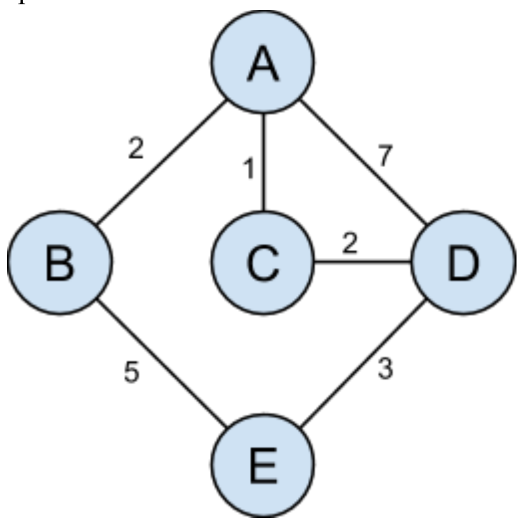
CS168 Fall 2014 Discussion Section 2: Routing

Inspired by EE122 Fall 2013 Discussion Section 2

Problem 1: Link-State Routing

The following is a network of routers using Link-State routing to communicate with each other. The numbers adjacent to each link represent the cost to traverse the link.



(a) After all routers have the global view of the network topology, run Dijkstra's algorithm on each node and fill up the following tables. Rows represent the iteration in each table, and columns represent destinations. Use the notation (cost, previous node) for each cell and specify *S* (set of nodes whose least cost path definitively known).
Node A's table has been filled for you. Note that these are NOT the node's routing tables; only the last row in each table matters in the end. Highlighted cell is chosen to be added to S.

i \ *Dest*	***B***	***C***	***D***	***E***	***S***
1	(2, A)	(1, A)	(7, A)	∞	AC
2	(2, A)	(1, A)	(3, C)	∞	ACB
3	(2, A)	(1, A)	(3, C)	(7, B)	ACBD
4	(2, A)	(1, A)	(3, C)	(6, D)	ACBDE

Node A

i \ *Dest*	***A***	***C***	***D***	***E***	***S***
1					
2					
3					
4					

Node B

Dest / i	A	B	D	E	S
1					
2					
3					
4					

Node C

Dest / i	A	B	C	E	S
1					
2					
3					
4					

Node D

Dest / i	A	B	C	D	S
1					
2					
3					
4					

Node E

(b) Now node B wants to send to D. What path does the packet traverse given the routing tables? What is the cost associated with this path?

(c) The cost of link CD suddenly shoots up to 20? Will there be a transient loop? Where?

Problem 2: Distance-Vector Routing

For the same network topology in Problem 1, consider the nodes communicating with each other using Distance-Vector routing.

Below are the initial routing tables before any routing updates are received. Rows represent the neighbors and columns represent the destination. An adjacent table indicates link costs. Highlighted cell indicates shortest path. For each node, the row corresponding to the same node also indicates the next hop for the shortest path that is chosen.

Nbr	Cost
A	0
B	2
C	1
D	7

To / From	A	B	C	D
A	0, A	2, A	1, A	7, A
B	-	0	-	-
C	-	-	0	-
D	-	-	-	0

Node A

Nbr	Cost
A	2
B	0
E	5

To / From	A	B	E
A	0	-	-
B	2, B	0, B	5, B
E	-	-	0

Node B

Nbr	Cost
A	1
C	0
D	2

To / From	A	C	D
A	0	-	-
C	1, C	0, C	2, C
D	-	-	0

Node C

Nbr	Cost
A	7
C	2
D	0
E	3

To / From	A	C	D	E
A	0	-	-	-
C	-	0	-	-
D	7, D	2, D	0, D	3, D
E	-	-	-	0

Node D

Nbr	Cost
B	5
D	3
E	0

To / From	B	D	E
B	0	-	-
D	-	0	-
E	5, E	3, E	0, E

Node E

Answer following questions, which indicate events that happen consecutively. Assume no packet exchanges other than the ones specified.

(a) C sends its update to A and D.

(a.i) What information is contained in C's update?

(a.ii) What do the routing tables for A and D look like after receiving C's update (You may not need to fill in all columns)?

Nbr	Cost
A	0
B	2
C	1
D	7

To / From					
A					
B					
C					
D					

Node A

Nbr	Cost
A	7
C	2
D	0
E	3

To / From					
A					
C					
D					
E					

Node D

(a.iii) Which nodes among A and D are expected to send routing updates after receiving C's update?

(b) A sends its update to B, C, and D.

(b.i) What information is contained in A's update?

(b.ii) What do the routing tables for B, C, and D look like after receiving A's update (You may not need to fill in all columns)?

Node B

Nbr	Cost
A	2
B	0
E	5

To / From					
A					
B					
E					

Node C

Nbr	Cost
A	1
C	0
D	2

To / From					
A					
C					
D					

Node D

Nbr	Cost
A	7
C	2
D	0
E	3

To / From					
A					
C					
D					
E					

(b.iii) At this point, what route does D use to reach B? It knows that it can route to A via C with total distance 3 and that A can reach B with distance 2. Should it use this information to optimize the route to B or should it wait for an update for C?

(b.iv) Which nodes among B, C, and D are expected to send routing updates after receiving A's update?

(c) D sends its update to A, C, and E.

(c.i) What information is contained in D's update?

(c.ii) What do the routing tables for A, C, and E look like after receiving D's update (You may not need to fill in all columns)?

Node A

Nbr	Cost
A	0
B	2
C	1
D	7

To / From					
A					
B					
C					
D					

Node C

Nbr	Cost
A	1
C	0
D	2

To / From					
A					
C					
D					

Node E

Nbr	Cost
B	5
D	3
E	0

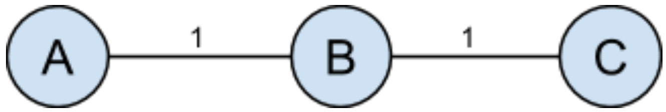
To / From					
B					
D					
E					

(c.iii) Which nodes among A, C, and E are expected to send routing updates after receiving D's update?

(d) Have the routing tables converged? Why or why not?

Problem 3: Count-To-Infinity Problem

Consider a simple topology:



(a) What values will the routing tables have when the system has stabilized (after many rounds)?

Node A

Nbr	*Cost*
A	0
B	1

To / From			
A			
B			

Node B

Nbr	*Cost*
A	1
B	0
C	1

To / From			
A			
B			
C			

Node C

Nbr	*Cost*
B	1
C	0

To / From			
B			
C			

(b) Now suppose the link from A to B goes down, such that A is no longer reachable:
(b.i) B notices the link outage and updates its routing table. What does B's updated routing table look like?

Nbr	*Cost*
A	
B	
C	

To / From			
A			
B			
C			

Node B

(b.ii) According to its routing table, what is the cost of B's minimum-cost path to A?

(c) B sends an update to C. What is C's routing table after receiving the update?

Nbr	*Cost*
B	1
C	0

To / From			
B			
C			

Node C

(d) After updating its table, C sends an update to B. What is B's routing table after receiving the update?

Nbr	*Cost*
A	
B	
C	

To / From			
A			
B			
C			

Node B

(f) How many updates are exchanged before the tables converge?

Problem 4: Poison Reverse

One solution to the count-to-infinity problem is "poison-reverse": if you are currently routing through a neighbor, tell that neighbor that your path to the destination has infinite cost.

(a) Continue on the network topology in Problem 3, before the link from A to B goes down, what is B's routing table (assuming that poison reverse was used when exchanging route information)?

Nbr	*Cost*
A	1
B	0
C	1

To / From			
A			
B			
C			

Node B

(b) B detects the link outage and sends an update to C.

(b.i) What information is contained in B's update?

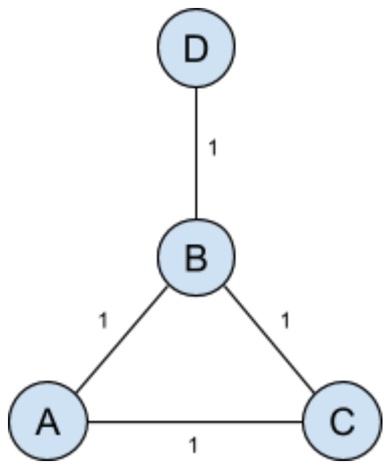
(b.ii) What does C's routing table look like after receiving the update?

Nbr	Cost
B	1
C	0

To / From			
B			
C			

Node C

(c) Now consider a more complex topology, with stabilized routing tables for A, B, and C:



Nbr	Cost
A	0
B	1
C	1

To / From	A	B	C	D
A	0, A	1, A	1, A	2, B
B	1	0	1	1
C	1	1	0	2

Node A

Nbr	Cost
A	1
B	0
C	1
D	1

To / From	A	B	C	D
A	0	1	1	∞
B	1, B	0	1, B	1, B
C	1	1	0	∞
D	∞	1	∞	0

Node B

Nbr	Cost
A	1
B	1
C	0

To / From	A	B	C	D
A	0	1	1	2
B	1	0	1	1
C	1, C	1, C	0, C	2, B

Node C

Suppose the link between B and D goes down. B notices this change and sends an update to A.

(c.i) What is A's routing table after processing B's update?

Nbr	Cost
A	0
B	1
C	1

To / From				
A				
B				
C				

Node A

(c.ii) A then sends an update back to B. What is B's routing table after processing A's update?

Nbr	*Cost*
A	
B	
C	
D	

To / *From*	*A*	*B*	*C*	*D*
A				
B				
C				
D				

Node B

(c.iii) How might you avoid the count-to-infinity problem here altogether?