HW6: Routing - Solutions

COM-208: Computer Networks

Link-state routing

Consider the network in Figure 1. Execute the link-state (Dijkstra's) algorithm we saw in class to compute the least-cost path from each of x, v, and t to all the other routers.

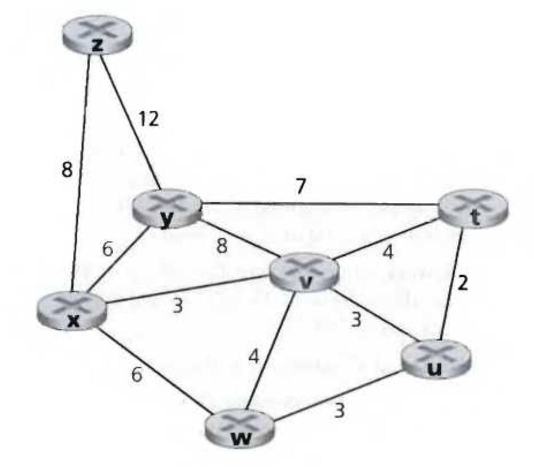


Figure 1: Network topology.

For each router i, $C(i)$ stands for cost to i, and $p(i)$ stands for predecessor to i.

1. Least-cost path from x to all network nodes.

step	nodes visited	C(t),p(t)	C(u),p(u)	C(v),p(v)	C(w),p(w)	C(y),p(y)	C(z),p(z)
0	x	∞	∞	3,x	6,x	6,x	8,x
1	x,v	7,v	6,v	3,x	6,x	6,x	8,x
2	x,v,u	7,v	6,v	3,x	6,x	6,x	8,x
3	x,v,u,w	7,v	6,v	3,x	6,x	6,x	8,x
4	x,v,u,w,y	7,v	6,v	3,x	6,x	6,x	8,x
5	x,v,u,w,y,t	7,v	6,v	3,x	6,x	6,x	8,x
6	x,v,u,w,y,t,z	7,v	6,v	3,x	6,x	6,x	8,x

2. Least-cost path from v to all network nodes.

step	nodes visited	C(t),p(t)	C(u),p(u)	C(w),p(w)	C(x),p(x)	C(y),p(y)	C(z),p(z)
0	v	4,v	3,v	4,v	3,v	8,v	∞
1	v,x	4,v	3,v	4,v	3,v	8,v	11,x
2	v,x,u	4,v	3,v	4,v	3,v	8,v	11,x
3	v,x,u,t	4,v	3,v	4,v	3,v	8,v	11,x
4	v,x,u,t,w	4,v	3,v	4,v	3,v	8,v	11,x
5	v,x,u,t,w,y	4,v	3,v	4,v	3,v	8,v	11,x
6	v,x,u,t,w,y,z	4,v	3,v	4,v	3,v	8,v	11,x

3. Least-cost path from t to all network nodes.

step	nodes visited	C(u),p(u)	C(v),p(v)	C(w),p(w)	C(x),p(x)	C(y),p(y)	C(z),p(z)
0	t	2,t	4,t	∞	∞	7,t	∞
1	t,u	2,t	4,t	5,u	∞	7,t	∞
2	t,u,v	2,t	4,t	5,u	7,v	7,t	∞
3	t,u,v,w	2,t	4,t	5,u	7,v	7,t	∞
4	t,u,v,w,x	2,t	4,t	5,u	7,v	7,t	15,x
5	t,u,v,w,x,y	2,t	4,t	5,u	7,v	7,t	15,x
6	t,u,v,w,x,y,z	2,t	4,t	5,u	7,v	7,t	15,x

Distance-vector routing

Consider the network in Figure 2. Execute the distance-vector (Bellman-Ford) algorithm we saw in class and show the information that router z knows after each iteration.

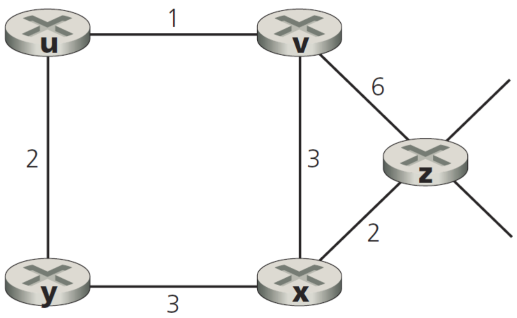


Figure 2: Network topology.

Each node in the topology has its own view of the network, which is updated independently from other nodes at the end of each step. Therefore, for every step of the algorithm, you also need to update each of the other cost tables. Otherwise, your solution may be incorrect.

In our solution we only show the cost table for node z, as required by the question. The cost table at node z consists of 5 columns (all possible destinations) and 3 rows (all possible sources—one row for node z and one row for each neighbor). Each entry of the table denotes the cost between the associated source-destination nodes.

Initially (at step 0), node z has the following view of the network:

		To				
		u	v	x	y	z
From	v	∞	∞	∞	∞	∞
	x	∞	∞	∞	∞	∞
	z	∞	6	2	∞	0

At step 1:

		To				
		u	v	x	y	z
From	v	1	0	3	∞	6
	x	∞	3	0	3	2
	z	7	5	2	5	0

At step 2:

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

At step 3:

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

We see that from step 2 to step 3 the cost tables did not change, indicating that the algorithm has converged.

Convergence

What is the maximum number of iterations required for the distance-vector (Bellman-Ford) algorithm that we saw in class to converge (i.e., to finish, assuming no change occurs in the network graph and link costs)? Justify your answer.

At each iteration, a node exchanges cost tables with its neighbors. Thus, if you are node A, and your neighbor is B, all of B's neighbors (which are all one or two hops from you) will know the least-cost path of one or two hops to you after one iteration (i.e., after B tells them its cost to you).

Let d be the "diameter" of the network, which is computed as follows:

- Find the least-cost path between each pair of nodes.
- The diameter is equal to the greatest length (in number of links) of any of those least-cost paths.

Using the reasoning above, after $d-1$ iterations, all nodes will know the least-cost path of cost $\leq d$ hops to all other nodes. Since any path of cost $> d$ hops is longer (and costlier) than any of the least-cost paths, the algorithm converges in at most $d-1$ iterations.

Poisoned reverse

Consider the network in Figure 3. The routers in the network run the distance-vector (Bellman-Ford) algorithm we saw in class, with poisoned reverse enabled. Suppose the algorithm has run for some time and has converged to the correct least-cost paths.

Table 4 contains the reachability information from each router to router A (e.g., from router D we can reach router A through router C with cost 3).

Now suppose that link $A - B$ goes down.

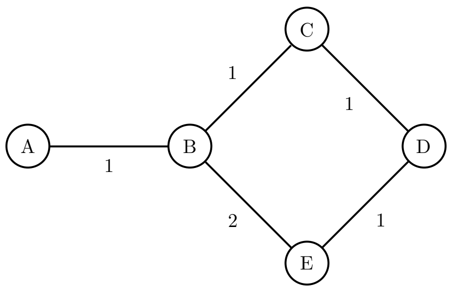


Figure 3: Network topology.

Describe the next 6 steps of the algorithm. For each step, show the reachability information from each router to router A (cost of the least-cost path and next hop).

See Figures 4 and 5.

To fill in the first table we use the second, auxiliary one: Figure 5 shows the routing announcements exchanged between routers and the computations done to update the routing tables. Only the information relevant to routes to router A is shown. At each step, each router sends to its neighbors the best route to A known in the previous step, except if the route is via that neighbor, in which case because of poisoned reverse, ∞ is sent; we mark such messages with (**pr**). Then, each router uses all the announcements it received from the neighbors to update its routing table.

Note that poisoned reverse does not help the algorithm converge.

		from router B	from router C	from router D	from router E
route to router A	step 0	1 via A	2 via B	3 via C	3 via B
	step 1	∞	2 via B	3 via C	3 via B
	step 2	∞	∞	3 via C	4 via D
	step 3	6 via E	∞	∞	4 via D
	step 4	6 via E	7 via B	∞	∞
	step 5	∞	7 via B	8 via C	∞
	step 6	∞	∞	8 via C	9 via D

Figure 4: Reachability information from each router to router A.

		router B	router C	router D	router E
step 0	initial table	1 via A	2 via B	3 via C	3 via B
step 1	announcements	$C \to B : \infty$ (pr) $E \to B : \infty$ (pr) link to A down	$B \to C : 1$ $D \to C : \infty$ (pr)	$C \to D : 2$ $E \to D : 3$	$B \to E : 1$ $D \to E : 3$
	table update	∞	$1 + 1 = 2$ via B	$2 + 1 = 3$ via C	$1 + 2 = 3$ via B
step 2	announcements	$C \to B : \infty$ (pr) $E \to B : \infty$ (pr) link to A down	$B \to C : \infty$ $D \to C : \infty$ (pr)	$C \to D : 2$ $E \to D : 3$	$B \to E : \infty$ $D \to E : 3$
	table update	∞	∞	$2 + 1 = 3$ via C	$3 + 1 = 4$ via D
step 3	announcements	$C \to B : \infty$ $E \to B : 4$ link to A down	$B \to C : \infty$ $D \to C : \infty$ (pr)	$C \to D : \infty$ $E \to D : \infty$ (pr)	$B \to E : \infty$ $D \to E : 3$
	table update	$4 + 2 = 6$ via E	∞	∞	$3 + 1 = 4$ via D
step 4	announcements	$C \to B : \infty$ $E \to B : 4$ link to A down	$B \to C : 6$ $D \to C : \infty$	$C \to D : \infty$ $E \to D : \infty$ (pr)	$B \to E : \infty$ (pr) $D \to E : \infty$
	table update	$4 + 2 = 6$ via E	$6 + 1 = 7$ via B	∞	∞
step 5	announcements	$C \to B : \infty$ (pr) $E \to B : \infty$ link to A down	$B \to C : 6$ $D \to C : \infty$	$C \to D : 7$ $E \to D : \infty$	$B \to E : \infty$ (pr) $D \to E : \infty$
	table update	∞	$6 + 1 = 7$ via B	$7 + 1 = 8$ via C	∞
step 6	announcements	$C \to B : \infty$ (pr) $E \to B : \infty$ link to A down	$B \to C : \infty$ $D \to C : \infty$ (pr)	$C \to D : 7$ $E \to D : \infty$	$B \to E : \infty$ $D \to E : 8$
	table update	∞	∞	$7 + 1 = 8$ via C	$8 + 1 = 9$ via D

Figure 5: Routing announcements exchanged between routers and routing table updates.

Will the algorithm converge to the correct least-cost path values? If yes, in how many steps?

No, the algorithm does not converge (the algorithm takes an infinite number of steps to converge to the correct least-cost path values).

Propose a simple way to make the algorithm converge faster.

Potential solutions:

- Routers keep track of the entire route-path, not just the next hop (e.g., BGP)
- Routers use an upper limit for the cost of a path (e.g., 30). The algorithm converges slowly, but will eventually converge.

Put forwarding and routing together

Consider the network in Figure 6, consisting of:

- End-systems A, B and X, DNS server C, and web server D.
- IP routers R_1, R_2, R_3, and R_4.
- The link costs are noted in Figure 6.
- End-systems A, B and X use C as their local DNS server.

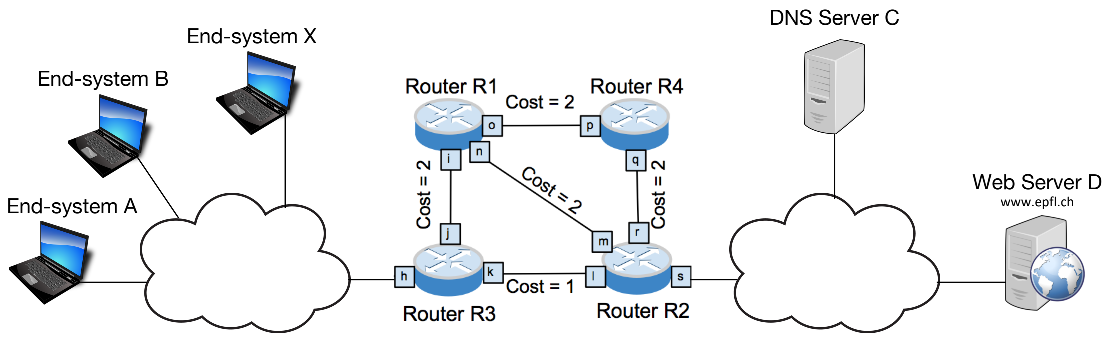


Figure 6: Network topology.

IP prefix/address allocation

Allocate an IP prefix to each IP subnet and an IP address to each network interface that needs one, following these rules:

- All IP addresses must be allocated from `8.8.8.0/24`.
- Each IP subnet must be allocated the smallest possible IP prefix and must have one broadcast IP address.
- IP router interfaces have IP addresses.

Explain how you compute each IP prefix and fill in Table 7.

Subnet number	IP prefix	Interfaces and IP addresses	Broadcast IP address
Example: 1	10.1.1.0/24	x: 10.1.1.0 y: 10.1.1.1 z: 10.1.1.2	10.1.1.255
1	8.8.8.0/29	a : 8.8.8.0 b : 8.8.8.1 x : 8.8.8.2 h : 8.8.8.3	8.8.8.7
2	8.8.8.8/30	c : 8.8.8.8 d : 8.8.8.9 s : 8.8.8.10	8.8.8.11
3	8.8.8.12/30	o : 8.8.8.12 p : 8.8.8.13	8.8.8.15
4	8.8.8.16/30	q : 8.8.8.16 r : 8.8.8.17	8.8.8.19
5	8.8.8.20/30	k : 8.8.8.20 l : 8.8.8.21	8.8.8.23
6	8.8.8.24/30	i : 8.8.8.24 j : 8.8.8.25	8.8.8.27
7	8.8.8.28/30	n : 8.8.8.28 m : 8.8.8.29	8.8.8.31

Figure 7: Allocation of IP prefixes and IP addresses for the network in Figure 6.

Forwarding tables

IP routers R_1, R_2, R_3, and R_4 participate in a least-cost path routing algorithm. All the links between the routers are in good condition (no link has failed or is failing), and the algorithm has converged.

Show the forwarding tables of routers R_3 and R_2.

a) Router R_3:

destination IP prefix	output link
8.8.8.0/29	h
8.8.8.8/30	k
8.8.8.12/30	j
8.8.8.16/30	k
8.8.8.20/30	k
8.8.8.24/30	j
8.8.8.28/30	k

b) Router R_2:

destination IP prefix	output link
8.8.8.0/29	l
8.8.8.8/30	s
8.8.8.12/30	r OR m
8.8.8.16/30	r
8.8.8.20/30	l
8.8.8.24/30	l
8.8.8.28/30	m

Routing = populating the forwarding tables

Routers R_1, R_2, R_3, and R_4 run the Bellman-Ford algorithm.

Show the final state of the Bellman-Ford tables for all the routers, **once the algorithm has converged**.

Answer by filling in the tables below. Here is an example table:

Router X:

from \ to	X	Y	Z
X	0	$5(Y)$	$10(Y)$
Y	5	0	5

$10(Y)$ means that the path from router X to router Z has cost 10 and goes through X's neighbor Y.

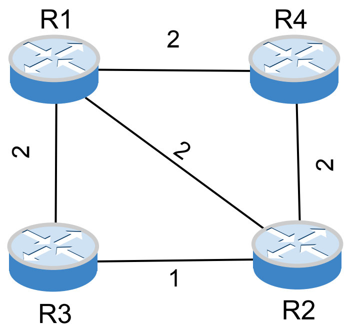


Figure 8: Network topology used in this question.

Router R_1:

from \ to	R_1	R_2	R_3	R_4
R_1	0	2 (R_2)	2 (R_3)	2 (R_4)
R_2	2	0	1	2
R_3	2	1	0	3
R_4	2	2	3	0

Router R_2:

from \ to	R_1	R_2	R_3	R_4
R_2	2 (R_1)	0	1 (R_3)	2 (R_4)
R_1	0	2	2	2
R_3	2	1	0	∞ (**pr**)
R_4	2	2	∞ (**pr**)	0

Router R_3:

from \ to	R_1	R_2	R_3	R_4
R_3	2 (R_1)	1 (R_2)	0	3 (R_2)
R_1	0	2	2	2
R_2	2	0	1	2
R_4				

Router R_4:

from \ to	R_1	R_2	R_3	R_4
R_4	2 (R_1)	2 (R_2)	3 (R_2)	0
R_1	0	2	2	2
R_2	2	0	1	2
R_3				

Routing in the presence of link failures

The link between routers R_1 and R_2 fails.

Show the Bellman-Ford tables for all the routers **from the moment the link fails and until the algorithm has re-converged**. Write clearly after how many interactions (neighbor exchanges) the algorithm re-converges.

Answer by filling in the tables below. We have provided tables for two iterations, but the algorithm may reconverge faster (in which case you just leave some tables empty).

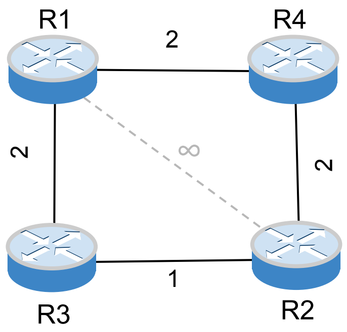


Figure 9: Network topology after the link failure.

State after the link failure:

Router R_1:

from \ to	R_1	R_2	R_3	R_4
R_1	0	3 (R_3)	2 (R_3)	2 (R_4)
R_2				
R_3	2	1	0	3
R_4	2	2	3	0

Router R_2:

from \ to	R_1	R_2	R_3	R_4
R_2	3 (R_3)	0	1 (R_3)	2 (R_4)
R_1				
R_3	2	1	0	∞ (**pr**)
R_4	2	2	∞ (**pr**)	0

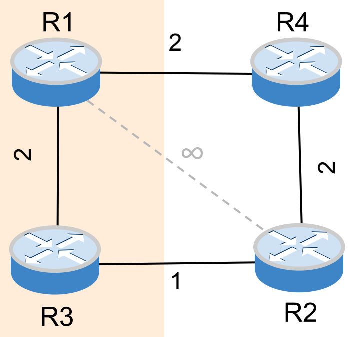
After the first exchange:

Router R_1:

to from	R_1	R_2	R_3	R_4
R_1	0	3 (R_3)	2 (R_3)	2 (R_4)
R_2				
R_3	2	1	0	3
R_4	2	2	3	0

Router R_2:

to from	R_1	R_2	R_3	R_4
R_2	3 (R_3)	0	1 (R_3)	2 (R_4)
R_1				
R_3	2	1	0	∞ (**pr**)
R_4	2	2	∞ (**pr**)	0



Router R_3:

to from	R_1	R_2	R_3	R_4
R_3	2 (R_1)	1 (R_2)	0	3 (R_2)
R_1	0	∞ (**pr**)	2	2
R_2	∞ (**pr**)	0	1	2
R_4				

Router R_4:

to from	R_1	R_2	R_3	R_4
R_4	2 (R_1)	2 (R_2)	3 (R_2)	0
R_1	0	3	2	2
R_2	3	0	1	2
R_3				

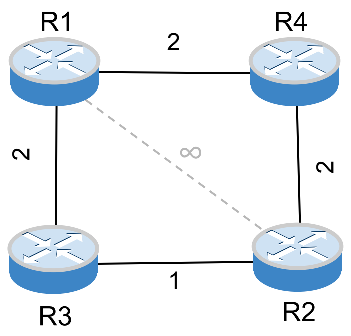
After the second exchange:

Router R_1:

from \ to	R_1	R_2	R_3	R_4
R_1				
R_2				
R_3				
R_4				

Router R_2:

from \ to	R_1	R_2	R_3	R_4
R_2				
R_1				
R_3				
R_4				



Router R_3:

from \ to	R_1	R_2	R_3	R_4
R_3				
R_1				
R_2				
R_4				

Router R_4:

from \ to	R_1	R_2	R_3	R_4
R_4				
R_1				
R_2				
R_3				