

Lecture 17: Network Layer

– Control Plane I

Sejong University Spring 2019: Computer Networks

2019. 5. 9.

Cheol Jeong

This material can only be used for students that signed up for this class at Sejong University and must not be distributed outside of the class. The contents are mainly based on the text book, “Computer Networking: A Top-Down Approach” by J. F. Kurose and K. W. Ross (7th Edition).

Contents of Chapter 5

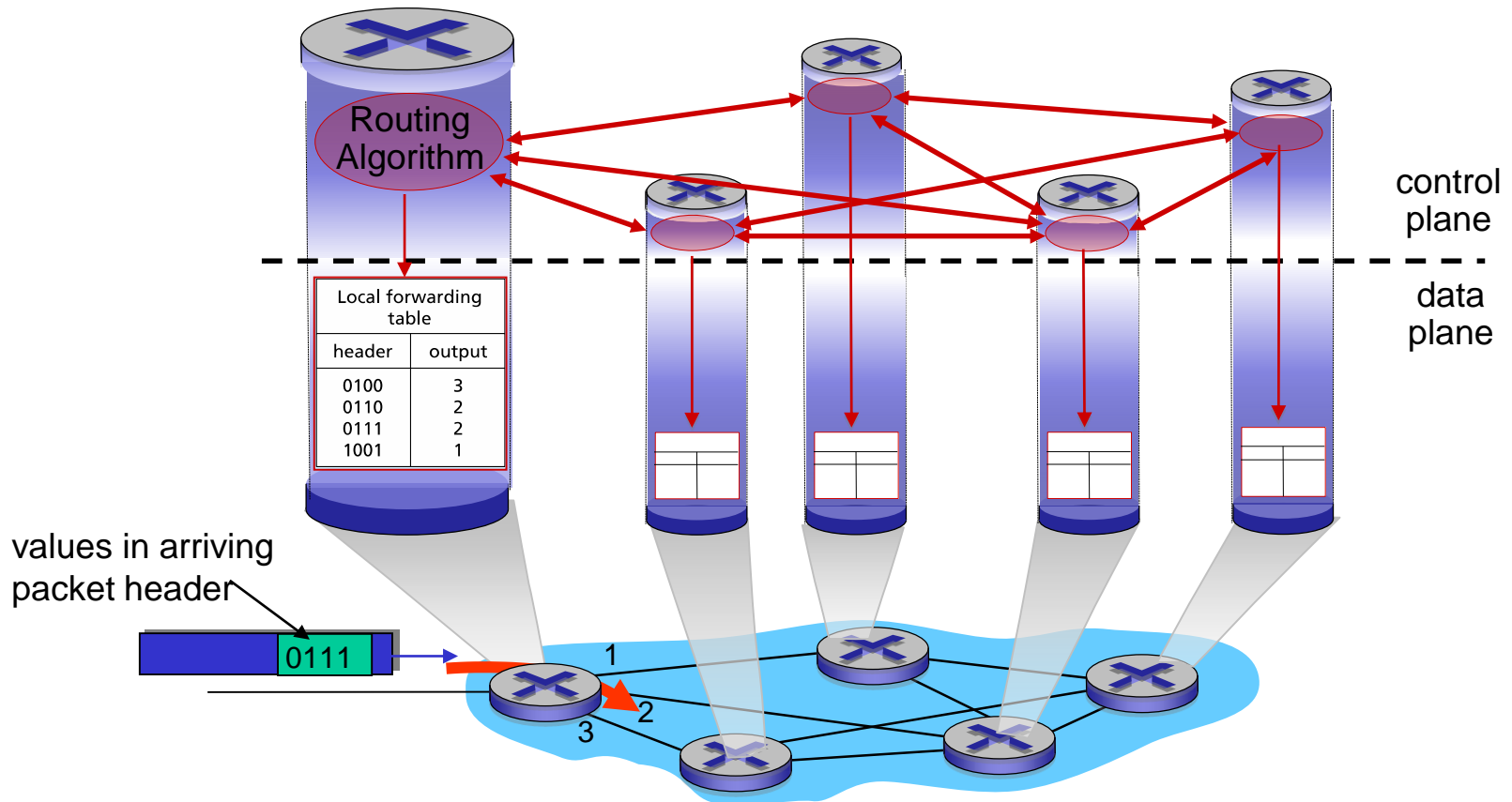
- ◇ Introduction
- ◇ Routing algorithms
- ◇ Intra-AS routing in the Internet: OSPF
- ◇ Routing among the ISPs: BGP
- ◇ The SDN control plane
- ◇ ICMP: The internet control message protocol
- ◇ Network management and SNMP

Contents of Chapter 5

- ◇ **Introduction**
- ◇ **Routing algorithms**
 - ◆ The link-state (LS) routing algorithm
 - ◆ The distance-vector (DV) routing algorithm
- ◇ Intra-AS routing in the Internet: OSPF
- ◇ Routing among the ISPs: BGP
- ◇ The SDN control plane
- ◇ ICMP: The internet control message protocol
- ◇ Network management and SNMP

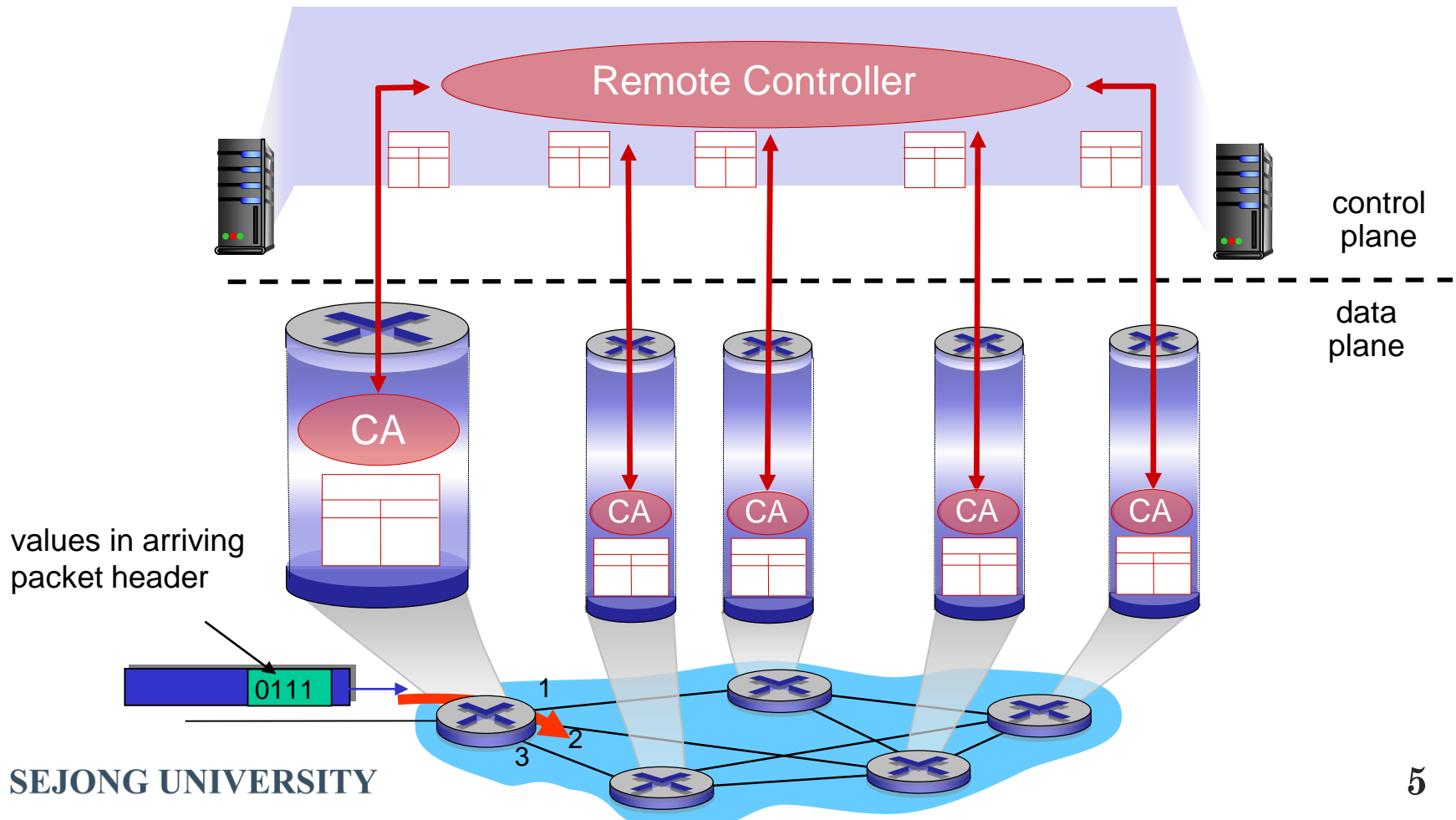
Control Plane: The Traditional Approach

- Routing algorithms determine values in forwarding tables.



Control Plane: The SDN Approach

- ◆ A remote controller determines and distributes values in forwarding tables: Software-defined networking (SDN)



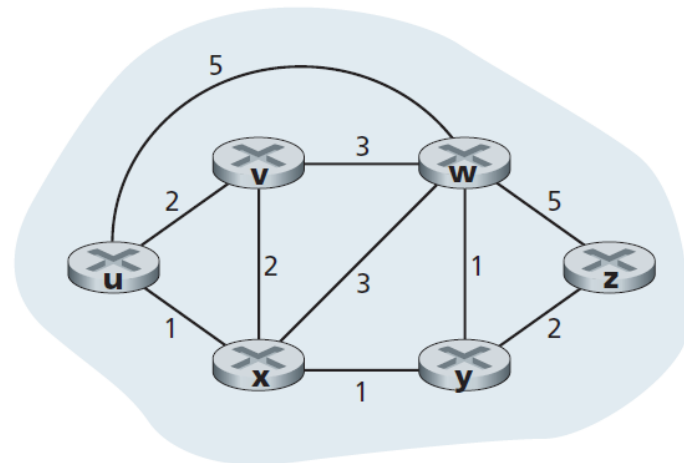
Routing Algorithms

◇ Routing algorithm

- ◆ Determine good paths from senders to receivers, through the network of routers.

◇ Abstract graph model of a computer network

- ◆ Graph $G = (N, E)$ with a set N of nodes and a set of E of edges
- ◆ $c(x, y)$: The cost of the edge between nodes x and y
 - ◆ If the pair (x, y) does not belong to E , we set $c(x, y) = \infty$
- ◆ (x_1, x_2, \dots, x_p) : A path such that $(x_i, x_{i+1}) \in E, i = 1, 2, \dots, p-1$
- ◆
$$c(x_1, x_2, \dots, x_p) = \sum_{i=1}^{p-1} c(x_i, x_{i+1})$$



Routing Algorithms

◆ Centralized routing algorithm

- ◆ Use complete, global knowledge about the connectivity between all nodes and all link costs
- ◆ Link-state (LS) algorithm

◆ Decentralized routing algorithm

- ◆ Each node begins with only the knowledge of the costs of its neighboring nodes.
- ◆ Gradually calculate the least-cost path through an iterative process of calculation and exchange of information with its neighboring nodes
- ◆ Distance-vector (DV) algorithm

Other Classification of Routing Algorithms

◇ **Static vs dynamic**

- ◆ Static: Routes change very slowly over time.
- ◆ Dynamic: Routes change as the network traffic loads or topology change.

◇ **Load-sensitive vs load-insensitive**

- ◆ Load-sensitive: Link costs vary dynamically to reflect the current level of congestion.
- ◆ Load-insensitive: A link's cost does not explicitly reflect its current level of congestion.

Link-State (LS) Routing Algorithm

◆ Notation

$D(v)$: cost of the least-cost path from the source node to destination v as of this iteration of the algorithm.

$p(v)$: previous node (neighbor of v) along the current least-cost path from the source to v .

N' : subset of nodes; v is in N' if the least-cost path from the source to v is definitively known.

Link-State (LS) Routing Algorithm

◇ Pseudocode

```
1  Initialization:
2     $N' = \{u\}$ 
3    for all nodes  $v$ 
4      if  $v$  is a neighbor of  $u$ 
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 each neighbor  $v$  of  $w$  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   least path cost to  $w$  plus cost from  $w$  to  $v$  */
15 until  $N' = N$ 
```



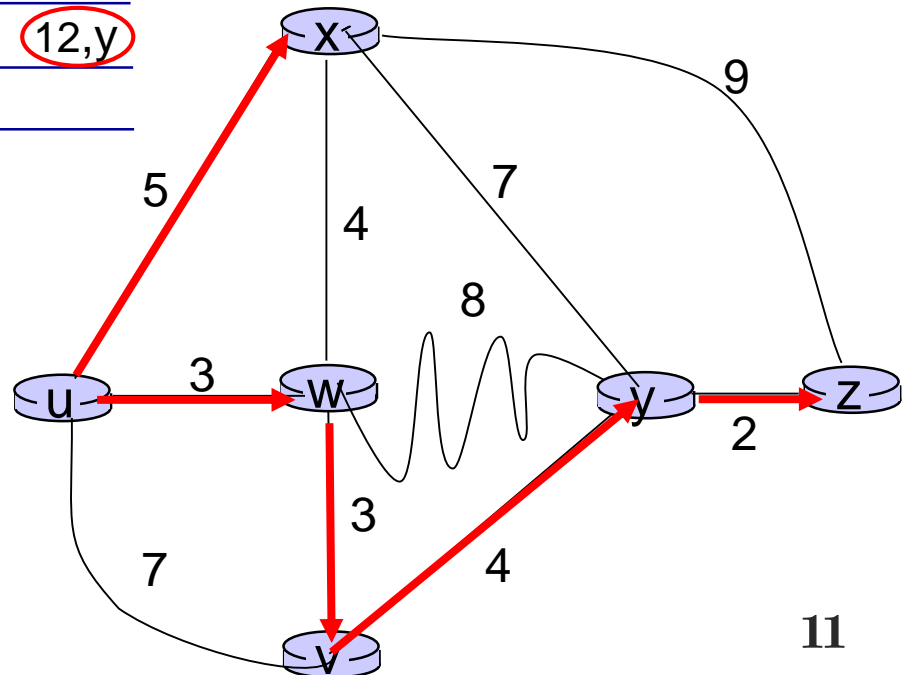
Link-State (LS) Routing Algorithm

◇ Dijkstra's algorithm

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

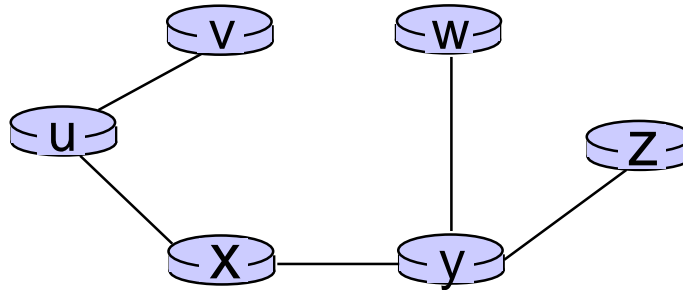
Notes:

- ❖ Construct shortest path tree by tracing predecessor nodes
- ❖ Ties can exist (can be broken arbitrarily)



Link-State (LS) Routing Algorithm

- Resulting shortest-path tree from u .



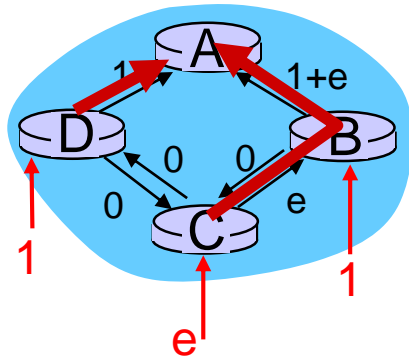
- Resulting forwarding table in u .

Destination	Link
v	(u, v)
w	(u, x)
x	(u, x)
y	(u, x)
z	(u, x)

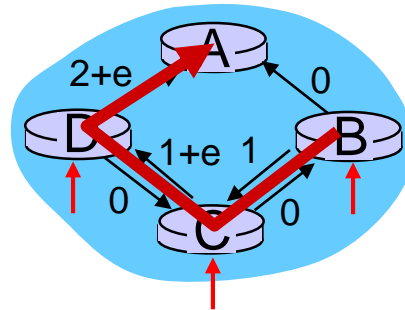
Link-State (LS) Routing Algorithm

◇ Oscillations with congestion-sensitive routing

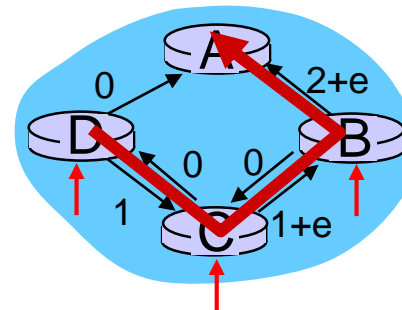
- ◆ Suppose that link costs are equal to the load carried on the link.



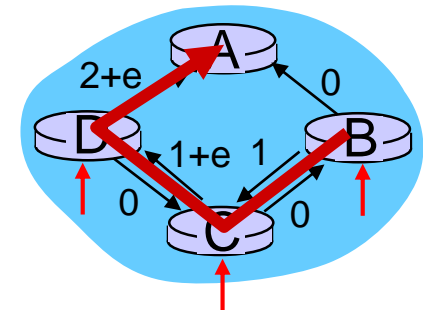
a. initial routing



b. x, y detect better path to w, clockwise



c. x, y, z detect better path to w, counterclockwise



d. x, y, z detect better path to w, clockwise

Distance-Vector (DV) Routing Algorithm

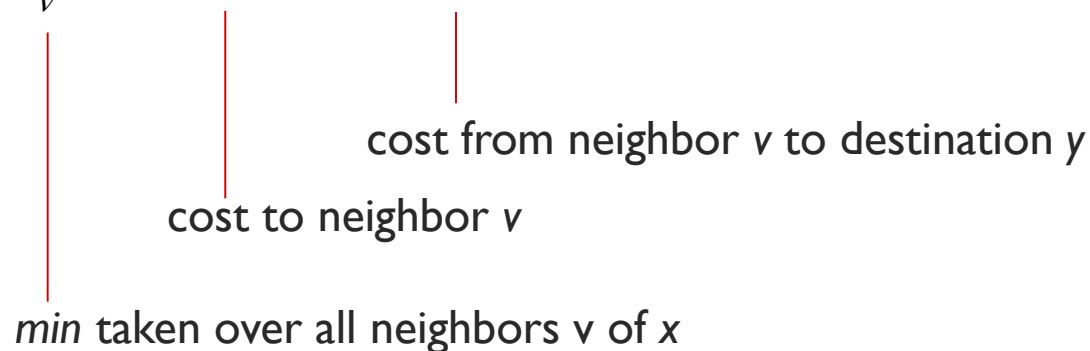
◇ Distance-vector (DV) algorithm

- ◆ Iterative, asynchronous, and distributed

◇ Bellman-Ford equation

- ◆ The cost of the least-cost path from x to y , $d_x(y)$

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



 \min_v min taken over all neighbors v of x

 $c(x, v)$ cost to neighbor v

 $d_v(y)$ cost from neighbor v to destination y

Distance-Vector (DV) Routing 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, $\mathbf{D}_x = [D_x(y): y \text{ in } N]$, containing x 's estimate of its cost to all destinations, y , in N
 - The distance vectors of each of its neighbors, that is, $\mathbf{D}_v = [D_v(y): y \text{ in } N]$ for each neighbor v of x

$D_x(y)$: An estimate of the cost of the least-cost path from x to node y

- ◆ **Node's action**
 - ◆ Compute their new distance vectors
$$D_x(y) = \min_v \{c(x,v) + D_v(y)\}$$
 - ◆ Update their forwarding table
 - ◆ Inform their neighbors if their distance vectors have changed

Distance-Vector (DV) Routing Algorithm

**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

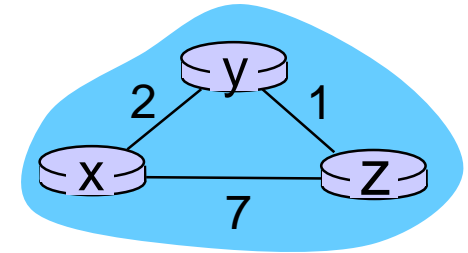
		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	7	1	0

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

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

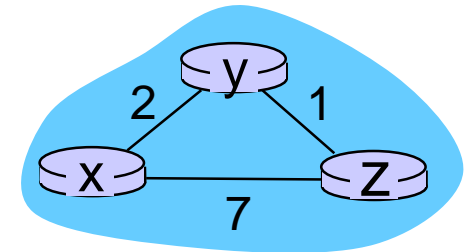
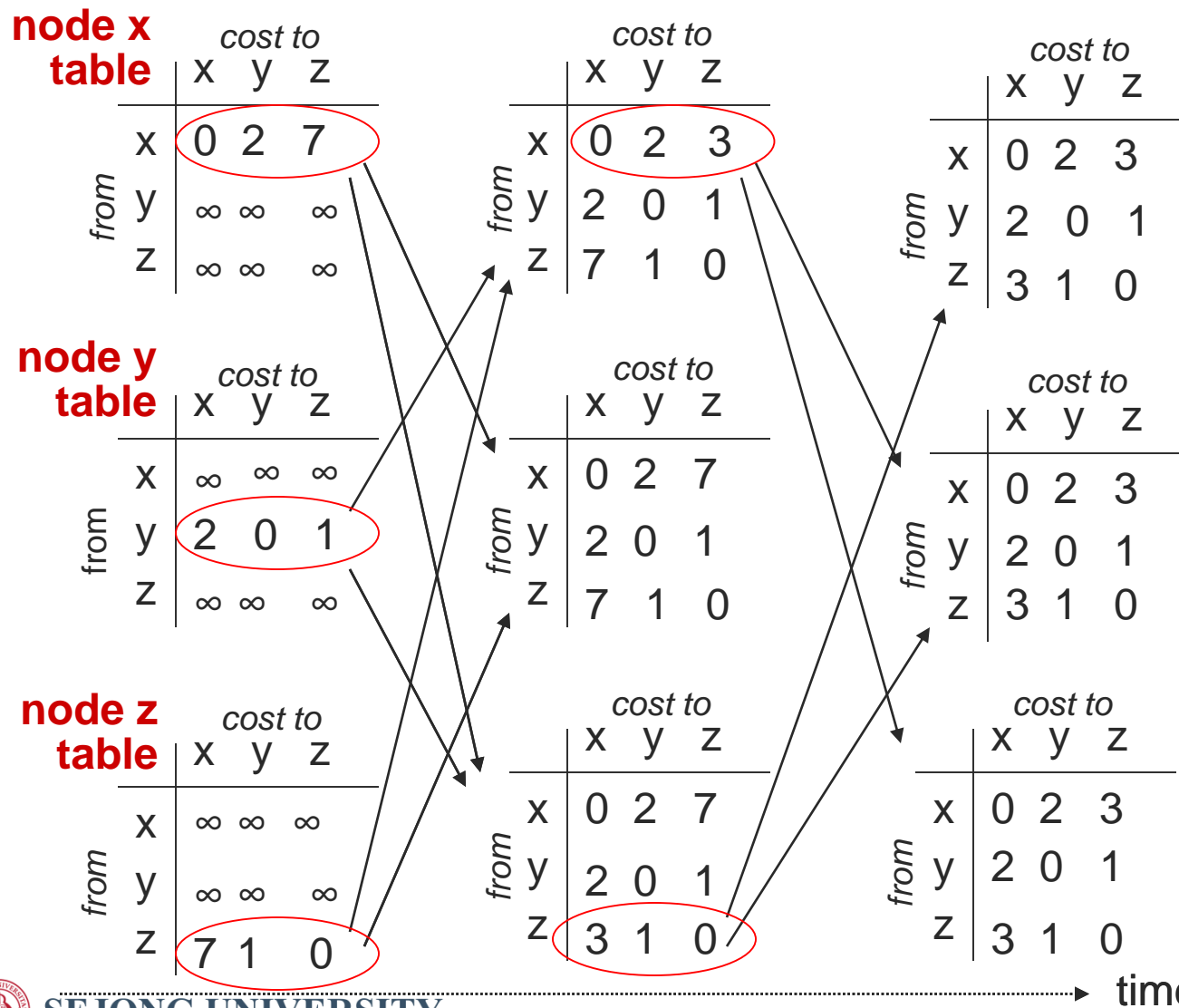
$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$$

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



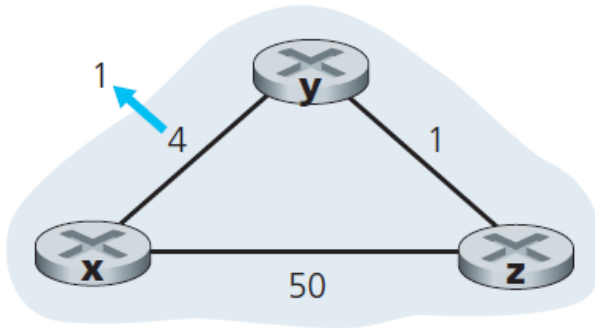
time

Distance-Vector (DV) Routing Algorithm

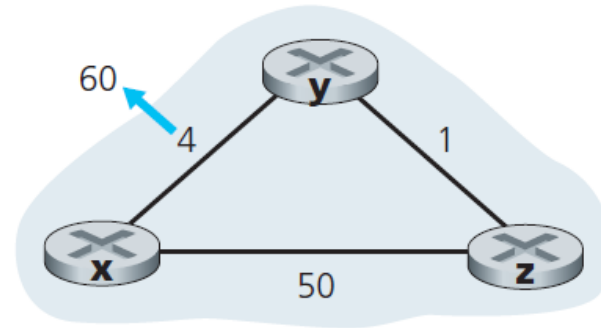


Distance-Vector (DV) Routing Algorithm

◇ Link-cost changes and link failure



a.



b.