

CN-Advanced L32

Distance Vector Routing

Dr. Ram P Rustagi
rprustagi@ksit.edu.in
<http://www.rprustagi.com>
<https://www.youtube.com/rprustagi>

Distance Vector Routing

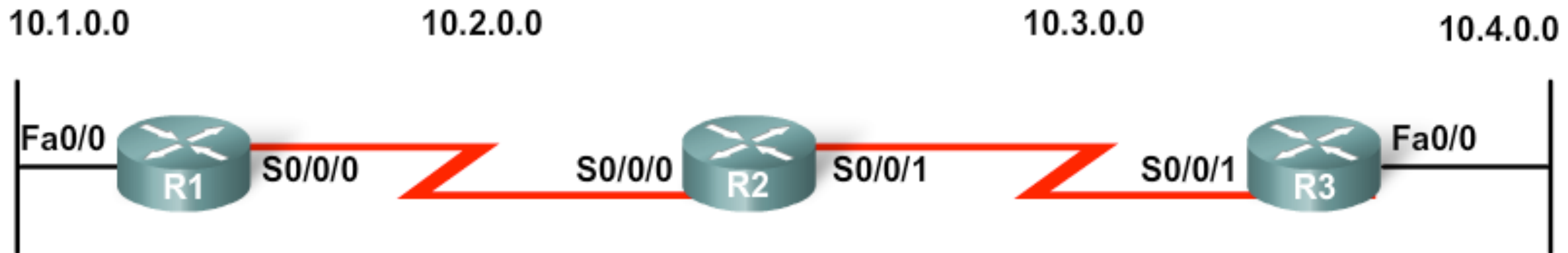
- Assumption
 - Each node knows cost of its directly connected link
 - A down (or non-existent) link cost is taken as infinity
- Distributed
 - Each node receives info from neighbors
 - Computes routing and distributes to neighbors
 - No central computation
- Iterative
 - Routing computation stops when no more info exchange
 - It is self terminating - no external control
- Asynchronous
 - Each node does not work in sync with others

Distance Vector Routing

- Does not allow a router to know exact topology
- Uses router as signposts along the path to dstⁿ
- Sends periodic updates
- Core of the DV protocol
 - Bellman Ford Algorithm
- Works best in following type of situations
 - Network is simple and flat
 - Does not require hierarchical design
 - Administrators do not have enough knowledge
 - Configure/troubleshoot link state protocols
 - Worst case **convergence** time is not a concern

DV Routing - Example

Initial Network



Network	Interface	Hop
10.1.0.0	Fa0/0	0
10.2.0.0	S0/0/0	0

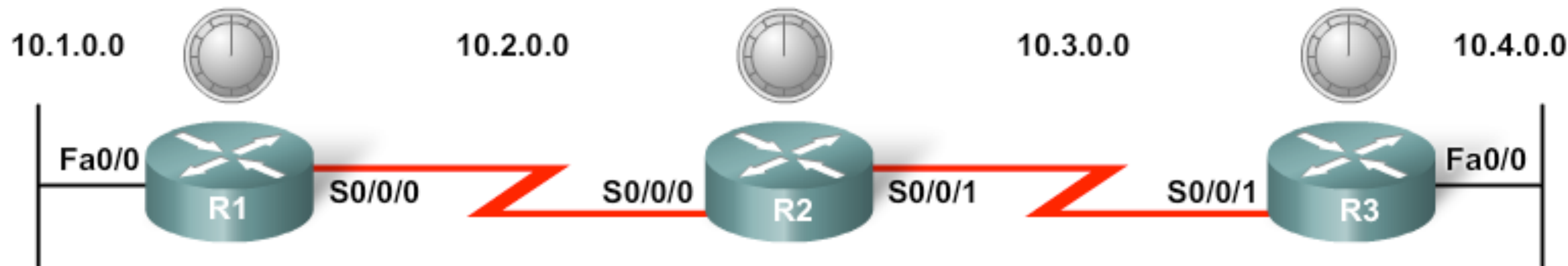
Network	Interface	Hop
10.2.0.0	S0/0/0	0
10.3.0.0	S0/0/1	0

Network	Interface	Hop
10.3.0.0	S0/0/0	0
10.4.0.0	Fa0/0	0

Src: CCNA Module 2

DV Routing - Example

- After the exchange of routing packets



Network	Interface	Hop
10.1.0.0	Fa0/0	0
10.2.0.0	S0/0/0	0
10.3.0.0	S0/0/0	1
10.4.0.0	S0/0/0	2

Network	Interface	Hop
10.2.0.0	S0/0/0	0
10.3.0.0	S0/0/1	0
10.1.0.0	S0/0/0	1
10.4.0.0	S0/0/1	1

Network	Interface	Hop
10.3.0.0	S0/0/1	0
10.4.0.0	Fa0/0	0
10.2.0.0	S0/0/1	1
10.1.0.0	S0/0/1	2

Src: CCNA Module 2

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let

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

then

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

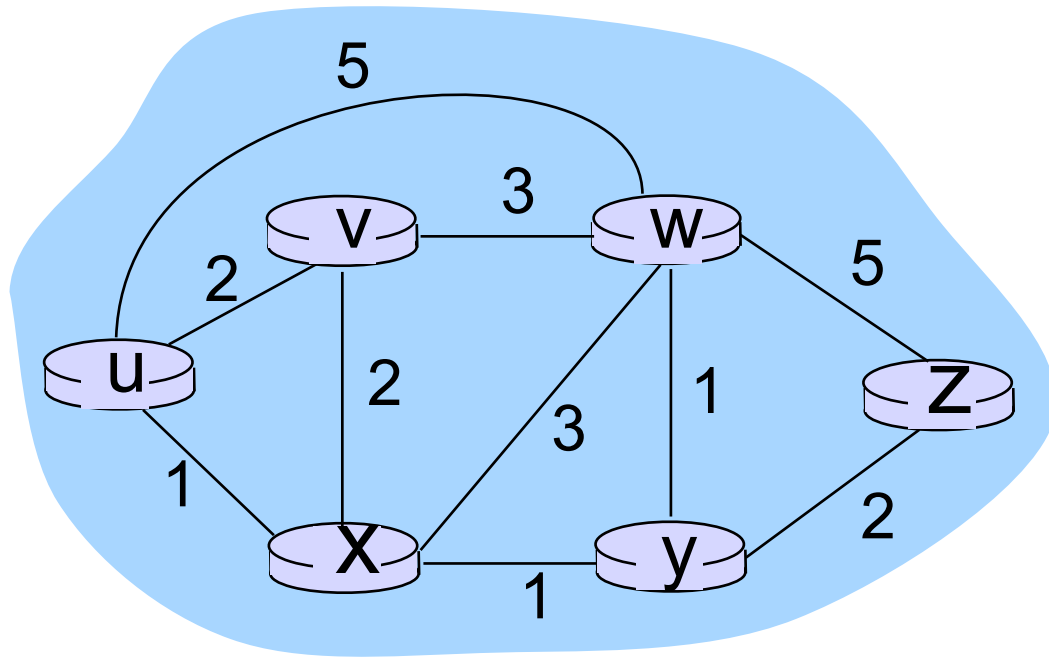
cost from neighbor v to destination y

cost to neighbor v

\min taken over all neighbors v of x

Bellman-Ford example

Compute $D_u(z)$:



node achieving minimum is
next hop in shortest path,
used in forwarding table

For u's neighbors: v, x, w, we have

$$d_v(z) = ?, d_x(z) = ?, d_w(z) = ?$$

$$d_v(z) = 5, d_x(z) = 3, d_w(z) = 3$$

B-F 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

- $D_x(y)$ = estimate of least cost from x to y
 - actual least cost from x to y is represented as $d_x(y)$
 - node x maintains distance vector for all nodes
 - $D_x = [D_x(y): y \in N]$
- Node x:
 - knows cost to each neighbor v: $c(x,v)$
 - maintains its neighbors' distance vectors.
 - For each neighbor v, it maintains $D_v = [D_v(y): y \in N]$

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 v , it updates its own DV using B-F equation:

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

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

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:

initialize $D_x(y) = c(x,y)$
 $= \infty$, when no edge (x,y)

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

recompute estimates

if DV to any dest has
changed, *notify*
neighbors

DV Algorithm

/* Initialisation - for each node x */

for all destinations y in N :

$D_x(y) = c(x,y)$ /* $c(x,y) = \infty$ for non neighbour */

for each neighbor w

$D_w(y) = ?$ for all destinations y in N

for each neighbour w

send distance vector $D_x = [D_x(y): y \text{ in } N]$ to w

DV Algorithm

Loop /* for each node x */

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

for each y in N:

$D_x(y) = \min_v \{c(x,v) + D_v(y)\}$ #for neighbours v

if $D_x(y)$ changed for any destination y

send $D_x = [D_x(y): y \text{ in } N]$ to all neighbour

Forever

**node x
table**

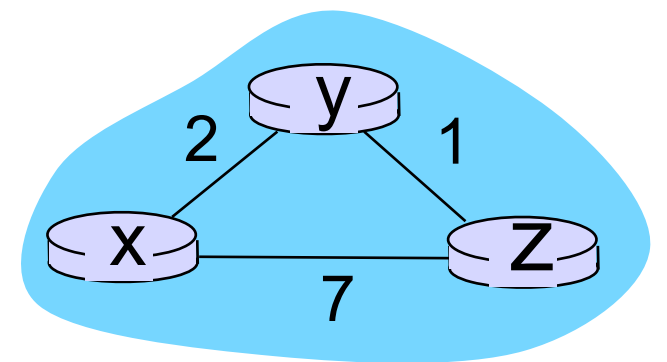
		cost to		
		x	y	z
from	x	0	2	7
	y	∞	∞	∞
	z	∞	∞	∞

**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



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	∞	∞	∞
	z	∞	∞	∞

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

**node y
table**

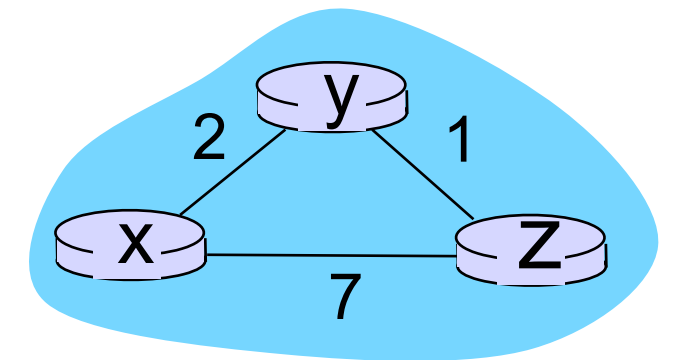
		cost to		
		x	y	z
from	x	∞	∞	∞
	y	2	0	1
	z	∞	∞	∞

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

**node z
table**

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

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



time

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

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

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

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

**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	∞	∞	∞

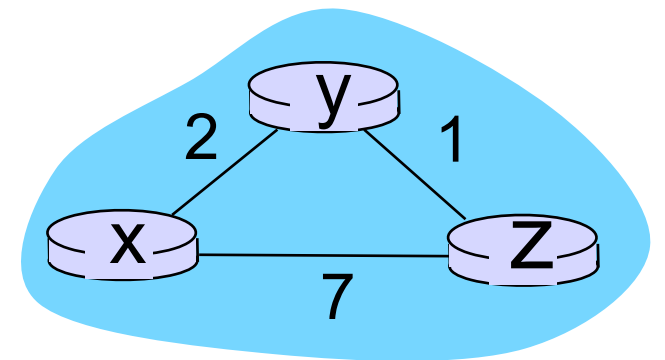
**node z
table**

		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			0



time

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

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

**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	∞	∞	∞

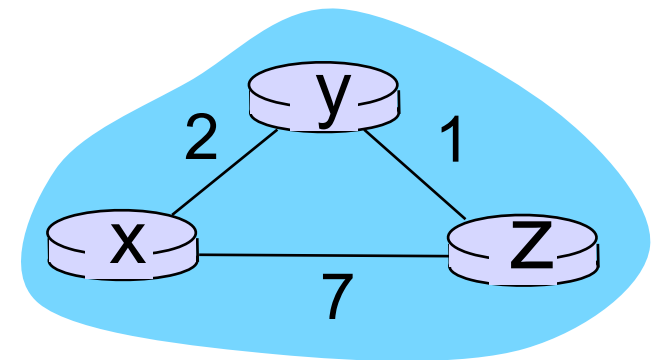
**node z
table**

		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



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	∞	∞	∞
	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

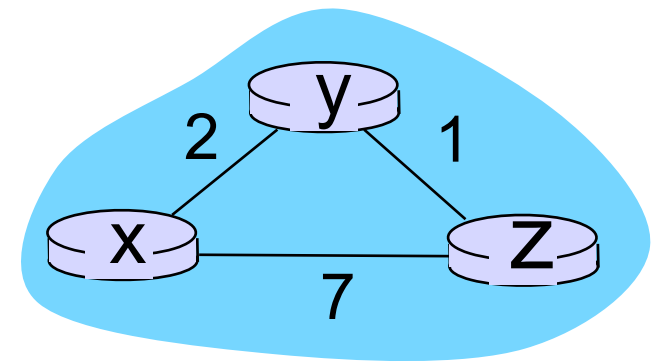
		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

		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

DV Routing

- Issues
 - Routing Loop
 - Count to Infinity
 - Slow Convergence
 - link cost changes (increases)

Summary

- Each node knows only its neighbours
- Each node knows cost to all nodes
 - A node does not know the topology of the network
- A node receives cost of all nodes from its neighbours