College of Engineering,Trivandrum

Experiment 11

Network Programming Lab

Author:
Alan Anto

Registration Number :
TVE16CS09

March 27, 2019

Contents

1 Distance Vector Routing Protocol

1.1 AIM

Implement and simulate algorithm for Distance vector routing protocol.

1.2 Theory

1.2.1 Distance Vector Routing Protocol

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).

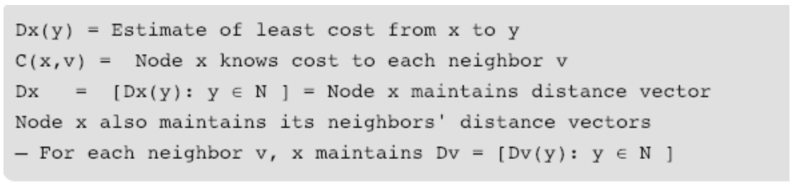
Each router maintains a Distance Vector table containing the distance between itself and ALL possible destination nodes. Distances,based on a chosen metric, are computed using information from the neighbors' distance vectors. A distance-vector routing protocol in data networks determines the best route for data packets based on distance. Distance-vector routing protocols measure the distance by the number of routers a packet has to pass, one router counts as one hop. Some distance-vector protocols also take into account network latency and other factors that influence traffic on a given route. To determine the best route across a network routers, on which a distance-vector protocol is implemented, exchange information with one another, usually routing tables plus hop counts for destination networks and possibly other traffic information. Distance-vector routing protocols also require that a router informs its neighbours of network topology changes periodically.

Routers that use distance-vector protocol determine the distance between themselves and a destination. The best route for Internet Protocol packets that carry data across a data network is measured in terms of the numbers of routers (hops) a packet has to pass to reach its destination network. Additionally some distance-vector protocols take into account other traffic information, such as network latency. To establish the best route, routers regularly exchange information with neighbouring routers, usually their routing table, hop count for a destination network and possibly other traffic related information. Routers that implement distance-vector protocol rely purely on the information provided to them by other routers, and do not assess the network topology.

2 Algorithm

1. A router transmits its distance vector to each of its neighbors in a routing packet.
2. Each router receives and saves the most recently received distance vector from each of its neighbors.
3. A router recalculates its distance vector when:
4 It receives a distance vector from a neighbor containing different information than before.
5 It discovers that a link to a neighbor has gone down.

The DV calculation is based on minimizing the cost to each destination

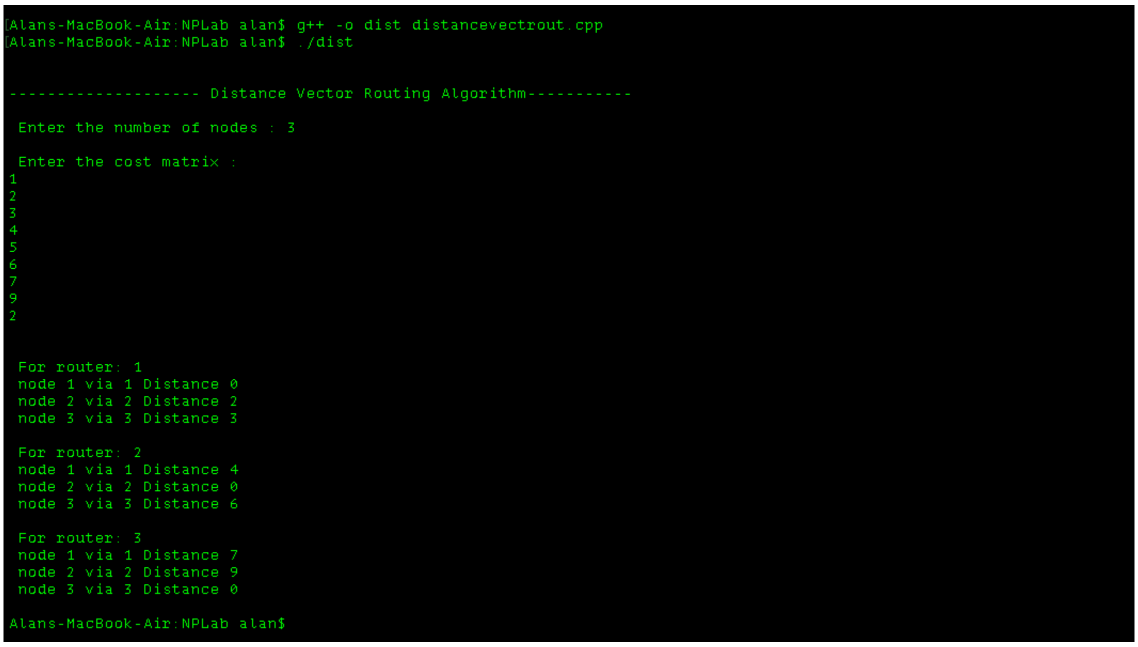


```
Dx(y) = Estimate of least cost from x to y
C(x,v) =  Node x knows cost to each neighbor v
Dx    =  [Dx(y): y ∈ N ] = Node x maintains distance vector
Node x also maintains its neighbors' distance vectors
— For each neighbor v, x maintains Dv = [Dv(y): y ∈ N ]
```

3 Program

```
#include<stdio.h>
#include<iostream>
using namespace std;
struct node
{
    unsigned dist[6];
    unsigned from[6];
}DVR[10];
int main()
{
    cout<<"\n\n------------------- Distance Vector Routing Algorithm-----
    int costmat[6][6];
    int nodes, i, j, k;
    cout<<"\n\n Enter the number of nodes : ";
    cin>>nodes; //Enter the nodes
    cout<<"\n Enter the cost matrix : \n" ;
    for(i = 0; i < nodes; i++)
    {
        for(j = 0; j < nodes; j++)
        {
            cin>>costmat[i][j];
            costmat[i][i] = 0;
            DVR[i].dist[j] = costmat[i][j];
            DVR[i].from[j] = j;
        }
    }
            for(i = 0; i < nodes; i++)
            for(j = i+1; j < nodes; j++)
            for(k = 0; k < nodes; k++)
                if(DVR[i].dist[j] > costmat[i][k] + DVR[k].dist[j])
                {   //We calculate the minimum distance
                    DVR[i].dist[j] = DVR[i].dist[k] + DVR[k].dist[j];
                    DVR[j].dist[i] = DVR[i].dist[j];
                    DVR[i].from[j] = k;
                    DVR[j].from[i] = k;
                }
        for(i = 0; i < nodes; i++)
        {
            cout<<"\n\n For router: "<<i+1;
            for(j = 0; j < nodes; j++)
                cout<<"\t\n node "<<j+1<<" via "<<DVR[i].from[j]+1<<" Dis
        }
    cout<<" \n\n ";
    return 0;
}
```

4 Output



```
[Alans-MacBook-Air:NPLab alan$ g++ -o dist distancevectrout.cpp
[Alans-MacBook-Air:NPLab alan$ ./dist

-------------------- Distance Vector Routing Algorithm-----------

 Enter the number of nodes : 3

 Enter the cost matrix :
1
2
3
4
5
6
7
9
2


 For router: 1
 node 1 via 1 Distance 0
 node 2 via 2 Distance 2
 node 3 via 3 Distance 3

 For router: 2
 node 1 via 1 Distance 4
 node 2 via 2 Distance 0
 node 3 via 3 Distance 6

 For router: 3
 node 1 via 1 Distance 7
 node 2 via 2 Distance 9
 node 3 via 3 Distance 0

Alans-MacBook-Air:NPLab alan$
```

5 Result

Implemented Distance Vectore routing algorithm in C++ and compiled using version 4.2.1 on macOS Mojave . The output was obtained and verified.

In this program the nodes find out the distance between each of the nodes initially from the router. Later on the shortest distance possible is calculated , this can be done by making use of any the paths possible. Later on the distance between each of the nodes, traversing through different routes is produced.