16Experiment 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: 16ETCS002011

Name: Anirudha P Kakrannaya

1. Algorithm/Flowchart

   Algorithm for Distance vector Routing

   Step 1: Start

   Step 2: Give graph as adjacency matrix to program w(u,v)

   Step 3: Find all edges in graph and store it in edge array. E[G]

   Step 4: Considering source vertex, assign it the cost zero. Add all the vertices to a list. For all vertices except source vertex assign a cost infinity.

   Step 5: Take one vertex at a time say and relax all the edges in the graph. Point worth noticing is that we can only relax the edges which are outgoing from the selected vertex.

   Step 6: For each edge (u, *v*) in E[G] do

   if d[u] + *w* (u, *v*) < d[*v*] then

   return FALSE

   return TRUE

   Step 7: Stop

2. Program

```
int main(){
    int V=5,edge[20][2],i,j,k=0;
    char ar[5]={'A','B','C','D','E'};
    printf("Bellman Algorithm for Router \n");
    printf("Number of router:%d\n",V);
    printf("Graph in matrix form:\n");
    int G[5][5]={{0, 1, 5, 0, 0},
    {1, 0, 3, 4, 0},
    {5, 3, 0, 0, 9},
    {0, 4, 0, 0, 1},
    {0, 0, 9, 1, 0}
};
    for(i=0;i<V;i++){
        for(j=0;j<V;j++){
            printf("%d\t",G[i][j]);
            if(G[i][j]!=0){
                edge[k][0]=i;
                edge[k++][1]=j;
            }
        }printf("\n");}
    if(!Bellman_Ford(G,V,k,edge,ar))
    printf("\nNegative weight cycle exists\n");
    return 0;
}
```


Name: Anirudha P Kakrannaya
Reg.no:16ETCS002011

Here input for graph is done using Adjacency matrix and initial matrix is calculated and removing loop are done here if there is any loop. Weights are added to edge matrix. And Bellamn_ford() function is called along with parameter Graph matrix, number of vertices ,edge matrix and router names are passed.

```
int Bellman_Ford(int G[5][5] , int V, int E, int edge[20][2],char ar[6]){
    int final[10][10],flag=1;
    for(int p=0;p<V;p++){
     char   c=ar[p];
     int next;
        ar[-1]='-';
    int i,u,v,k,distance[20],parent[20],S;
    for(i=0;i<V;i++)
        distance[i] = 1000 , parent[i] = -1 ;
        for(int i=0;i<6;i++){
            char check=ar[i];
            if(c==check)
                S=i+1; }
        distance[S-1]=0 ;
    for(i=0;i<V-1;i++){
        for(k=0;k<E;k++){
            u = edge[k][0] , v = edge[k][1] ;
            if(distance[u]+G[u][v] < distance[v])
                distance[v] = distance[u] + G[u][v] , parent[v]=u ;}
    }
    for(k=0;k<E;k++) {
            u = edge[k][0] , v = edge[k][1] ;
            if(distance[u]+G[u][v] < distance[v])
                flag = 0 ;
        }
```

Here v-1 times iteration performed for V vertices. Distance matrix is used to store shortest distance from source. This operation performed for all vertices

```
    printf("\nShortest path from Source %c is\n",ar[S-1]);
    printf("=========================================\n");
    if(flag){
        printf("Router\tCost\tParent\tPath");
        for(i=0;i<V;i++){
            int m=i,a[10],o=9;
            int some;
            a[o--]=i;
             next=parent[i];
            a[o--]=next;
            while(next!=-1){
                next=parent[next];
                a[o--]=next;}
         printf("\n");
```

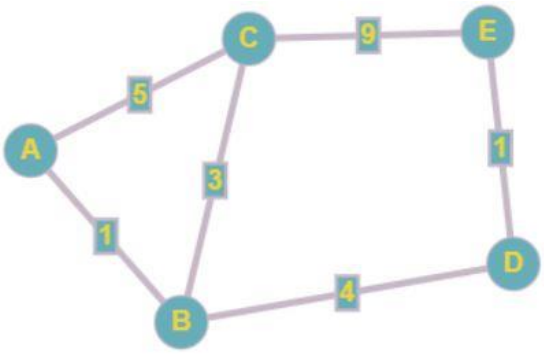

Name: Anirudha P Kakrannaya
Reg.no:16ETCS002011

```
            if(distance[i]==1000){
                printf("%c\t%c\t%c\t",
                        ar[i],ar[-1],ar[parent[i]]);
                final [p][i]=0;
            }else{
            printf("%c\t%d\t%c\t",
                    ar[i],distance[i],ar[parent[i]]);
            final[p][i]=distance[i];}
           for(int h=10;h>0;h--){
               if(a[h]==-1){
                   for(int g=h+1;g<10;g++){
                       if(g==9){
                        printf("%c",ar[a[g]]);
                       }else
                       printf("%c-->",ar[a[g]]);}
                    break;}}
        }
        printf("\n=========================================");
```

Displaying source router vertex and possible shortest path from source. If cannot reach other router represented by '-'means no path.

3. Results

```
Bellman Algorithm for Router
Number of router:5
Graph in matrix form:
0       1       5       0       0
1       0       3       4       0
5       3       0       0       9
0       4       0       0       1
0       0       9       1       0
```



Adjacency matrix of taken graph entered in program.

Name: Anirudha P Kakrannaya
Reg.no:16ETCS002011

```
Shortest path from Source A is
========================================
Router  Cost    Parent  Path
A       0       -       A
B       1       A       A-->B
C       4       B       A-->B-->C
D       5       B       A-->B-->D
E       6       D       A-->B-->D-->E
========================================
Shortest path from Source B is
========================================
Router  Cost    Parent  Path
A       1       B       B-->A
B       0       -       B
C       3       B       B-->C
D       4       B       B-->D
E       5       D       B-->D-->E
========================================
Shortest path from Source C is
========================================
Router  Cost    Parent  Path
A       4       B       C-->B-->A
B       3       C       C-->B
C       0       -       C
D       7       B       C-->B-->D
E       8       D       C-->B-->D-->E
========================================
----------------------------------------
Shortest path from Source D is
========================================
Router  Cost    Parent  Path
A       5       B       D-->B-->A
B       4       D       D-->B
C       7       B       D-->B-->C
D       0       -       D
E       1       D       D-->E
========================================
Shortest path from Source E is
========================================
Router  Cost    Parent  Path
A       6       B       E-->D-->B-->A
B       5       D       E-->D-->B
C       8       B       E-->D-->B-->C
D       1       E       E-->D
E       0       -       E
========================================
Final adjacency matrix using Bellman-Ford algorithm
0       1       4       5       6
1       0       3       4       5
4       3       0       7       8
5       4       7       0       1
6       5       8       1       0

RUN SUCCESSFUL (total time: 183ms)
```

All shortest path from each router is displayed and final cost matrix is displayed to user.

4. Analysis and Discussions

Distance Vector Routing Protocol is one of two major routing protocols for communications methods that use data packets sent over Internet Protocol. Distance Vector Routing Protocol requires routing hardware to report the distances of various nodes within a network or IP topology to determine the best and most efficient routes for data packets.

Adaptive Routing is based on current measurements of traffic and topology.
1. Centralized 2. Isolated 3. Distributed

Distance vector routing comes under distributed adaptive Routing. It is considered as very adaptive because

- Stable and proven method distance vector was the original routing algorithm

Name: Anirudha P Kakrannaya
Reg.no:16ETCS002011

- Easy to implement and administer
- Bandwidth requirements negligible for a typical LAN environment
- Requires less hardware and processing power than other routing methods

Distance Vector is a relatively simple approach and easy to use, implement and maintain and does not require High-level knowledge to deploy. Moreover, it does not demand high bandwidth level to send their periodic updates as the size of the packets are relatively small. Furthermore, distance vector protocols do not require a large amount of CPU resources or memory to store the routing data.

5. Conclusions

C program for Distance vector protocol is developed using bellman ford algorithm and verified with results.

6. Comments

a. Limitations of the experiment

Routers must recalculate their routing tables before forwarding changes

b. Limitations of the results obtained
Negative cycles possible if negative edge is present practically it is not possible but in adjacent matrix if we enter negative then it is going to be problem.

c. Learning
Learnt the working od Distance Vector Routing protocol and C program is created.

d. Recommendations
This program further modified to a greater number of routers in my program it works for only maximum of 6 vertices.

Name: Anirudha P Kakrannaya
Reg.no:16ETCS002011