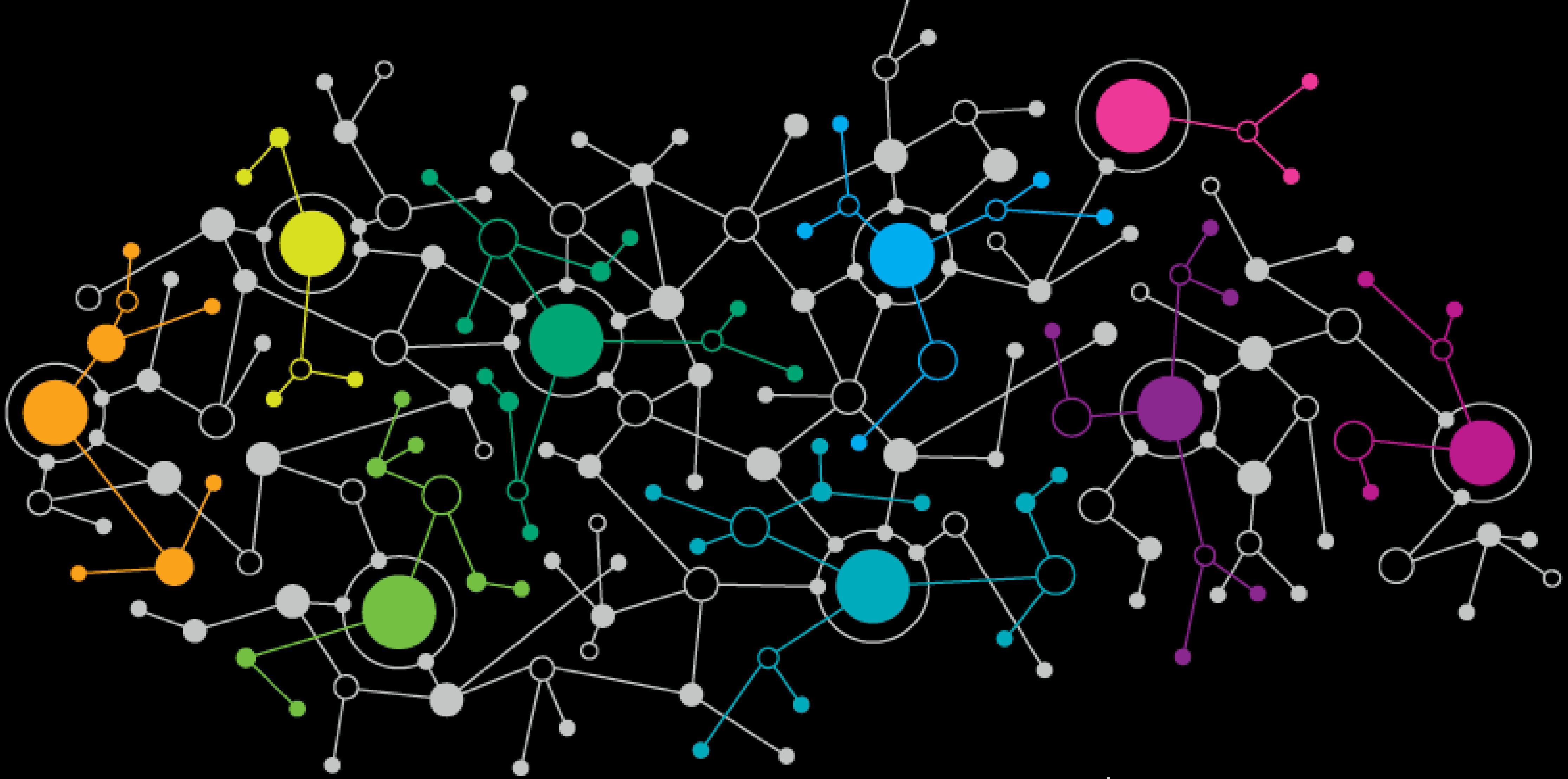


Introduction to Networking

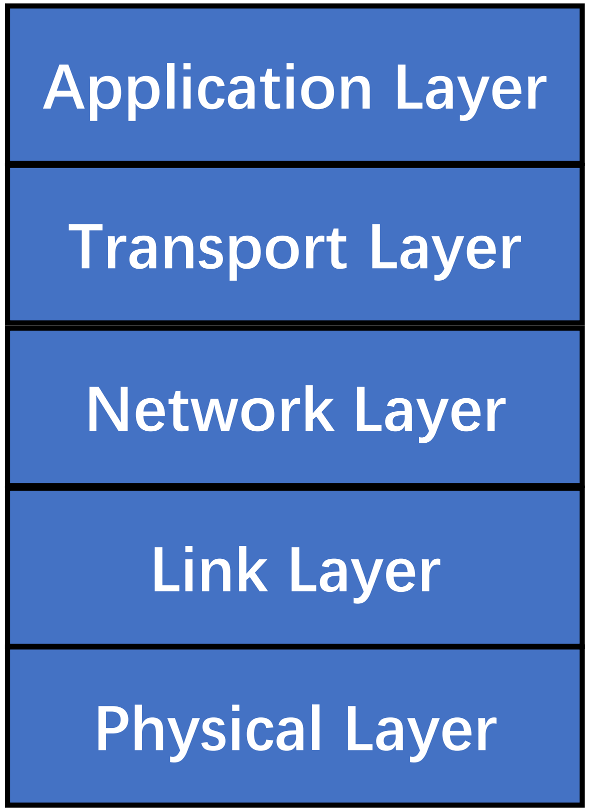
CAN201 – Week 8

Lecturer: Dr. Wenjun Fan

Announcements

- **Addresses**
  - Email: Wenjun.Fan@xjtlu.edu.cn
  - Office: EE 214
- **Office hours (appointment in advance)**
  - Monday: 12:00 – 13:00
  - Tuesday: 12:00 – 13:00

Recap: Top-Down Approach



Application Layer

Transport Layer

Network Layer

Link Layer

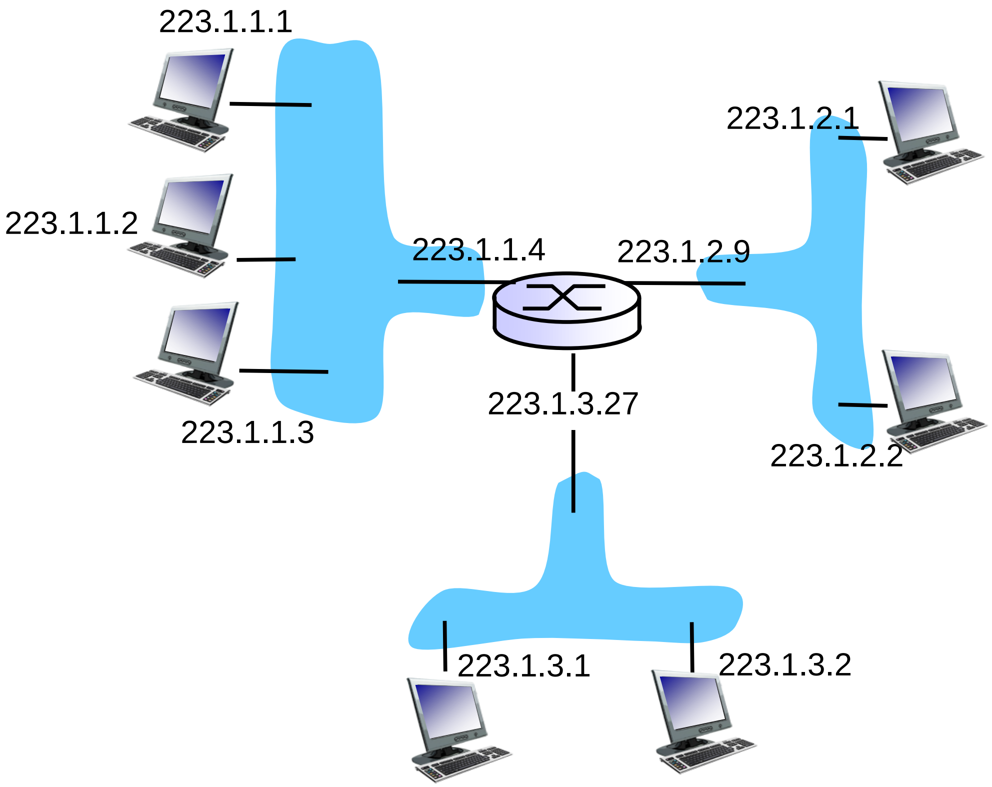
Physical Layer

Lecture 7 – Network Layer Data Plane (2)

- **Roadmap**
  1. IPv4 addressing
  2. NAT
  3. IPv6
  4. Generalized Forward and SDN

IP addressing: introduction

- ***Interface:*** **the boundary between host/router and physical link**
  - Host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
  - Router typically has multiple interfaces
- ***IP addresses associated with each interface (**rather than with the host or router containing that interface**).***
- **Q: if a router has 3 interfaces, how many IP addresses it has?**
- ***IP address:* 32-bit identifier for host and router *interface***

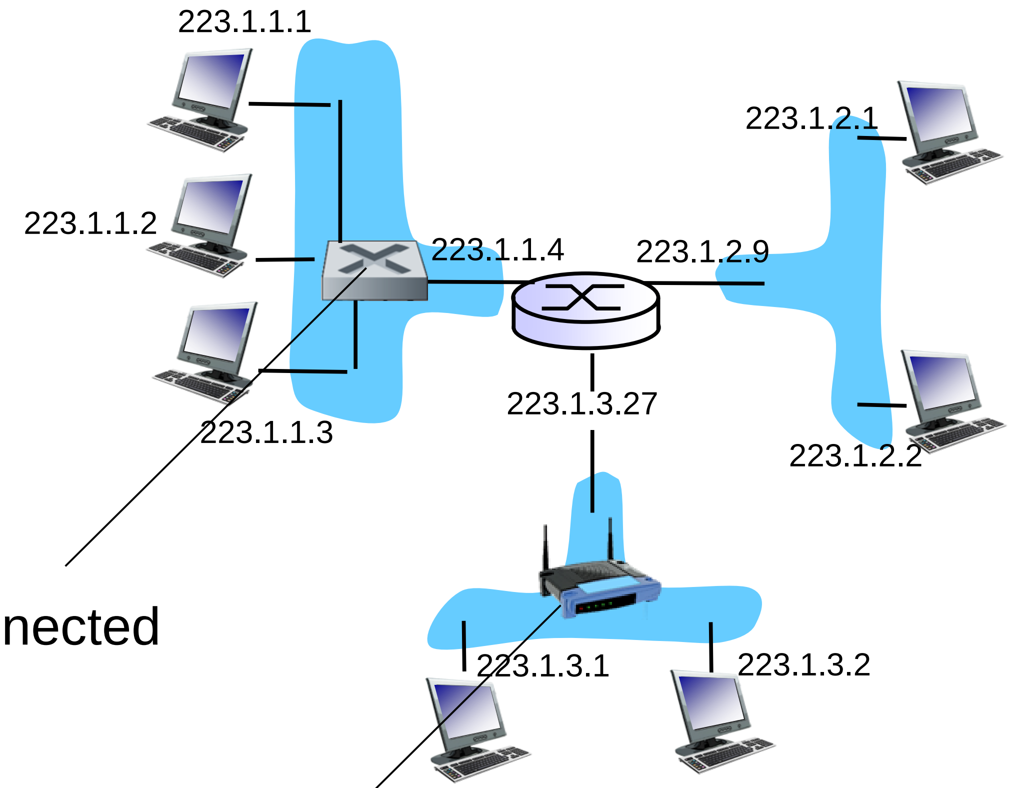


$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

IP addressing: introduction

Q: how are interfaces actually connected?

A: by switch, we'll learn it in the future lecture.



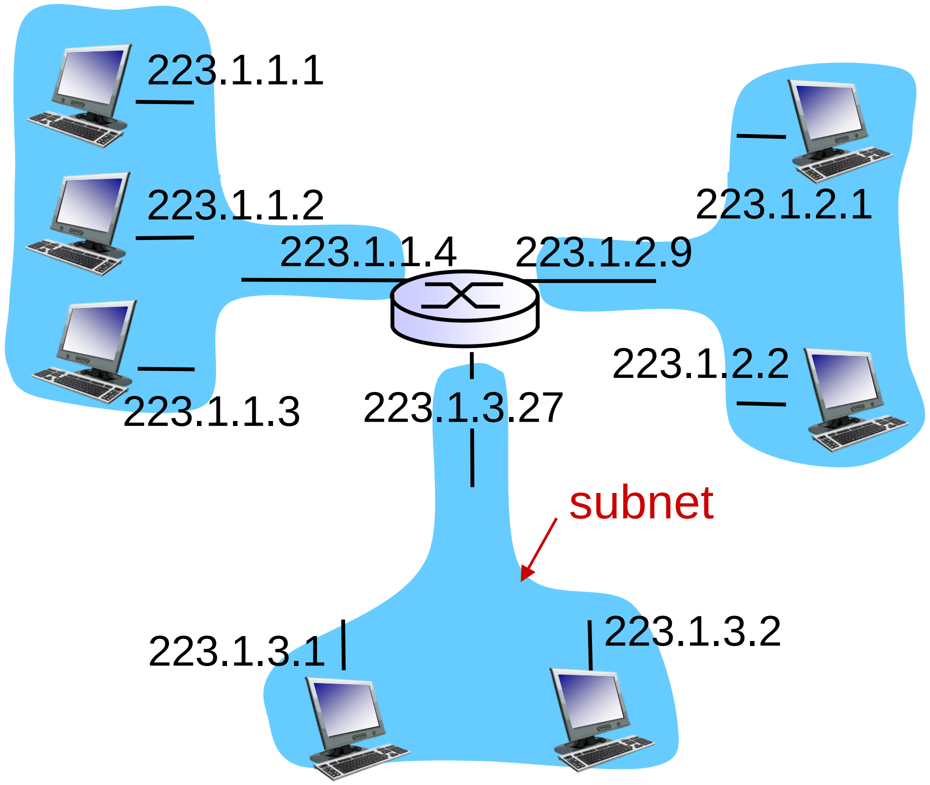
Wired Ethernet interfaces connected by Ethernet switches

Does switch contain IP addresses? Why?

Wireless WiFi interfaces connected by WiFi base station

Subnets

- **IP address, two parts:**
  - 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*

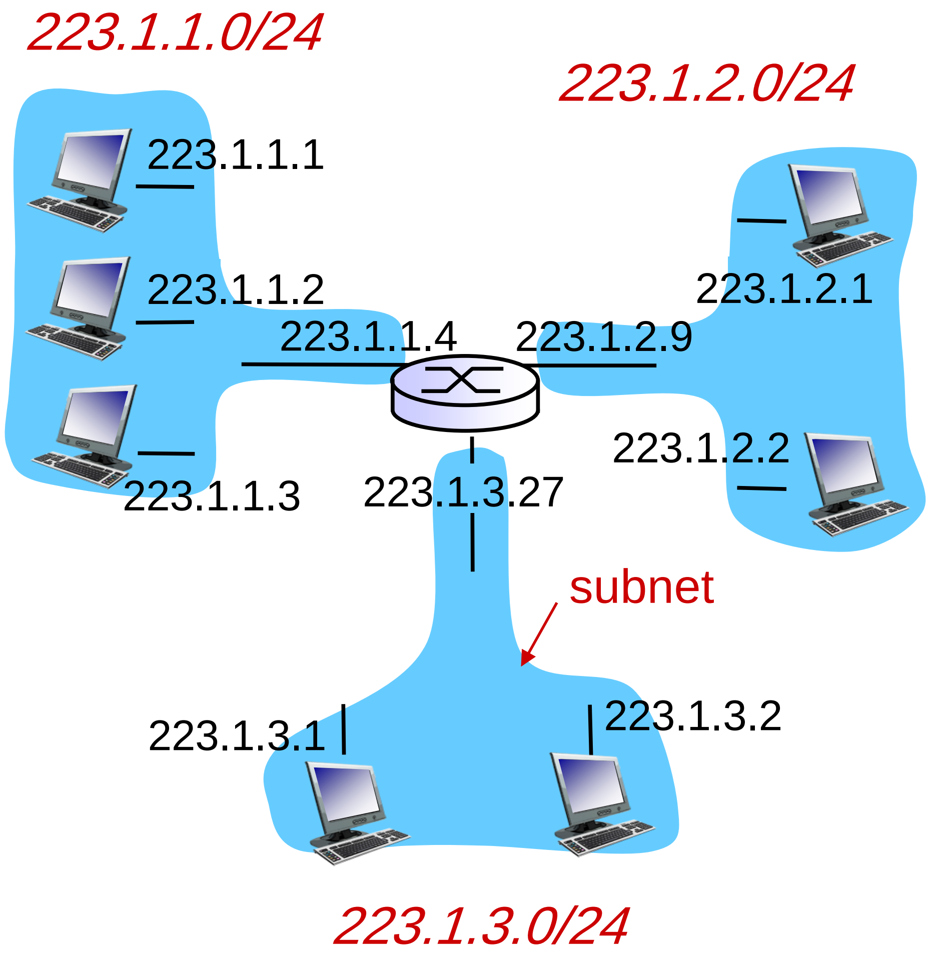


network consists of 3 subnets

Subnets

Recipe

- To determine the subnets, detach each interface from its host or router to create isolated networks.
- Each isolated network is called a ***subnet.***



subnet mask: /24

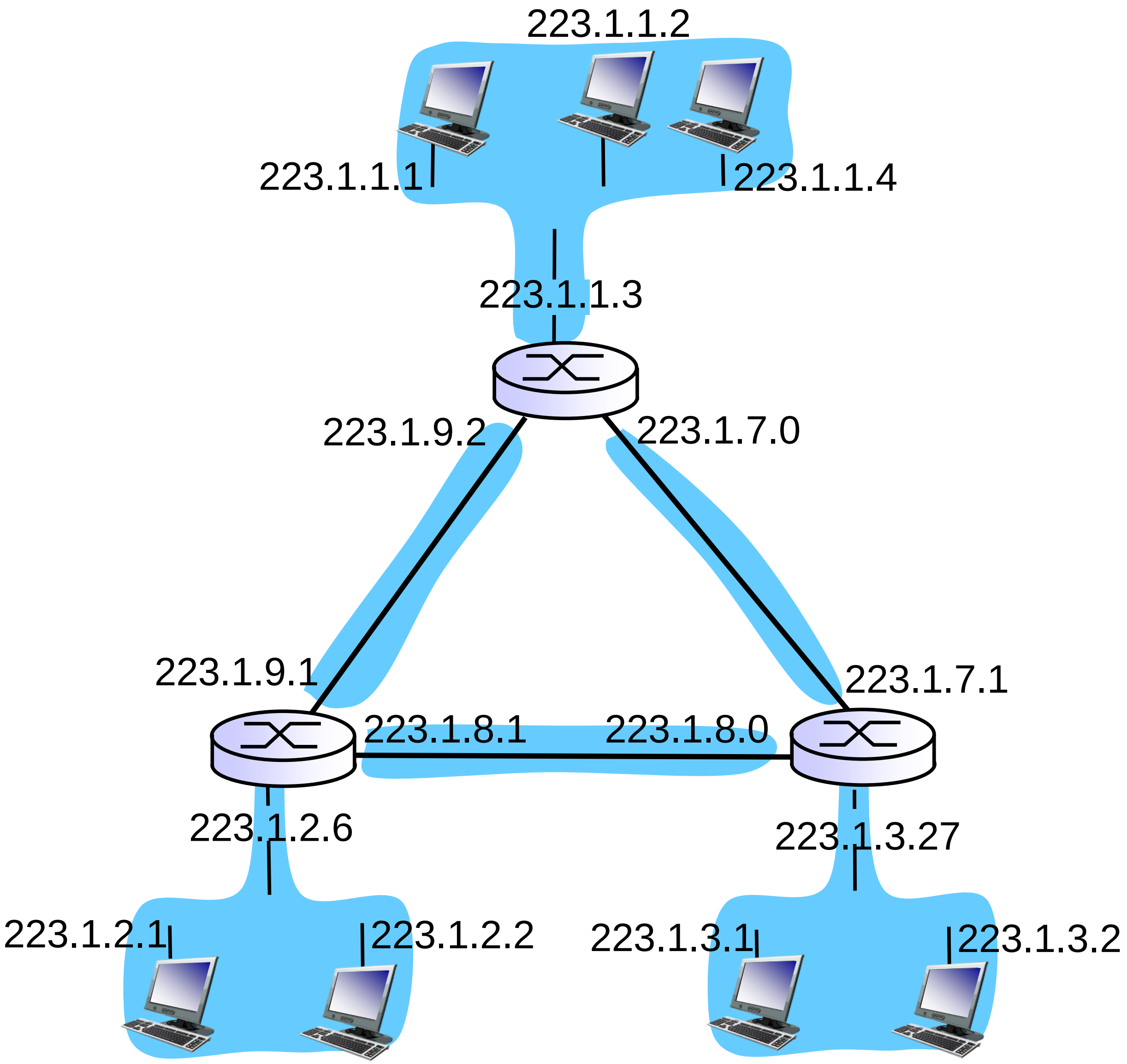
Subnets

- **How many subnets?**

- **What subnets are they? Assuming /24 for all subnet masks.**

Subnet 1 includes h1, h2, h3, and r1 ✗

Subnet 1 includes the interfaces of h1, h2, h3, and r1 ✓



IP addressing: Network Classes

Classful addressing

- The network portion of an IP address were constrained to be 8, 16, or 24 bits in length.
- Subnets with 8-, 16-, and 24-bit subnet addresses were known as class A, B and C networks.
- It became problematic
  - A class C (/24) subnet could accommodate only up to $2^8 – 2 = 254$ hosts
  - A class B (/16) subnet supporting $2^{16} – 2 = 65,534$ hosts, which is too large

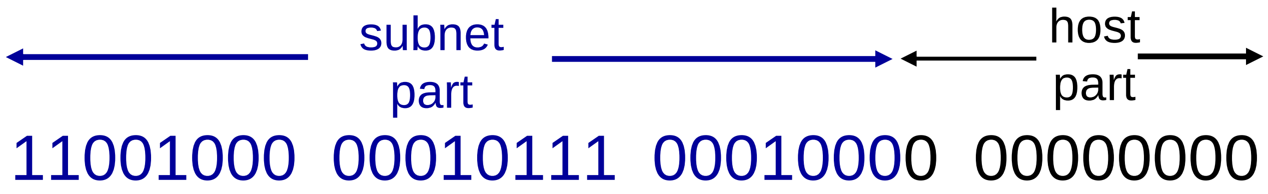
Under classful addressing, an organization with 2,000 hosts, which subnet class should be allocated? What is the problem?

This led to a rapid depletion of the class B address space and poor utilization of the assigned address space.

IP addressing: CIDR

CIDR: Classless Inter Domain Routing

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



subnet part

host part

11001000 00010111 00010000 00000000

200.23.16.0/23

Network name with address format: a.b.c.0/x
Subnet mask: /x
Network prefix: x

IP addresses: how to get one?

Q: How does a *host interface* get an 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**” or **zeroconf** protocol

  because of DHCP’s ability to automate the network-related aspects of connecting a host into a network.

DHCP: Dynamic Host Configuration Protocol

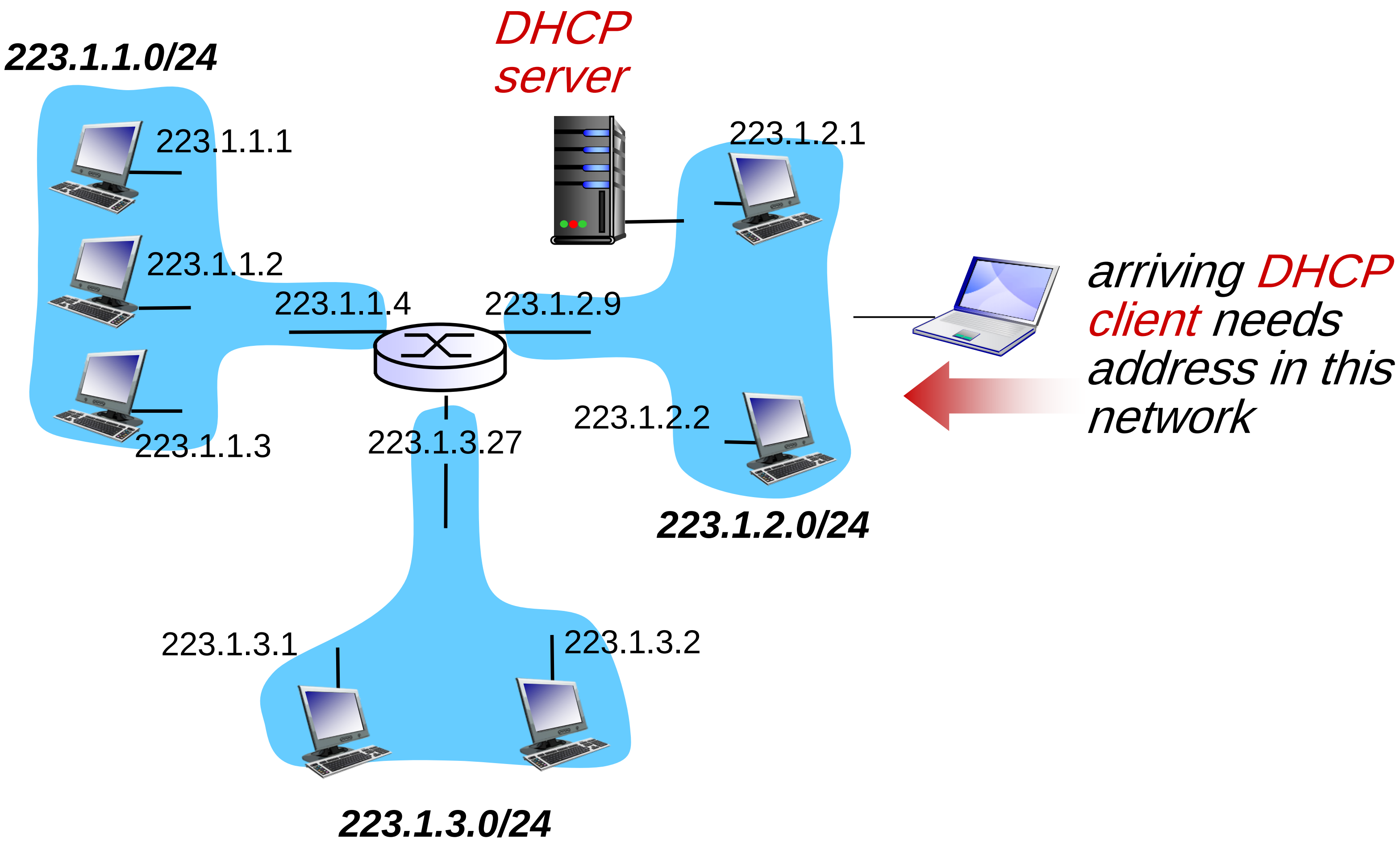
Goal: **allow host to *dynamically* obtain its IP address from network server when it joins network**

- 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)

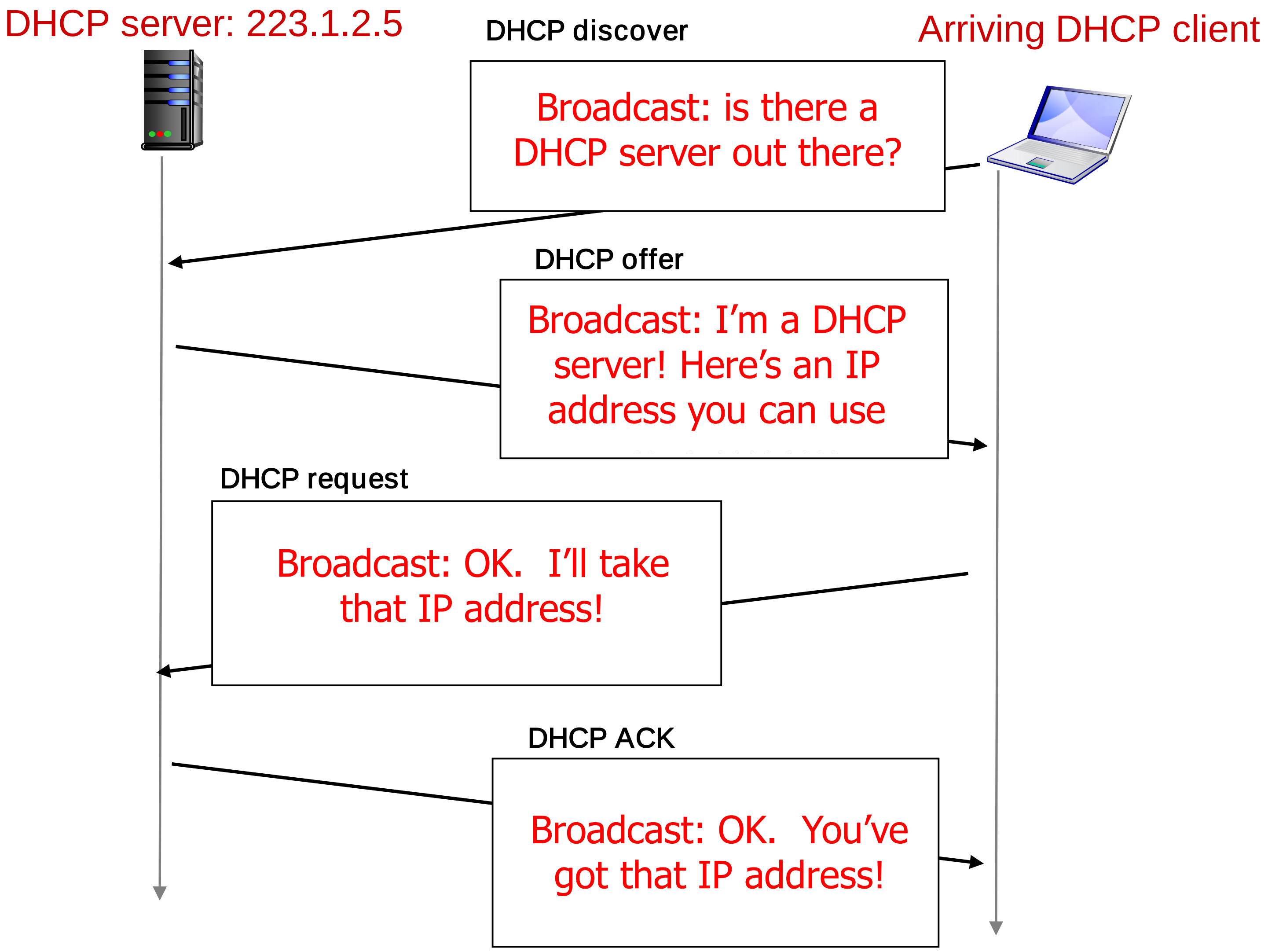
DHCP overview:

- host broadcasts "DHCP discover" msg [optional]
- DHCP server responds with "DHCP offer" msg [optional]
- host requests IP address: "DHCP request" msg
- DHCP server sends address: "DHCP ack" msg

DHCP client-server scenario



DHCP client-server scenario



DHCP server: 223.1.2.5

Arriving DHCP client

DHCP discover

Broadcast: is there a DHCP server out there?

DHCP offer

Broadcast: I'm a DHCP server! Here's an IP address you can use

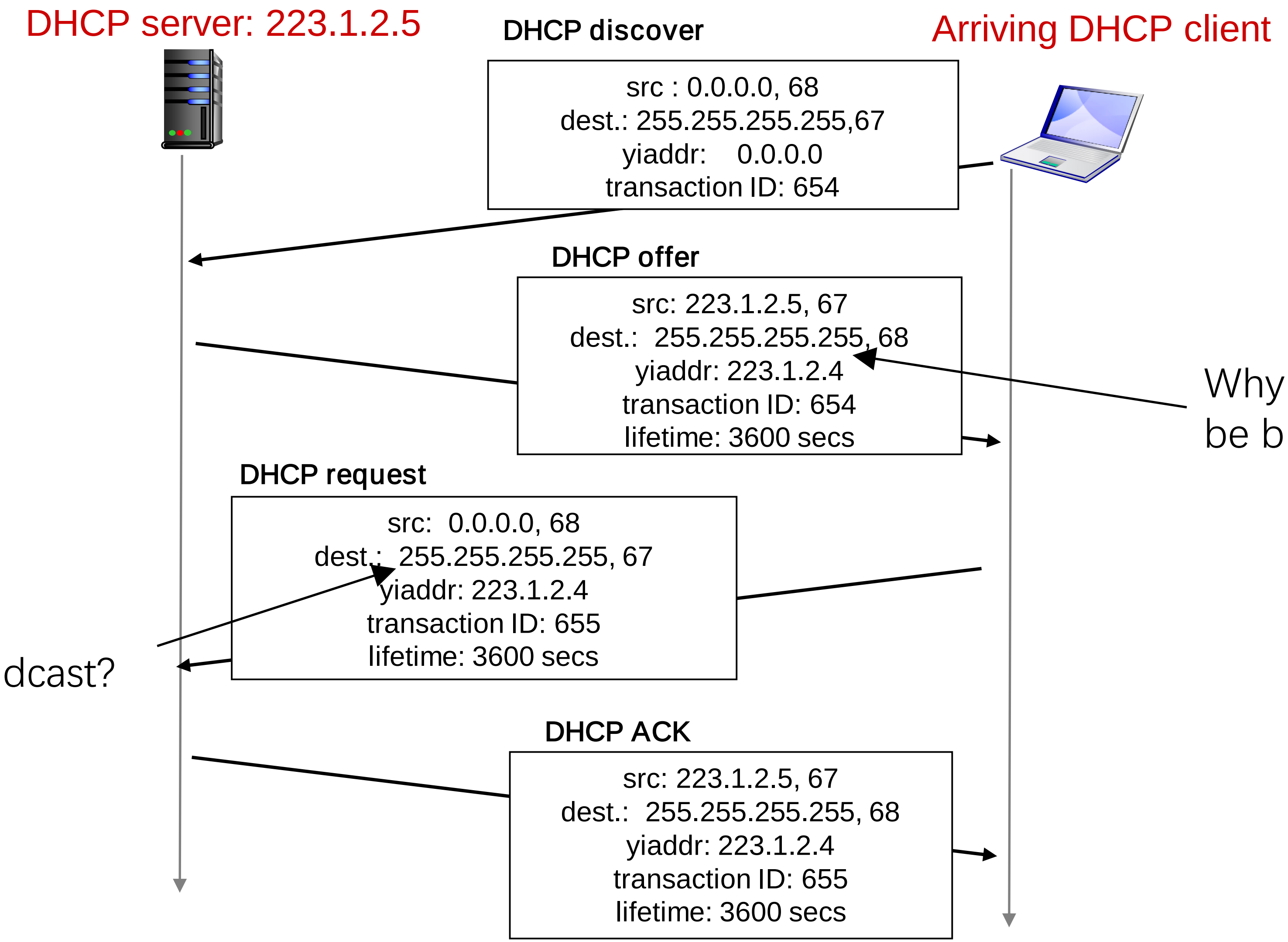
DHCP request

Broadcast: OK. I'll take that IP address!

DHCP ACK

Broadcast: OK. You've got that IP address!

DHCP client-server scenario



DHCP server: 223.1.2.5

Arriving DHCP client

DHCP discover

src : 0.0.0.0, 68
dest.: 255.255.255.255,67
yiaddr: 0.0.0.0
transaction ID: 654

DHCP offer

src: 223.1.2.5, 67
dest.: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 654
lifetime: 3600 secs

DHCP request

src: 0.0.0.0, 68
dest.: 255.255.255.255, 67
yiaddr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

DHCP ACK

src: 223.1.2.5, 67
dest.: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

Why must this server reply be broadcast?

Why must this client request also be broadcast?

DHCP: more than IP addresses

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

- Address of first-hop router (a.k.a. gateway) for client
- Name and IP address of DNS sever
- Network mask (indicating network versus host portion of address)

IP addresses: how to get one?

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

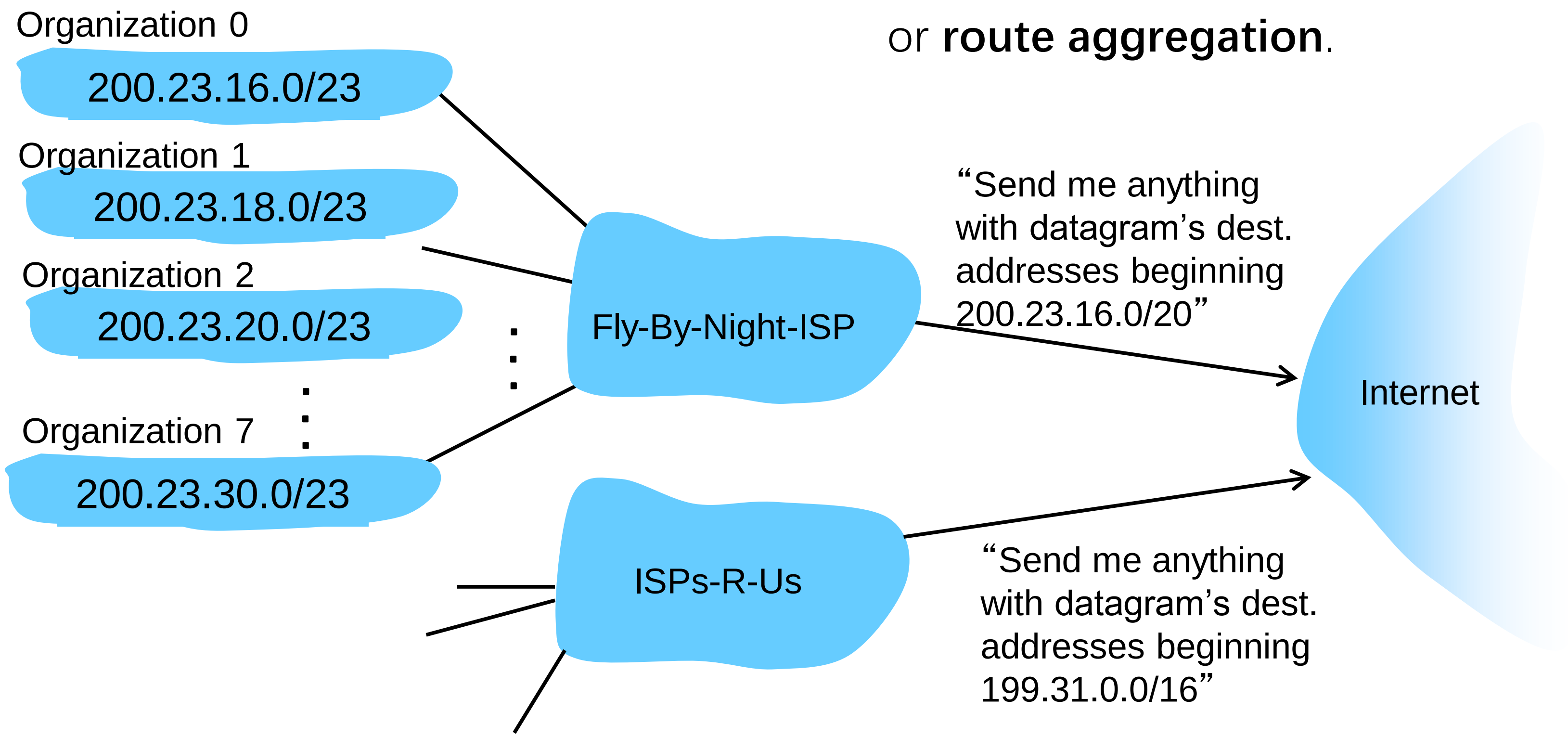
A: **gets allocated portion of its provider ISP's address space**

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:**

The ability to use a single prefix to advertise multiple networks is often referred to as **address aggregation** or **route aggregation**.



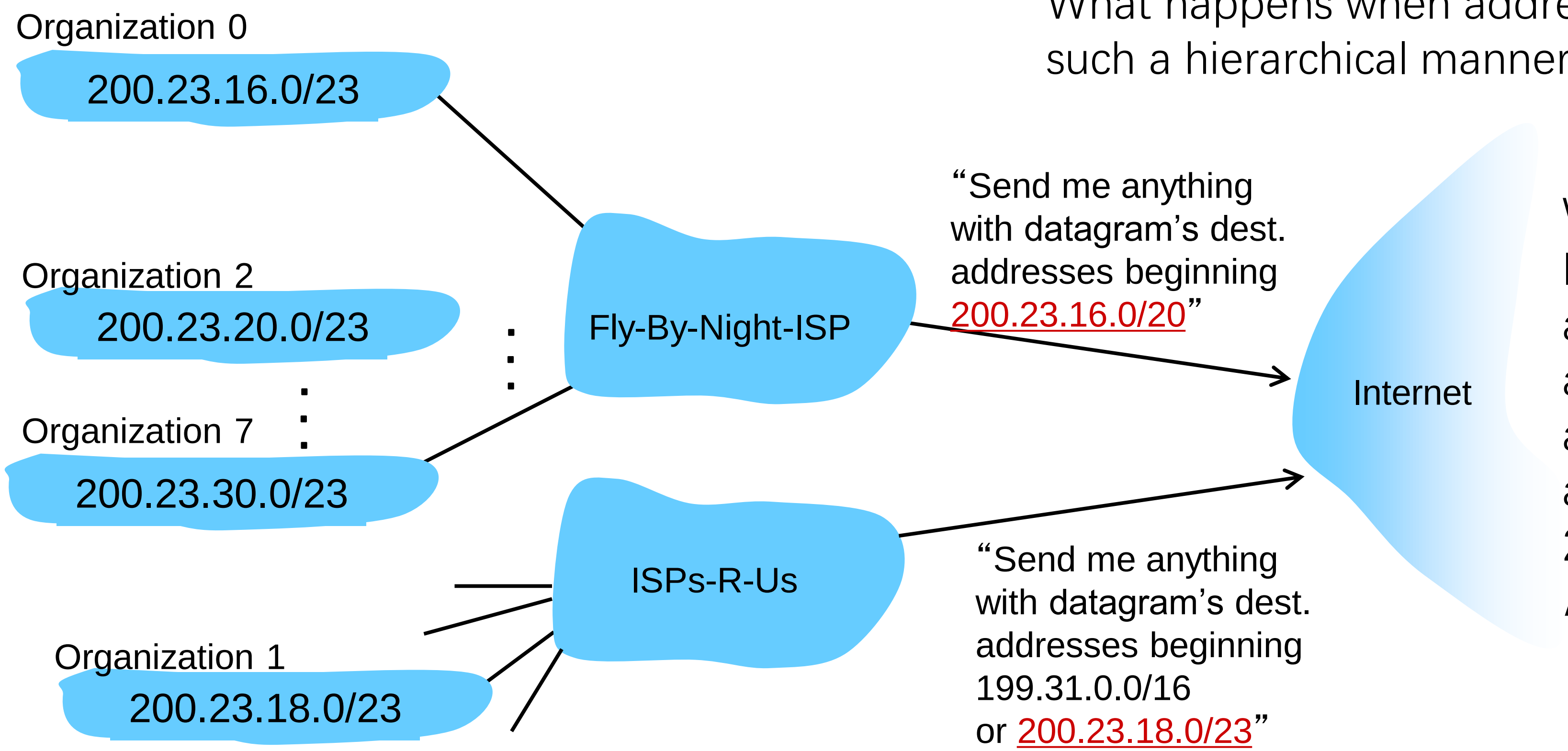
“Send me anything with datagram’s dest. addresses beginning 200.23.16.0/20”

“Send me anything with datagram’s dest. addresses beginning 199.31.0.0/16”

Hierarchical addressing: more specific routes

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

What happens when addresses are not allocated in such a hierarchical manner? See this example.



"Send me anything with datagram's dest. addresses beginning 200.23.16.0/20"

"Send me anything with datagram's dest. addresses beginning 199.31.0.0/16 or 200.23.18.0/23"

When other routers in the Internet see the address blocks 200.23.16.0/20 and 200.23.18.0/23 and want to route to an address in the block 200.23.18.0/23, they use ***longest prefix matching***

IP addressing

Q: **How does an ISP get block of addresses?**

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

- Allocates addresses
- Manages DNS
- Assigns domain names, resolves disputes

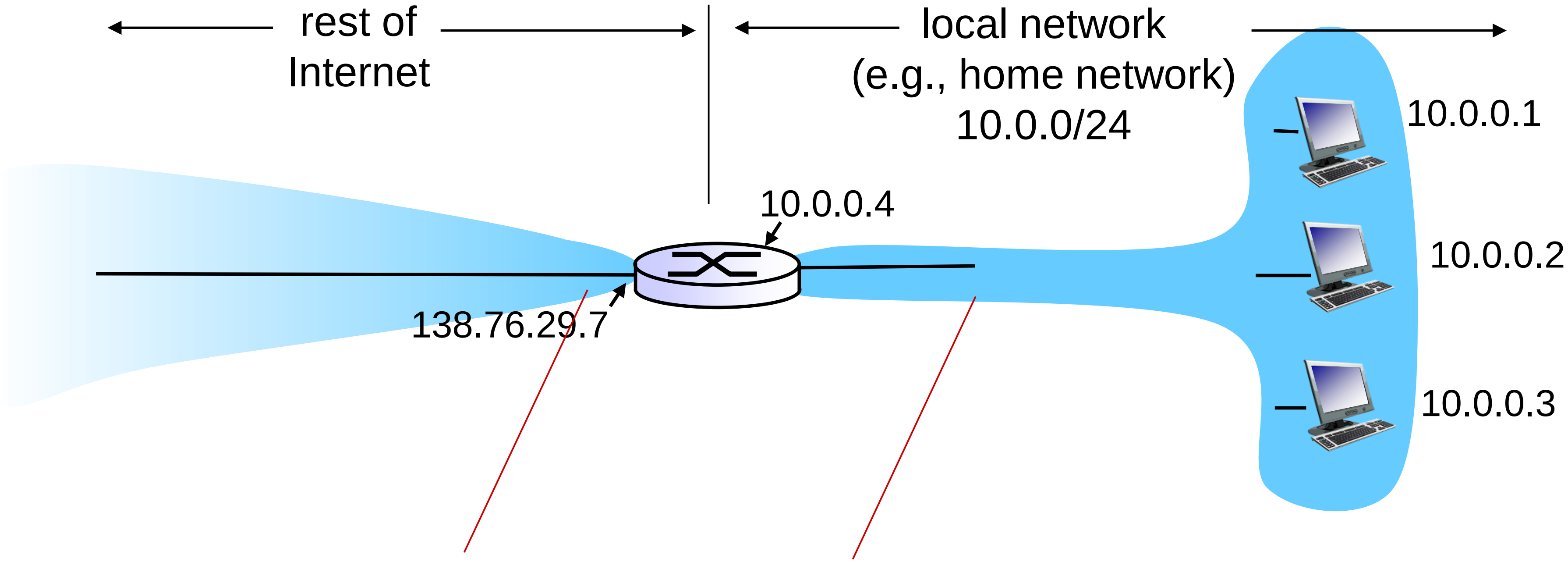
Lecture 7 – Network Layer Data Plane (2)

- **Roadmap**

1. IPv4 addressing
2. NAT
3. IPv6
4. Generalized Forward and SDN

NAT: network address translation

Why NAT? Because if the subnet grows bigger, it is hard to change the allocated address block if the ISP had already allocated the contiguous portions of address range. Also, a typical network admin is hard to estimate how many future addresses needed in the first place.



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 or visible by outside world (a security plus)

NAT: network address translation

If all datagrams arriving at the NAT router from the WAN have the same destination IP address (specifically, that of the WAN-side interface of the NAT router), then how does the router know the internal host to which it should forward a given datagram? The answer is to use a **NAT translation table.**

Implementation: NAT router must:

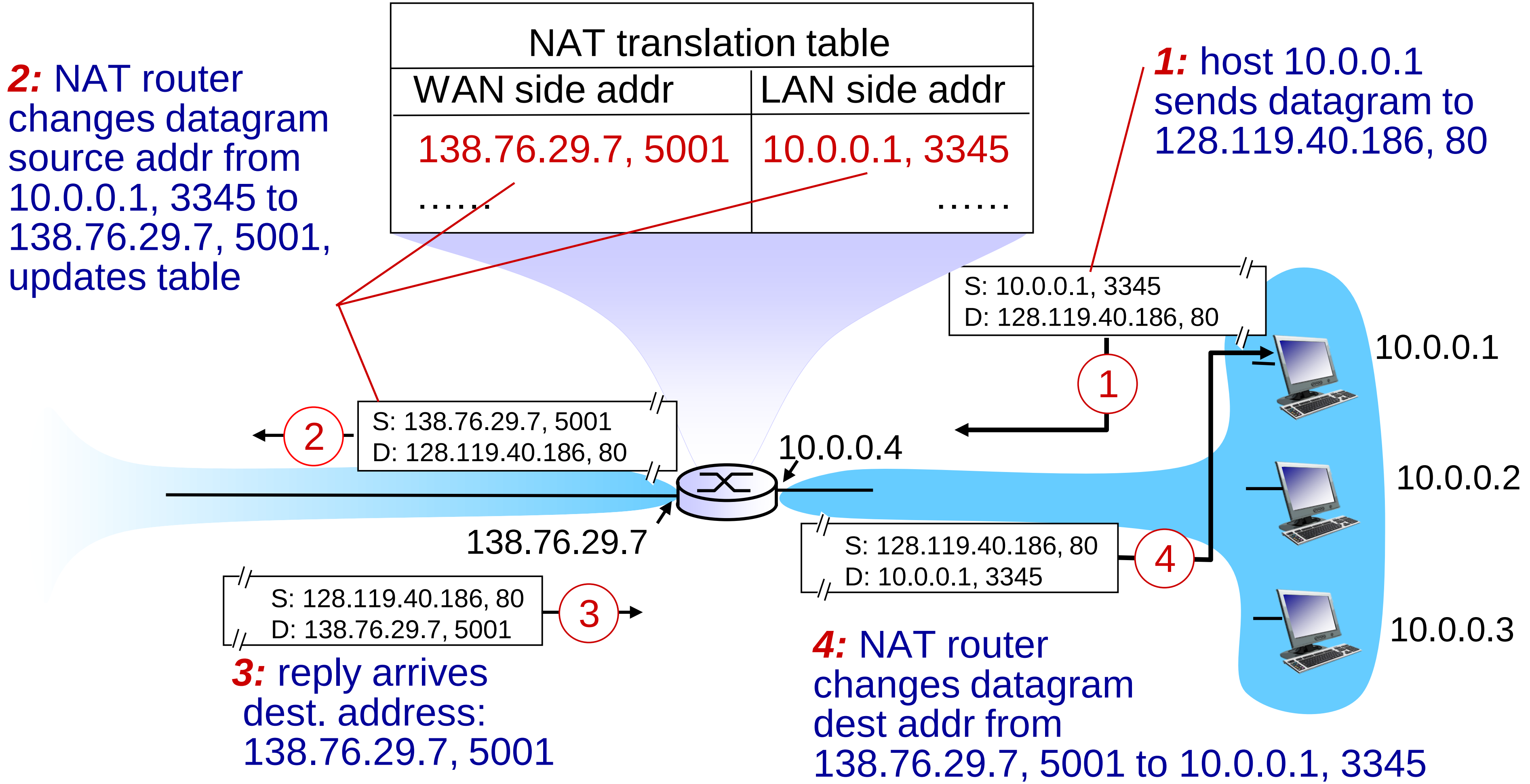
- *Remember (in **NAT translation table**)* every (source IP address, port #) to (NAT IP address, new port #) translation pair

- *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

- *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

S: 10.0.0.1, 3345
D: 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

S: 138.76.29.7, 5001
D: 128.119.40.186, 80

3: reply arrives dest. address: 138.76.29.7, 5001

S: 128.119.40.186, 80
D: 138.76.29.7, 5001

4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

S: 128.119.40.186, 80
D: 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
  - Address shortage should be solved by IPv6
  - Violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications. A peer behind NAT can only work as a client rather than a server.
    - NAT traversal is a technical solution to this problem.
  - NAT traversal
    - What if client wants to connect to server behind NAT?
    - Port Mapping is a solution.

Lecture 7 – Network Layer Data Plane (2)

- **Roadmap**
  1. IPv4 addressing
  2. NAT
  3. IPv6
  4. Generalized Forward and SDN

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

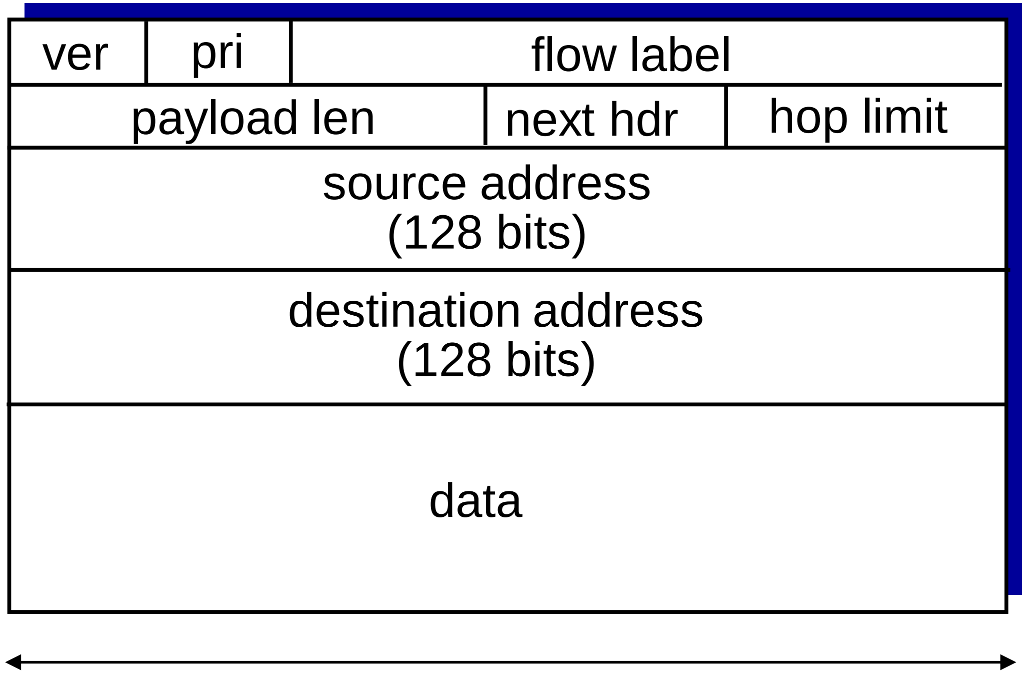
IPv6 datagram format:

- Fixed-length 40 byte header
- No fragmentation allowed (at intermediate routers, see "packet too big" new ICMP message type)

IPv6 datagram format

- ***Priority (Traffic Class):*** **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**



ver | pri | flow label

payload len | next hdr | hop limit

source address
(128 bits)

destination address
(128 bits)

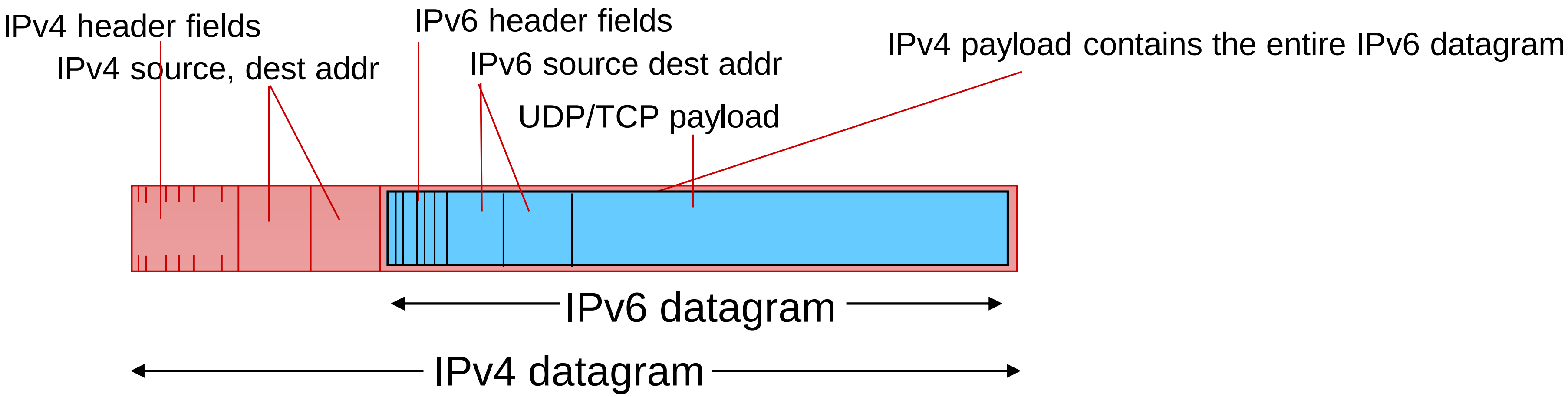
data

Other changes from IPv4

- ***ICMPv6:*** **new version of ICMP**
  - No Fragmentation or reassembly
  - If an IPv6 datagram received by a router is too large to be forwarded over the outgoing link, the router simply drops the datagram and sends a "**Packet Too Big**" ICMP error message back to the sender.
- ***Checksum*****: removed entirely to reduce processing time at each hop**
  - Because the transport-layer (for example, TCP and UDP) and link-layer (for example, Ethernet) protocols in the Internet layers perform check summing.
- ***Options:*** **allowed, but outside of header, can be indicated by the "Next Header" field**

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 (**set the protocol number field in IPv4 datagram at 41**)**
  - The intervening set of IPv4 routers between two IPv6 routers as a tunnel



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 and use***
  - 20 years and counting!
  - Think of application-level changes in last 20 years: WWW, Facebook, streaming media (Youtube), Skype, Twitter…
  - *Why?*

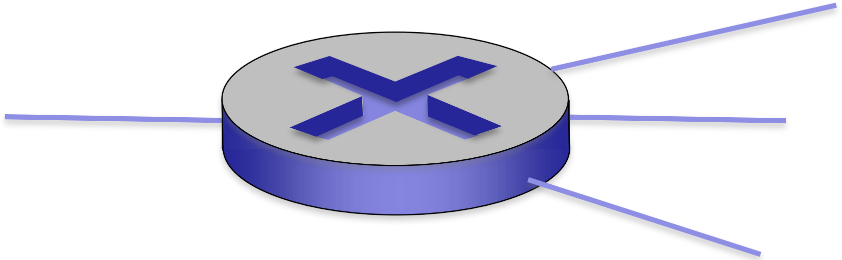
Lecture 7 – Network Layer Data Plane (2)

- **Roadmap**
  1. IPv4 addressing
  2. NAT
  3. IPv6
  4. Generalized Forward and SDN

Generalized Forwarding and SDN

Q:How does destination-based forwarding work?
A: lookup a destination (match), send the pkt (action).
That leads to a more general "match-plus-action" paradigm.

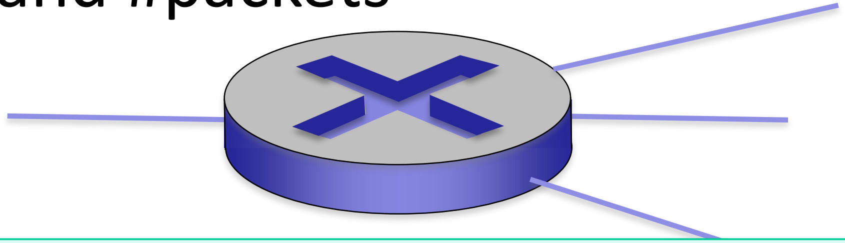
- ***Flow*: defined by header fields**
- **Generalized forwarding: simple packet-handling rules**
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping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



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

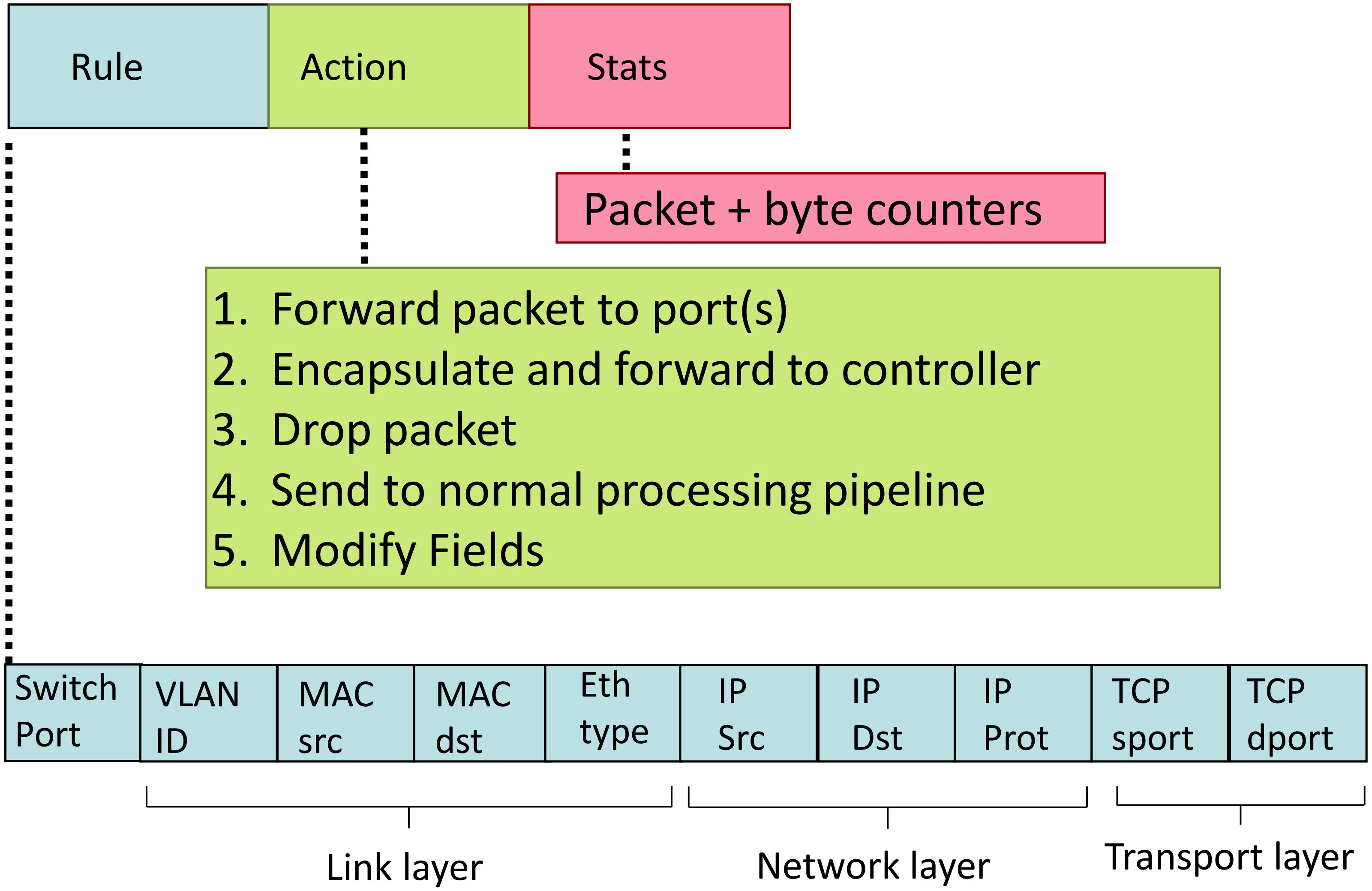
Generalized Forwarding and SDN

- ***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



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



Rule	Action	Stats

Packet + byte counters

1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields

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

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

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

Examples

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

OpenFlow abstraction

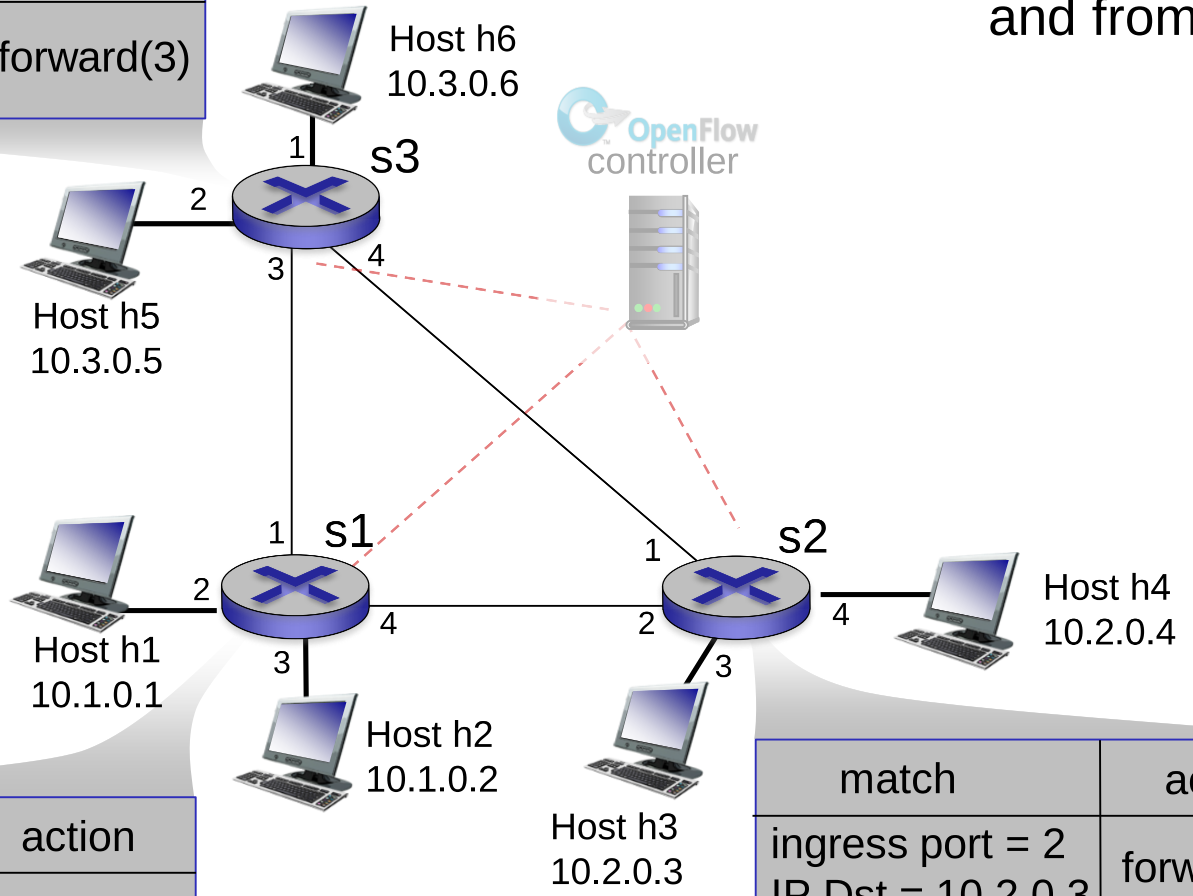
match+action: unifies different kinds of devices

- **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

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)

Lecture 7 – Network Layer Control Plane (1)

- **Roadmap**
  1. Overview
  2. Routing protocol (1) - Dijkstra’s algorithm

Network-layer functions

Recall: two network-layer functions:

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

data plane

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

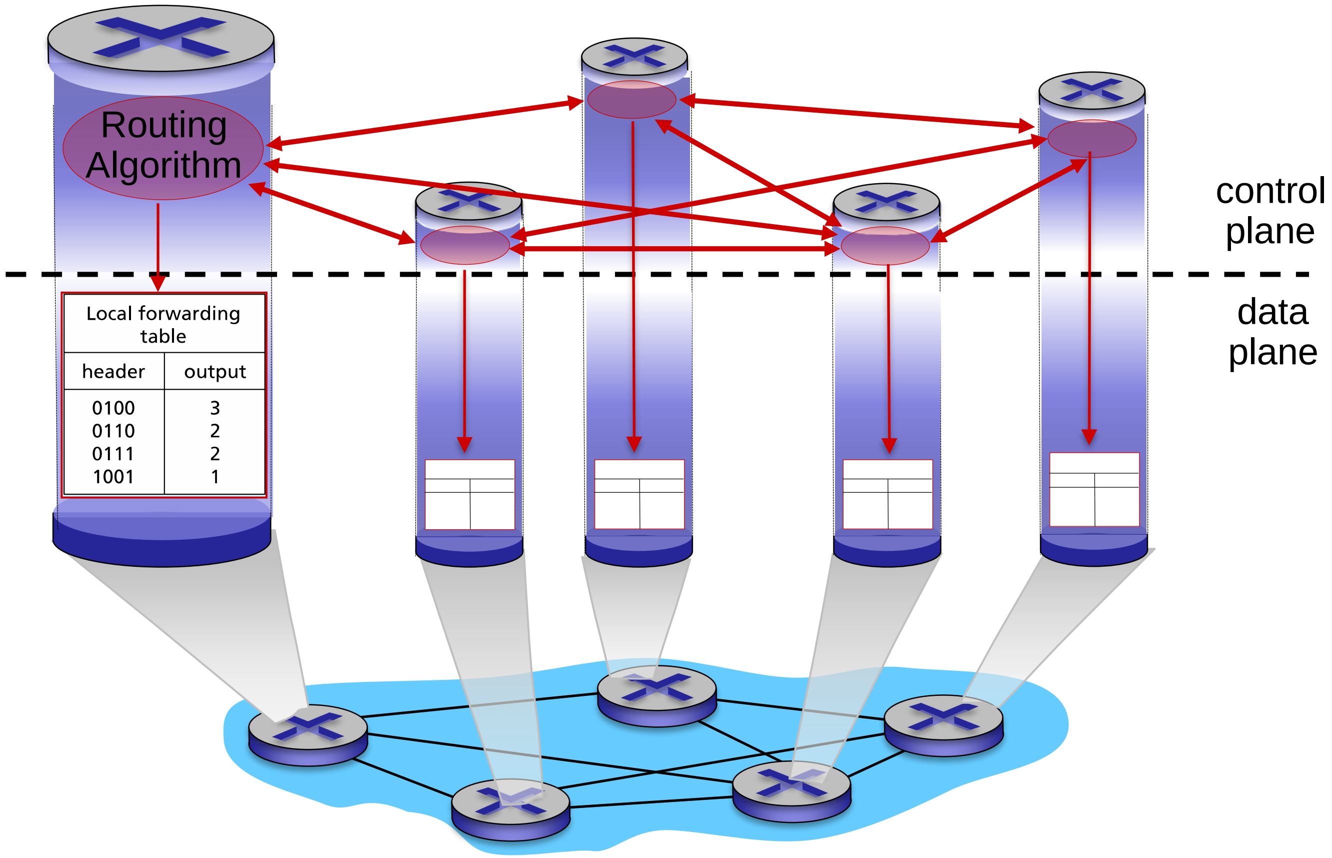
control plane

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

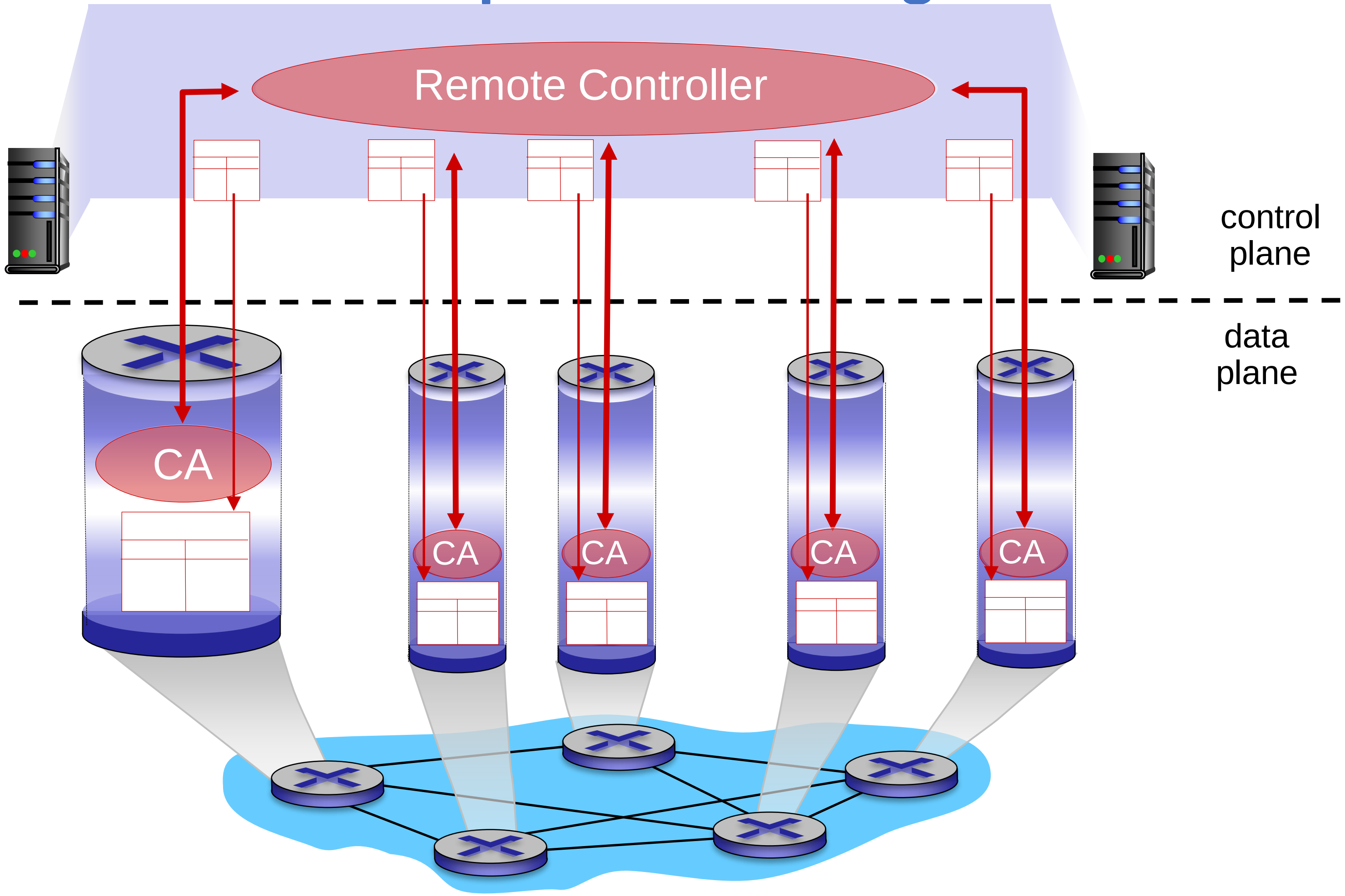


Local forwarding table

header	output
0100	3
0110	2
0111	2
1001	1

Logically centralized control plane

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



Lecture 7 – Network Layer Control Plane (1)

- **Roadmap**
  1. Overview
  2. Routing protocol (1) - Dijkstra’s algorithm

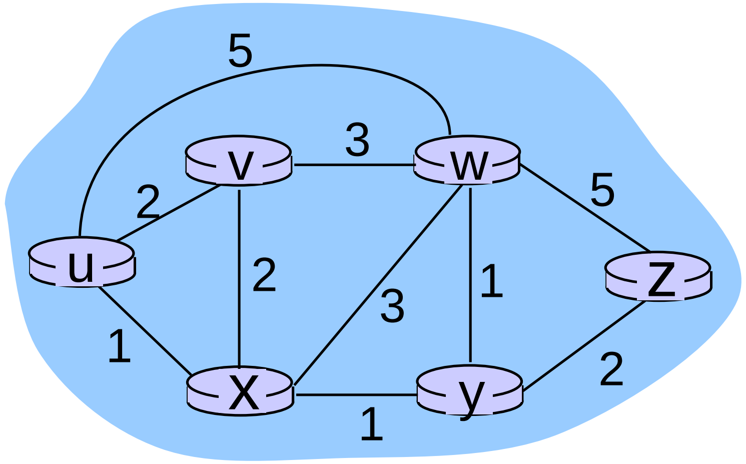
Routing protocols

Routing protocol goal: **determine "good" paths (equivalently, routes), from sending host 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

A graph is often used to formulate routing problems..



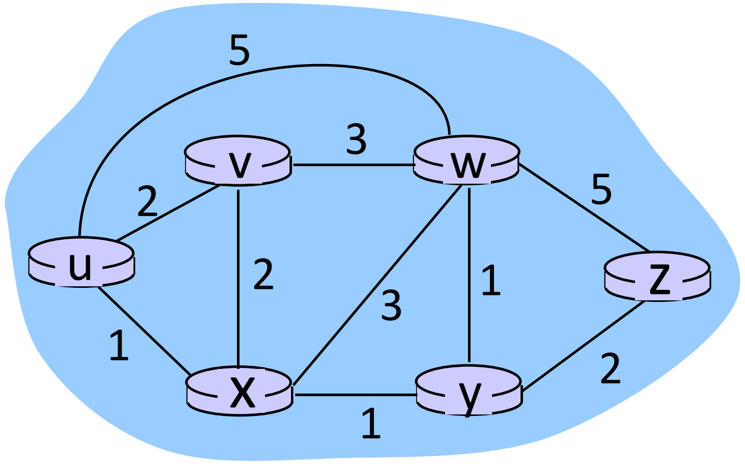
graph: G = (N,E)

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

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

aside: graph abstraction is useful in other network contexts, e.g., P2P, where *N* is set of peers and *E* is set of TCP connections

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 directly 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/centralized:

- All routers have complete topology, link cost info
- "link state" algorithms (e.g., Dijkstra)

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

A link-state routing algorithm - Dijkstra

Dijkstra's algorithm

- **Net topology and link costs known to all nodes**
  - Accomplished via "link state broadcast"
  - All nodes have same info
- **Computes least cost paths from one node ("source") to all other nodes**
  - Gives *forwarding table* for that node
- **Iterative: after k iterations, know least cost path to k dest.'s**

Notation:

- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to dest. v
- N': set of nodes whose least cost path definitively known

Dijsktra's algorithm

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
      then $D(v) = c(u,v)$
    else $D(v) = \infty$

Loop
  find w not in N' such that D(w) is a minimum
  add w to N'
  update D(v) for all v adjacent to w and not in N' :
      $D(v) = \min( D(v), D(w) + c(w,v) )$
  /* new cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
until all nodes in N'
```

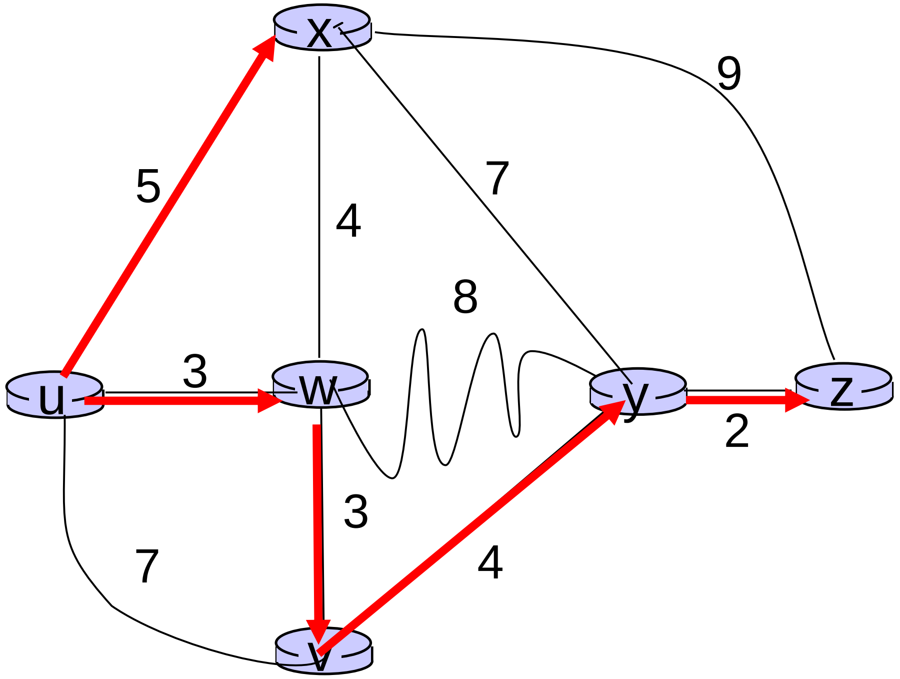
Notation:

- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijkstra’s algorithm: example 1

computing routing table in u:

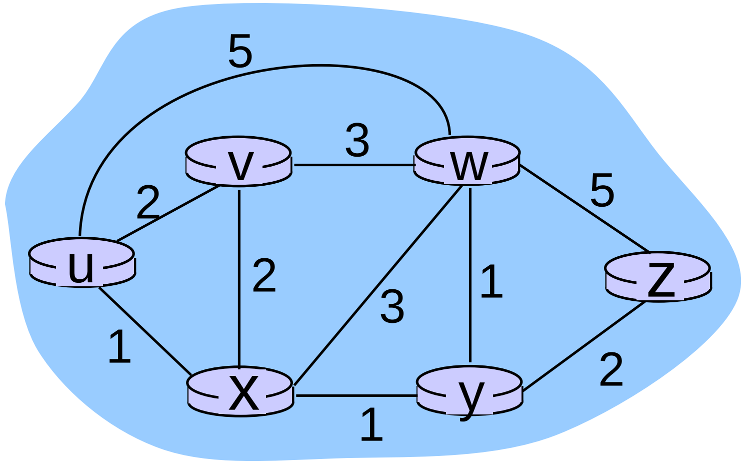
		D(v)	D(w)	D(x)	D(y)	D(z)
Step	N'	p(v)	p(w)	p(x)	p(y)	p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					



notes:

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)

Dijkstra’s algorithm: example 2

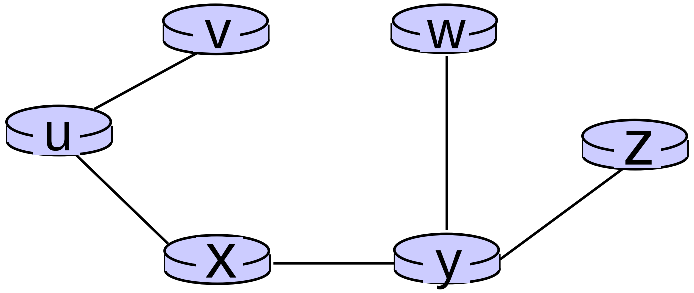


computing routing table in u:

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

Dijkstra’s algorithm: example 2 - result

resulting shortest-path tree from u:



resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Thanks.

- **Addresses**
  - Email: Wenjun.Fan@xjtlu.edu.cn
  - Office: EE 214
- **Office hours**
  - Monday: 12:00 – 13:00
  - Tuesday: 12:00 – 13:00