

# Chapter 4

## Network Layer

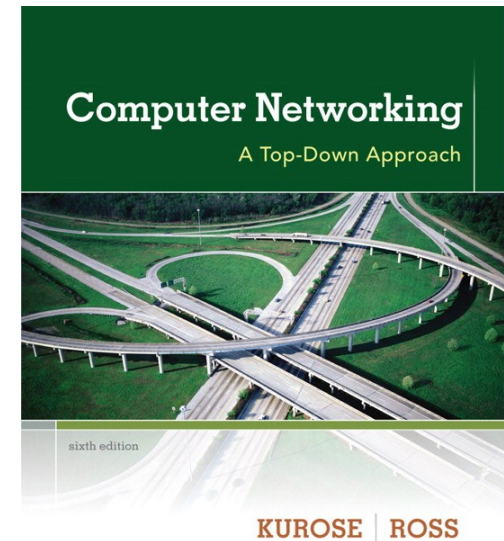
### A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- ❖ If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- ❖ If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

© All material copyright 1996-2013  
J.F Kurose and K.W. Ross, All Rights Reserved



**Computer  
Networking: A Top  
Down Approach**  
6<sup>th</sup> edition  
Jim Kurose, Keith Ross  
Addison-Wesley  
March 2012

# Chapter 4: network layer

## *chapter goals:*

- ❖ understand principles behind network layer services:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - routing (path selection)
  - broadcast, multicast
- ❖ instantiation, implementation in the Internet

# Chapter 4: outline

## 4.1 introduction

## 4.2 virtual circuit and datagram networks

## 4.3 what's inside a router

## 4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

## 4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

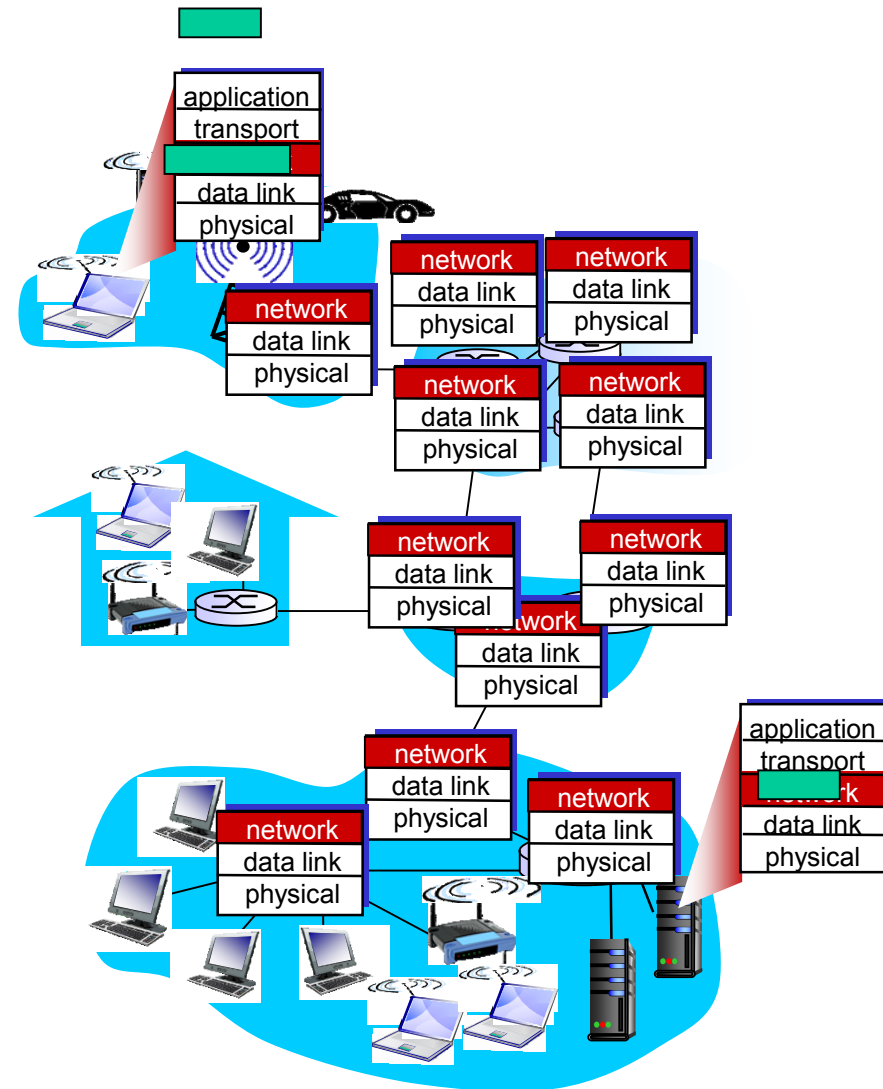
## 4.6 routing in the Internet

- RIP
- OSPF
- BGP

## 4.7 broadcast and multicast routing

# Network layer

- ❖ transport segment from sending to receiving host
- ❖ on sending side encapsulates segments into datagrams
- ❖ on receiving side, delivers segments to transport layer
- ❖ network layer protocols in *every* host, router
- ❖ router examines header fields in all IP datagrams passing through it



# Two key network-layer functions

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

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

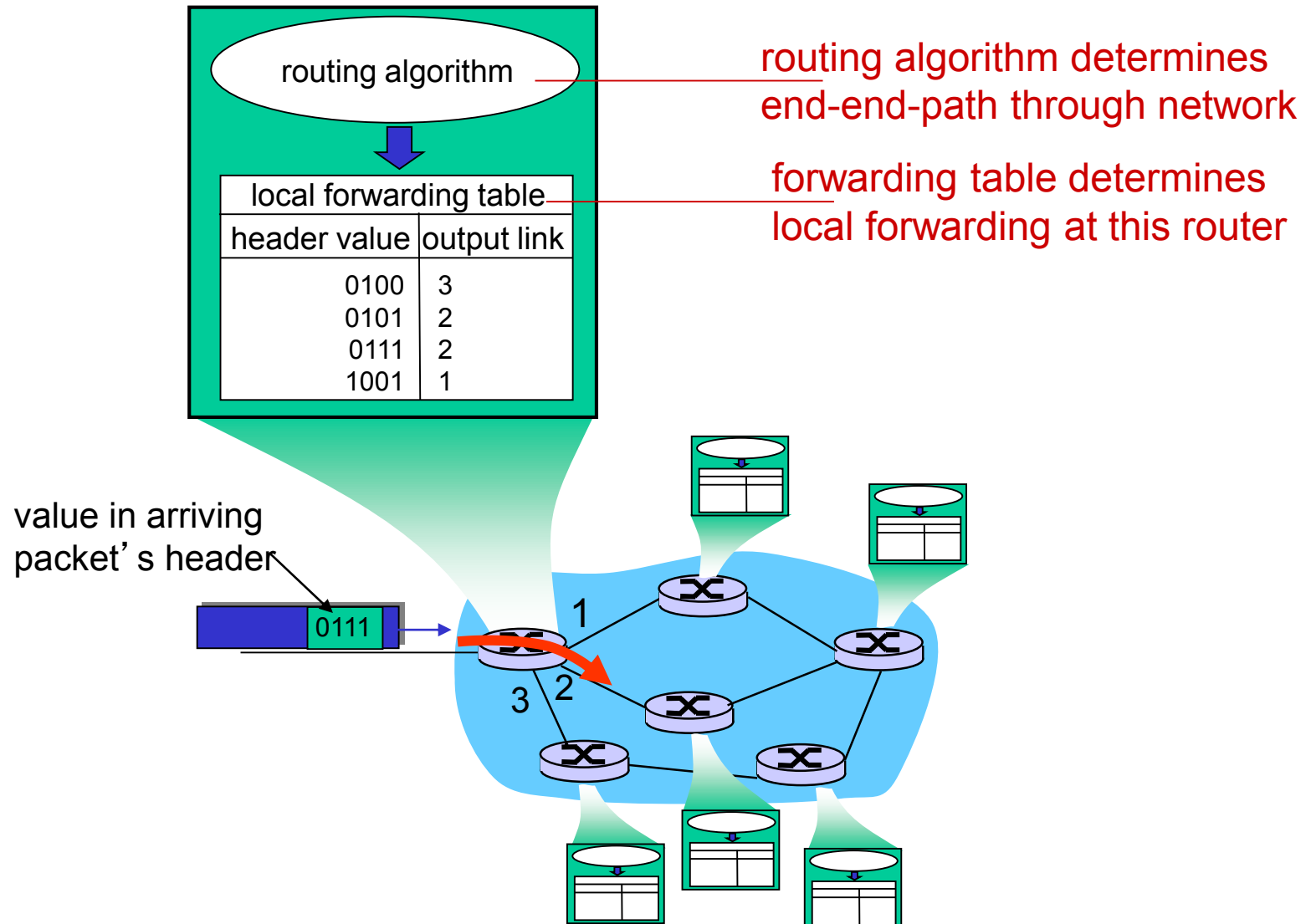
  - *routing algorithms*

*analogy:*

- ❖ *routing*: process of planning trip from source to dest

- ❖ *forwarding*: process of getting through single interchange

# Interplay between routing and forwarding



# Connection setup

- ❖ 3<sup>rd</sup> important function in *some* network architectures:
  - ATM, frame relay, X.25
- ❖ before datagrams flow, two end hosts *and* intervening routers establish virtual connection
  - routers get involved
- ❖ network vs transport layer connection service:
  - *network*: between two hosts (may also involve intervening routers in case of VCs)
  - *transport*: between two processes

# Network service model

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

*example services for individual datagrams:*

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

*example services for a flow of datagrams:*

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



# Chapter 4: outline

## 4.1 introduction

## 4.2 virtual circuit and datagram networks

## 4.3 what's inside a router

## 4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

## 4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

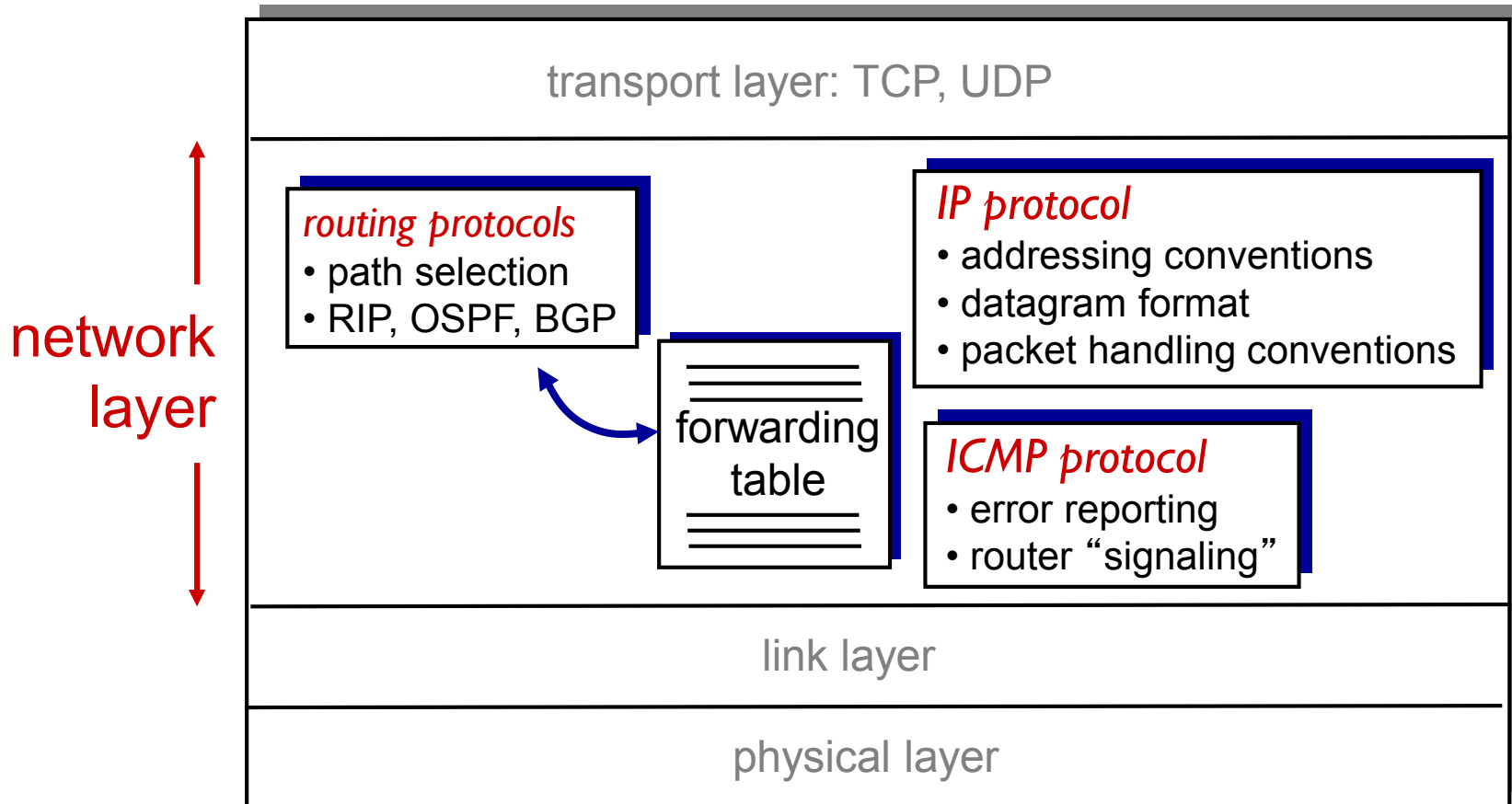
## 4.6 routing in the Internet

- RIP
- OSPF
- BGP

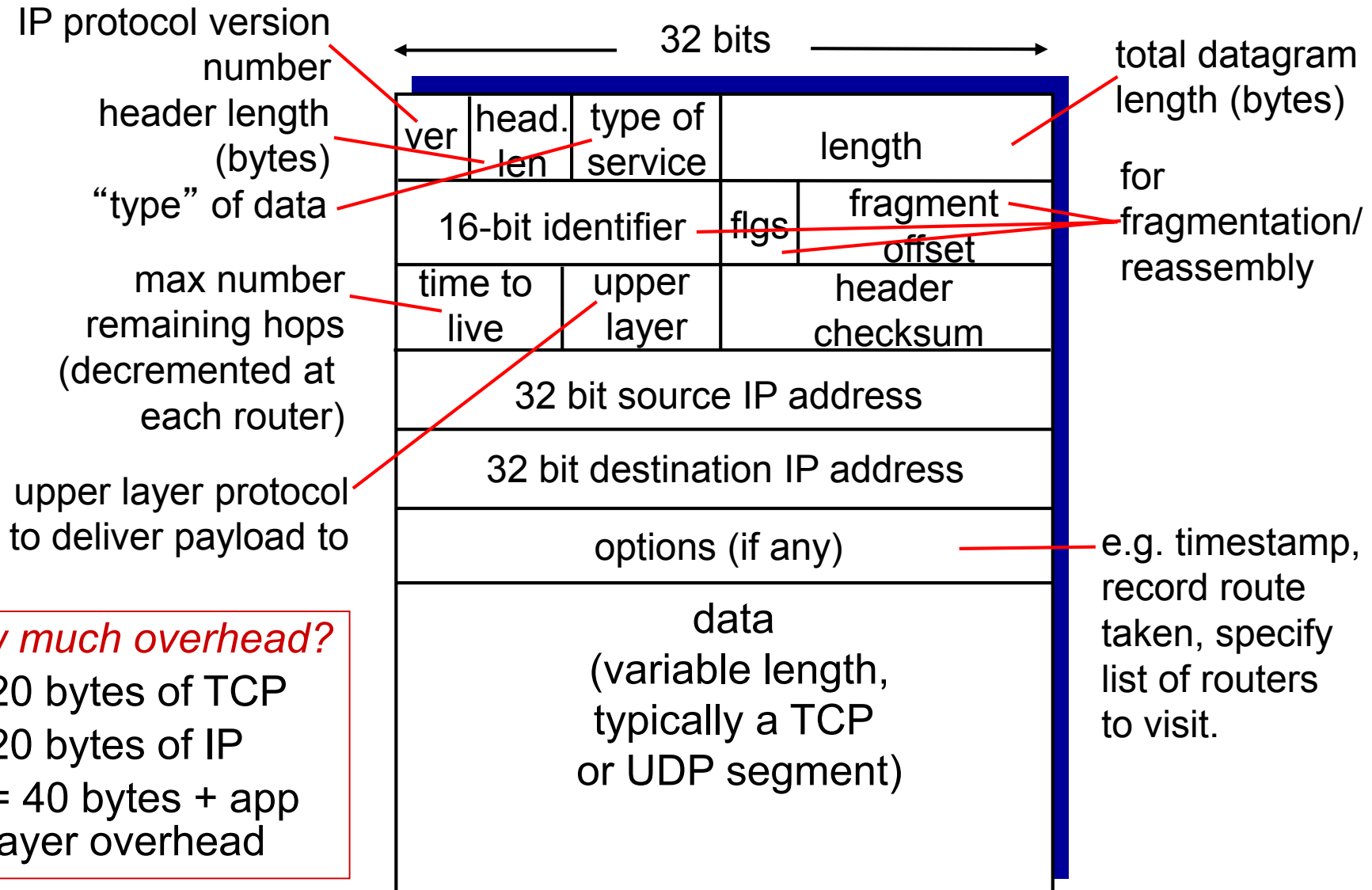
## 4.7 broadcast and multicast routing

# The Internet network layer

host, router network layer functions:

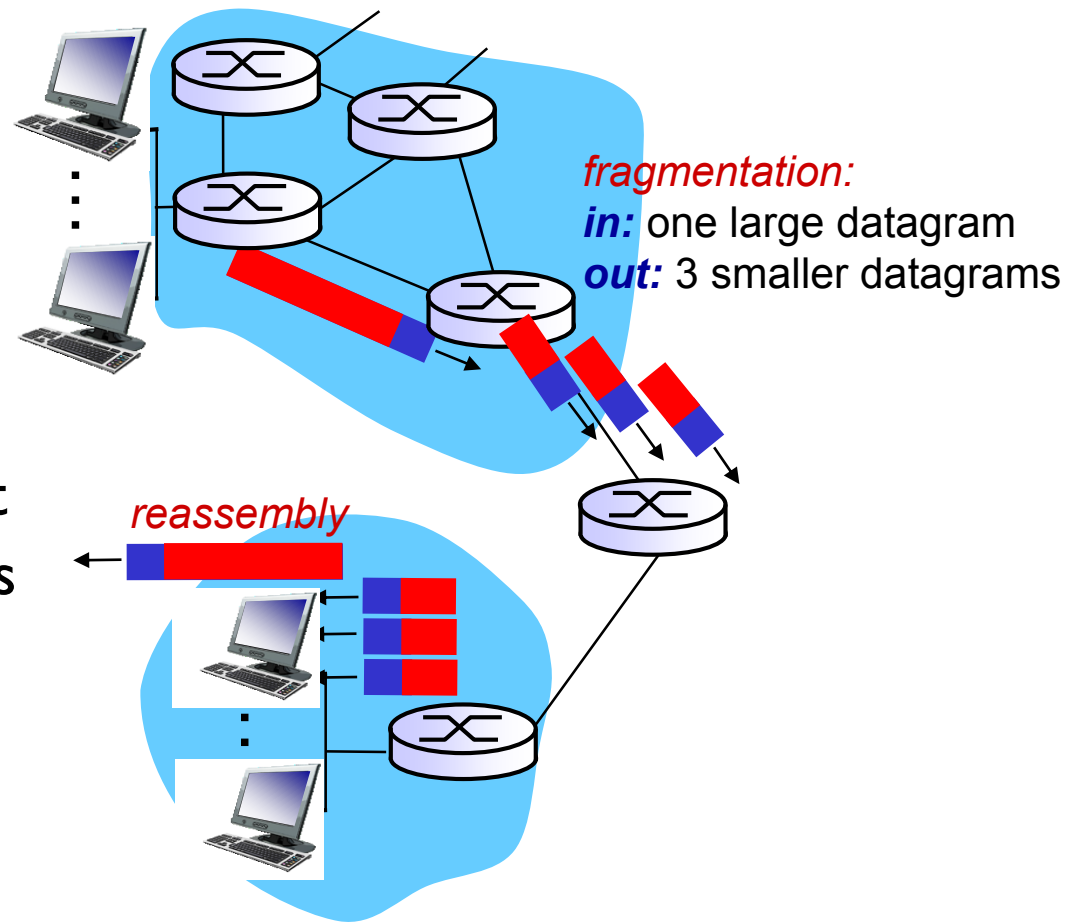


# IP datagram format



# IP fragmentation, reassembly

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



# IP fragmentation, reassembly

## *example:*

- ❖ 4000 byte datagram
- ❖ MTU = 1500 bytes

	length	ID	fragflag	offset	
	=4000	=x	=0	=0	

*one large datagram becomes  
several smaller datagrams*

1480 bytes in  
data field

offset =  
 $1480/8$

	length	ID	fragflag	offset	
	=1500	=x	=1	=0	

	length	ID	fragflag	offset	
	=1500	=x	=1	=185	

	length	ID	fragflag	offset	
	=1040	=x	=0	=370	

# Chapter 4: outline

## 4.1 introduction

## 4.2 virtual circuit and datagram networks

## 4.3 what's inside a router

## 4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

## 4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

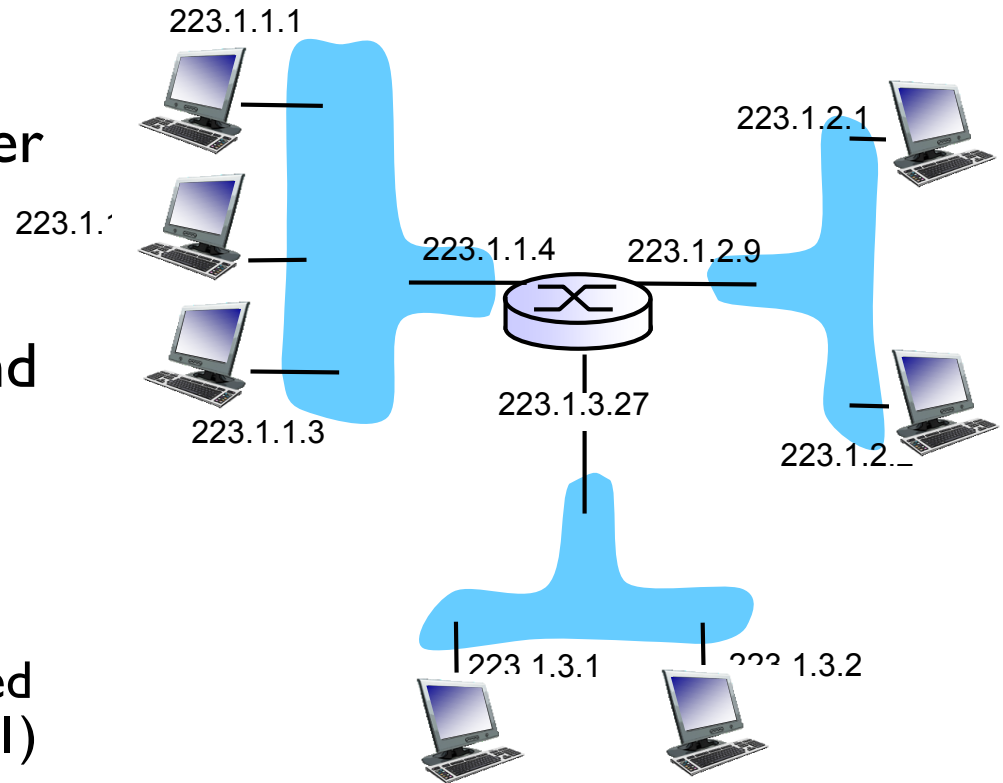
## 4.6 routing in the Internet

- RIP
- OSPF
- BGP

## 4.7 broadcast and multicast routing

# IP addressing: introduction

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



$$223.1.1.1 = \underbrace{11011111}_{223} \underbrace{00000001}_1 \underbrace{00000001}_1 \underbrace{00000001}_1$$

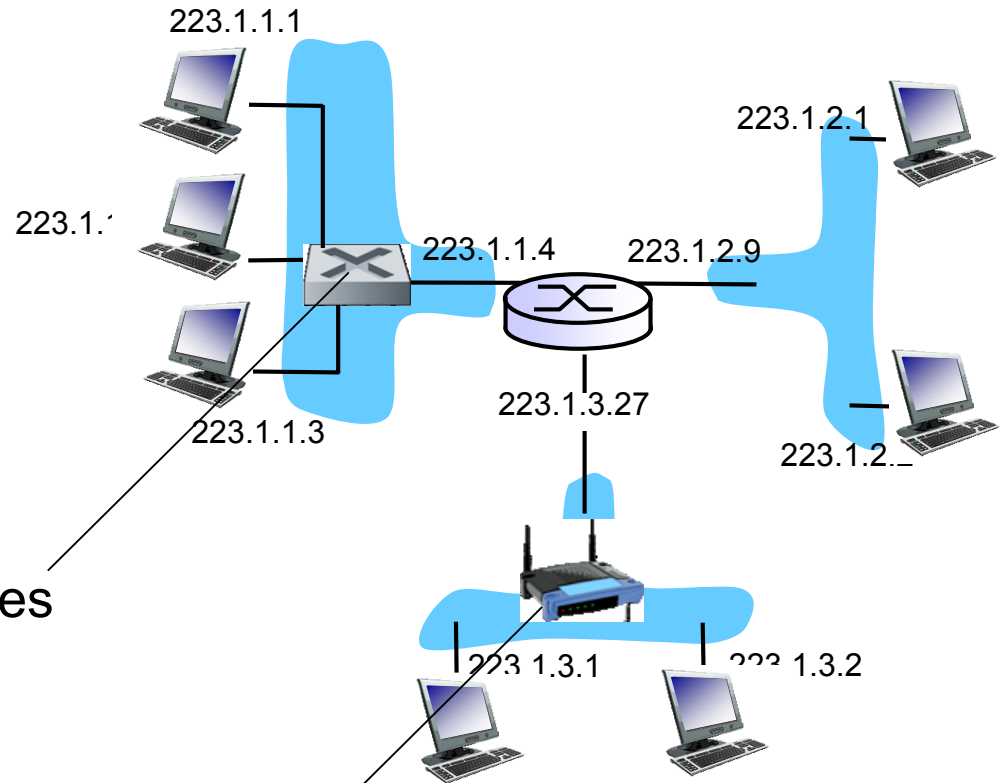
# IP addressing: introduction

*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*



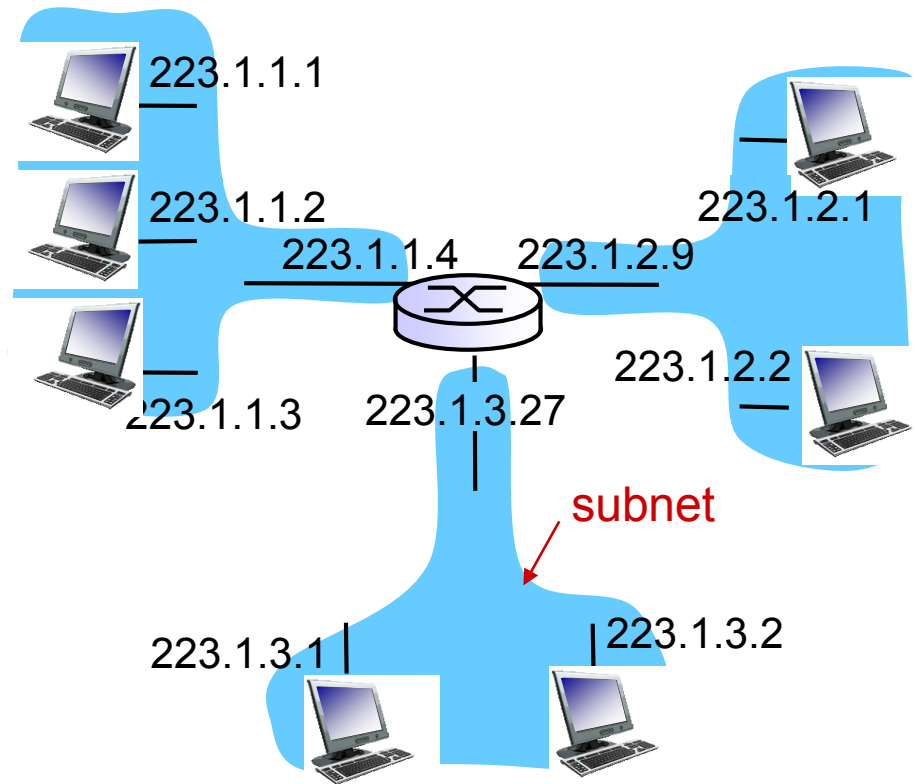
# Subnets

## ❖ IP address:

- subnet part - high order bits
- host part - low order bits

## ❖ *what 's a subnet ?*

- Is a subnetwork
- device interfaces with same subnet part of IP address
- can physically reach each other *without intervening router*

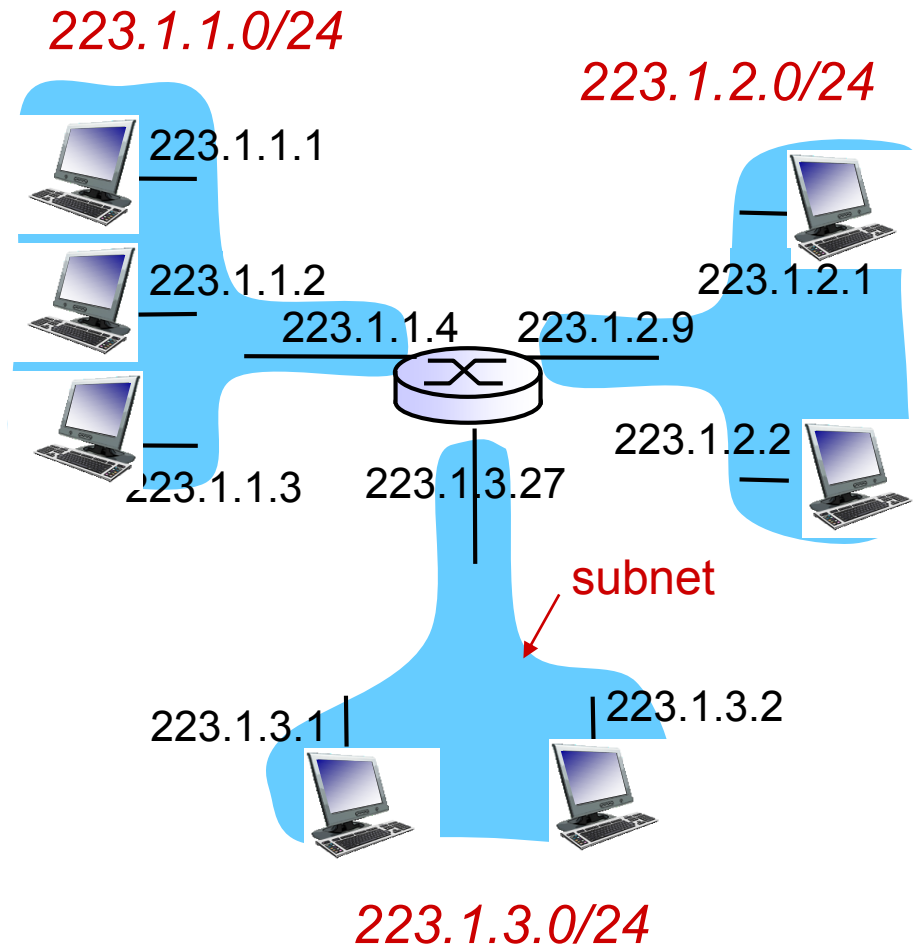


network consisting of 3 subnets

# Subnets

## *recipe*

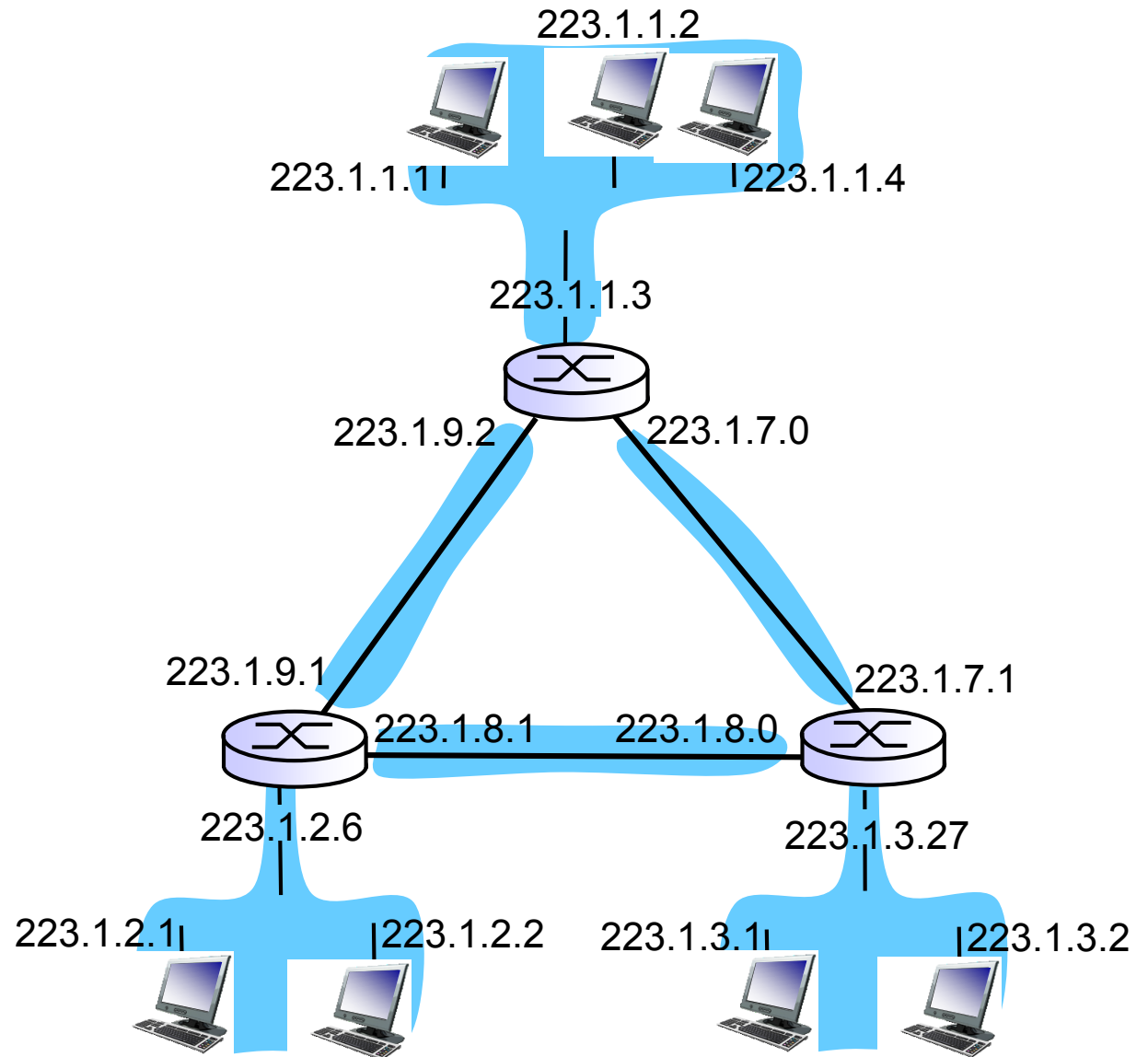
- ❖ to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- ❖ each isolated network is called a *subnet*



subnet mask: /24

# Subnets

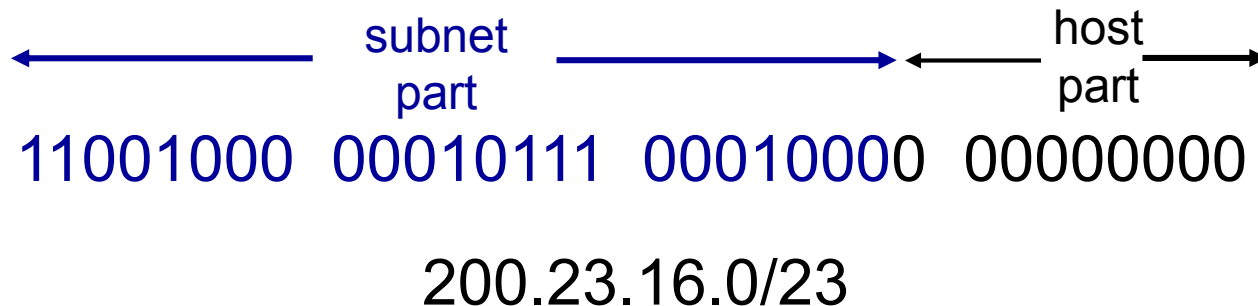
how many?



# IP addressing: CIDR

## CIDR: Classless InterDomain Routing

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



# 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 address from as server
  - “plug-and-play”

# DHCP: Dynamic Host Configuration Protocol

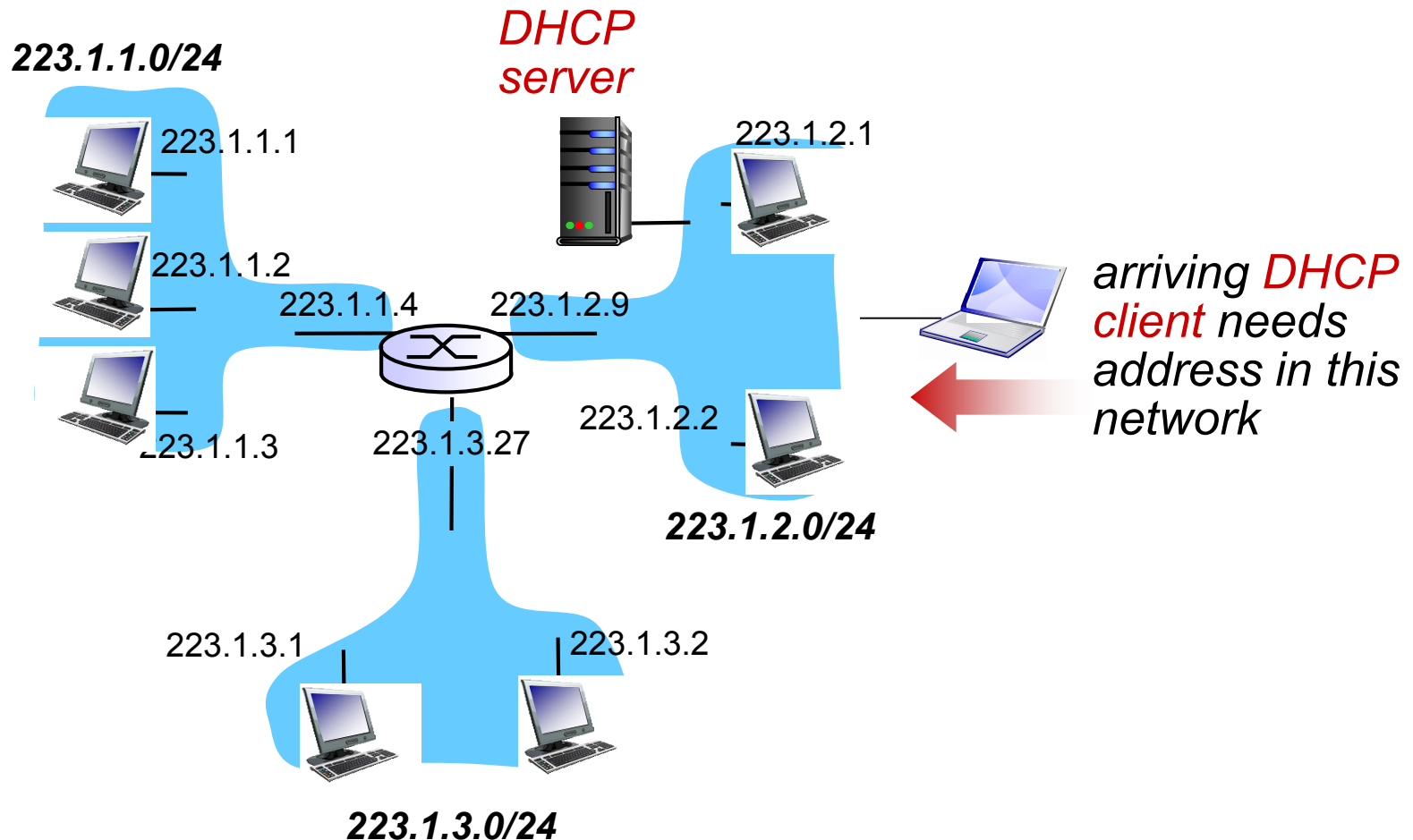
*goal:* allow host to *dynamically* obtain its IP address from network server when it joins network

- 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

DHCP discover

arriving  
client



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!

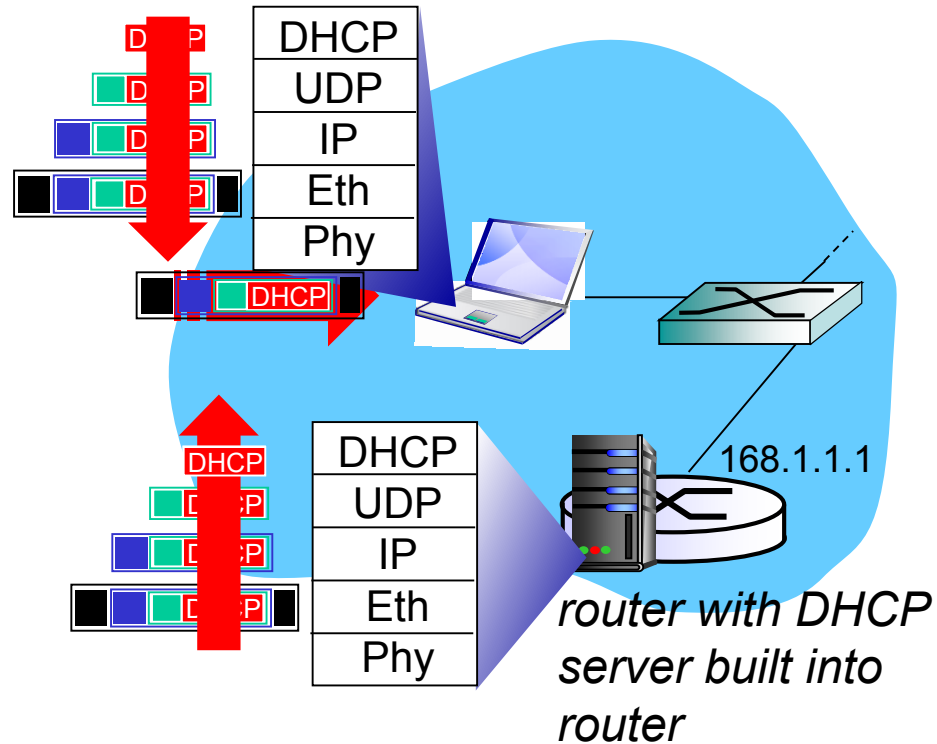


# DHCP: more than IP addresses

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

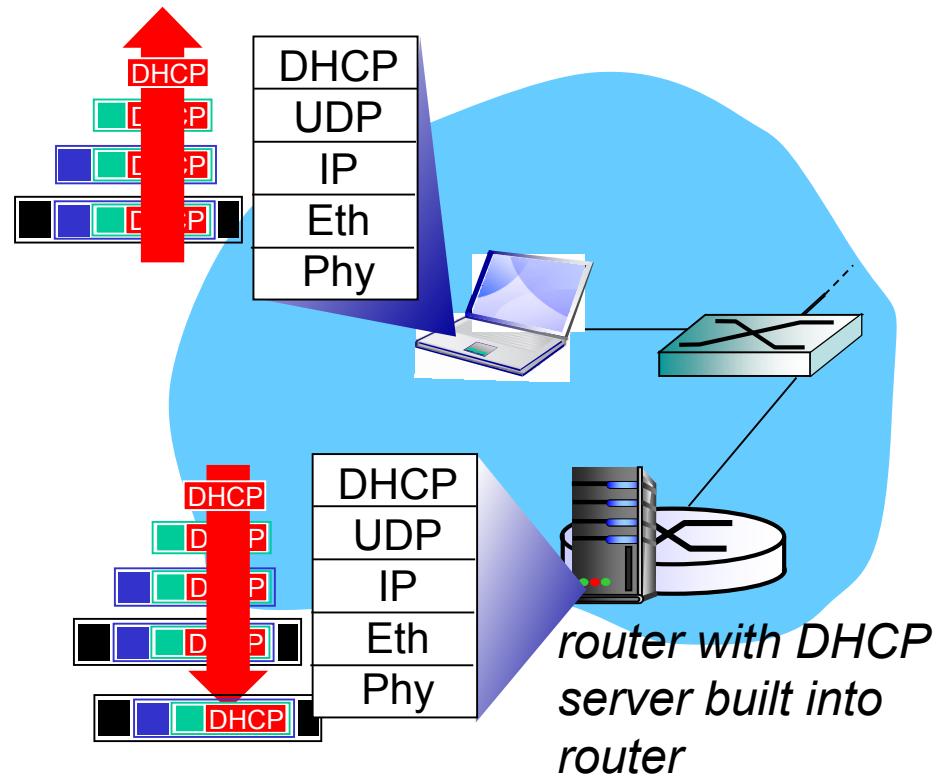
- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

# DHCP: example



- ❖ 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: FFFFFFFF) on LAN, received at router running DHCP server
- ❖ Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

# DHCP: example



- ❖ DCP 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 DNS server, IP address of its first-hop router

# IP addresses: how to get one?

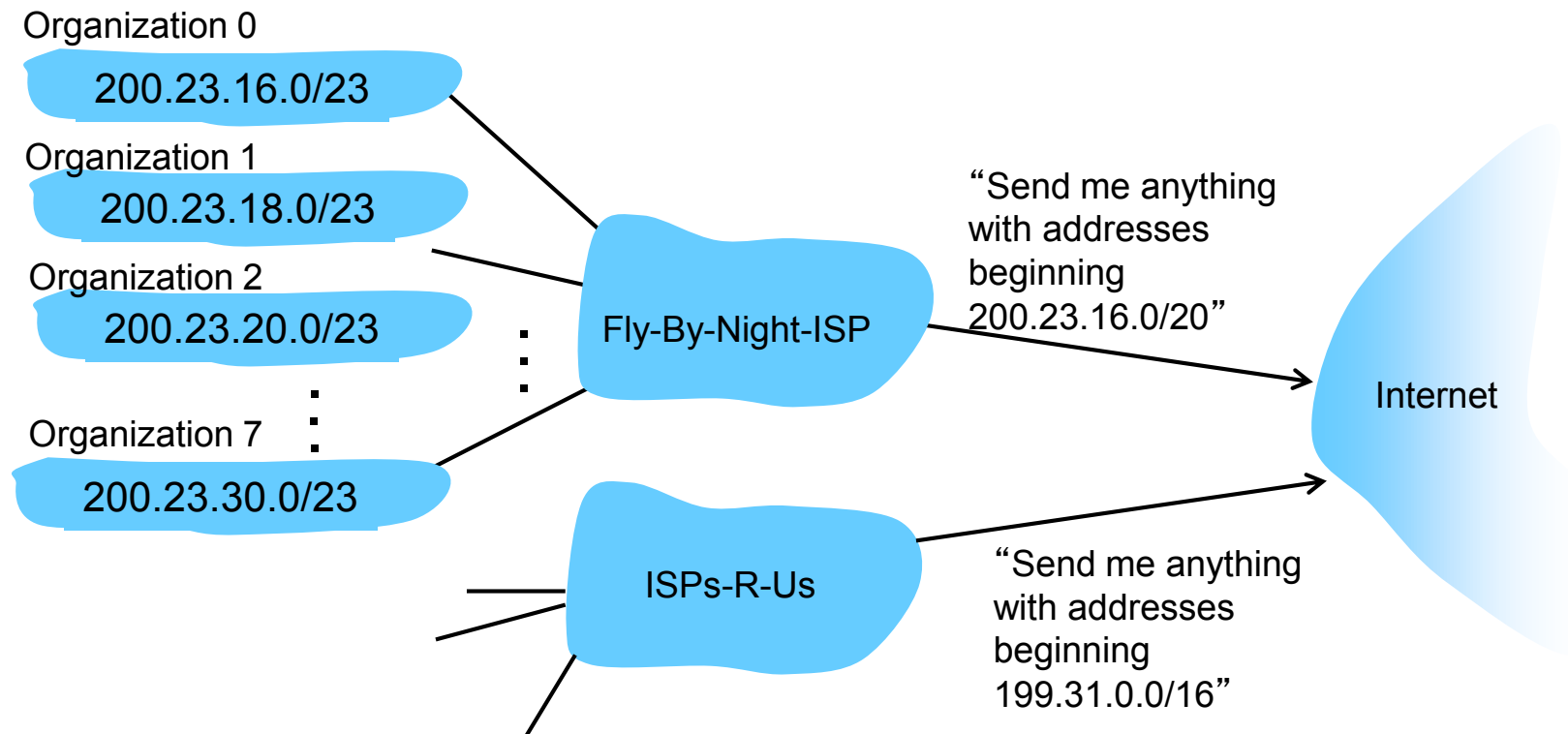
**Q:** how does *network* get subnet part of IP addr?

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

ISP's block	<u>11001000</u>	<u>00010111</u>	<u>00010000</u>	00000000	200.23.16.0/20
Organization 0	<u>11001000</u>	<u>00010111</u>	<u>00010000</u>	00000000	200.23.16.0/23
Organization 1	<u>11001000</u>	<u>00010111</u>	<u>00010010</u>	00000000	200.23.18.0/23
Organization 2	<u>11001000</u>	<u>00010111</u>	<u>00010100</u>	00000000	200.23.20.0/23
...	.....			....	....
Organization 7	<u>11001000</u>	<u>00010111</u>	<u>00011110</u>	00000000	200.23.30.0/23

# Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



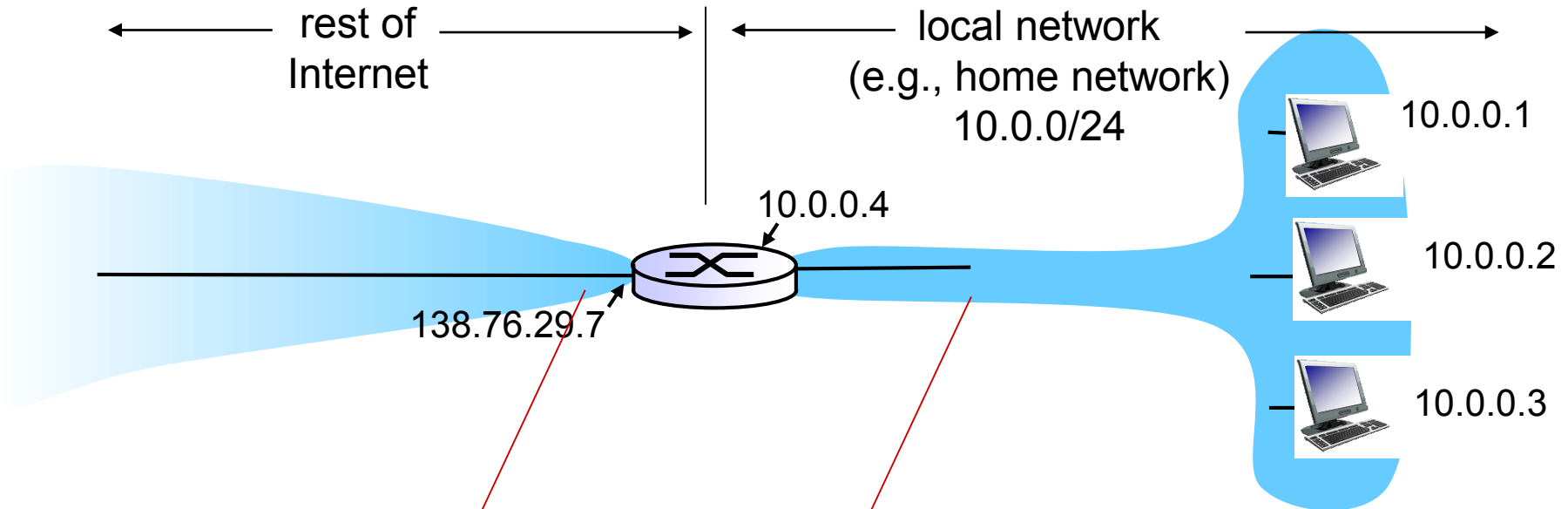
# IP addressing: the last word...

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

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

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

# NAT: network address translation



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

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

# NAT: network address translation

*motivation:* local network uses just one IP address as far as outside world is concerned:

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

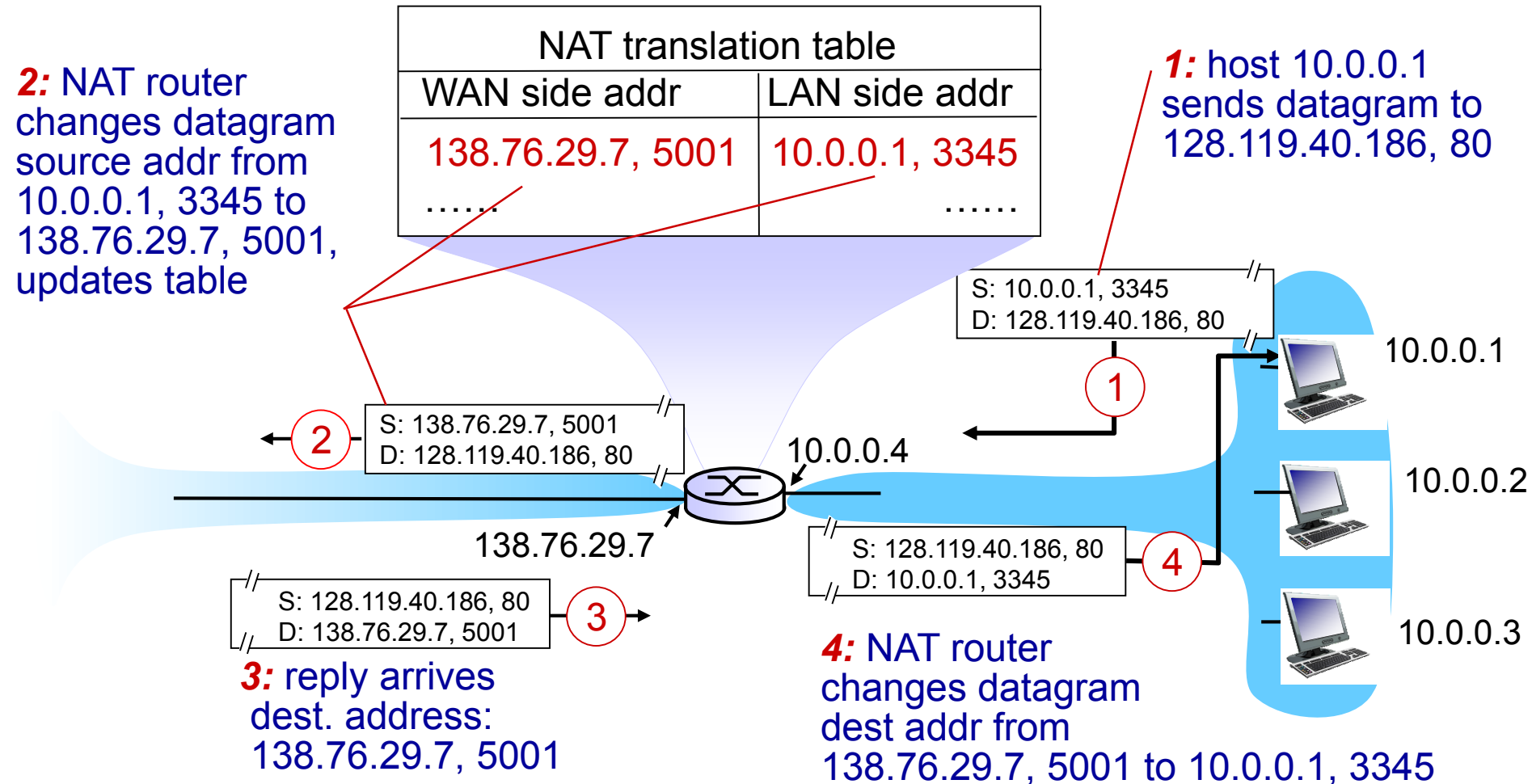


# NAT: network address translation

*implementation:* NAT router must:

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

# NAT: network address translation



# NAT: network address translation

- ❖ 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- ❖ NAT is controversial:
  - routers should only process up to layer 3
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - address shortage should instead be solved by IPv6

# Chapter 4: outline

## 4.1 introduction

## 4.2 virtual circuit and datagram networks

## 4.3 what's inside a router

## 4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

## 4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

## 4.6 routing in the Internet

- RIP
- OSPF
- BGP

## 4.7 broadcast and multicast routing

# ICMP: internet control message protocol

- ❖ used by hosts & routers to communicate network-level information

- error reporting:  
unreachable host, network, port, protocol
- echo request/reply (used by ping)

- ❖ network-layer “above” IP:

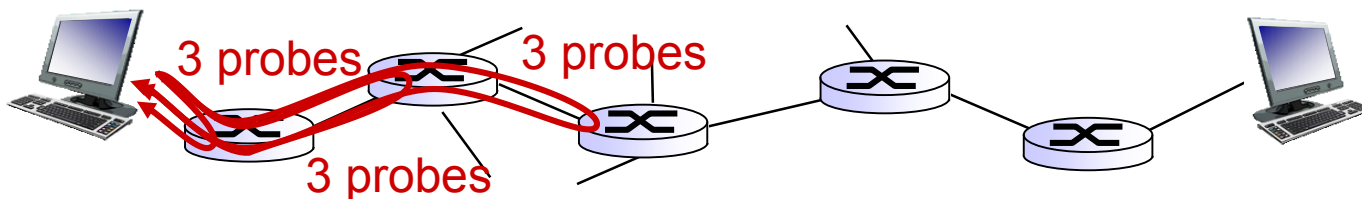
- ICMP msgs carried in IP datagrams

- ❖ **ICMP message:** type, code plus first 8 bytes of IP datagram causing error

<u>Type</u>	<u>Code</u>	<u>description</u>
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

# Traceroute and ICMP

- ❖ source sends series of UDP segments to dest
    - first set has TTL = 1
    - second set has TTL=2, etc.
    - unlikely port number
  - ❖ when  $n$ th set of datagrams arrives to  $n$ th router:
    - router discards datagrams
    - and sends source ICMP messages (type 11, code 0)
    - ICMP messages includes name of router & IP address
  - ❖ when ICMP messages arrives, source records RTTs
- stopping criteria:*
- ❖ UDP segment eventually arrives at destination host
  - ❖ destination returns ICMP “port unreachable” message (type 3, code 3)
  - ❖ source stops



# IPv6: motivation

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

## *IPv6 datagram format:*

- fixed-length 40 byte header
- no fragmentation allowed

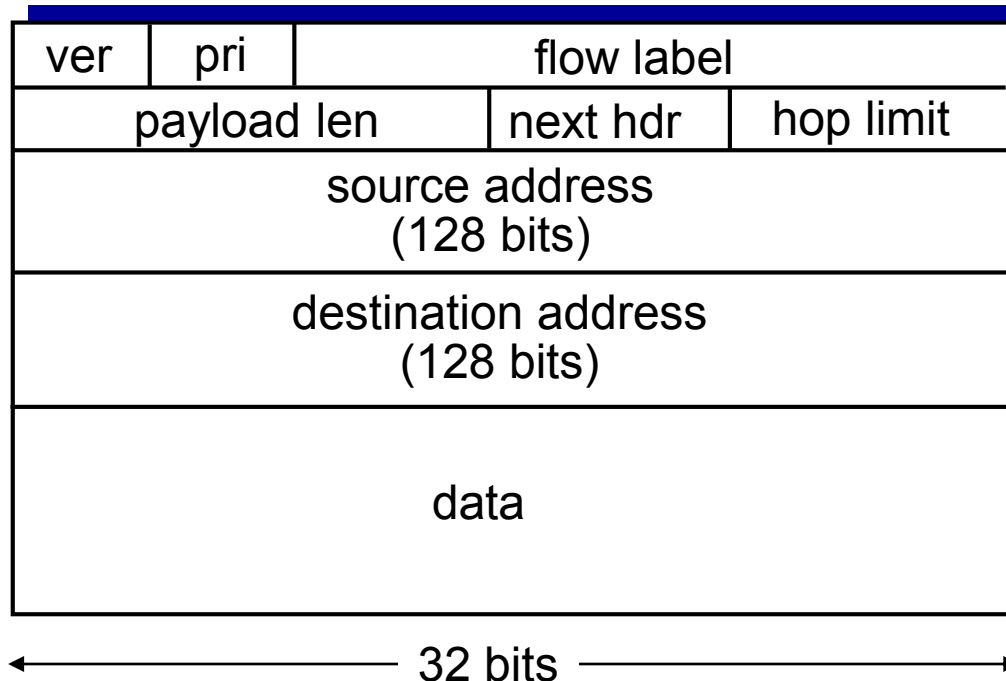
# IPv6 datagram format

*Priority/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



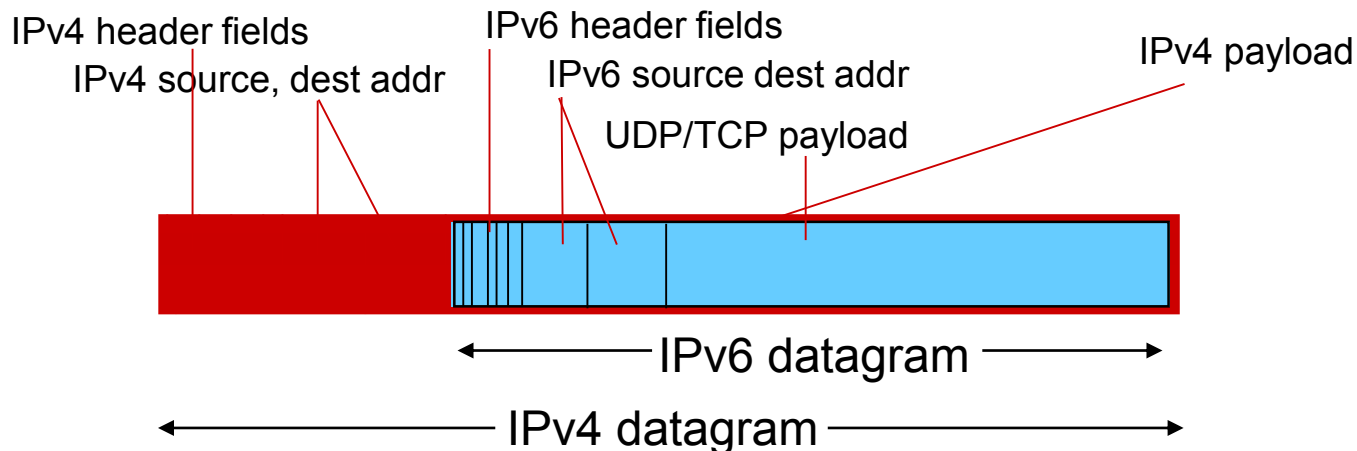


# Other changes from IPv4

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

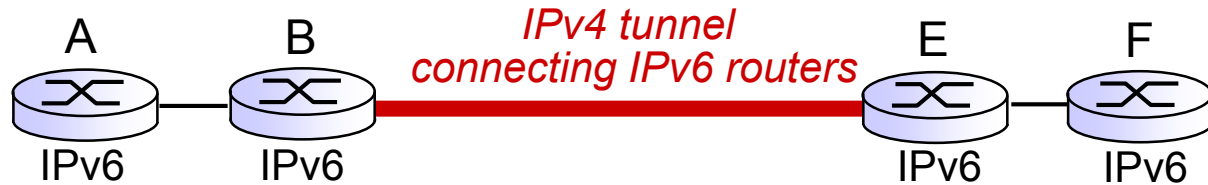
# Transition from IPv4 to IPv6

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

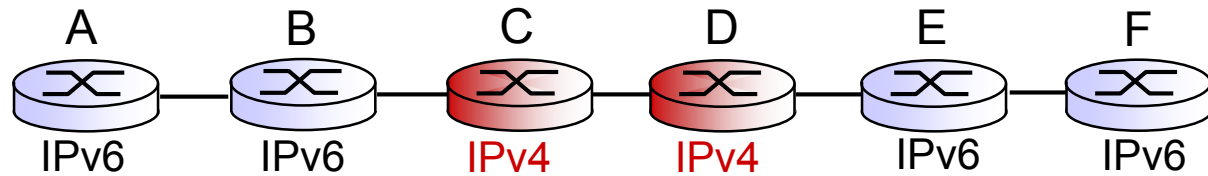


# Tunneling

logical view:

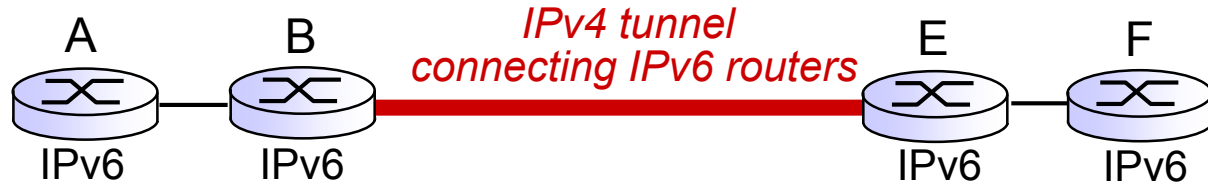


physical view:

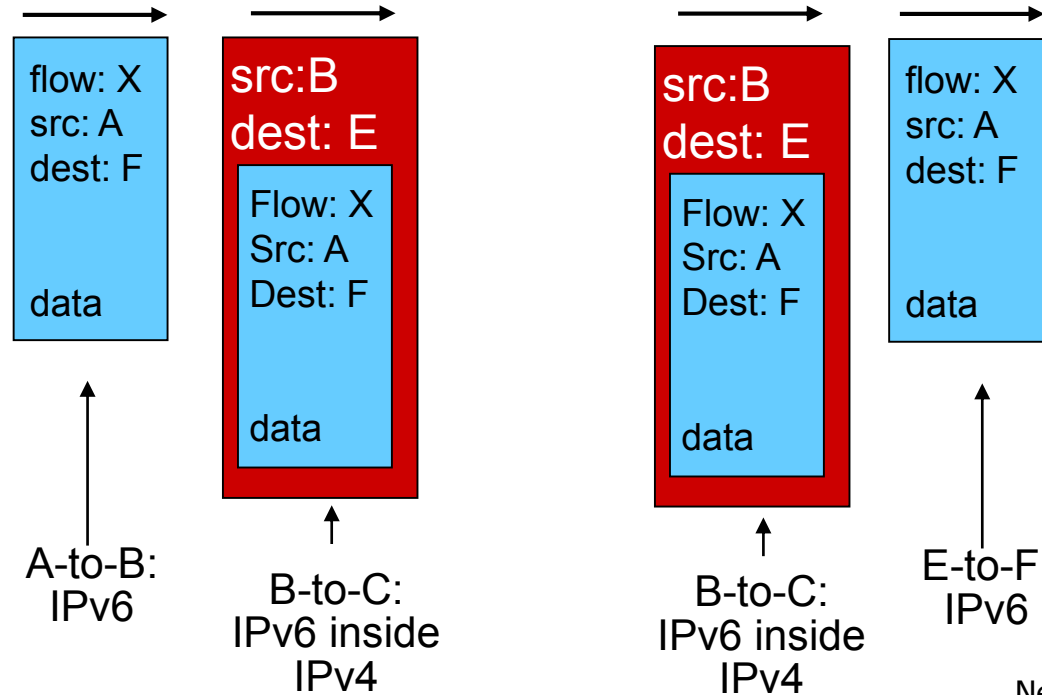
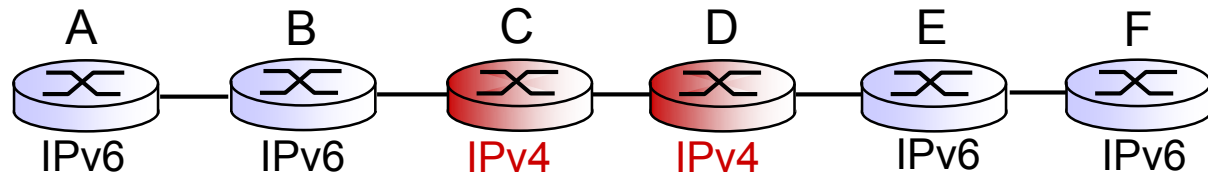


# Tunneling

logical view:



physical view:



# IPv6: adoption

- ❖ US National Institutes of Standards estimate [2013]:
  - ~3% of industry IP routers
  - ~11% of US gov't routers
- ❖ *Long (long!) time for deployment, use*
  - 20 years and counting!
  - think of application-level changes in last 20 years: WWW, Facebook, ...
  - *Why?*

# Chapter 4: outline

## 4.1 introduction

## 4.2 virtual circuit and datagram networks

## 4.3 what's inside a router

## 4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

## 4.5 routing algorithms

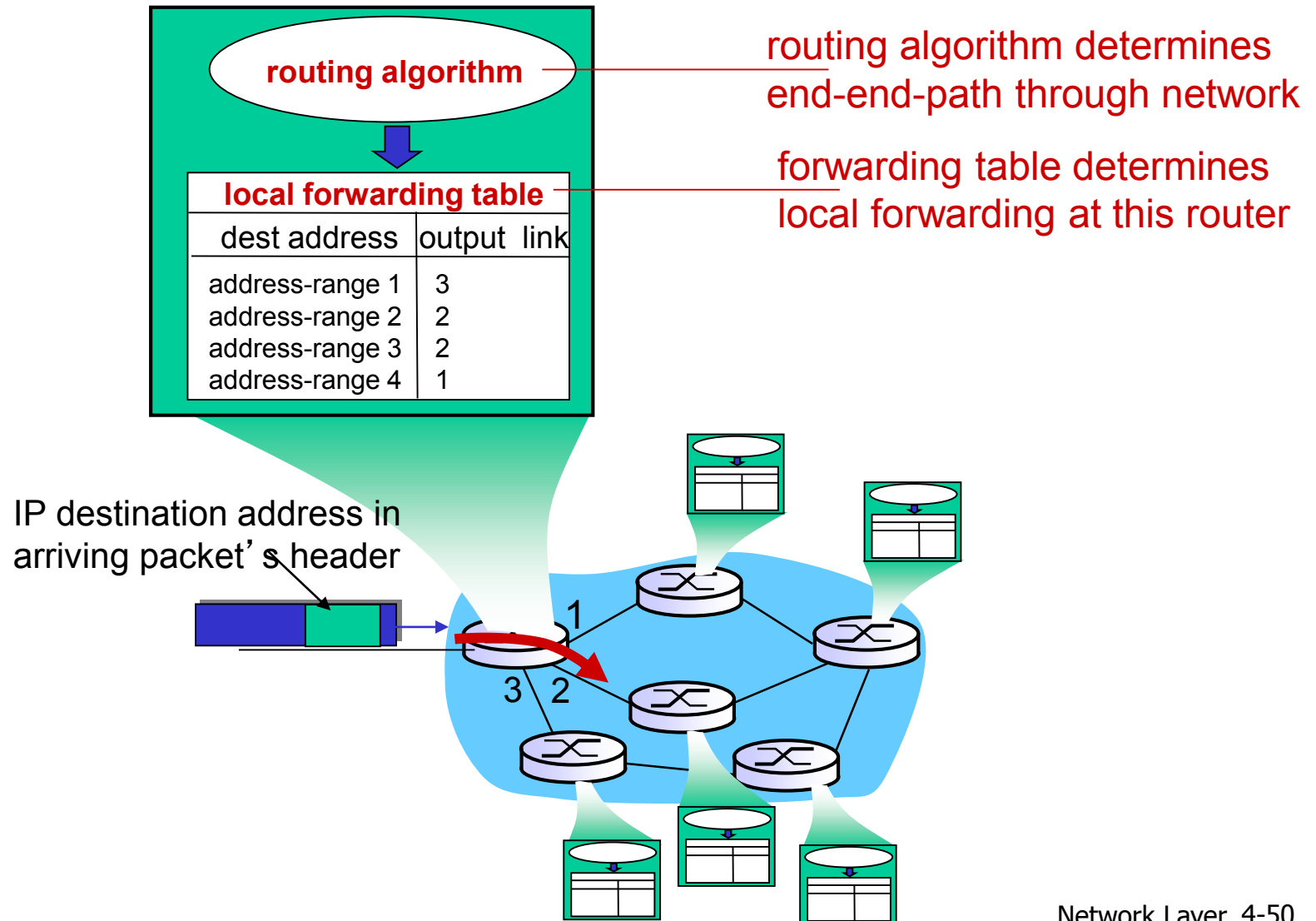
- link state
- distance vector
- hierarchical routing

## 4.6 routing in the Internet

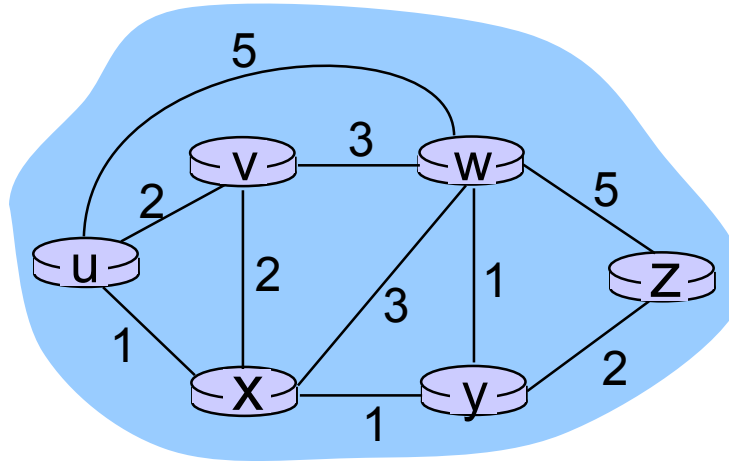
- RIP
- OSPF
- BGP

## 4.7 broadcast and multicast routing

# Interplay between routing, forwarding



# Graph abstraction



graph:  $G = (N, E)$

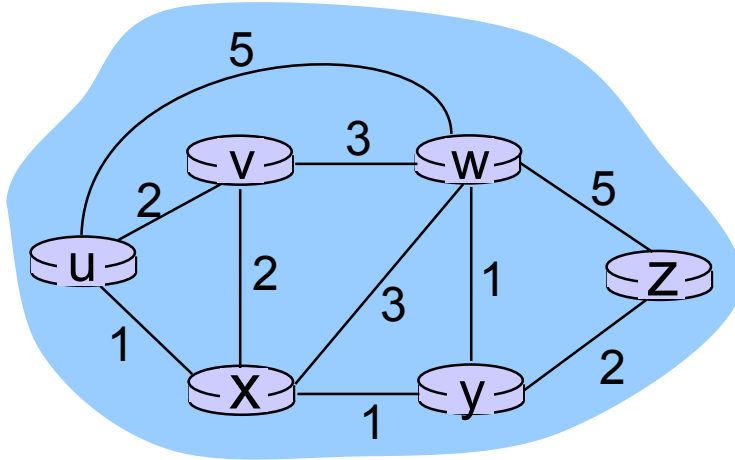
$N$  = set of routers =  $\{ u, v, w, x, y, z \}$

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

*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



$c(x, x') = \text{cost of link } (x, x')$   
e.g.,  $c(w, z) = 5$

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

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

**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

# Chapter 4: outline

## 4.1 introduction

## 4.2 virtual circuit and datagram networks

## 4.3 what's inside a router

## 4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

## 4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

## 4.6 routing in the Internet

- RIP
- OSPF
- BGP

## 4.7 broadcast and multicast routing

# A Link-State Routing Algorithm

## *Dijkstra's algorithm*

- ❖ net topology, 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;  $= \infty$  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

1 **Initialization:**

2  $N' = \{u\}$

3 for all nodes  $v$

4 if  $v$  adjacent to  $u$

5 then  $D(v) = c(u,v)$

6 else  $D(v) = \infty$

7

8 **Loop**

9 find  $w$  not in  $N'$  such that  $D(w)$  is a minimum

10 add  $w$  to  $N'$

11 update  $D(v)$  for all  $v$  adjacent to  $w$  and not in  $N'$  :

12  **$D(v) = \min( D(v), D(w) + c(w,v) )$**

13 /\* new cost to  $v$  is either old cost to  $v$  or known

14 shortest path cost to  $w$  plus cost from  $w$  to  $v$  \*/

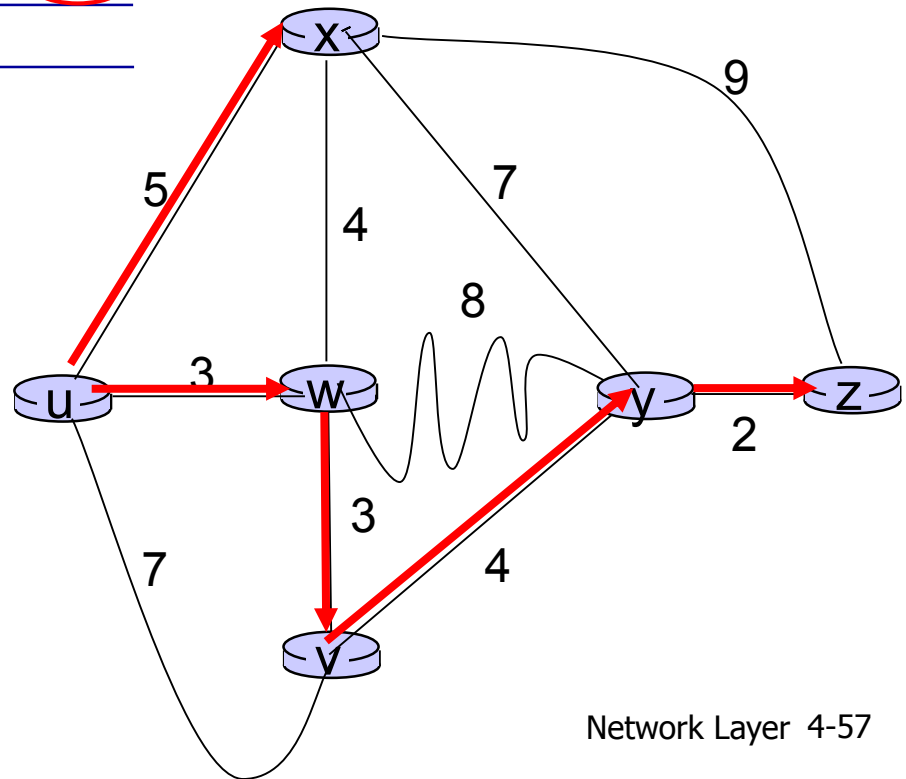
15 **until all nodes in  $N'$**

# Dijkstra's algorithm: example

Step	N'	D( <b>v</b> ) p(v)	D( <b>w</b> ) p(w)	D( <b>x</b> ) p(x)	D( <b>y</b> ) p(y)	D( <b>z</b> ) p(z)
0	u	7,u	<b>3,u</b>	5,u	$\infty$	$\infty$
1	uw	6,w		<b>5,u</b>	11,w	$\infty$
2	uwx	<b>6,w</b>			11,w	14,x
3	uwxv				<b>10,v</b>	14,x
4	uwxvy					<b>12,y</b>
5	uwxvyz					

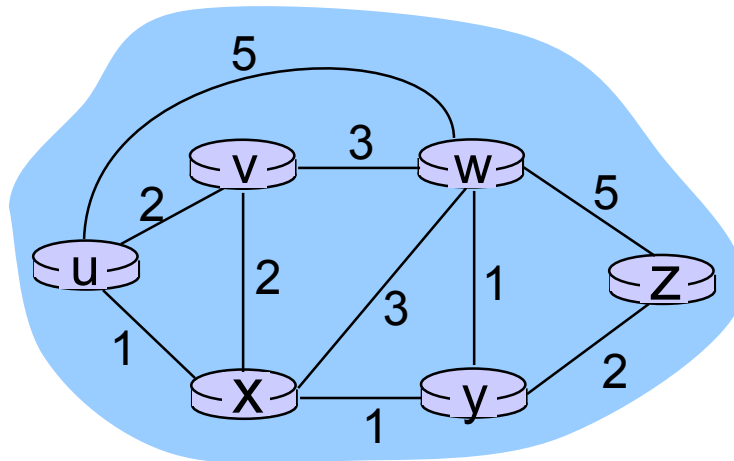
## notes:

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



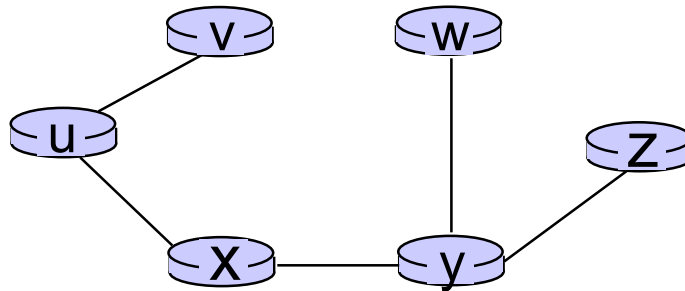
# Dijkstra's algorithm: another example

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	$\infty$	$\infty$
1	ux	2,u	4,x		2,x	$\infty$
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)

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)



# Chapter 4: outline

## 4.1 introduction

## 4.2 virtual circuit and datagram networks

## 4.3 what's inside a router

## 4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

## 4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

## 4.6 routing in the Internet

- RIP
- OSPF
- BGP

## 4.7 broadcast and multicast routing

# Distance vector algorithm

*Bellman-Ford equation (dynamic programming)*

let

$d_x(y) :=$  cost of least-cost path from  $x$  to  $y$

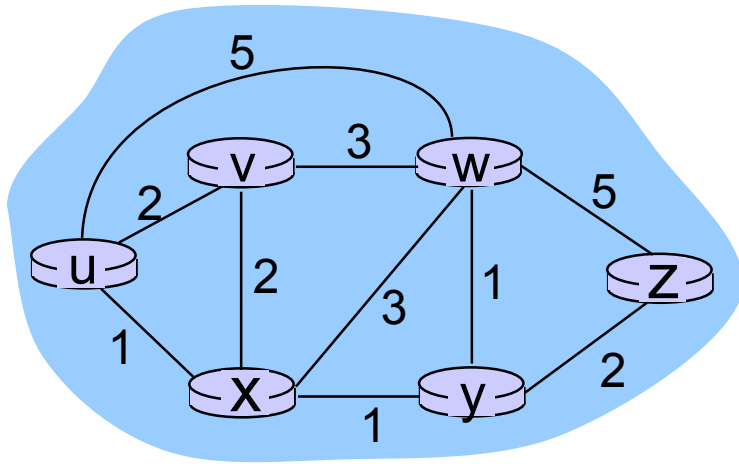
then

$$d_x(y) = \min_v \{ c(x,v) + d_v(y) \}$$

cost from neighbor  $v$  to destination  $y$   
cost to neighbor  $v$

$\min$  taken over all neighbors  $v$  of  $x$

# Bellman-Ford example



clearly,  $d_v(z) = 5$ ,  $d_x(z) = 3$ ,  $d_w(z) = 3$

B-F equation says:

$$\begin{aligned} d_u(z) &= \min \{ c(u,v) + d_v(z), \\ &\quad c(u,x) + d_x(z), \\ &\quad c(u,w) + d_w(z) \} \\ &= \min \{ 2 + 5, \\ &\quad 1 + 3, \\ &\quad 5 + 3 \} = 4 \end{aligned}$$

node achieving minimum is next  
hop in shortest path, used in forwarding table

# Distance vector algorithm

- ❖  $D_x(y)$  = estimate of least cost from  $x$  to  $y$ 
  - $x$  maintains distance vector  $\mathbf{D}_x = [D_x(y): y \in N]$
- ❖ node  $x$ :
  - knows cost to each neighbor  $v$ :  $c(x,v)$
  - maintains its neighbors' distance vectors. For each neighbor  $v$ ,  $x$  maintains  $\mathbf{D}_v = [D_v(y): y \in N]$

# Distance vector algorithm

## *key idea:*

- ❖ from time-to-time, each node sends its own distance vector estimate to neighbors
- ❖ when  $x$  receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \text{ for each node } y \in N$$

- ❖ under natural conditions, the estimate  $D_x(y)$  converge to the actual least cost  $d_x(y)$

$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$$

$$= \min\{2+0, 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$$

$$= \min\{2+1, 7+0\} = 3$$

**node x  
table**

		cost to		
		x	y	z
from	x	0	2	7
	y	∞	∞	∞
	z	∞	∞	∞

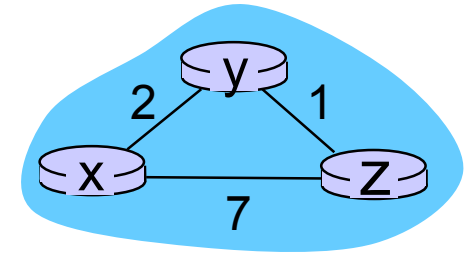
		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	7	1	0

**node y  
table**

		cost to		
		x	y	z
from	x	∞	∞	∞
	y	2	0	1
	z	∞	∞	∞

**node z  
table**

		cost to		
		x	y	z
from	x	∞	∞	∞
	y	∞	∞	∞
	z	7	1	0



time

$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$$

$$= \min\{2+0, 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$$

$$= \min\{2+1, 7+0\} = 3$$

**node x  
table**

		cost to		
		x	y	z
from	x	0	2	7
	y	∞	∞	∞
	z	∞	∞	∞

**node y  
table**

		cost to		
		x	y	z
from	x	∞	∞	∞
	y	2	0	1
	z	∞	∞	∞

**node z  
table**

		cost to		
		x	y	z
from	x	∞	∞	∞
	y	∞	∞	∞
	z	7	1	0

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	7	1	0

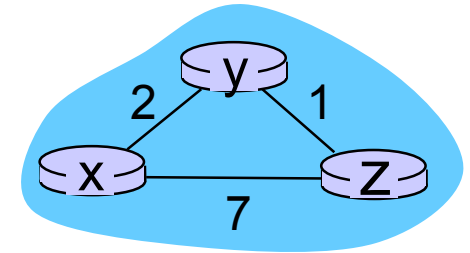
		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	3	1	0

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	3	1	0

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	3	1	0

		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	3	1	0

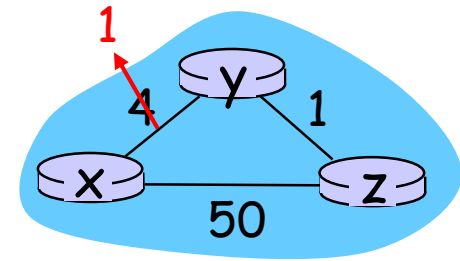


time

# Distance vector: link cost changes

## *link cost changes:*

- ❖ node detects local link cost change
- ❖ updates routing info, recalculates distance vector
- ❖ if DV changes, notify neighbors



“good  
news  
travels  
fast”

$t_0$ : y detects link-cost change, updates its DV, informs its neighbors.

$t_1$ : z receives update from y, updates its table, computes new least cost to x, sends its neighbors its DV.

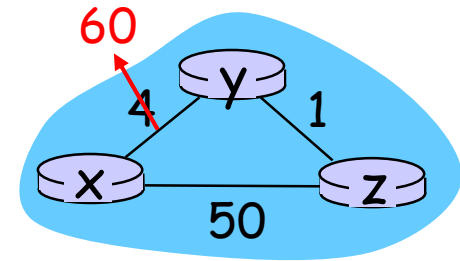
$t_2$ : y receives z's update, updates its distance table. y's least costs do *not* change, so y does *not* send a message to z.



# Distance vector: link cost changes

## *link cost changes:*

- ❖ node detects local link cost change
- ❖ *bad news travels slow* - “count to infinity” problem!
- ❖ 44 iterations before algorithm stabilizes: see text



## *poisoned reverse:*

- ❖ If Z routes through Y to get to X :
  - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- ❖ will this completely solve count to infinity problem?

# Comparison of LS and DV algorithms

## *message complexity*

- ❖ **LS:** with  $n$  nodes,  $E$  links,  $O(nE)$  msgs sent
- ❖ **DV:** exchange between neighbors only
  - convergence time varies

## *speed of convergence*

- ❖ **LS:**  $O(n^2)$  algorithm requires  $O(nE)$  msgs
  - may have oscillations
- ❖ **DV:** convergence time varies
  - may be routing loops
  - count-to-infinity problem

**robustness:** what happens if router malfunctions?

## *LS:*

- node can advertise incorrect *link* cost
- each node computes only its own table

## *DV:*

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

# Chapter 4: outline

## 4.1 introduction

## 4.2 virtual circuit and datagram networks

## 4.3 what's inside a router

## 4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

## 4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

## 4.6 routing in the Internet

- RIP
- OSPF
- BGP

## 4.7 broadcast and multicast routing

# Hierarchical routing

our routing study thus far - idealization

- ❖ all routers identical
- ❖ network “flat”

... *not* true in practice

*scale:* with 600 million destinations:

- ❖ can't store all dest's in routing tables!
- ❖ routing table exchange would swamp links!

*administrative autonomy*

- ❖ internet = network of networks
- ❖ each network admin may want to control routing in its own network

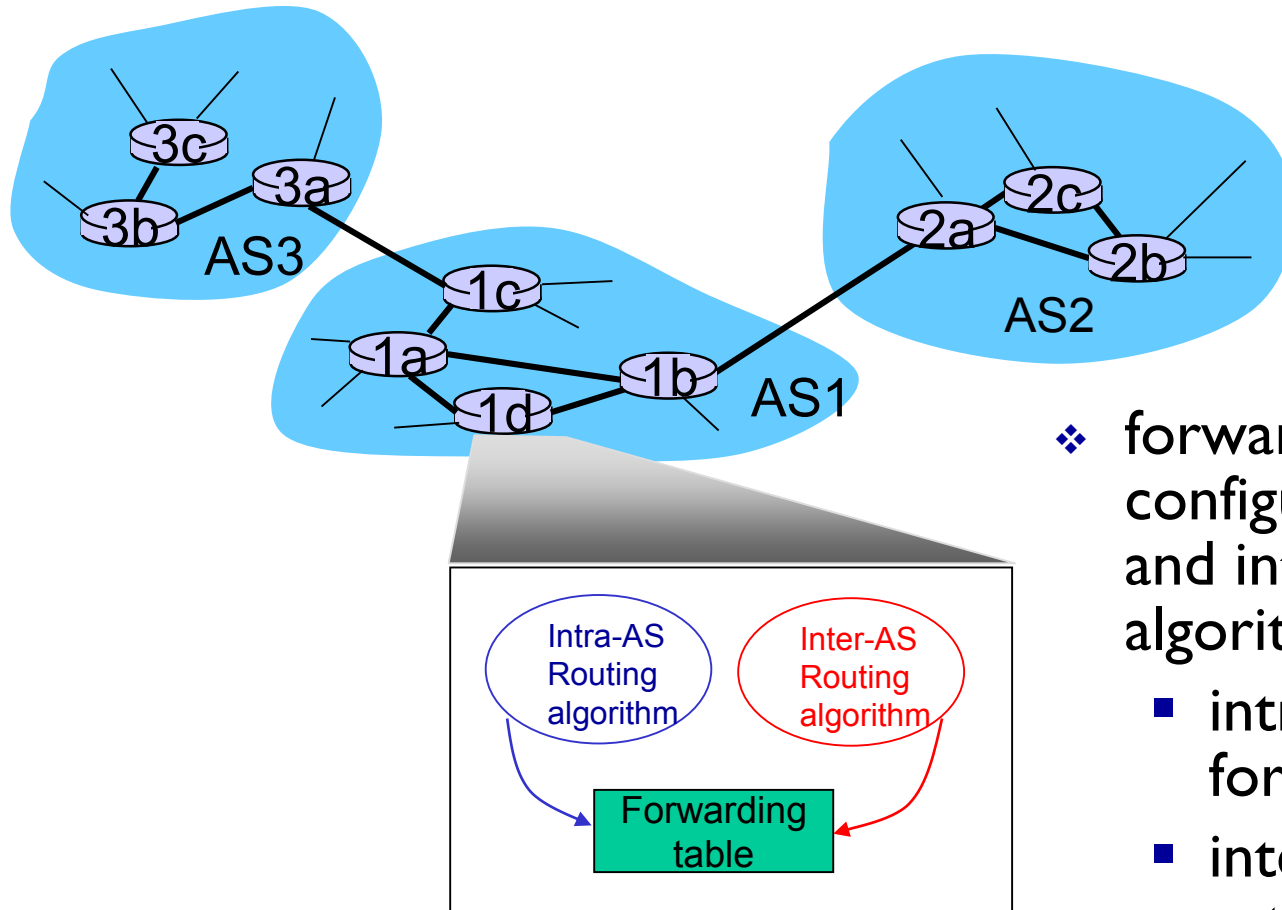
# Hierarchical routing

- ❖ aggregate routers into regions, “**autonomous systems**” (AS)
- ❖ routers in same AS run same routing protocol
  - “**intra-AS**” routing protocol
  - routers in different AS can run different intra-AS routing protocol

## *gateway router:*

- ❖ at “edge” of its own AS
- ❖ has link to router in another AS

# Interconnected ASes



- ❖ forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS sets entries for internal destinations
  - inter-AS & intra-AS sets entries for external destinations

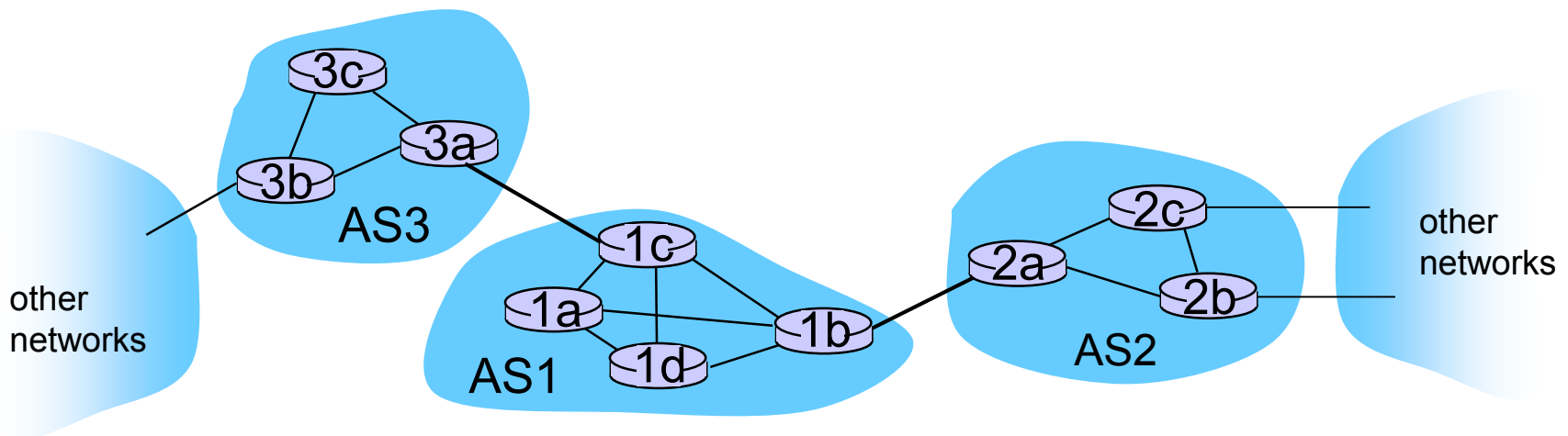
# Inter-AS tasks

- ❖ suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

*AS1 must:*

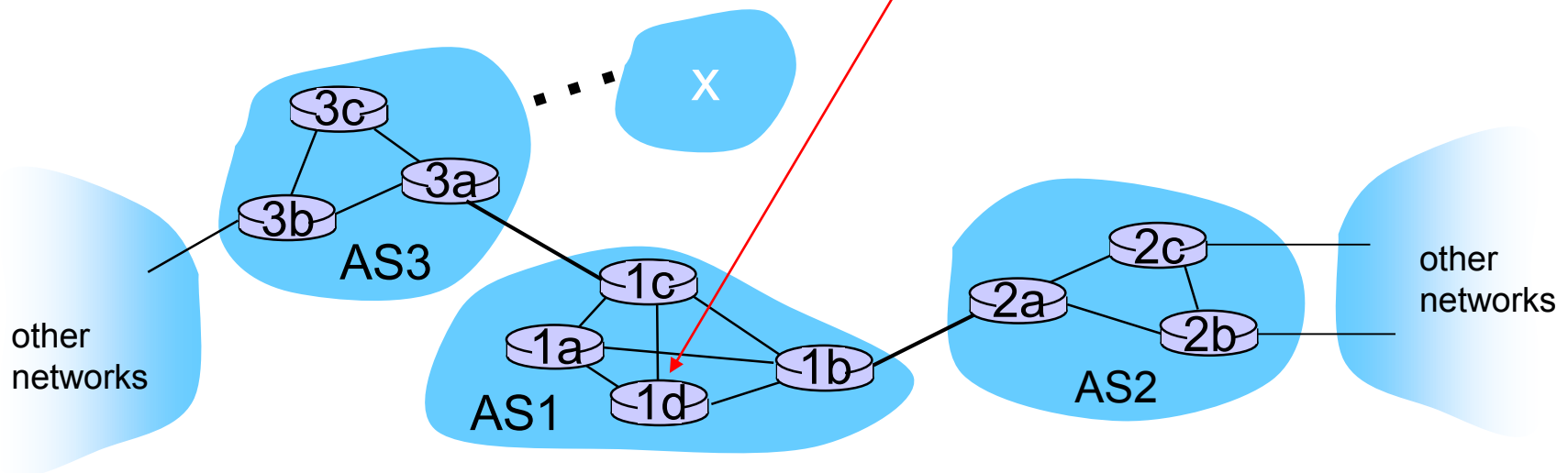
1. learn which destds are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

*job of inter-AS routing!*



# Example: setting forwarding table in router 1d

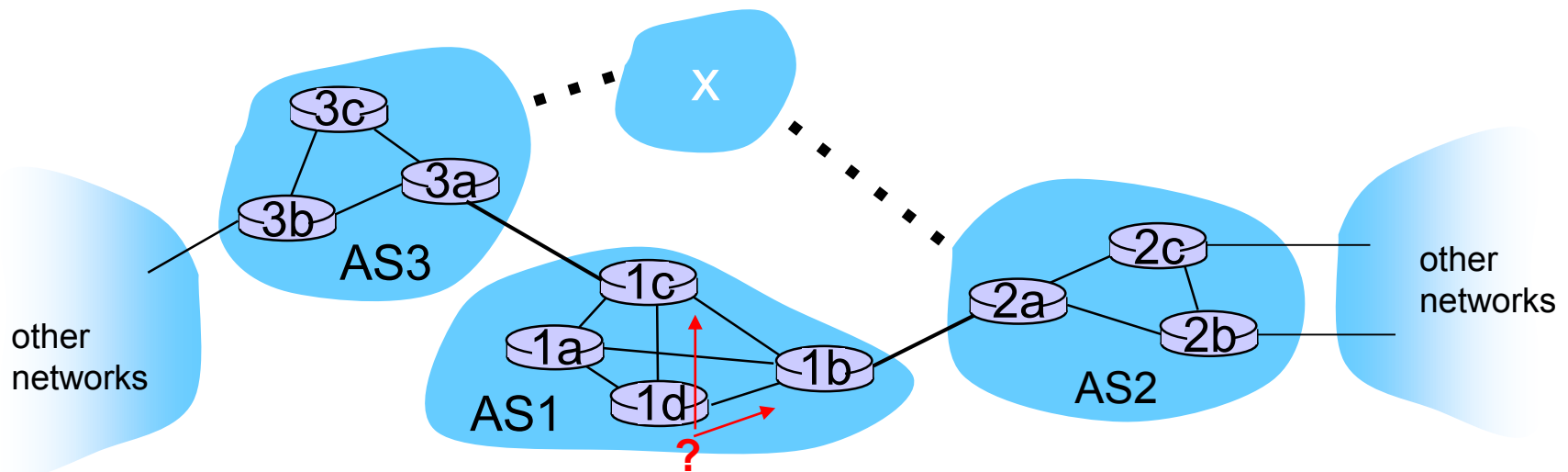
- ❖ suppose AS1 learns (via inter-AS protocol) that subnet **x** reachable via AS3 (gateway 1c), but not via AS2
  - inter-AS protocol propagates reachability info to all internal routers
- ❖ router 1d determines from intra-AS routing info that its interface **l** is on the least cost path to 1c
  - installs forwarding table entry **(x,l)**





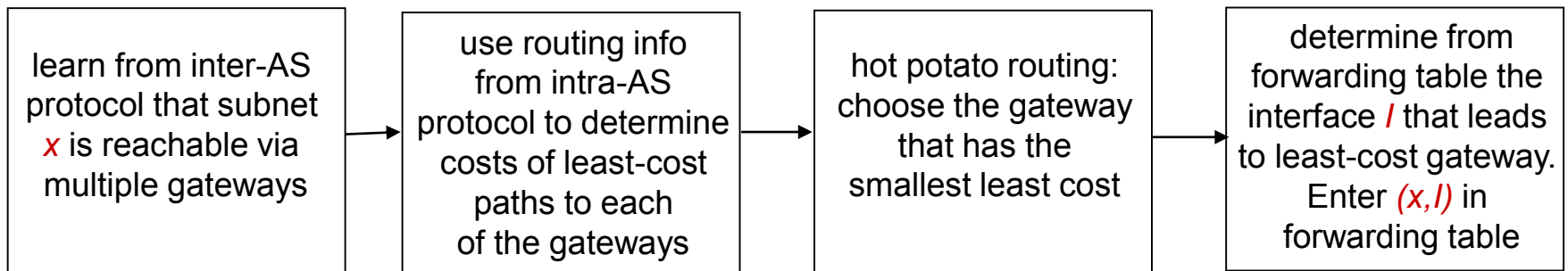
# Example: choosing among multiple ASes

- ❖ now suppose AS1 learns from inter-AS protocol that subnet **x** is reachable from AS3 *and* from AS2.
- ❖ to configure forwarding table, router 1d must determine which gateway it should forward packets towards for dest **x**
  - this is also job of inter-AS routing protocol!



# Example: choosing among multiple ASes

- ❖ now suppose AS1 learns from inter-AS protocol that subnet **x** is reachable from AS3 *and* from AS2.
- ❖ to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest **x**
  - this is also job of inter-AS routing protocol!
- ❖ **hot potato routing: send** packet towards closest of two routers.



# Chapter 4: outline

## 4.1 introduction

## 4.2 virtual circuit and datagram networks

## 4.3 what's inside a router

## 4.4 IP: Internet Protocol

- datagram format
- IPv4 addressing
- ICMP
- IPv6

## 4.5 routing algorithms

- link state
- distance vector
- hierarchical routing

## 4.6 routing in the Internet

- RIP
- OSPF
- BGP

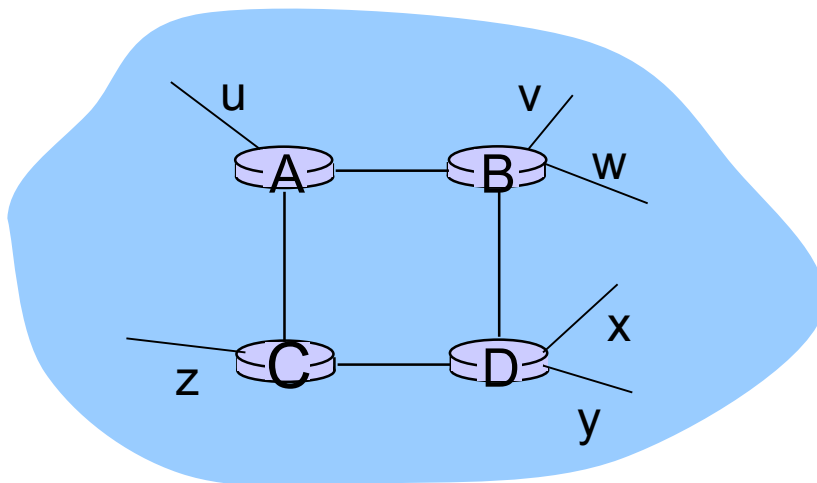
## 4.7 broadcast and multicast routing

# Intra-AS Routing

- ❖ also known as *interior gateway protocols (IGP)*
- ❖ most common intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

# RIP ( Routing Information Protocol)

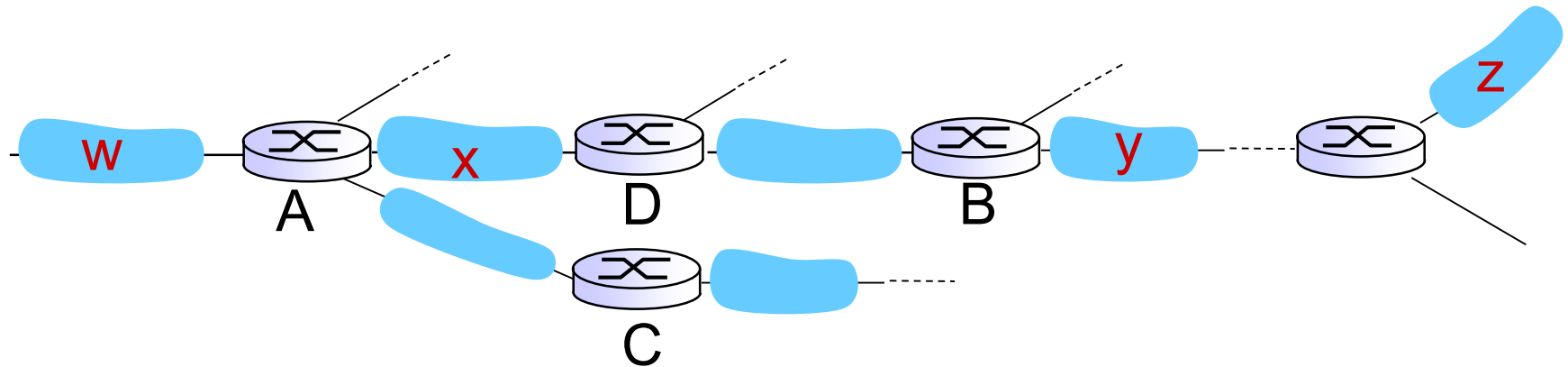
- ❖ included in BSD-UNIX distribution in 1982
- ❖ distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors every 30 sec in response message (aka **advertisement**)
  - each advertisement: list of up to 25 destination **subnets** (in IP addressing sense)



from router A to destination **subnets**:

<u>subnet</u>	<u>hops</u>
u	1
v	2
w	2
x	3
y	3
z	2

# RIP: example



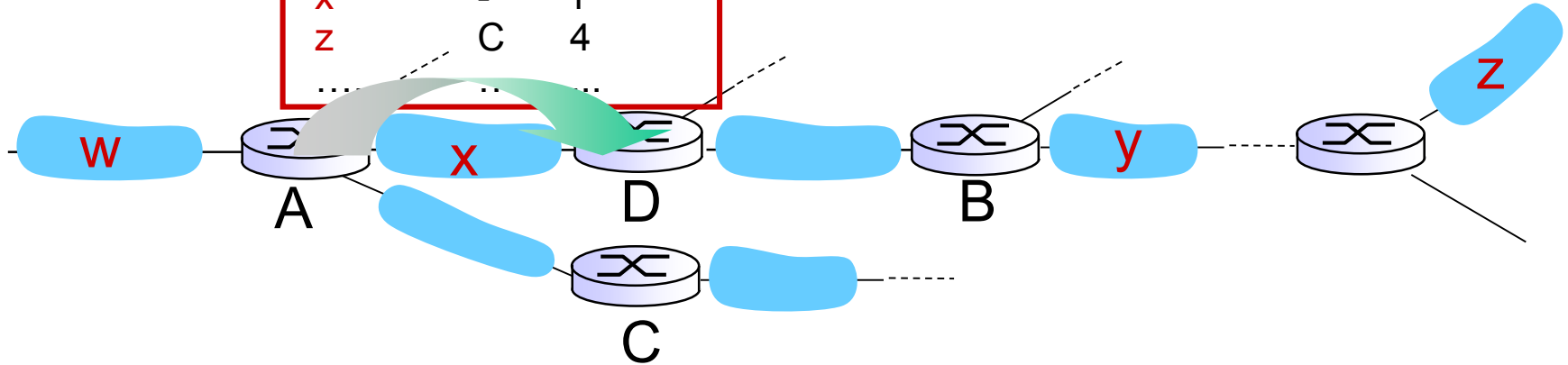
routing table in router D

destination subnet	next router	# hops to dest
w	A	2
y	B	2
x	--	1
....	....	....

# RIP: example

A-to-D advertisement

dest	next	hops
W	-	1
X	-	1
Z	C	4
...	...	...



routing table in router D

destination subnet	next router	# hops to dest
W	A	2
y	B	2
Z	<del>B</del> → A	<del>7</del> → 5
X	--	1
....	....	....

# RIP: link failure, recovery

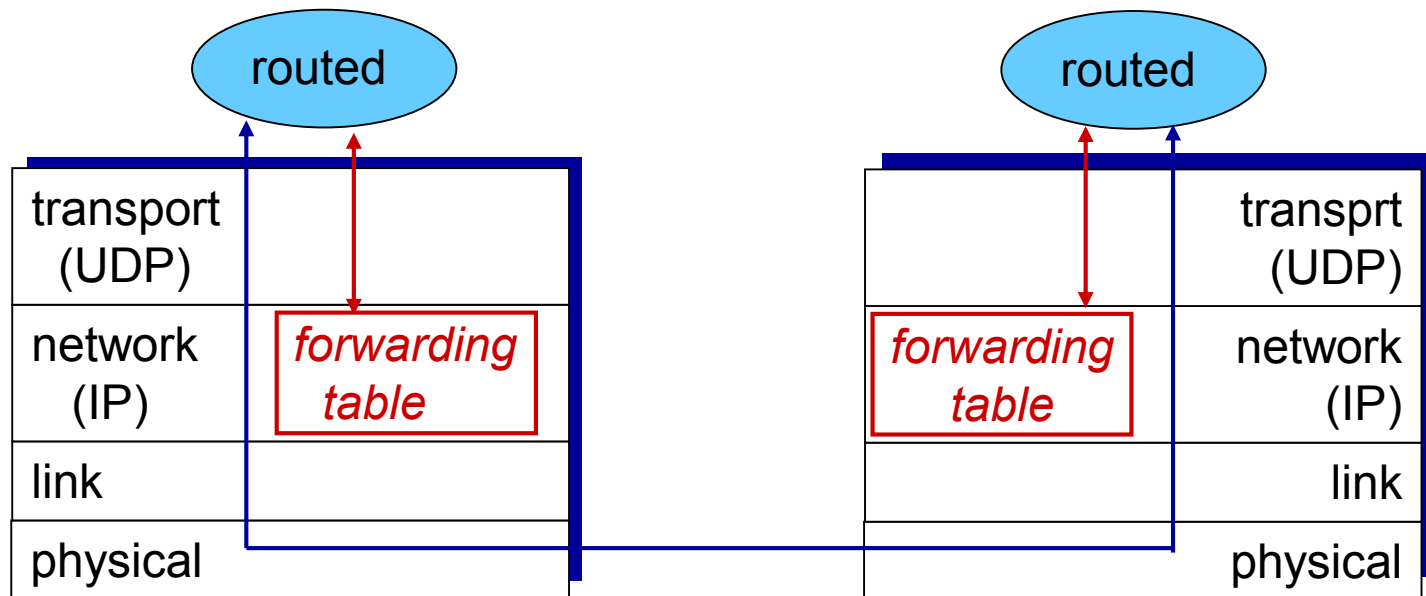
if no advertisement heard after 180 sec -->  
neighbor/link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly (?) propagates to entire net
- *poison reverse* used to prevent ping-pong loops (infinite distance = 16 hops)



# RIP table processing

- ❖ RIP routing tables managed by *application-level* process called route-d (daemon)
- ❖ advertisements sent in UDP packets, periodically repeated



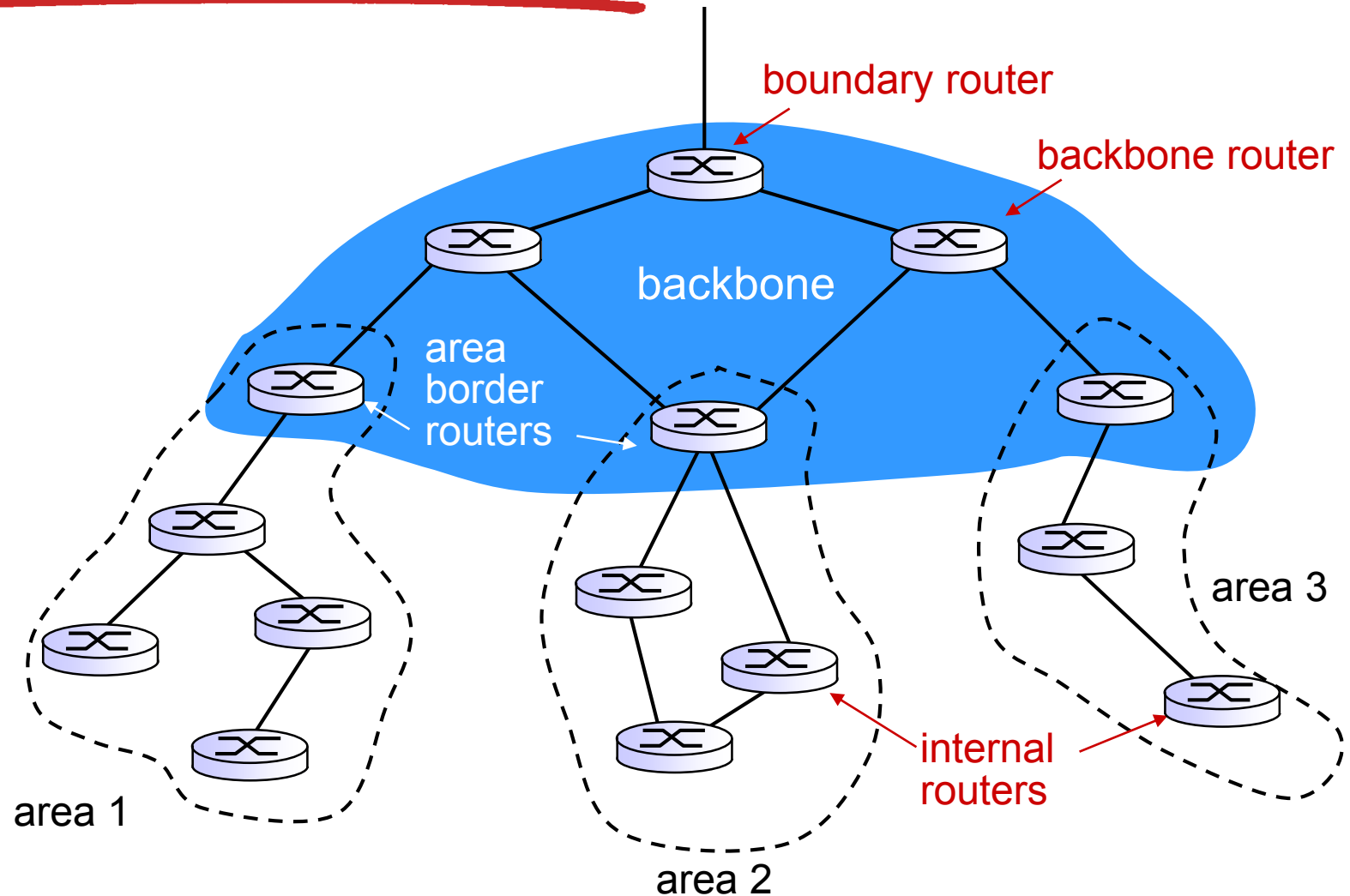
# OSPF (Open Shortest Path First)

- ❖ “open”: publicly available
- ❖ uses link state algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm
- ❖ OSPF advertisement carries one entry per neighbor
- ❖ advertisements flooded to *entire* AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP)
- ❖ *IS-IS routing* protocol: nearly identical to OSPF

# OSPF “advanced” features (not in RIP)

- ❖ **security**: all OSPF messages authenticated (to prevent malicious intrusion)
- ❖ **multiple** same-cost **paths** allowed (only one path in RIP)
- ❖ for each link, multiple cost metrics for different **TOS** (e.g., satellite link cost set “low” for best effort ToS; high for real time ToS)
- ❖ integrated uni- and **multicast** support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- ❖ **hierarchical** OSPF in large domains.

# Hierarchical OSPF



# Hierarchical OSPF

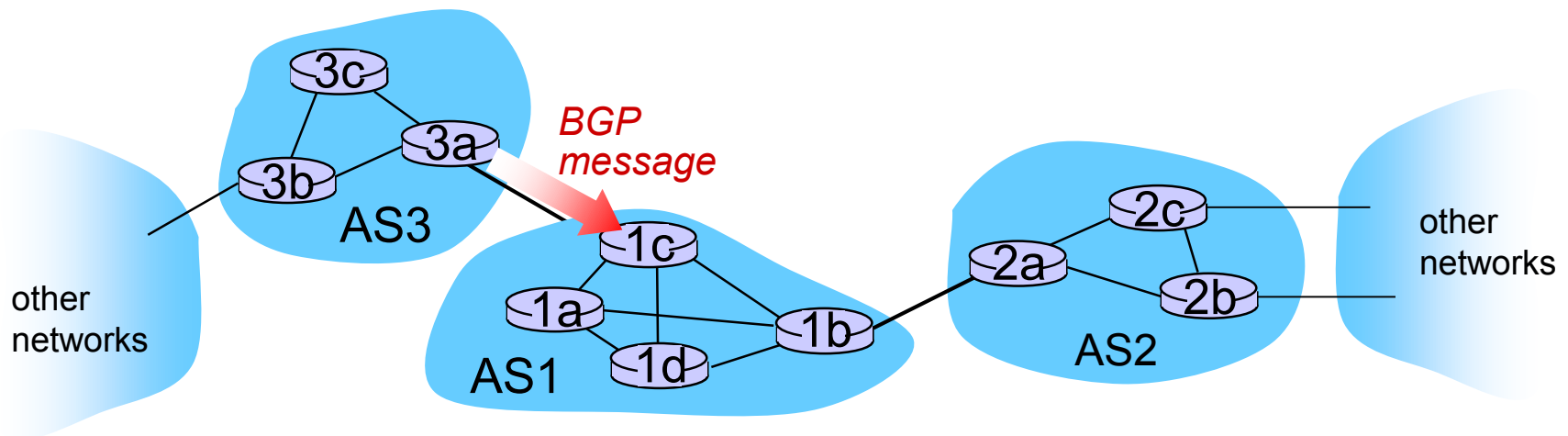
- ❖ *two-level hierarchy*: local area, backbone.
  - link-state advertisements only in area
  - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- ❖ *area border routers*: “summarize” distances to nets in own area, advertise to other Area Border routers.
- ❖ *backbone routers*: run OSPF routing limited to backbone.
- ❖ *boundary routers*: connect to other AS' s.

# Internet inter-AS routing: BGP

- ❖ **BGP (Border Gateway Protocol):** *the de facto inter-domain routing protocol*
  - “glue that holds the Internet together”
- ❖ BGP provides each AS a means to:
  - **eBGP:** obtain subnet reachability information from neighboring ASs.
  - **iBGP:** propagate reachability information to all AS-internal routers.
  - determine “good” routes to other networks based on reachability information and policy.
- ❖ allows subnet to advertise its existence to rest of Internet: “*I am here*”

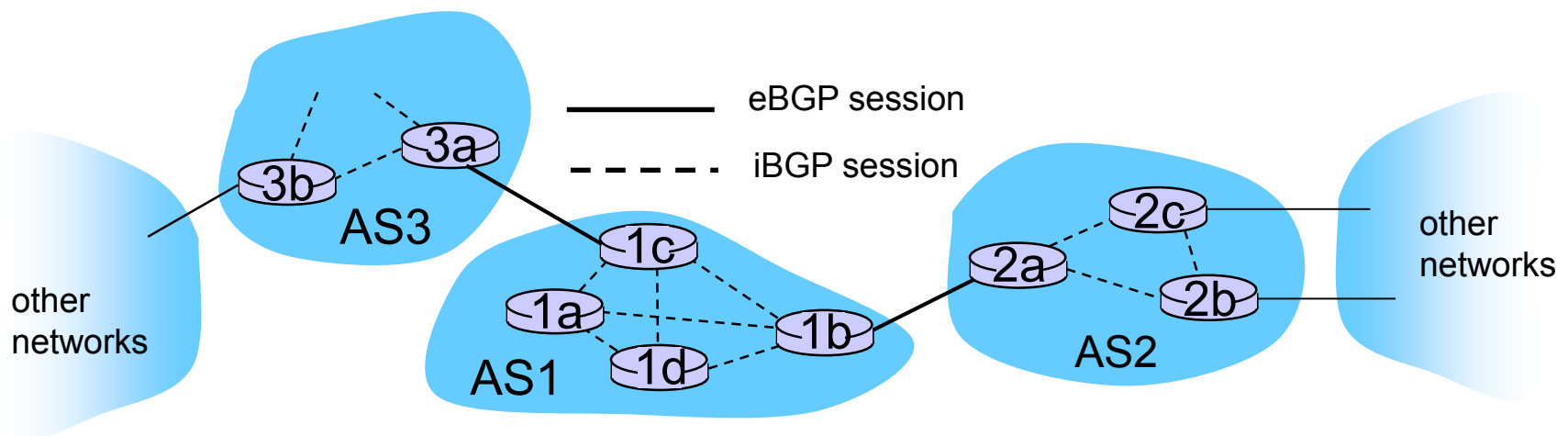
# BGP basics

- ❖ **BGP session:** two BGP routers (“peers”) exchange BGP messages:
  - advertising *paths* to different destination network prefixes (“path vector” protocol)
  - exchanged over semi-permanent TCP connections
- ❖ when AS3 advertises a prefix to AS1:
  - AS3 *promises* it will forward datagrams towards that prefix
  - AS3 can aggregate prefixes in its advertisement



# BGP basics: distributing path information

- ❖ using eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
  - 1c can then use iBGP to distribute new prefix info to all routers in AS1
  - 1b can then re-advertise new reachability info to AS2 over 1b-to-2a eBGP session
- ❖ when router learns of new prefix, it creates entry for prefix in its forwarding table.





# Path attributes and BGP routes

- ❖ advertised prefix includes BGP attributes
  - prefix + attributes = “route”
- ❖ two important attributes:
  - **AS-PATH**: contains ASs through which prefix advertisement has passed: e.g., AS 67, AS 17
  - **NEXT-HOP**: indicates specific internal-AS router to next-hop AS. (may be multiple links from current AS to next-hop-AS)
- ❖ gateway router receiving route advertisement uses **import policy** to accept/decline
  - e.g., never route through AS x
  - *policy-based* routing

# BGP route selection

- ❖ router may learn about more than 1 route to destination AS, selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria

# BGP messages

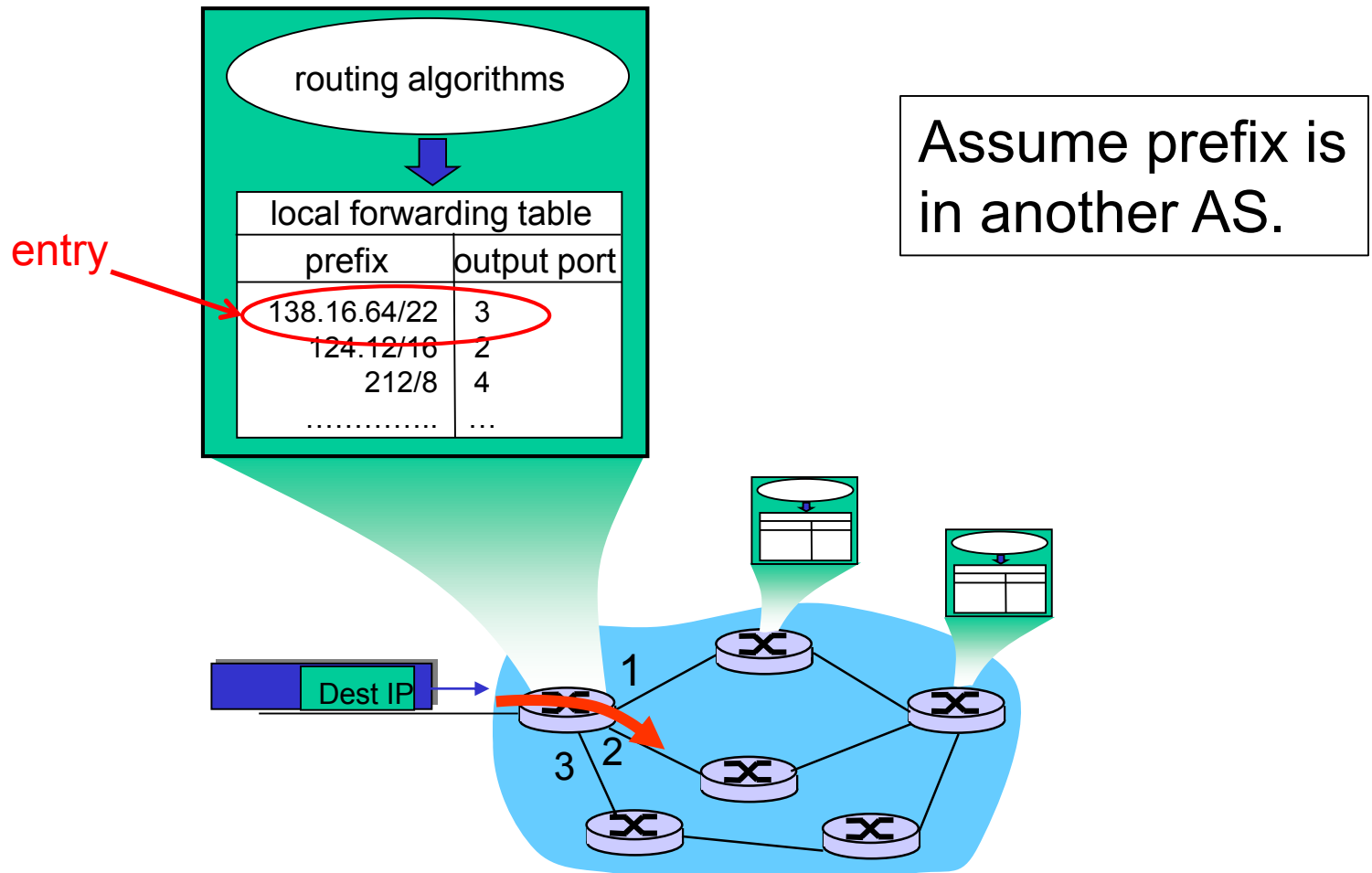
- ❖ BGP messages exchanged between peers over TCP connection
- ❖ BGP messages:
  - **OPEN:** opens TCP connection to peer and authenticates sender
  - **UPDATE:** advertises new path (or withdraws old)
  - **KEEPALIVE:** keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - **NOTIFICATION:** reports errors in previous msg; also used to close connection

# Putting it Altogether:

## *How Does an Entry Get Into a Router's Forwarding Table?*

- ❖ Answer is complicated!
- ❖ Ties together hierarchical routing (Section 4.5.3) with BGP (4.6.3) and OSPF (4.6.2).

# How does entry get in forwarding table?

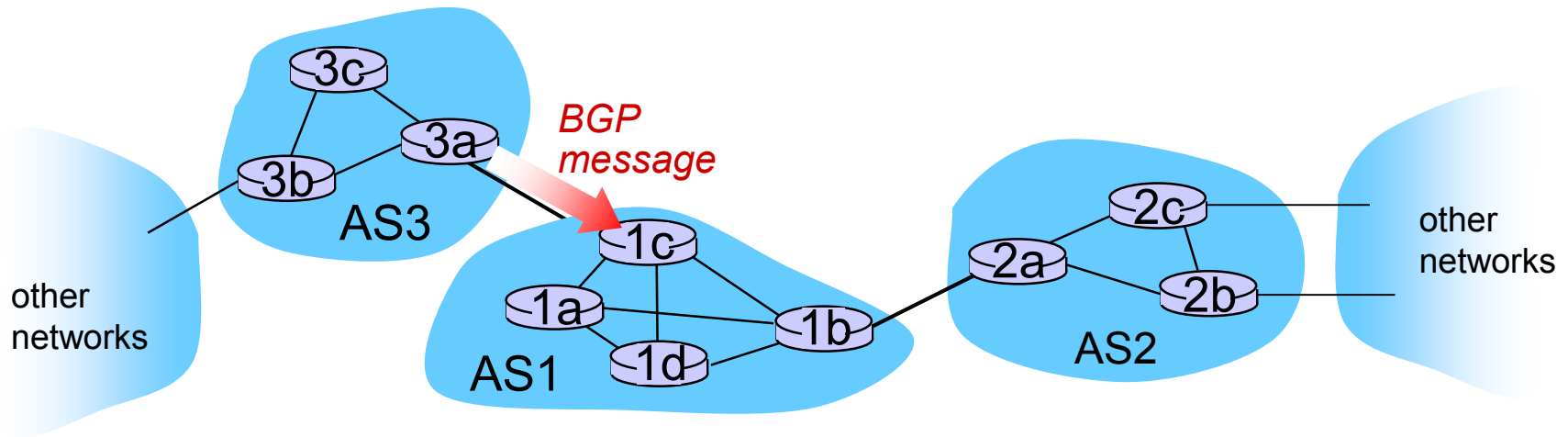


# How does entry get in forwarding table?

## High-level overview

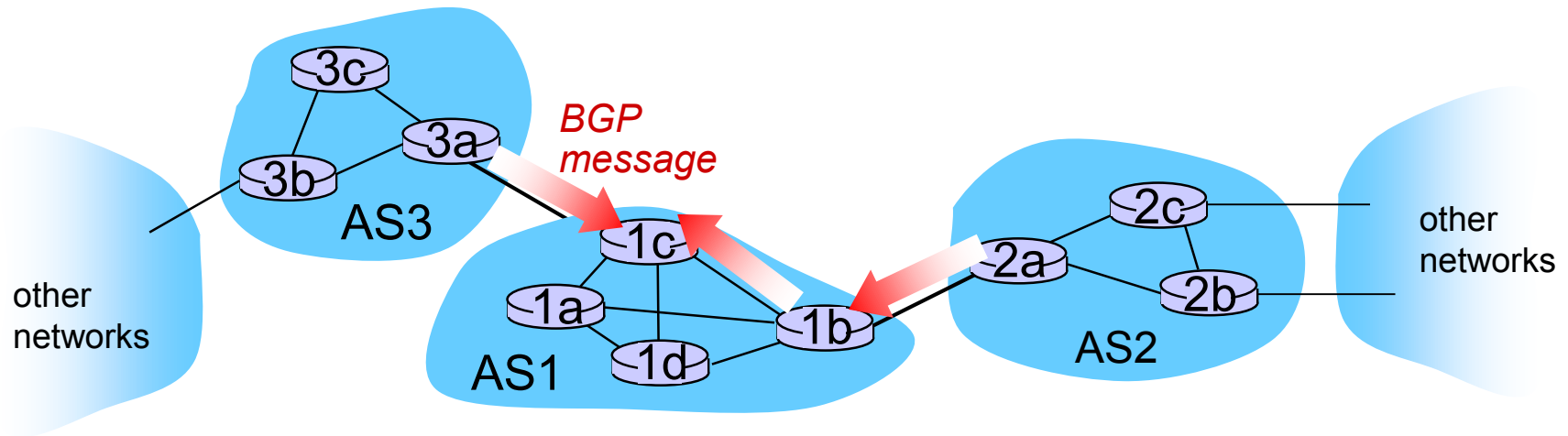
1. Router becomes aware of prefix
2. Router determines output Interface for prefix
3. Router enters prefix-interface in forwarding table

# Router becomes aware of prefix



- ❖ BGP message contains “routes”
- ❖ “route” is a prefix and attributes: AS-PATH, NEXT-HOP,...
- ❖ Example: route:
  - ❖ Prefix: 138.16.64/22 ; AS-PATH: AS3 AS131 ; NEXT-HOP: 201.44.13.125

# Router may receive multiple routes



- ❖ Router may receive multiple routes for same prefix
- ❖ Has to select one route



# Select best BGP route to prefix

- ❖ Router selects route based on shortest AS-PATH

- ❖ Example:

- ❖ AS2 AS17 to 138.16.64/22

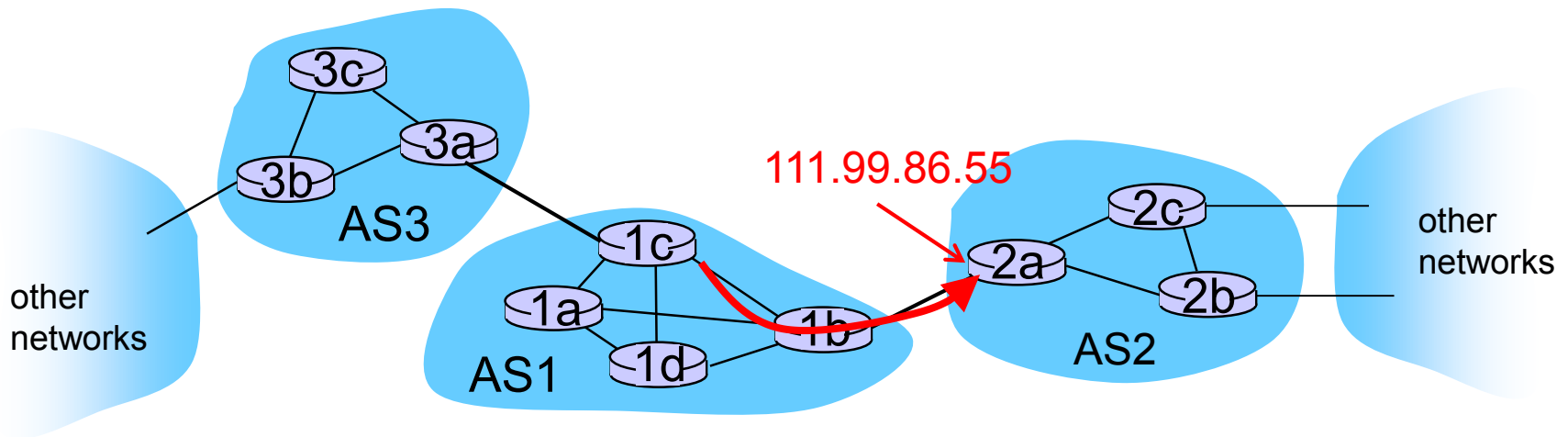
select

- ❖ AS3 AS131 AS201 to 138.16.64/22

- ❖ What if there is a tie?

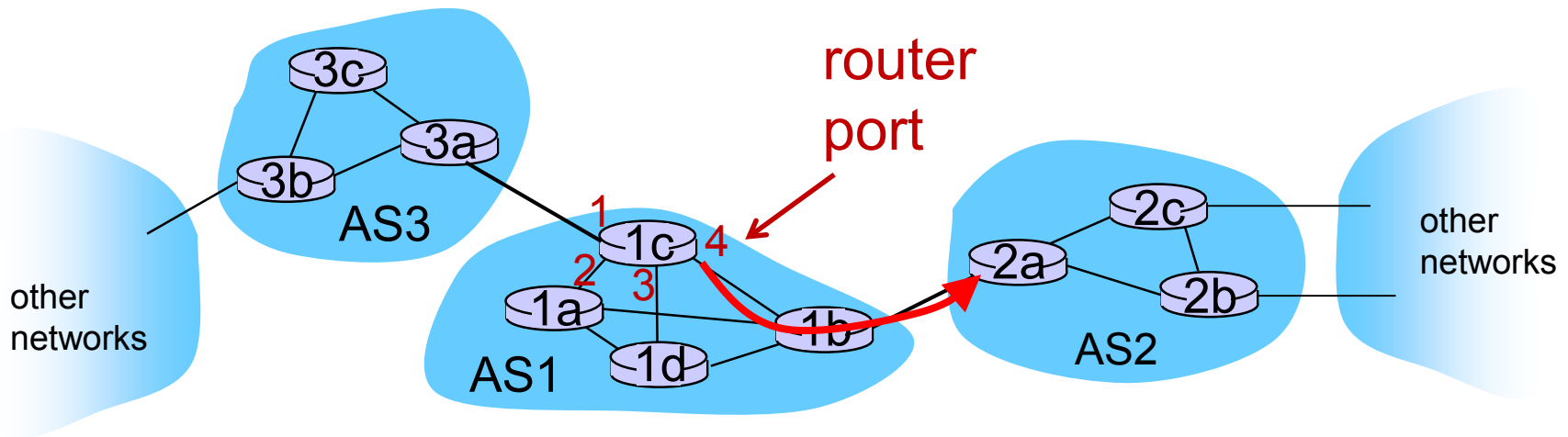
# Find best intra-route to BGP route

- ❖ Use selected route's NEXT-HOP attribute
  - Route's NEXT-HOP attribute is the IP address of the router interface that begins the AS PATH.
- ❖ Example:
  - ❖ AS-PATH: AS2 AS17 ; NEXT-HOP: 111.99.86.55
- ❖ Router uses OSPF to find shortest path from 1c to 111.99.86.55



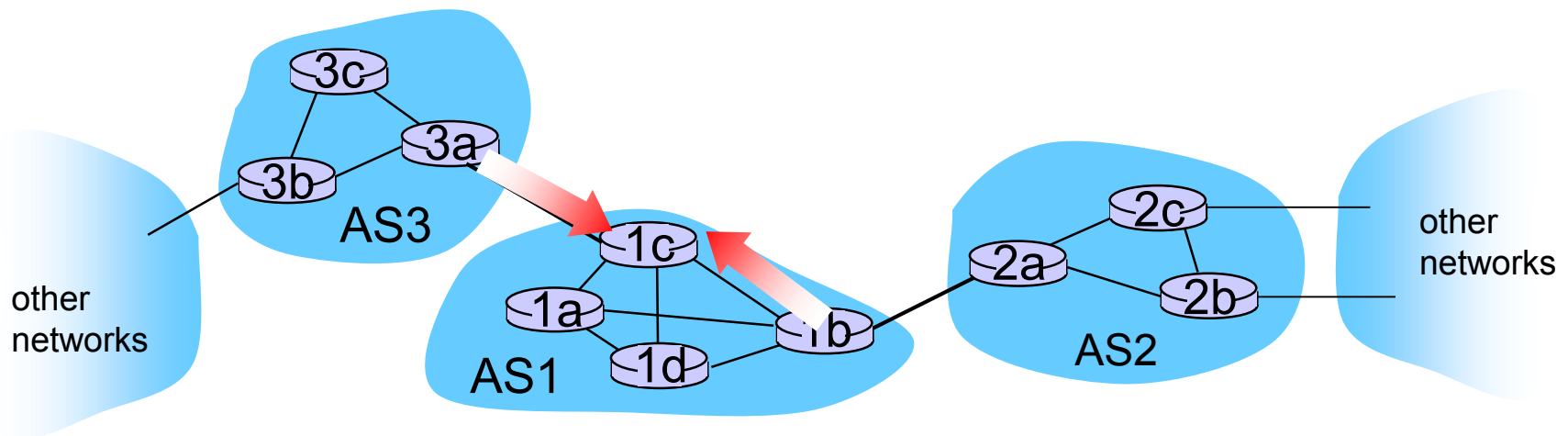
# Router identifies port for route

- ❖ Identifies port along the OSPF shortest path
- ❖ Adds prefix-port entry to its forwarding table:
  - (138.16.64/22 , port 4)



# Hot Potato Routing

- ❖ Suppose there two or more best inter-routes.
- ❖ Then choose route with closest NEXT-HOP
  - Use OSPF to determine which gateway is closest
  - Q: From 1c, chose AS3 AS131 or AS2 AS17?
  - A: route AS3 AS131 since it is closer

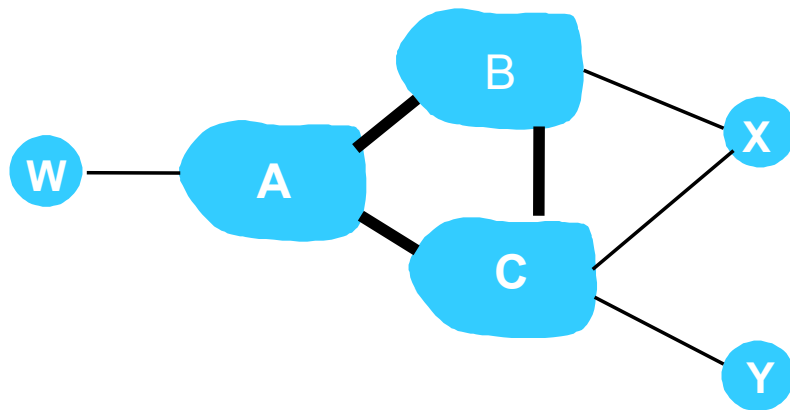




# How does entry get in forwarding table?

## Summary

1. Router becomes aware of prefix
  - via BGP route advertisements from other routers
2. Determine router output port for prefix
  - Use BGP route selection to find best inter-AS route
  - Use OSPF to find best intra-AS route leading to best inter-AS route
  - Router identifies router port for that best route
3. Enter prefix-port entry in forwarding table

# BGP routing policy



legend:  provider network  
 customer network:

- ❖ A,B,C are *provider networks*
- ❖ X,W,Y are customer (of provider networks)
- ❖ X is *dual-homed*: attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C

# Why different Intra-, Inter-AS routing ?

## *policy:*

- ❖ inter-AS: admin wants control over how its traffic routes, who routes through its net.
- ❖ intra-AS: single admin, so no policy decisions needed

## *scale:*

- ❖ hierarchical routing saves table size, reduced update traffic

## *performance:*

- ❖ intra-AS: can focus on performance
- ❖ inter-AS: policy may dominate over performance

# Chapter 4: *done!*

---