Networking Lab Assignment 11

Distance vector routing protocol

Albin Antony

28 March 2019

1 Distance vector routing protocol

1.1 Aim

Implement and simulate algorithm for Distance vector routing protocol.

1.2 Theory

A distance-vector routing (DVR) protocol requires that a router inform its neighbors of topology changes periodically. 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. DVR uses Bellman-Ford's Algorithm.

1.3 Algorithm

Algorithm 1 Algorithm for Distance Vector Routing Protocol

```
Input a matrix, cost of size N X N.
Initialize a matrix, d of size N X N with values of cost
.
Iterate through each node i.
    Iterate through each node j.
* Iterate through each node k.
       d[i][j]=min(d[i][j],cost[i][k]+d[k][j])

Display d[i][1..N] for all node i.
```

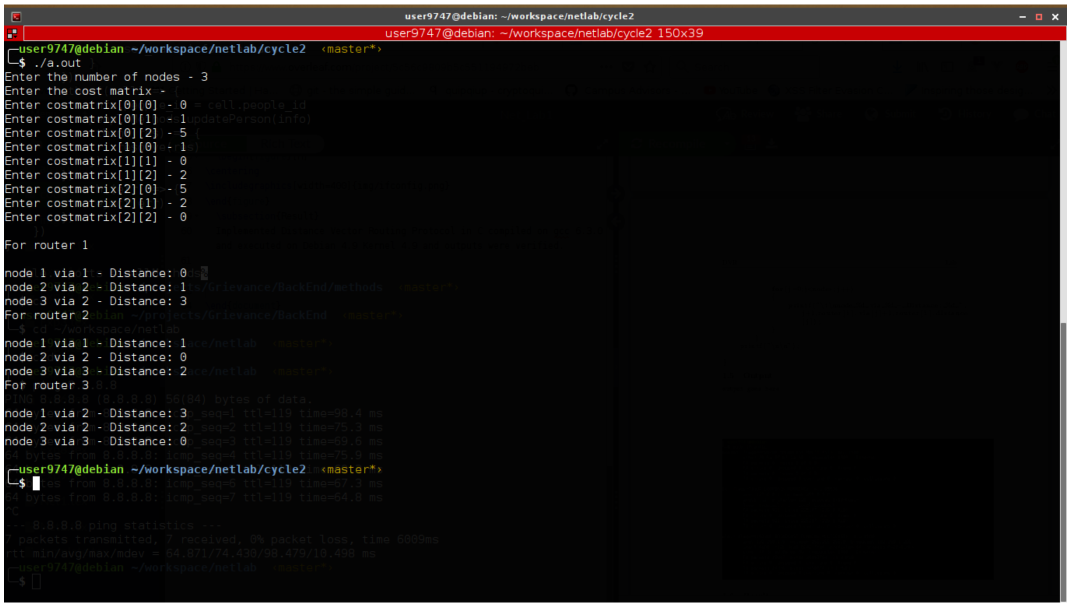
1.4 Program

```
#include<stdio.h>
struct node
{
    unsigned distance[20];
    unsigned via[20];
}router[10];
int main()
{
    int costmatrix[20][20];
    int nodes,i,j,k,count=0;
    printf("Enter the number of nodes - ");
    scanf("%d",&nodes);//Enter the nodes
    printf("Enter the cost matrix -\n");
    for(i=0;i<nodes;i++)
    {
        for(j=0;j<nodes;j++)
        {
            printf("Enter costmatrix[%d][%d] - ",i,j);
            scanf("%d",&costmatrix[i][j]);
            costmatrix[i][i]=0;
            router[i].distance[j]=costmatrix[i][j];//
                initialising the distance equal to cost
                matrix
            router[i].via[j]=j;//initialising the via part
        }
    }
        do
        {
            count=0;
            for(i=0;i<nodes;i++)
            for(j=0;j<nodes;j++)
            for(k=0;k<nodes;k++)
                if(router[i].distance[j]>costmatrix[i][k]+
                    router[k].distance[j])
                {//Calculating the minimum distance
                    router[i].distance[j]=router[i].distance
                        [k]+router[k].distance[j];
                    router[i].via[j]=k;
                    count++;
                }
        }while(count!=0);
        for(i=0;i<nodes;i++)
        {
            printf("\nFor router %d\n",i+1);
```

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

}
```

1.5 Output



```
user9747@debian: ~/workspace/netlab/cycle2
user9747@debian: ~/workspace/netlab/cycle2 150x39
┌─user9747@debian ~/workspace/netlab/cycle2  ‹master*›
└─$ ./a.out
Enter the number of nodes - 3
Enter the cost matrix -
Enter costmatrix[0][0] - 0
Enter costmatrix[0][1] - 1
Enter costmatrix[0][2] - 5
Enter costmatrix[1][0] - 1
Enter costmatrix[1][1] - 0
Enter costmatrix[1][2] - 2
Enter costmatrix[2][0] - 5
Enter costmatrix[2][1] - 2
Enter costmatrix[2][2] - 0

For router 1

node 1 via 1 - Distance: 0
node 2 via 2 - Distance: 1
node 3 via 2 - Distance: 3
For router 2
node 1 via 1 - Distance: 1
node 2 via 2 - Distance: 0
node 3 via 3 - Distance: 2
For router 3
node 1 via 2 - Distance: 3
node 2 via 2 - Distance: 2
node 3 via 3 - Distance: 0
┌─user9747@debian ~/workspace/netlab/cycle2  ‹master*›
└─$
```

1.6 Result

Implemented Distance Vector Routing Protocol in C compiled on gcc 6.3.0 and executed on Debian 4.9 Kernel 4.9 and outputs were verified.