The Network Layer

lec 20-3 The Distance-Vector Routing

School of Software Engineering
South China University of Technology
Dr. Chunhua Chen
chunhuachen@scut.edu.cn
2020 Spring

The Distance-Vector (DV) Routing Algorithm

- Distributed
- each node receives some information from one or more of its *directly attached* neighbors, performs a calculation, and then distributes the results of its calculation back to its neighbors
- Iterative
  - this process continues on until no more information is exchanged between neighbors
- Asynchronous
  - it does not require all of the nodes to operate in lockstep with each other.

Bellman-Ford equation

$d_x(y)$ be the cost of the least-cost path from node x to node y.

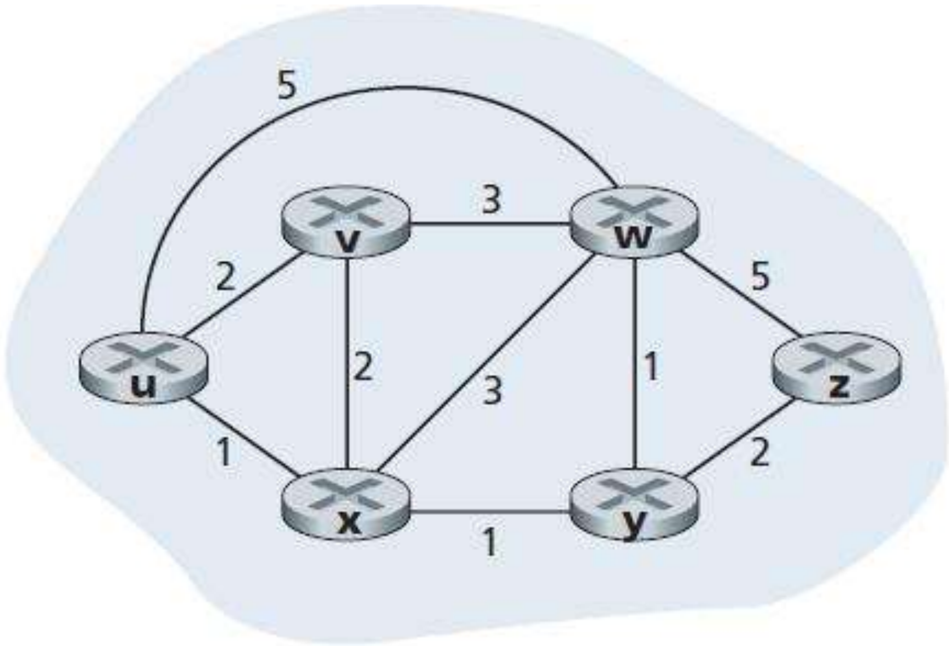
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

where the min_v in the equation is taken over all of x's neighbors

$d_u(z)$: (u, x) (x, y) (y, z) = 4

Neighbor of u: v, w and x

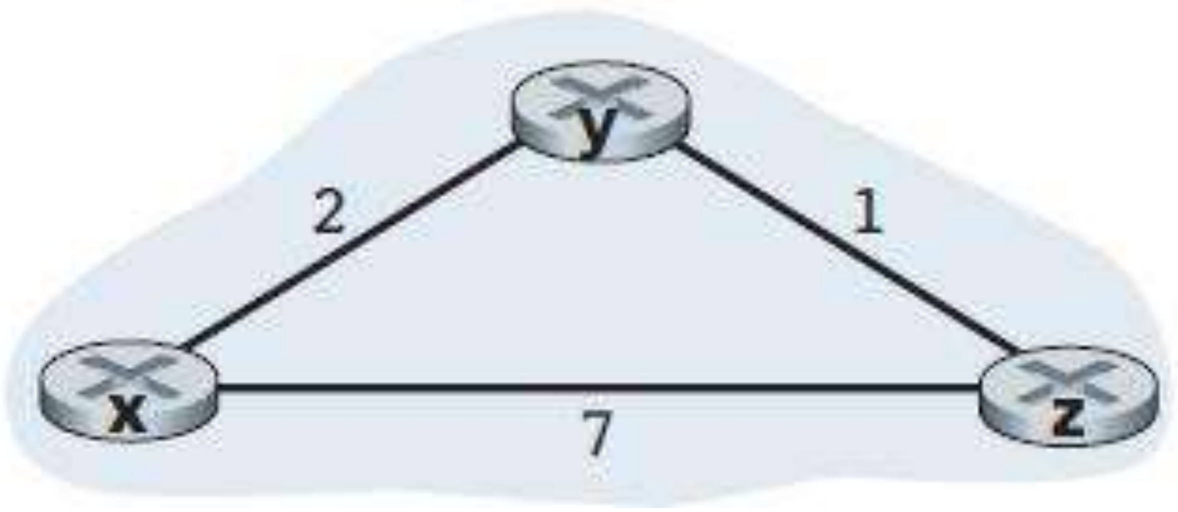
1. $c(u,v) + d_v(z)$
2. $c(u,w) + d_w(z)$
3. $c(u,x) + d_x(z)$



The DV Algorithm

- Each node x begins with $D_x(y)$, an estimate of the cost of the least-cost path from x to node y, for all nodes in N.
- $\boldsymbol{D}_x = [D_x(y): y \text{ in } N]$ be node x's distance vector, which is the vector of cost estimates from x to all other nodes, y, in N.

$$\boldsymbol{D}_z = [D_z(x), D_z(y), D_z(z)]$$
$$= [7, \quad 1, \quad 0]$$



The DV Algorithm

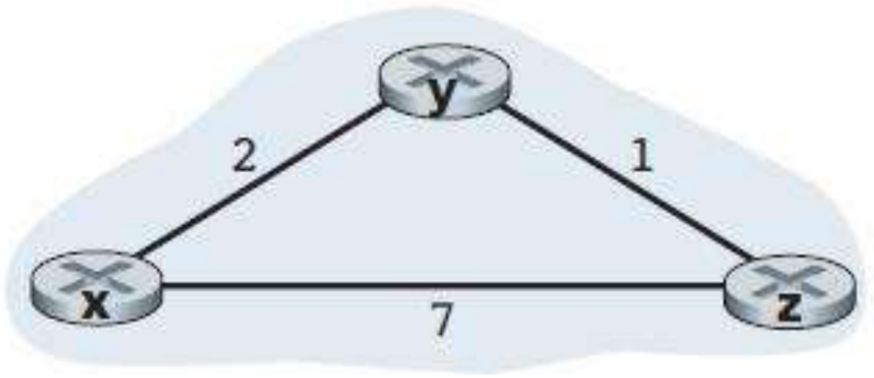
- With the DV algorithm, each node *x* maintains the following routing information:

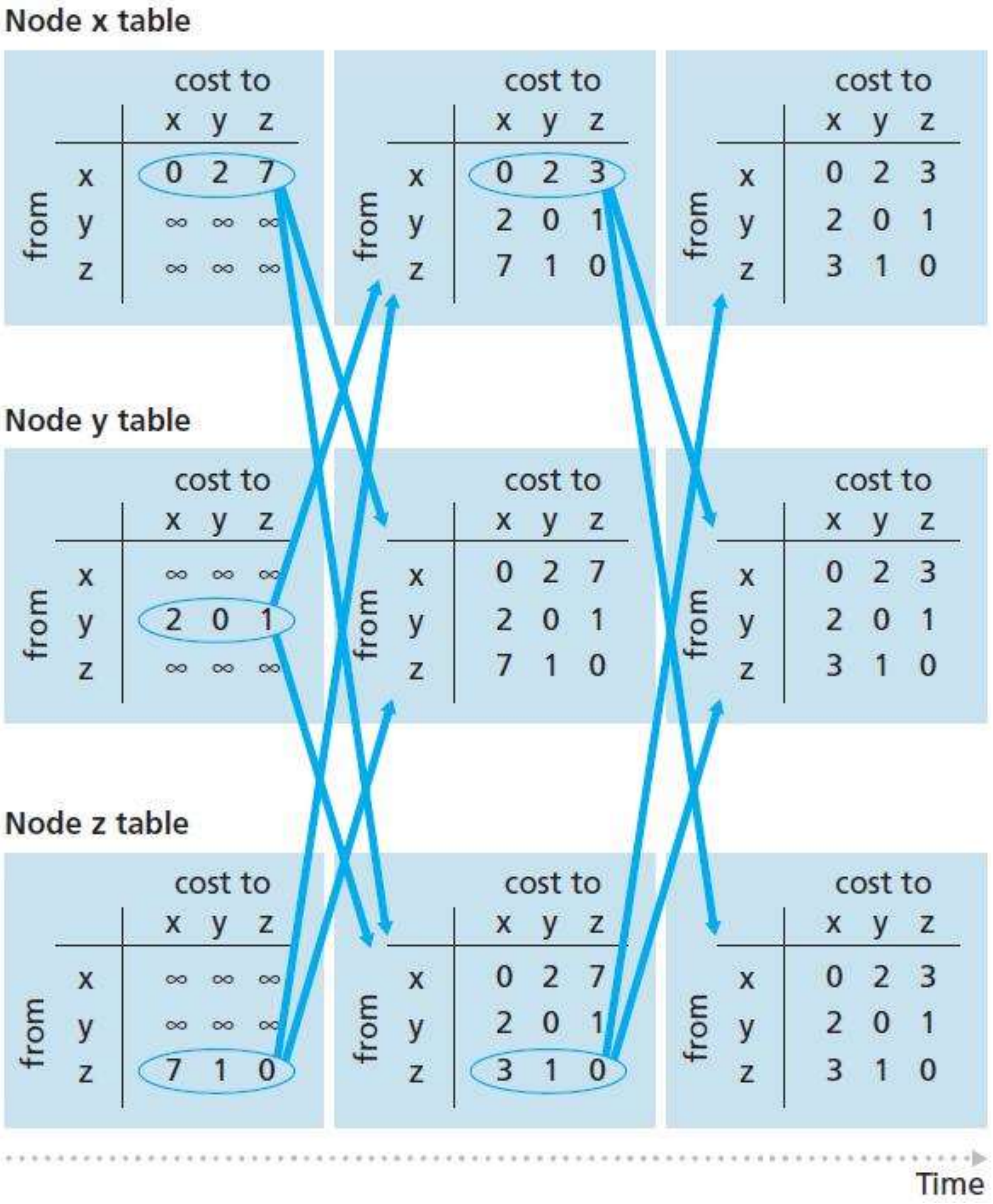
- For each neighbor v, the cost $c(x,v)$ from x to directly attached neighbor, v
- Node x's distance vector, that is, $\boldsymbol{D}_x = [D_x(y)\text{: } y \text{ in } N]$, containing x's estimate of its cost to all destinations, y, in N
- The distance vectors of each of its neighbors, that is, $\boldsymbol{D}_v = [D_v(y)\text{: } y \text{ in } N]$ for each neighbor v of x

The DV Algorithm run by node x

```
Initialization:
   for all destinations y in N:
      D_x(y) = c(x,y)    /* if y is not a neighbor then c(x,y) = ∞ */
   for each neighbor w
      D_w(y) = ? for all destinations y in N
   for each neighbor w
      send distance vector D_x = [D_x(y): y in N] to w

loop
   wait (until I see a link cost change to some neighbor w or
         until I receive a distance vector from some neighbor w)

   for each y in N:
      D_x(y) = min_v{c(x,v) + D_v(y)}

   if D_x(y) changed for any destination y
      send distance vector D_x = [D_x(y): y in N] to all neighbors

forever
```

The DV algorithm is *decentralized* and does not use such global information. Indeed, the only information a node will have is the costs of the links to its directly attached neighbors and information it receives from these neighbors.





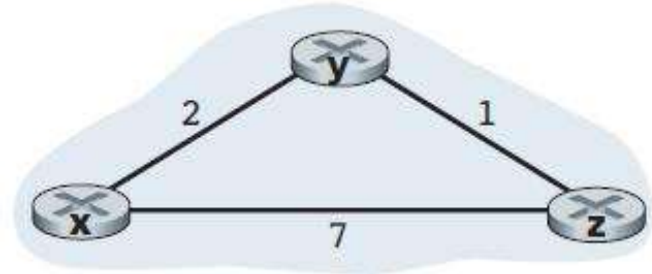
At node x, after receiving updates

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



$$D_x(x) = 0$$

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

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