

Computer Networks

Instructor: Niklas Carlsson

Email: niklas.carlsson@liu.se

Notes derived from "*Computer Networking: A Top Down Approach*", by Jim Kurose and Keith Ross, Addison-Wesley.

The slides are adapted and modified based on slides from the book's companion Web site, as well as modified slides by Anirban Mahanti and Carey Williamson.

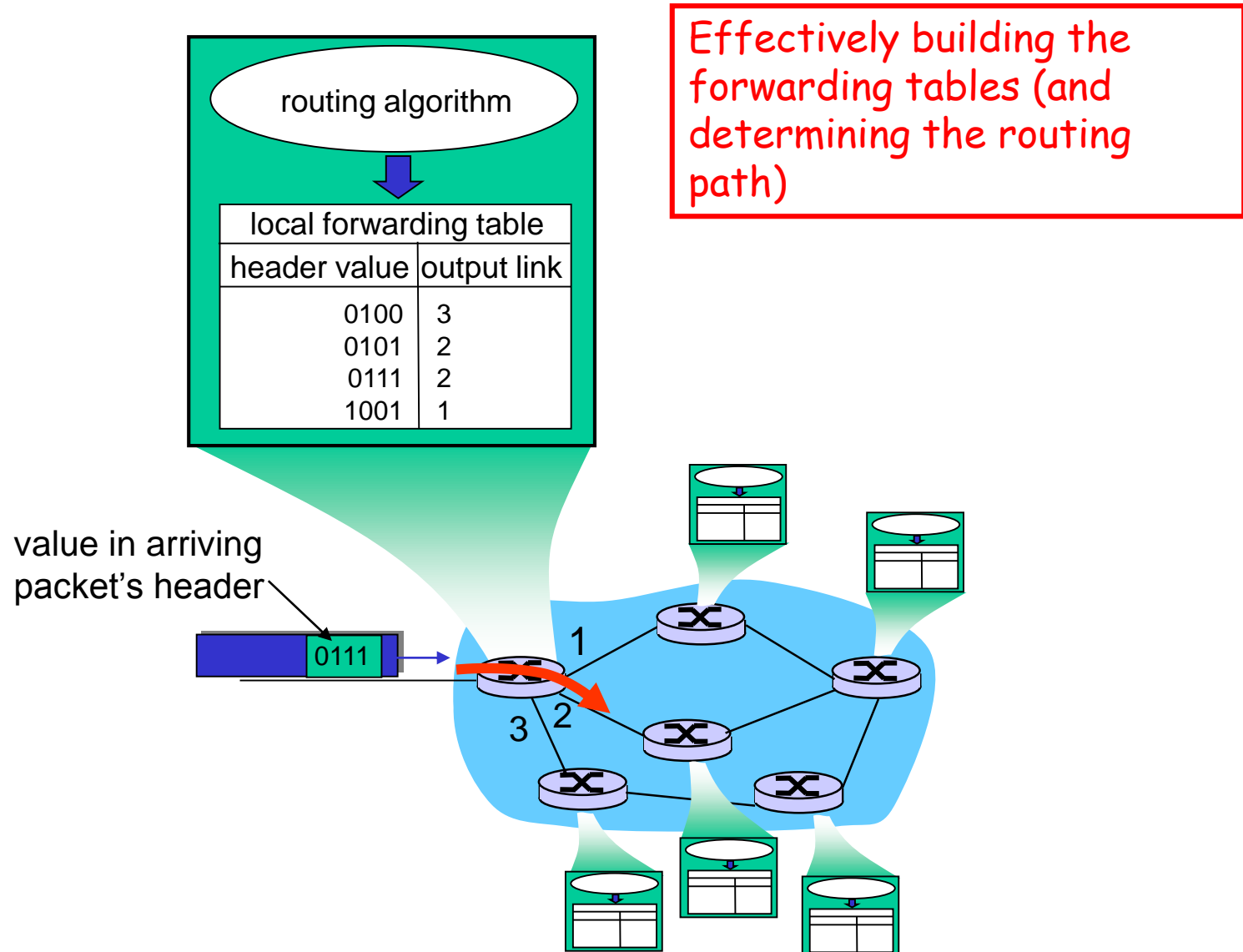
Routing Algorithms

Link State

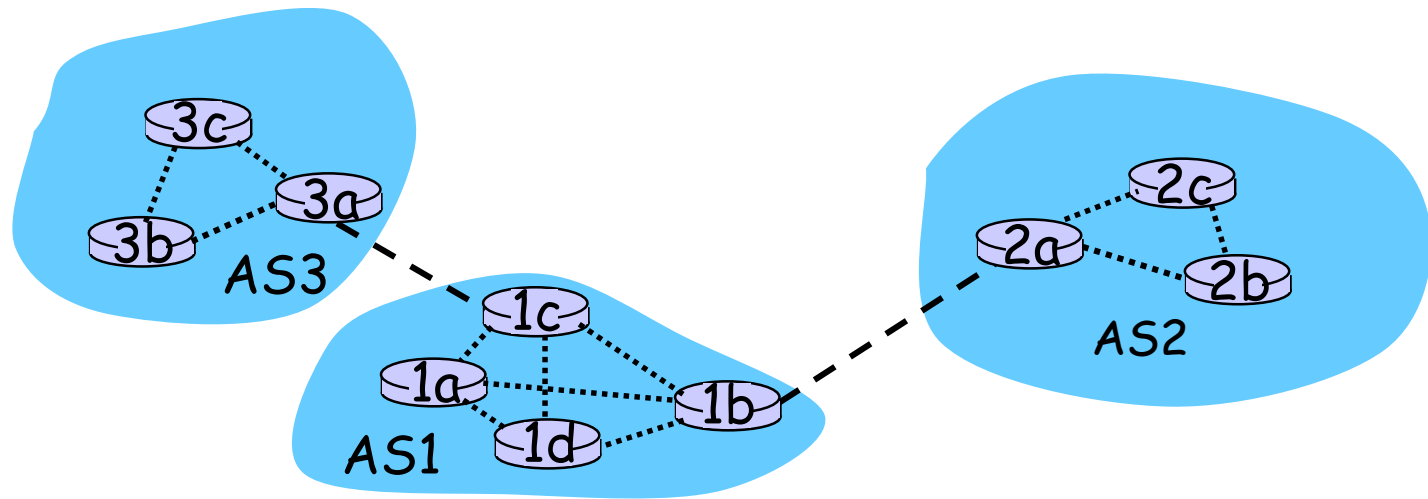
Distance Vector

Hierarchical Routing

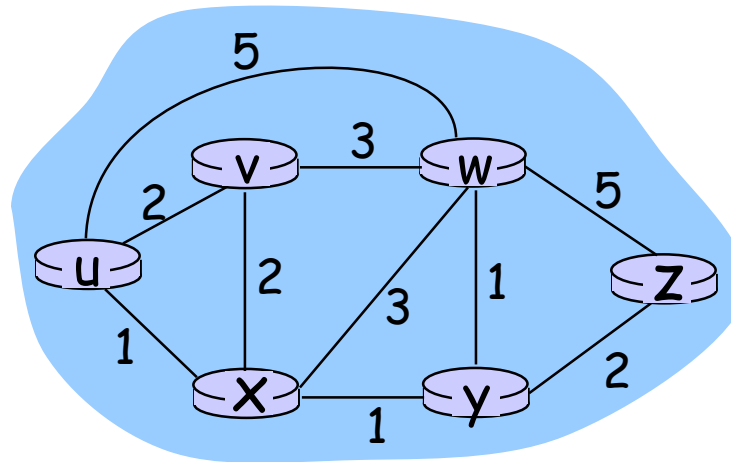
Interplay between routing and forwarding



Let's start with the case where we operate the full (sub) network ...



Graph abstraction



Graph: $G = (N, E)$

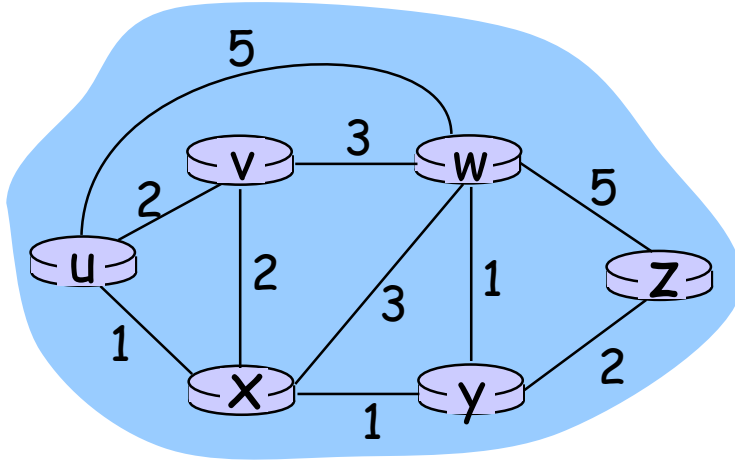
N = set of routers = $\{ u, v, w, x, y, z \}$

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

Remark: Graph abstraction is useful in other network contexts too

Example: 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 $(x_1, x_2, x_3, \dots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$

Question: What's the least-cost path between u and z ?

Routing algorithm: find "good" paths from source to destination router.

Routing Algorithm Classification

1. Global vs decentralized ?

Global:

- ❑ all routers have complete topology, link cost info
- ❑ "link state" algorithms

Decentralized:

- ❑ router knows about physically-connected neighbors
- ❑ Iterative, distributed computations
- ❑ "distance vector" algorithms

2. Static vs dynamic?

Static:

- ❑ routes change slowly over time

Dynamic:

- ❑ routes change more quickly
 - periodic update
 - in response to link cost changes

3. Load sensitivity?

- Many Internet routing algorithms are load insensitive

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

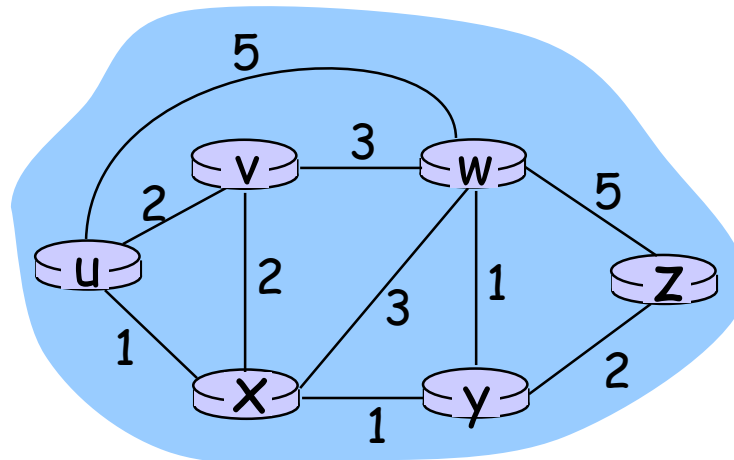
13 /* new cost to v is either old cost to v or known

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

15 **until all nodes in N'**

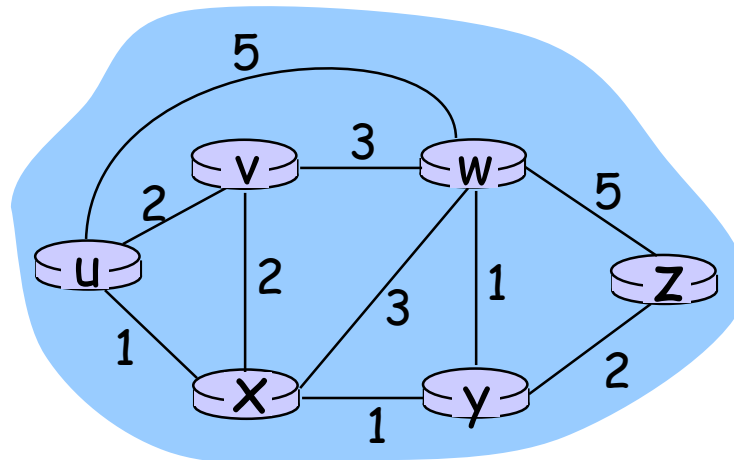
Dijkstra's algorithm: example

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	∞	∞
1						
2						
3						
4						
5						



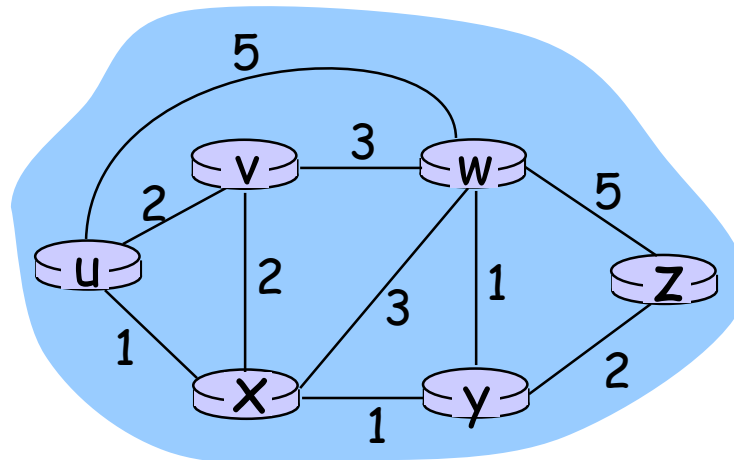
Dijkstra's algorithm: example

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	∞	∞
1	ux					
2						
3						
4						
5						



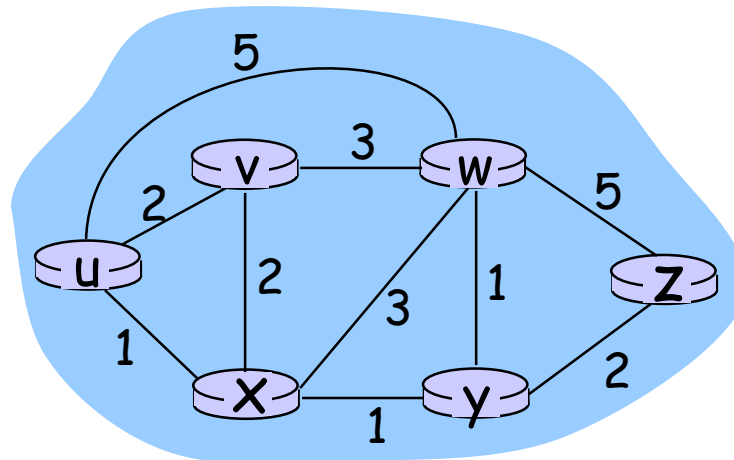
Dijkstra's algorithm: example

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	∞	∞
1	ux	2,u	4,x		2,x	∞
2						
3						
4						
5						



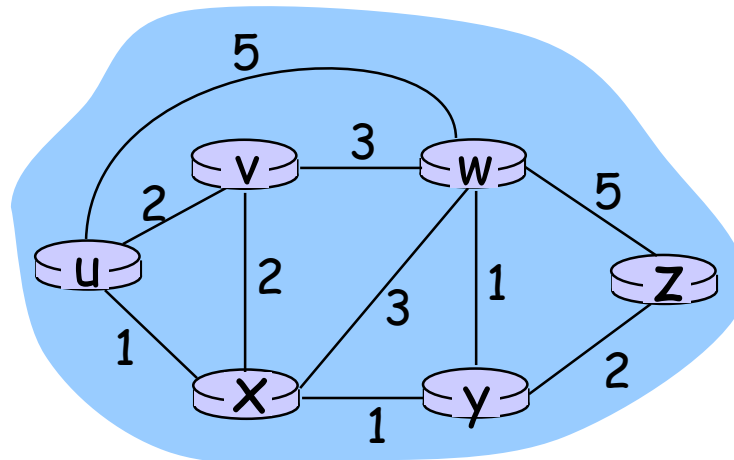
Dijkstra's algorithm: example

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	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy					
3						
4						
5						




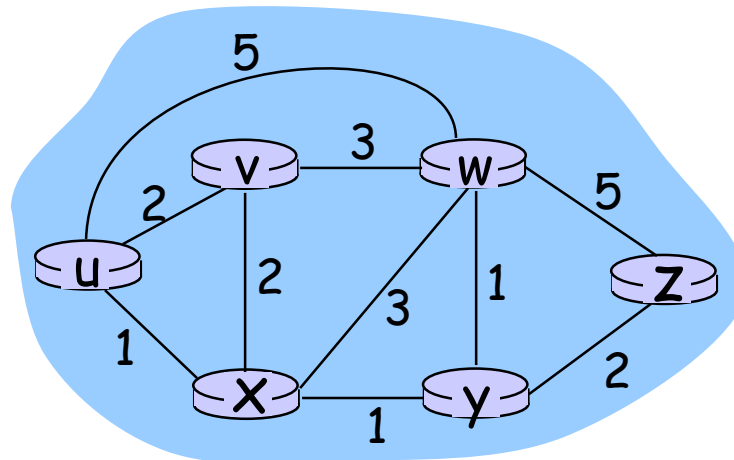
Dijkstra's algorithm: example

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	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3						
4						
5						



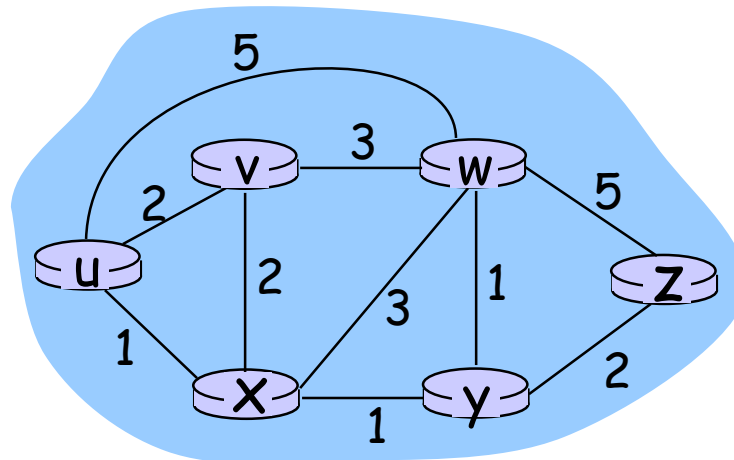
Dijkstra's algorithm: example

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	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv 					
4						
5						



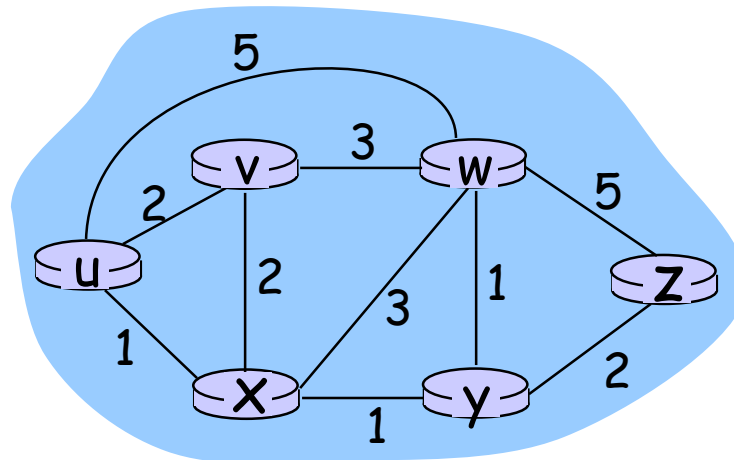
Dijkstra's algorithm: example

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	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4						
5						



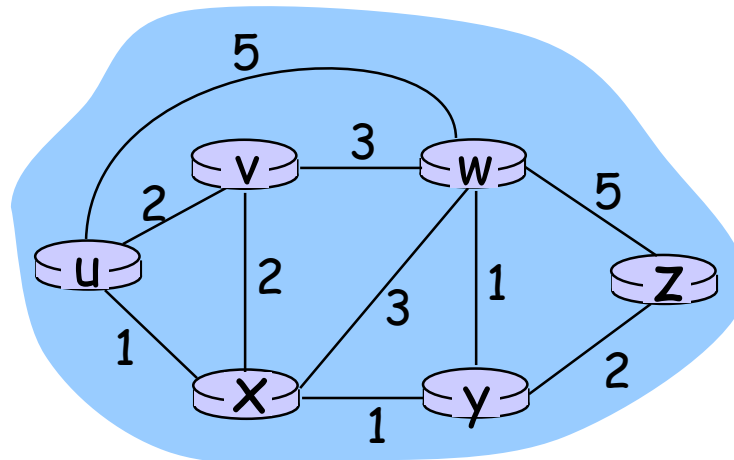
Dijkstra's algorithm: example

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	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					
5						



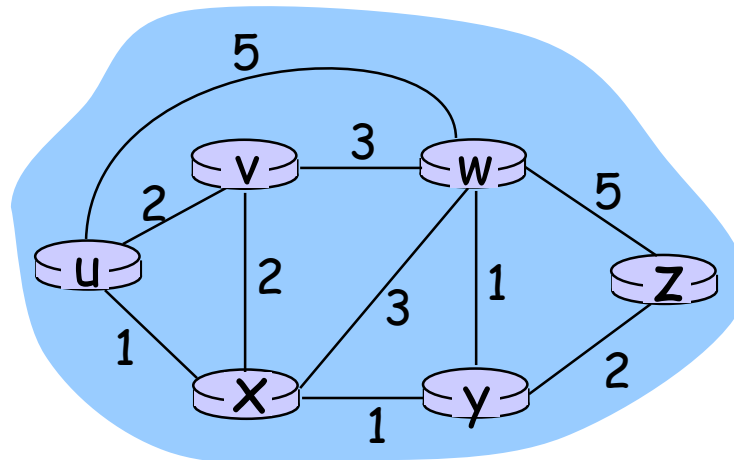
Dijkstra's algorithm: example

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	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5						



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	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



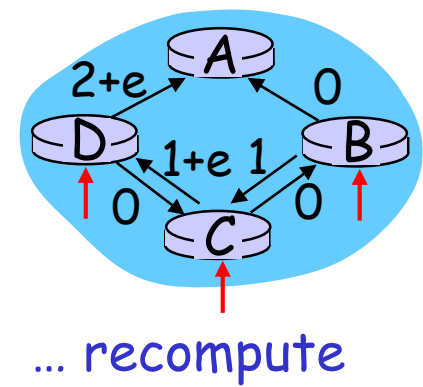
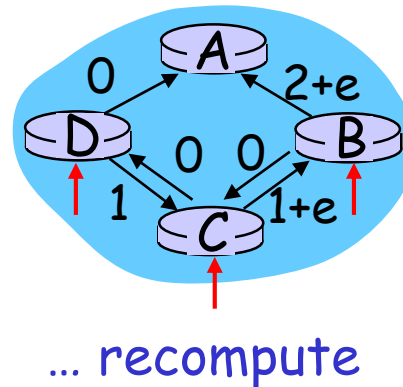
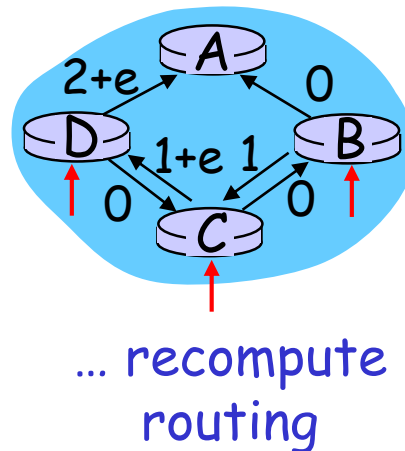
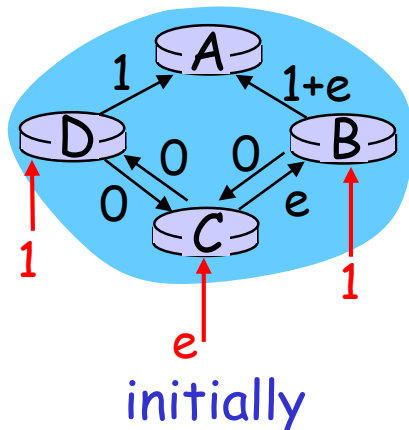
Dijkstra's algorithm, discussion

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 (1)

Bellman-Ford Equation (dynamic programming)

Define

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

Distance Vector Algorithm (1)

Bellman-Ford Equation (dynamic programming)

Define

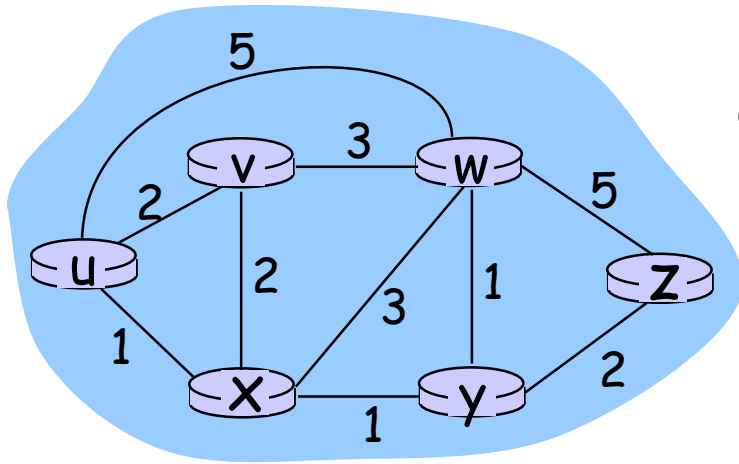
$d_x(y) :=$ cost of least-cost path from x to y

Then

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

where min is taken over all neighbors of x

Bellman-Ford example (2)



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 (3)

- $D_x(y)$ = estimate of least cost from x to y
- Distance vector: $D_x = [D_x(y): y \in N]$

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

Distance vector algorithm (4)

Basic idea:

- Each node periodically sends its own distance vector estimate to neighbors
- When 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 some conditions, the estimate $D_x(y)$ converge 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 when its DV changes
 - neighbors then notify their neighbors if necessary

Distance Vector Algorithm (5)

Iterative, asynchronous:

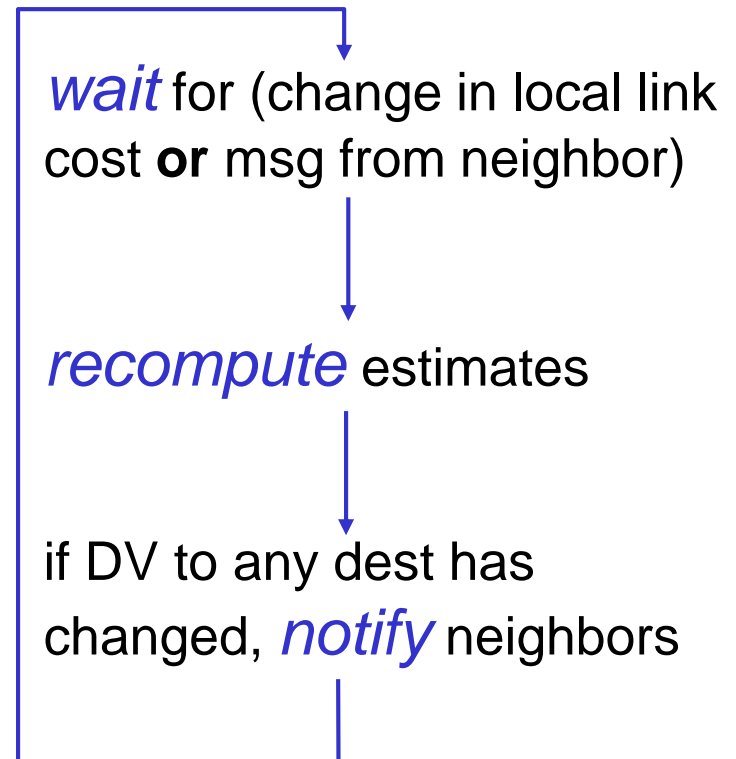
each local iteration caused by:

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

Distributed:

- ❑ each node notifies neighbors when its DV changes
 - neighbors then notify their neighbors if necessary

Each node:



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			
	y			
	z			

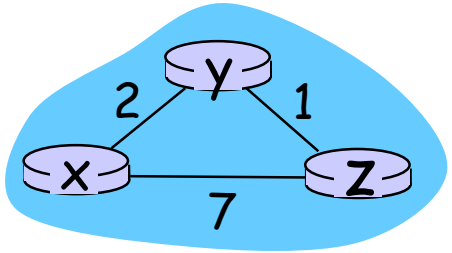
		cost to		
		x	y	z
from	x			
	y			
	z			

		cost to		
		x	y	z
from	x			
	y			
	z			

		cost to		
		x	y	z
from	x			
	y			
	z			

		cost to		
		x	y	z
from	x			
	y			
	z			

		cost to		
		x	y	z
from	x			
	y			
	z			



.....> time

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			
	y			
	z			

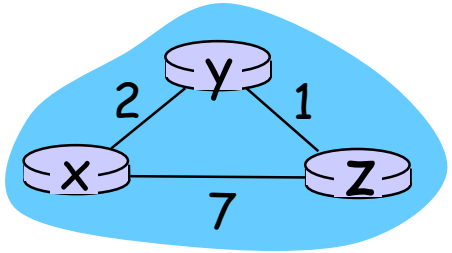
		cost to		
		x	y	z
from	x	0	2	7
	y			
	z			

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

		cost to		
		x	y	z
from	x			
	y			
	z			

		cost to		
		x	y	z
from	x			
	y			
	z			

		cost to		
		x	y	z
from	x			
	y			
	z			



.....> time

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			
	y	2	0	1
	z	7	1	0

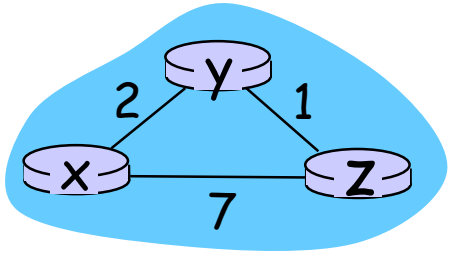
		cost to		
		x	y	z
from	x	0	2	7
	y			
	z	7	1	0

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

		cost to		
		x	y	z
from	x			
	y			
	z			

		cost to		
		x	y	z
from	x			
	y			
	z			

		cost to		
		x	y	z
from	x			
	y			
	z			



.....> time

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			
	y	2	0	1
	z	7	1	0

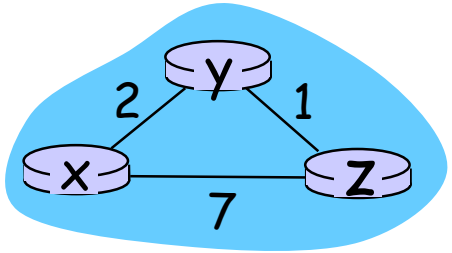
		cost to		
		x	y	z
from	x	0	2	7
	y			
	z	7	1	0

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

		cost to		
		x	y	z
from	x			
	y			
	z			

		cost to		
		x	y	z
from	x			
	y			
	z			

		cost to		
		x	y	z
from	x			
	y			
	z			



.....> 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		
from		x	y	z
	x	0	2	7
	y	∞	∞	∞
	z	∞	∞	∞

node y table

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

node z table

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

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

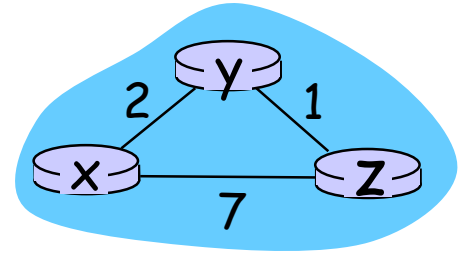
		cost to		
from		x	y	z
	x	0	2	7
	y			
	z	7	1	0

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

		cost to		
from		x	y	z
	x			
	y			
	z			

		cost to		
from		x	y	z
	x			
	y			
	z			

		cost to		
from		x	y	z
	x			
	y			
	z			



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

node y table

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

node z table

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

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

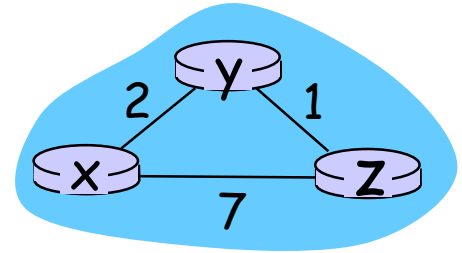
		cost to		
from		x	y	z
	x	0	2	7
	y			
	z	7	1	0

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

		cost to		
from		x	y	z
	x			
	y			
	z			

		cost to		
from		x	y	z
	x			
	y			
	z			

		cost to		
from		x	y	z
	x			
	y			
	z			



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

node y table

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

node z table

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

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

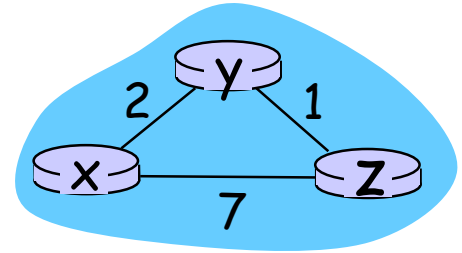
		cost to		
from		x	y	z
	x	0	2	7
	y	2	0	1
	z	7	1	0

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

		cost to		
from		x	y	z
	x			
	y			
	z			

		cost to		
from		x	y	z
	x			
	y			
	z			

		cost to		
from		x	y	z
	x			
	y			
	z			



time

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

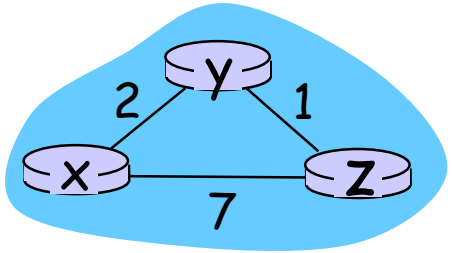
		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

		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			
	y			
	z			

		cost to		
		x	y	z
from	x			
	y			
	z			

		cost to		
		x	y	z
from	x			
	y			
	z			



.....> time

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

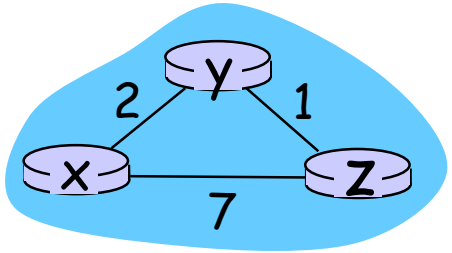
		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

		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			
	y			
	z			

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

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



.....> time

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

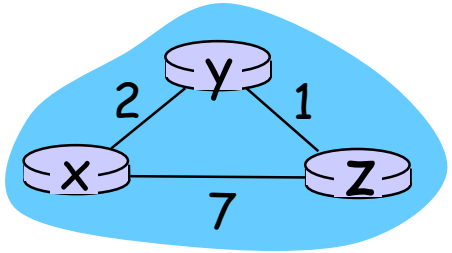
		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

		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			
	y	2	0	1
	z	3	1	0

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

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



.....> time

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

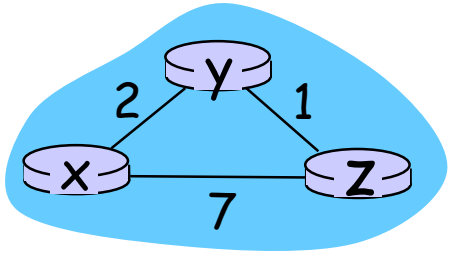
		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

		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			
	y	2	0	1
	z	3	1	0

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

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



.....> time

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

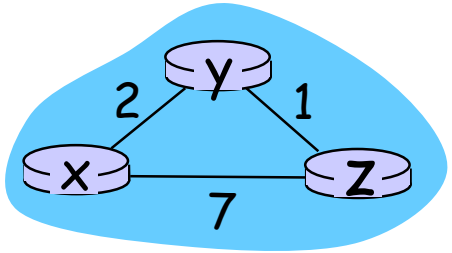
		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

		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



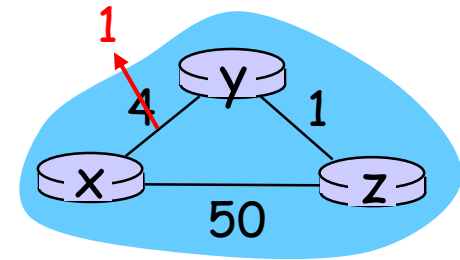
.....> time

Maybe another example ...

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”

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

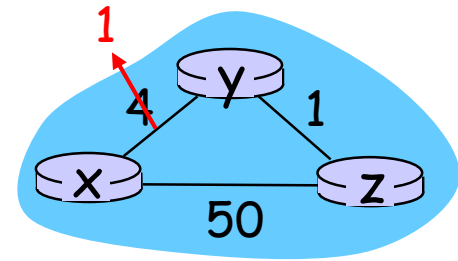
At time t_1 , z receives the update from y and updates its table. It computes a new least cost to x and sends its neighbors its DV.

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 .

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

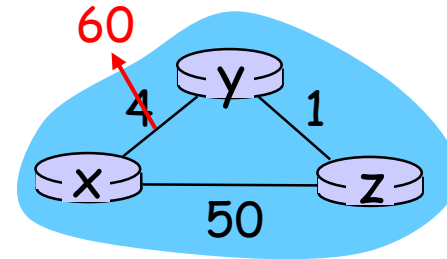


“good
news
travels
fast”

Distance Vector: link cost changes

Link cost changes:

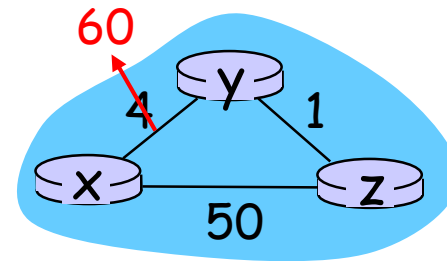
- ❑ good news travels fast
- ❑ bad news travels slow - "count to infinity" problem!
- ❑ 44 iterations before algorithm stabilizes: see text



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

Speed of Convergence

- ❑ LS: $O(n^2)$ algorithm requires $O(nE)$ msgs
 - may have oscillations
- ❑ DV: convergence time varies
 - may have 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 and Routing on the Internet

Hierarchical Routing: Motivation

- ❑ Our routing study thus far - idealization
 - all routers identical, network "flat"
- ❑ **scale:** with billions of 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

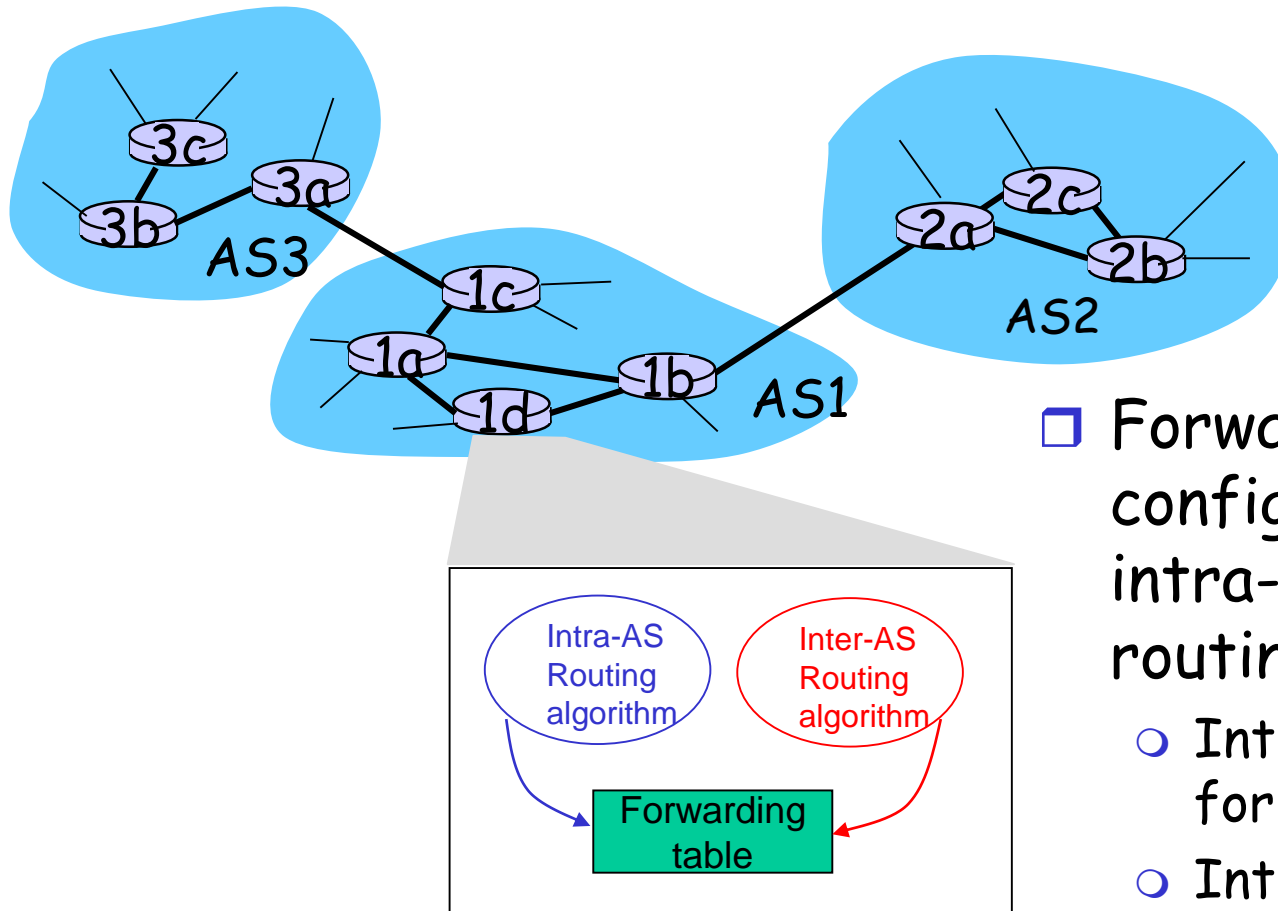
Hierarchical Routing

- ❑ 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
- ❑ Establishes a “peering” relationship
- ❑ Peers run an “inter-AS routing” protocol

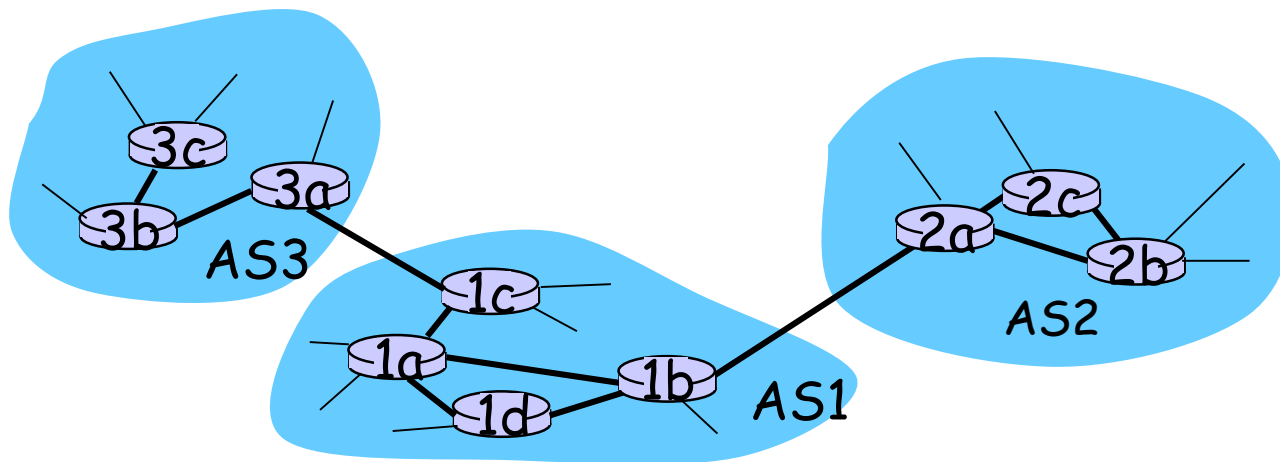
Interconnected ASes



- ❑ Forwarding table is 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

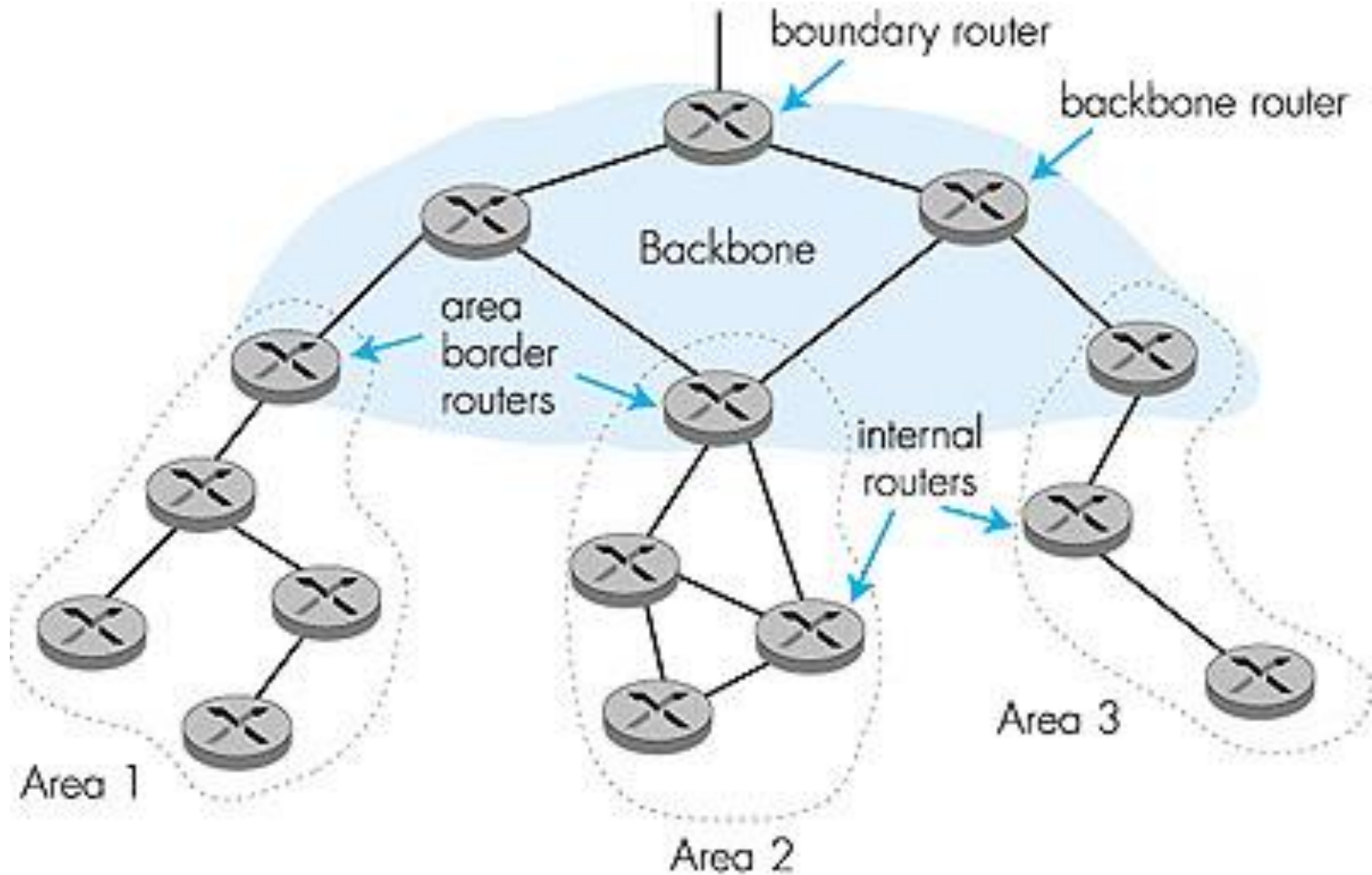
- ❑ Obtain reachability information from neighboring AS(s)
- ❑ Propagate this info to all routers within the AS
- ❑ All Internet gateway routers run a protocol called BGPv4 (we will talk about this soon)



Intra-AS Routing

- ❑ Also known as **Interior Gateway Protocols (IGP)**
- ❑ Most common Intra-AS routing protocols:
 - RIP: Routing Information Protocol (DV protocol)
 - OSPF: Open Shortest Path First (Link-State)
 - Including hierarchical OSPF for large domains
 - EIGRP: Enhanced Interior Gateway Routing Protocol
 - Cisco protocol: originally proprietary, made open in 2013

Hierarchical OSPF

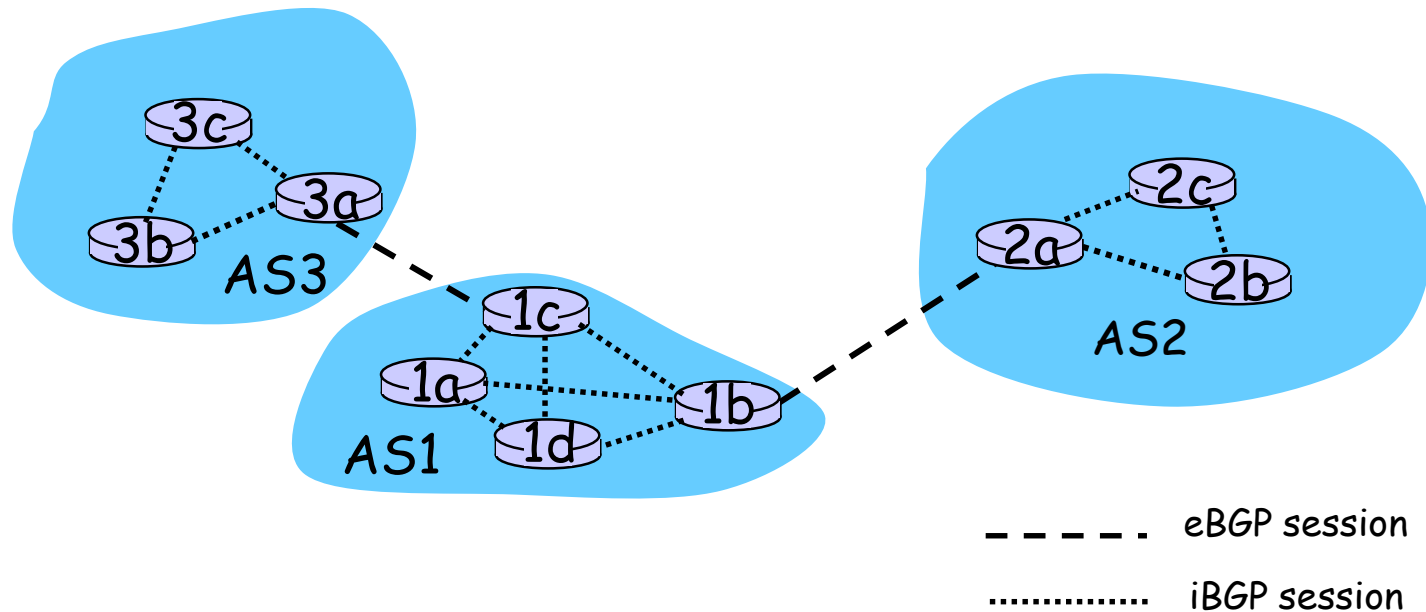


Internet inter-AS routing: BGP

- ❑ **BGP (Border Gateway Protocol):** *the de facto standard*
- ❑ BGP provides each AS a means to:
 1. Obtain subnet reachability information from neighboring ASs.
 2. Propagate the reachability information to all routers internal to the AS.
 3. Determine "good" routes to subnets based on reachability information and policy.
- ❑ Allows a subnet to advertise its existence to rest of the Internet: *"I am here"*

BGP basics

- ❑ Pairs of routers (BGP peers) exchange routing info over semi-permanent TCP connections: **BGP sessions**
- ❑ Note that BGP sessions do not correspond to physical links.
- ❑ When AS2 advertises a prefix to AS1, AS2 is **promising** it will forward any datagrams destined to that prefix towards the prefix.
 - AS2 can aggregate prefixes in its advertisement



Path attributes & BGP routes

- ❑ When advertising a prefix, advert includes BGP attributes.
 - prefix + attributes = "route"
- ❑ Two important attributes:
 - **AS-PATH**: contains the ASs through which the advert for the prefix passed: AS 67 AS 17
 - **NEXT-HOP**: Indicates the specific internal-AS router to next-hop AS. (There may be multiple links from current AS to next-hop-AS.)
- ❑ When gateway router receives route advert, uses **import policy** to accept/decline.

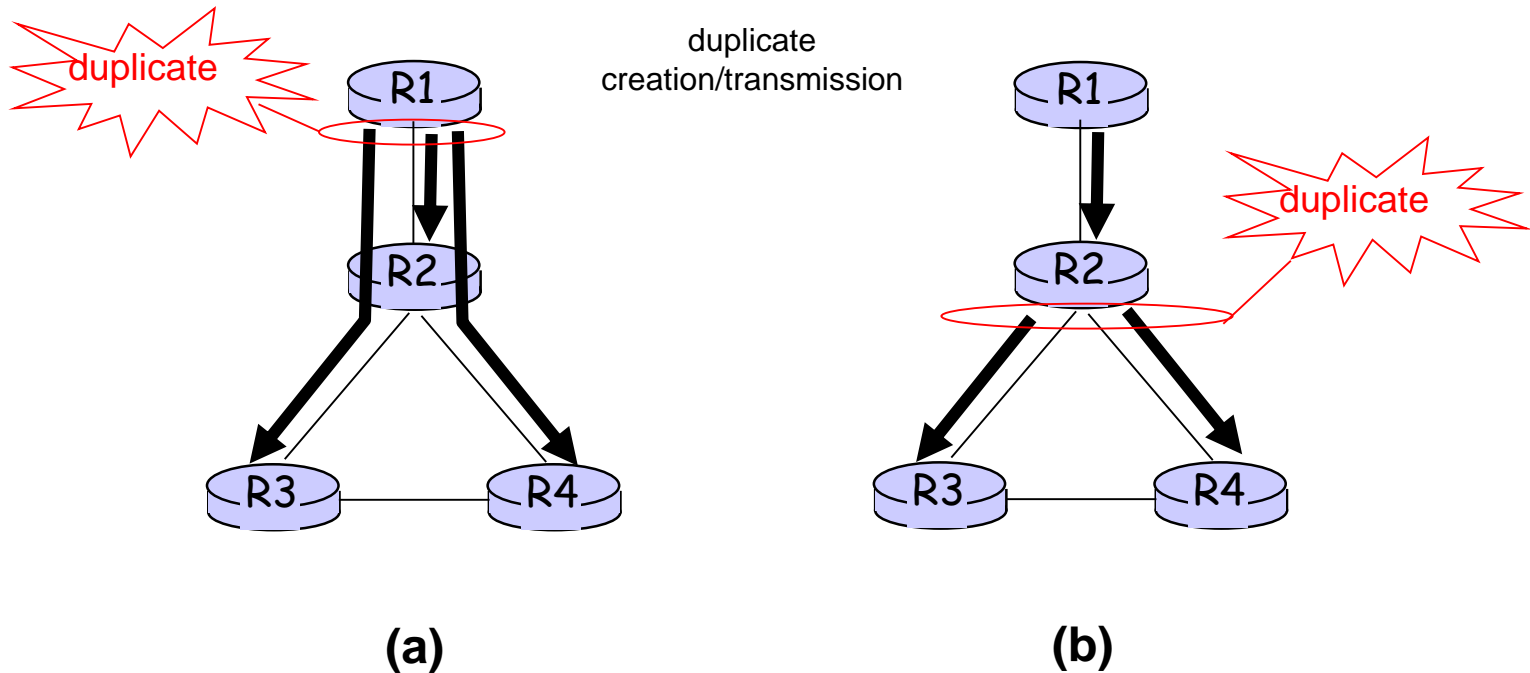
BGP route selection

- ❑ Router may learn about more than 1 route to some prefix. Router must select route.
- ❑ Elimination rules:
 1. Local preference value attribute: policy decision
 2. Shortest AS-PATH
 3. Closest NEXT-HOP router: hot potato routing
 4. Additional criteria

Quiz based on China Telecom Incident

- ❑ Slides (20-46) used for quiz found here:
<http://www.ida.liu.se/~nikca/papers/wiscs14.slides.pdf>
- ❑ Attack described and characterized in
 - R. Hiran, N. Carlsson, and P. Gill, "Characterizing Large-scale Routing Anomalies: A Case Study of the China Telecom Incident", Proc. PAM, Hong Kong, China, Mar. 2013.

Multicast/Broadcast



Source-duplication versus in-network duplication.
(a) source duplication, (b) in-network duplication

Network Layer Routing: summary

What we've covered:

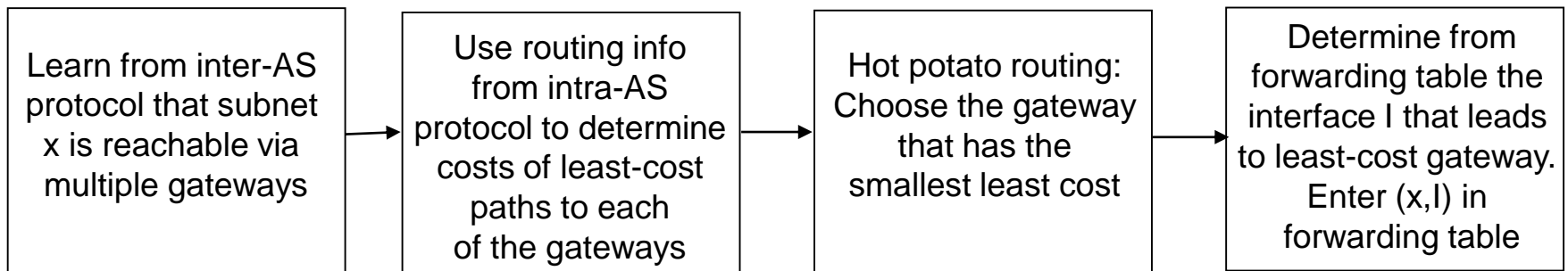
- ❑ network layer services
- ❑ routing principles: link state and distance vector
- ❑ hierarchical routing
- ❑ Internet routing protocols RIP, OSPF, BGP
- ❑ what's inside a router?

Next stop:
the Data
link layer!

More slides ...

Example: Choosing among multiple ASes

- ❑ Now suppose AS1 learns from the 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 the job on inter-AS routing protocol!
- ❑ **Hot potato routing**: send packet towards closest of two routers.

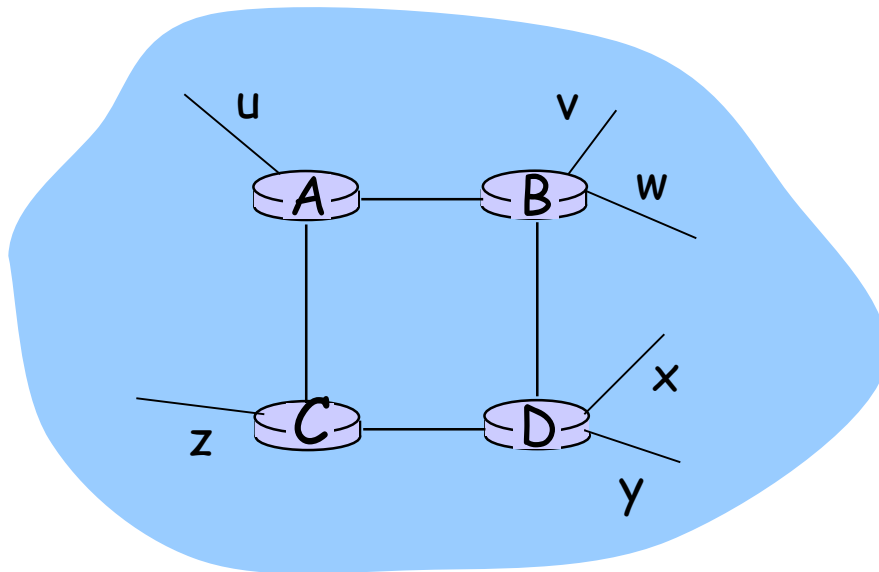


Hierarchical Routing

Routing in the Internet

RIP (Routing Information Protocol)

- ❑ Distance vector algorithm
- ❑ Included in BSD-UNIX Distribution in 1982
- ❑ Distance metric: # of hops (max = 15 hops)



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

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 router
- ❑ Advertisements disseminated to **entire** AS (via flooding)
 - Carried in OSPF messages directly over IP (rather than TCP or UDP)

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; high for real time)
- ❑ Integrated uni- and **multicast** support:
 - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- ❑ **Hierarchical** OSPF in large domains.