PRACTICAL - 6

AIM:- Implement Distance vector routing algorithm.

INTRODUCTION:- Distance-vector routing (DVR) requires each router to notify its neighbors of topological changes on a frequent basis. The Bellman-Ford algorithm, or the old ARPANET routing mechanism, is another name for it.

Characteristic of Distance Vector Routing Protocol:

1. The Distance vector algorithm is distributed, iterative, and asynchronous.
2. It is distributed in the sense that each node gets data from one or more of its directly associated neighbors, performs calculations, and then distributes the results to the rest of the network.
3. Iterative: The procedure is iterative because it continues until no more information can be transferred between neighbors.
4. It is asynchronous in that it does not require all of its nodes to function in lockstep with one another.
5. A dynamic algorithm is the Distance Vector Algorithm.
6. ARPANET and RIP are two of the most common applications.

CODE:-

```
#include <stdio.h>

struct node
{
    unsigned dist[20];
    unsigned from[20];
} rt[10];
int main()
{
    printf("Nikhil gupta\n");
    printf("2K20/MC/86\n");
```

```
    int costmat[20][20];
    int nodes, i, j, k, count = 0;
    printf("\nEnter the number of nodes : ");
    scanf("%d", &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++)
            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");
}
```

OUTPUT:-

```
Nikhil Gupta
2K20/MC/86

Enter the number of nodes : 4

Enter the cost matrix :
1 3 5 3
2 3 4 3
2 4 5 7
4 6 4 3


 For router 1

node 1 via 1 Distance 0
node 2 via 2 Distance 3
node 3 via 3 Distance 5
node 4 via 4 Distance 3

 For router 2

node 1 via 1 Distance 2
node 2 via 2 Distance 0
node 3 via 3 Distance 4
node 4 via 4 Distance 3

 For router 3

node 1 via 1 Distance 2
node 2 via 2 Distance 4
node 3 via 3 Distance 0
node 4 via 1 Distance 5

 For router 4

node 1 via 1 Distance 4
node 2 via 2 Distance 6
node 3 via 3 Distance 4
node 4 via 4 Distance 0
```