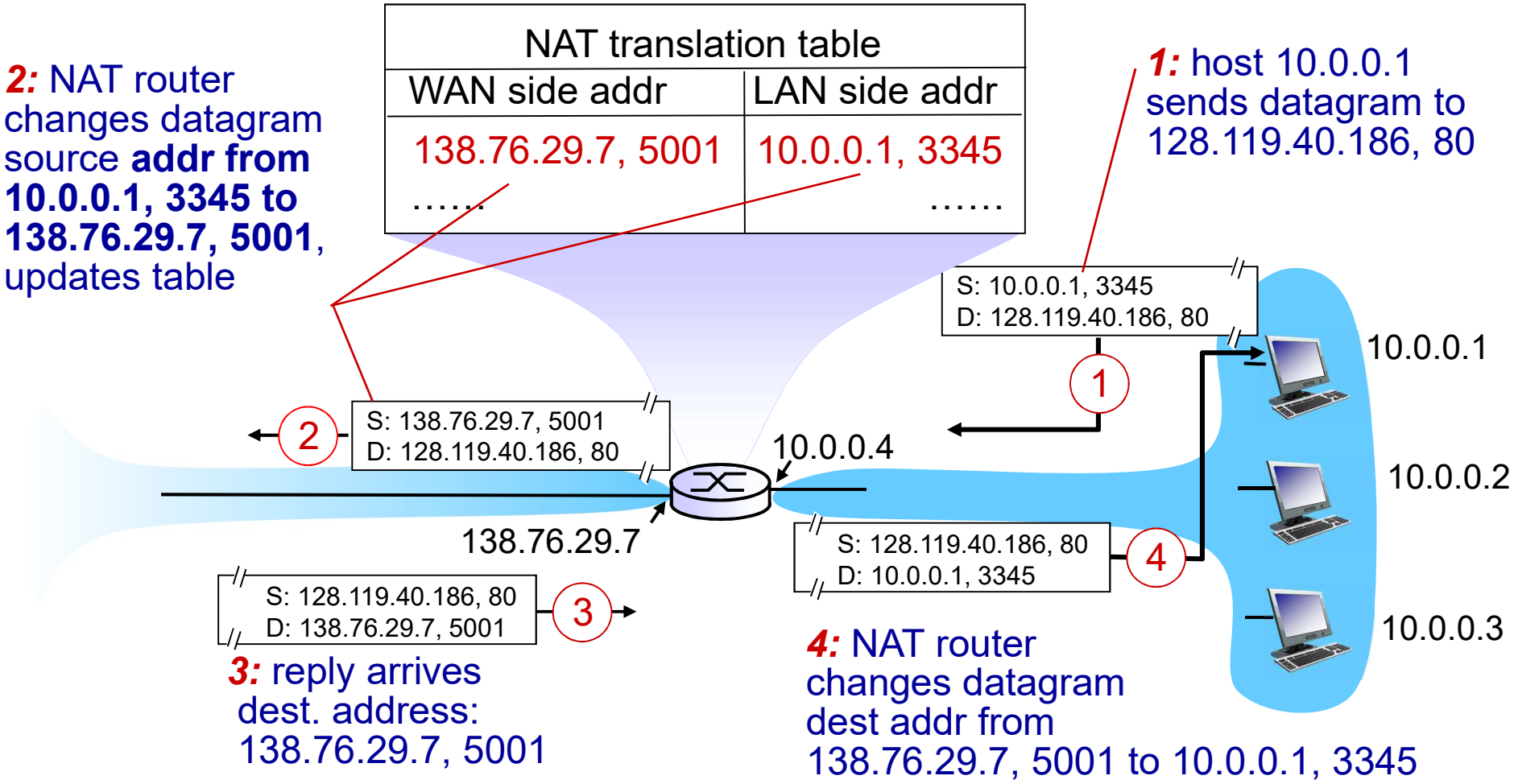
datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

16-bit port-number field: 60,000 simultaneous connections with a single LAN-side address!

Motivation of NAT

- Every IP-capable device needs an IP address.
- Proliferation of small office, home office (SOHO) subnets
- 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



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

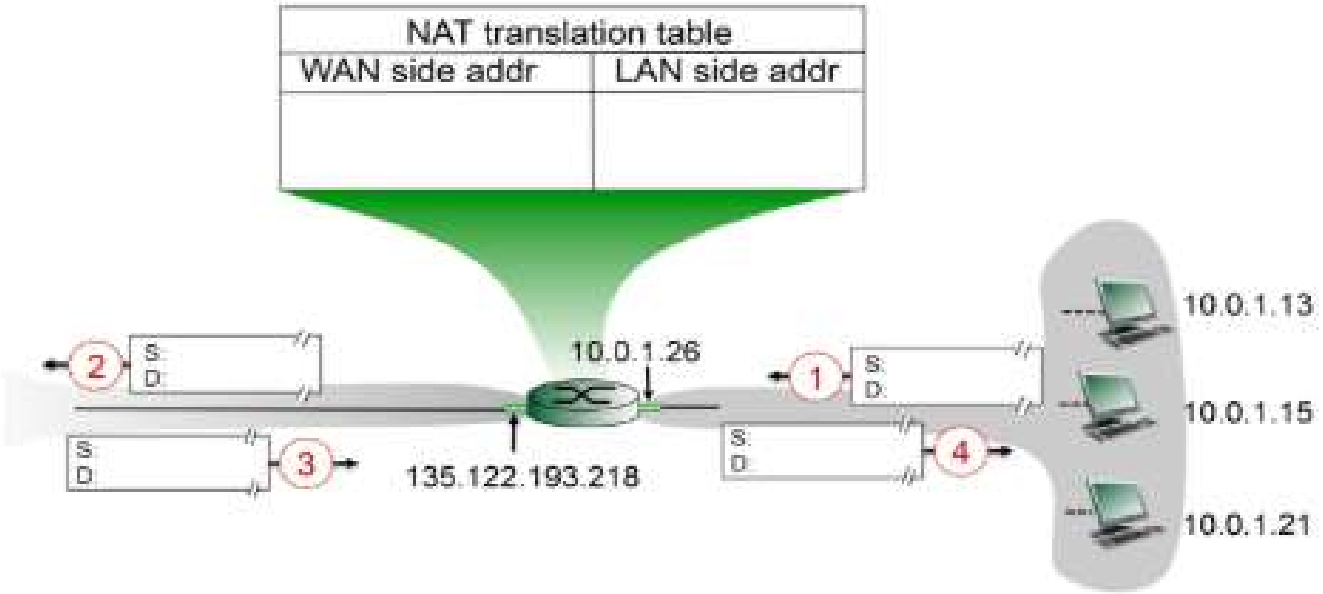
implementation of NAT

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

2) NAT router must *remember every translation pair in NAT translation table ;* (source IP address, port #) to (NAT IP address, new port #)

3) . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr

4) *incoming datagrams:* NAT router must *replace* (NAT IP address, new port #) in *dest* fields of every incoming datagram with corresponding *(source IP address, port #) stored in NAT table*



Suppose that the host with IP address 10.0.1.21 sends an IP datagram destined to host 128.119.168.188. The source port is 3376, and the destination port is 80.

1. Consider the datagram at step 1, after it has been sent by the host but before it has reached the NATted router. What are the source and destination IP addresses for this datagram? What are the source and destination port numbers for the TCP segment in this IP datagram?

2. Now consider the datagram at step 2, after it has been transmitted by the NATted router. What are the source and destination IP addresses for this datagram? What are the source and destination port numbers for the TCP segment in this IP datagram? Identify the differences in datagram's IP addresses and port numbers between step 1 and step 2. Specify the entry that has been made in the router's NAT table.

3. Now consider the datagram at step 3, just before it is received by the NATted router. What are the source and destination IP addresses for this datagram? What are the source and destination port numbers for the TCP segment in this IP datagram?

4. Last, consider the datagram at step 4, after it has been transmitted by the NATted router but before it has been received by the host. What are the source and destination IP address for this datagram? What are the source and destination port numbers for the TCP segment in this IP datagram? Identify the differences in datagram's IP addresses and port numbers between step 3 and step 4. Has a new entry been made in the router's NAT table, or removed from the NAT table? Explain your answer.

NAT is *controversial*

- address shortage should instead be solved by IPv6
- routers should only process up to layer 3
- *NAT violates the principle of end-to-end connectivity* originally envisioned in the design of the Internet.
  - Port numbers are used for addressing processes, not for addressing hosts. NAT uses port # for addressing hosts.

    ⇒ *it causes problems for servers running on the home network*

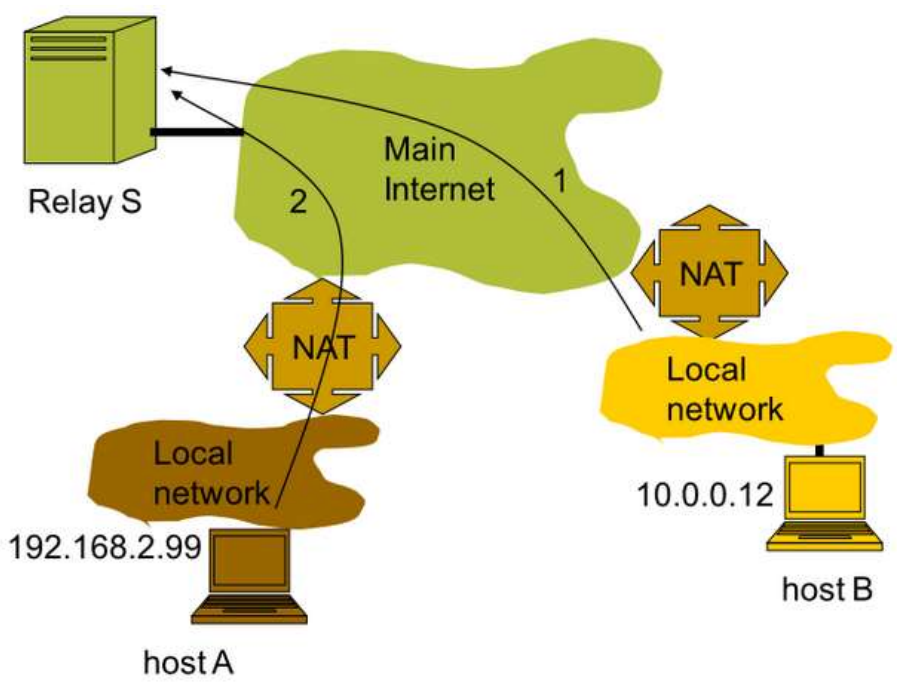
    e.g., P2P applications; a peer behind a NAT cannot act as a server

    ⇒ NAT possibility must be taken into account by app designers,

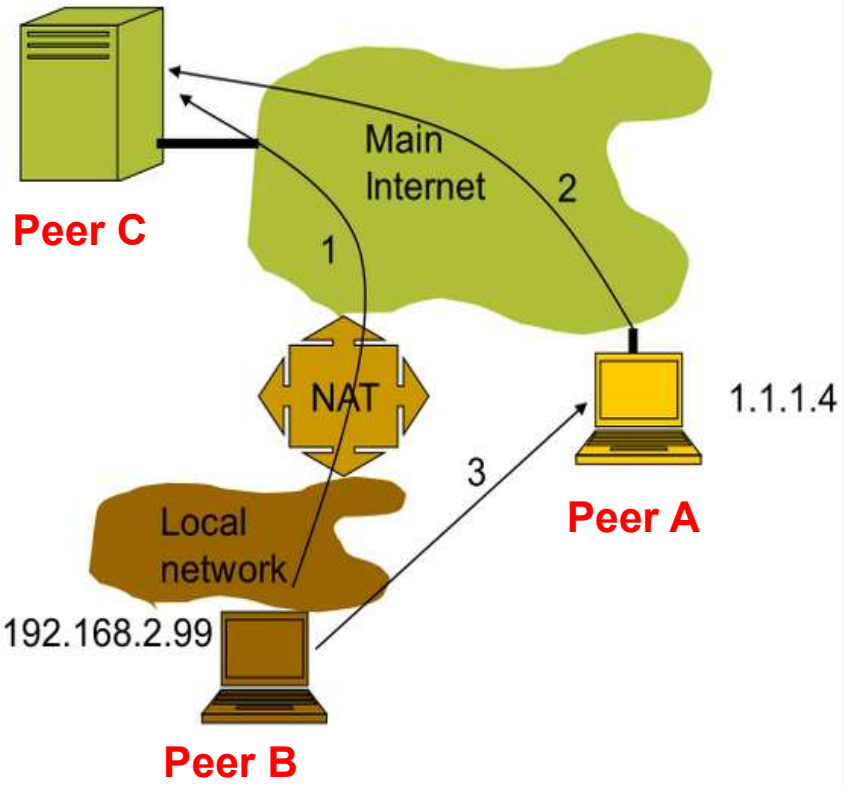
    ⇒ *NAT traversal* techniques

NAT traversal techniques

Relaying



Connection reversal



http://www.keyboardbanger.com/network-address-translation-nat-basics/

Chapter 4.3
Internet Protocol (IP)

- datagram format & fragmentation
- IPv4 addressing
- network address translation (NAT)
- 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*

What ever happened to IPv5?
Where did it go?

IPv6 addressing

※ 128-bit address is represented as 16-bit hexadecimal fields(0~F) separated by colons, i.e., x:x:x:x:x:x:x:x, x = 16 bits.

0010000000000001 0000000000000000 0011001000111000
1101111111100001 0000000001100011 0000000000000000
0000000000000000 1111111011111011
➔
2001:0000:3238:DFE1:0063:0000:0000:FEFB
➔
2001:0000:3238:DFE1:63:0000:0000:FEFB
➔
2001:0000:3238:DFE1:63::FEFB
➔
2001:0:3238:DFE1:63::FEFB

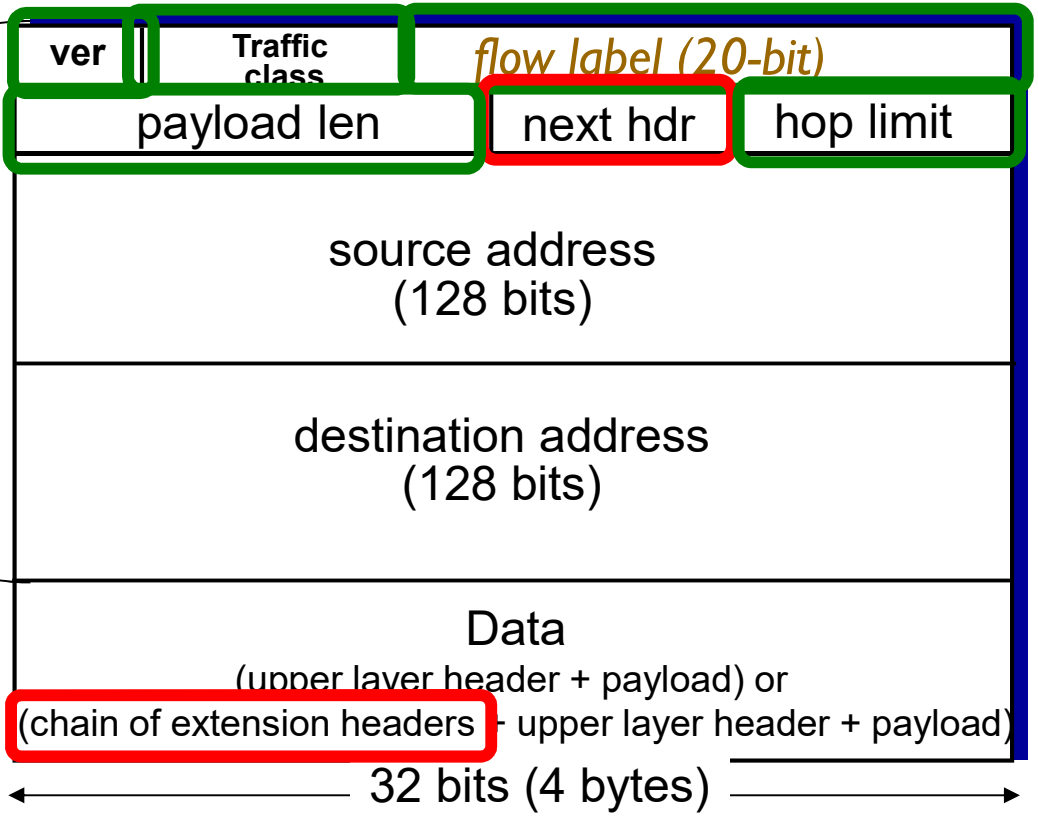
IPv6 datagram format

- ✓ *flow Label:* identify datagrams in same "flow." - flow may be audio/video transmission, but concept of"flow" not well defined.
- ✓ *next header:* identify the protocol to which the data field of this datagram will be delivered.

✓ no fragmentation allowed

at router

✓ fixed-length 40 byte header



Chaining Extension Headers in IPv6 Packets

Ex) TCP/UDP header or ICMPv6/OSPFv6 header

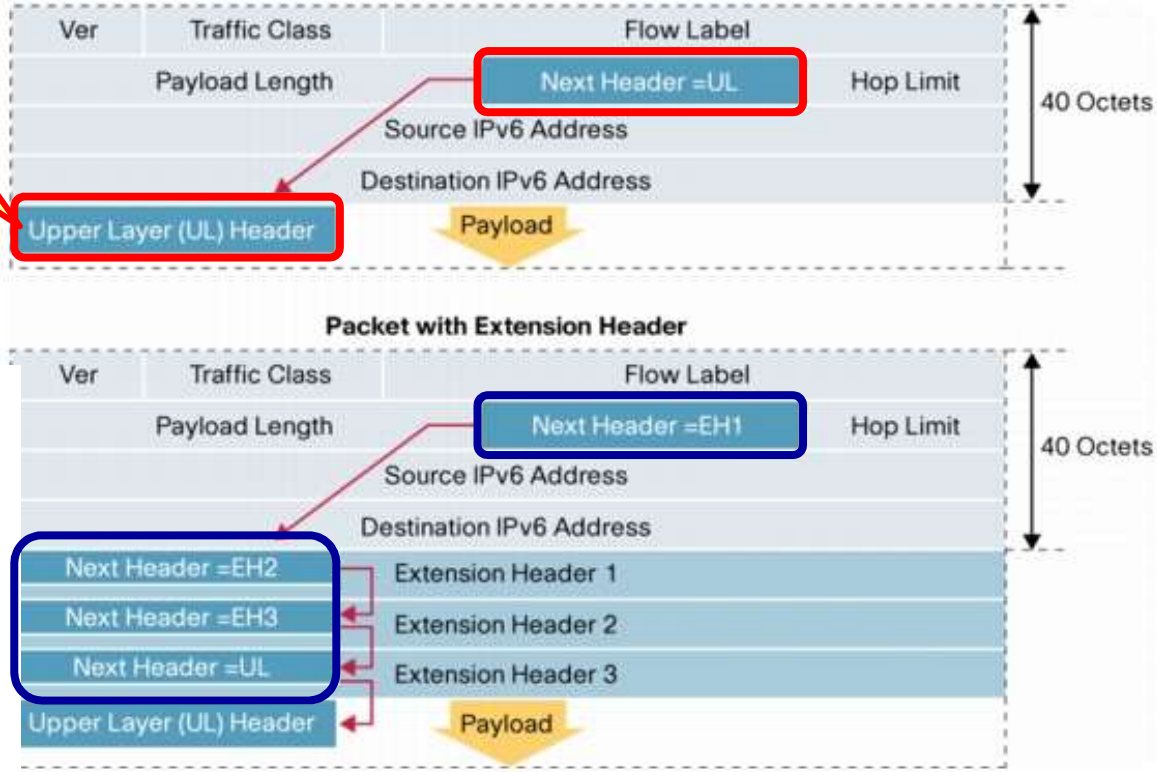


Table 1. IPv6 Extension Headers and their Recommended Order in a Packet

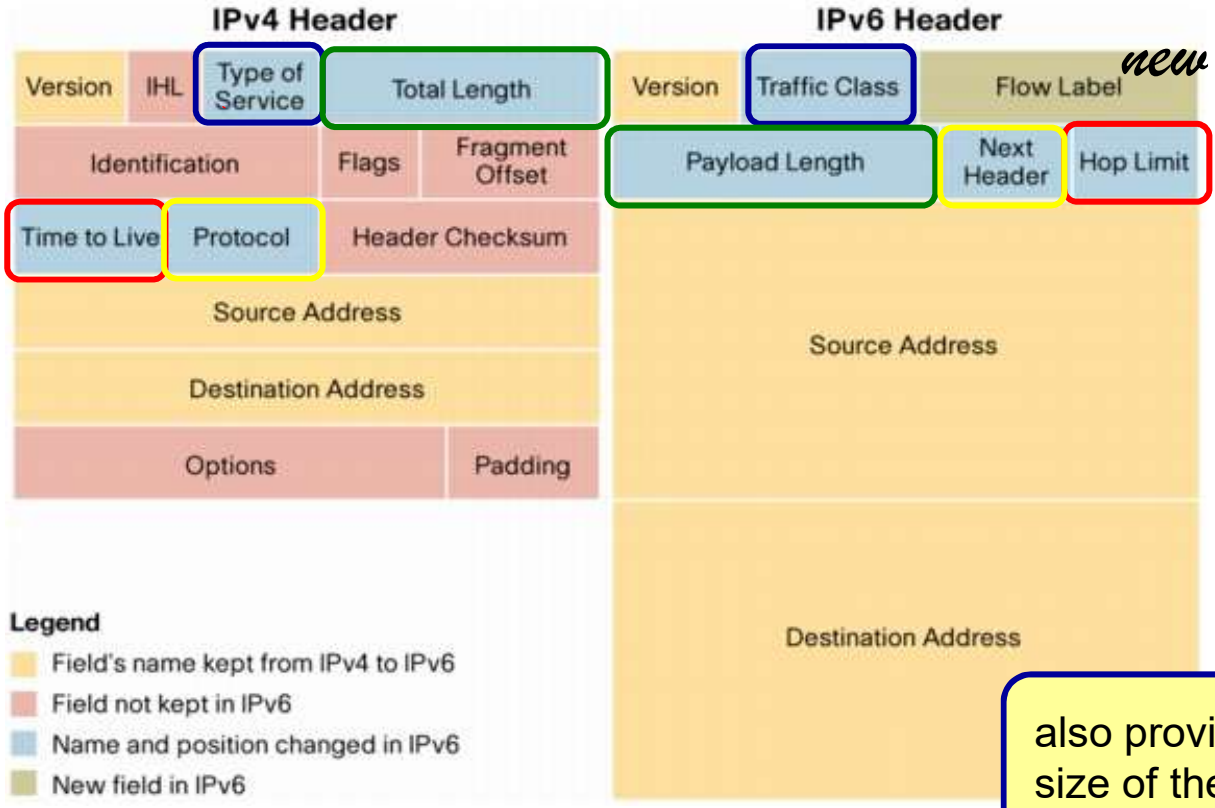
Order	Header Type	Next Header Code
	Basic IPv6 Header	-
2	Hop-by-Hop Options	0
3	Destination Options (with Routing Options)	60
4	Routing Header	43
5	Fragment Header	44
6	Authentication Header	51
7	Encapsulation Security Payload Header	
8	Destination Options	60
9	Mobility Header	135
	No next header	59
Upper Layer	TCP	6
Upper Layer	UDP	17
Upper Layer	ICMPv6	58

RFC2460 also recommends the order in which they should be chained in an IPv6 packet:

While an IPv6 router cannot perform packet fragmentation, the IPv6 sender may fragment an IPv6 packet at the source.

...es_white_paper0900aecd8054d37d.html

Changes from IPv4



also providing the MTU size of the next hop

- *fixed-length* 40 byte *header* (32 bytes for src/dst addresses)
- *no fragmentation allowed* ***at router*** - "*Packet Too Big*" *ICMPv6 msg*
- *no header checksum*: removed entirely to reduce processing time at each hop
- *options:* allowed as *extension headers*, indicated by "Next Header" field

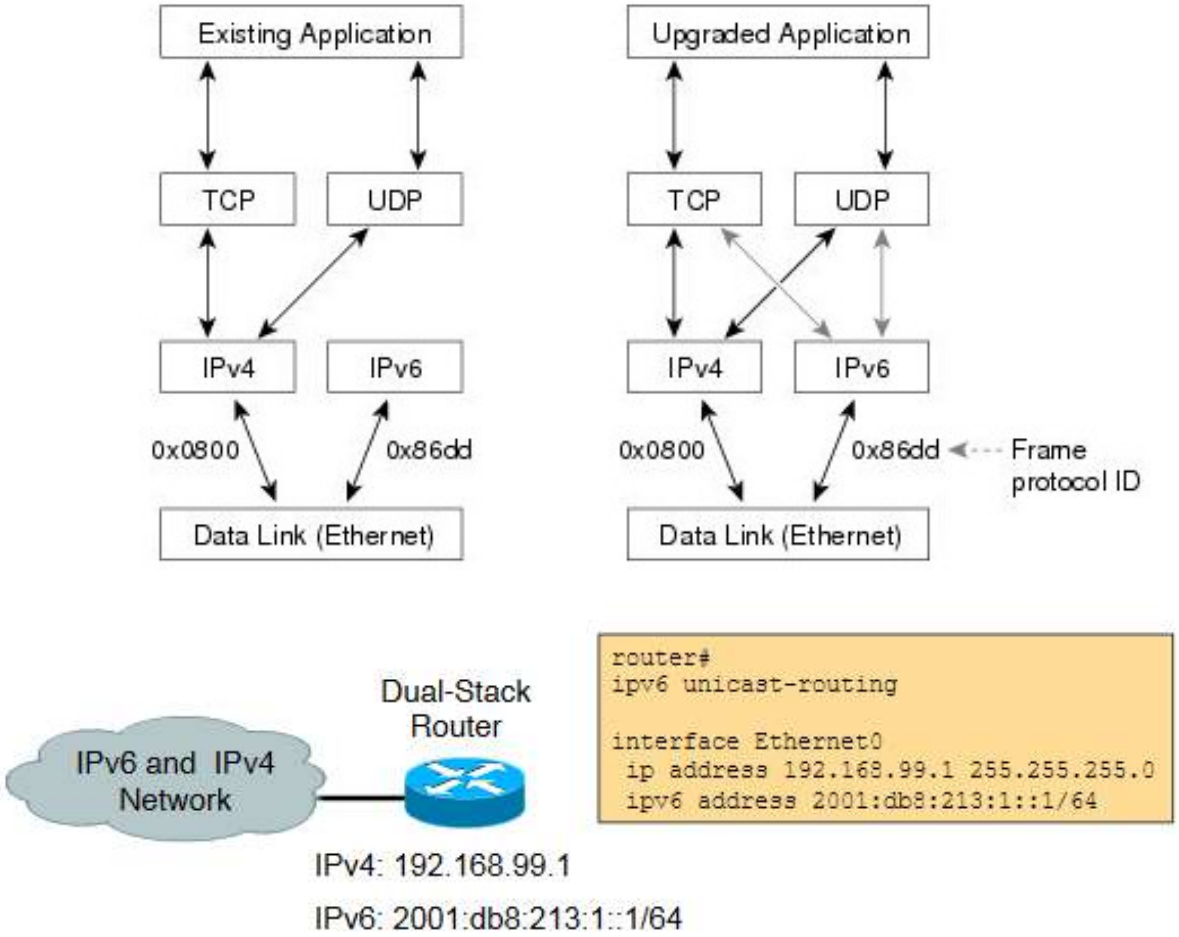
http://www.cisco.com/en/US/technologies/tk648/tk872/technologies_white_paper0900aecd8054d37d.html

Transition from IPv4 to IPv6

- not all routers can be upgraded simultaneously
  - no "flag days"
  - how will network operate with mixed IPv4 & IPv6 routers?

1. **Dual Stack** IPv4/IPv6
2. NAT-PT (NAT-protocol **translation**)
3. **Tunneling**

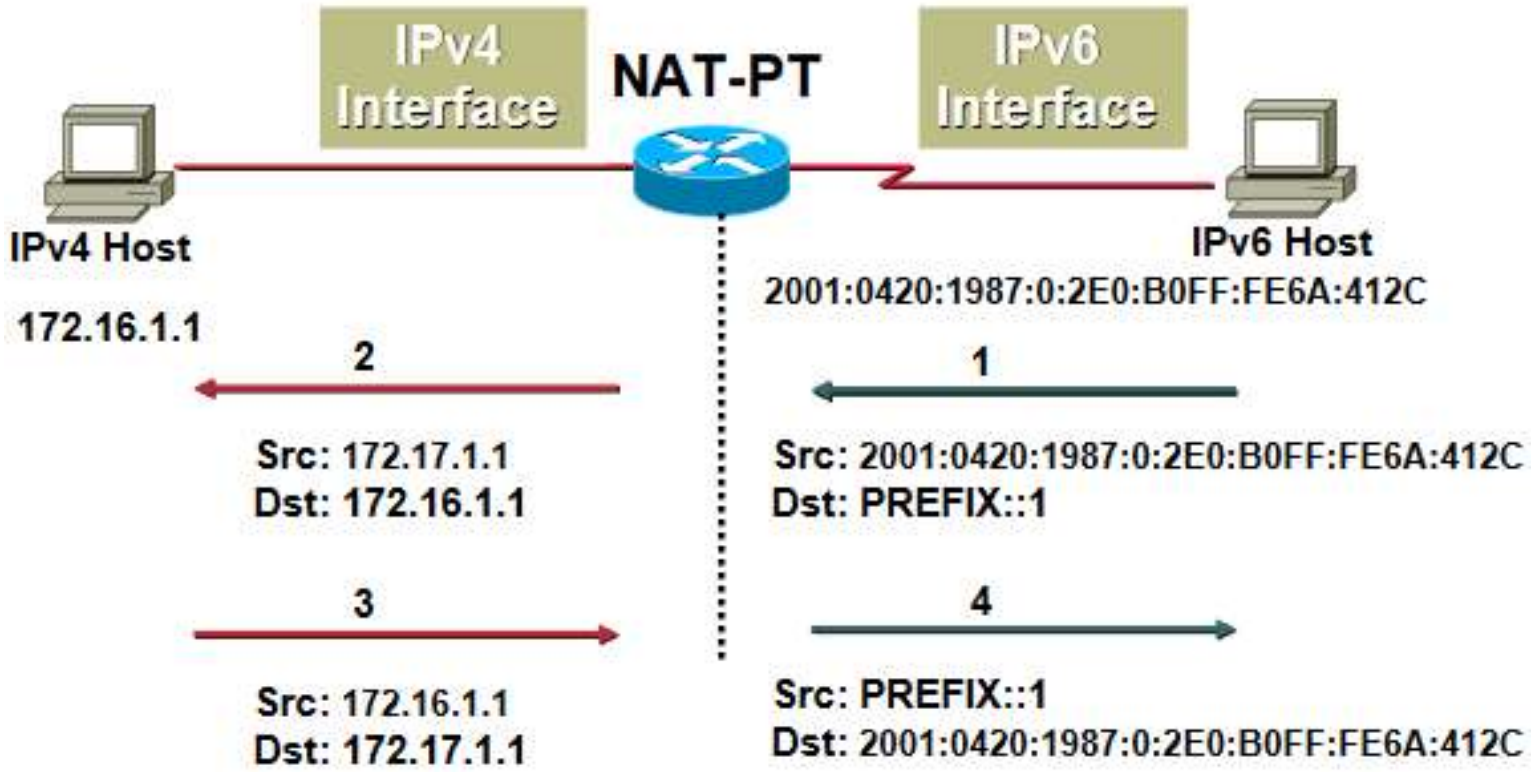
Dual-stack : routers have both IPv4/IPv6 addresses on one interface.



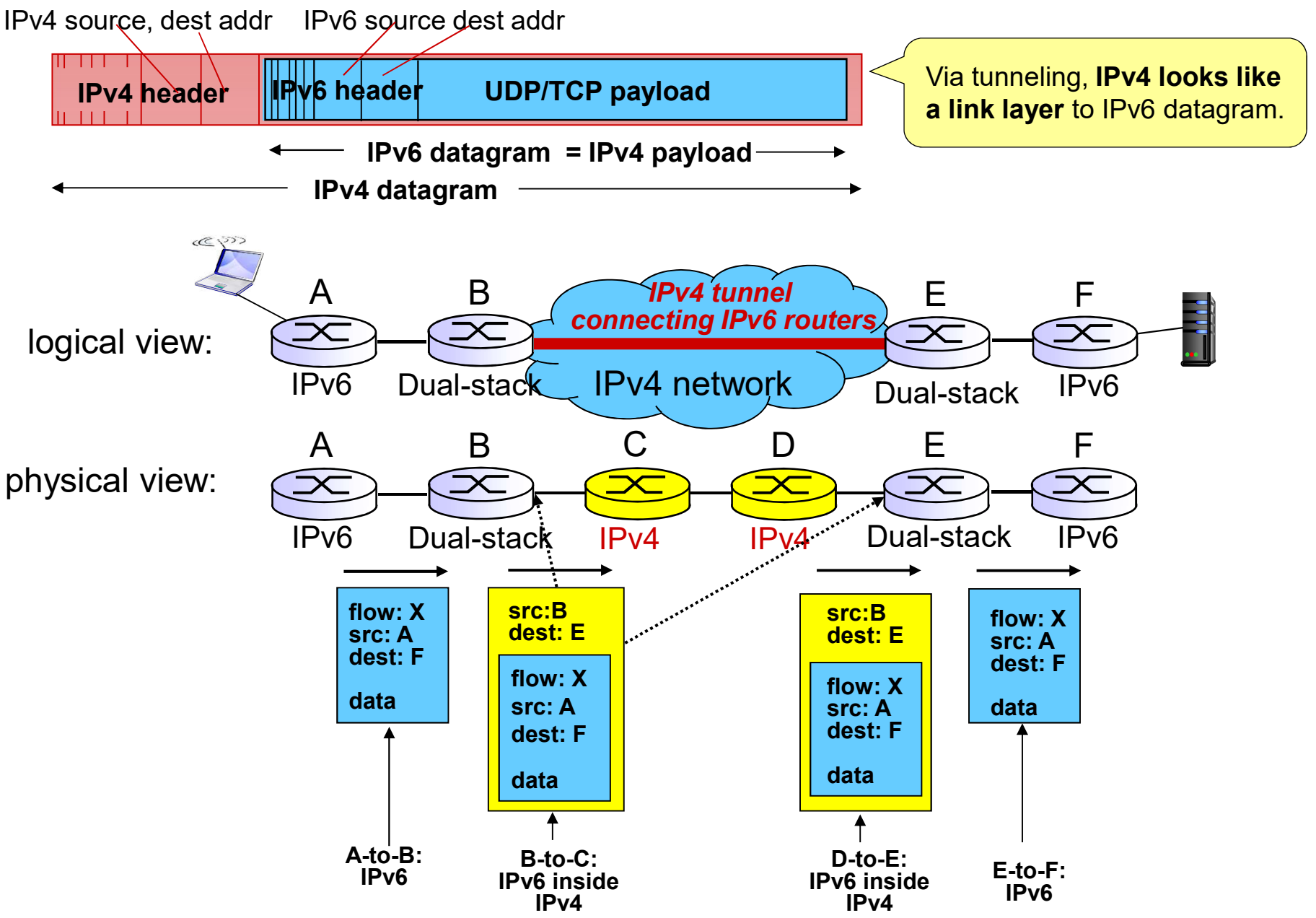
※ Telnet, Ping, Traceroute, SSH, DNS, TFTP are also modified considering IPv6 address.

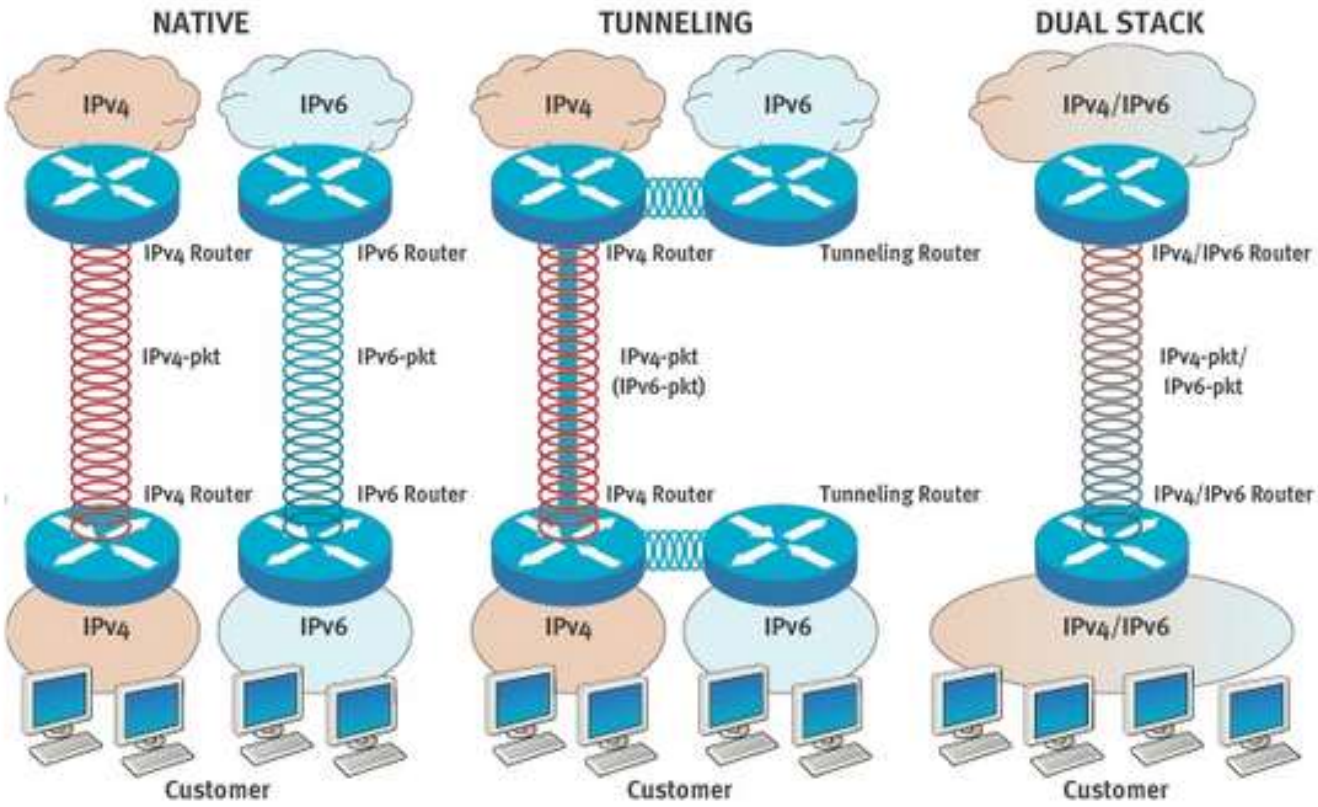
http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/configuration/15-2mt/ipv6-15-2mt-book/ip6-addrg-bsc-con.html

NAT-PT (NAT-Protocol Translation) : protocol translation between IPv4 and IPv6 as well as addresses



Tunneling: IPv6 datagram carried as payload in IPv4 datagram





https://orhanergun.net/2016/10/5153/
http://slideplayer.com/slide/4794330/

IPv6: adoption

- Google: 8% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable

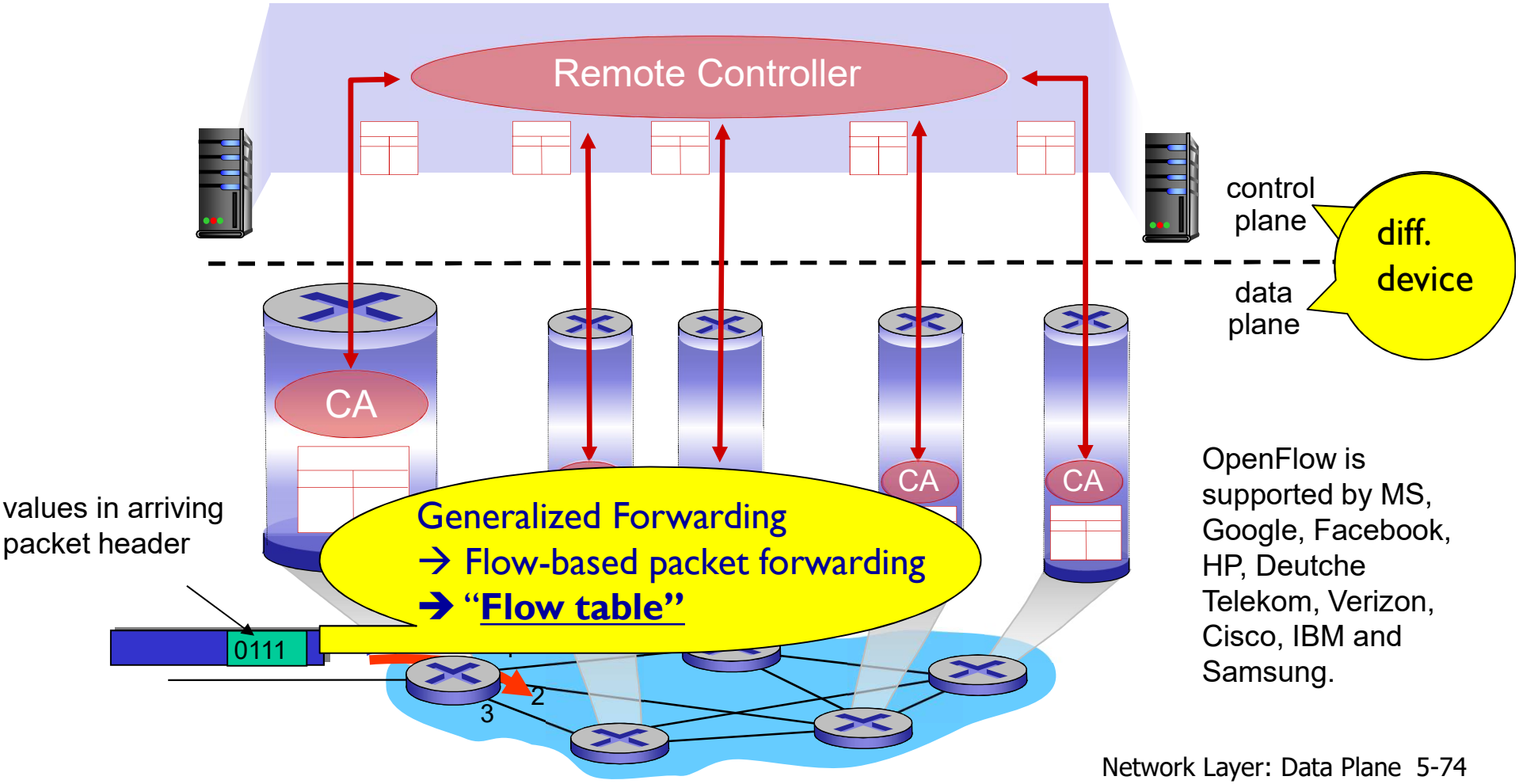
- *Long (long!) time for deployment, use*
  - 20 years and counting!
  - think of application-level changes in last 20 years: WWW, Facebook, streaming media, Skype, …
  - *Why?*

4.4 Generalized Forward and SDN

- Software-Defined Networking (SDN)
- match-plus-action

Logically centralized control plane

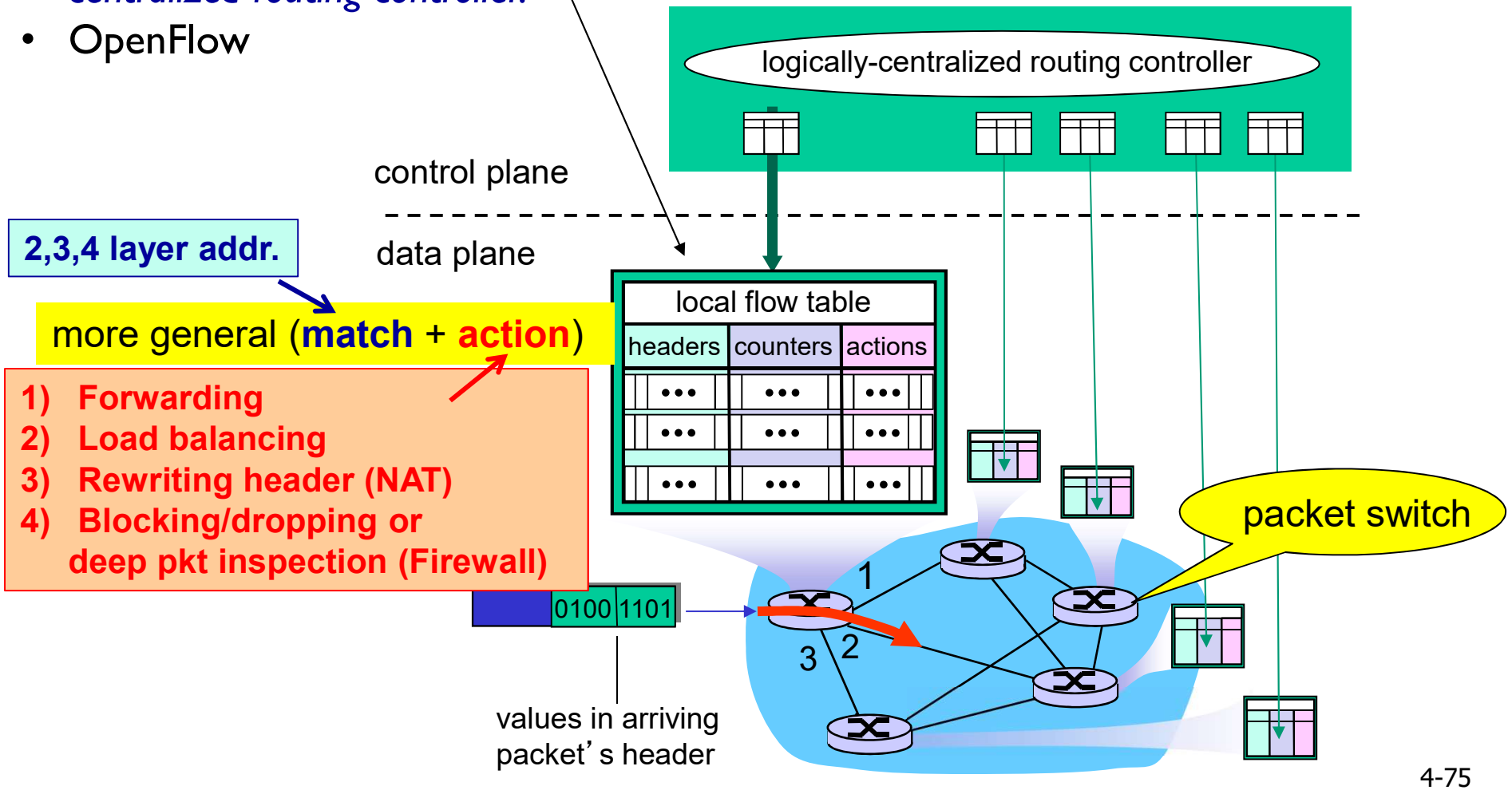
A distinct (typically remote) controller interacts with local control agents (CAs)



OpenFlow is supported by MS, Google, Facebook, HP, Deutche Telekom, Verizon, Cisco, IBM and Samsung.

Software-Defined Networking (SDN)

- unified approach providing middleboxes and link-layer functions in an integrated manner. ➔ *Network-wide behavior can be programmed!!*
- Each router contains a *flow table that is computed and distributed by a logically centralized routing controller.*
- OpenFlow



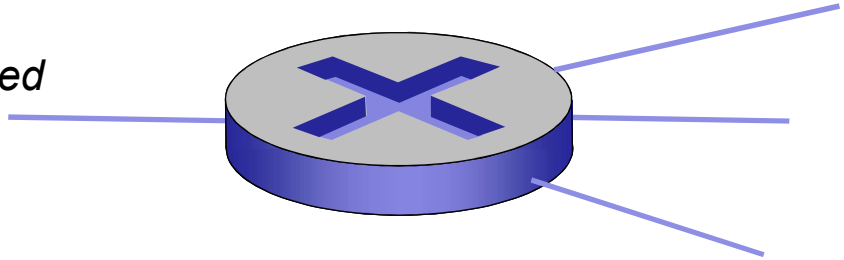
OpenFlow data plane abstraction

- *flow*: defined by header fields

A group of packets which share certain properties for delivery ("***Match***").

- 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

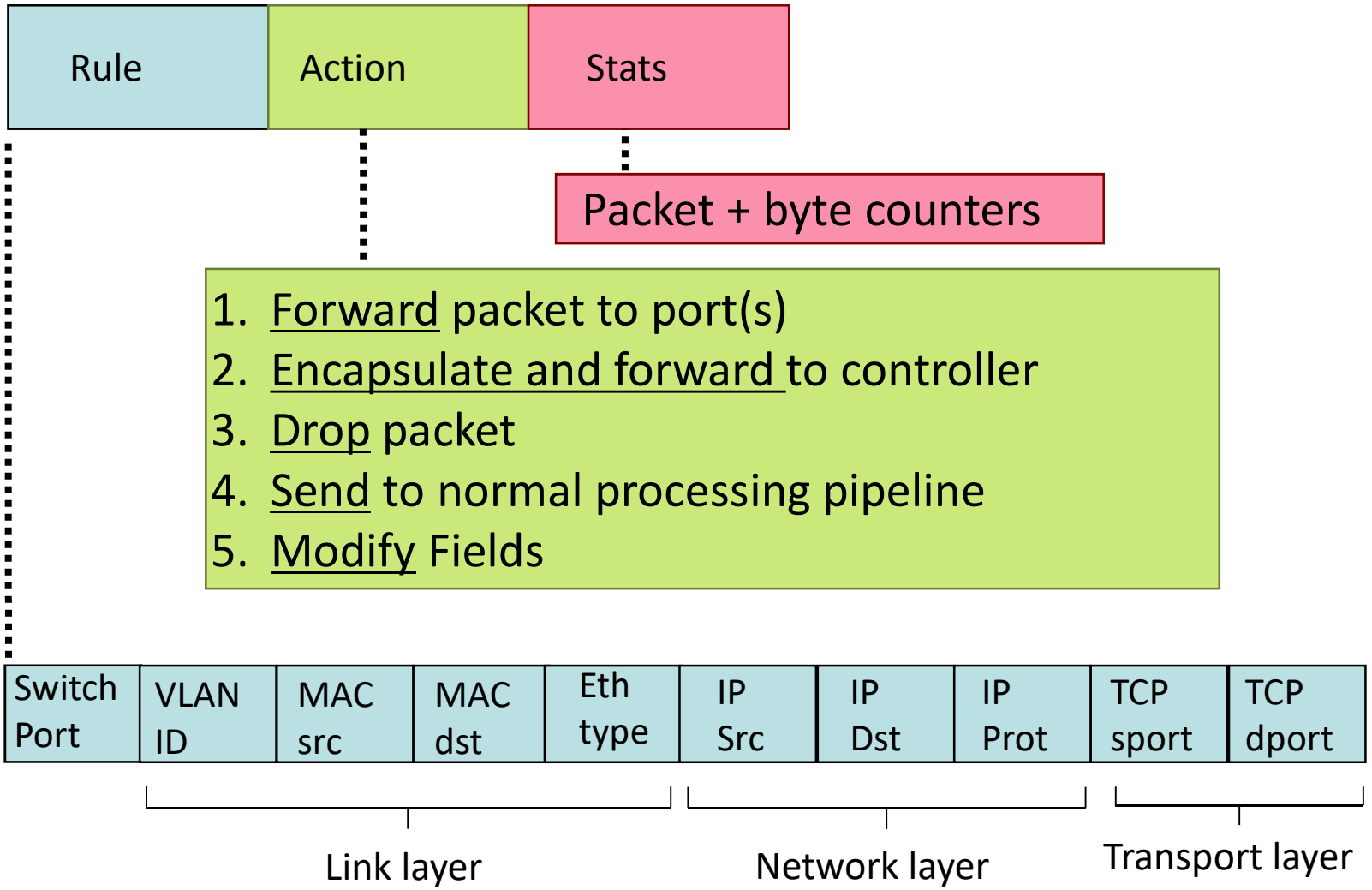
*Flow table in a router (computed and distributed by controller) define router's **match+action** rules*



1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → send to controller

* : wildcard

OpenFlow: Flow Table Entries



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

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

Examples

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

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 3

Examples

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

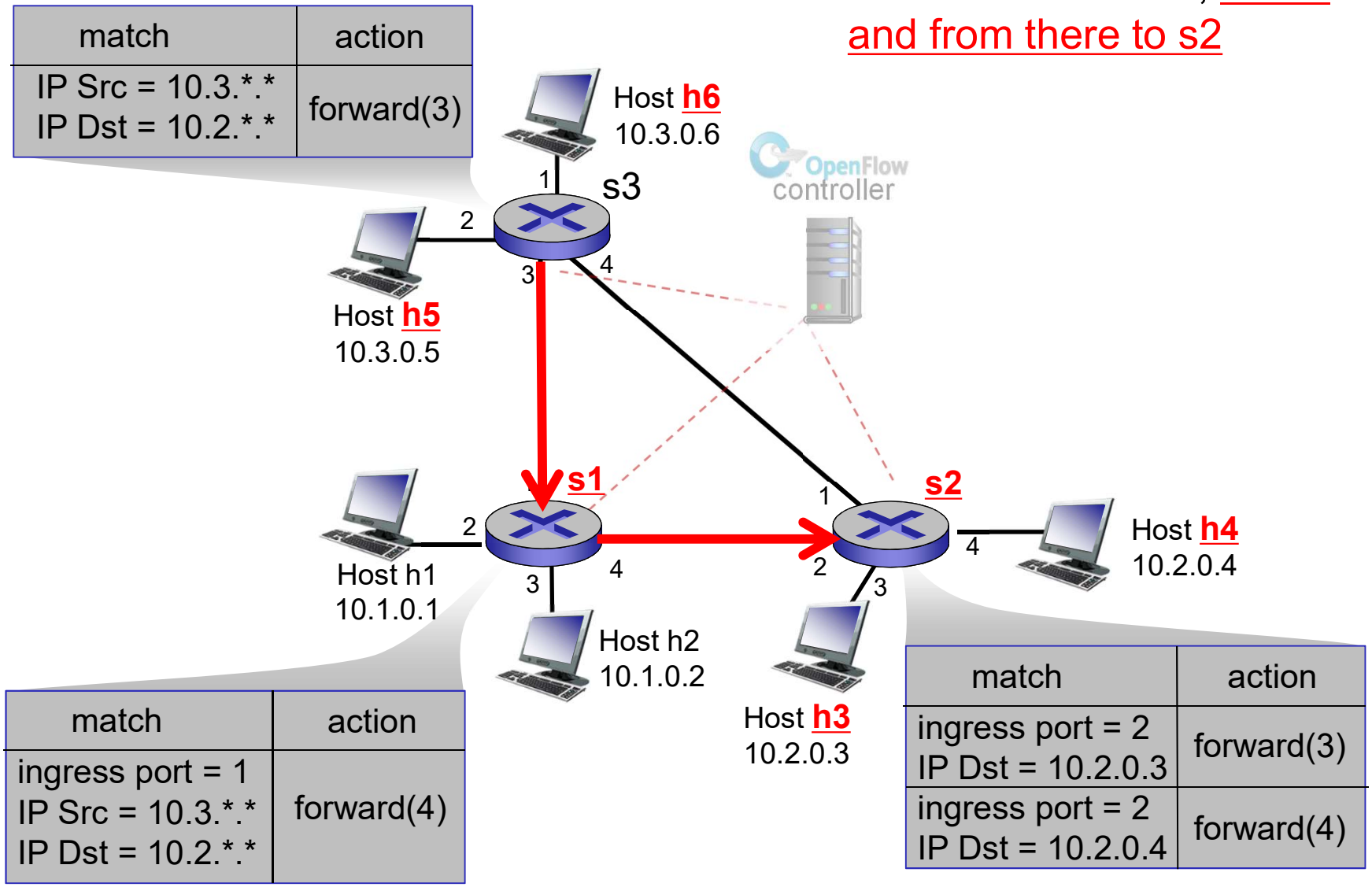
Firewall:

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**

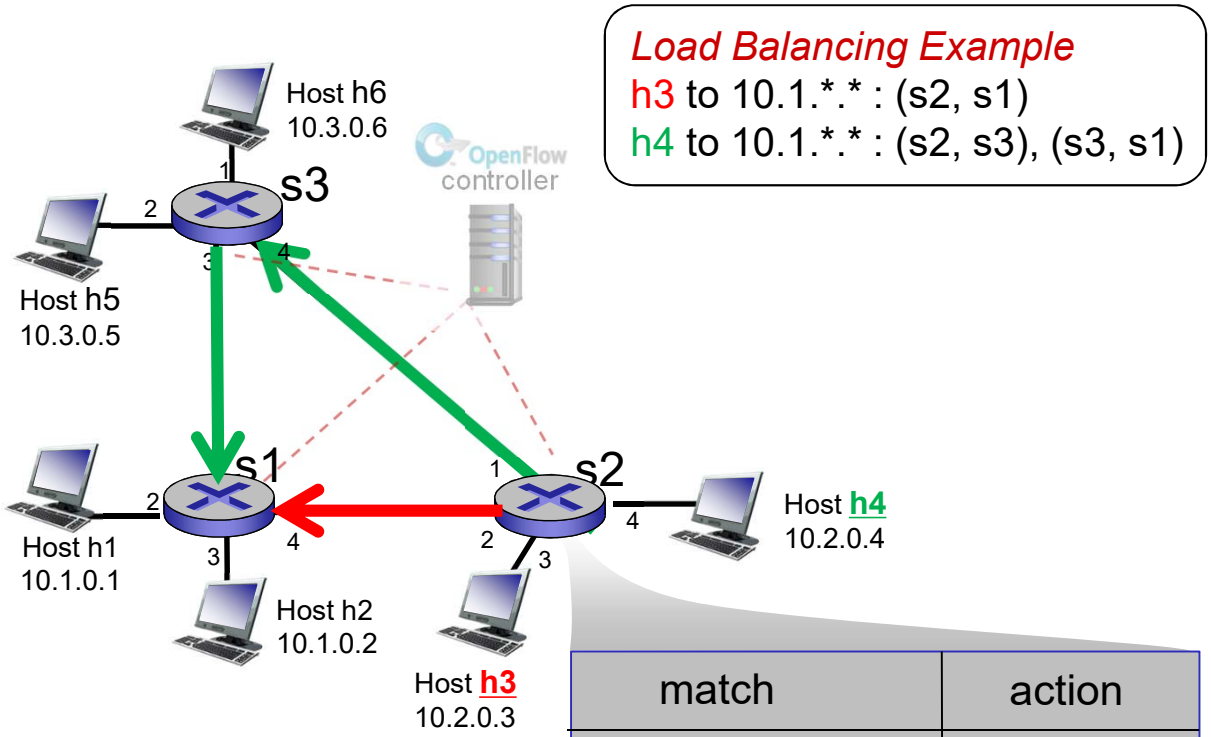
do not forward (block) all datagrams sent by host 128.119.1.1

OpenFlow example

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

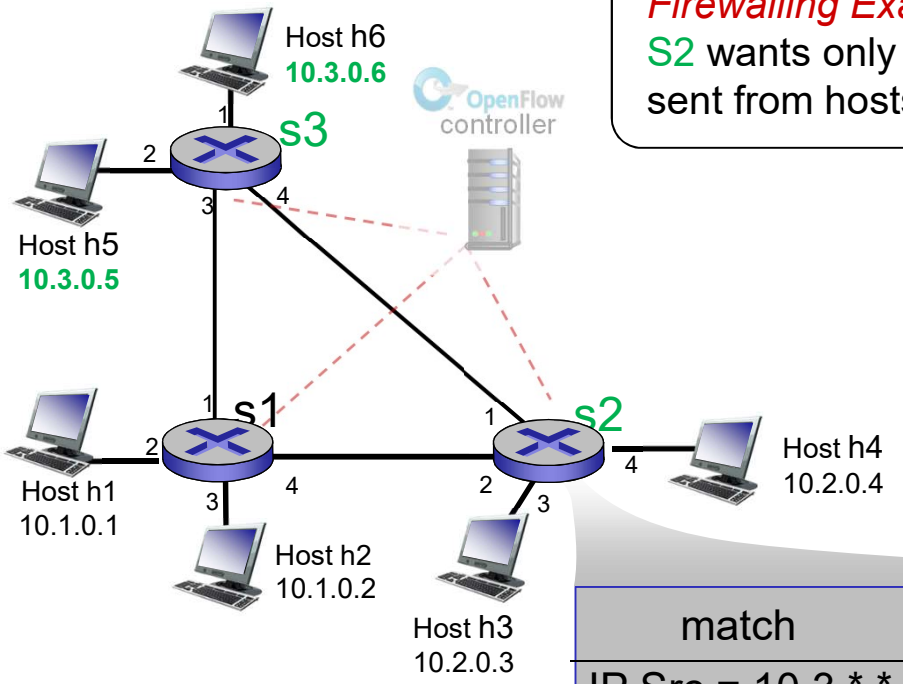


OpenFlow example



match	action
ingress port = 3 IP Dst = 10.1.*.*	forward(2)
ingress port = 4 IP Dst = 10.1.*.*	forward(1)

OpenFlow example



Firewalling Example
S2 wants only to receive traffic sent from hosts attached to s3

match	action
IP Src = 10.3.*.* IP Dst = 10.2.0.3	forward(3)
IP Src = 10.3.*.* IP Dst = 10.2.0.4	forward(4)

Chapter 4: *done!*

Question:

How do forwarding tables (destination-based forwarding) or flow tables (generalized forwarding) *computed*?

Answer:

by the *control plane* (chapter 5!)