

# Advance Computer Networks

#2 Routing Algorithm 😊

Semester Ganjil 2012

PTIIK – Universitas Brawijaya

# Today's Lecture

- This Course
  - Goals :
    - Understand Routing Algorithm
    - Distance Vector & Link State Routing
  - Outline/Schedule
  - Grading Policy
- Dynamic Routing ( Link State vs Distance Vector )
- Dynamic Routing vs Static Routing ?

# References

- Kurose & Ross, “Computer Networking : Top down Approach”, 6<sup>th</sup> Ed., Pearson/Addison, 2012.
- Chapter 4
  - Network Layer

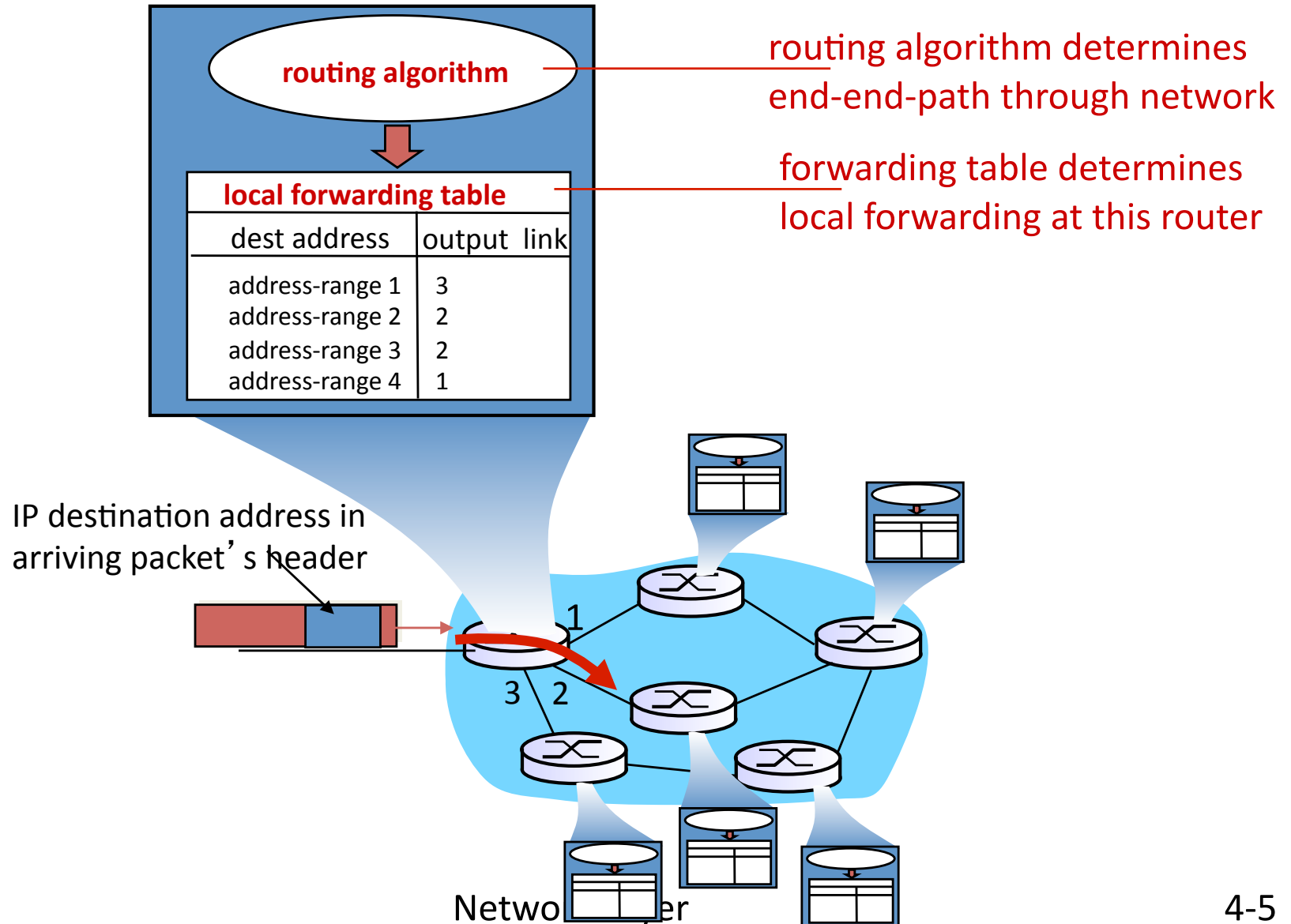
# #1 – Routing Algorithm

Routing & Forwarding

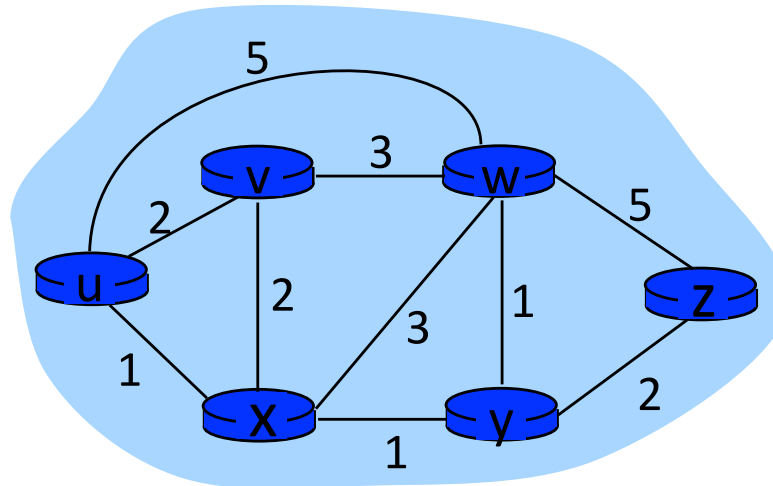
Graph Theory

Routing Algorithm

# Interplay between routing, forwarding



# Graph abstraction



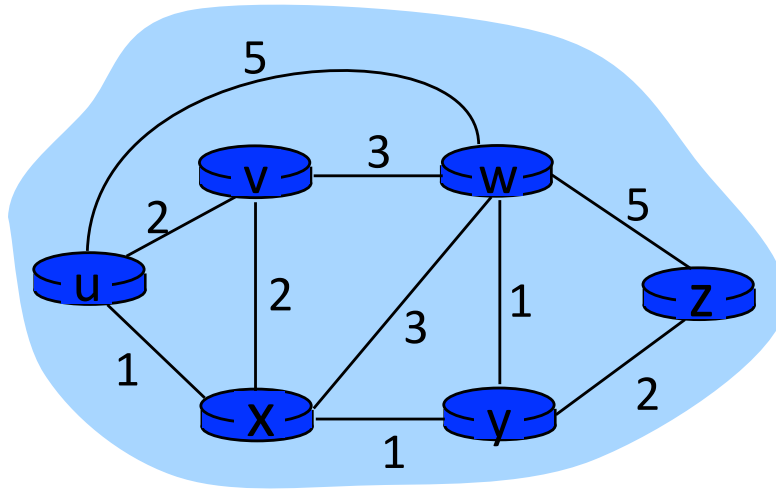
graph:  $G = (N,E)$

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

$E$  = set of links = { (u,v), (u,x), (u,w), (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



# 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  $v$  \*/

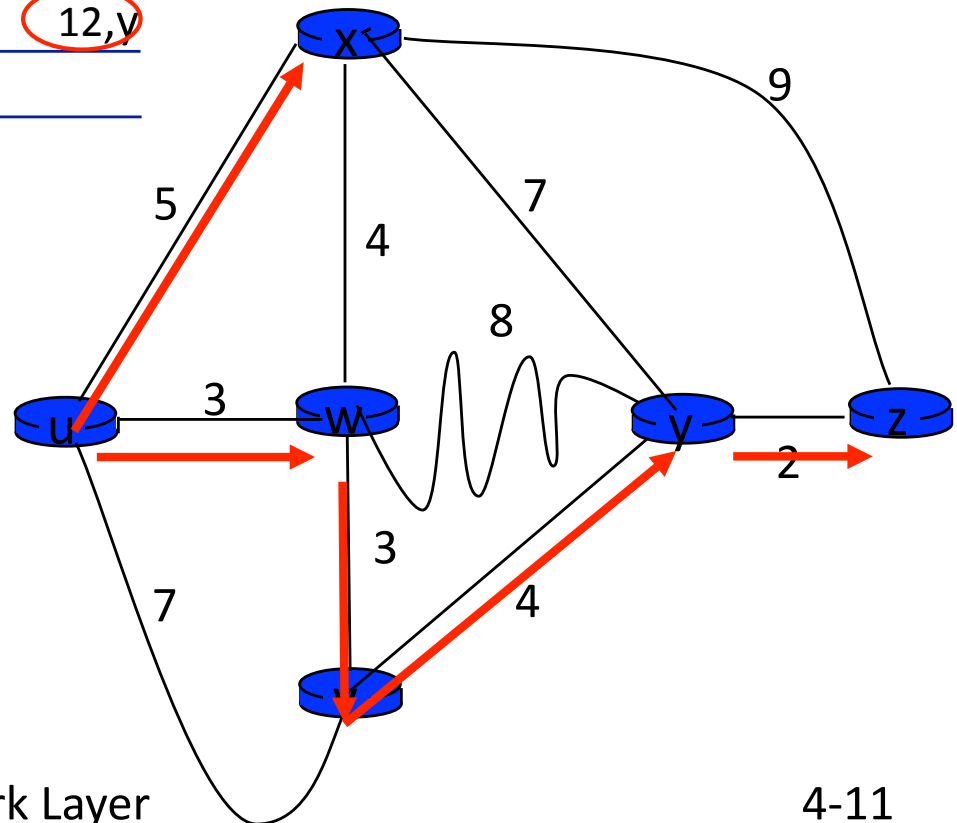
15 **until all nodes in  $N'$**

# 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	$\infty$	$\infty$
1	uw	6,w		5,u	11,w	$\infty$
2	uwx	6,w			11,w	14,x
3	uwxv				10,y	14,x
4	uwxvy				12,y	
5	uwxvyz					

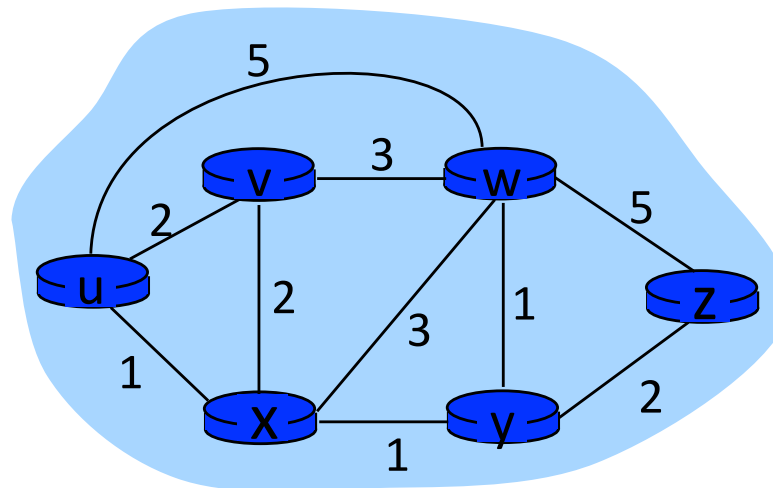
## notes:

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



# Dijkstra's algorithm: another example

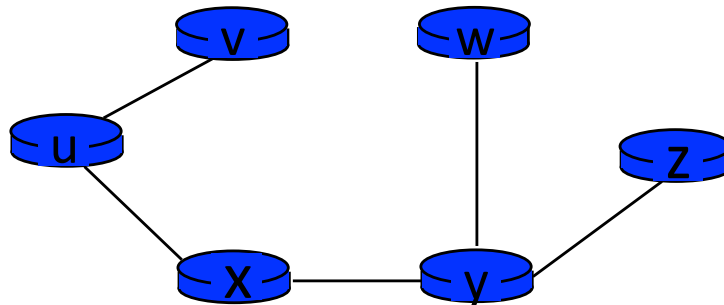
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	$\infty$	$\infty$
1	ux	2,u	4,x		2,x	$\infty$
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Network Layer

# 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)

Network Layer

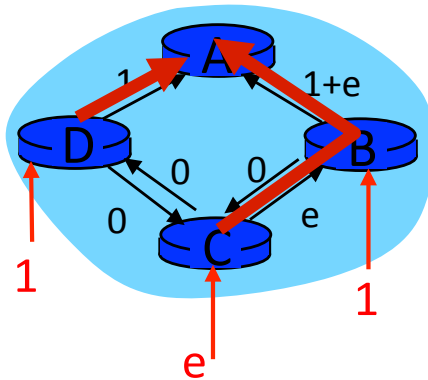
# Dijkstra's algorithm, discussion

*algorithm complexity:* n nodes

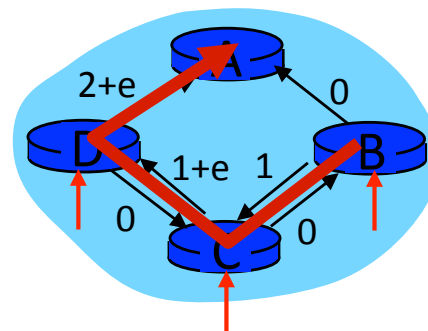
- each iteration: need to check all nodes, w, not in N
- $n(n+1)/2$  comparisons:  $O(n^2)$
- more efficient implementations possible:  $O(n \log n)$

*oscillations possible:*

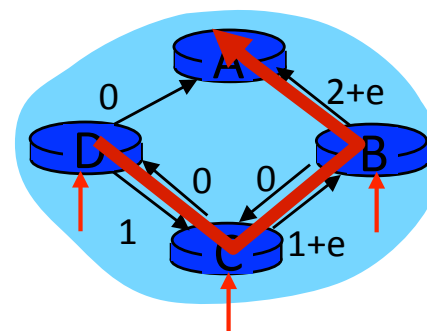
- e.g., support link cost equals amount of carried traffic:



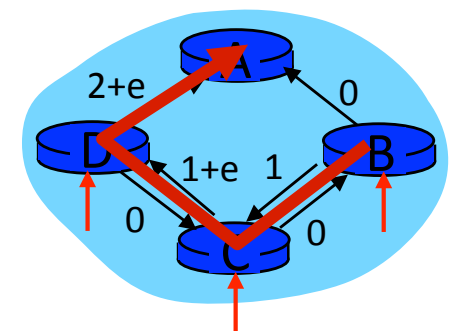
initially



given these costs,  
find new routing....  
resulting in new costs



given these costs,  
find new routing....  
resulting in new costs



given these costs,  
find new routing....  
resulting in new costs

Network Layer

# Distance vector algorithm

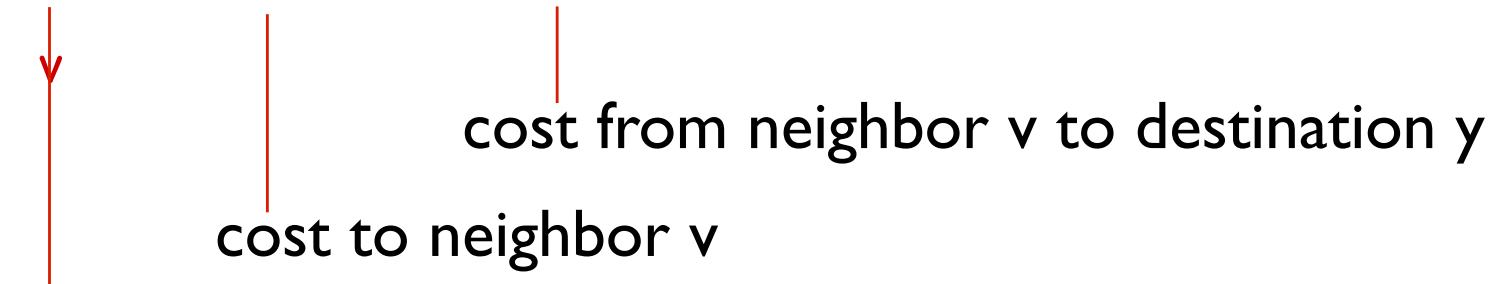
*Bellman-Ford equation (dynamic programming)*

let

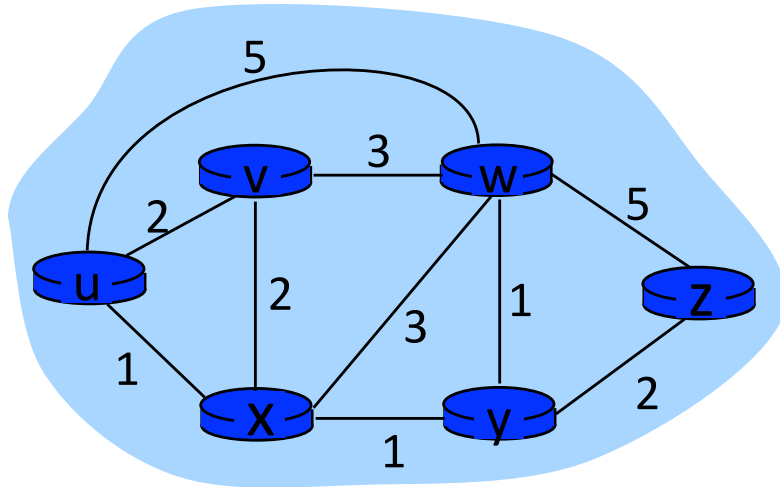
$d_x(y) :=$  cost of least-cost path from  $x$  to  $y$

then

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

  
 $\min$  taken over all neighbors  $v$  of  $x$   
cost to neighbor  $v$   
cost from neighbor  $v$  to destination  $y$

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

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

# Distance vector algorithm

## *iterative, asynchronous:*

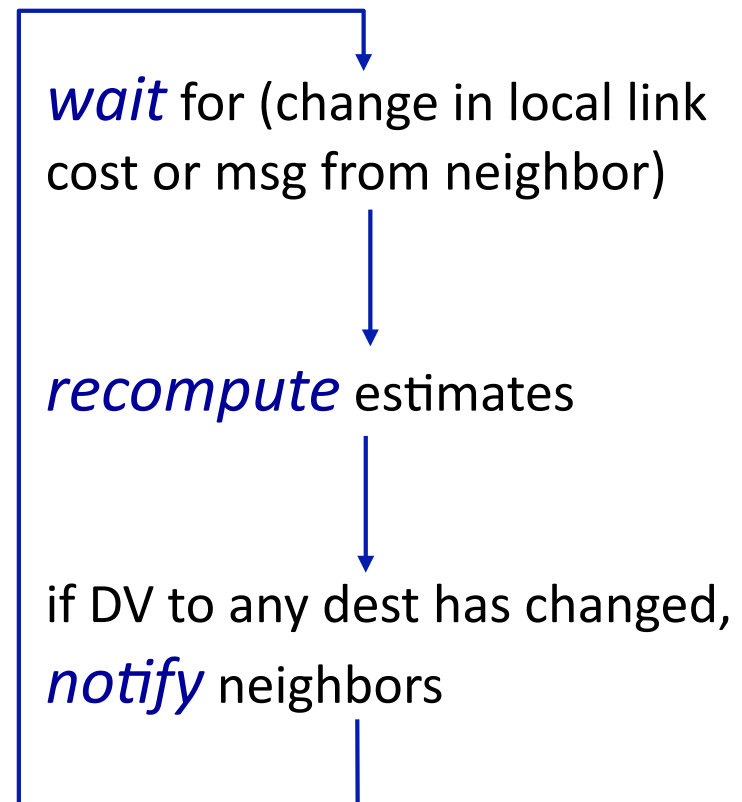
each local iteration  
caused by:

- 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	$\infty$	$\infty$	$\infty$
	z	$\infty$	$\infty$	$\infty$

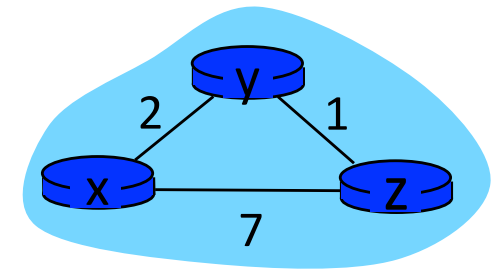
		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	7	1	0

**node y table**

		cost to		
		x	y	z
from	x	$\infty$	$\infty$	$\infty$
	y	2	0	1
	z	$\infty$	$\infty$	$\infty$

**node z table**

		cost to		
		x	y	z
from	x	$\infty$	$\infty$	$\infty$
	y	$\infty$	$\infty$	$\infty$
	z	7	1	0



..... → time

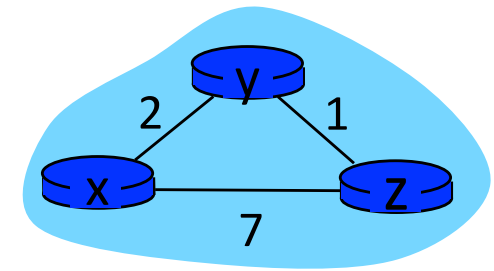
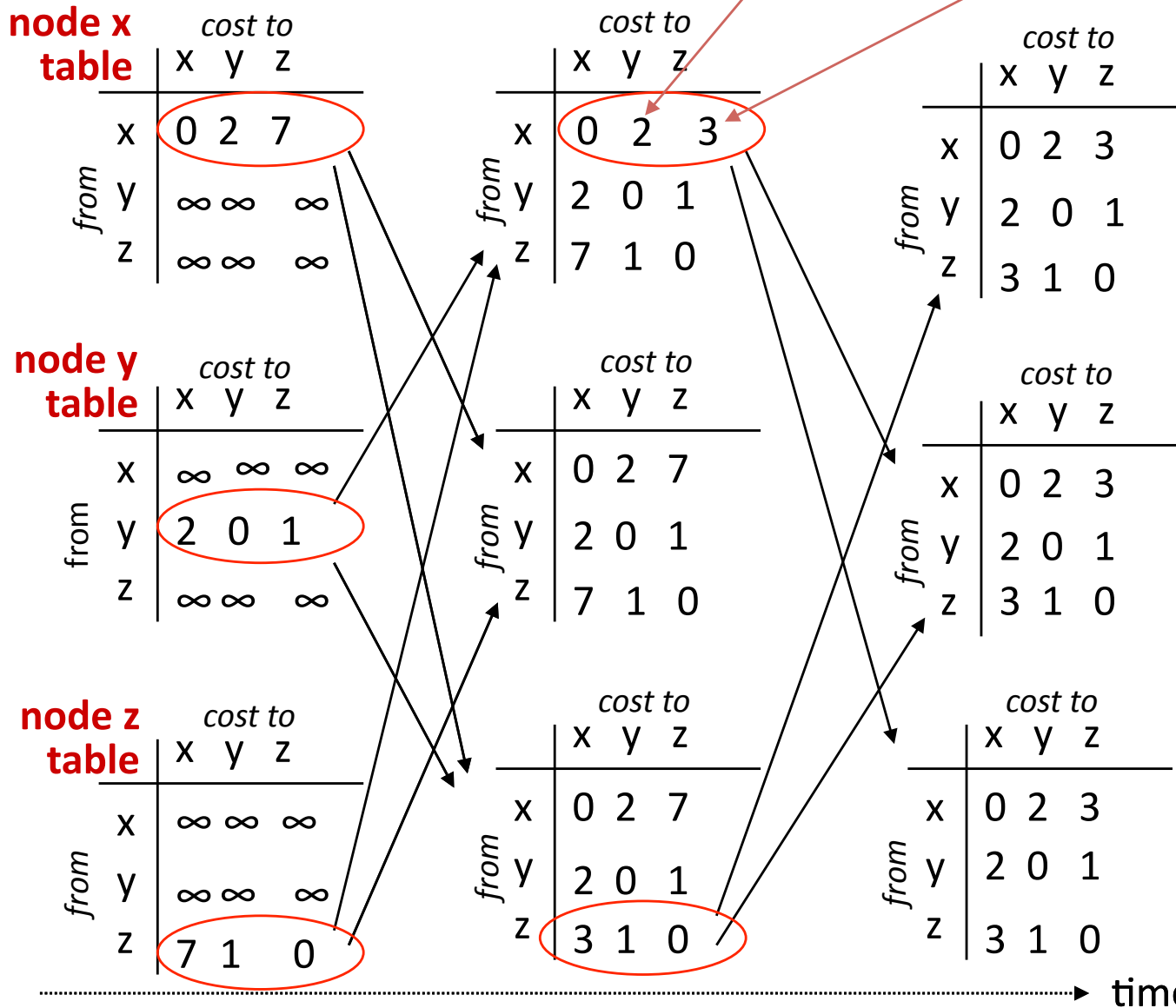
Network Layer

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

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

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

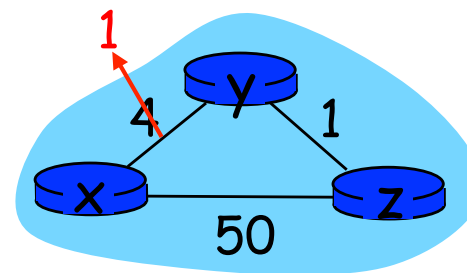
$$= \min\{2+1, 7+0\} = 3$$



# 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  $z$ .

# 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

# 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*

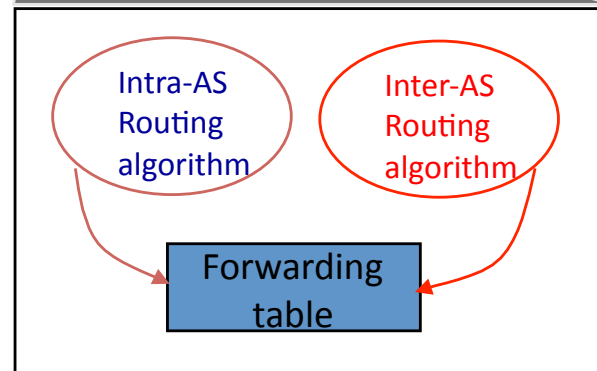
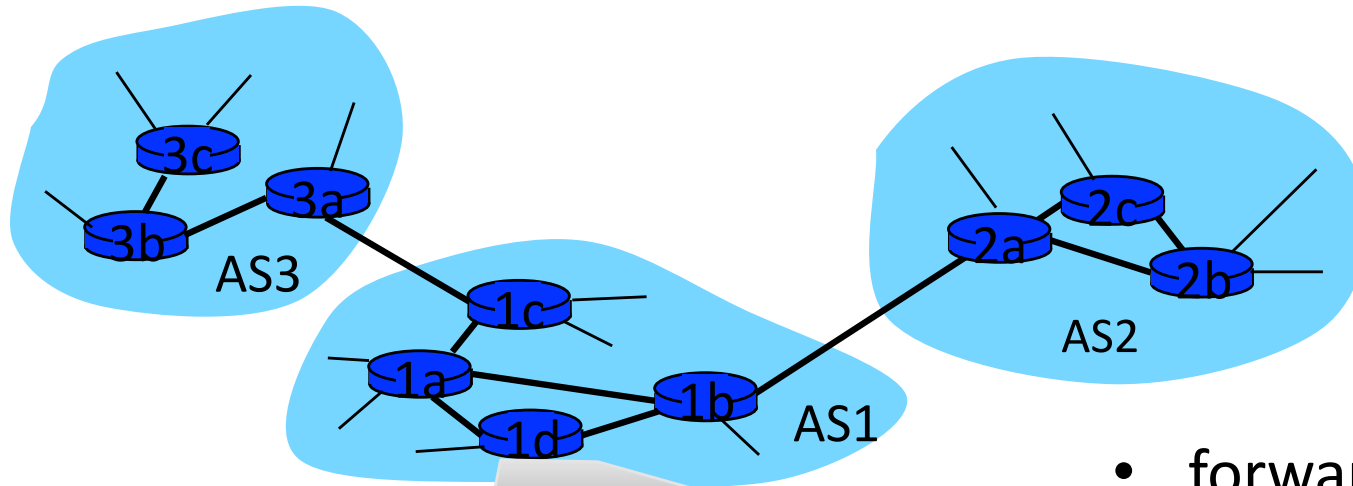
- internet = network of networks
- each network admin may want to control routing in its own network



# 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 dests
  - inter-AS & intra-AS sets entries for external dests

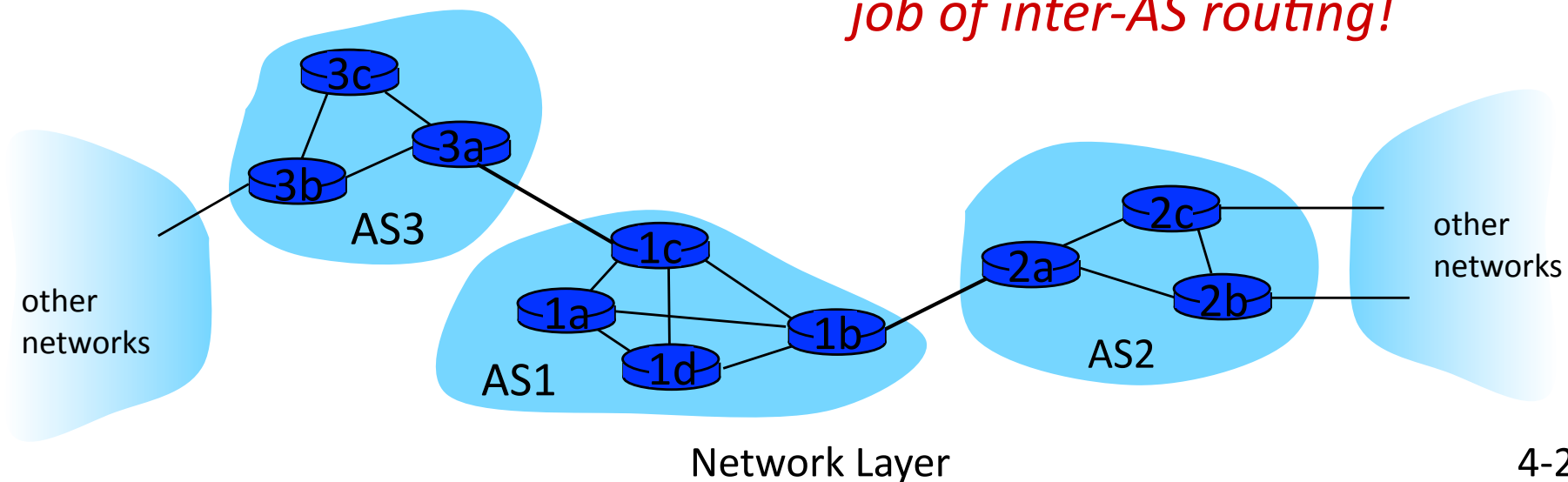
# Inter-AS tasks

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

*AS1 must:*

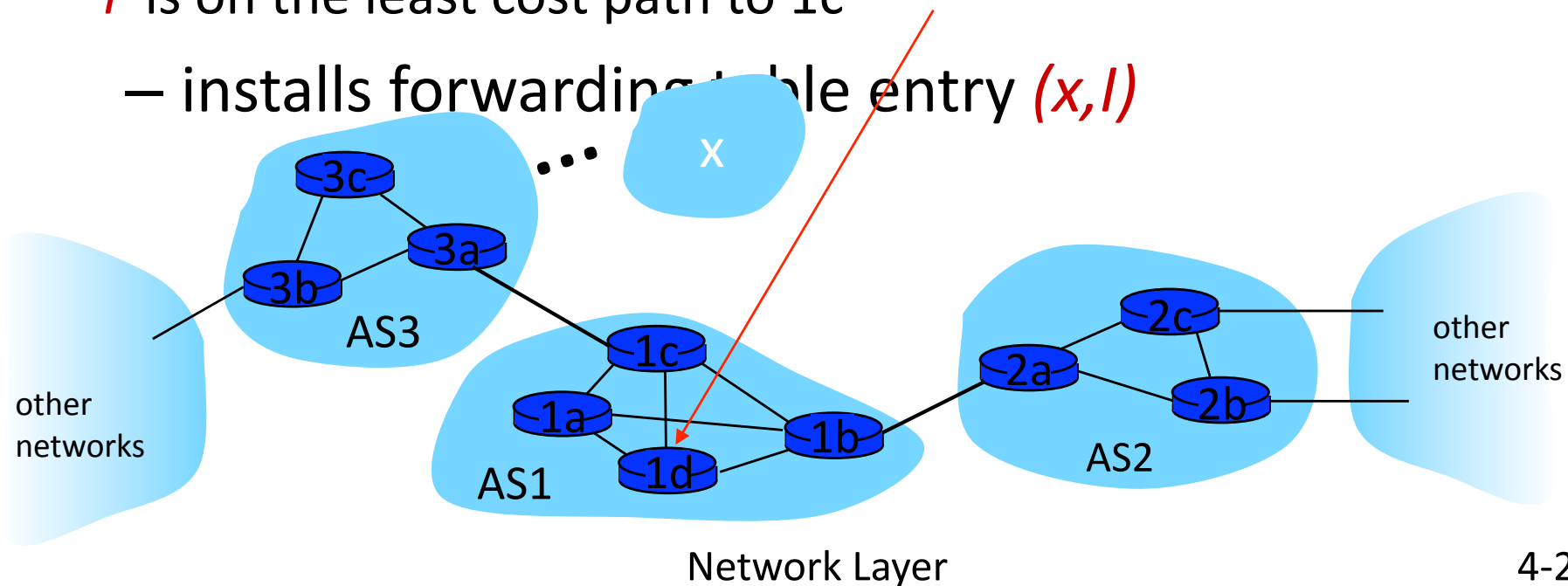
1. learn which destds are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

*job of inter-AS routing!*



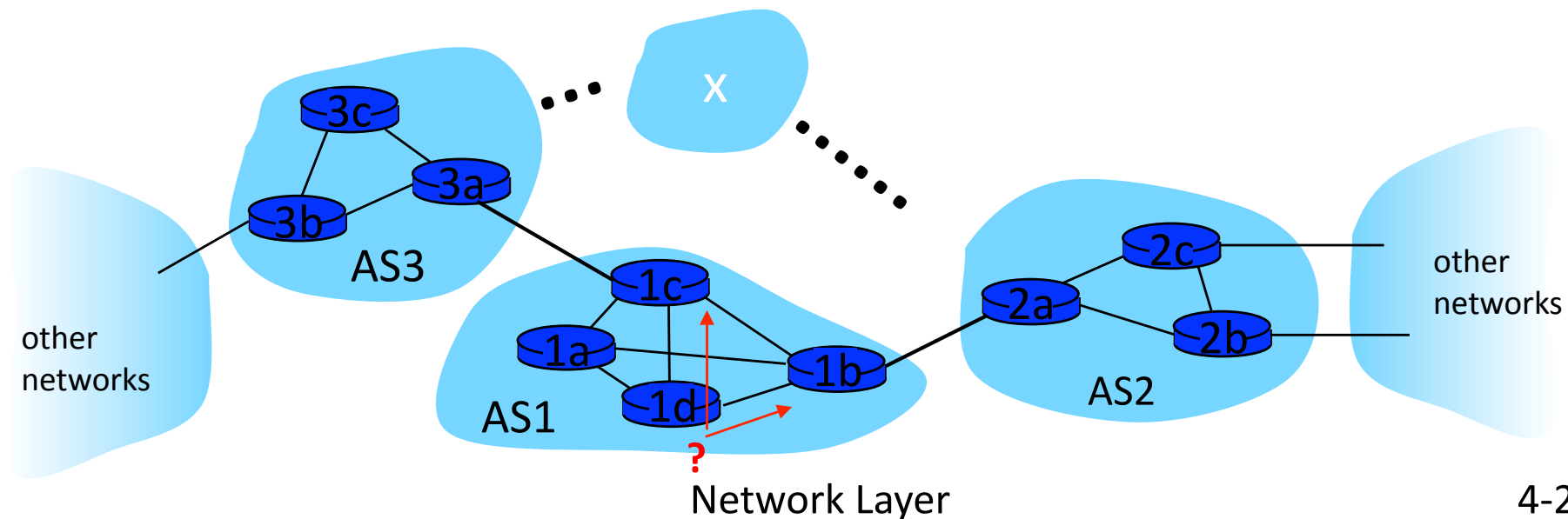
# Example: setting forwarding table in router 1d

- 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  $(x, l)$



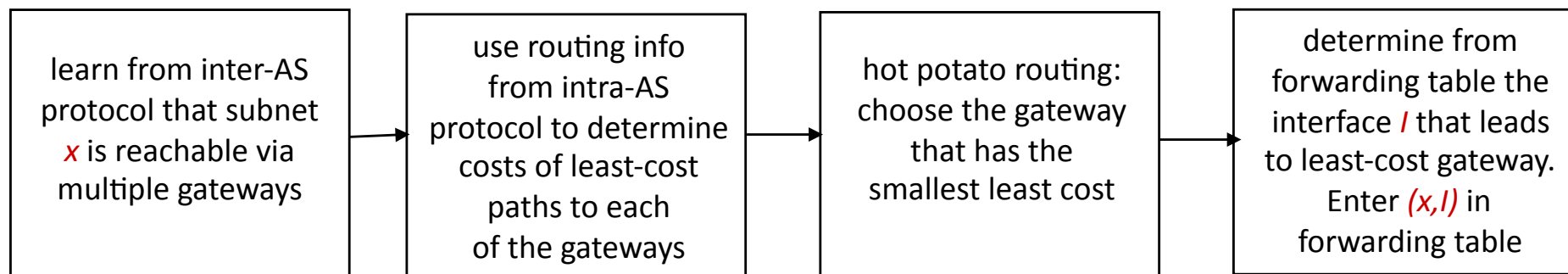
# Example: choosing among multiple ASes

- 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.



# Assignment

- Prepare a discussion group
  - 5 Member each group
- Prepare 3 Guest VM in Linux Fedora Core 16 without GUI. Connect those 3 guest VM network. Each VM must able to ping each other. ( Use VmWare, KVM, Virtual Box, Etc )
- Install Quagga Routing Application in each version ( Use the newest version of Quagga )