Goutam Bhat – *goubh275*
Santiago Pagola - *sanpa993*

Lab 4 – Distance Vector Routing

Table of Contents

How Distance Vector Routing works

The Distance Vector Routing, hereafter DVR, works as follows: each node in the network maintains a table of distances to all the nodes in the network.

Initially, each node knows only the distances to its neighbors, and whenever a node updates its distance table, it notifies this by sending the updated costs to its neighbors.

Whenever a node 'j' receives a message from node 'k', node 'j' finds the distances to all other nodes via 'k', and ultimately updates the distance table. Then it finds the new minimum costs to all the remaining nodes in the network.

If any of the minimum costs changed, then the node will send the new costs to all its neighbors. Eventually all the distance nodes will converge to the correct minimum paths to all other nodes.

Consider for instance the following graph representing three nodes:

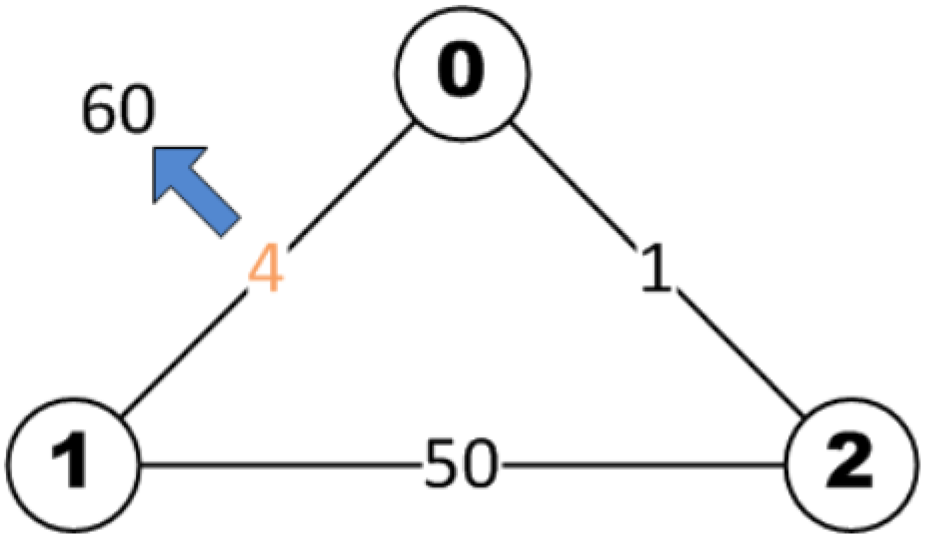


Illustration 1: A simple network topology with 3 nodes. The blue arrow represents that the cost of the link between nodes 0 and 1 will vary after a while, thus changing from value 4 to 60.

Goutam Bhat – *goubh275*
Santiago Pagola - *sanpa993*

In this example, initially node N0 knows the costs to its neighbors N1 and N2 (4 and 1 respectively). N0 will send this information to N1 and N2. At this point, N2 only knows the costs to its neighbors 1 and 0 (50 and 1 respectively). When N2 receives the information from N0, it sees that N0 has a path to N1 with cost 4, so the path from N2 to N1 via N0 will only have the cost 1 + 4 = 5, which is less than the current shortest path to N1 which N2 knows. So N2 will update it's distance table, and send the new minimum costs to N1 and N0. This will continue till all the distance tables converge.

Poison Reverse

The DVR routing can get into count-to-infinity problems. Suppose after all the distance tables have converged, the cost of the link between N0 and N1 changes to 60. N0 will see this change and try to find a new minimum path to N1. To do so, it will check it's distance table, which states that there is a path via N2 to N1 of cost 6 which is less than 60, so N0 will update its distance table and minimum cost with this information. However N0 doesn't know that the path to N1 which N2 'knows' is via N0 itself, so the calculated cost of 6 is actually incorrect. N0 will send this updated costs to N2, at which point N2 will calculate the new path to N1 via N0 of cost 6 + 1 = 7 and send this to N0. So both N2 and N0 will keep looping, i.e. incrementing their costs to N1 till the cost from N2 to N1 via N0 becomes 51, at which points N2 will take the direct path to N1 with cost 50. However because of the looping problem, the convergence will be slow.

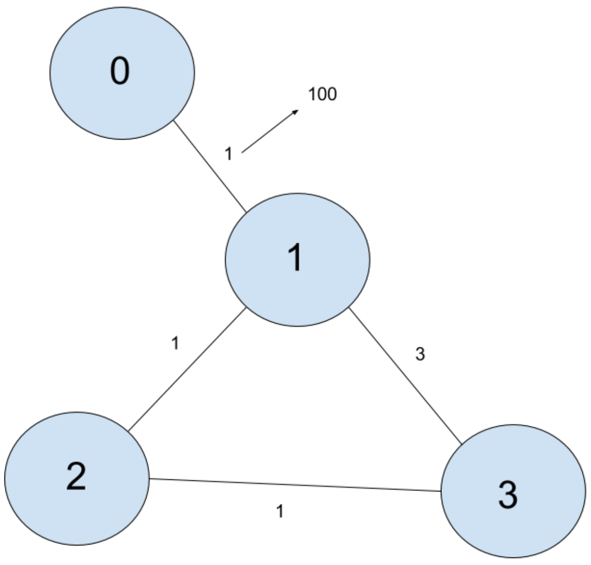
To circumvent this problem, DVR uses poison reverse technique. Whenever a node I is sending the minimum cost information to it neighbor node j, the node will first check if any of the minimum cost paths is via node j itself. If that is the case for e.g node k., node I well set the minimum cost to node k to infinity. This makes sense as if the minimum path is via node j, then node j should already have the information about minimum path to node k. So Node I sending this information to node j will be redundant. However in case of the link N0-N1 changing in Illustration 1 (as discussed above), poison reverse will prevent the count-to-infinity problem. Since the minimum path from N2 to N1 is via N0, N2 will advertise to N0 that it has no paths to N1. So when the link cost increases, since N0 has to other path to N1 (apart from the direct path) in it's distance table, it will set the minimum cost to N1 to 60. When N2 receives this message, it will change its minimum cost to N1 to 50 (the direct path). Since the minimum cost path is not via N0, N2 sends this new information to N0 now. So N0 will update its minimum cost to N1 to 51. So the convergence is much faster in this case with poison reverse.

How algorithms were tested

The algorithms were tested by running the DVR on the provided networks using the simulator and looking at the final converged distance table for all the nodes and verifying that they were correct. We also checked for the existence of *"count to infinity"* problem when the variable `LINKCHANGES` was set to true, both in the case of poison reverse and no poison reverse by looking at the distance table updates after the link cost changes. We also checked the time for convergence both in case of poison reverse and no poison reverse and verified that convergence was achieved faster using the poison reverse technique.

Poisoned Reverse – when it fails

The poison reverse will fail in case of loops with 3 or more nodes. For instance, consider the following network.



After all the distance tables have converged (while using Poison reverse), we will have the following distance tables.

Node 0:

	To N1	To N2	To N3
Via neighbor N1	1	2	3
Via neighbor N2	-	-	-
Via neighbor N3	-	-	-

Goutam Bhat – *goubh275*
Santiago Pagola - *sanpa993*

Node 1:

	To N0	To N2	To N3
Via neighbor N0	1	-	-
Via neighbor N2	-	1	2
Via neighbor N3	6	4	3

Node 2:

	To N0	To N1	To N3
Via neighbor N0	-	-	-
Via neighbor N1	2	1	4
Via neighbor N3	-	-	1

Node 3:

	To N0	To N1	To N2
Via neighbor N0	-	-	-
Via neighbor N1	4	3	4
Via neighbor N2	3	2	1

Note: Since poison reverse is used, in case of distance table of N1, the path to N0 via N2 is undefined since the shortest path from N2 to N0 is via N1 itself, and hence N2 never advertises a path to N0 to N1.

Now suppose the cost of link N0-N1 changes from 1-100. As N1 gets this information, it will see in its distance table that there is a path via N3 to N0 with cost only 6 < 100, so N0 will update its distance table accordingly and send this info to N2. N2 will update the cost to N0 to 7. Since the path from N2 to N0 is not via N3, N2 will send this new cost to N3. N3 will now update the cost to N0 to 8 and send this to N1 and so on. Since the loop has 3 nodes in this case, the count-to-infinity problem occurs despite poison reverse.

Solution to Poison Reverse failure

The poison reverse failure can be prevented by keeping a count of the number of hops for each minimum path and putting a limit on the number of hops allowed. So each node will store the minimum paths to all other nodes via all its neighbors. It will also store the number of hops for each of these minimum paths. When sending the cost information to it’s neighbors, the node will also send the number of hops for each of the minimum cost paths. If there is a routing loop, the number of hops will start growing. So if we restrict the number of hops in a path, i.e. set the cost to infinity in case some path has more than the allowed number of hops, we can prevent the routing loops from happening, and thus stop count-to-infinity problem. RIP uses a limit of 15 hops (see [1]).

References

[1] https://en.wikipedia.org/wiki/Routing_Information_Protocol