Chap 4 Homework 2 R21-23 P20-25, 27

Zilong Wang 18281218 CS1804 wangzilong@bjtu.edu.cn

R21. Is it necessary that every autonomous system use the same intra-AS routing algorithm? Why or why not?

No. it is not necessary that every autonomous system use the same intra-AS routing algorithm. Because each autonomous system routing has administrative autonomy for routing within an AS.

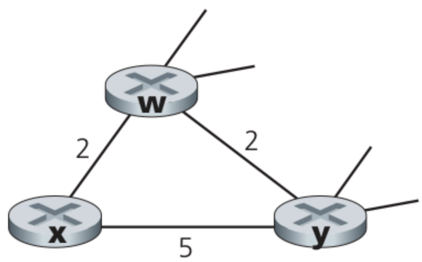
R22. Compare and contrast link-state and distance-vector routing algorithms.

Link state algorithms: Computes the least-cost path between source and destination using complete, global knowledge about the network. Distance-vector routing: The calculation of the least-cost path is carried out in an iterative, distributed manner. A node only knows the neighbor to which it should forward a packet in order to reach given destination along the least-cost path, and the cost of that path from itself to the destination.

R23. Discuss how a hierarchical organization of the Internet has made it possible to scale to millions of users.

Routers are organized into autonomous systems (ASs). Within an AS, all routers run the same intra-AS routing protocol. The problem of scale is solved since an router in an AS need only know about routers within its AS and the subnets that attach to the AS. To route across ASs, the inter-AS protocol is based on the AS graph and does not take individual routers into account.

P20. Consider the network fragment shown below. x has only two attached neighbors, w and y. w has a minimum-cost path to destination *u* (not shown) of 5, and *y* has a minimum-cost path to *u* of 6. The complete paths from *w* and *y* to *u* (and between *w* and *y*) are not shown. All link costs in the network have strictly positive integer values.



a. Give *x*'s distance vector for destinations *w*, *y*, and *u*.
b. Give a link-cost change for either *c(x, w)* or *c(x, y)* such that *x* will inform its neighbors of a new minimum-cost path to *u* as a result of executing the distance-vector algorithm.

c. Give a link-cost change for either *c(x, w)* or *c(x, y)* such that *x* will not inform its neighbors of a new minimum-cost path to *u* as a result of executing the distance-vector algorithm.

a)

Consider the diagram:

Minimum cost path from node w to node u= 5.

Minimum cost path from node w to node y =6.

Distance-vectors from node x are as follows:

Vx(w)=2

Vx(y)=5

Vx(u)=

Considering the neighbors of node y and node w from x in the first iteration completed

Vx(w)=2

Vx(y)=4

Vx(u)=7

b)

Give a link-cost change for either c(x,w) or c(x,y) such that x will inform its neighbors of a new minimum-cost path to u.

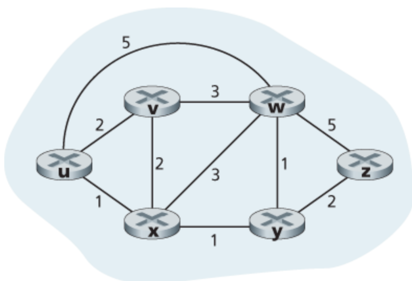
The result of executing the distance-vector algorithm is node x again informs its neighbors of the new cost.

c)

Give a link-cost change for either c(x,w) or c(x,y) such that x will not inform its neighbors of a new minimum-cost path to u.

The result of executing the distance-vector algorithm is not cause x to inform its neighbors of a new minimum-cost path to u.

P21. Looking at Figure 4.27, enumerate the paths from *v* to *y* that do not contain any loops.



v-w-y	*v-w-z-y*	*v-x-w-z-y*
v-x-y	*v-u-x-y*	*v-w-u-x-y*
v-w-x-y	*v-u-w-z-y*	*v-u-x-w-z-y*
v-x-w-y	*v-u-w-x-y*	
v-y-w-y	*v-u-x-w-y*	

P22. Repeat problem P21 for paths from *x* to *w*, *w* to *u*, and *z* to *x*

***x* to *w*:**

x-w　　*x-u-v-w*　　*x-y-w*

x-v-w　　*x-u-w*　　*x-y-z-w*

***w* to *u*:**

w-u　　*w-v-u*　　*w-x-u*

w-x-v-u　　*w-y-x-u*　　*w-y-x-v-u*

w-z-y-x-u　　*w-z-y-x-v-u*

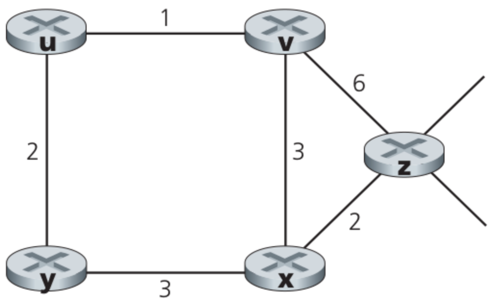
***z* to *x*:**

z-y-x　　*z-w-x*　　*z-w-y-x*

z-y-w-x　　*z-w-v-x*　　*z-w-u-x*

z-y-w-v-x　　*z-w-y-v-x*　　*z-y-w-v-u-x*

P23. Consider the network shown below, and assume that each node initially knows the costs to each of its neighbors. Consider the distance-vector algorithm and shown the distance table entries at node *z*.



Distance vector routing algorithm exchanges the information with the neighbors and works asynchronously.
According to the distance vector algorithm, any node m computes the distance vector using the following formulas:

$$D_m(m) = 0$$
$$D_m(n) = \min\{c(m,n) + D_n(n), c(m,n) + D_o(n)\}$$
$$D_m(o) = \min\{c(m,n) + D_n(o), c(m,o) + D_o(o)\}$$

Note: NA is used when there is no distance value.

Construct the distance vector table for node z from the network diagram:

	u	v	x	y	z
v	NA	NA	NA	NA	NA
x	NA	NA	NA	NA	NA
z	NA	6	2	NA	0

Now update the table with costs of all the neighboring nodes.

	u	v	x	y	z
v	1	0	3	NA	6
x	NA	3	0	3	2
z	NA	6	2	NA	0

Update the table with minimum costs using the distance vector routing algorithm:
Example: v to y, two paths are available. v-u-y and v-x-y with costs 3 and 6 respectively. So, v-u-y is the path with minimum cost. Hence update the table with this value.

	u	v	x	y	z
v	1	0	3	3	5
x	4	3	0	3	2
z	6	5	2	5	0

Therefore, at node z, the above table will be computed by the distance vector routing algorithm.

P24. Consider a general topology (that is, not the specific network shown above) and a synchronous version of the distance-vector algorithm. Suppose that at each iteration, a node exchanges its distance vectors with its neighbors and receives their distance vectors. Assuming that the algorithm begins with each node knowing only the costs to its immediate neighbors, what is the maximum number of iterations required before the distributed algorithm converges? Justify your answer.

The general topology is considered.

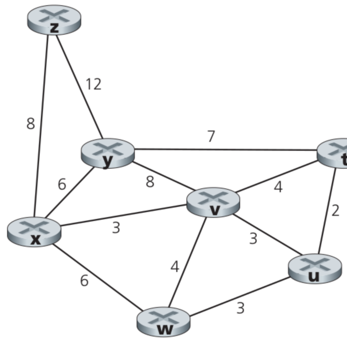
- *The distance table entries are computed using the distance-vector algorithm synchronous version.*
- *The nodes in the network has limited knowledge about their neighbors.*
- *They know only their neighbors costs.*

The maximum number of iterations required for the algorithm convergence can be calculated as follows:
In each iteration, the nodes in the network will exchange information of distance tables with their neighbors.

- *After the first iteration, all the neighboring nodes to the current node will be aware of shortest path cost to current node. For example, let X and Y represent two nodes and they are neighbors. Then after first iteration, all the neighbors of Y will be aware of shortest path cost to node X.*
- *Assume that d (the networks diameter) is the length of the longest path and without loops between any two nodes in the network.*
- *From the above analogy, after performing d-1 iterations, all nodes will have the knowledge about the shortest path cost of d to all other nodes.*
- *If the path length is greater than d hops, then the path contains loops. The removal of loops converges the algorithm result to at most d-1 iterations.*
- *Any path with greater than d hops consists of loops which leads the result of the algorithm to converge in at most d-1 iterations.*

Therefore, the result of the distance vector algorithm converges in at most d-1 iterations.

P25. Consider the following network. With the indicated link costs, use Dijkstra's shortest-path algorithm to compute the shortest path from *x* to all network nodes. Show hos the algorithm works by computing a table similar to Table 4.3.



Step	N'	D(t),p(t)	D(u),p(u)	D(v),p(v)	D(w),p(w)	D(y),p(y)	D(z),p(z)
0	*x*	*INF*	*INF*	*3,x*	*6,x*	*6,x*	*8,x*
1	*xv*	*7,v*	*6,v*	*3,x*	*6,x*	*6,x*	*8,x*
2	*xvu*	*7,v*	*6,v*	*3,x*	*6,x*	*6,x*	*8,x*
3	*xvuw*	*7,v*	*6,v*	*3,x*	*6,x*	*6,x*	*8,x*
4	*xvuwy*	*7,v*	*6,v*	*3,x*	*6,x*	*6,x*	*8,x*
5	*xvuwyt*	*7,v*	*6,v*	*3,x*	*6,x*	*6,x*	*8,x*
6	*xvuwytz*	*7,v*	*6,v*	*3,x*	*6,x*	*6,x*	*8,x*

P27. Consider the network shown in Problem P25. Using Dijkstra's algorithm, and showing your work using a table similar to Table 4.3, do the following:

a. **Compute the shortest path from *y* to all network nodes.**
b. **Compute the shortest path from *t* to all network nodes.**
c. **Compute the shortest path from *s* to all network nodes.**
d. **Compute the shortest path from *u* to all network nodes.**
e. **Compute the shortest path from *w* to all network nodes.**
f. **Compute the shortest path from *v* to all network nodes.**
g. **Compute the shortest path from *z* to all network nodes.**

a.

Num	*N'*	*D(u),p(u)*	*D(v),p(v)*	*D(w),p(w)*	*D(x),p(x)*	*D(y),p(y)*	*D(z),p(z)*
0	*t*	*2,t*	*4,t*	*INF*	*INF*	*7,t*	*INF*
1	*tu*	*2,t*	*4,t*	*5,u*	*INF*	*7,t*	*INF*
2	*tuv*	*2,t*	*4,t*	*5,u*	*7,v*	*7,t*	*INF*
3	*tuvw*	*2,t*	*4,t*	*5,u*	*7,v*	*7,t*	*INF*
4	*tuvwx*	*2,t*	*4,t*	*5,u*	*7,v*	*7,t*	*15,x*
5	*tuvwxy*	*2,t*	*4,t*	*5,u*	*7,v*	*7,t*	*15,x*
6	*tuvwxyz*	*2,t*	*4,t*	*5,u*	*7,v*	*7,t*	*15,x*

b.

Num	*N'*	*D(t),p(t)*	*D(v),p(v)*	*D(w),p(w)*	*D(x),p(x)*	*D(y),p(y)*	*D(z),p(z)*
0	*u*	*2,u*	*3,u*	*3,u*	*INF*	*INF*	*INF*
1	*ut*	*2,u*	*3,u*	*3,u*	*INF*	*9,t*	*INF*

2	utv	2,u	3,u	3,u	6,v	9,t	INF
3	utvw	2,u	3,u	3,u	6,v	9,t	INF
4	utvwx	2,u	3,u	3,u	6,v	9,t	14,x
5	utvwxy	2,u	3,u	3,u	6,v	9,t	14,x
6	utvwxyz	2,u	3,u	3,u	6,v	9,t	14,x

c.

Num	N’	D(t),p(t)	D(u),p(u)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	v	4,v	3,v	4,v	3,v	8,v	∞
1	vx	4,v	3,v	4,v	3,v	8,v	11,x
2	vxu	4,v	3,v	4,v	3,v	8,v	11,x
3	vxut	4,v	3,v	4,v	3,v	8,v	11,x
4	vxutw	4,v	3,v	4,v	3,v	8,v	11,x
5	vxutwy	4,v	3,v	4,v	3,v	8,v	11,x
6	vxutwyz	4,v	3,v	4,v	3,v	8,v	11,x

d.

Step	N’	D(t),p(t)	D(u),p(u)	D(v),p(v)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	w		3,w	4,w	6,w	∞	∞
1	wu	5,u	3,w	4,w	6,w	∞	∞
2	wuv	5,u	3,w	4,w	6,w	12,v	∞
3	wuvt	5,u	3,w	4,w	6,w	12,v	∞
4	wuvtx	5,u	3,w	4,w	6,w	12,v	14,x
5	wuvtxy	5,u	3,w	4,w	6,w	12,v	14,x
6	wuvtxyz	5,u	3,w	4,w	6,w	12,v	14,x