

جامعة حماة  
الكلية التطبيقية  
قسم تقنيات حاسوب  
السنة الرابعة  
إدارة شبكات /٢/

المحاضرة السابعة  
م. إناس عدي

# 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 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  $N'$

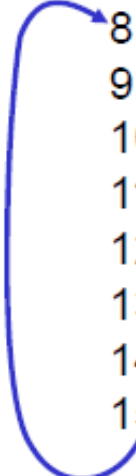
11 update  $D(v)$  for all  $v$  adjacent to  $w$  and not in  $N'$  :

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

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

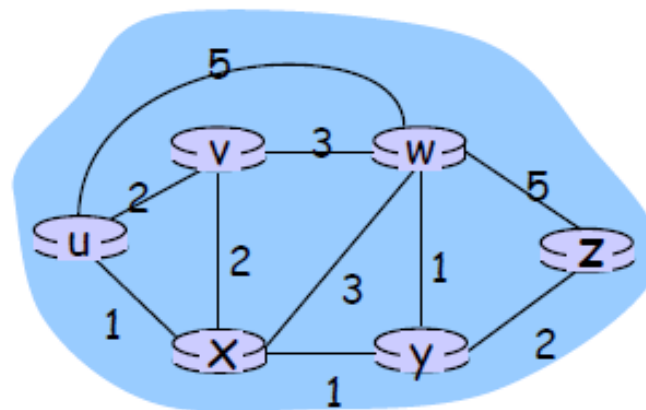
14 shortest path cost to  $w$  plus cost from  $w$  to  $v$  \*/

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



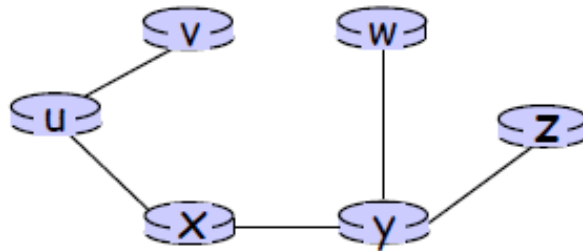
# Dijkstra's algorithm: example

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

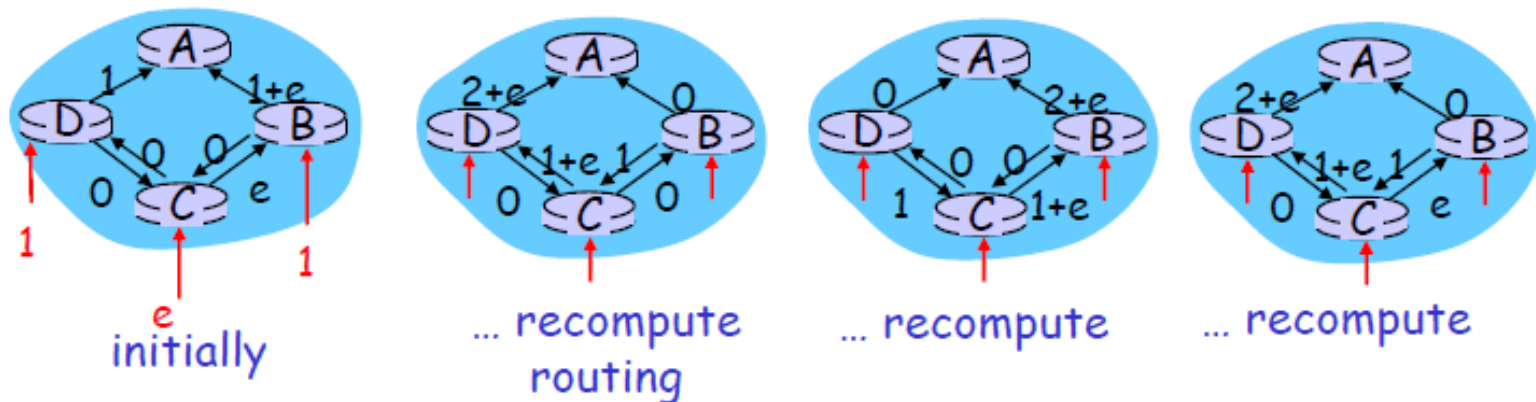
# Dijkstra's algorithm, discussion

**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 Algorithm

## Bellman-Ford Equation (dynamic programming)

Define

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

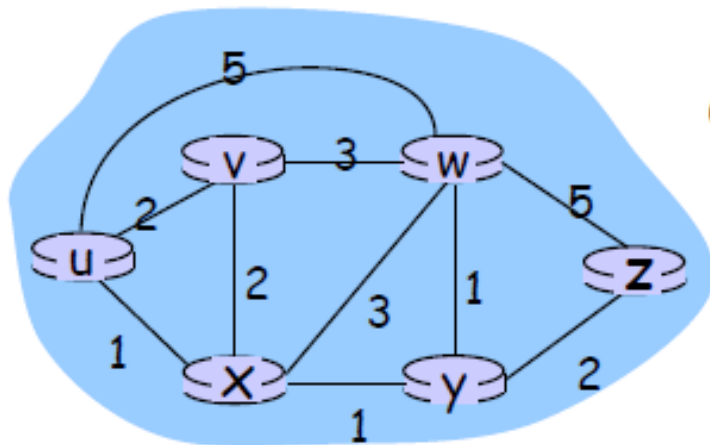
Then

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

where min is 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 that achieves minimum is next hop in shortest path → forwarding table

# Distance Vector Algorithm

- $D_x(y)$  = estimate of least cost from  $x$  to  $y$
- Node  $x$  knows cost to each neighbor  $v$ :  
 $c(x,v)$
- Node  $x$  maintains distance vector  $D_x = [D_x(y): y \in N]$
- Node  $x$  also maintains its neighbors' distance vectors
  - For each neighbor  $v$ ,  $x$  maintains  $D_v = [D_v(y): y \in N]$

# Distance vector algorithm (4)

## Basic idea:

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

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

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

## Distance Vector Algorithm (5)

### Iterative, asynchronous:

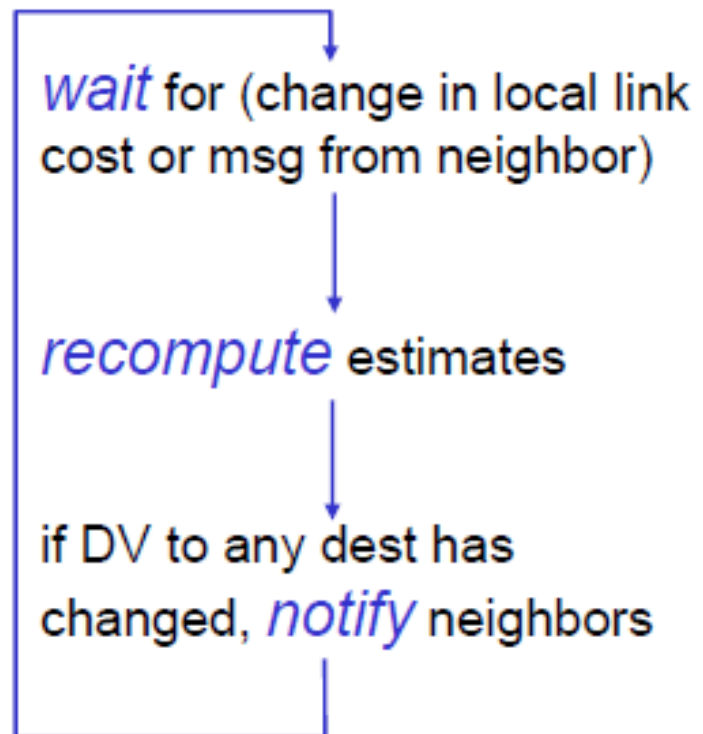
each local iteration caused by:

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

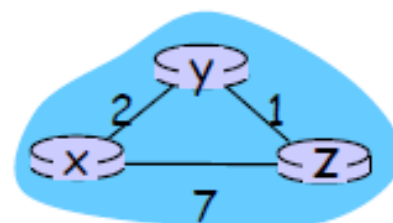
		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	7	1	0

**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 →

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

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

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

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

**node x table**

		cost to		
		x	y	z
from	x	0	2	7
	y	$\infty$	$\infty$	$\infty$
	z	$\infty$	$\infty$	$\infty$

**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

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

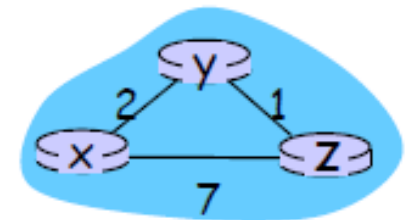
		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

		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

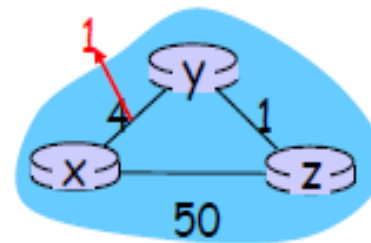


time

## 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



At time  $t_0$ ,  $y$  detects the link-cost change, updates its DV, and informs its neighbors.

At time  $t_1$ ,  $z$  receives the update from  $y$  and updates its table. It computes a new least cost to  $x$  and sends its neighbors its DV.

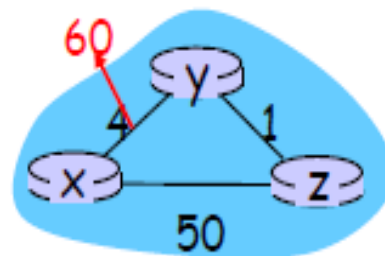
At time  $t_2$ ,  $y$  receives  $z$ 's update and updates its distance table.  $y$ 's least costs do not change and hence  $y$  does *not* send any message to  $z$ .

"good  
news  
travels  
fast"

# Distance Vector: link cost changes

## Link cost changes:

- ❑ good news travels fast
- ❑ 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: Network Layer

- ❑ 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 200 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

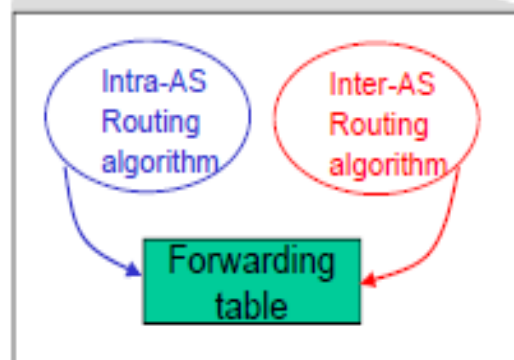
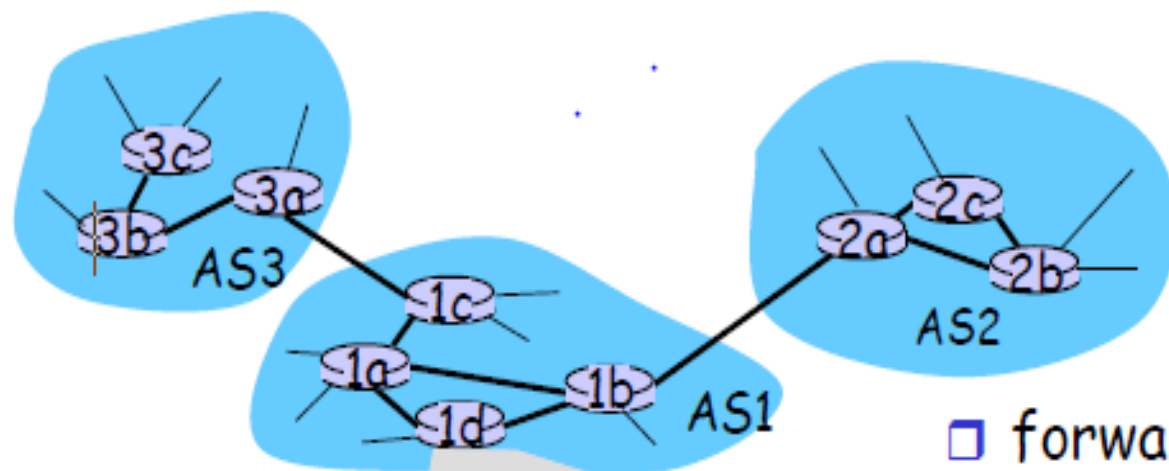
# Hierarchical Routing

- ❑ 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

- ❑ Direct 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

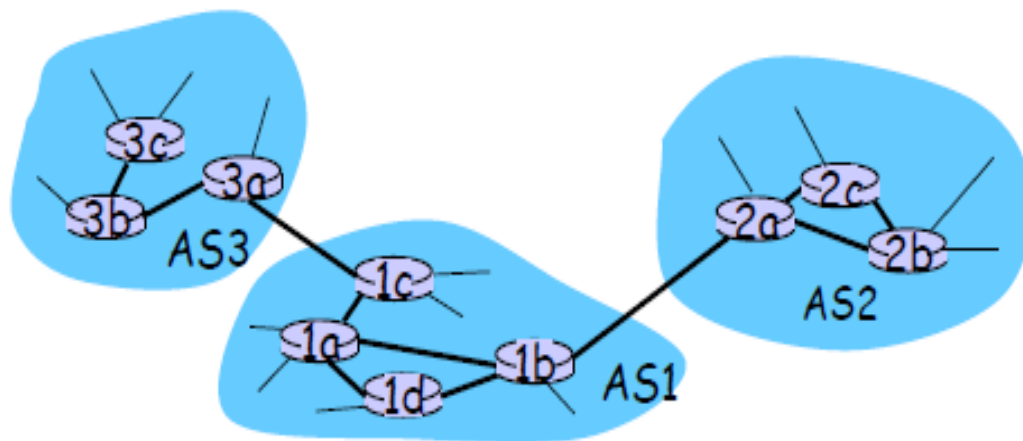
# Inter-AS tasks

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

## AS1 must:

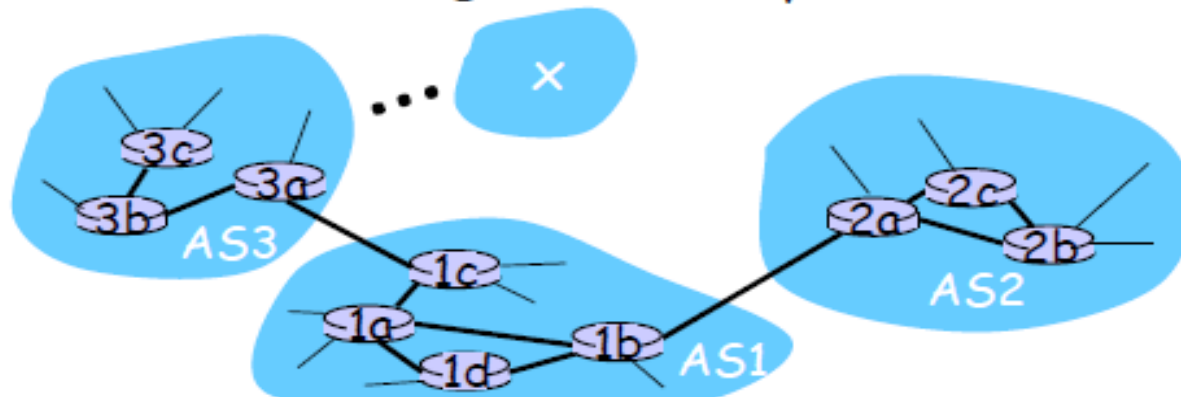
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

Job of inter-AS routing!



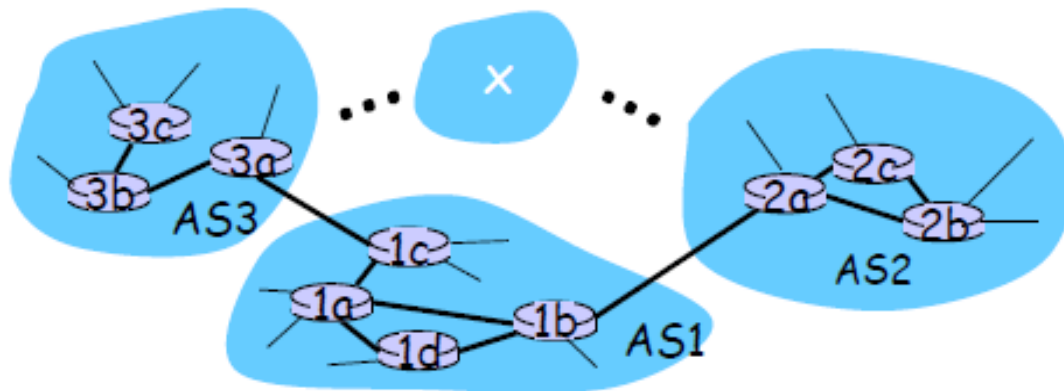
## Example: Setting forwarding table in router 1d

- 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  $I$  is on the least cost path to 1c.
  - installs forwarding table entry  $(x, I)$



## 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!





## 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.

