

Distance-vector routing

- The distance-vector routing (DVR) is designed to periodically update the routing data in the network model based on the Bellman-Ford algorithm.
- The distance vector routing protocol is applied to assign the best and the shortest route for the data.
- In this network protocol, the distance refers to the distance (vector) between neighbouring nodes, and the routing refers to the established route.

Key Features of the protocol

Information About the Network

Every router is responsible for sharing the network knowledge in the channel more responsibly with the neighbouring nodes.

Routing Pattern of the Network

Sharing of routing information is done only between directly connected network nodes in the channel.

Sharing of Data Periodically

Each node in the connection is designed to share the updated routing data with each of the nodes in the network.

Algorithm Applied in Distance Vector Routing

The basis of distance vector routing is designed on the working of the Bellman-Ford Algorithm.

According to the algorithm, each of the nodes in the network is designed to maintain a distance-vector table carrying the distance between itself and its direct neighbouring nodes in the connection.

$$d_x(y) = \min_v \{c(x, v) + d_v(y)\}$$

Then using the algorithm, we can deduce the following points for the routing algorithm in the network:

The sharing of distance vectors between neighbouring nodes is done using routing tables.

Each routing table is shared with the latest distance-vector values in the network.

Each routing table data is shared at regular intervals in the network, with an update of the distance vector beyond the neighbouring nodes.

Where,

$d_x(y)$ - The least distance from x to y.

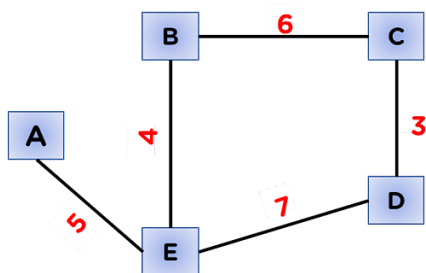
$c(x, v)$ - Node x's cost from each of its neighbour v.

$d_v(y)$ - Distance of each neighbor from initial node.

\min_v - Selecting the minimum distance for the data packet.

Network Model

To better understand the working of the routing protocol, we can do so by using the following steps in the example:



Initial Step

In the first step, we will design the routing table for node A, using the only neighboring node, i.e., node E.

Node A

Destination	Vector	Hop
A	0	A
B	∞	-
C	∞	-
D	∞	-
E	5	E

Next, we will go through the same step for node E, i.e.,

Node E

Destination	Vector	Hop
A	5	A
B	4	B
C	∞	-
D	7	D
E	0	E

Similarly, we repeat the previous step and design the routing table for each of the nodes.

Update Step

Now, after performing the initial step, we will perform the update step for all the nodes; for this, let's look into node A,

To update node A, we will use all the distance vectors from the nodes,


From node A to node B,

A to B:

$(A,E) + (E-B)$

$5 + 4$

9



From node A to node C,

A to C:

$(A,E) + (E-C)$

$5 + \infty$

From node A to node D,

A to D:

$(A,E) + (E-D)$

$5 + 7$

12

From node A to node E.

A to E:

(A,E)

5

The final updated table for node A we get would be this,

Node A		
Destination	Vector	Hop
A	0	A
B	9	E
C	-	-
D	12	E
E	5	E

Similarly, we can perform the update step for all the nodes in the model, and this update step is to be followed for (n-1) iterations where n - Number of nodes. Going by our example model, we will perform the update step atleast four times, i.e., $(5-1) = 4$.

At the end of the update step, we will get the most efficient routing data for each node in the network model, where the sharing of routing data at regular intervals will still continue in the network.