CS 305: Computer Networks

Fall 2024

Lecture 10：Network Layer – The Data Plane

Tianyue Zheng

Department of Computer Science and Engineering
Southern University of Science and Technology (SUSTech)

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

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

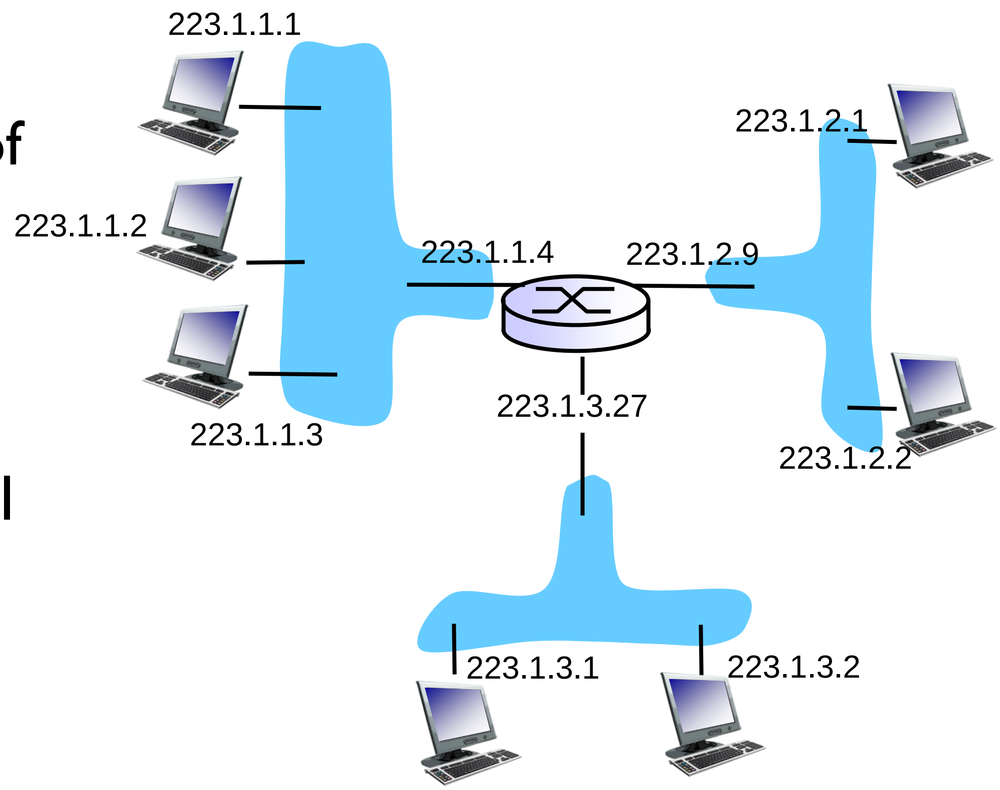
4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

Overview

- IP addressing
- Subnet
- How to assign/obtain IP address?

IP addressing: introduction

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

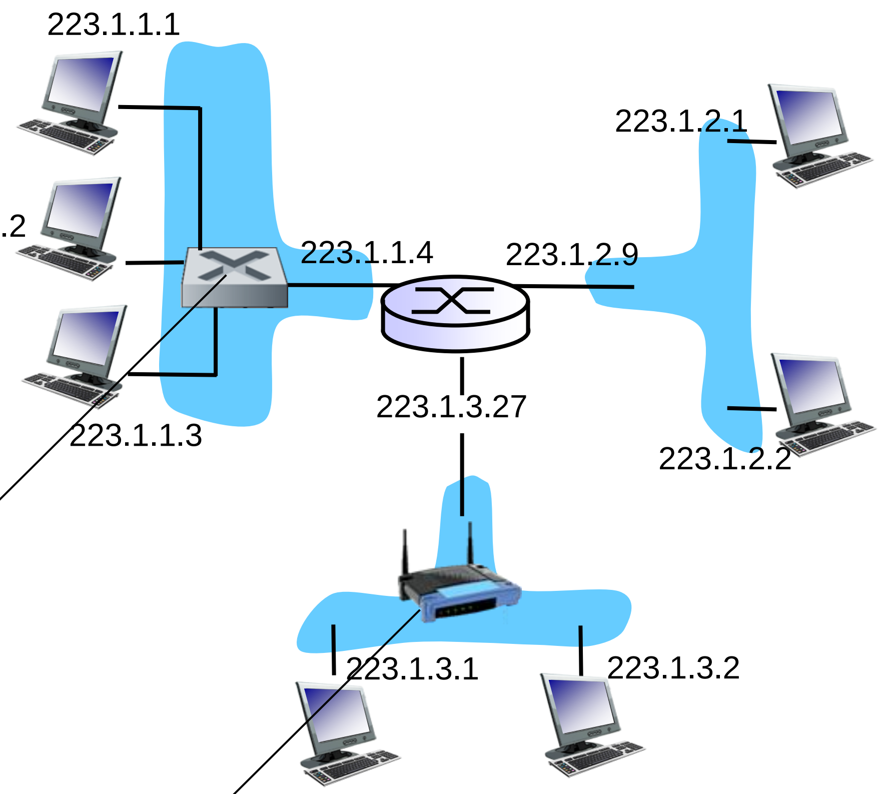


$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

IP addressing: introduction

Q: how are interfaces actually connected?

A: we'll learn about that in chapter 5, 6.



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)

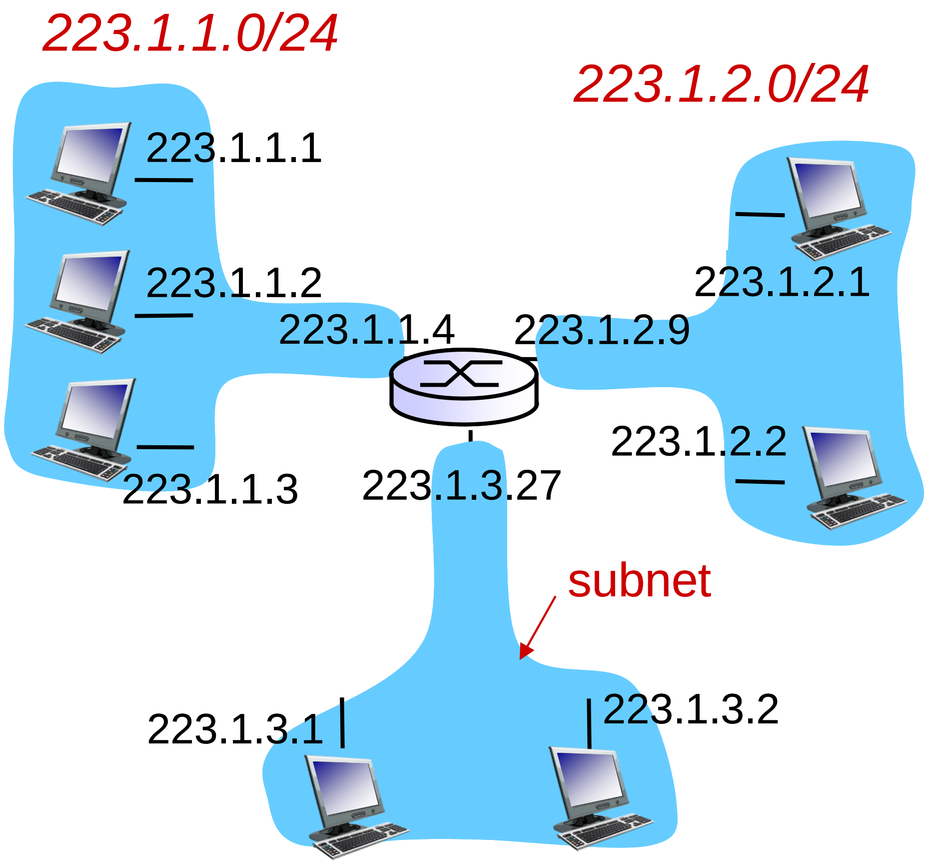
A: wireless WiFi interfaces connected by WiFi base station

Overview

- IP addressing
- Subnet
- How to assign/obtain IP address?

Subnets

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



network consisting of 3 subnets

223.1.3.0/24

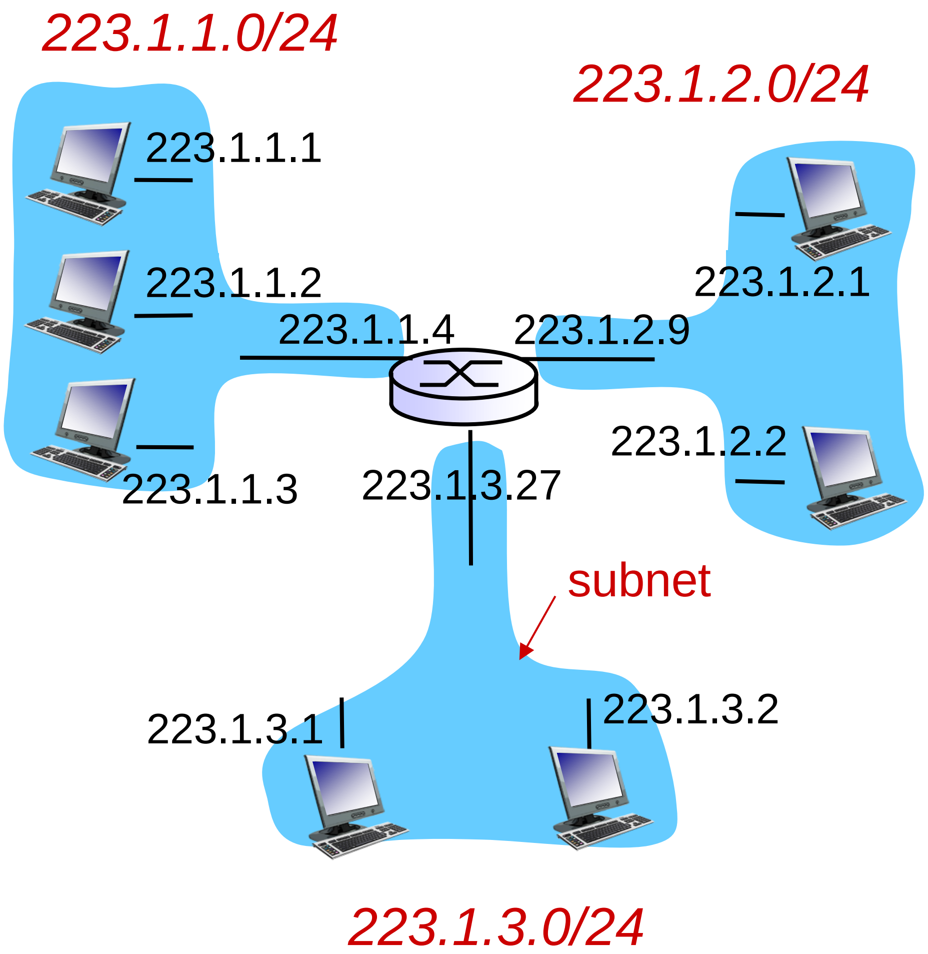
subnet mask: /24

255.255.255.0

Subnets

recipe

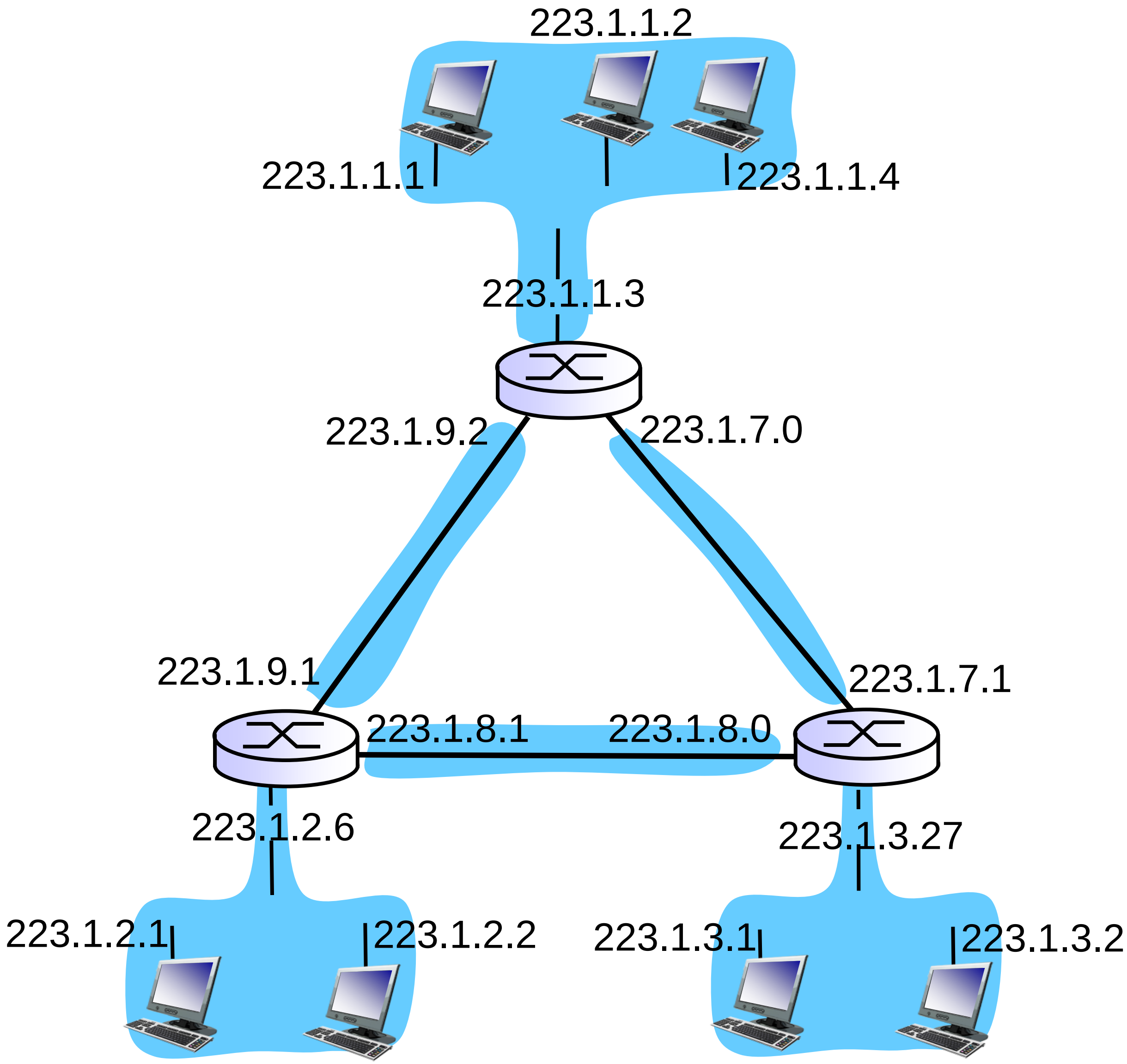
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a *subnet*



subnet mask: /24
255.255.255.0

Subnets

how many?



subnet mask: /24
255.255.255.0

Overview

- IP addressing
- Subnet
- How to assign/obtain IP address?

Subnet addressing: CIDR

CIDR: Classless InterDomain Routing

- A method to assign blocks of IP address
- subnet portion of address of arbitrary length
- Subnet address format: a.b.c.d/x, where x is number of bits in subnet portion of address

subnet part | host part

11001000 00010111 00010000 00000000

- Subnet mask: 255.255.254.0
- Subnet address/block of addresses: 200.23.16.0/23
- Available IP addresses in this subnet?

Overview

- How does an ISP get a block of addresses?
- How does a subnet get a block of addresses?
- How does a host get an IP?

How does an ISP get block of addresses?

Q: how does an ISP get block of addresses?

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

- allocates addresses
- manages DNS
- assigns domain names

How does a subnet get block of addresses?

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

How does a subnet get block of addresses?

Suppose all of the interfaces in each of these three subnets are required to have the prefix 166.4.20.128/25.

- Subnet 1 is required to support at least 62 interfaces
- Subnet 2 is required to support at least 30 interfaces
- Subnet 3 is required to support at least 28 interfaces

Provide three network addresses (of the form a.b.c.d/x) that satisfy these constraints.

How does a subnet get block of addresses?

Block of addresses: 166.4.20.128/25. Subnet 1: at least 62 interfaces; Subnet 2: at least 30 interfaces; Subnet 3: at least 28 interfaces

This block of IP addresses can be written as

10100110 00000100 00010100 10000000

Subnet 1: since $2^6 = 64 > 62$, we can assign the following block

10100110 00000100 00010100 10000000

which can be represented as 166.4.20.128/26.

Subnet 2: since $2^5 = 32 > 30$, we can assign the following block

10100110 00000100 00010100 11000000

which can be represented as 166.4.20.192/27.

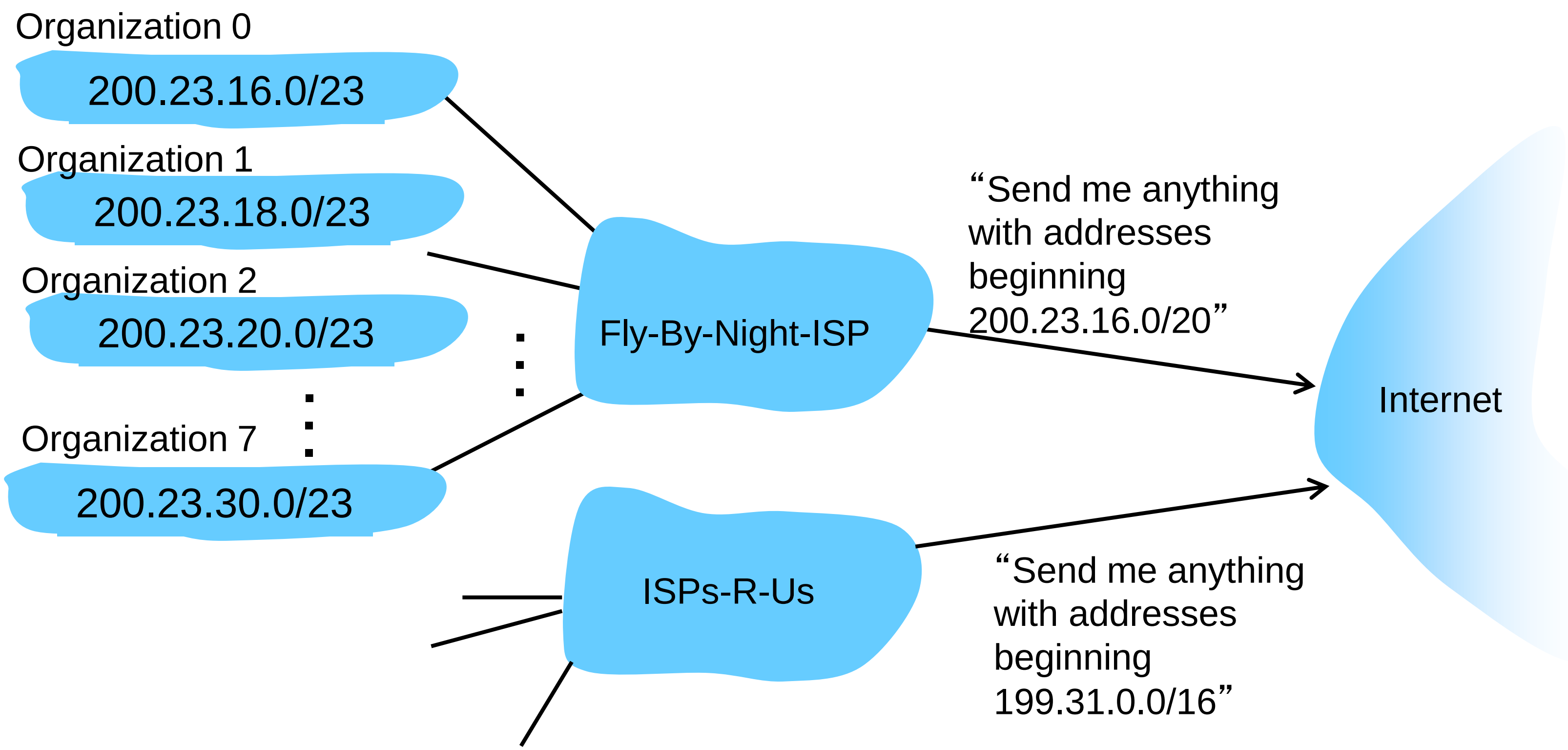
Subnet 3: since $2^5 = 32 > 38$, we can assign the following block

10100110 00000100 00010100 11100000

which can be represented as 166.4.20.224/27.

Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:

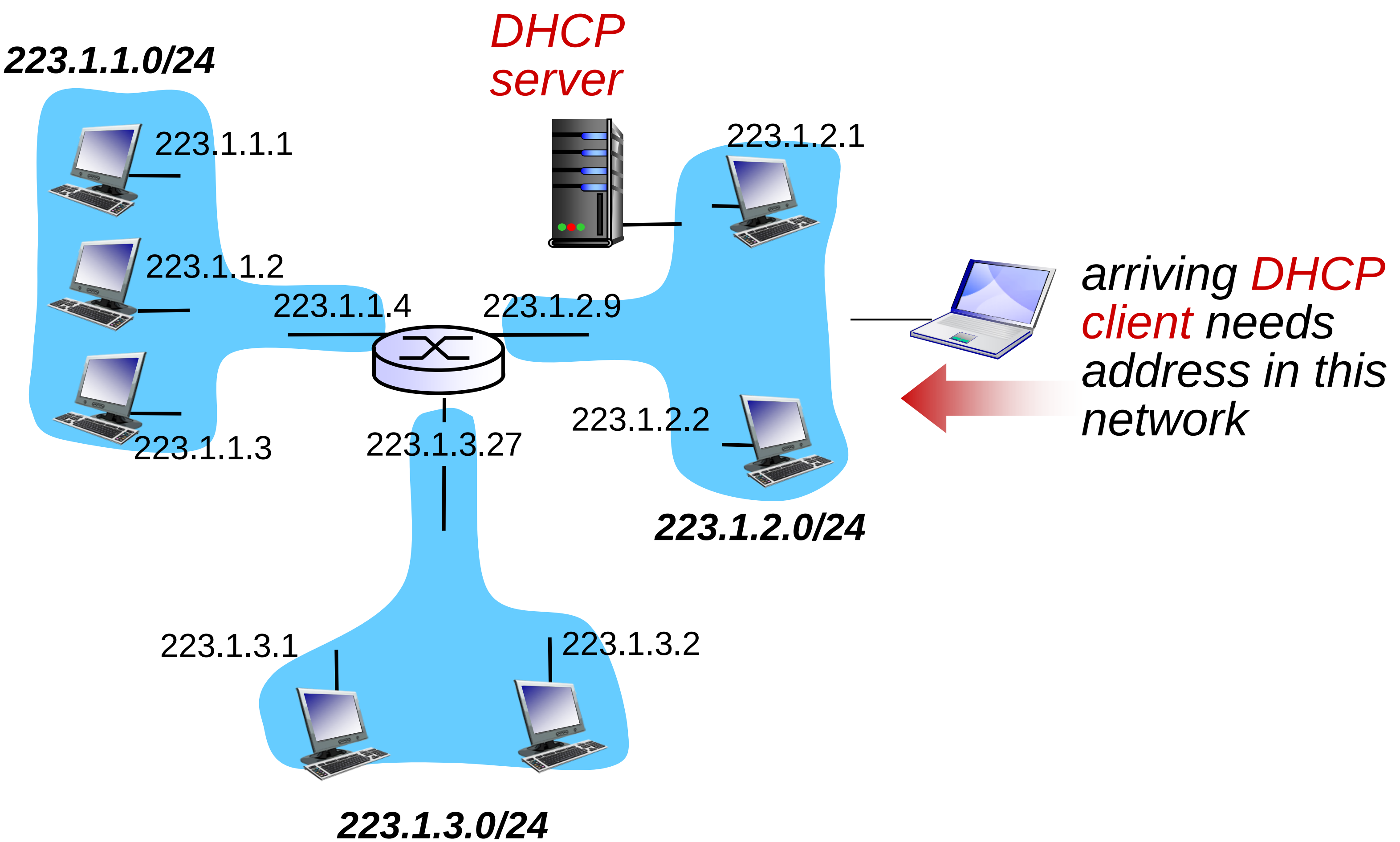


How does a host get an IP?

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

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

DHCP client-server scenario



DHCP: Dynamic Host Configuration Protocol

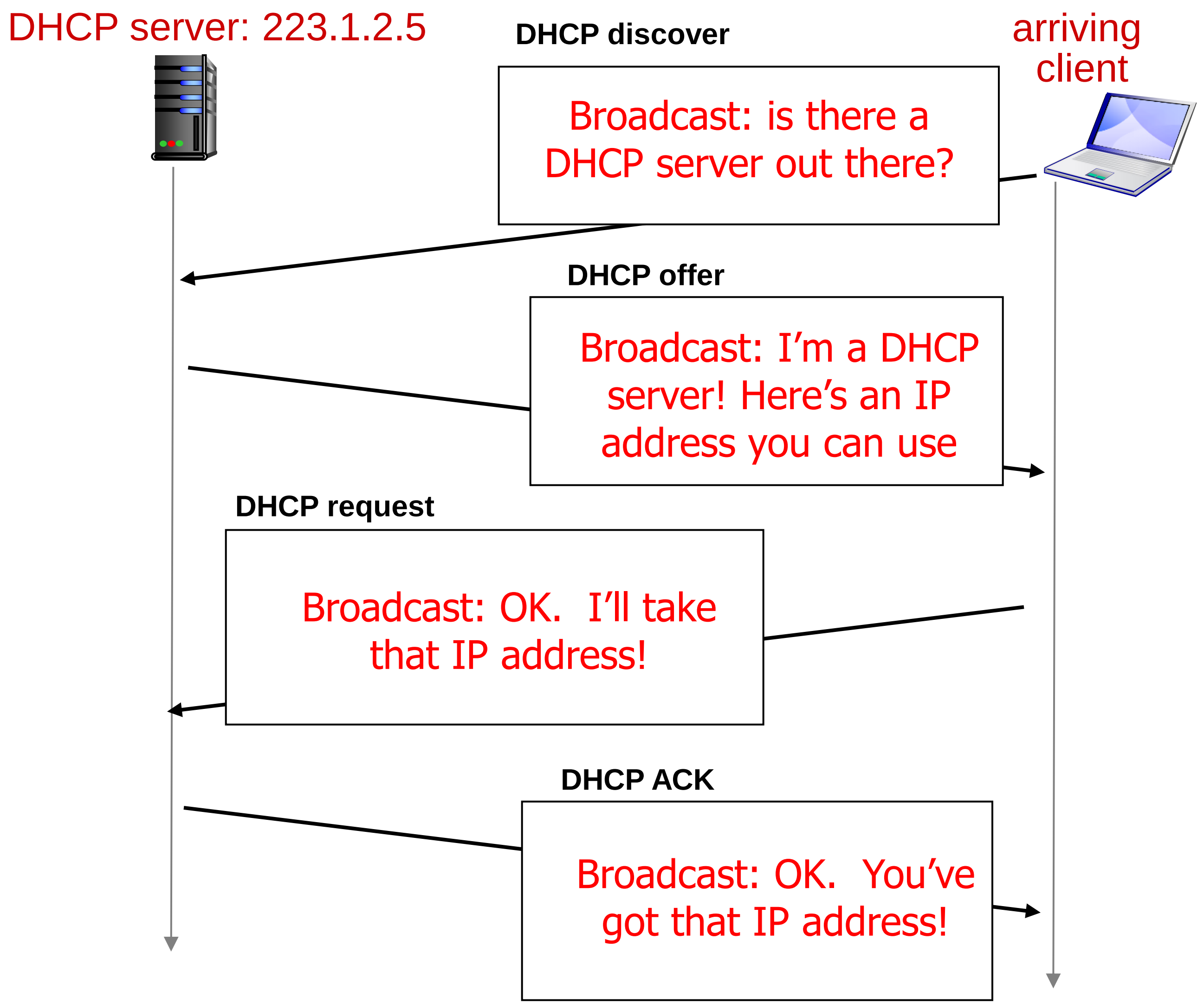
goal: allow host to *dynamically* obtain its IP address from network server when it joins network

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

DHCP 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

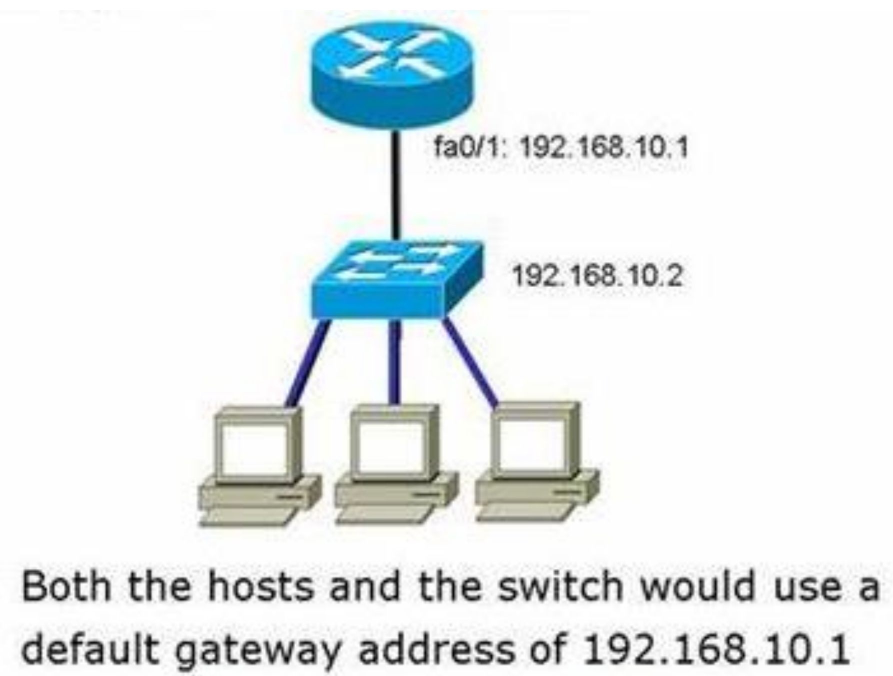
DHCP client-server scenario



DHCP: more than IP addresses

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

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



Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

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

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

NAT: network address translation

Local area networks (LANs):

- E.g., a residence, school, laboratory, university campus or office building
- a range of addresses would need to be allocated by the ISP to cover all of the LAN's IP devices
- If the subnet grew bigger, a larger block of addresses would have to be allocated.
- Huge number of LAN …

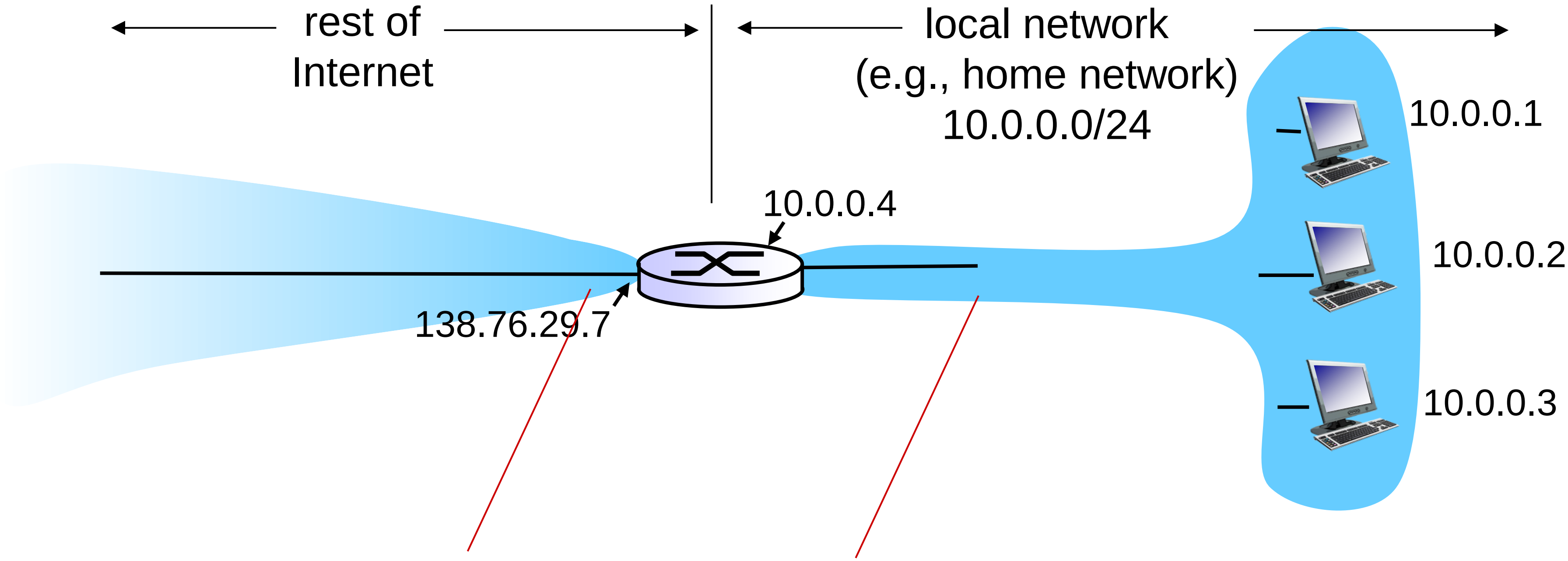
NAT: network address translation

NAT: reserve blocks of IP addresses for LANs

- Private network; private IP address

RFC 1918 name	IP address range	Number of addresses	Largest CIDR block (subnet mask)	Host ID size	Mask bits	*Classful* description[Note 1]
24-bit block	10.0.0.0 – 10.255.255.255	16 777 216	10.0.0.0/8 (255.0.0.0)	24 bits	8 bits	single class A network
20-bit block	172.16.0.0 – 172.31.255.255	1 048 576	172.16.0.0/12 (255.240.0.0)	20 bits	12 bits	16 contiguous class B networks
16-bit block	192.168.0.0 – 192.168.255.255	65 536	192.168.0.0/16 (255.255.0.0)	16 bits	16 bits	256 contiguous class C networks

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 (e.g., 138.76.29.7 in the example) as far as outside world is concerned:

- 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 network 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

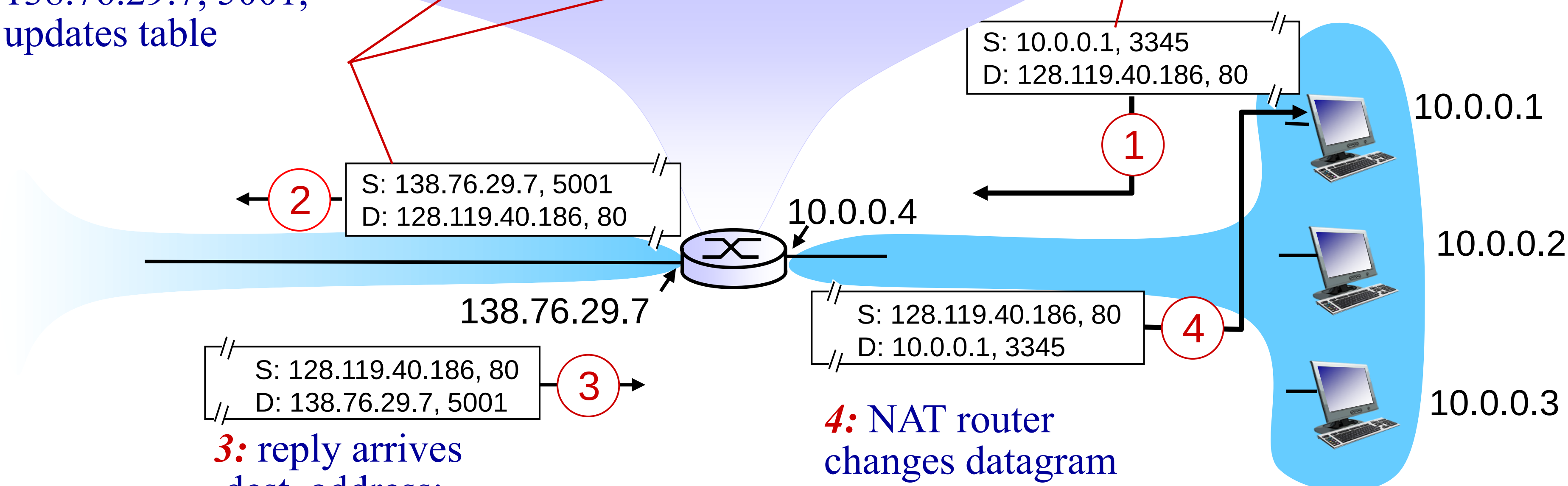
NAT: network address translation

NAT translation table

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

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table



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

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

NAT: network address translation

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
  - routers should only process up to layer 3
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications, server processes
    - Well-known port number

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

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

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

IPv6: motivation

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

IPv6 datagram format:

- 128-bit address space: why?
- fixed-length 40 byte header; why?
- no fragmentation allowed；why?

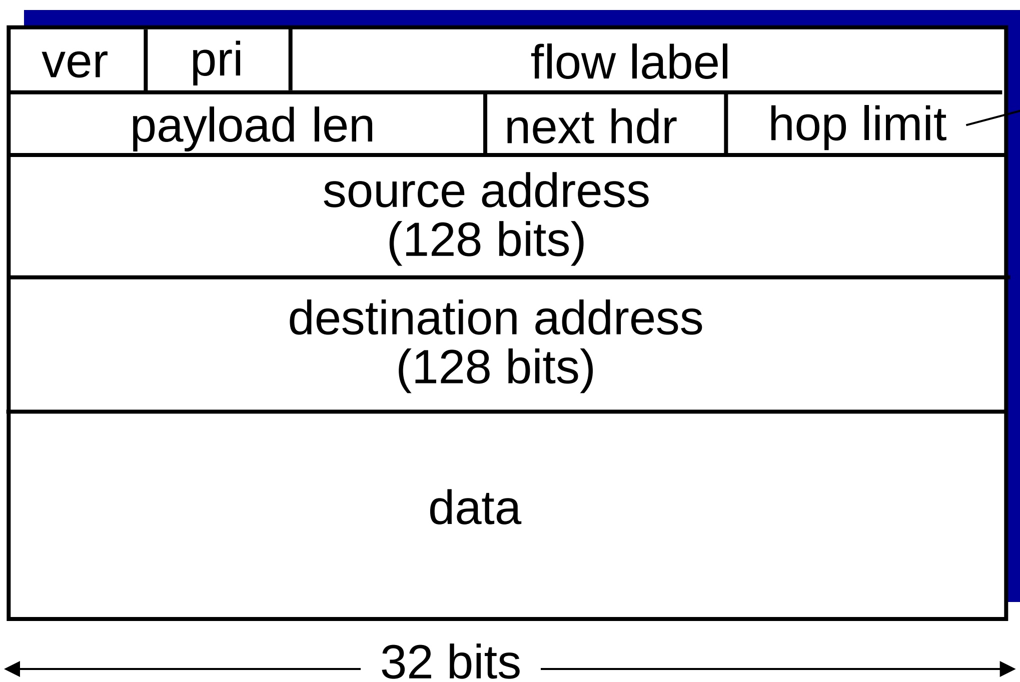
IPv6 datagram format

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 (for example, to TCP or UDP).



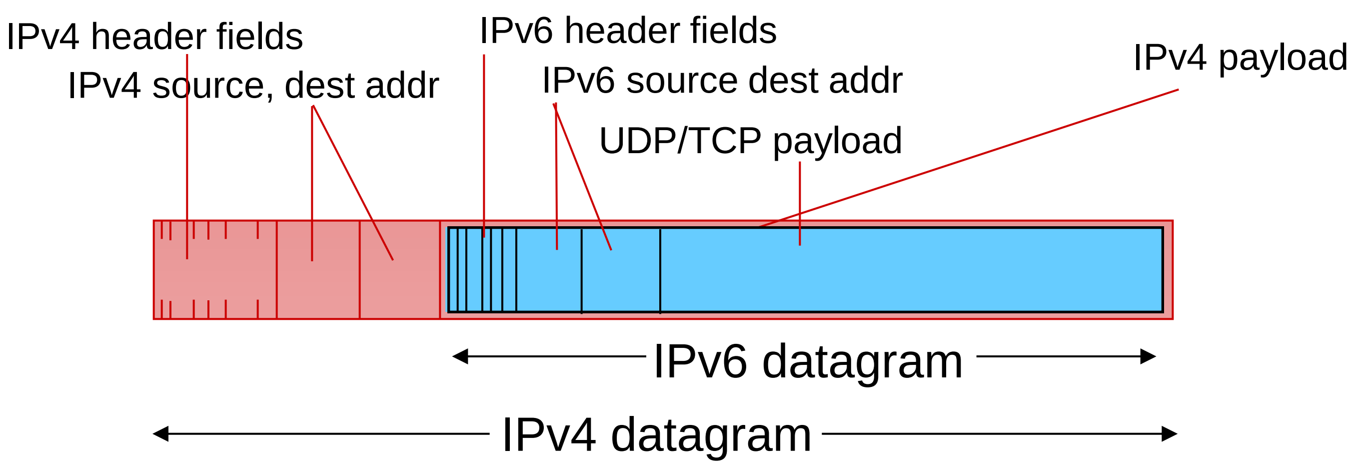
decremented by one by each router that forwards the Datagram; if reaches zero, the datagram is discarded.

Other changes from IPv4

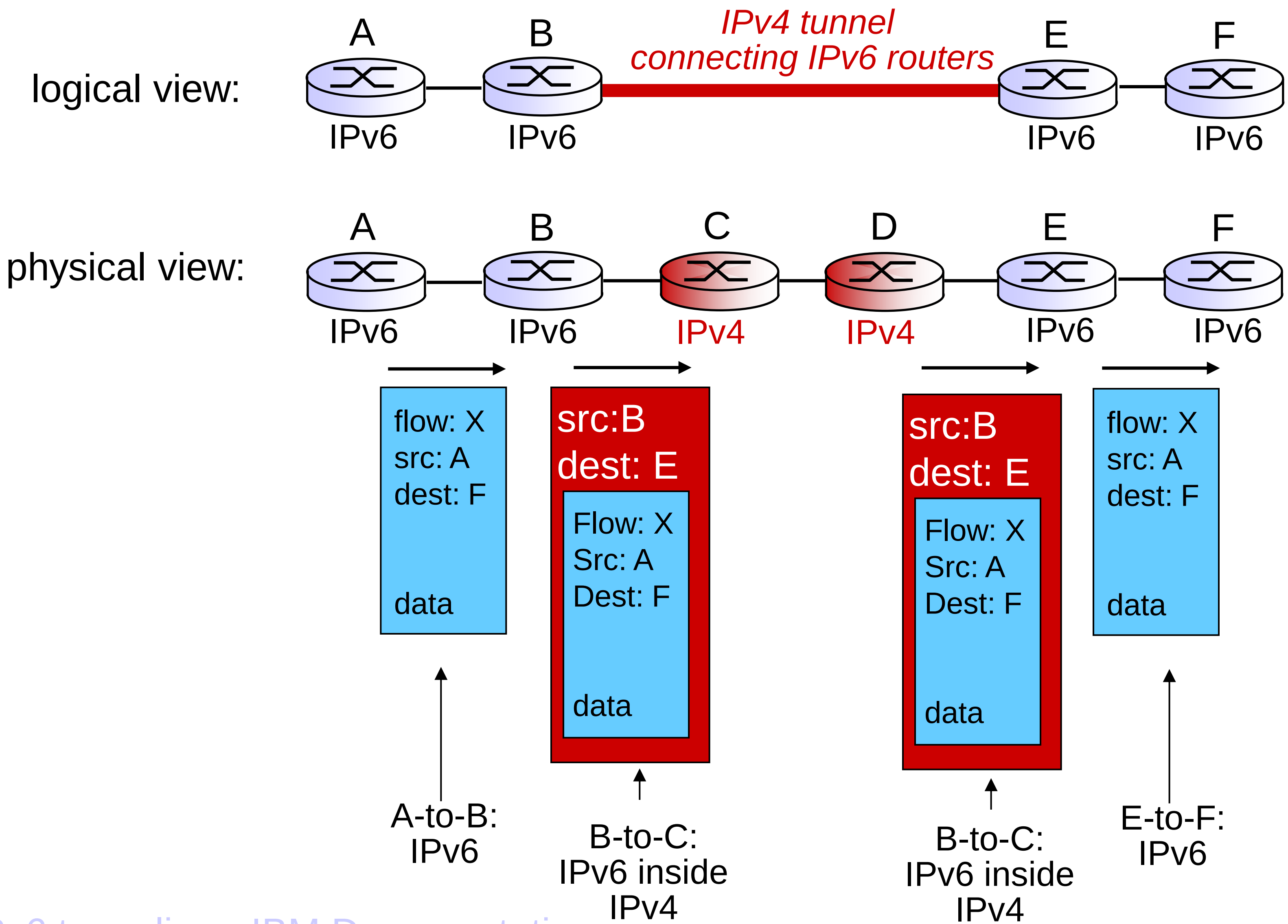
- *checksum*: removed entirely to reduce processing time at each hop; why?
- *options:* allowed, but outside of header, indicated by "Next Header" field
- *no fragmentation:*
  - too large to be forwarded over the outgoing link
  - the router simply drops the datagram and sends a "Packet Too Big" ICMP error message

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



[IPv6 tunneling - IBM Documentation]

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?*

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

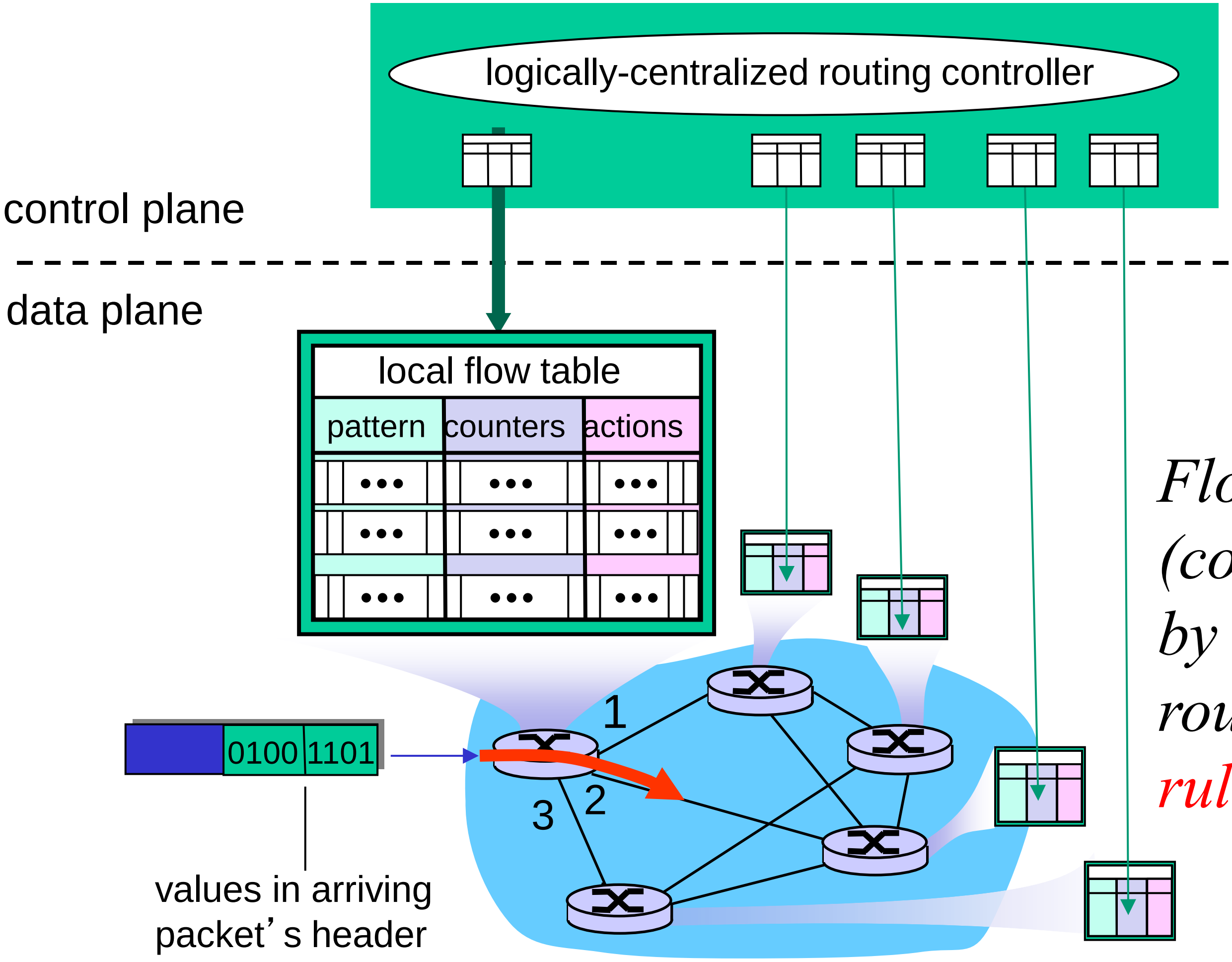
4.2 What's inside a router

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

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

Generalized Forwarding and SDN

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

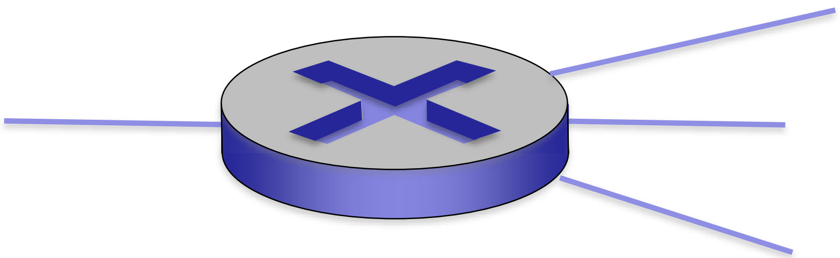


Flow table in a router (computed and distributed by controller) define router's match+action rules

OpenFlow data plane abstraction

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:* #packets; time last updated



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

OpenFlow: Flow Table Entries

Pattern	Action	counter

#packets; time last updated

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

Switch Port	VLAN ID	MAC src	MAC dst	Eth type	IP Src	IP Dst	IP Prot	TCP/UDP sport	TCP/UDP dport

Link layer

Network layer

Transport layer

Destination-based forwarding:

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP/UDP sport	TCP/UDP 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:

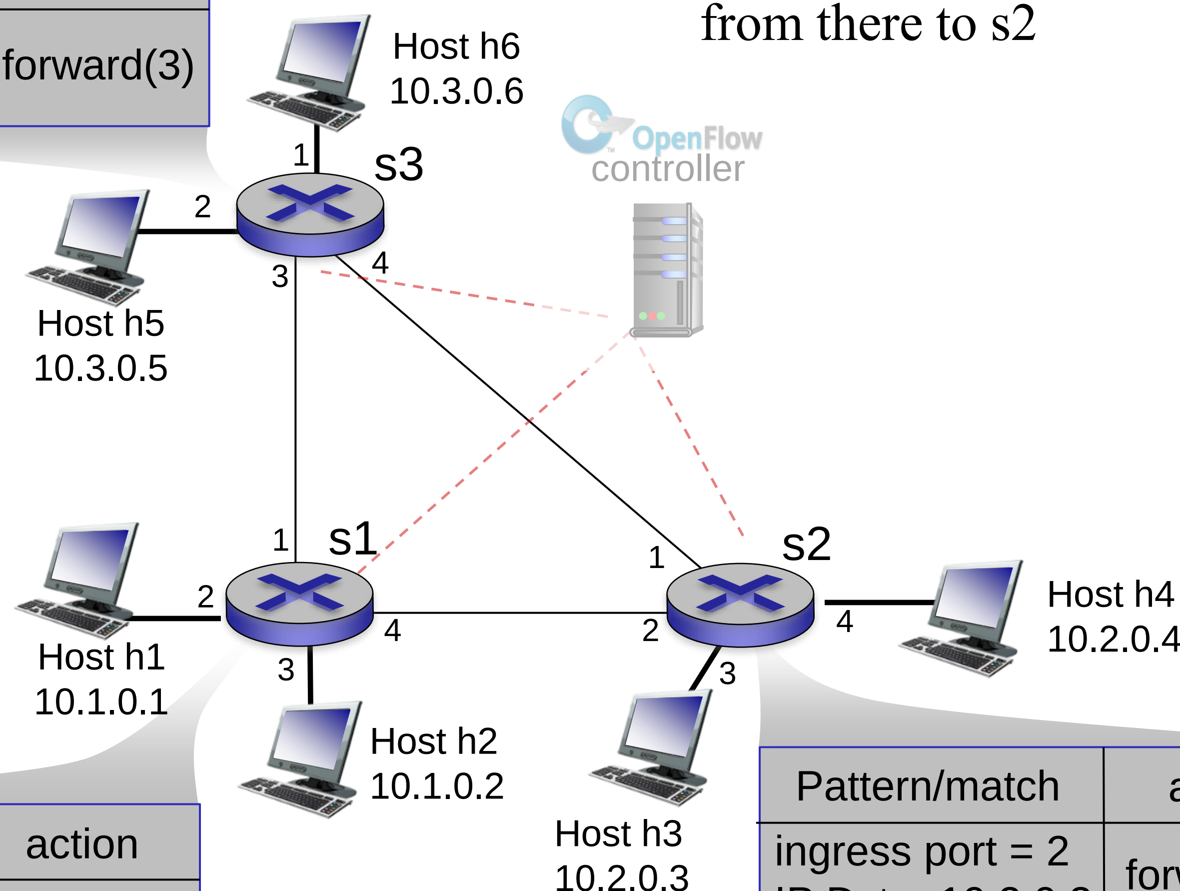
Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP/UDP sport	TCP/UDP dport	Action
*	22:A7:23:11:E1:02	*	*	*	*	*	*	*	*	port3

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

OpenFlow example

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

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



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

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

Firewall

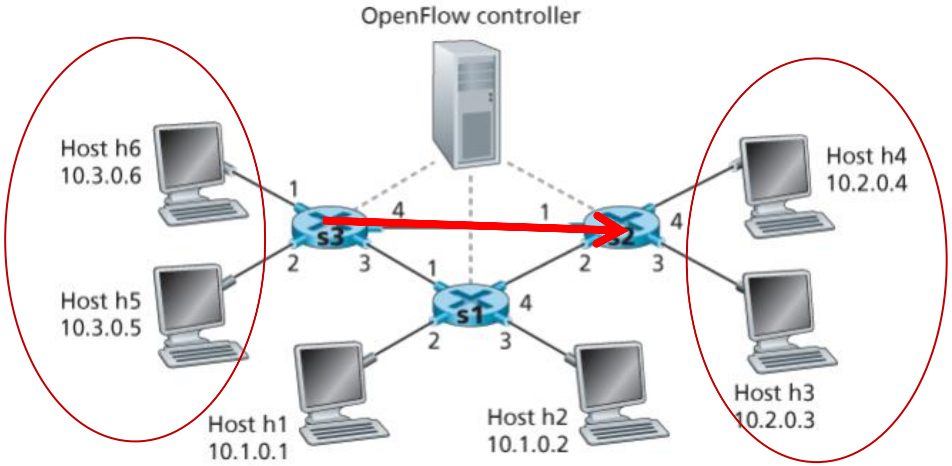
Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP/UDP dport	TCP/UDP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

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/UDP dport	TCP/UDP dport	Forward
*	*	*	*	*	128.119.1.1	*	*	*	*	drop

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

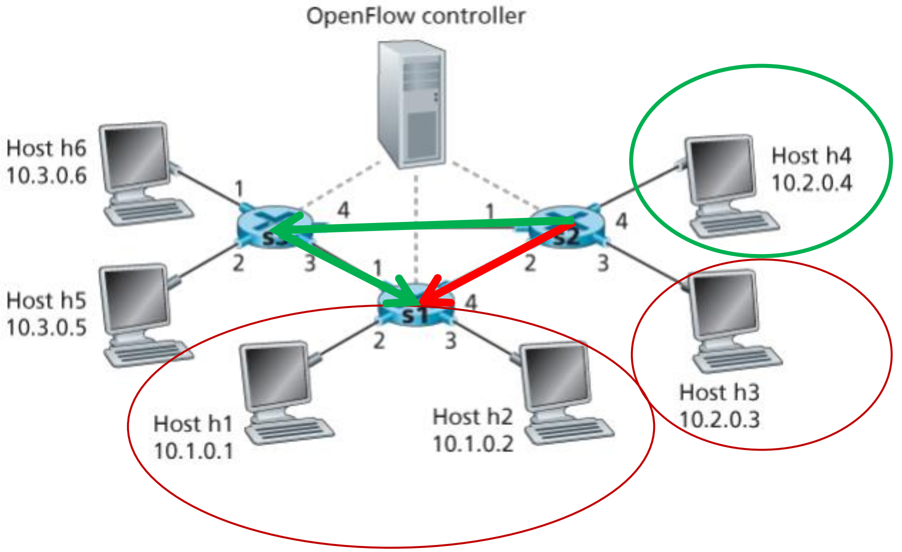
Firewall



s2 Flow Table (Example 3)	
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)

there were no other entries in s2's flow table

Load Balancing



s2 Flow Table (Example 2)	
Match	Action
Ingress port = 3; IP Dst = 10.1.*.*	Forward(2)
Ingress port = 4; IP Dst = 10.1.*.*	Forward(1)
…	…

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
- 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!*

4.1 Overview of Network layer: data plane and control plane

4.2 What's inside a router

4.3 IP: Internet Protocol

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

4.4 Generalized Forward and SDN

- match plus action
- OpenFlow example

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

Answer: by the control plane (next chapter)

Chapter 5: network layer control plane

chapter goals: understand principles behind network control plane

- traditional routing algorithms
- SDN controllers
- Internet Control Message Protocol
- network management

and their instantiation, implementation in the Internet:

- OSPF, BGP, OpenFlow, ICMP, SNMP

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Network-layer functions

Recall: two network-layer functions:

- *forwarding:* move packets from router's input to appropriate router output

data plane

- *routing:* determine route taken by packets from source to destination

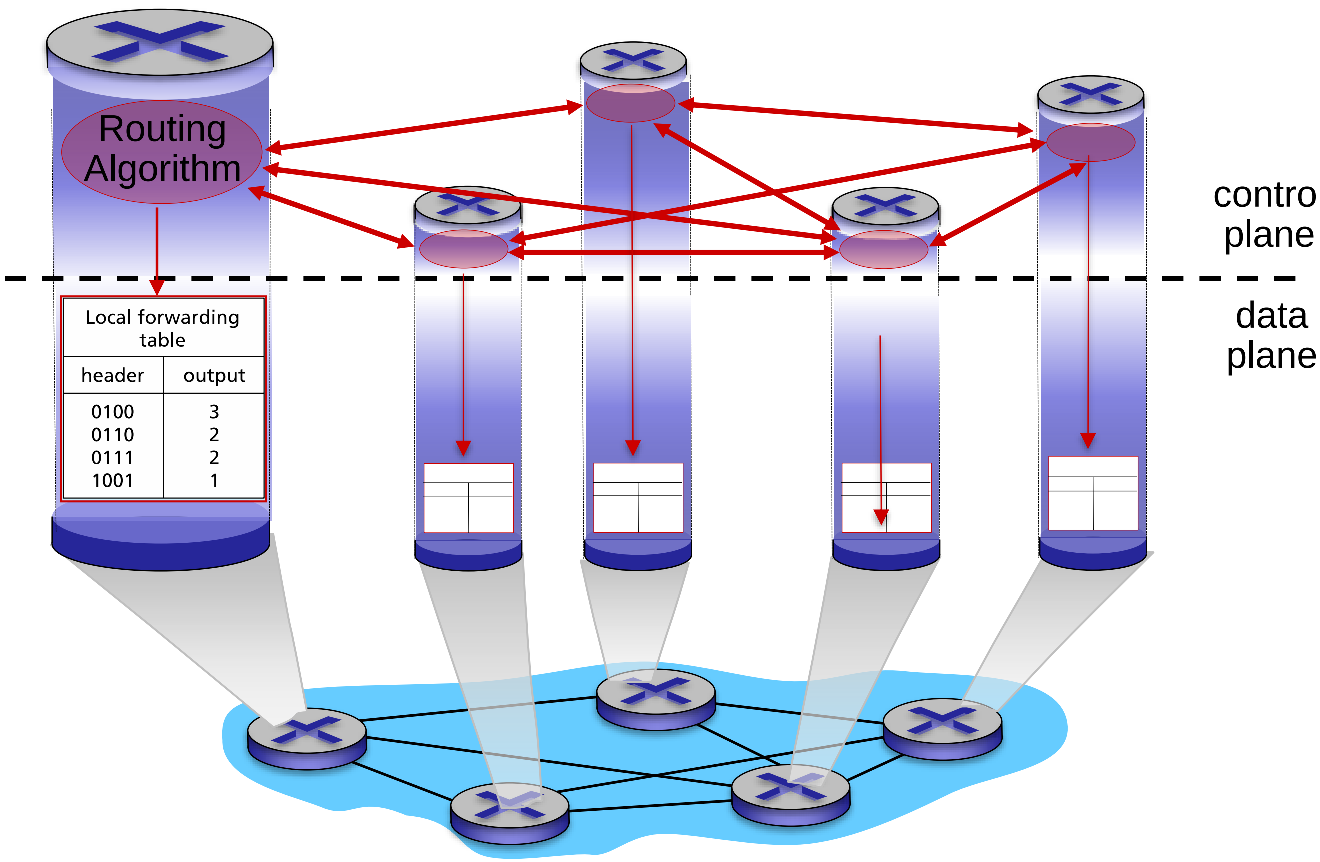
control plane

Two approaches to structuring network control plane:

- per-router control (traditional)
- logically centralized control (software defined networking)

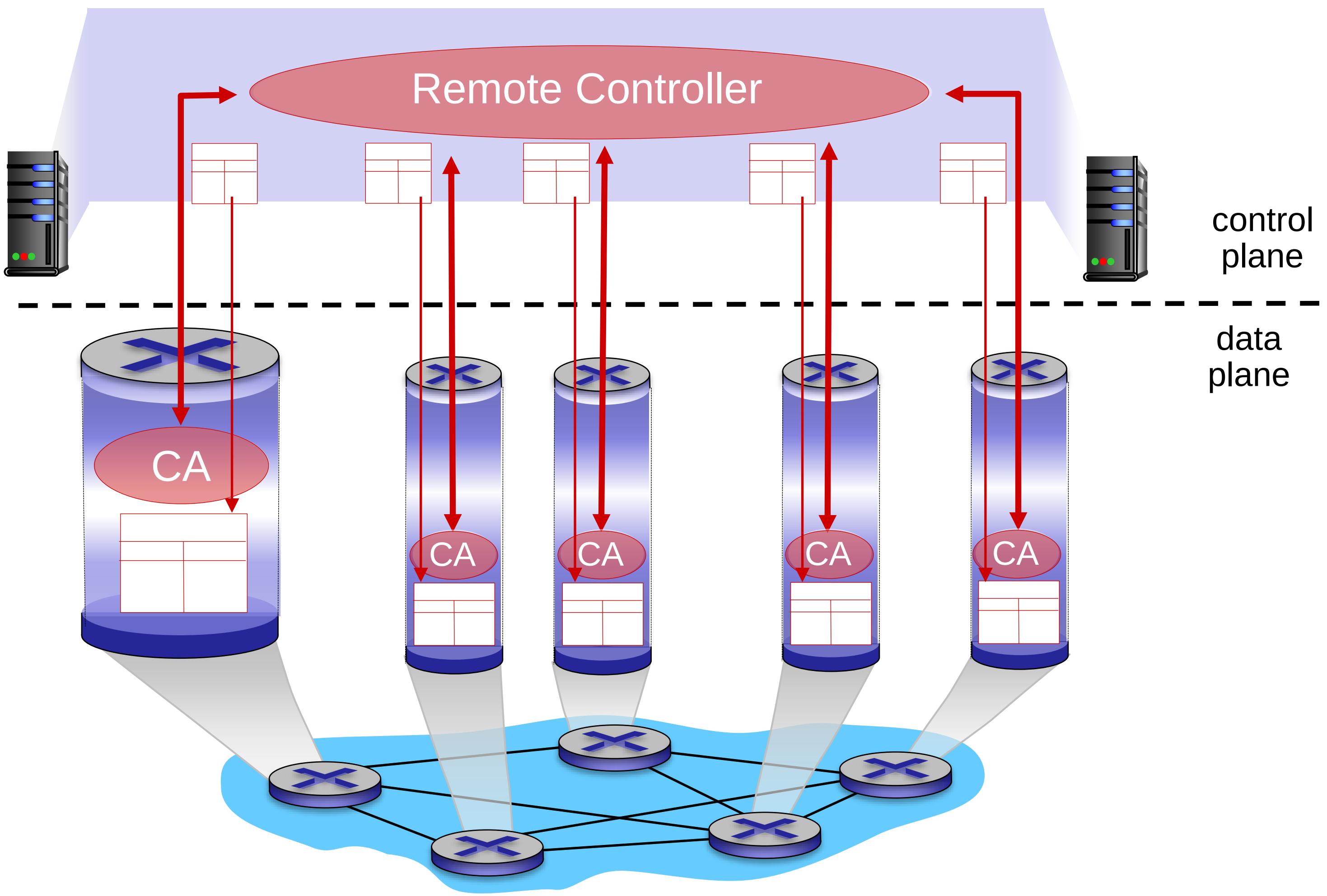
Per-router control plane

Individual routing algorithm components *in each and every router* interact with each other in control plane to compute forwarding tables



Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables



Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

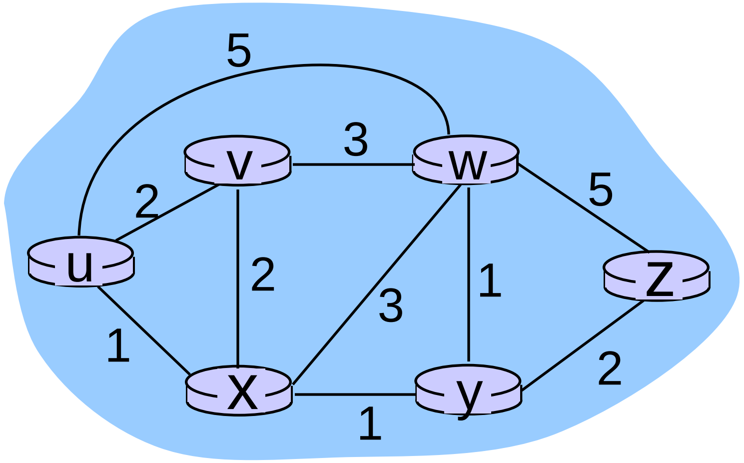
5.7 Network management and SNMP

Routing protocols

Routing protocol goal: determine “good” paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- path: sequence of routers packets will traverse in going from given initial source host to given final destination host
- “good”: least “cost”, “fastest”, “least congested”
- routing: a “top-10” networking challenge!

Graph abstraction of the network

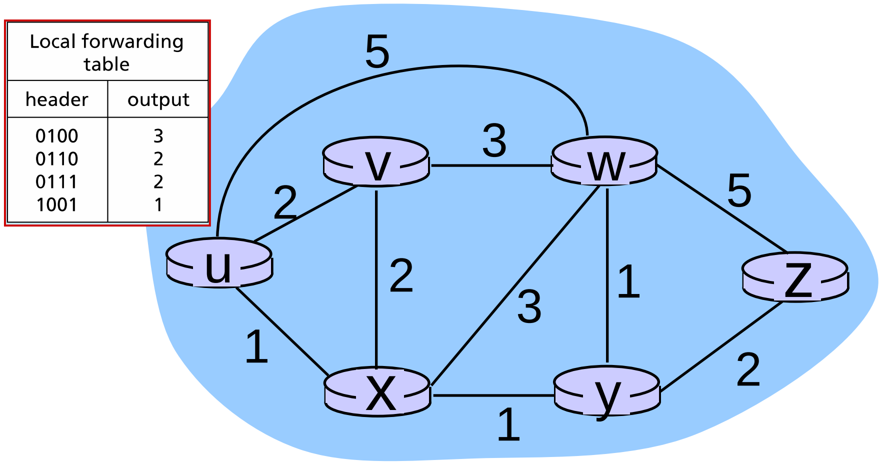


graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Graph abstraction: costs



c(x,x') = cost of link (x,x')
e.g., c(w,z) = 5

cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

cost of path $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

key question: what is the least-cost path between u and z ?
routing algorithm: algorithm that finds that least cost path

Routing algorithm classification

Q: global or decentralized information?

global:

- all routers have complete topology, link cost info
- "link state" algorithms

decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Q: static or dynamic?

static:

- routes change slowly over time

dynamic:

- routes change more quickly
  - periodic update
  - in response to link cost changes

Routing algorithm classification

Q: load-sensitive or load insensitive?

Load-sensitive:

- Link costs vary dynamically to reflect the current level of congestion

Load-insensitive

- A link's cost does not explicitly reflect its current level of congestion

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state (global)
- distance vector (decentralized)

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP