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:

Register No:

1. Program

```
#include<stdio.h>
struct node
{
    unsigned dist[20];
    unsigned from[20];
}rt[10];
int main()
{
    int costmat[20][20];
    int nodes,i,j,k,count=0;
    printf("\nEnter the number of nodes : ");
    scanf("%d",&nodes);//Enter the nodes
    printf("\nEnter the cost matrix :\n");
    for(i=0;i<nodes;i++)
    {
        for(j=0;j<nodes;j++)
        {
            scanf("%d",&costmat[i][j]);
            costmat[i][i]=0;
            rt[i].dist[j]=costmat[i][j];
            rt[i].from[j]=j;
        }
    }
        do
        {
            count=0;
            for(i=0;i<nodes;i++)
            //We choose arbitary vertex k and we calculate the direct
             distance from the node i to k using the cost matrix
            //and add the distance from k to node j
            for(j=0;j<nodes;j++)
            for(k=0;k<nodes;k++)
                if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
                {
                    rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
                    rt[i].from[j]=k;
                    count++;
                }
        }while(count!=0);
        for(i=0;i<nodes;i++)
        {
            printf("\n\n For router %d\n",i+1);
```


```
            for(j=0;j<nodes;j++)
            {
printf("\t\nnode %d via %d Distance %d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);
            }
        }
    printf("\n\n");
}
```

2. Results

```
Enter the number of nodes : 3

Enter the cost matrix :
1 2 7
2 0 1
7 3 2


 For router 1

node 1 via 1 Distance 0
node 2 via 2 Distance 2
node 3 via 2 Distance 3

 For router 2

node 1 via 1 Distance 2
node 2 via 2 Distance 0
node 3 via 3 Distance 1

 For router 3

node 1 via 2 Distance 5
node 2 via 2 Distance 3
node 3 via 3 Distance 0
```


3. Algorithm/Flowchart

1. Start.
2. Create a value type struct named node having array members dist(distance) and from (distance from other nodes) and create the structure reference rt.
3. Get the user input for the number of nodes.
4. Run a nested loop to take the user input for the elements of the cost matrix. Access the structure and push the distances in the structure element. 'dist' will have all the distances and 'from' will store the position.
5. Run a do while loop, choose an arbitrary vertex, in which run nested loops to compare the initial routing distance which is to be greater than the distance from the arbitrary vertex in addition with the distance from present node to the arbitrary vertex.
6. Update the routing table distance to get the least distance.
7. Exit the while loop.
8. Print the results using loops.
9. Stop.

4. Analysis and Discussions:

Distance Vector Routing Definition?

Answer:

A distance-vector routing (DVR) protocol requires that a router inform its neighbors of topology changes periodically. Historically known as the old ARPANET routing algorithm (or known as Bellman-Ford algorithm).

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

Answer:-

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.

Disadvantages of Distance Vector routing –

- It is slower to converge than link state.
- It is at risk from the count-to-infinity problem.
- It creates more traffic than link state since a hop count change must be propagated to all routers and processed on each router. Hop count updates take place on a periodic basis, even if there are no changes in the network topology, so bandwidth-wasting broadcasts still occur.
- For larger networks, distance vector routing results in larger routing tables than link state since each router must know about all other routers. This can also lead to congestion on WAN links.

5. 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.

6. Comments:

a. Limitations of the experiment:

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

b. Limitations of the results obtained:

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

c. Learning:

   Distance vector routing, its advantages and limitations

d. 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.