College of Engineering,Trivandrum

Experiment 12

Network Programming Lab

Author:
Alan Anto

Registration Number :
TVE16CS09

April 3, 2019

Contents

1 Link state routing protocol

1.1 AIM

Implement and simulate algorithm for Link state routing protocol.

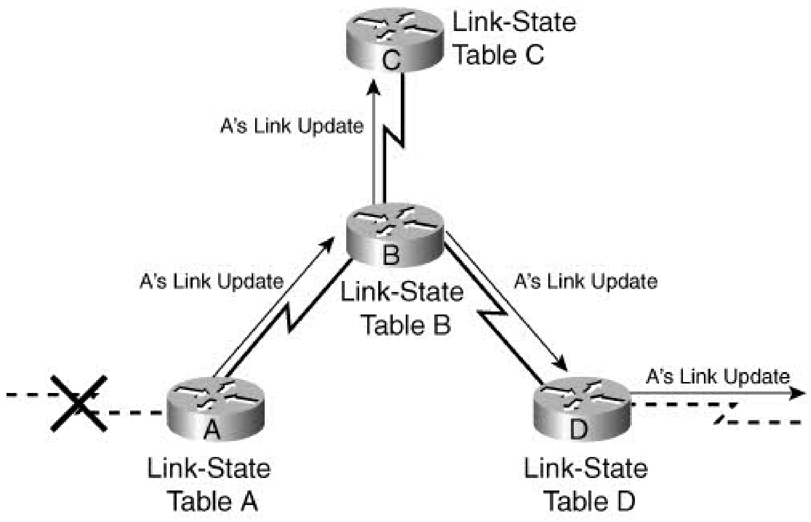
1.2 Theory

1.2.1 Link state routing protocol

Link state routing is the second family of routing protocols. While distance vector routers use a distributed algorithm to compute their routing tables, link-state routing uses link-state routers to exchange messages that allow each router to learn the entire network topology. Based on this learned topology, each router is then able to compute its routing table by using a shortest path computation.

Features of link state routing protocols –

Link state packet – A small packet that contains routing information.
Link state database – A collection information gathered from link state packet.
Shortest path first algorithm (Dijkstra algorithm) – A calculation performed on the database results into shortest path
Routing table – A list of known paths and interfaces.

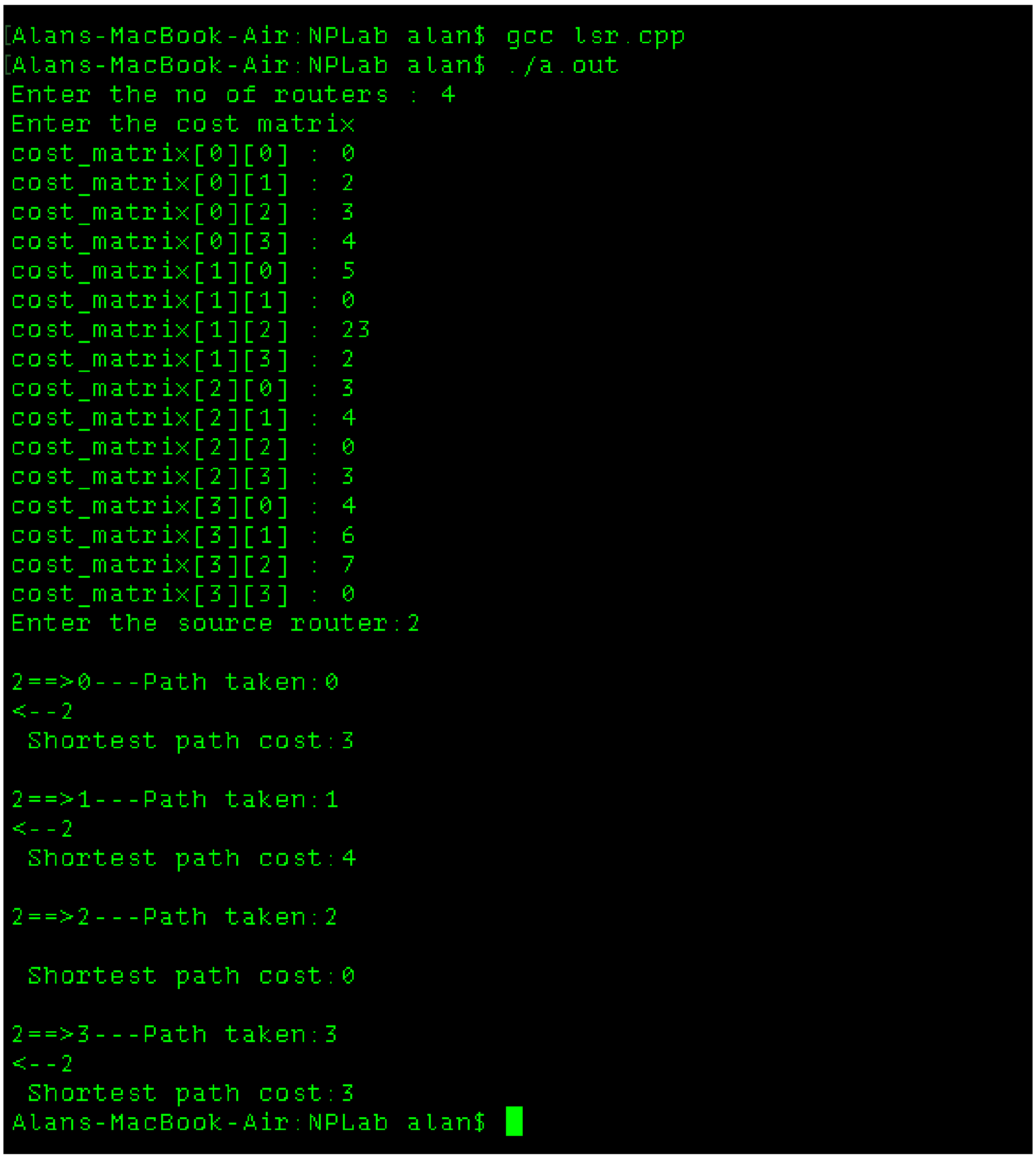


2 Program

```
#include <stdio.h>
#include <string.h>
int main()
{
    int count,source,i,j,k,w,v,min;
    int cost_matrix[100][100],distance[100],last[100];
    int flag[100];
    printf("Enter the no of routers : ");
    scanf("%d",&count);
    printf("Enter the cost matrix\n");
    for(i=0;i<count;i++)
    {
        for(j=0;j<count;j++)
        {
            printf("cost_matrix[%d][%d] : ",i,j);
            scanf("%d",&cost_matrix[i][j]);
            if(cost_matrix[i][j]<0)cost_matrix[i][j]=1000;
        }
    }
    printf("Enter the source router:");
    scanf("%d",&source);
    for(v=0;v<count;v++)
    {
        flag[v]=0;
        last[v]=source;
        distance[v]=cost_matrix[source][v];
    }
    flag[source]=1;
    for(i=0;i<count;i++)
    {
        min=1000;
        for(w=0;w<count;w++)
        {
            if(!flag[w])
                if(distance[w]<min)
                {
```

```
                v=w;
                min=distance[w];
            }
        }
        flag[v]=1;
        for(w=0;w<count;w++)
        {
            if(!flag[w])
                if(min+cost_matrix[v][w]<distance[w])
                {
                    distance[w]=min+cost_matrix[v][w];
                    last[w]=v;
                }
            }
        }
        for(i=0;i<count;i++)
        {
            printf("\n%d==>%d----Path taken:%d \n",source,i,i);
            w=i;
            while(w!=source)
            {
                printf("<--%d",last[w]);w=last[w];
            }
            printf("\n Shortest path cost:%d \n",distance[i]);
        }
    }
```

3 Output



```
[Alans-MacBook-Air:NPLab alan$ gcc lsr.cpp
[Alans-MacBook-Air:NPLab alan$ ./a.out
Enter the no of routers : 4
Enter the cost matrix
cost_matrix[0][0] : 0
cost_matrix[0][1] : 2
cost_matrix[0][2] : 3
cost_matrix[0][3] : 4
cost_matrix[1][0] : 5
cost_matrix[1][1] : 0
cost_matrix[1][2] : 23
cost_matrix[1][3] : 2
cost_matrix[2][0] : 3
cost_matrix[2][1] : 4
cost_matrix[2][2] : 0
cost_matrix[2][3] : 3
cost_matrix[3][0] : 4
cost_matrix[3][1] : 6
cost_matrix[3][2] : 7
cost_matrix[3][3] : 0
Enter the source router:2

2==>0---Path taken:0
<--2
 Shortest path cost:3

2==>1---Path taken:1
<--2
 Shortest path cost:4

2==>2---Path taken:2

 Shortest path cost:0

2==>3---Path taken:3
<--2
 Shortest path cost:3
Alans-MacBook-Air:NPLab alan$
```

4 Result

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