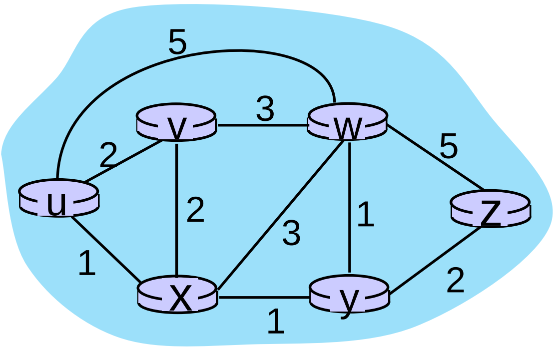
Computer Networks II

Routing Algorithms

Distance Vector Routing Protocol

Amitangshu Pal
Computer Science and Engineering
IIT Kanpur

Distance Vector Algorithm : Notations



- Graph: $G = (N,E)$
- $c_{a,b}$: cost of direct link connecting a and b
  - e.g., $c_{w,z} = 5$, $c_{u,z} = \infty$

- $D_x(y)$ = estimated cost of least-cost path from x to y
- x maintains its own distance vector $D_x = [D_x(y): y \in N]$
- x also maintains its neighbors distance vector
  - For each neighbor v, x maintains $D_v = [D_v(y): y \in N]$

Distance Vector Algorithm

Based on Bellman-Ford (BF) equation:

Bellman-Ford equation

Let $D_x(y)$: estimated cost of least-cost path from x to y

Then:

$$D_x(y) = \min_v \{ c_{x,v} + D_v(y) \}$$

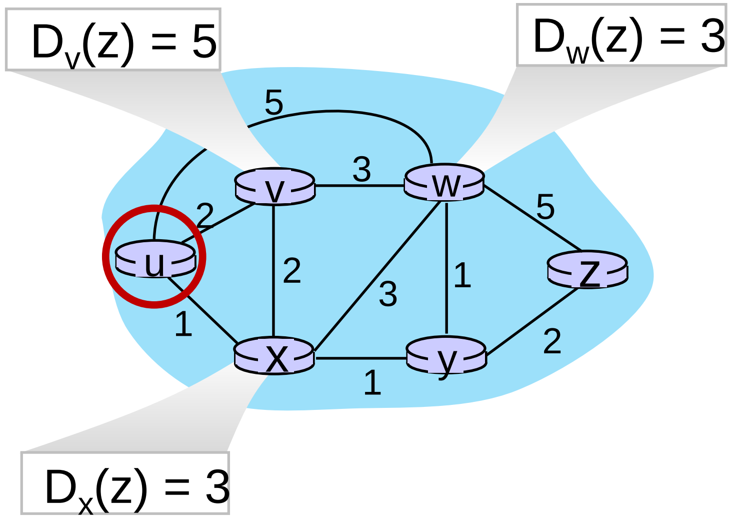
v's estimated least-cost-path cost to y

min taken over all neighbors v of x

direct cost of link from x to v

Bellman-Ford Example

Suppose that u's neighboring nodes, x,v,w, know that for destination z:



Bellman-Ford equation says:

$$D_u(z) = \min \{ c_{u,v} + D_v(z), \; c_{u,x} + D_x(z), \; c_{u,w} + D_w(z) \}$$

$$= \min \{2 + 5, \; 1 + 3, \; 5 + 3\} = 4$$

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 any 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

Each node:

wait for (change in local link cost or msg from neighbor)

↓

recompute DV estimates using DV received from neighbor

↓

if DV to any destination changes, send new DV to the neighbors, else do nothing

Iterative, asynchronous: each local iteration caused by:

- Local link cost change
- DV update message from neighbor

Distributed, self-stopping: each node notifies neighbors only when its DV changes

- Neighbors then notify their neighbors – only if necessary
- No notification received, no actions taken!

Node x table

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

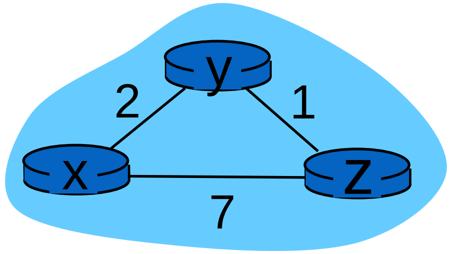
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$
$= \min\{2+1\ ,\ 7+0\} = 3$

$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$
$= \min\{2+0\ ,\ 7+1\} = 2$

Node y table

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



Node z table

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

Node x table

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

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

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

Node y table

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

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

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

Node z table

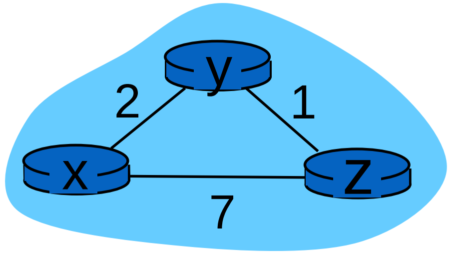
from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

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

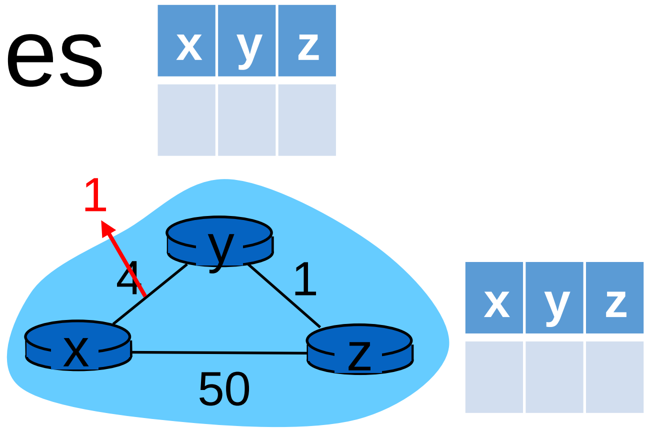
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$
$= \min\{2+1,\ 7+0\} = 3$

$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$
$= \min\{2+0,\ 7+1\} = 2$



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

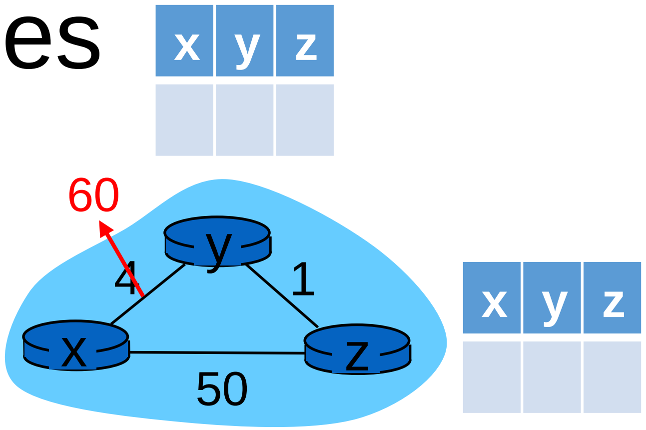
"Good news travels fast"

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.

Distance Vector: Link Cost Changes



Link cost changes:

- 44 iterations before algorithm stabilizes
- Bad news travels slow - "count to infinity" problem!

- y sees direct link to x has new cost 60, but z has said it has a path at cost of 5. So y computes "my new cost to x will be 6, via z; notifies z of new cost of 6 to x.
- z learns that path to x via y has new cost 6, so z computes "my new cost to x will be 7 via y), notifies y of new cost of 7 to x.
- y learns that path to x via z has new cost 7, so y computes "my new cost to x will be 8 via z), notifies z of new cost of 8 to x.
- z learns that path to x via y has new cost 8, so z computes "my new cost to x will be 9 via y), notifies y of new cost of 9 to x.

Distance Vector: Link Cost Changes

Link cost changes:

- Bad news travels slow - “count to infinity” problem!

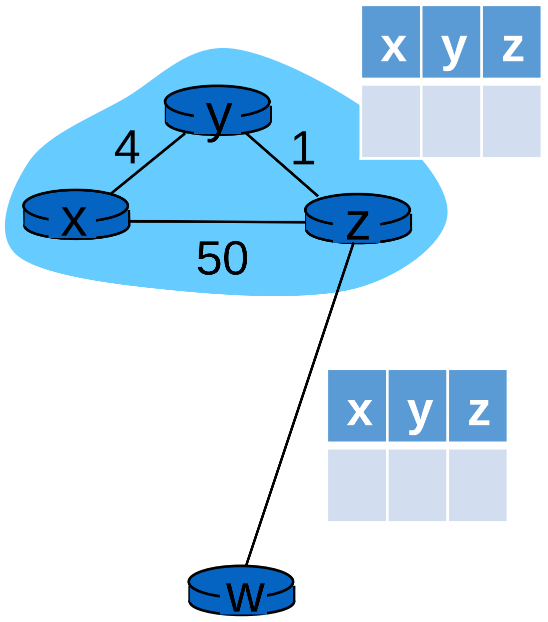
A	B	C	D	E	
	1	2	3	4	Initially
	3	2	3	4	After 1 exchange
	3	4	3	4	After 2 exchanges
	5	4	5	4	After 3 exchanges
	5	6	5	6	After 4 exchanges
	7	6	7	6	After 5 exchanges
	7	8	7	8	After 6 exchanges
		⋮			
	•	•	•	•	

Distance Vector: Link Cost Changes

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?
  - Do not work for loops with more than two nodes

- Other solutions:
  - Limit the maximum cost
  - Variation of DV: Path vector routing
    - Nodes send not only the cost, but also the entire path to the destination
    - More overhead



Comparison of LS and DV algorithms

Message complexity

LS: n routers, $O(n^2)$ messages sent

DV: exchange between neighbors; convergence time varies

Speed of convergence

LS: $O(n^2)$ algorithm, $O(n^2)$ messages

- may have oscillations

DV: convergence time varies

- May have routing loops
- Count-to-infinity problem

Robustness: what happens if router malfunctions, or is compromised?

LS:

- Router can advertise incorrect link cost
- Each router computes only its own table

DV:

- DV router can advertise incorrect path cost ("I have a really low cost path to everywhere"): black-holing
- Each router's table used by others: error propagate thru network

Summary

- Distance vector routing algorithm:
  - Limitations: count-to-infinity problem
  - Comparison of LS and DV