# Experiment 4: Distance Vector Routing

**Aim:** To generate routing tables for a network of routers using Distance Vector Routing

**Objective:** After carrying out this experiment, students will be able to:

* Generate routing tables for a given network using Distance Vector Routing
* Analyze the reasons why Distance Vector Routing is adaptive in nature

**Problem statement:** You are required to write a program that can generate routing tables for a network of routers. Take the number of nodes and the adjacency matrix as input from user. Your program should use this adjacency matrix and create routing tables for all the nodes in the network. The routing table should consist of one entry per destination. This entry should contain the total cost and the outgoing line to reach that destination.

**Analysis:** While analyzing your program, you are required to address the following points:

* Why is Distance Vector Routing classified as an adaptive routing algorithm?
* Limitations of Distance Vector Routing

**MARKS DISTRIBUTION**

|  |  |  |
| --- | --- | --- |
| **Component** | **Maximum Marks** | **Marks Obtained** |
| Preparation of Document | 7 |  |
| Results | 7 |  |
| Viva | 6 |  |
| **Total** | **20** |  |

Submitted by: Chirag Mittal

Register No: 16ETCS002029

1. **Algorithm/Flowchart**

Start

1. Let M be a matrix of size |V| x |V| which stores the graph edge information in adjacency matrix form

2. for each node i in graph, initialize a structure with two arrays – cost and via\_node both of size |V|. Store these structures in an array called table of size |V|.

3. for i from 1 to |V|:

4. for j from 1 to |V|:

5.for k from 1 to |V|:

6. if M[i][j] > M[i][k] + M[k][j]:

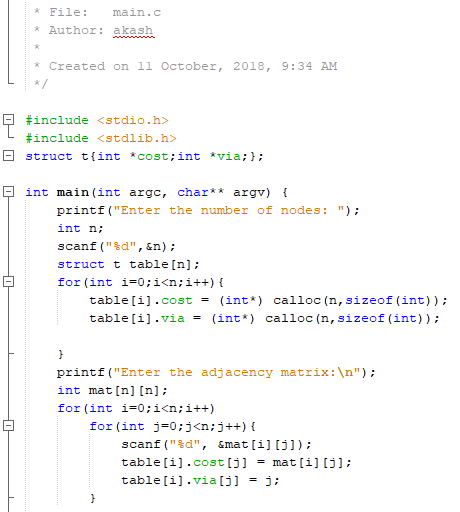
7. table[i].cost[j] = M[i][k] + M[k][j]

8. table[i].via\_node[j] = k;

9. display the cost and via\_node arrays of each node i.

End

1. **Program**



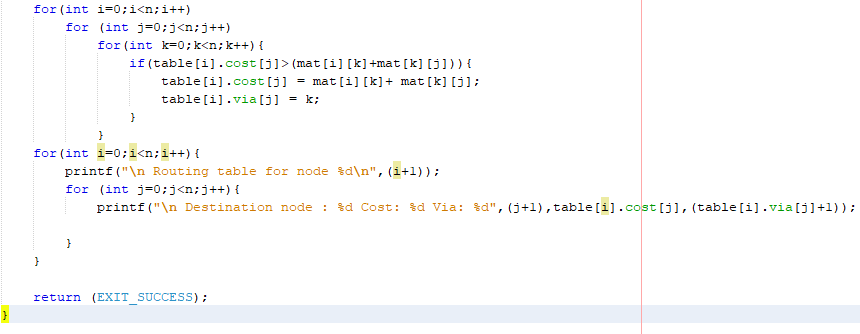


Figure 4.1 C Program for Distance Vector Routing

Figure 4.1 shows a C Program implementation of the algorithm mentioned in section 1, to implement distance vector routing for a network. The program defines a structure for a node which holds the cost and via arrays. The jth entry of the cost array denotes the cost to route a packet to destination j node. The jth entry of the via array denotes the node to which a packet must be delivered in order to send it to destination j. There are as many of these structures as there are nodes in the network, one for each node.

The program reads the number of nodes and then dynamically allocates memory for the arrays in each structure and creates an array of these structures.

It reads the network graph as an adjacency matrix. For situations where no edge exists between two nodes, the adjacency matrix has a very large value entered (99 in this case), instead of infinity.

Then it implements the algorithm stated, simply using for loops and determines the cost and node via which a packet must be delivered to send a packet from node i to node j.

1. **Results**

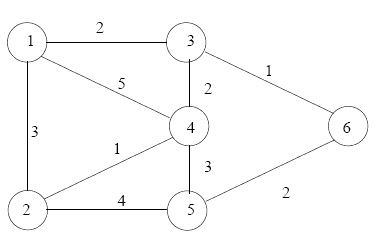


Figure 4.2 Image of network displaying edges between nodes and the cost.

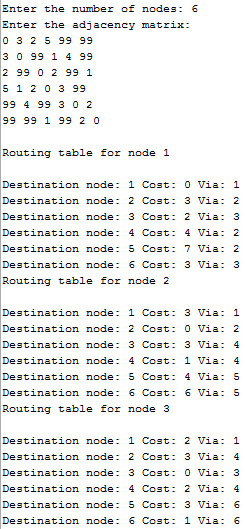
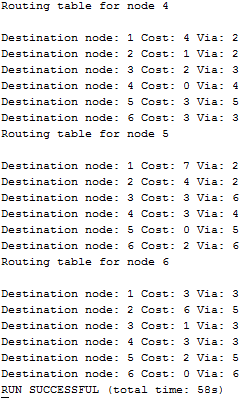
 

Figure 4.3 Results of execution of program shown in figure 4.1

Figure 4.3 shows the result of executing the program shown in figure 4.1, with input adjacency matrix describing the network shown in figure 4.2. Whenever there is no edge between node i and j, the corresponding matrix entry is entered as 99 (a high value, infinity ideally). The program then performs distance vector routing and displays the routing table for each node. For a node i, the routing table displays the cost and node via which a packet must be delivered to, in order to send it to node j.

1. **Analysis and Discussions**

Distance vector routing is a simple distributed routing protocol. Distance vector routing allows routers to automatically discover the destinations reachable inside the network as well as the shortest path to reach each of these destinations. Neighboring nodes in the subnet exchange their tables periodically to update each other on the state of the subnet, which makes this a dynamic algorithm. If a neighbor claim to have a path to a node which is shorter than the node’s current path, then that node starts using that neighbor as the route to that node.

It is classified as an adaptive routing algorithm since nodes may select a new route for each packet (even packets belonging to the same transmission) in response to changes in condition and topology of the networks.

In distance vector routing, the source knows the next hop to forward the data in order to transmit to destination. It is good in the sense that it need not know the entire network topology. A node is not aware of the full path to the destination.

However, it is not used now due to the count-to-infinity problem. In an n-node linear network topology, when a new node enters the network, it takes n exchanges to update the routing information for all nodes. However, if a node goes offline, the other nodes invent new routes to it. It is never reflected that a node has died, because the entire network topology is unknown. It takes infinity to know if a node has died.

1. **Conclusions**

In this experiment we implemented distance vector routing algorithm to compute the routing table for each node to determine the shortest path to other nodes, as well as the next hop for transmission. The given network graph was input as an adjacency matrix and the C program successfully and correctly displayed the routing table for each node.

1. **Comments**
   1. **Limitations of the experiment**

Ideally, non-existent edges are represented with cost infinity, but here it is just represented by a very large number.

* 1. **Limitations of the results obtained**

It is not a complete implementation, as the dynamic aspects to update the routing tables are not regarded.

* 1. **Learning**

Distance vector routing, its advantages and limitations

* 1. **Recommendations**

Due to the count to infinity problem, it was replaced by the link state packet routing algorithm, which allows nodes to know the entire network topology and then compute shortest paths using Djikstra’s algorithm.