Introduction to Computer Networks

19AIE211

Distance Vector Routing Algorithm

Team - R

Done by,

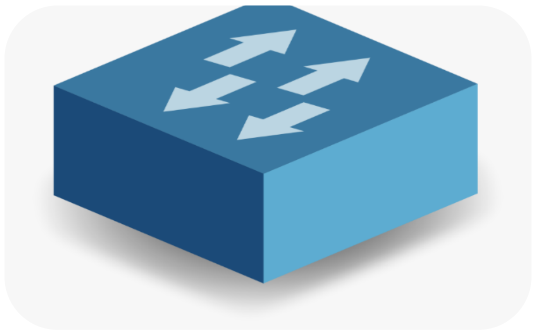
K V Sai Krishna	– CB.EN.U4AIE19035
M Sai Sashank	– CB.EN.U4AIE19040
P Sri Vaishno	– CB.EN.U4AIE19048
Sagala Nitish	– CB.EN.U4AIE19054
Roshan Tushar S	– CB.EN.U4AIE19071

Table of Contents

Introduction

The objective of our project is to implement the Distance Vector Routing algorithm (DVR) with different animations and software implementations. The basic devices that are used in the project are a Router and a switch.

Router helps to send or receive data from or to computers in another network.



A Switch is a data link device that can detect the system to which the data is to be sent by inspecting the message.

Distance Vector Routing

- A distance vector routing (DVR) protocol requires that a router inform its neighbours of topology changes periodically. Historically known as the old ARPANET routing algorithm (or known as Bellman Ford algorithm).
- The Distance vector routing protocol is an autonomous routing protocol, which considers the entire autonomous system as a graph, where routers are the nodes and the networks are the lines or paths connecting the nodes.
- Each path has a cost value, which a datagram has to pay to travel between nodes. The paths with smaller cost are more preferred than the larger ones. Each router in the system organizes this information in a routing table.
- Every router shares this routing table to its neighbors after regular intervals. These Routers use distance vector to update their routing table and calculate the shortest path in the network. This process continues until every router's routing table is updated.

Bellman-Ford Algorithm

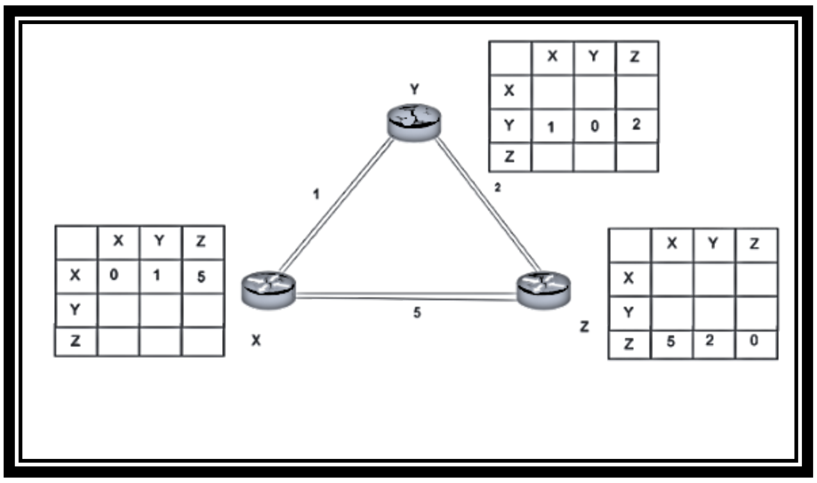
- The distance vector routing protocol uses the basic principle of the Bellman-Ford Algorithm to identify the shortest path through the network.

- Each path connecting the nodes is associated with the cost. So, to calculate the shortest distance follow the following steps:

C(x, v)	cost for direct link from x to v
Dx(y)	estimate of least cost from x to y
$D_v = [D_v(y): y \in N]$	Node x maintains its neighbors distance vectors
$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\}$	Each node v periodically sends Dv to its neighbors and neighbors update their own distance vectors

Example

Consider 3-routers X, Y and Z as shown in figure. Each router has their routing table. Every routing table will contain distance to the destination nodes.



	X	Y	Z
X			
Y	1	0	2
Z			

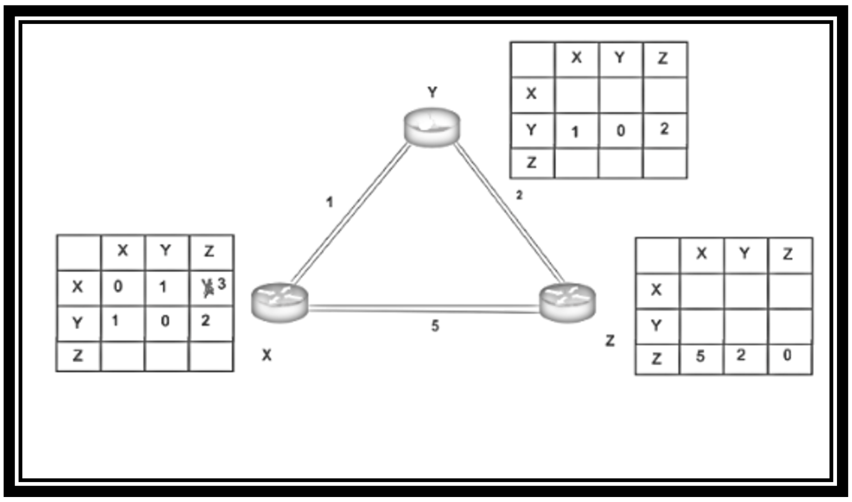
	X	Y	Z
X	0	1	5
Y			
Z			

	X	Y	Z
X			
Y			
Z	5	2	0

Consider router X, X will share it routing table to neighbours and neighbours will share it routing table to it to X and distance from node X to destination will be calculated using bellmen- ford equation.

Dx(y) = min {C(x, v) + Dv(y)} for each node y ∈ N

As we can see that distance will be less going from X to Z when Y is intermediate node(hop) so it will be update in routing table X

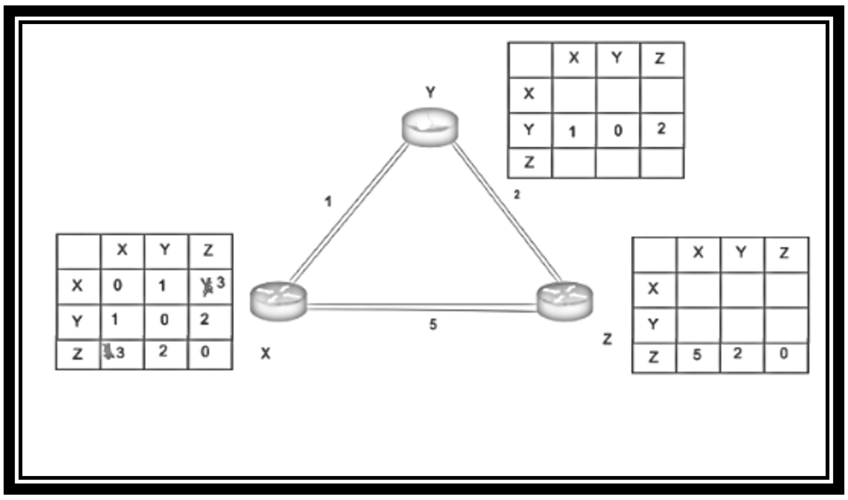


	X	Y	Z
X	0	1	~~5~~ 3
Y	1	0	2
Z			

	X	Y	Z
X			
Y	1	0	2
Z			

	X	Y	Z
X			
Y			
Z	5	2	0

Similarly for Z also,

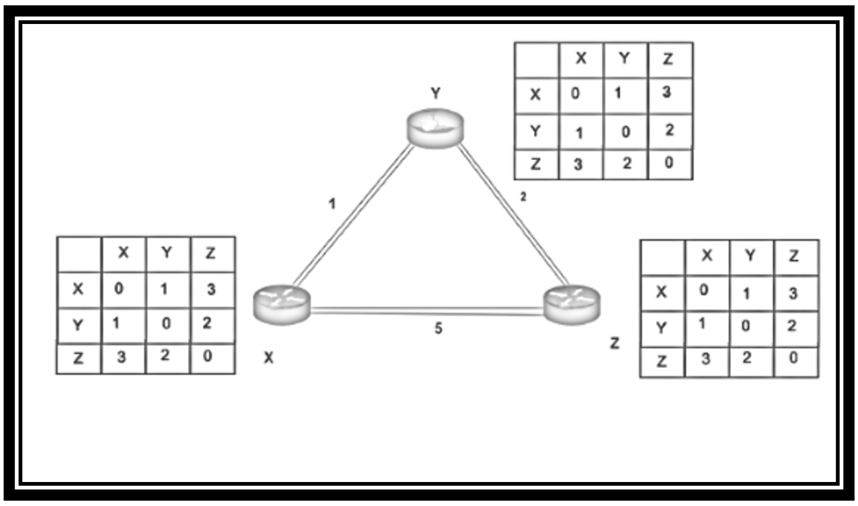


	X	Y	Z
X	0	1	~~5~~ 3
Y	1	0	2
Z	~~5~~ 3	2	0

	X	Y	Z
X			
Y	1	0	2
Z			

	X	Y	Z
X			
Y			
Z	5	2	0

Finally, the routing table is shown below;



	X	Y	Z
X	0	1	3
Y	1	0	2
Z	3	2	0

	X	Y	Z
X	0	1	3
Y	1	0	2
Z	3	2	0

	X	Y	Z
X	0	1	3
Y	1	0	2
Z	3	2	0

Pseudo Code for Bellman Ford Algorithm

The Bellman-Ford algorithm is an example of Dynamic Programming. It starts with a starting vertex and calculates the distances of other vertices which can be reached by one edge. It then continues to find a path with two edges and so on. The Bellman-Ford algorithm follows the bottom-up approach.

This algorithm takes as input a directed weighted graph and a starting vertex. It produces all the shortest paths from the starting vertex to all other vertices.

Now let's describe the notation that we used in the pseudocode. For the first loop is to initialize the vertices. The algorithm initially set the distance from starting vertex to all other vertices to infinity. The distance between starting vertex to itself is 0.

In the Next loop, that is after the initialization step, the algorithm started calculating the shortest distance from the starting vertex to all other vertices. This step runs (|V| - 1) times. Within this step, the algorithm tries to explore different paths to reach other vertices and calculates the distances. If the algorithm finds any distance of a vertex that is smaller than the previously stored value then it relaxes the edge and stores the new value. That is implemented in the last relax function.

Finally, when the algorithm iterates (|V| - 1) times and relaxes all the required edges, the algorithm gives a last check to find out if there is any negative cycle in the graph.

In the next loop, if there is a negative cycle then the distances will keep decreasing. In such a case, the algorithm terminates and gives an output that the graph contains a negative cycle hence the algorithm can't compute the shortest path. If there is no negative cycle found, the algorithm returns the shortest distances.

Bell-man Ford Algorithm Simulation

Input Format

Initialize a Node:

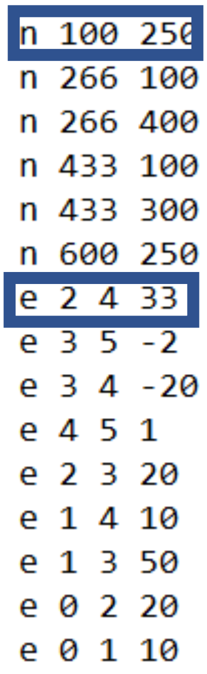
Syntax: n (x-coordinate) (y-coordinate)

Eg: **n 100 250** (this is a node with its x coordinate is 100, y coordinate is 250)

Initialize an Edge:

Syntax: e (from-vertex) (to-vertex) (edge-weight)

Eg: **e 2 4 33** (an edge between vertex 2 and 4 with weight 33)



```
n 100 25
n 266 100
n 266 400
n 433 100
n 433 300
n 600 250
e 2 4 33
e 3 5 -2
e 3 4 -20
e 4 5 1
e 2 3 20
e 1 4 10
e 1 3 50
e 0 2 20
e 0 1 10
```

Idea of the algorithm

- The Bellman-Ford Algorithm computes the cost of the cheapest paths from a source node to all other nodes in the graph.
- The algorithm proceeds in an interactive manner, by beginning with a bad estimate of the cost and then improving it until the correct value is found.
- The first estimate is:
    - ➢ The starting node has cost 0, as his distance to itself is obviously 0.
    - ➢ All other node has cost infinity, which is the worst estimate possible.
- Afterwards, the algorithm checks every edge for the following condition: Are the cost of the source of the edge plus the cost for using the edge smaller than the cost of the edge's target?
- If this is the case, we use a short-cut: It is more profitable to use the edge which was just checked, than using the path used so far. Therefore, the cost of the edge's target gets updated: They are set to the cost of the source plus the cost for using the edge.
- Looking at all edges of the graph and updating the cost of the nodes is called a phase. Unfortunately, it is not sufficient to look at all edges only once. After the first phase, the cost of all nodes for which the shortest path only uses one edge have been calculated correctly.
- How many phases ware necessary? To answer this question, the observation that a shortest path has to use less edges than there are nodes in the graph. Thus, we need at most one phase less than the number of nodes in the graph. A shortest path that uses more edges than the number of nodes would visit some node twice and thus build a circle.

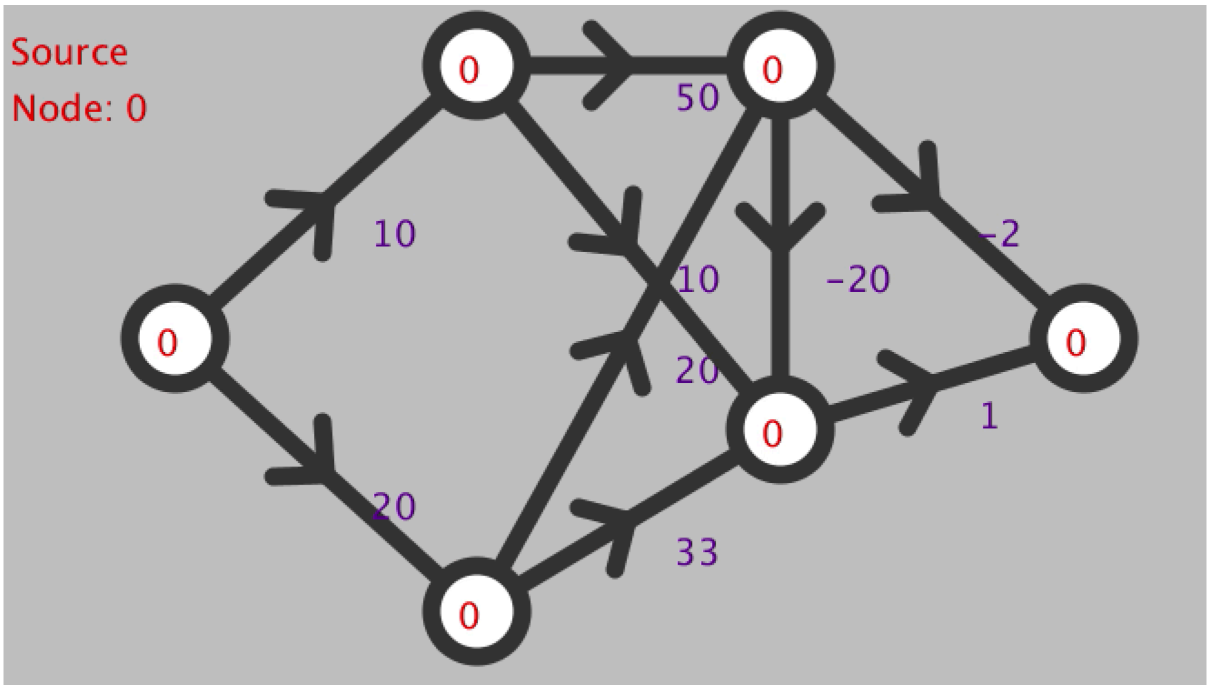
Construction of the shortest path

- Each time when updating the cost of some node, the algorithm saves the edge that was used for the update as the predecessor of the node.
- At the end of the algorithm, the shortest path to each node can be constructed by going backwards using the predecessor edges until the starting node is reached.

Run the simulation

First choose a starting node!

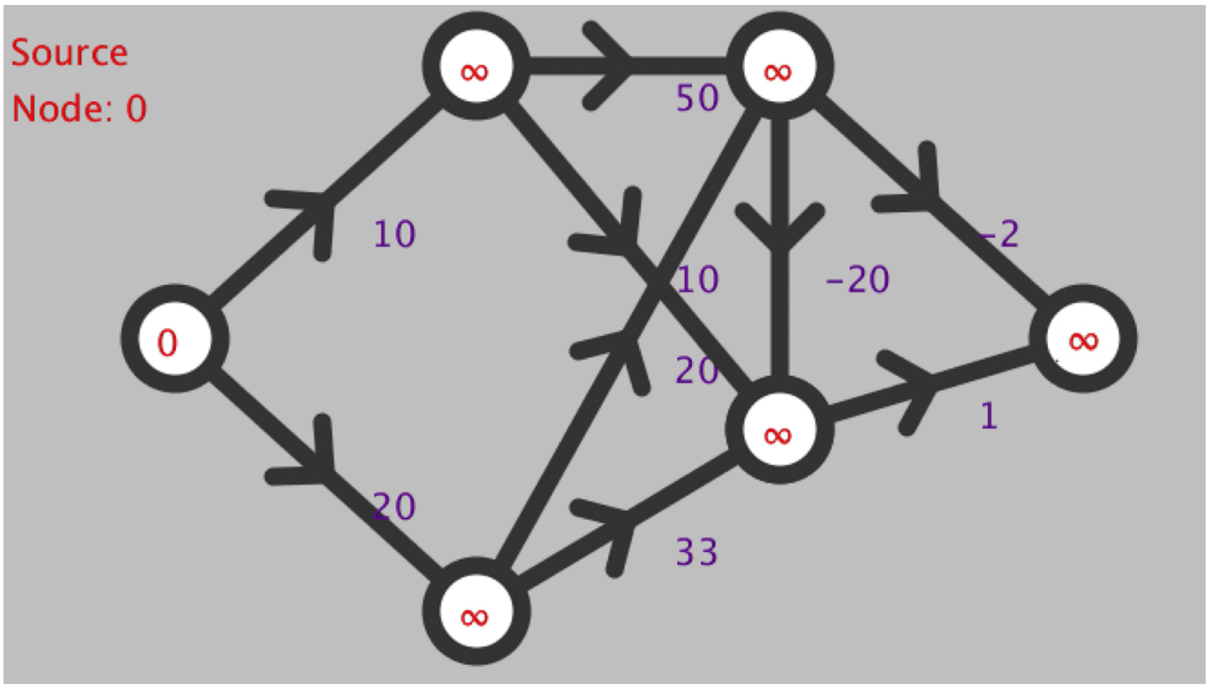
- From this node, the distances to all other nodes are calculated.
- Just click on the corresponding node in the graph.



Now the algorithm can begin!

Initialization

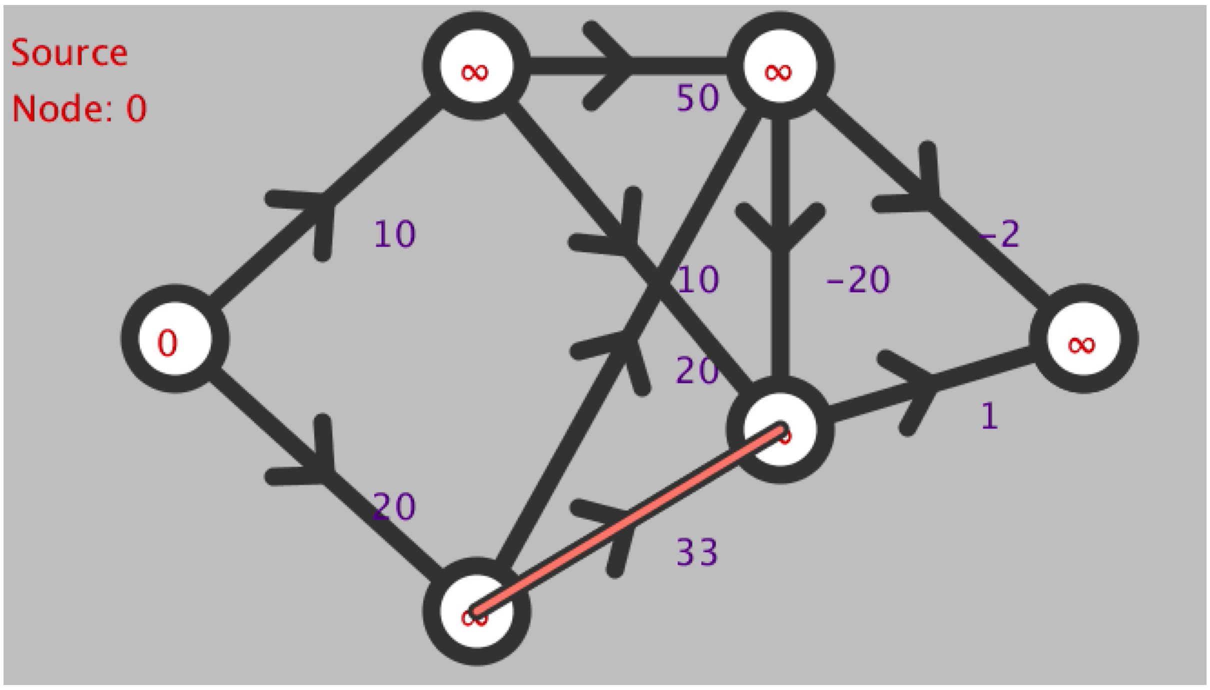
- The algorithm now has saved a first estimate for the cost of each node.
- Thus, we number in the node corresponds to the first estimate of the cost of a cheapest path from the starting node to this node.
- As no calculation has been done yet, we estimate infinite cost for all nodes apart from the starting node itself.



Determining shortest paths!

- Up to now, all shortest paths have been calculated that use at most 0 Edges.
- In the following phase, all shortest paths are being calculated that use at most an edge.
- In each phase, all edges are checked in order to assert whether a reduction of the cost of the target node is possible.

Two possibilities



Checking improvement using red edge

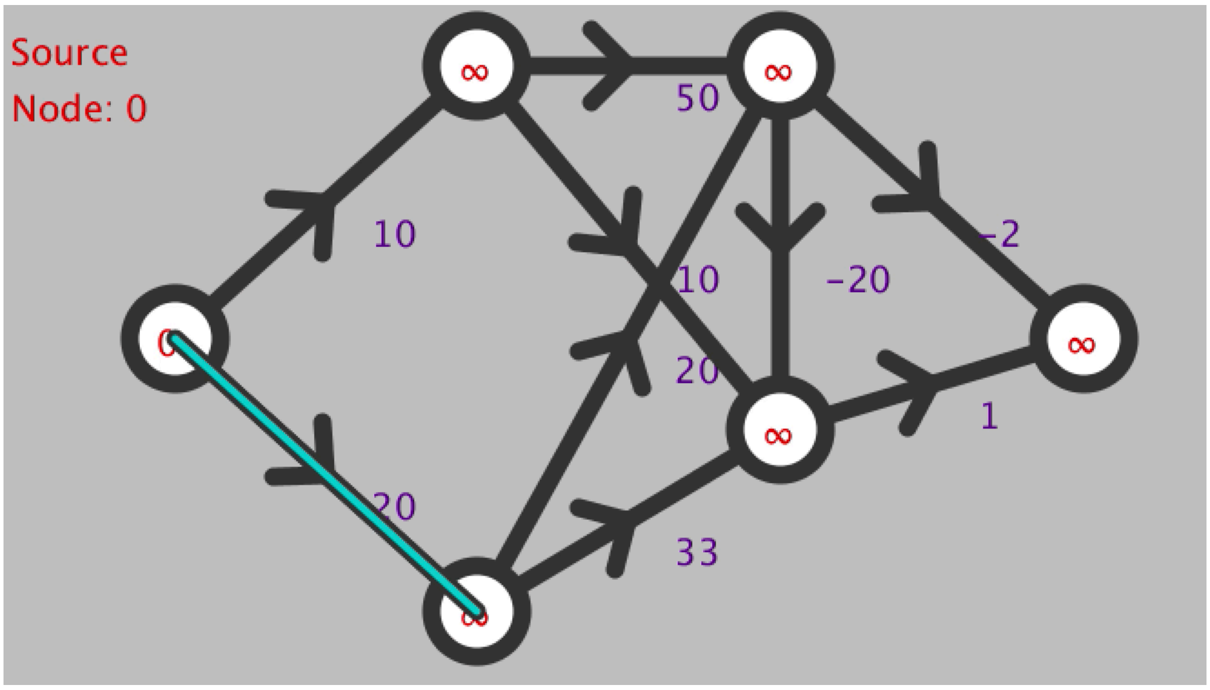
We are now checking whether the red edge allows a better path to its target node than the one known previously.

We thus check whether the currently known cost of the target node are greater than the cost of the source node plus the cost of the edge.

Cost of source node: ∞

Cost of edge: 33

Cost of target node cannot be updated.



Checking improvement using red edge

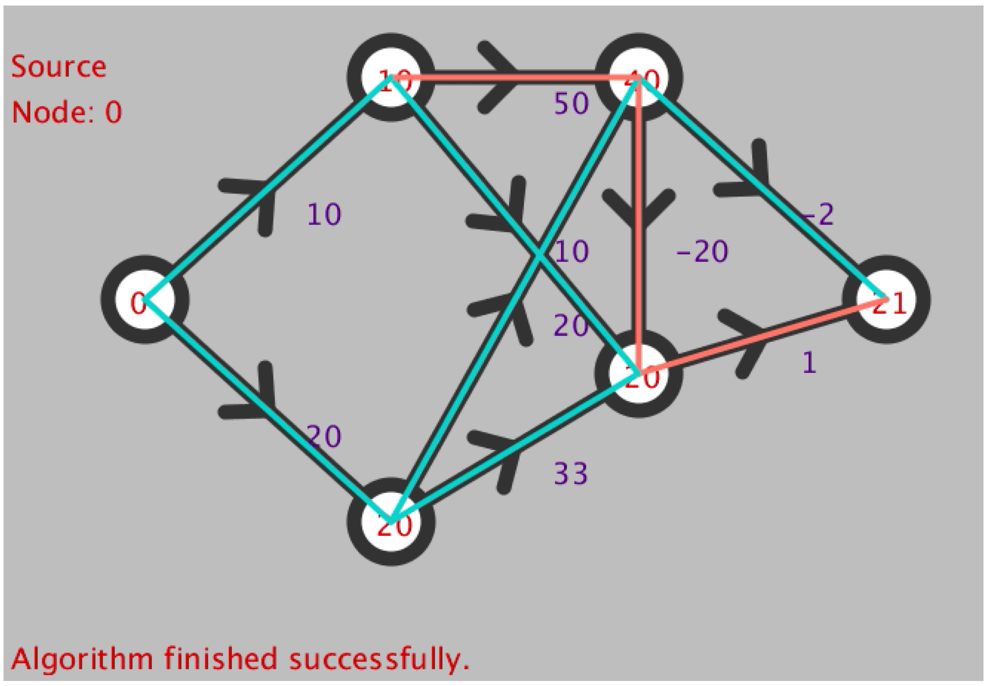
We are now checking whether the red edge allows a better path to its target node than the one known previously.

We thus check whether the currently known cost of the target node are greater than the cost of the source node plus the cost of the edge.

Cost of source node: 0

Cost of edge: 10

Cost of target node have to be updated.



Result: Algorithm finished successfully.

This graph does not contain a negative circle reachable from the starting node, and all cost have been computed correctly.

DVR Animation Video

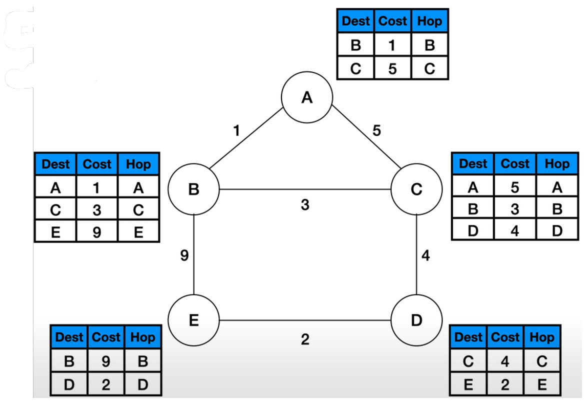
The initial routing consists of the information from the source to its adjacent neighbours like dest, cost, hop. In the video if we consider our source node as C then A, B, D are its neighbours and cost are the distances between them and hop is the destination itself since these are the initial tables and the hop is the destination. Similarly, the initial routing table for each of the node.

Again, let us consider our source node as C, to calculate the final routing table of C, we are gathering information from its adjacent nodes i.e., A, B, D.

Firstly, using the initial routing table of C, we are updating the table with the help of information from A. Now, we are using this updated routing table of C and updating it one more with the information from B and finally updating the routing table once again with the information from D.

As you can see the final routing table of C which includes the cost from C to every other node (A, B, D, E) and hop indicating the path we need to follow.

Similarly, we are considering each node as source node and calculating their final routing tables which is shown at the end of the video.



Router A

Dest	Cost	Hop
B	1	B
C	5	C

Router B

Dest	Cost	Hop
A	1	A
C	3	C
E	9	E

Router C

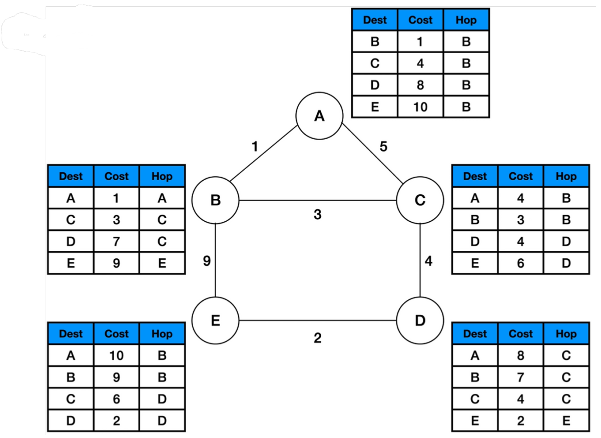
Dest	Cost	Hop
A	5	A
B	3	B
D	4	D

Router E

Dest	Cost	Hop
B	9	B
D	2	D

Router D

Dest	Cost	Hop
C	4	C
E	2	E



Router A

Dest	Cost	Hop
B	1	B
C	4	B
D	8	B
E	10	B

Router B

Dest	Cost	Hop
A	1	A
C	3	C
D	7	C
E	9	E

Router C

Dest	Cost	Hop
A	4	B
B	3	B
D	4	D
E	6	D

Router E

Dest	Cost	Hop
A	10	B
B	9	B
C	6	D
D	2	D

Router D

Dest	Cost	Hop
A	8	C
B	7	C
C	4	C
E	2	E

Initial Routing tables Final Routing tables

RIP in DVR using Cisco Packet Tracer

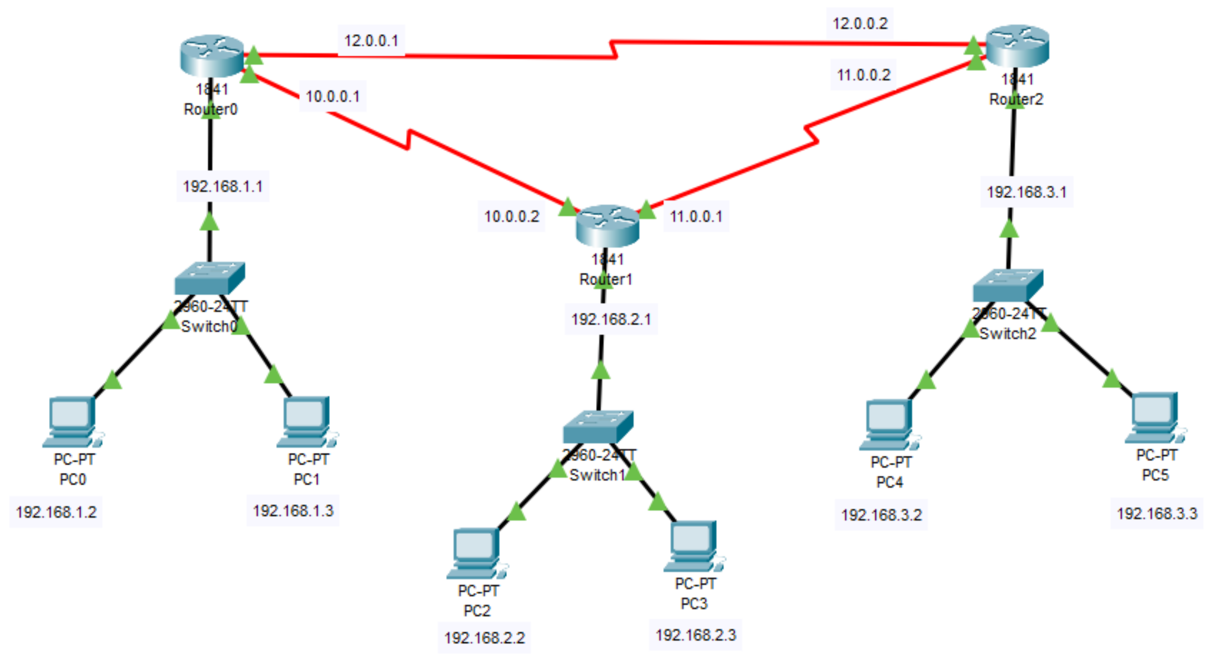
First, we take 6 devices and then we will connect them to the routers through three routers here we take three switches one switch for two devices and connect them with copper straight through because they are different type of devices and if they were same type of devices then we can connect with the copper crossover.

And next here we will connect all the fast ether net zero with the switches port you can give any of them because we don't need to configure the switch but we need to configure the router so we cannot connect just normally with the router we need to remember on which port we are connecting the switch so that we can add the same Ip address as the same network here also we are using copper straight over and we will connect routers it fast ether net 0 here you need to remember it because we need to configure the router with the IP address that will be same as the end devices IP address.

Next, we created a network that have same IP address same net id now we need to you can go to configuration and see that there are only two port we need more than 2 port to connect 3 routers so for that we need to go to physical and then WI C 2 T and turn off the switch and take this inside this box and then turn on the switch again similarly we have to do this for remaining two routers.

And then we connected them this is our serial port so we need to connect with serial where so here serial port 0 to serial port 0 we are connecting with serial DT line here serial port 0 1 to this router serial port 0 To configure them we need to give them IP address for that go to configuration and here we need to give the IP address connecting the routers IP address 192.168.1.1 here we gave a Class C IP address and we can turn on this router you and then we can see this type green signal so this line is on and now we need to have the same Ip address a net ID in this whole network so we need to have the same net ID and the host ID will be different so we can add from 0 to 255 devices here we are using our second device and the default gateway will be the same as routers IP address because that is the only route from where the device can send and receive data.

Now here we are configuring the these roots this was connected to our serial port zero so we need to give our IP address here and here we will use a Class A IP address you can use Class C IP address here too it doesn't matter here we gave 10.0.0.1 an IP address for the second routers now we can just add the networks before sending the data in the static side we need to give Network the mask and then the next hop but inside the routing information protocol here you can see only network is added we don't need to give the next hop or the root name here because this is actually dynamic routing so here it will actually find the hope by itself it does not we don't need to give them the hope name but we just need to give them the network id and it will find the best path available for it to send the data from one device to another device or from our source to our destination it will find the path itself and it will also find the best path available.



Advantages and Disadvantages

Advantages:

- Distance vector routing protocol is easy to implement in small networks. Debugging is very easy in the distance vector routing protocol.
- This protocol has a very limited redundancy in a small network.

Disadvantages:

- A broken link between the routers should be updated to every other router in the network immediately. The DVR takes a considerable time for the updation. This problem is also known as count-to-infinity.
- The time required by every router in a network to produce an accurate routing table is called convergence time. In the large and complex network, this time is excessive.
- Every change in the routing table is propagated to other neighboring routers periodically which create traffic on the network.

Key Takeaways

- Distance vector routing protocol is a dynamic routing protocol.
- It is based on the principle of Bellman-Ford's algorithm.
- This protocol helps the router to find the shortest path through the network.
- Each router is aware of every other router in the network.
- Each router shares the updated records of the routing table with the neighboring routers which help them to update their routing table. The neighboring routers, in turn, send their updated information to their neighboring routers.
- The Routing Information Protocol (RIP) implements the distance vector routing protocol to find the best path in the network along with some considerations.

GITHUB

https://github.com/roshantushar/Distance-Vector-Routing-Algorithm