

Chapter 4

Network Layer

Computer Networking: A Top Down Approach

6th edition

Jim Kurose, Keith Ross

Addison-Wesley

March 2012

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

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

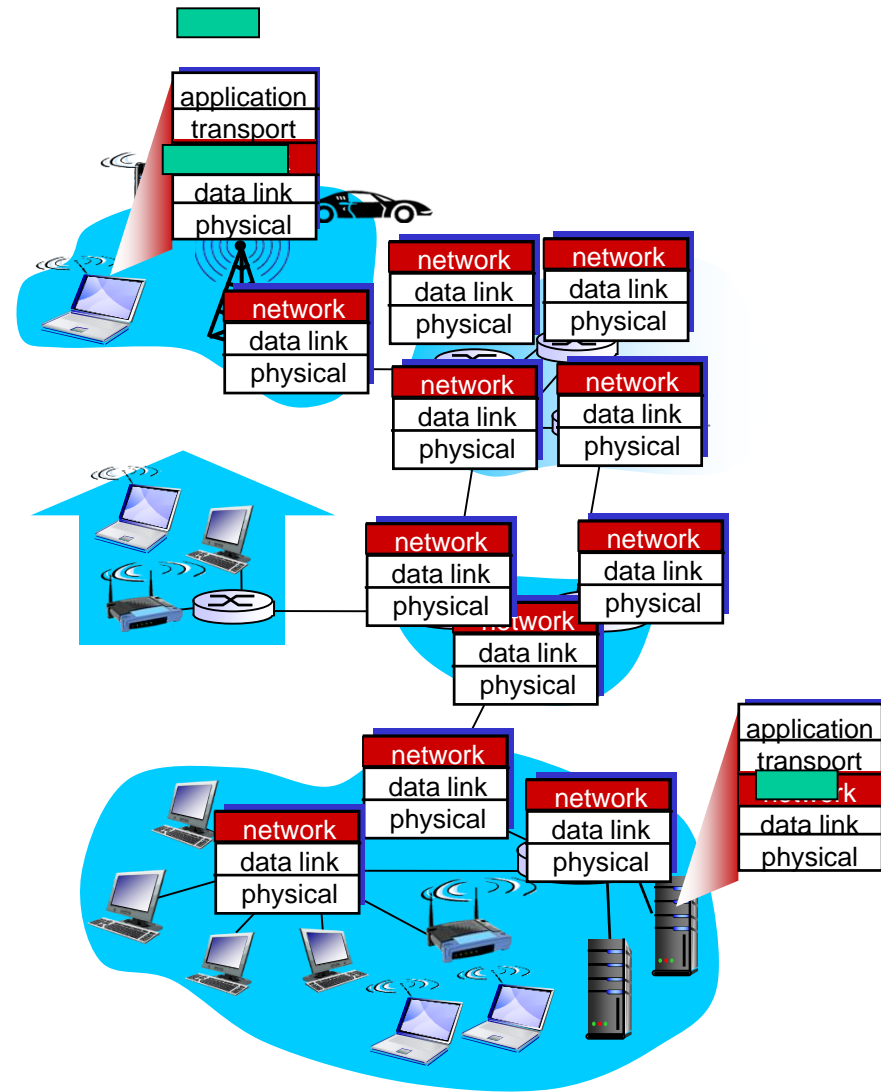
4.6 routing in the Internet

- 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.

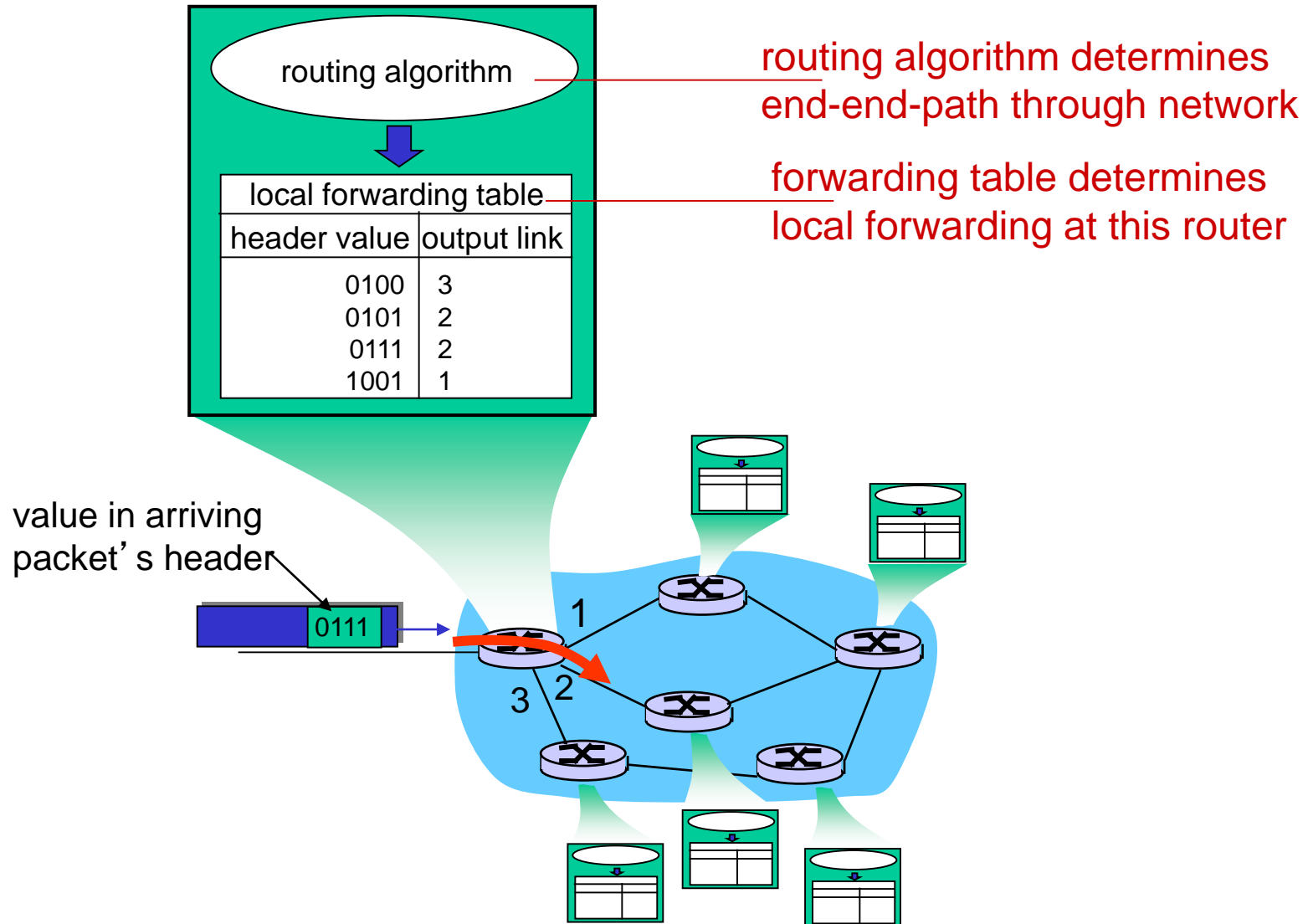
- *routing algorithms*

analogy:

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

❖ *forwarding*: process of getting through single interchange

Interplay between routing and forwarding



Connection setup

- 3rd important function in *some* network architectures
- ❖ before datagrams flow, two end hosts *and* intervening routers establish virtual connection
 - routers get involved

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 bounded delay

example services for a flow of datagrams:

- ❖ in-order datagram delivery
- ❖ guaranteed minimum bandwidth to flow
- ❖ Security services

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

Connection, connection-less service

- ❖ *datagram* network provides network-layer *connectionless* service
- ❖ *virtual-circuit* network provides network-layer *connection* service
- ❖ analogous to TCP/UDP connection-oriented / connectionless transport-layer services
- ❖ Features:
 - *service*: host-to-host
 - *no choice*: network provides one or the other
 - *implementation*: in network core

Virtual circuits (VC)

“source-to-dest path behaves much like telephone circuit”

performance-wise network actions along source-to-dest path

VC implementation

a VC consists of:

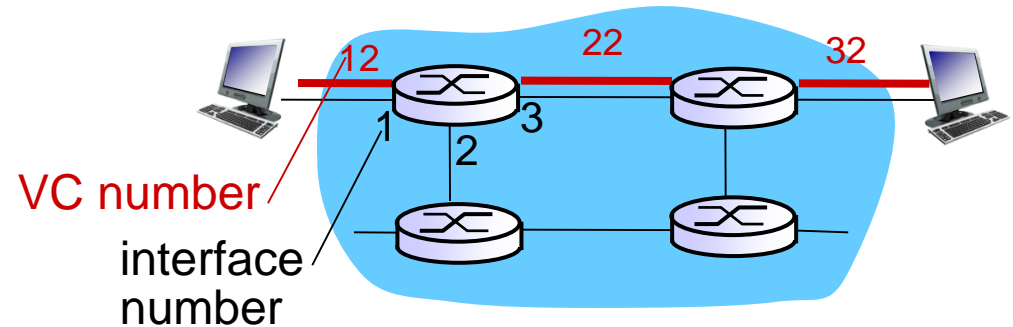
1. *path* from source to destination
2. *VC numbers*, one number for each link along path
3. *entries in forwarding tables* in routers along path

❖ packet belonging to VC carries VC number

❖ VC number can be changed on each link.

- new VC number comes from forwarding table

VC forwarding table



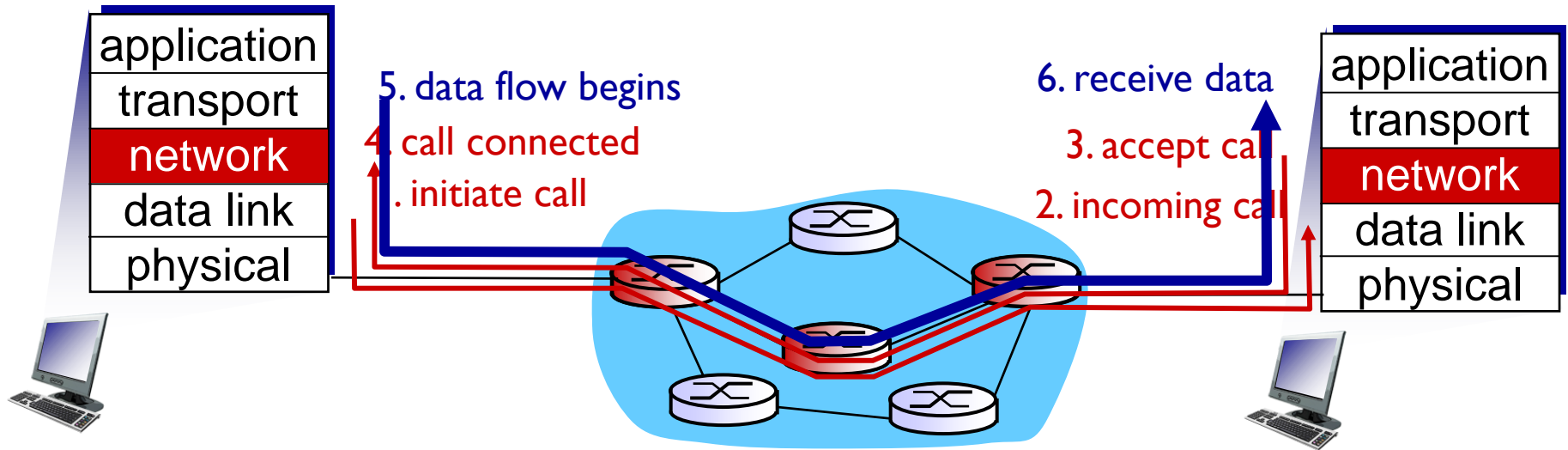
*forwarding table in
northwest router:*

Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
...

VC routers maintain connection state information!

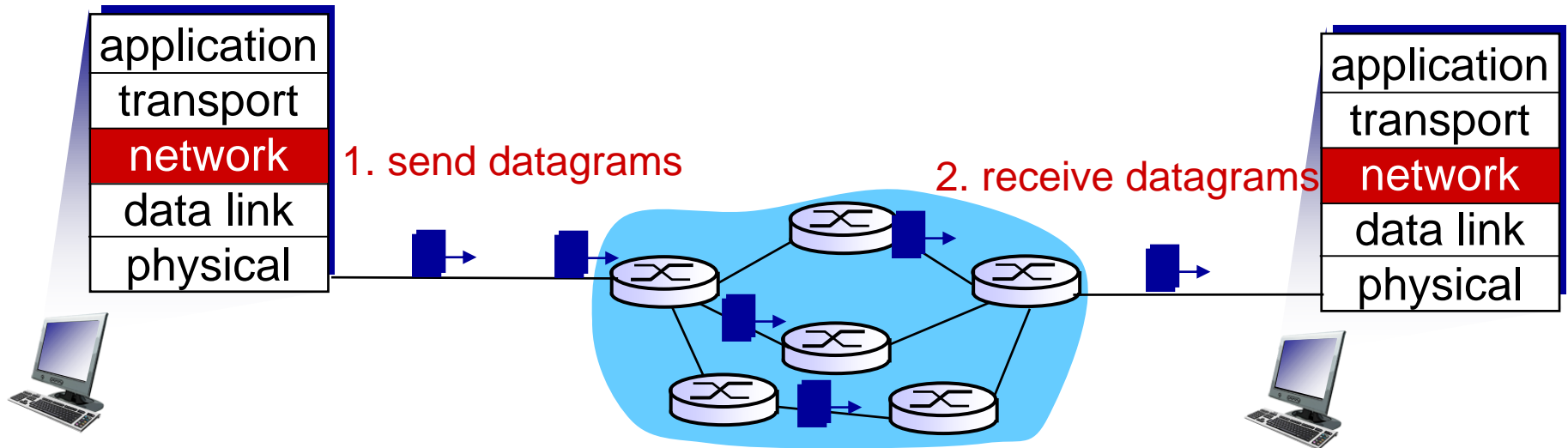
Virtual circuits: signaling protocols

- ❖ used to setup, maintain teardown VC
- ❖ used in ATM, frame-relay, X.25
- ❖ not used in today's Internet

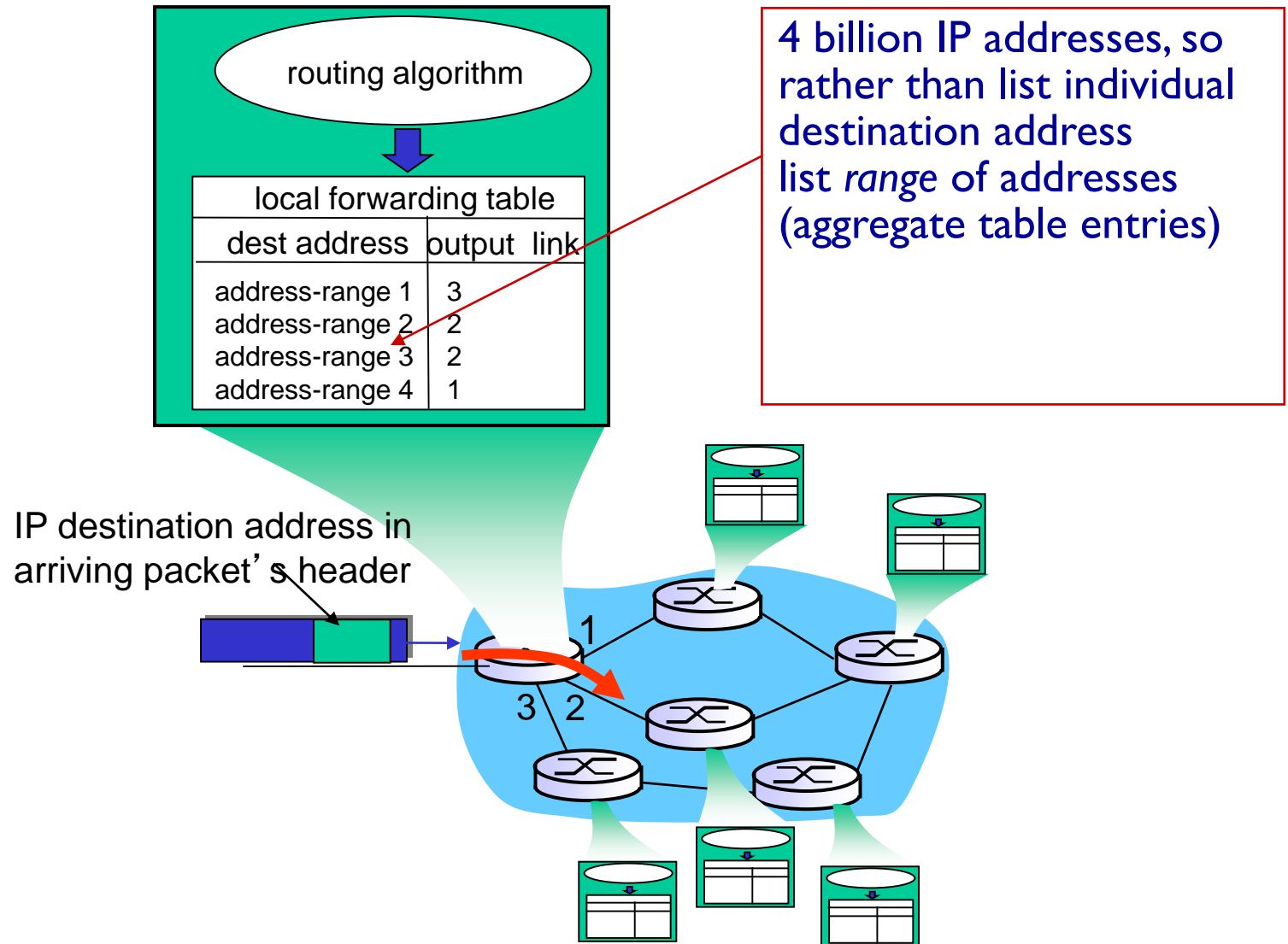


Datagram networks

- ❖ no call setup at network layer
- ❖ routers: no state about end-to-end connections
 - no network-level concept of “connection”
- ❖ packets forwarded using destination host address



Datagram forwarding table



Datagram forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

but what happens if ranges don't divide up so nicely?

Longest prefix matching

longest prefix matching

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

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

examples:

DA: 11001000 00010111 00010110 10100001

which interface?

DA: 11001000 00010111 00011000 10101010

which interface?

Datagram or VC network: why?

Internet (datagram)

- ❖ data exchange among computers
 - “elastic” service, no strict timing req.
- ❖ many link types
 - different characteristics
 - uniform service difficult
- ❖ “smart” end systems (computers)
 - can adapt, perform control, error recovery
 - ***simple inside network, complexity at “edge”***

ATM (VC)

- ❖ evolved from telephony
- ❖ human conversation:
 - strict timing, reliability requirements
 - need for guaranteed service
- ❖ “dumb” end systems
 - telephones
 - ***complexity inside 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

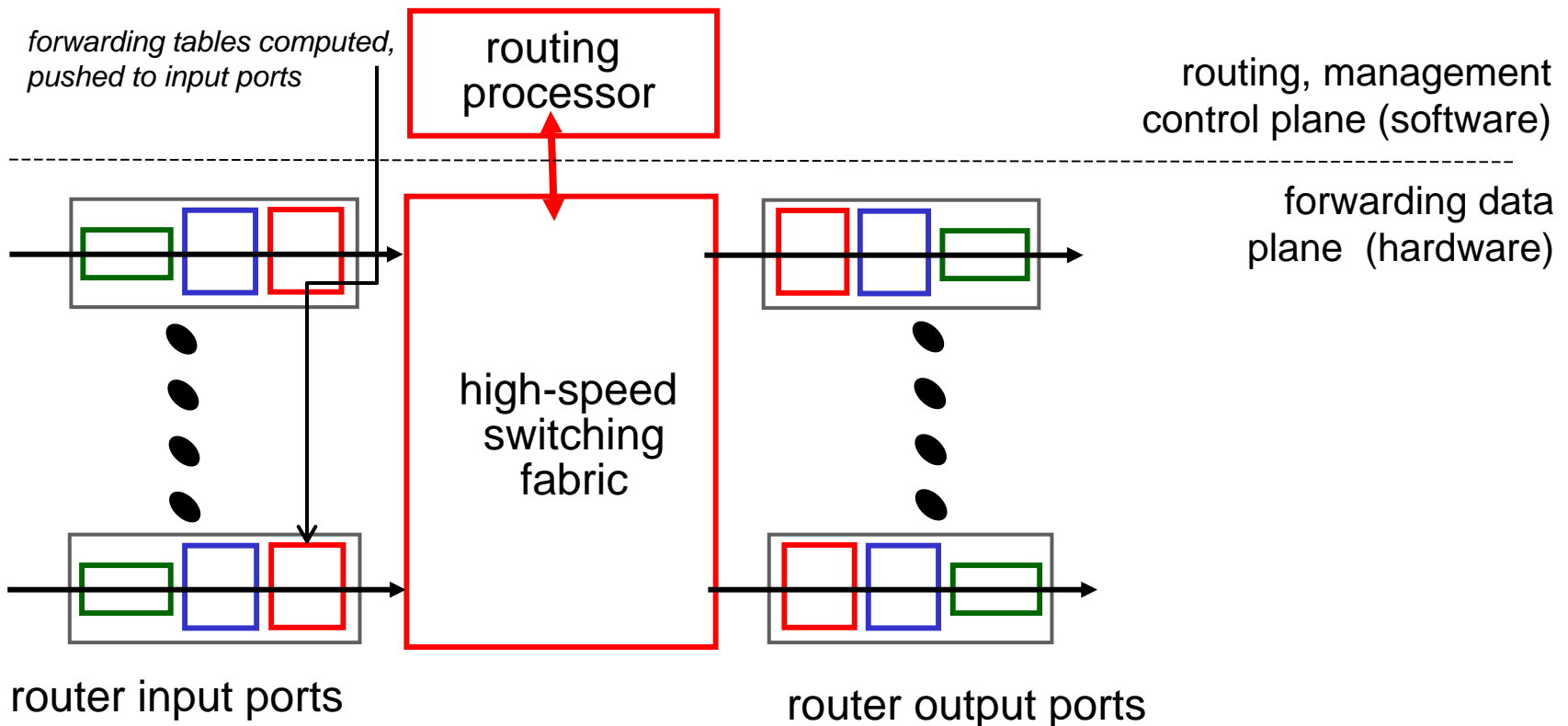
- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

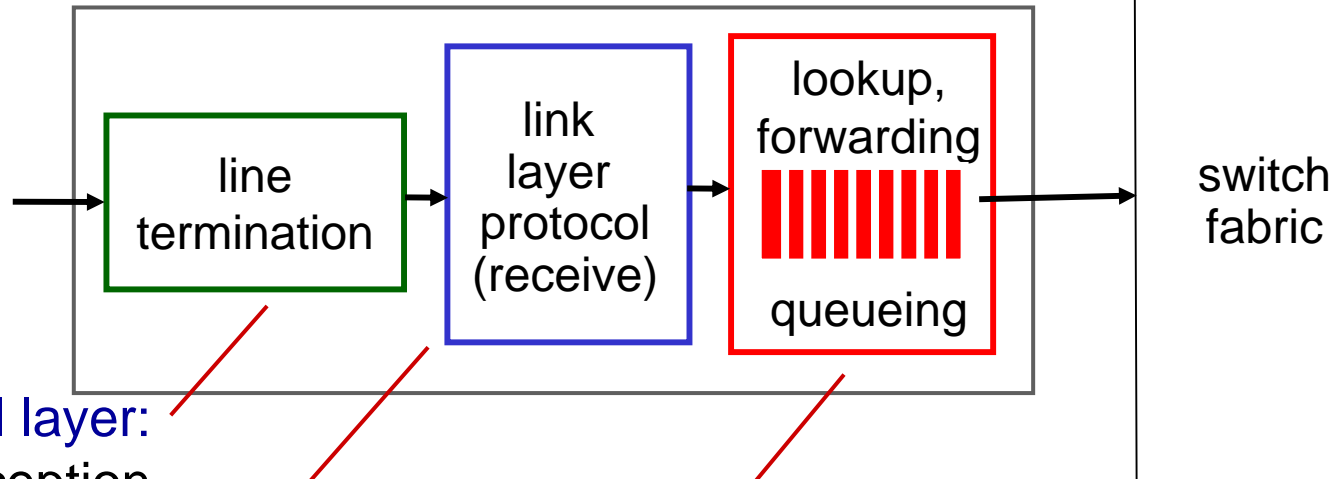
Router architecture overview

two key router functions:

- ❖ run routing algorithms/protocol
- ❖ *forwarding* datagrams from incoming to outgoing link



Input port functions



physical layer:
bit-level reception

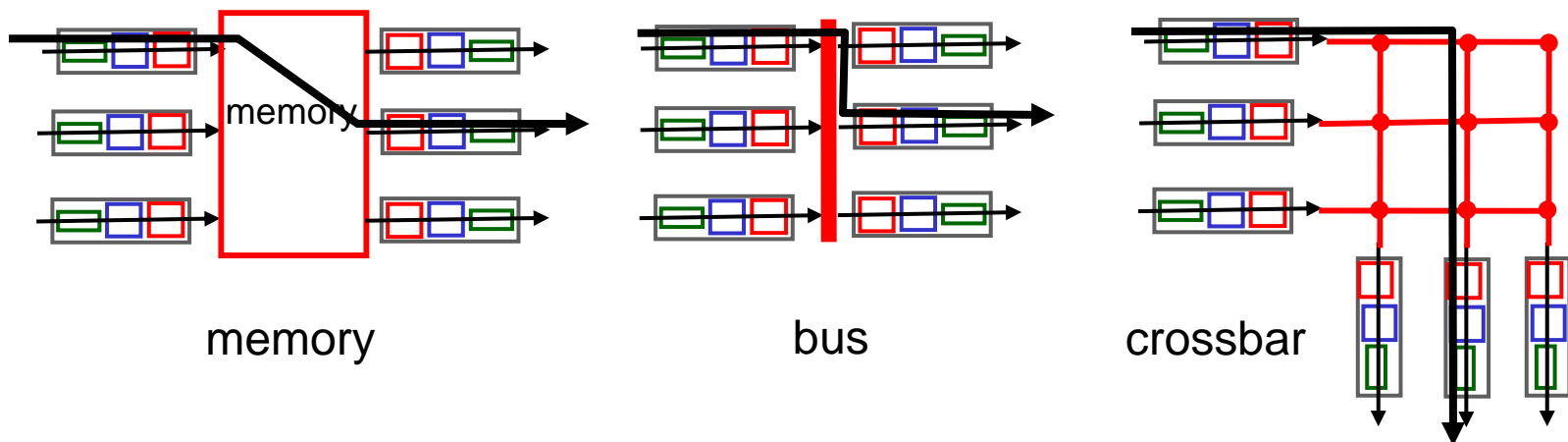
data link layer:
e.g., Ethernet
see chapter 5

decentralized switching:

- ❖ given datagram dest., lookup output port using forwarding table in input port memory
- ❖ goal: complete input port processing at 'line speed'
- ❖ queueing: if datagrams arrive faster than forwarding rate into switch fabric

Switching fabrics

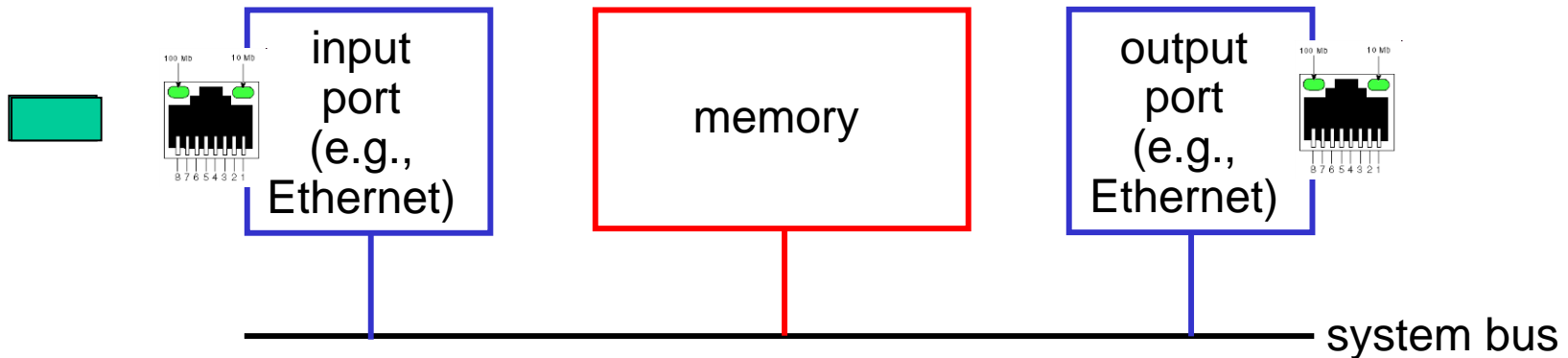
- ❖ transfer packet from input buffer to appropriate output buffer
- ❖ switching rate: rate at which packets can be transferred from inputs to outputs
 - often measured as multiple of input/output line rate
 - N inputs: switching rate N times line rate desirable
- ❖ three types of switching fabrics



Switching via memory

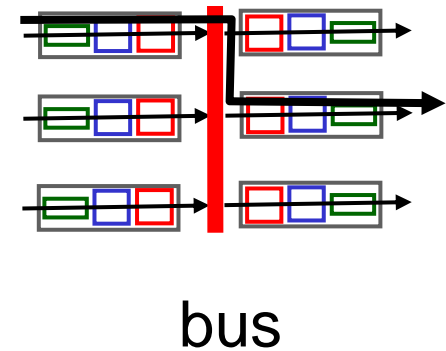
first generation routers:

- ❖ traditional computers with switching under direct control of CPU
- ❖ packet copied to system's memory
- ❖ speed limited by memory bandwidth (2 bus crossings per datagram)



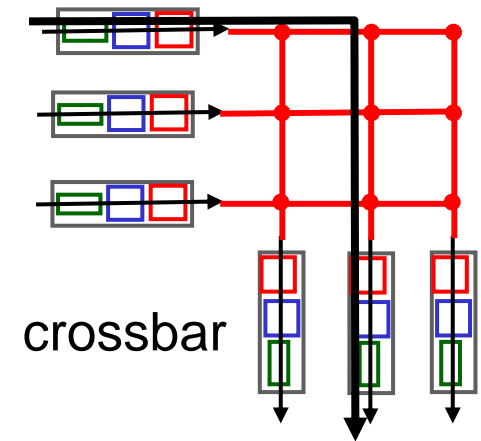
Switching via a bus

- ❖ datagram from input port memory to output port memory via a shared bus
- ❖ *bus contention*: switching speed limited by bus bandwidth

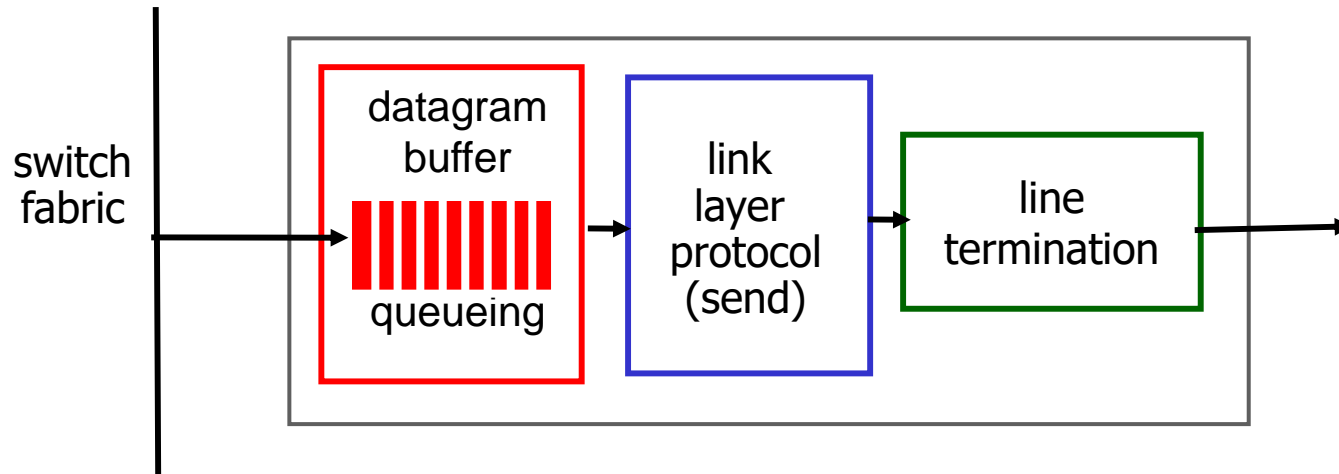


Switching via interconnection network

- ❖ overcome bus bandwidth limitations
- ❖ banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- ❖ advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.

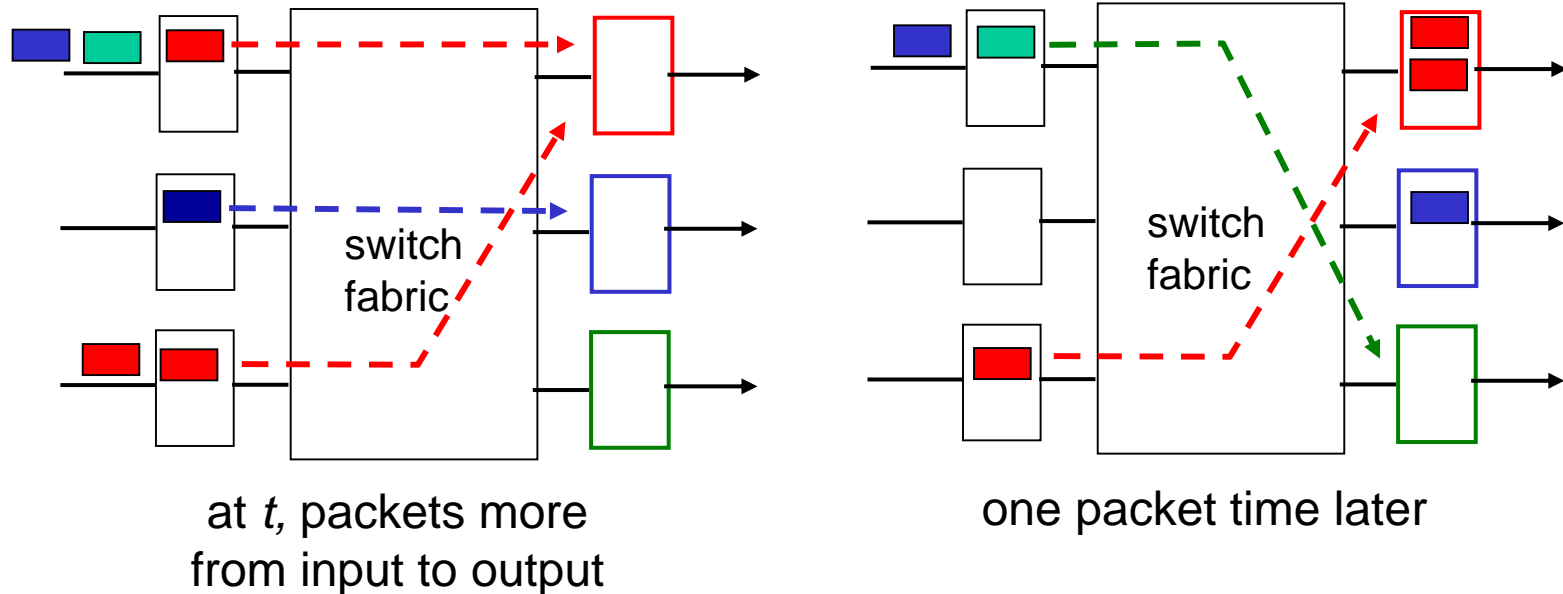


Output ports



- ❖ *buffering* required from fabric faster rate
 - ❖ *scheduling discipline* chooses among queued datagrams for transmission
- Datagram (packets) can be lost due to congestion, lack of buffers

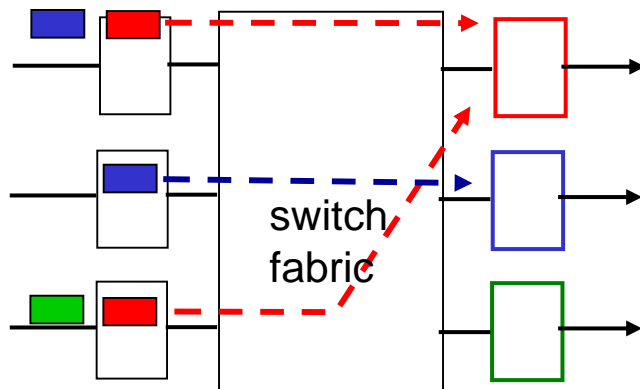
Output port queueing



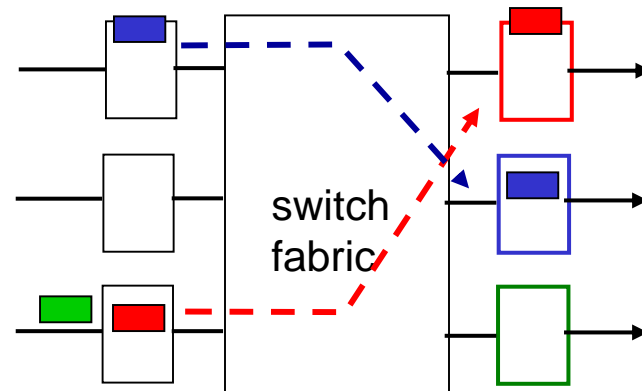
- ❖ buffering when arrival rate via switch exceeds output line speed
- ❖ *queueing (delay) and loss due to output port buffer overflow!*

Input port queuing

- ❖ fabric slower than input ports combined -> queueing may occur at input queues
 - *queueing delay and loss due to input buffer overflow!*
- ❖ **Head-of-the-Line (HOL) blocking:** queued datagram in front of queue prevents others in queue from moving forward



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



one packet time later:
green packet
experiences HOL
blocking

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

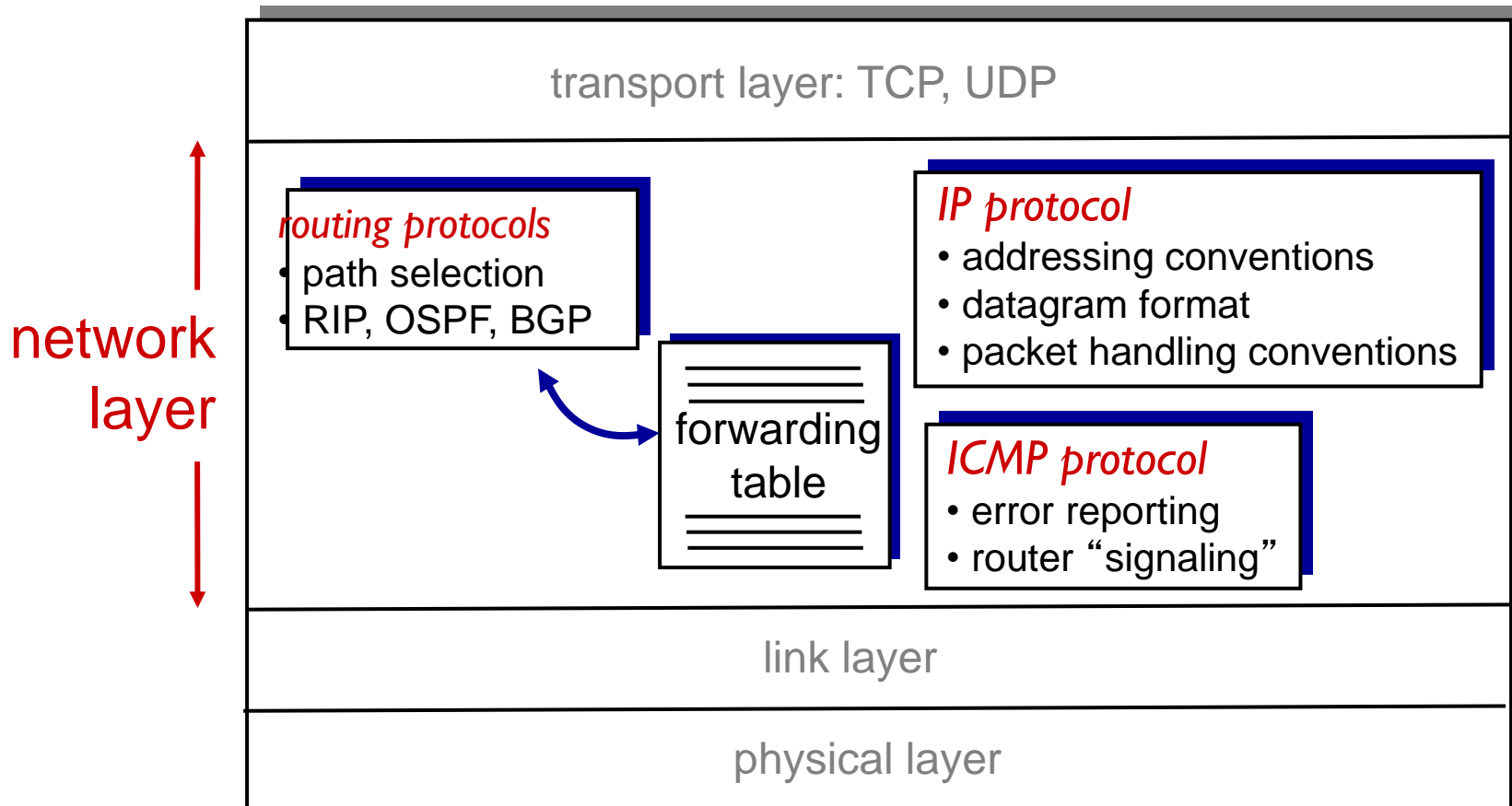
4.6 routing in the Internet

- RIP
- OSPF
- BGP

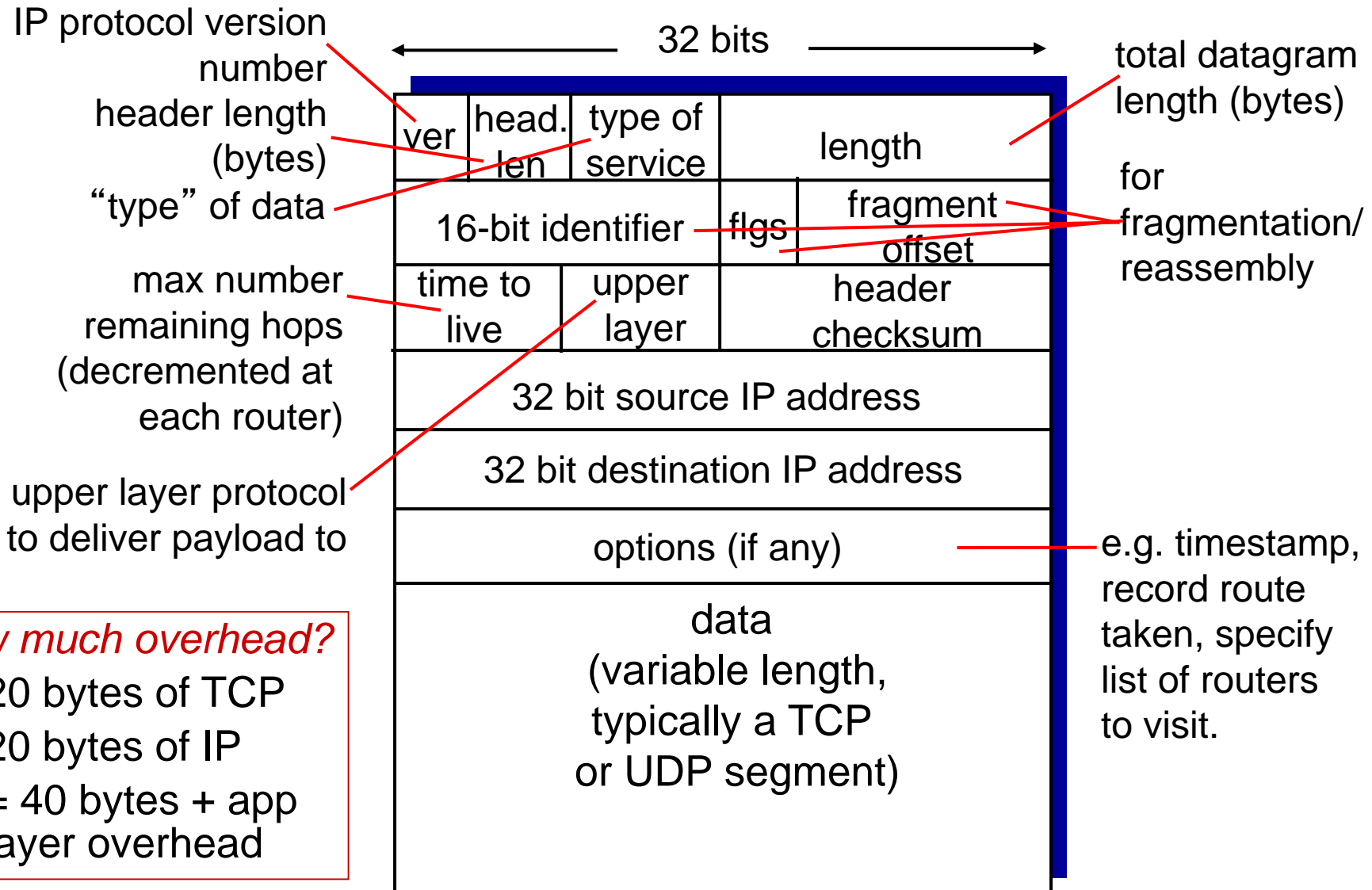
4.7 broadcast and multicast routing

The Internet network layer

host, router network layer functions:

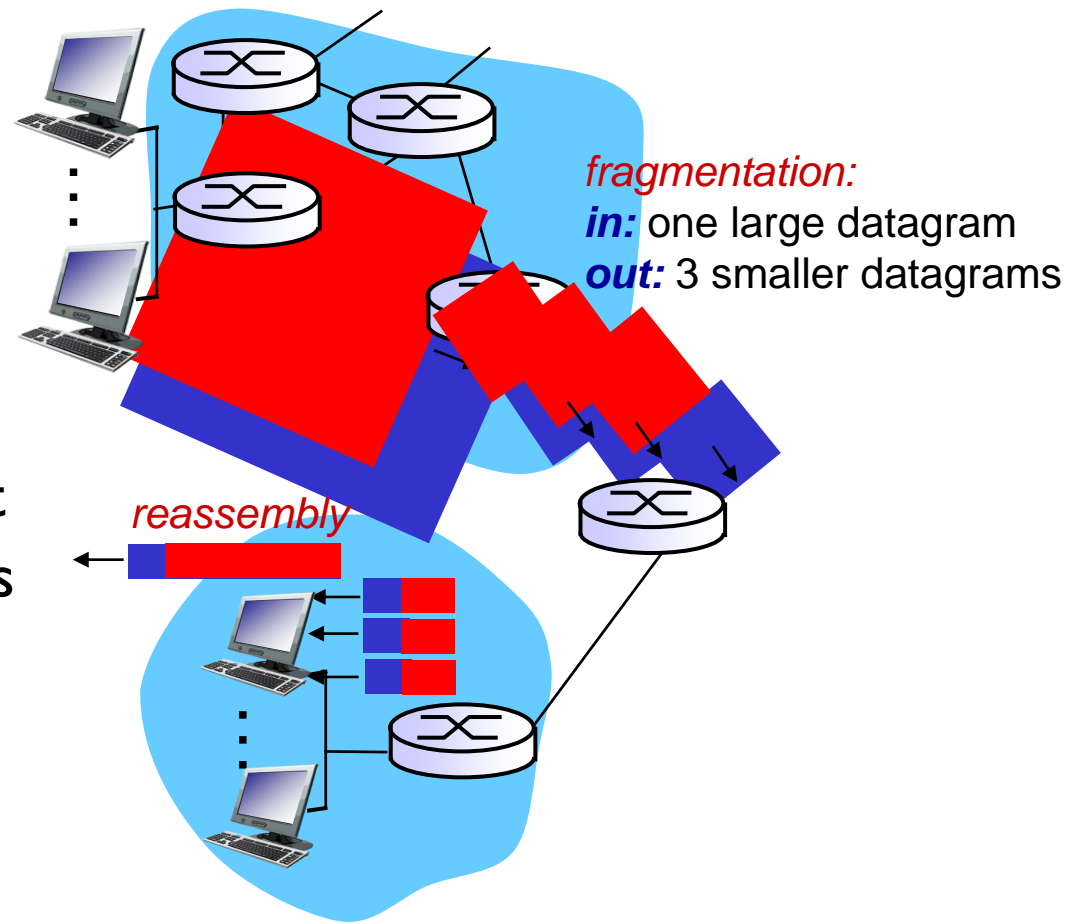


IP datagram format



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

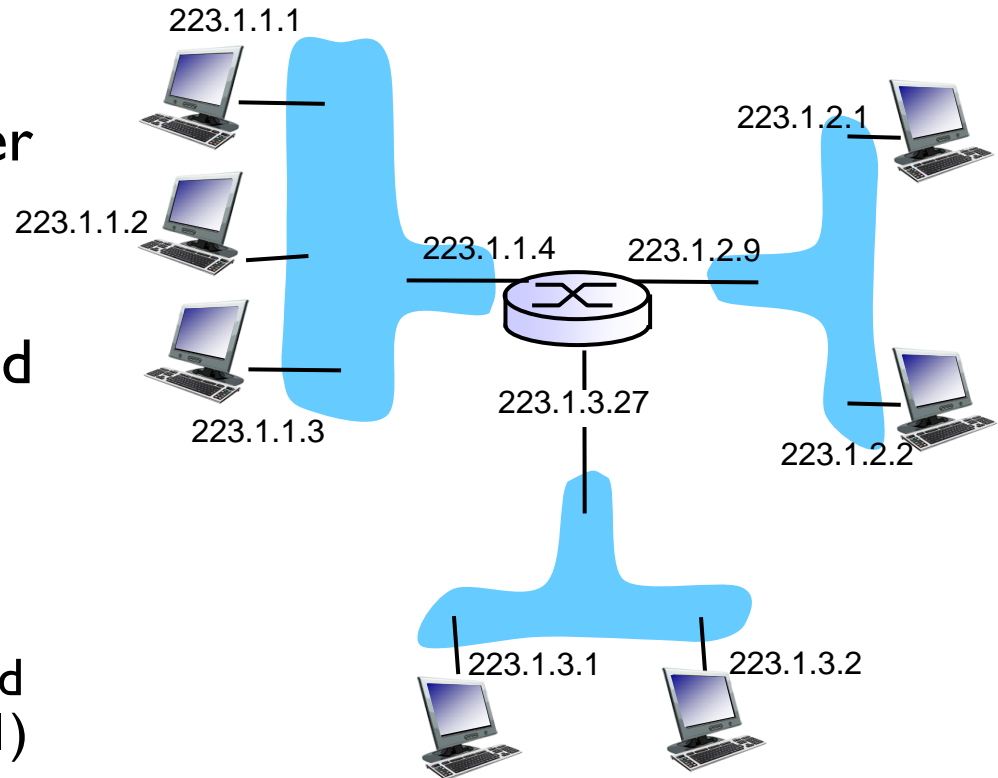
4.6 routing in the Internet

- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

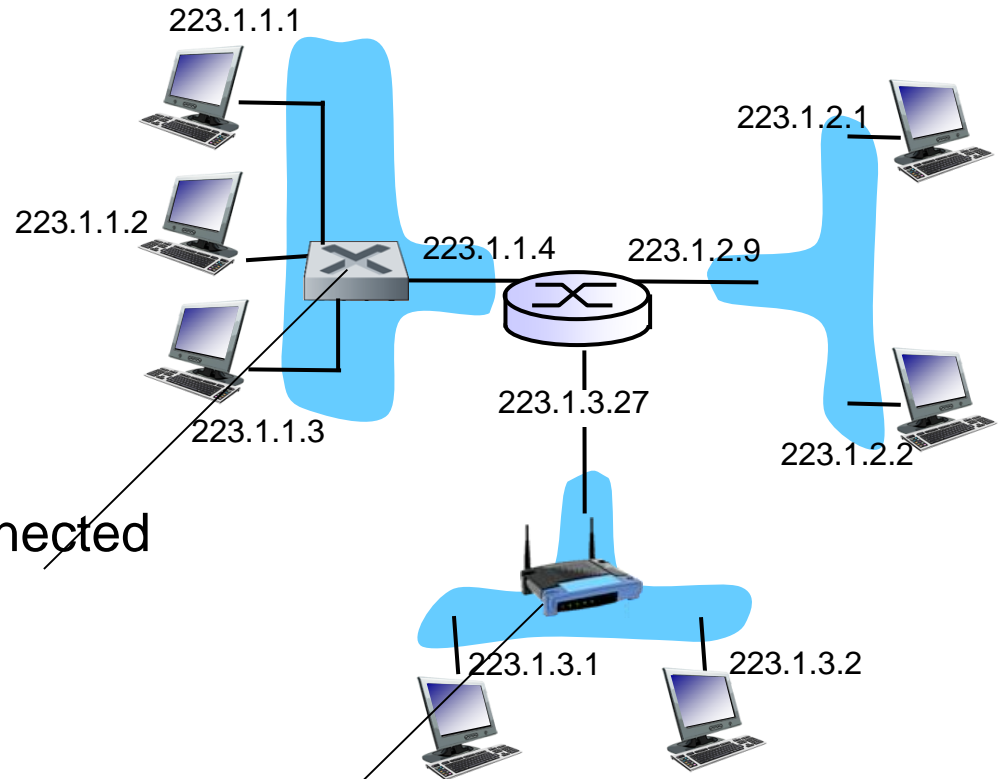
IP addressing: introduction

- ❖ **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



wired Ethernet interfaces connected
by Ethernet switches

A: wireless WiFi interfaces
connected by WiFi base station

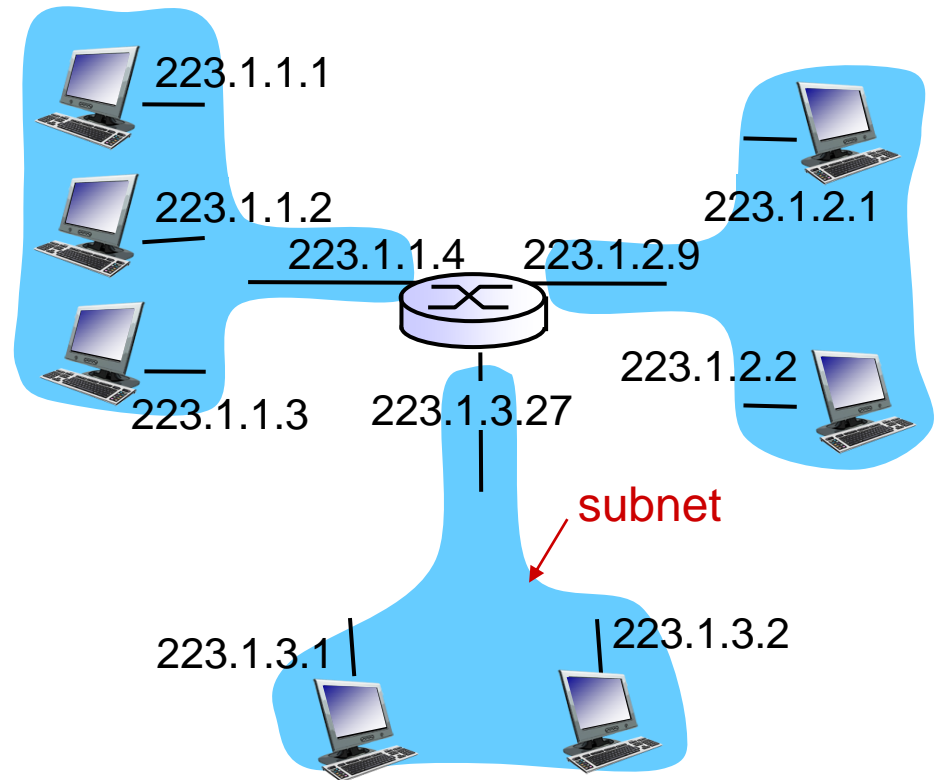
Subnets

❖ IP address:

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

❖ *what 's a subnet ?*

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

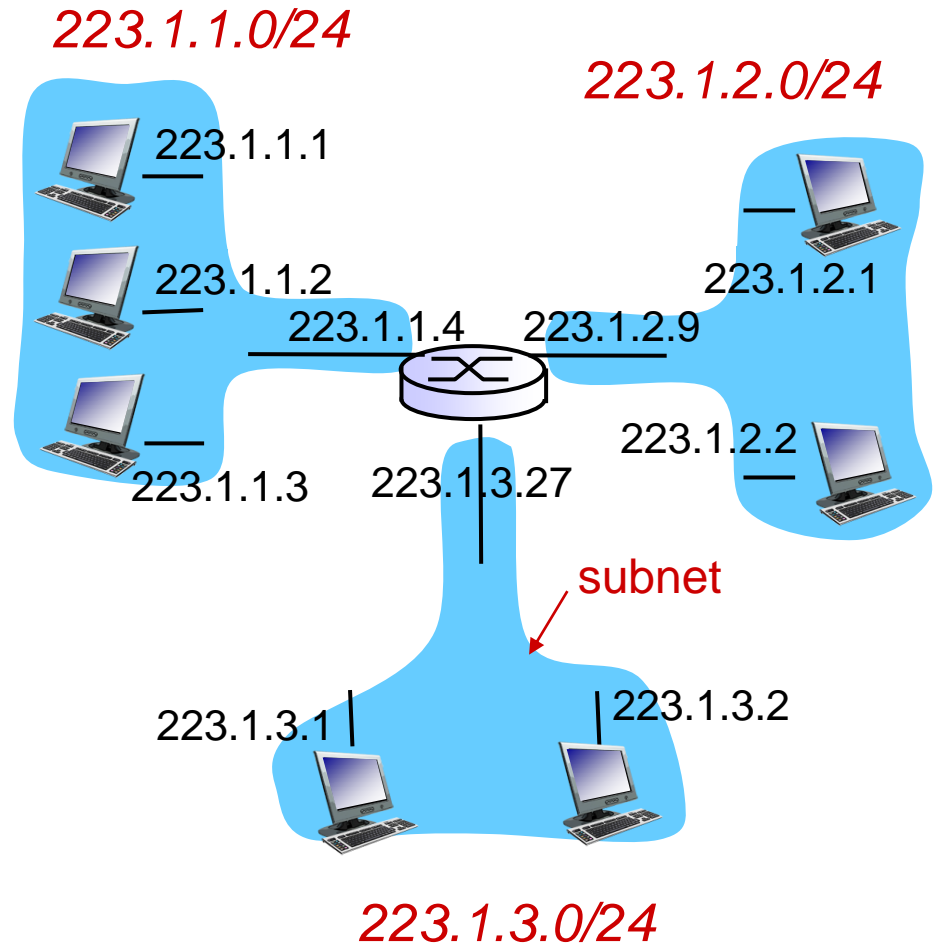


network consisting of 3 subnets

Subnets

recipe

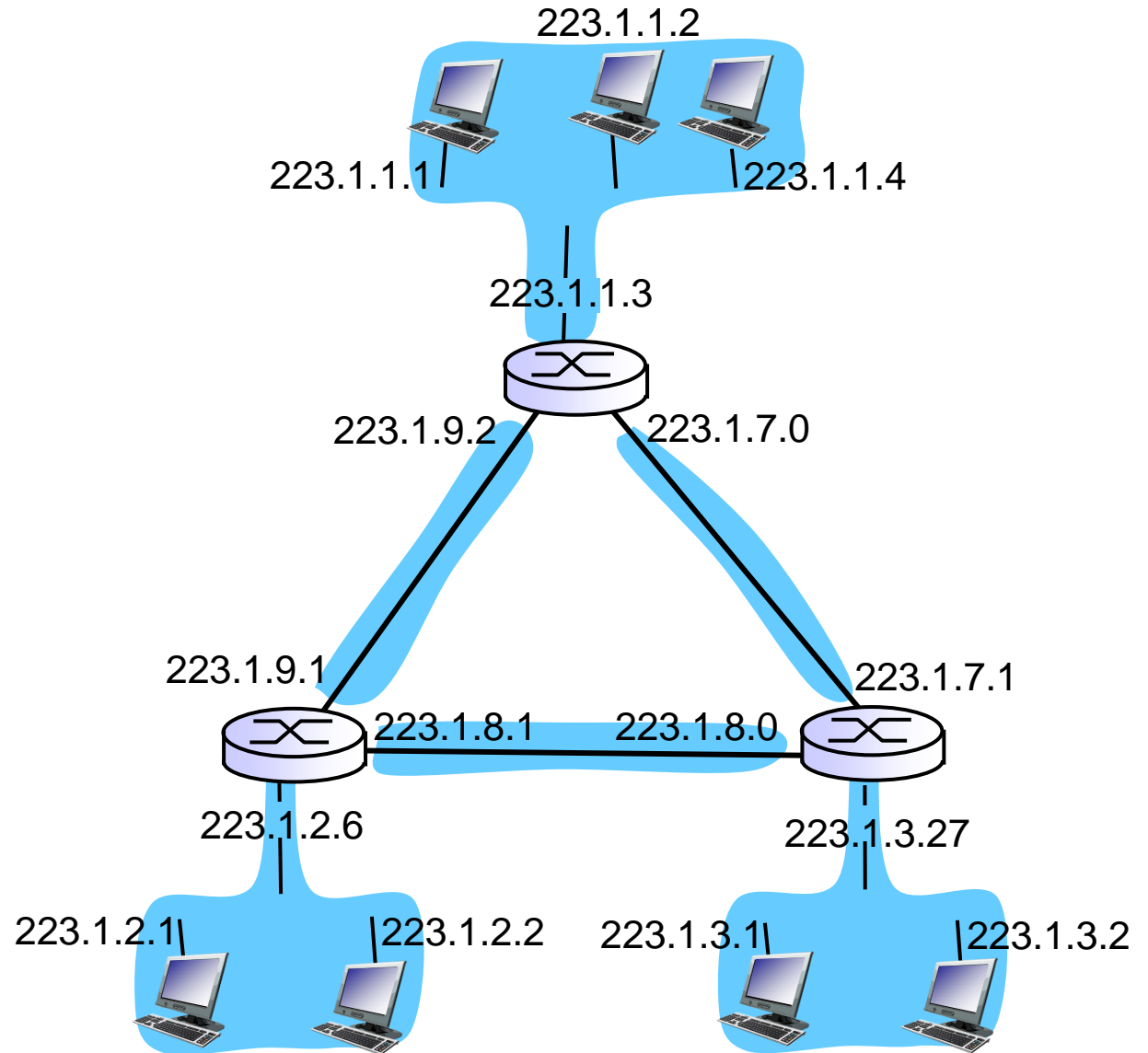
- ❖ to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- ❖ each isolated network is called a *subnet*



subnet mask: /24

Subnets

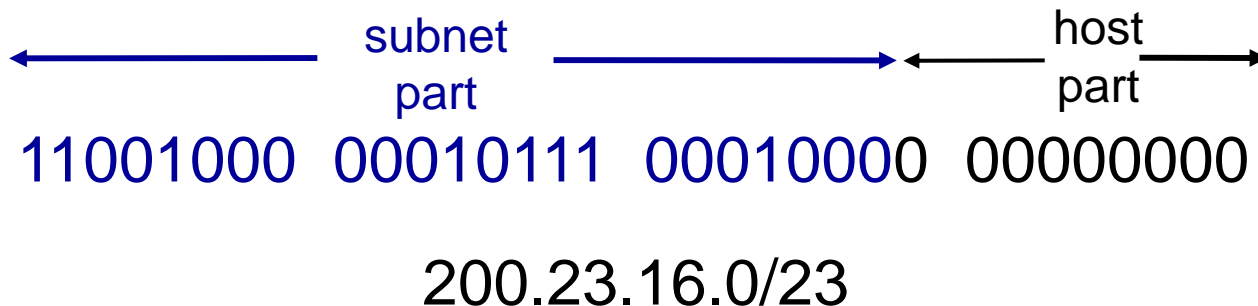
how many?



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

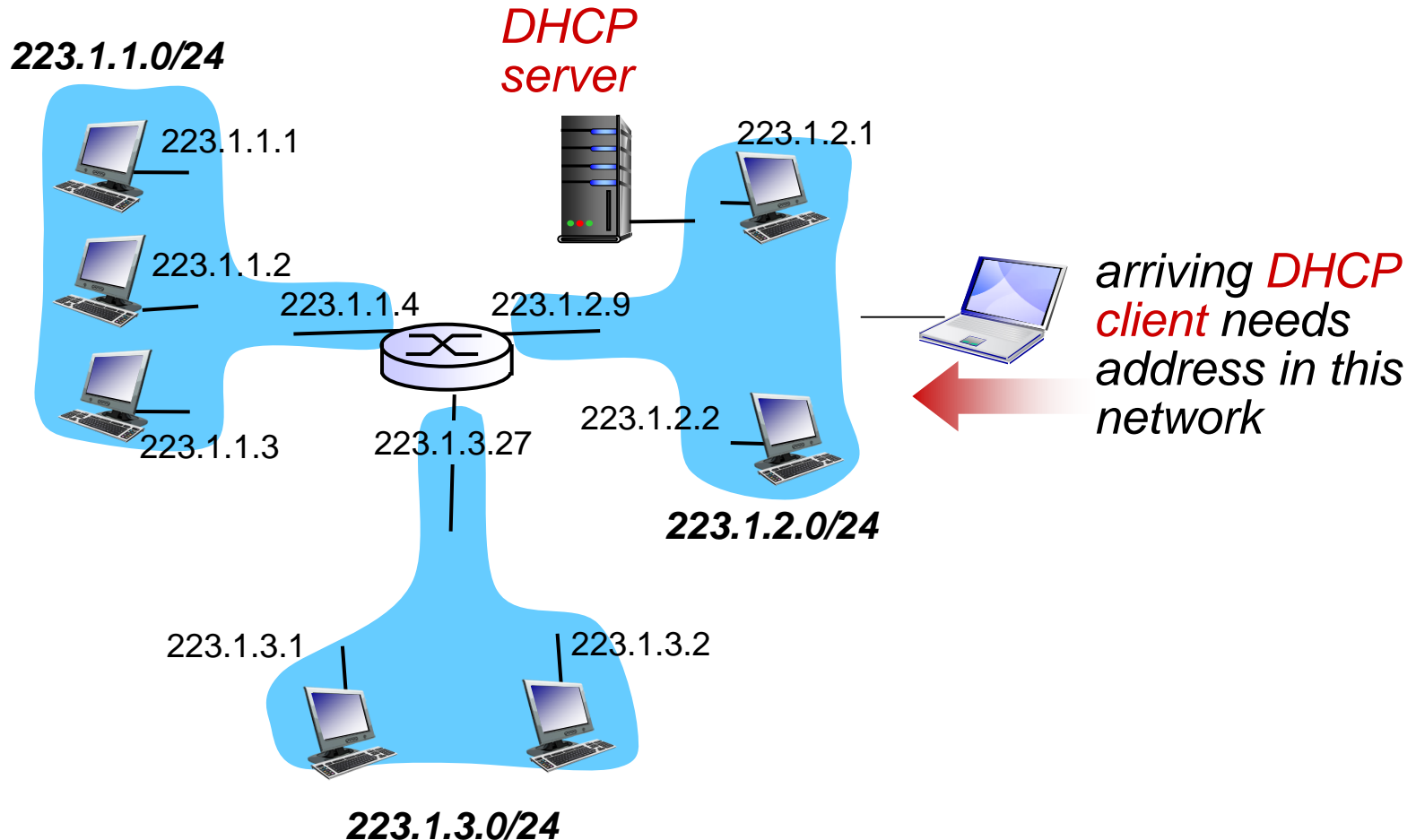
goal: allow host to *dynamically* obtain its IP address from network server when it joins network

- 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

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

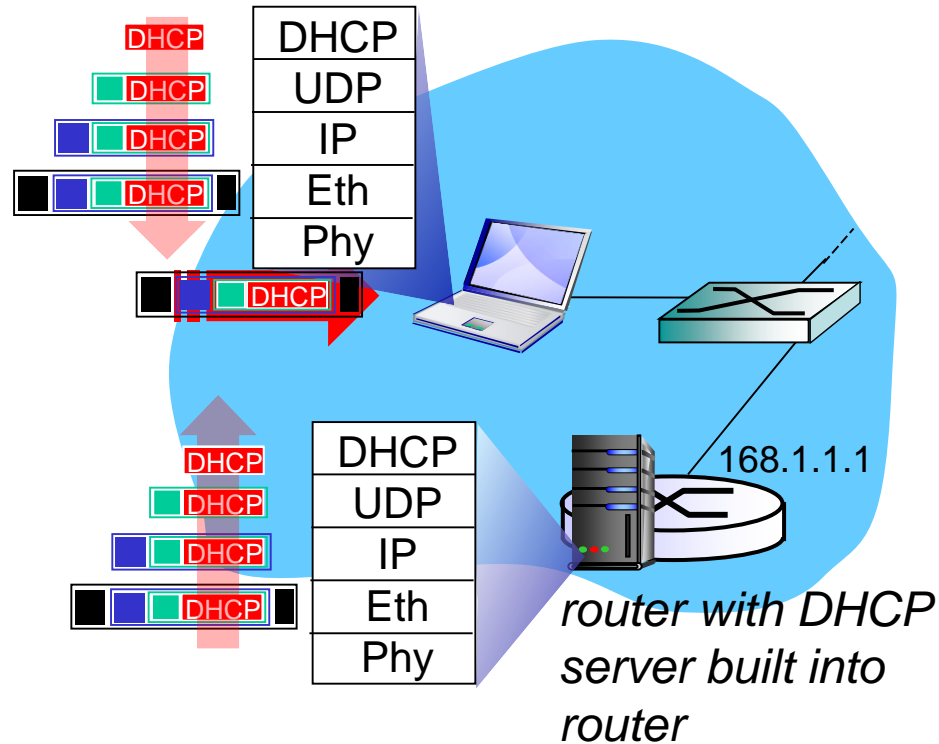
Broadcast: OK. You've
got that IP address!
.....

DHCP: more than IP addresses

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

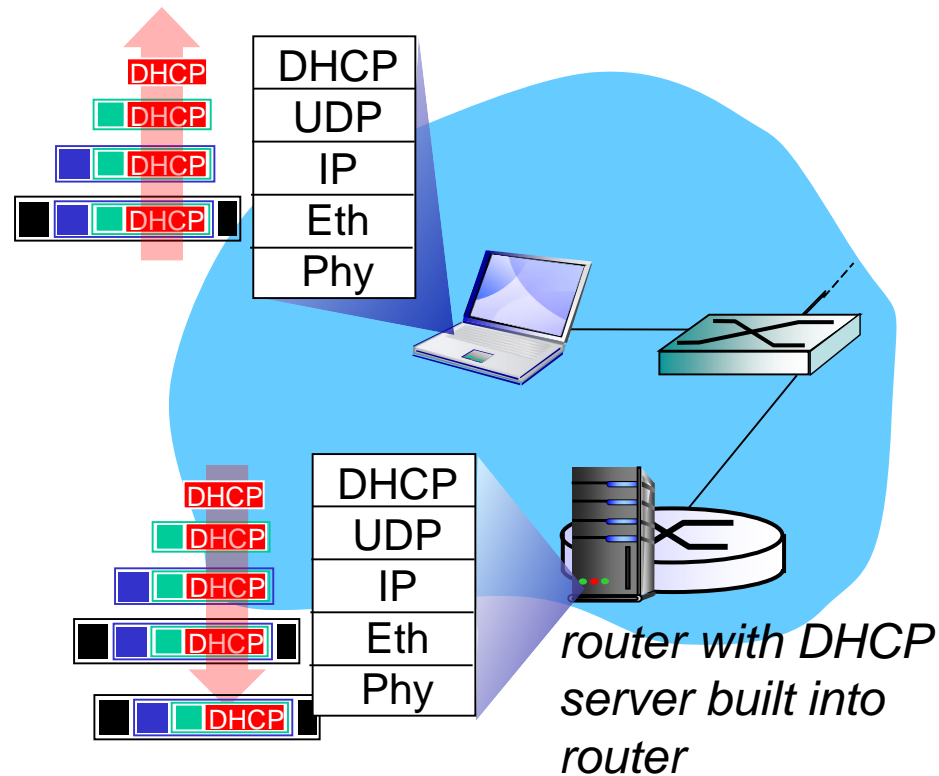
- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

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: FFFFFFFFFFFFFFFF) 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

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

IPv6: motivation

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

IPv6 datagram format:

- fixed-length 40 byte header
- no fragmentation allowed

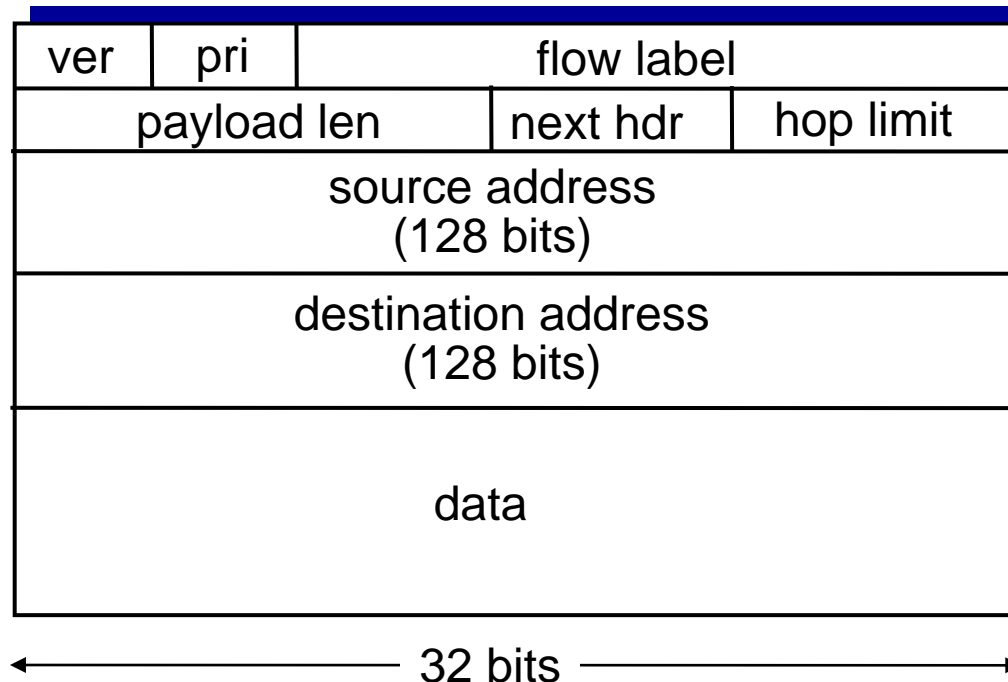
IPv6 datagram format

priority: identify priority among datagrams in flow

flow Label: identify datagrams in same “flow.”

(concept of “flow” not well defined).

next header: identify upper layer protocol for data

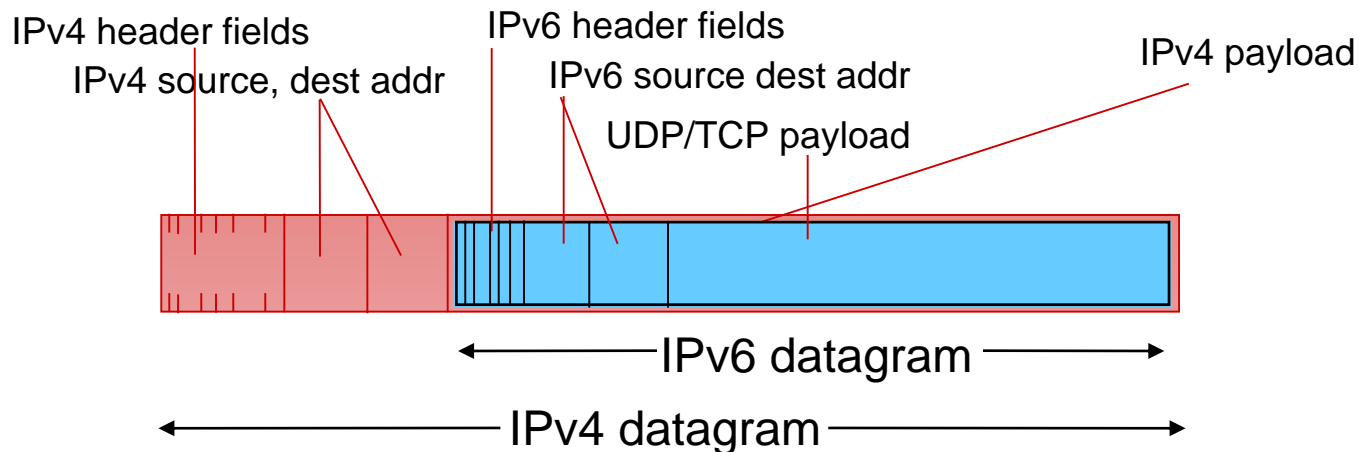


Other changes from IPv4

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

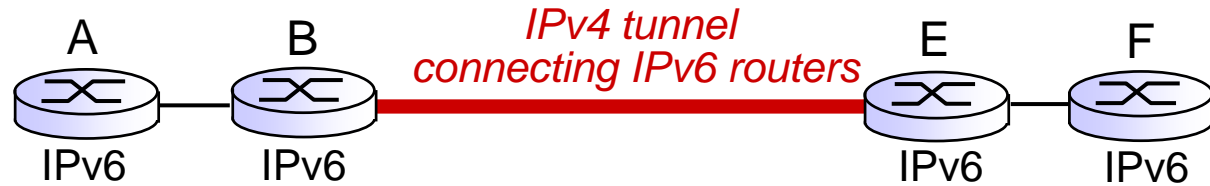
Transition from IPv4 to IPv6

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

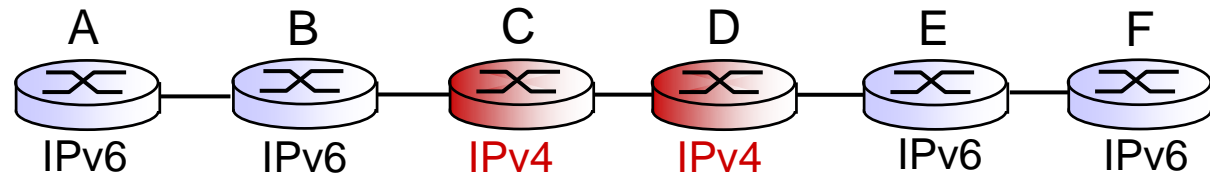


Tunneling

logical view:

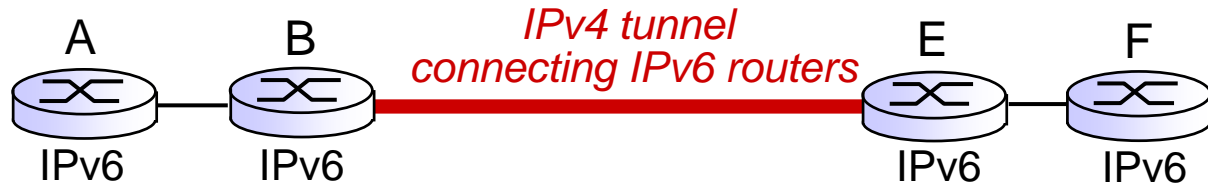


physical view:

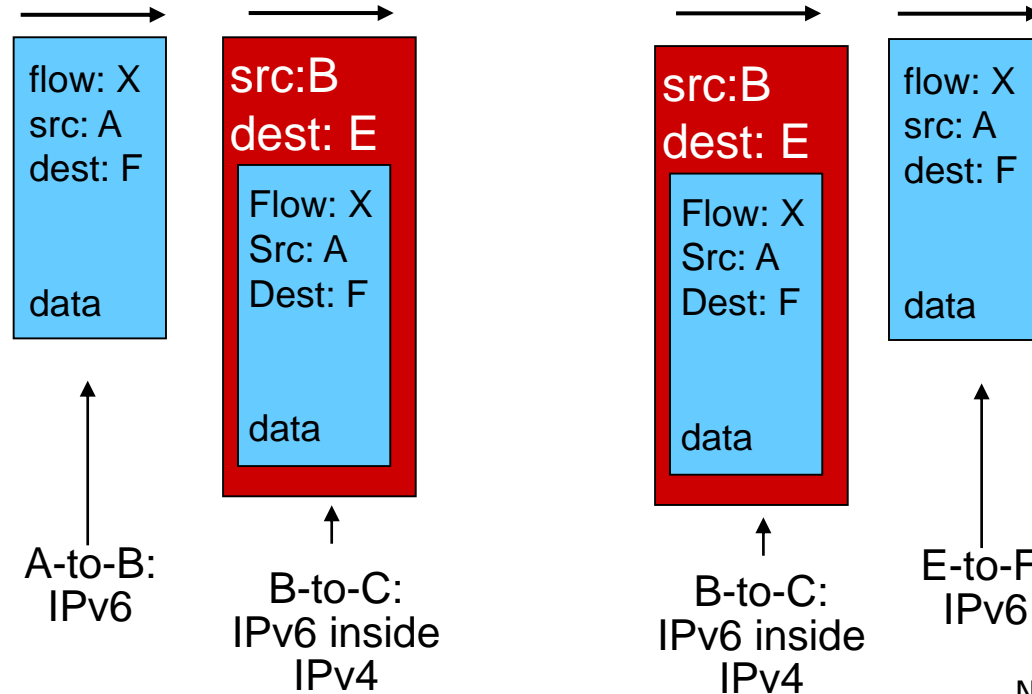
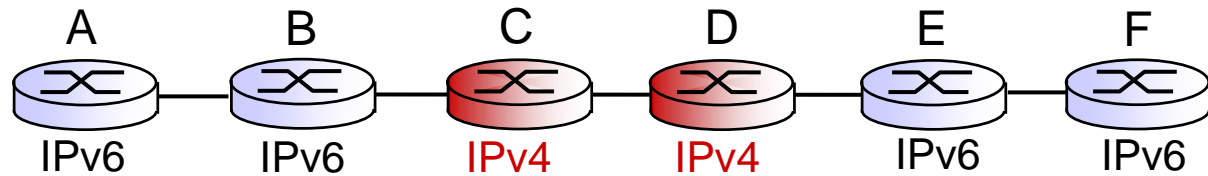


Tunneling

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

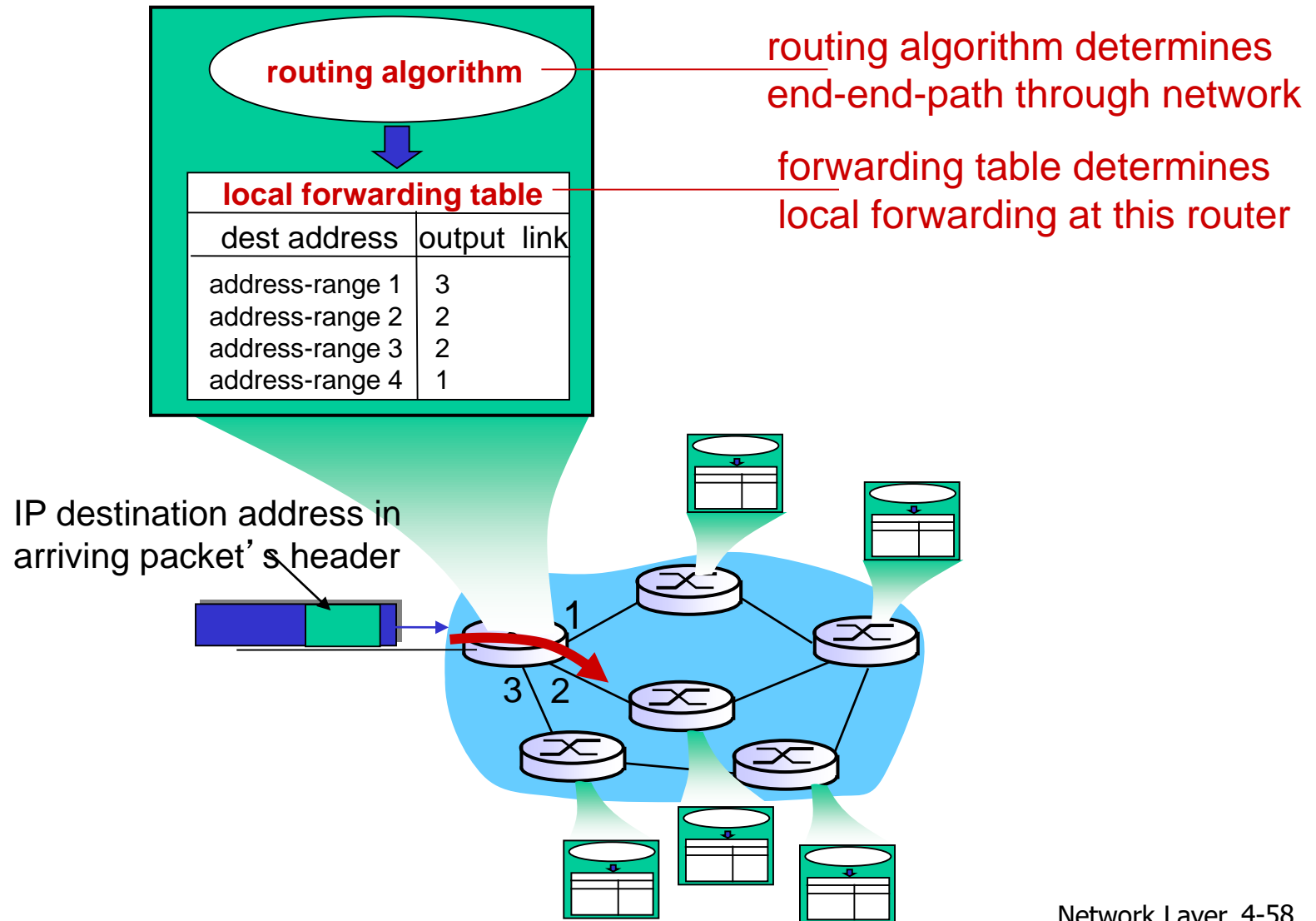
- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet

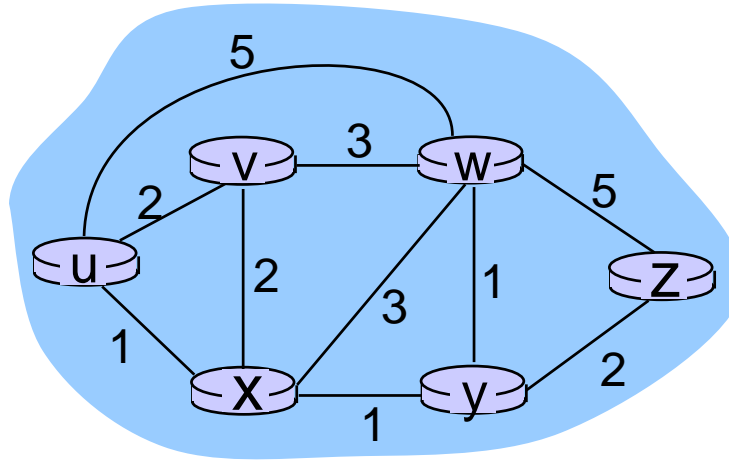
- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Interplay between routing, forwarding



Graph abstraction



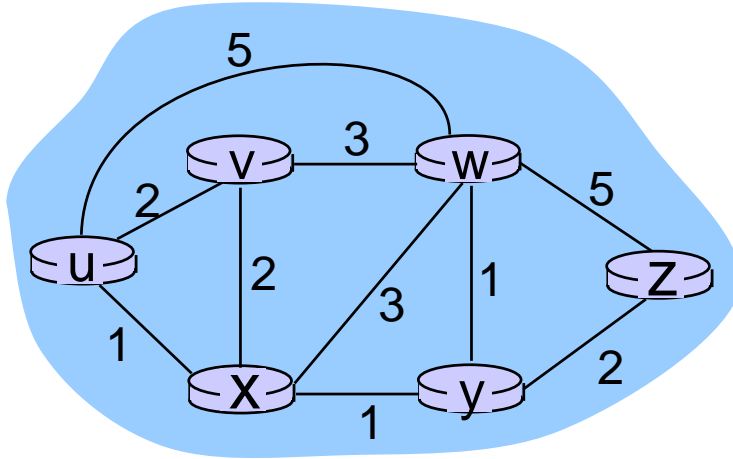
graph: $G = (N, E)$

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



$c(x,x')$ = 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 */

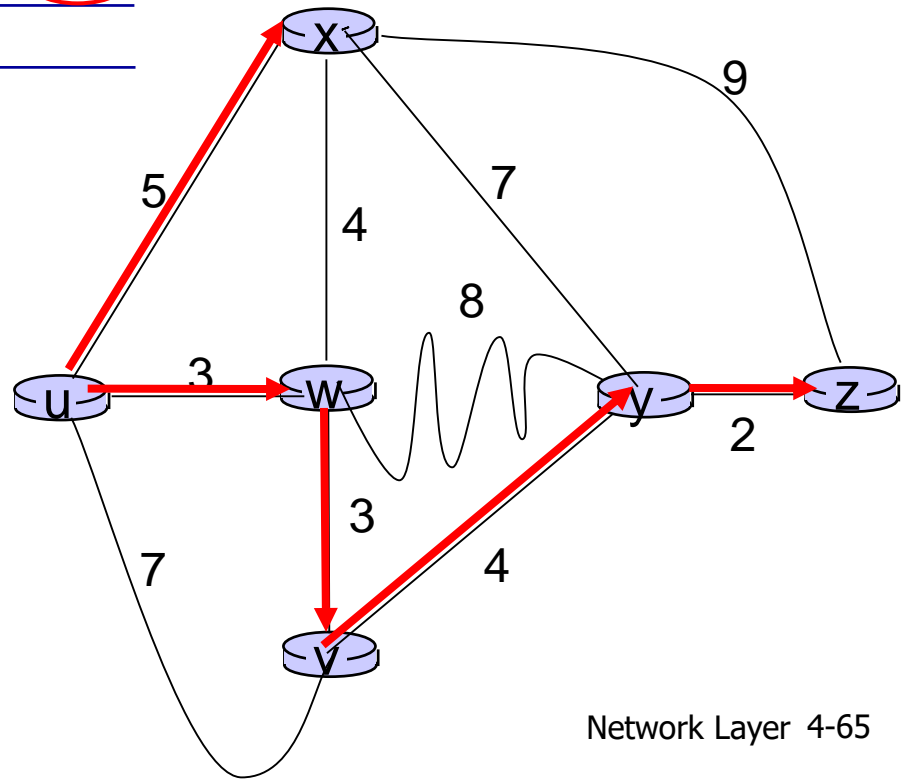
15 **until all nodes in N'**

Dijkstra's algorithm: example

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) 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)



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

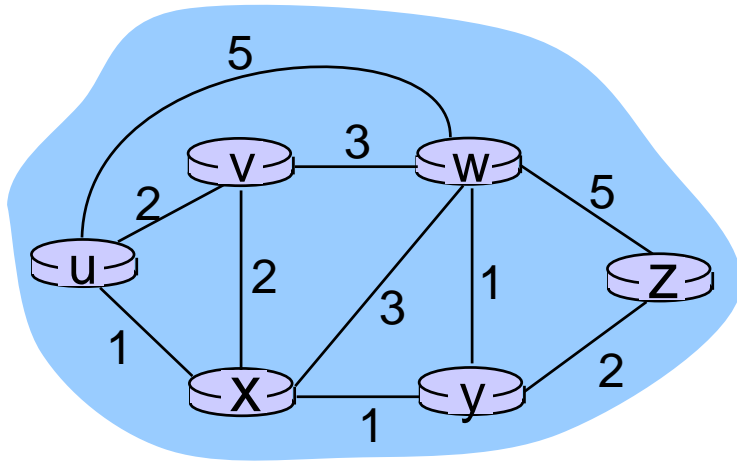
then

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

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

Distance vector algorithm

iterative, asynchronous:

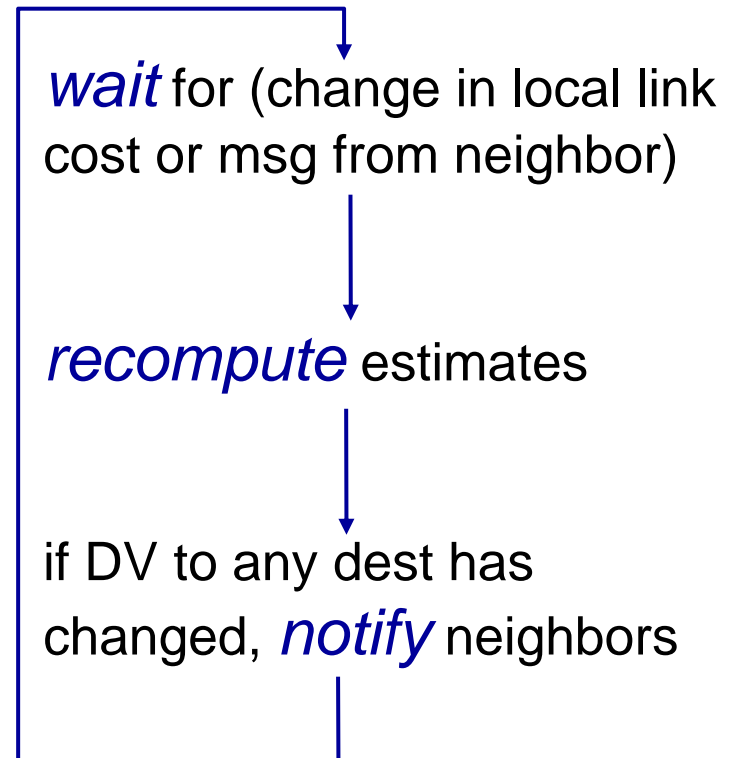
each local iteration
caused by:

- ❖ local link cost change
- ❖ DV update message from neighbor

distributed:

- ❖ each node notifies neighbors *only* when its DV changes
 - neighbors then notify their neighbors if necessary

each node:



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

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

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

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

**node x
table**

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

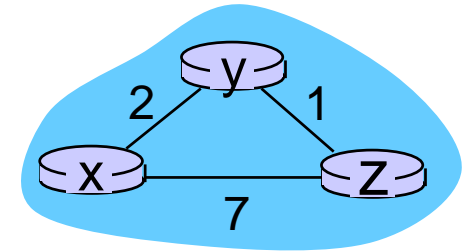
		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	7	1	0

**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

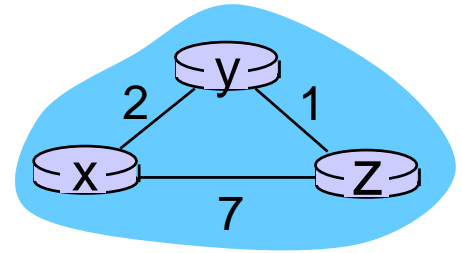
		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

		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