

# Chapter 4: Network Layer

## Chapter goals:

- ❑ understand principles behind network layer services:
  - routing (path selection)
  - dealing with scale
  - how a router works
  - advanced topics: IPv6, multicast
- ❑ instantiation and implementation in the Internet

## Overview:

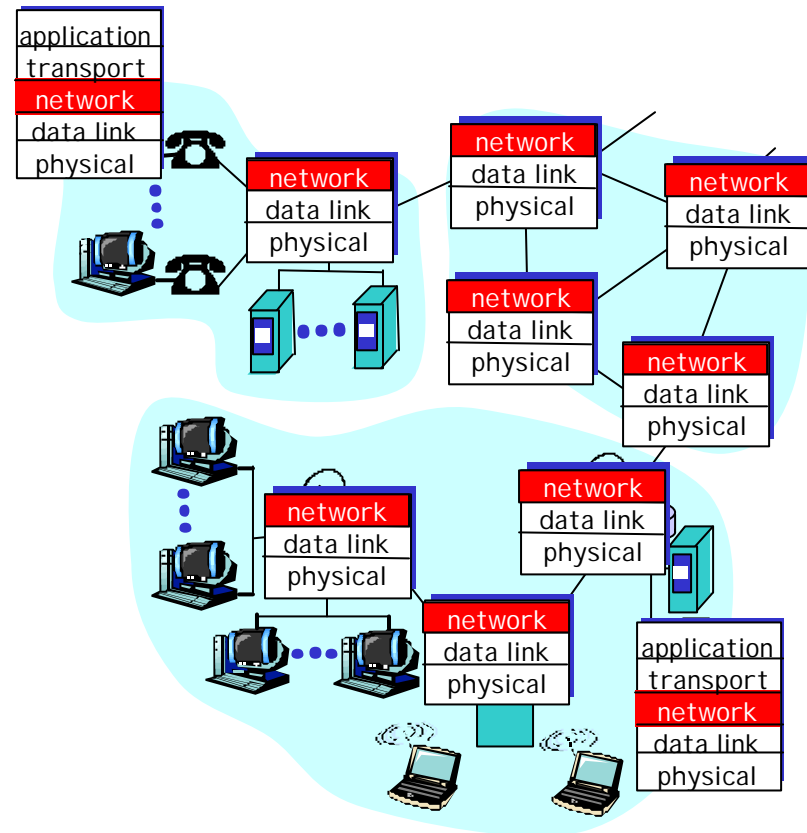
- ❑ network layer services
- ❑ routing principle: path selection
- ❑ hierarchical routing
- ❑ IP
- ❑ Internet routing protocols
  - reliable transfer
  - intra-domain
  - inter-domain
- ❑ what's inside a router?
- ❑ IPv6
- ❑ multicast routing

# Network layer functions

- ❑ transport packet from sending to receiving hosts
- ❑ network layer protocols in every host, router

three important functions:

- ❑ *path determination*: route taken by packets from source to dest. *Routing algorithms*
- ❑ *switching*: move packets from router's input to appropriate router output
- ❑ *call setup*: some network architectures require router call setup along path before data flows



# Network service model

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

service abstraction

- ☐ guaranteed bandwidth?
- ☐ preservation of inter-packet timing (no jitter)?
- ☐ loss-free delivery?
- ☐ in-order delivery?
- ☐ congestion feedback to sender?

The most important abstraction provided by network layer:

virtual circuit  
or  
datagram?

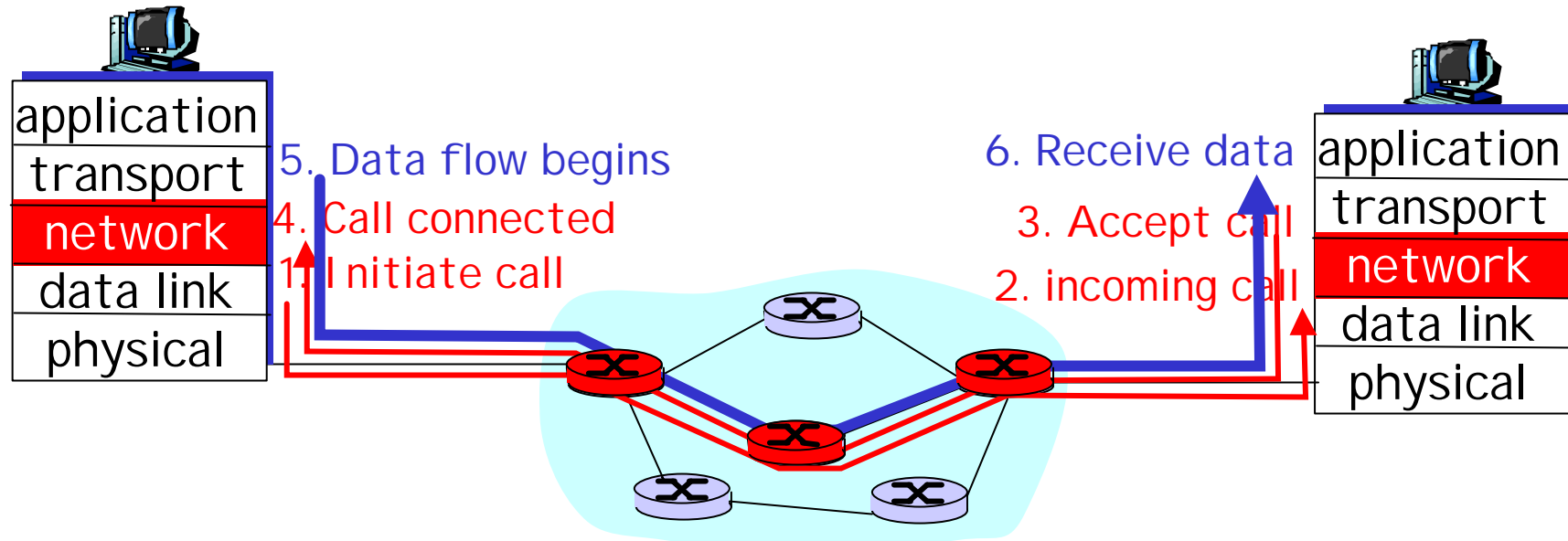
# Virtual circuits

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

- performance-wise
  - network actions along source-to-dest path
- 
- ❑ call setup, teardown for each call *before* data can flow
  - ❑ each packet carries VC identifier (not destination host OD)
  - ❑ every router on source-dest path s maintain “state” for each passing connection
    - transport-layer connection only involved two end systems
  - ❑ link, router resources (bandwidth, buffers) may be *allocated* to VC
    - to get circuit-like perf.

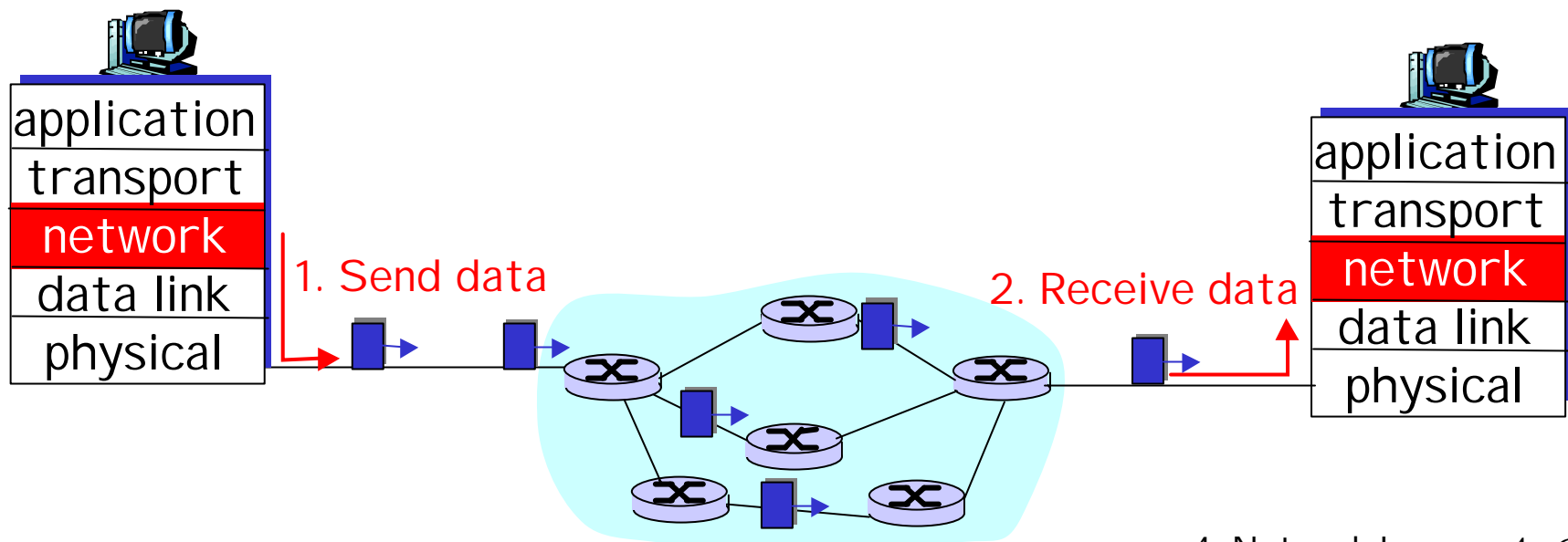
# Virtual circuits: signaling protocols

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



# Datagram networks: the Internet model

- ❑ no call setup at network layer
- ❑ routers: no state about end-to-end connections
  - no network-level concept of “connection”
- ❑ packets typically routed using destination host I D
  - packets between same source-dest pair may take different paths



# Network layer service models:

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

- ❑ Internet model being extended: Intserv, Diffserv
  - Chapter 6

# Datagram or VC network: why?

## Internet

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

## ATM

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



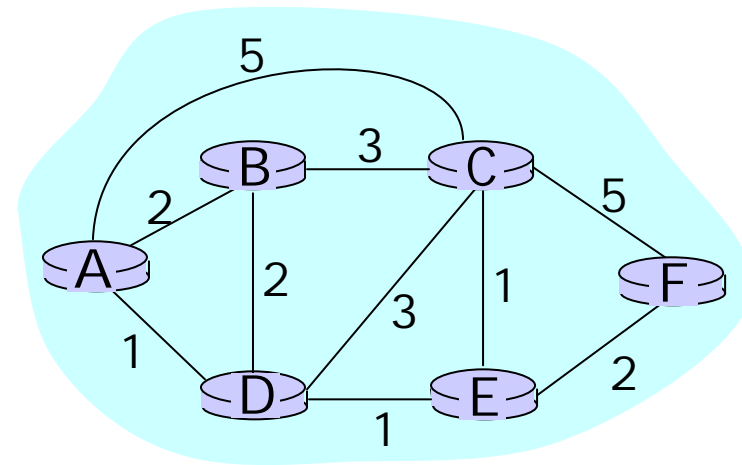
# Routing

## Routing protocol

**Goal:** determine “good” path (sequence of routers) thru network from source to dest.

Graph abstraction for routing algorithms:

- ❑ graph nodes are routers
- ❑ graph edges are physical links
  - link cost: delay, \$ cost, or congestion level



- ❑ “good” path:
  - typically means minimum cost path
  - other def’s possible

# Routing Algorithm classification

## 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

## 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'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 **routing table** for that node
- ❑ iterative: after  $k$  iterations, know least cost path to  $k$  dest.'s

## Notation:

- ❑  **$c(i,j)$** : link cost from node  $i$  to  $j$ . cost infinite 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  $v$ , that is next  $v$
- ❑  **$N$** : set of nodes whose least cost path definitively known

# Dijkstra's Algorithm

1 **Initialization:**

2  $N = \{A\}$

3 for all nodes  $v$

4 if  $v$  adjacent to  $A$

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

6 else  $D(v) = \text{infty}$

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

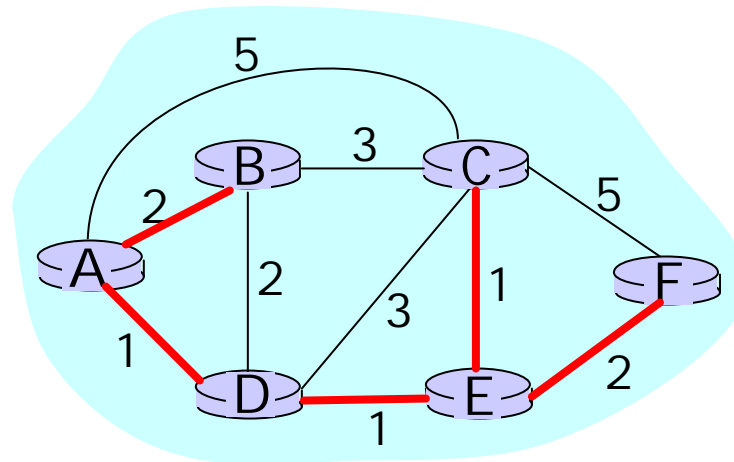
13 /\* new cost to  $v$  is either old cost to  $v$  or known

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

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

# Dijkstra's algorithm: example

Step	start N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
→ 0	A	2,A	5,A	1,A	infinity	infinity
→ 1	AD	2,A	4,D		2,D	infinity
→ 2	ADE	2,A	3,E			4,E
→ 3	ADEB		3,E			4,E
→ 4	ADEBC					4,E
5	ADEBCF					



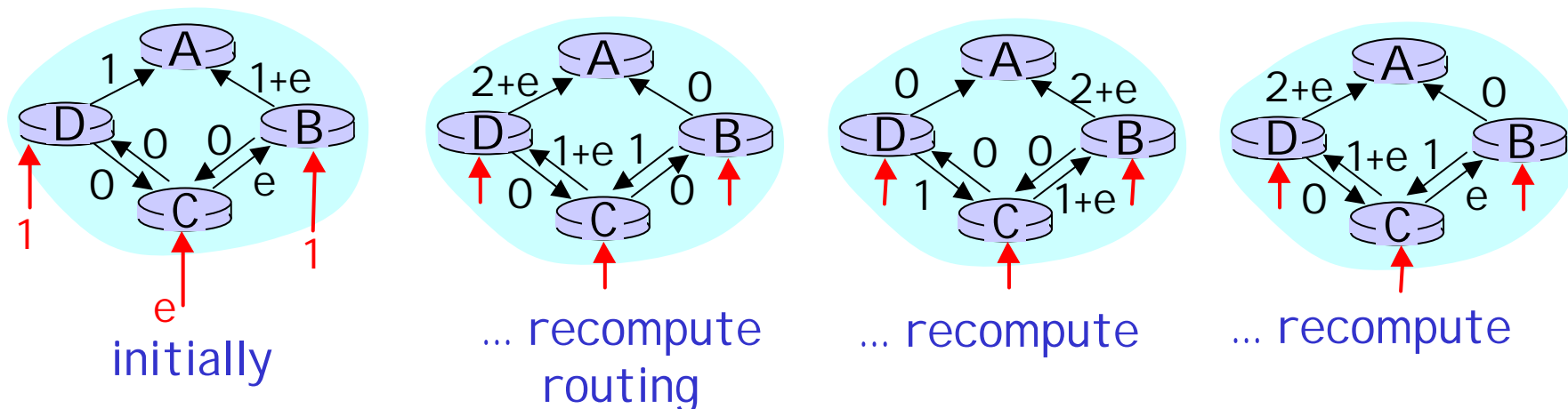
# Dijkstra's algorithm, discussion

**Algorithm complexity:**  $n$  nodes

- each iteration: need to check all nodes,  $w$ , not in  $N$
- $n^*(n+1)/2$  comparisons:  $O(n^2)$
- more efficient implementations possible:  $O(n \log n)$

**Oscillations possible:**

- e.g., link cost = amount of carried traffic



# Distance Vector Routing Algorithm

## iterative:

- ❑ continues until no nodes exchange info.
- ❑ *self-terminating*: no "signal" to stop

## asynchronous:

- ❑ nodes need *not* exchange info/iterate in lock step!

## distributed:

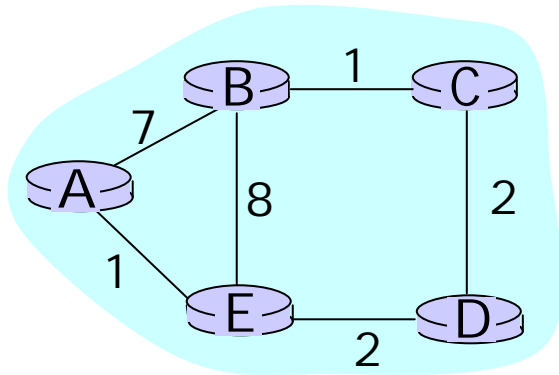
- ❑ each node communicates *only* with directly-attached neighbors

## Distance Table data structure

- ❑ each node has its own
- ❑ row for each possible destination
- ❑ column for each directly-attached neighbor to node
- ❑ example: in node X, for dest. Y via neighbor Z:

$$\begin{aligned} D^X(Y,Z) &= \text{distance from X to Y, via Z as next hop} \\ &= c(X,Z) + \min_w \{D^Z(Y,w)\} \end{aligned}$$

# Distance Table: example



$$D^E(C,D) = c(E,D) + \min_w \{D^D(C,w)\} \\ = 2+2 = 4$$

$$D^E(A,D) = c(E,D) + \min_w \{D^D(A,w)\} \\ = 2+3 = 5 \text{ loop!}$$

$$D^E(A,B) = c(E,B) + \min_w \{D^B(A,w)\} \\ = 8+6 = 14 \text{ loop!}$$

cost to destination via

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destination



# Distance table gives routing table

		cost to destination via		
destination	$D^E()$	A	B	D
	A	1	14	5
	B	7	8	5
	C	6	9	4
	D	4	11	2

		Outgoing link to use, cost
destination	A	A,1
	B	D,5
	C	D,4
	D	D,4

Distance table  Routing table

# Distance Vector Routing: overview

## Iterative, asynchronous:

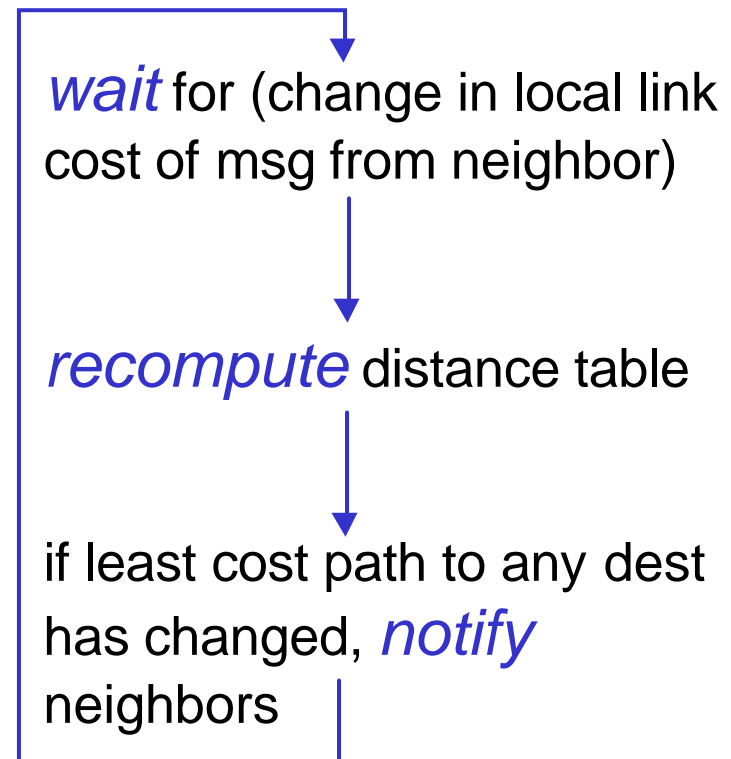
each local iteration caused by:

- ❑ local link cost change
- ❑ message from neighbor: its least cost path change from neighbor

## Distributed:

- ❑ each node notifies neighbors *only* when its least cost path to any destination changes
  - neighbors then notify their neighbors if necessary

## Each node:



# Distance Vector Algorithm:

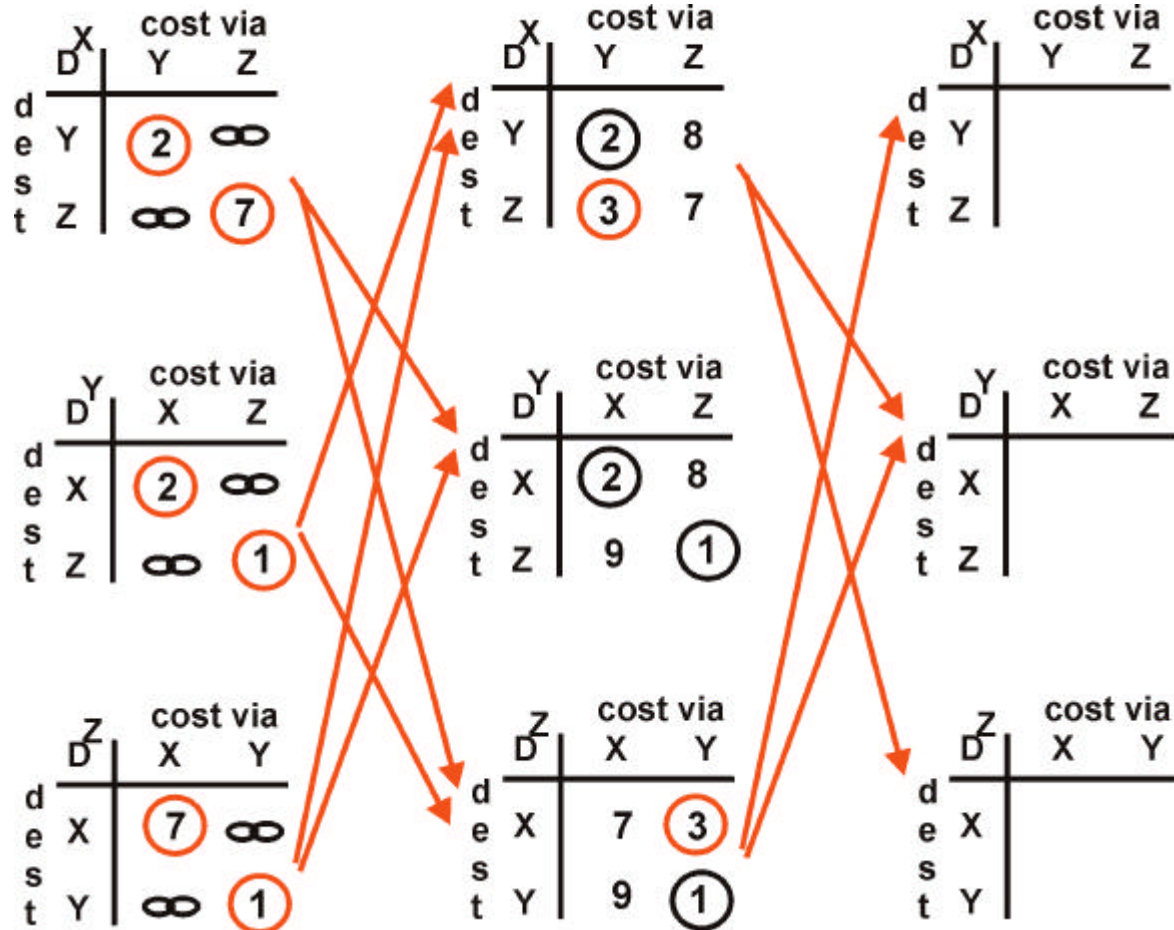
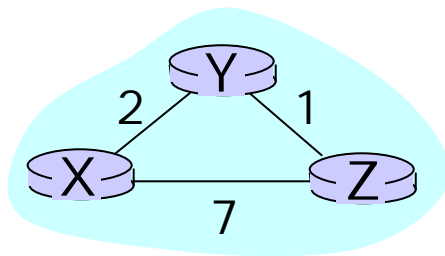
At all nodes, X:

- 1 Initialization:
- 2 for all adjacent nodes v:
- 3      $D^X(*,v) = \text{infty}$      /\* the \* operator means "for all rows" \*/
- 4      $D^X(v,v) = c(X,v)$
- 5 for all destinations, y
- 6     send  $\min_w D^X(y,w)$  to each neighbor /\* w over all X's neighbors \*/

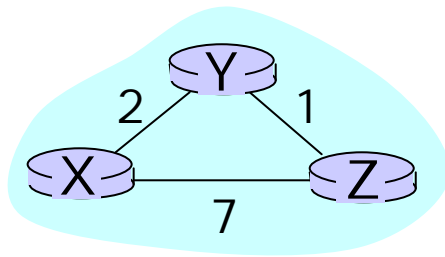
## Distance Vector Algorithm (cont.):

```
→ 8 loop
9   wait (until I see a link cost change to neighbor V
10      or until I receive update from neighbor V)
11
12   if (c(X,V) changes by d)
13     /* change cost to all dest's via neighbor v by d */
14     /* note: d could be positive or negative */
15     for all destinations y:  $D^X(y,V) = D^X(y,V) + d$ 
16
17   else if (update received from V wrt destination Y)
18     /* shortest path from V to some Y has changed */
19     /* V has sent a new value for its  $\min_w DV(Y,w)$  */
20     /* call this received new value is "newval" */
21     for the single destination y:  $D^X(Y,V) = c(X,V) + \text{newval}$ 
22
23   if we have a new  $\min_w D^X(Y,w)$  for any destination Y
24     send new value of  $\min_w D^X(Y,w)$  to all neighbors
25
26 forever
```

# Distance Vector Algorithm: example



# Distance Vector Algorithm: example



		cost via	
		Y	Z
destination	X		
	D		
source	Y	2	$\infty$
	Z	$\infty$	7

		cost via	
		X	Z
destination	Y		
	D		
source	X	2	$\infty$
	Z	$\infty$	1

		cost via	
		X	Y
destination	Z		
	D		
source	X	7	$\infty$
	Y	$\infty$	1

		cost via	
		Y	Z
destination	X		
	D		
source	Y	2	8
	Z	3	7

$$D^X(Y,Z) = c(X,Z) + \min_w \{D^Z(Y,w)\}$$

$$= 7 + 1 = 8$$

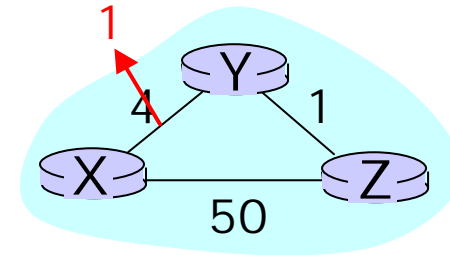
$$D^X(Z,Y) = c(X,Y) + \min_w \{D^Y(Z,w)\}$$

$$= 2 + 1 = 3$$

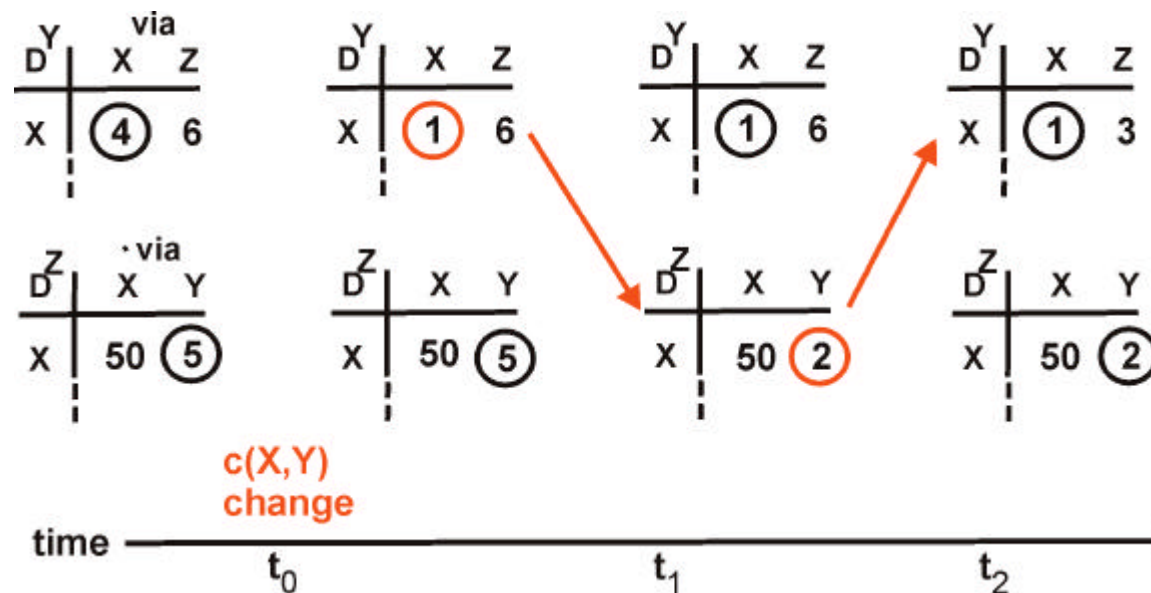
# Distance Vector: link cost changes

## Link cost changes:

- ❑ node detects local link cost change
- ❑ updates distance table (line 15)
- ❑ if cost change in least cost path, notify neighbors (lines 23,24)



“good  
news  
travels  
fast”

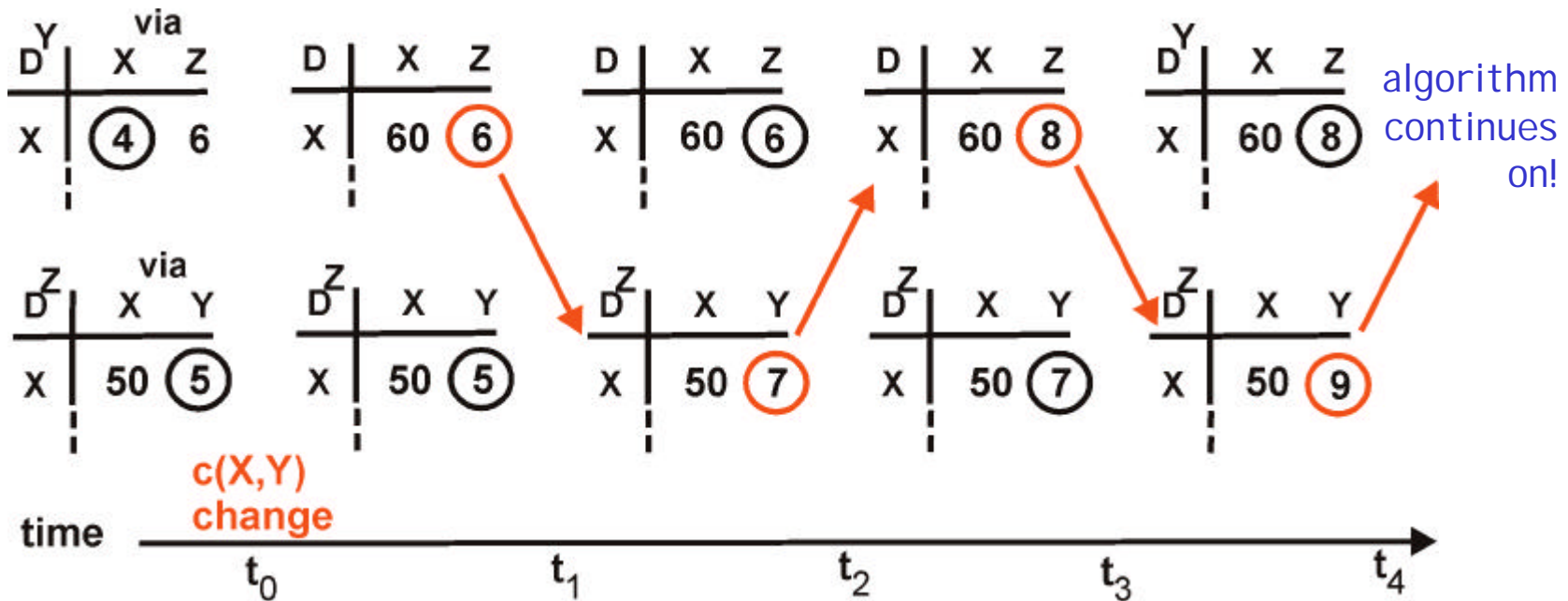
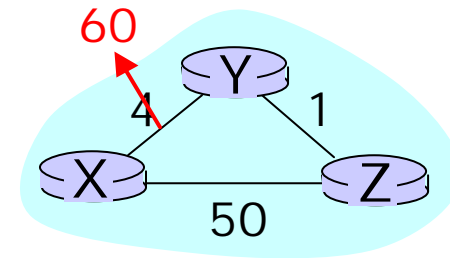


algorithm  
terminates

# Distance Vector: link cost changes

## Link cost changes:

- good news travels fast
- bad news travels slow - "count to infinity" problem!

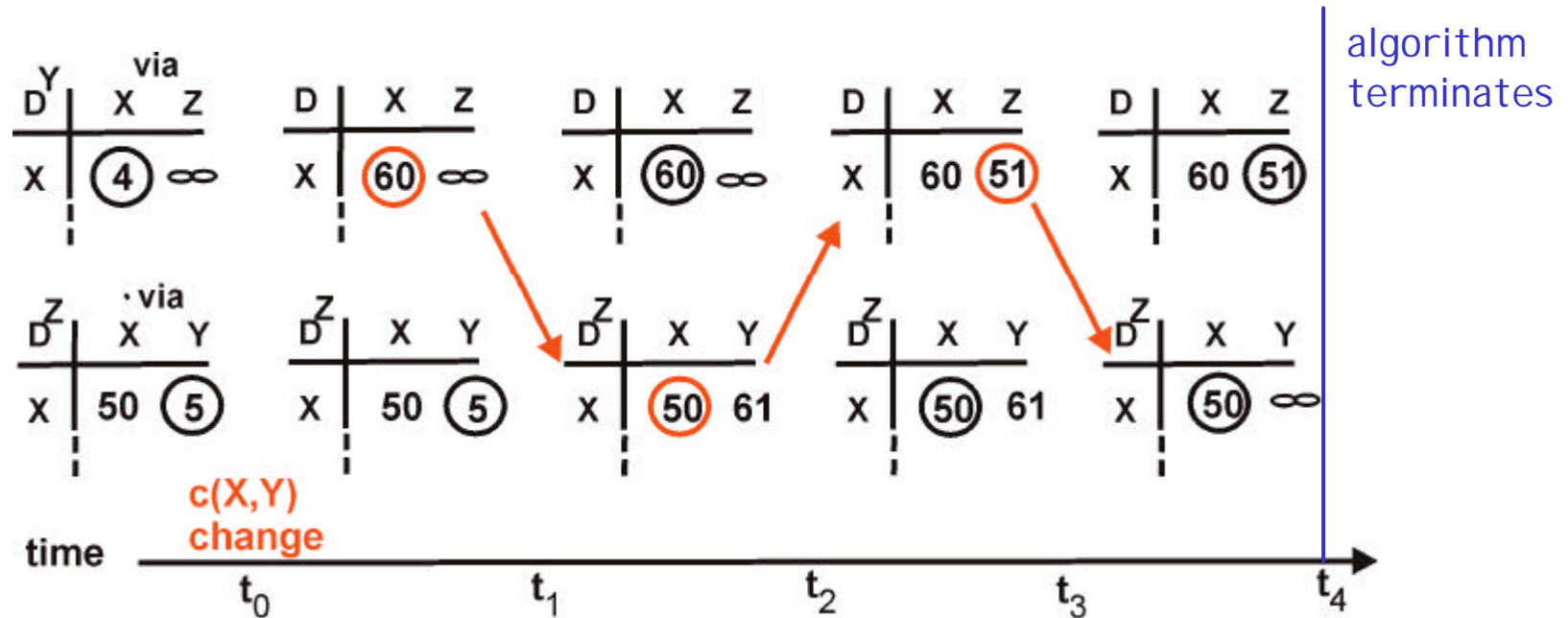
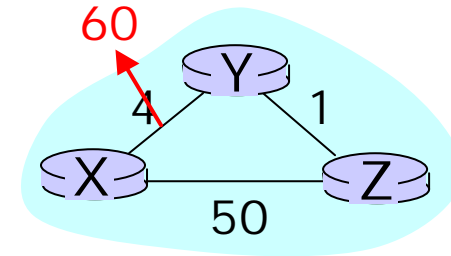




# Distance Vector: poisoned reverse

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 each
- ❑ 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

# Hierarchical Routing

Our routing study thus far - idealization

- ❑ all routers identical
- ❑ network “flat”

... *not* true in practice

**scale:** with 50 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

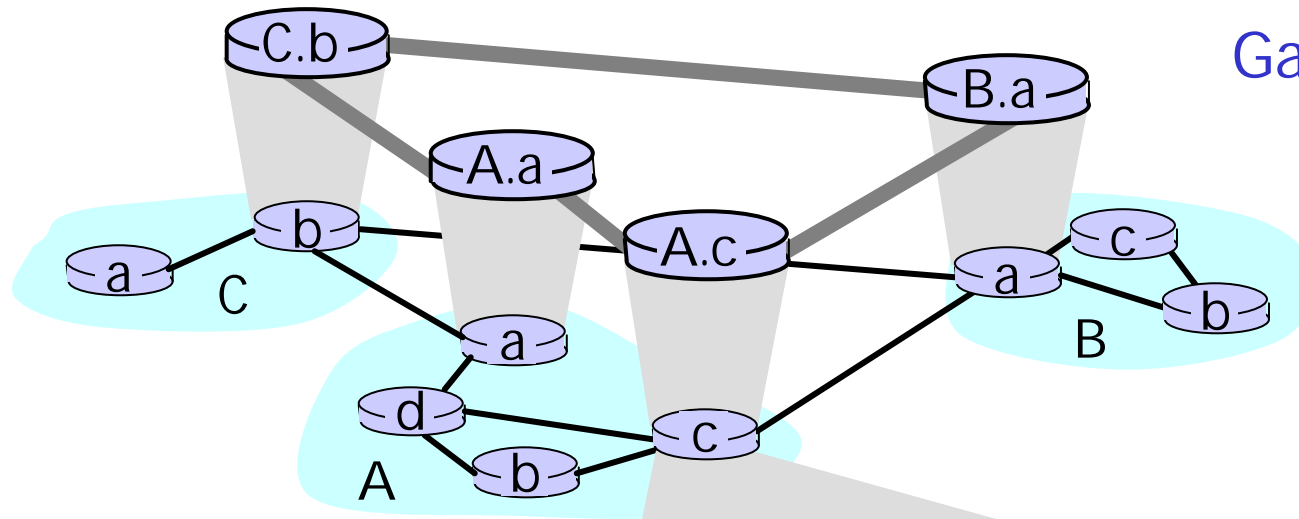
# Hierarchical Routing

- ❑ 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 routers

- ❑ special routers in AS
- ❑ run intra-AS routing protocol with all other routers in AS
- ❑ *also* responsible for routing to destinations outside AS
  - run **inter-AS routing** protocol with other gateway routers

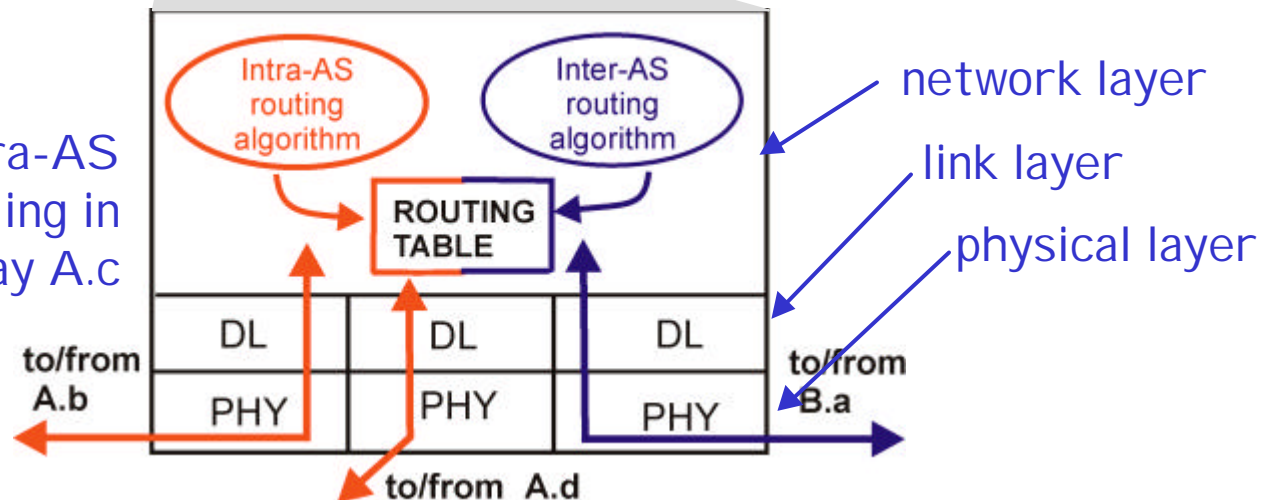
# Intra-AS and Inter-AS routing



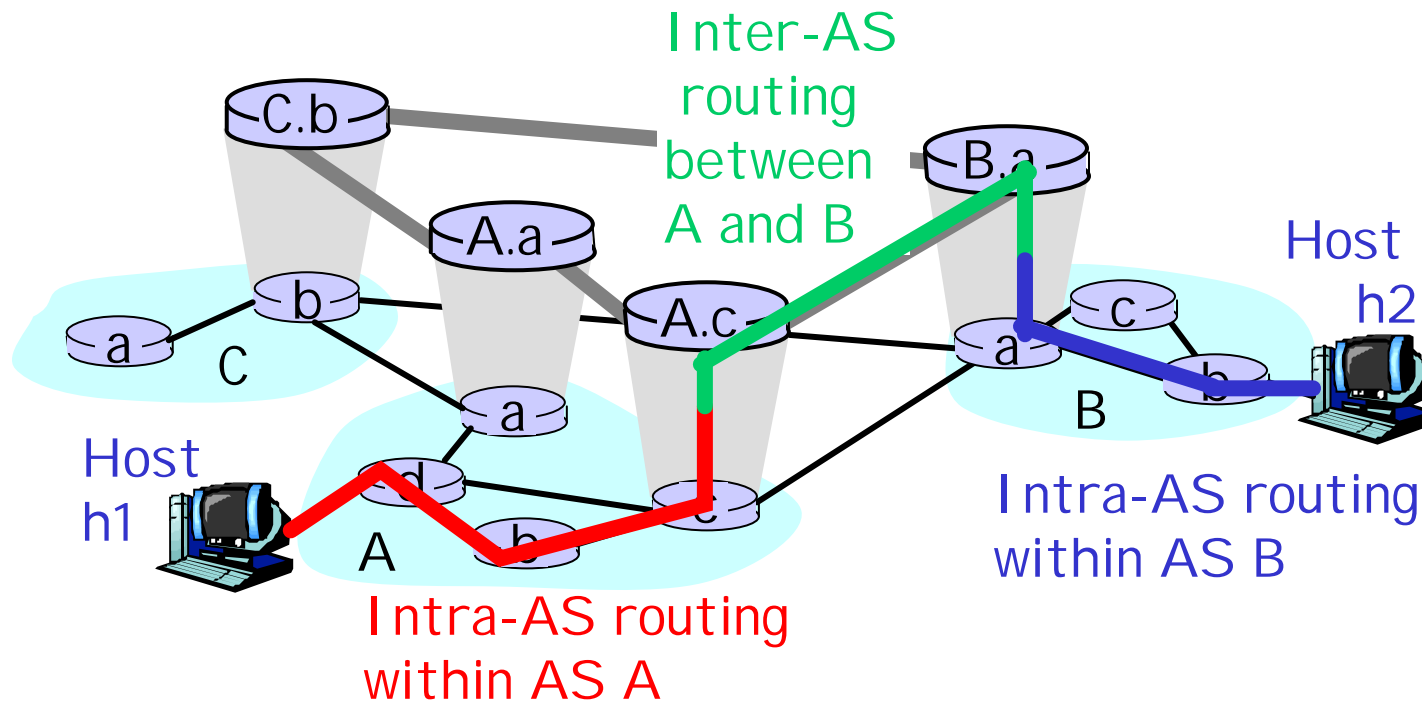
## Gateways:

- perform inter-AS routing amongst themselves
- perform intra-AS routing with other routers in their AS

inter-AS, intra-AS  
routing in  
gateway A.c



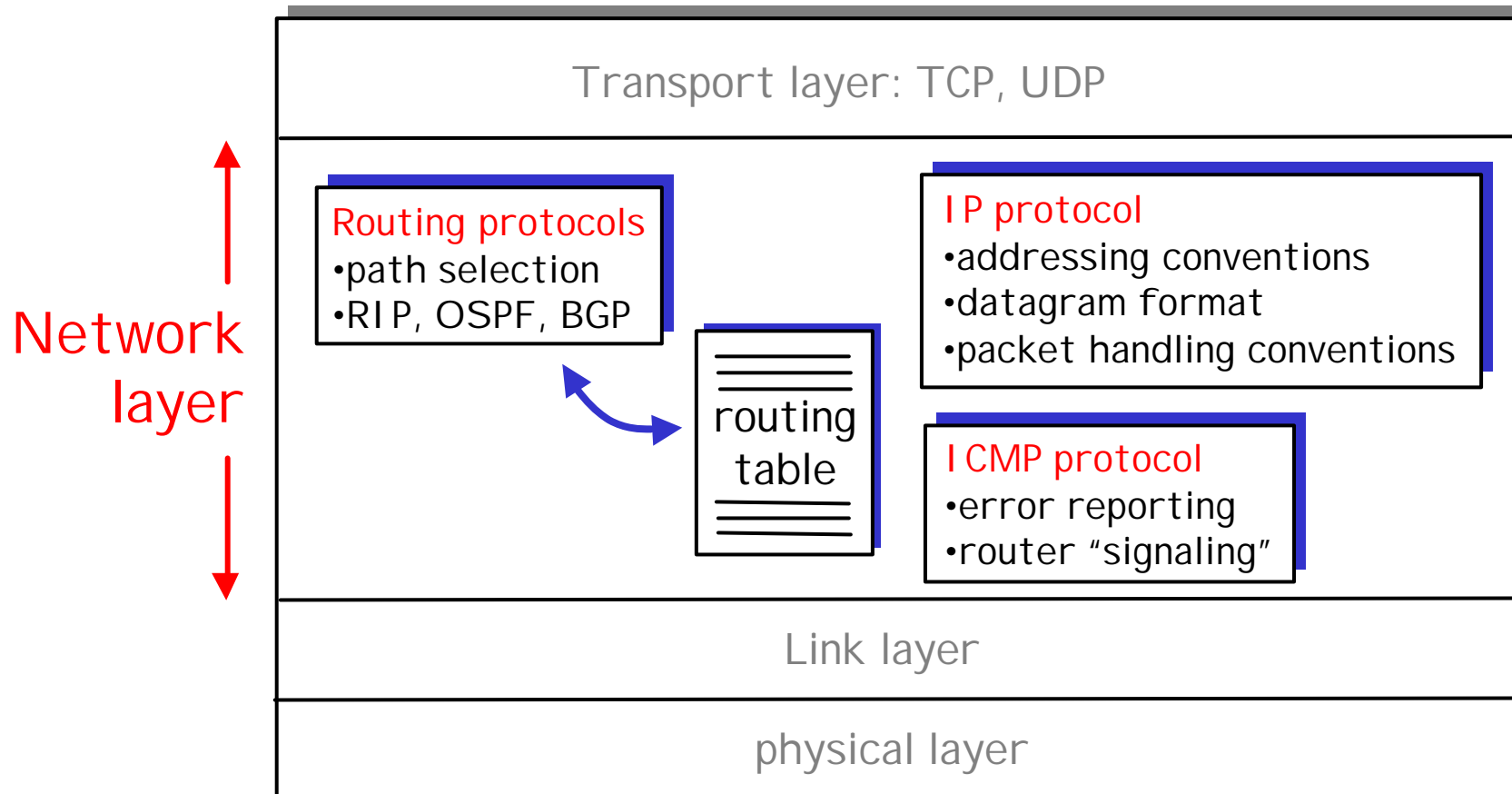
# Intra-AS and Inter-AS routing



- We'll examine specific inter-AS and intra-AS Internet routing protocols shortly

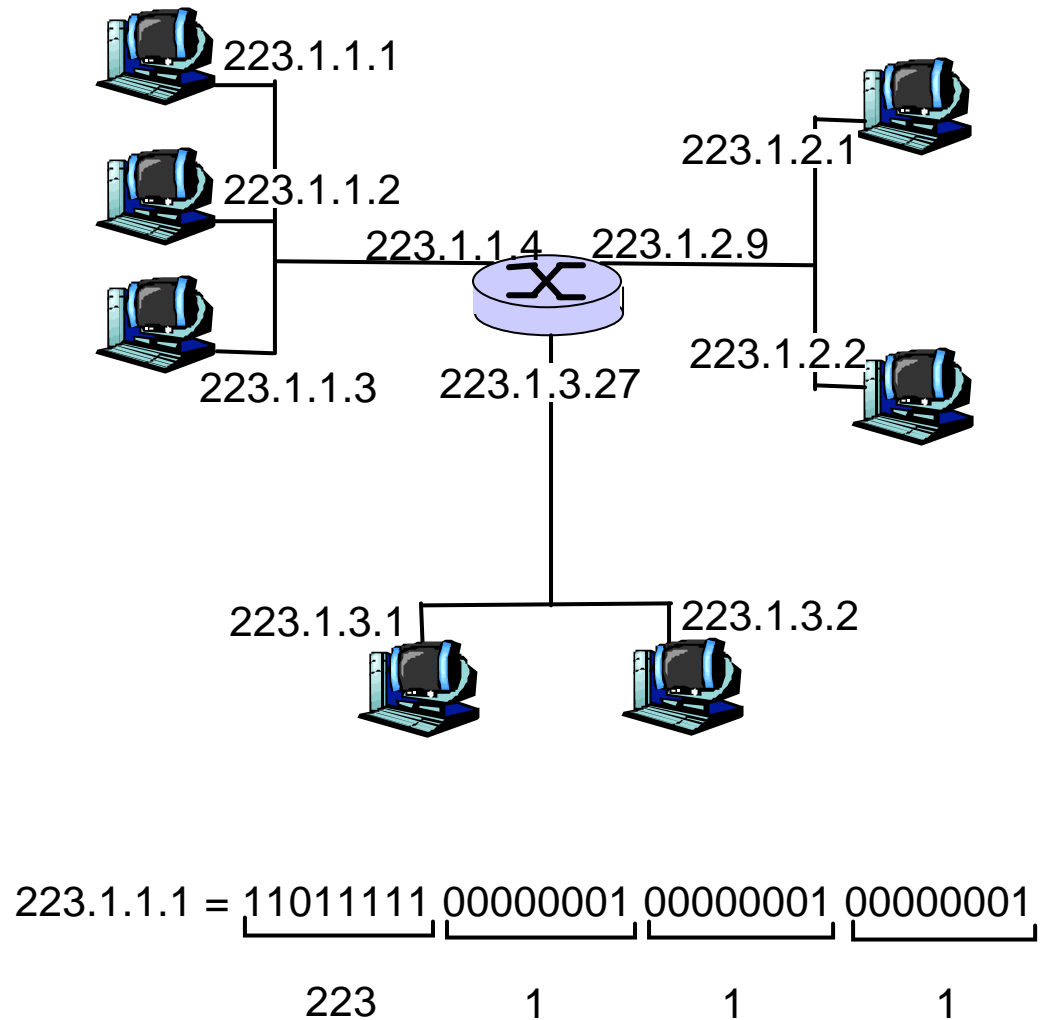
# The Internet Network layer

Host, router network layer functions:



# 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 may have multiple interfaces
  - IP addresses associated with interface, not host, router





# IP Addressing

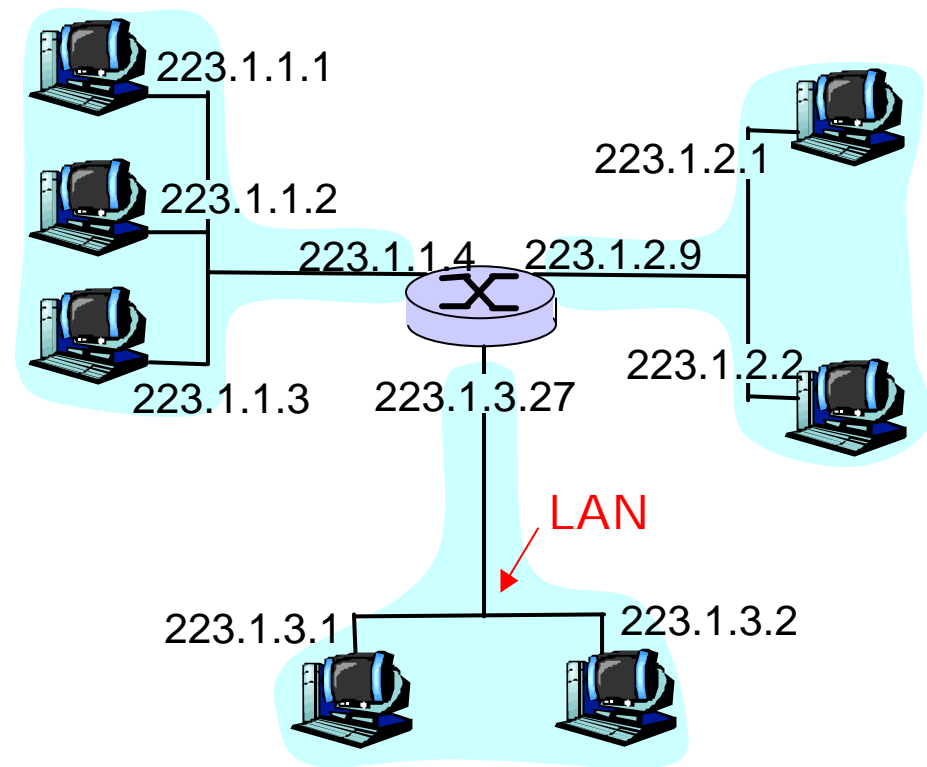
## ❑ IP address:

- network part (high order bits)
- host part (low order bits)

## ❑ *What's a network ?*

(from IP address perspective)

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



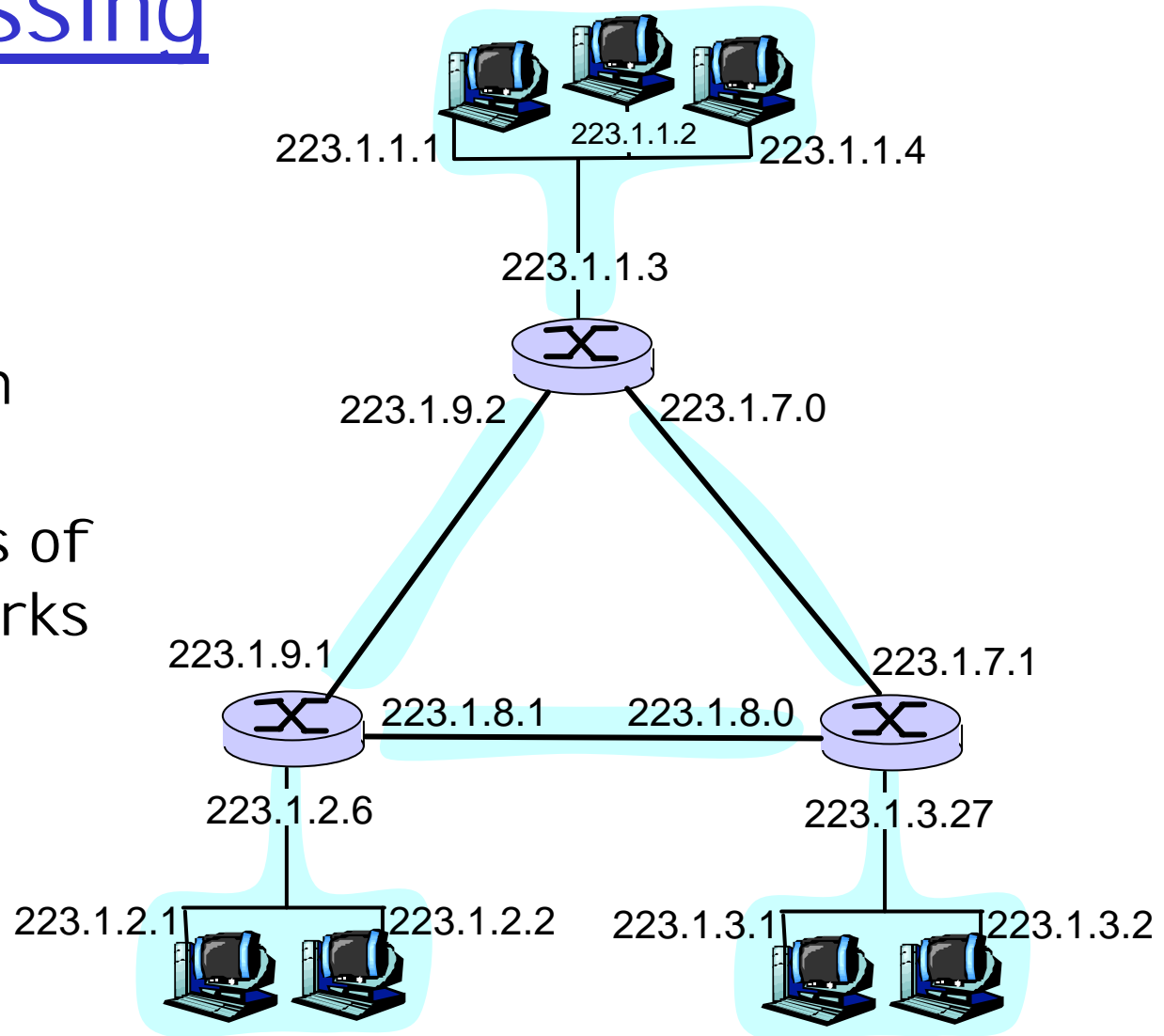
network consisting of 3 IP networks  
(for IP addresses starting with 223,  
first 24 bits are network address)

# IP Addressing

How to find the networks?

- ❑ Detach each interface from router, host
- ❑ create "islands of isolated networks"

Interconnected system consisting of six networks

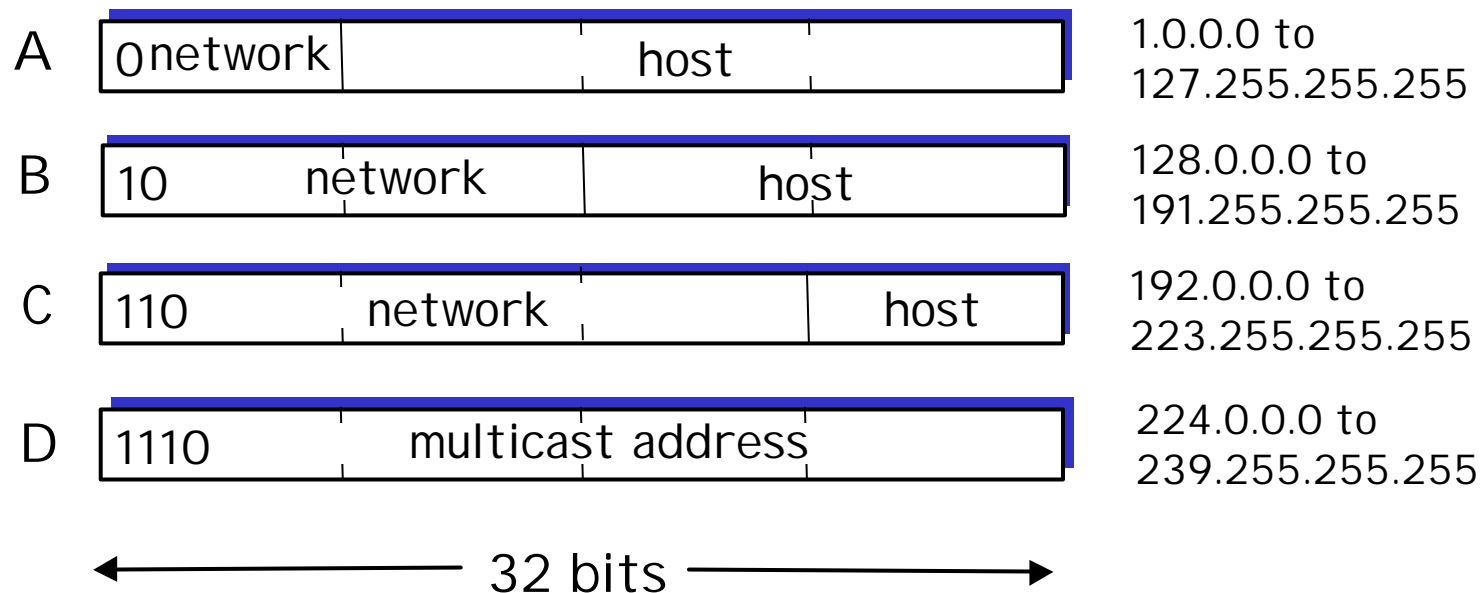


# IP Addresses

given notion of “network”, let’s re-examine IP addresses:

“class-full” addressing:

class



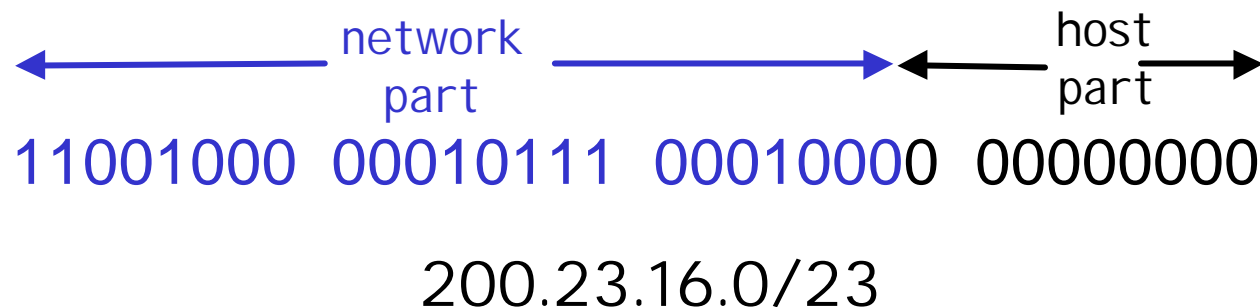
# I P addressing: C I D R

## ❑ classful addressing:

- inefficient use of address space, address space exhaustion
- e.g., class B net allocated enough addresses for 65K hosts, even if only 2K hosts in that network

## ❑ C I D R: C lassless I nterD omain R outing

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



# I P addresses: how to get one?

Hosts (host portion):

- ❑ hard-coded by system admin in a file
- ❑ **DHCP: Dynamic Host Configuration Protocol:**  
dynamically get address: "plug-and-play"
  - host broadcasts "**DHCP discover**" msg
  - DHCP server responds with "**DHCP offer**" msg
  - host requests I P address: "**DHCP request**" msg
  - DHCP server sends address: "**DHCP ack**" msg

# IP addresses: how to get one?

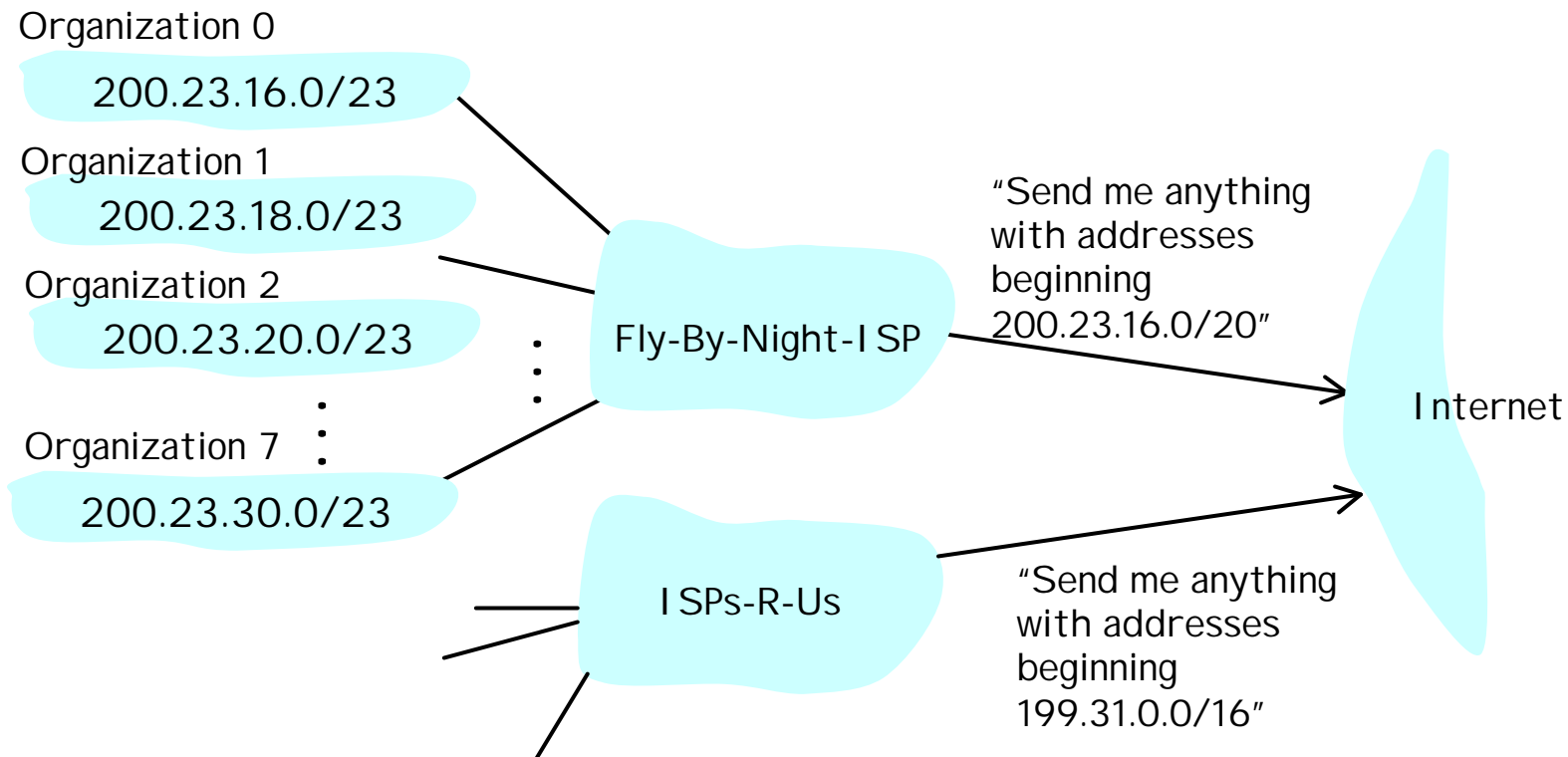
Network (network portion):

❑ get allocated portion of ISP's address space:

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

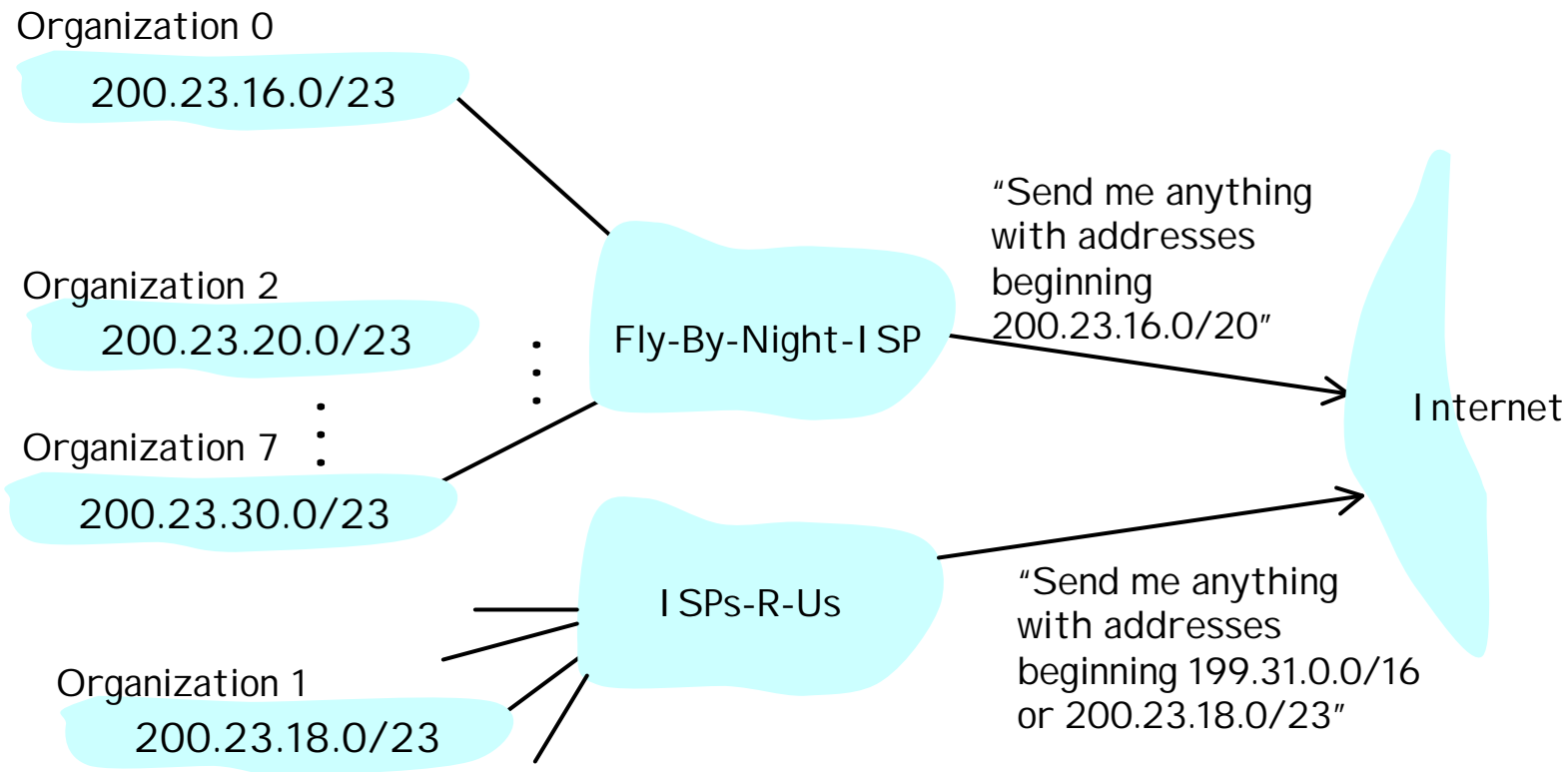
# Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:



# Hierarchical addressing: more specific routes

I SPs-R-Us has a more specific route to Organization 1





## I P addressing: the last word...

Q: How does an I SP get block of addresses?

A: **ICANN**: **I**nternet **C**orporation for **A**ssigned  
**N**ames and **N**umbers

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

# Getting a datagram from source to dest.

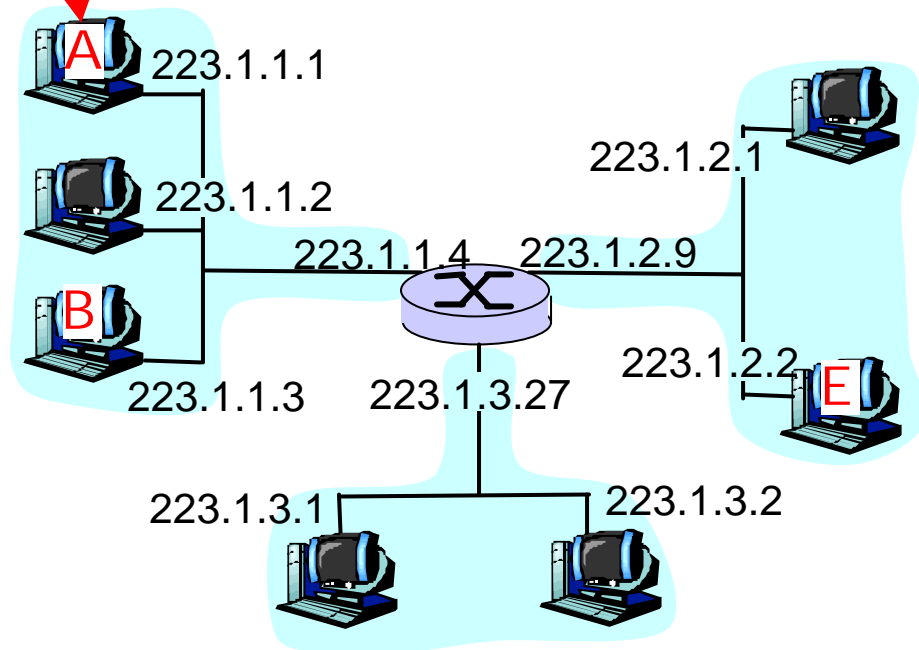
IP datagram:

misc	source	dest	
fields	IP addr	IP addr	data

- ❑ datagram remains unchanged, as it travels source to destination
- ❑ addr fields of interest here

routing table in A

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2

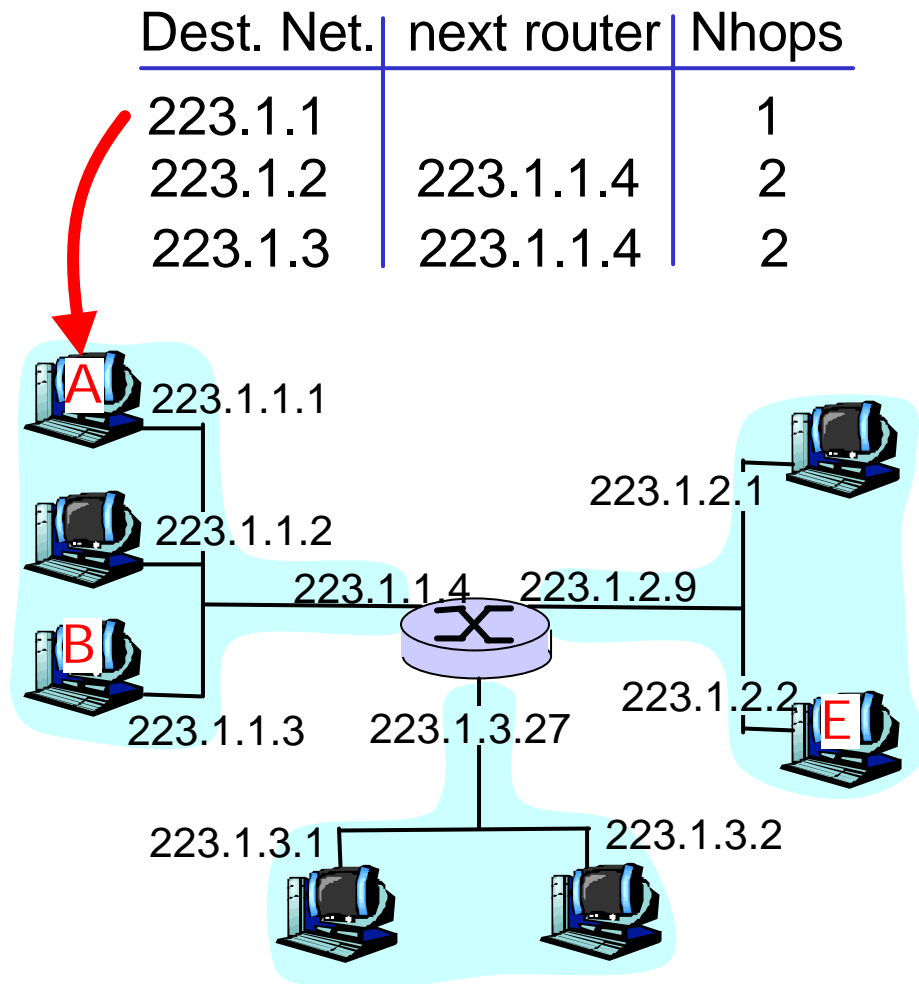


# Getting a datagram from source to dest.

misc fields	223.1.1.1	223.1.1.3	data
-------------	-----------	-----------	------

Starting at A, given IP datagram addressed to B:

- ❑ look up net. address of B
- ❑ find B is on same net. as A
- ❑ link layer will send datagram directly to B inside link-layer frame
  - B and A are directly connected

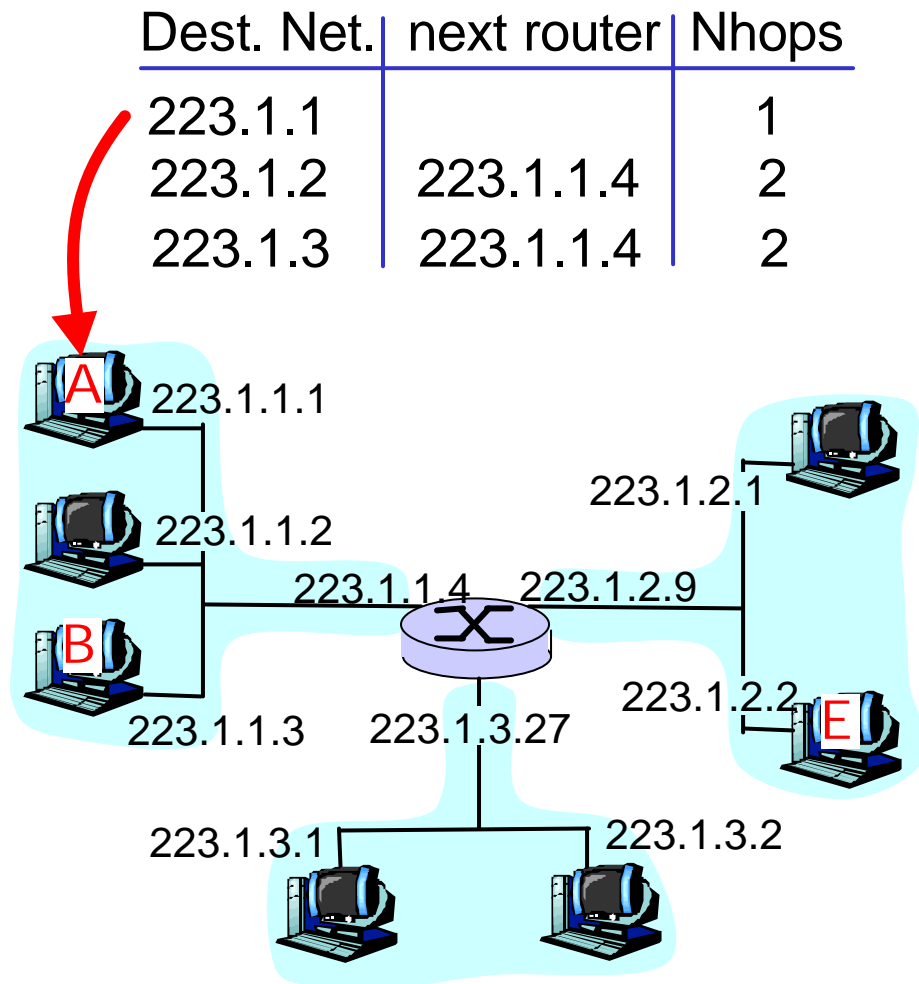


# Getting a datagram from source to dest.

misc fields	223.1.1.1	223.1.2.3	data
-------------	-----------	-----------	------

Starting at A, dest. E:

- ❑ look up network address of E
- ❑ E on *different* network
  - A, E not directly attached
- ❑ routing table: next hop router to E is 223.1.1.4
- ❑ link layer sends datagram to router 223.1.1.4 inside link-layer frame
- ❑ datagram arrives at 223.1.1.4
- ❑ continued.....



# Getting a datagram from source to dest.

misc fields	223.1.1.1	223.1.2.3	data
-------------	-----------	-----------	------

Arriving at 223.1.4,  
destined for 223.1.2.2

- ❑ look up network address of E
- ❑ E on *same* network as router's interface 223.1.2.9
  - router, E directly attached
- ❑ link layer sends datagram to 223.1.2.2 inside link-layer frame via interface 223.1.2.9
- ❑ datagram arrives at 223.1.2.2!!! (hooray!)

Dest. network	next router	Nhops	interface
223.1.1	-	1	223.1.1.4
223.1.2	-	1	223.1.2.9
223.1.3	-	1	223.1.3.27

