Lab Assignment 3
Network and Communication

Name: Kulvir Singh
Reg. No.: 19BCE2074

Question 1:

Implement the Distance Vector Routing Protocol

Aim :

To implement the distance vector routing protocol using C programming language and display the output accordingly.

Algorithm :

A router transmits its distance vector to each of its neighbors in a routing packet. Each router receives and saves the most recently received distance vector from each of its neighbors. A router recalculates its distance vector when:

It receives a distance vector from a neighbor containing different information than before. 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]

Code Text :

```
#include<stdio.h>
struct node
{
unsigned dist[20];
unsigned from[20];
}rt[10];
int main()
{
```

```
int costmat[20][20];
int nodes,i,j,k,count=0;
printf("\n Kulvir Singh \n 19BCE2074 \n\n DISTANCE VECTOR PROTOCOL \n");
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];//initialise the distance equal to cost matrix
rt[i].from[j]=j;
}
}
do
{
count=0;
for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the direct distance from the node i
to k using the cost matrix
//and add the distance from k to node j
for(j=0;j<nodes;j++)
for(k=0;k<nodes;k++)
if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
{//We calculate the minimum distance
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");
getch();
}
```

Code Screenshots :

```c
#include<stdio.h>
struct node
{
unsigned dist[20];
unsigned from[20];
}rt[10];
int main()
{
int costmat[20][20];
int nodes,i,j,k,count=0;
printf("\n Kulvir Singh \n 19BCE2074 \n\n DISTANCE VECTOR PROTOCOL \n");
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];//initialise the distance equal to cost matrix
rt[i].from[j]=j;
}
}
do
{
count=0;
for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the direct dis
//and add the distance from k to node j
for(j=0;j<nodes;j++)
for(k=0;k<nodes;k++)
```

```c
31    for(k=0;k<nodes;k++)
32    if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
33    {//We calculate the minimum distance
34    rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
35    rt[i].from[j]=k;
36    count++;
37    }
38    }while(count!=0);
39    for(i=0;i<nodes;i++)
40    {
41    printf("\n\n For router %d\n",i+1);
42    for(j=0;j<nodes;j++)
43    {
44    printf("\t\nnode %d via %d Distance %d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);
45    }
46    }
47    printf("\n\n");
48    getch();
49    }
```

Output Screenshots :

```
kulvir06@ubuntu:~/Desktop/net com$ gcc dv.c -o dv
kulvir06@ubuntu:~/Desktop/net com$ ./dv

 Kulvir Singh
 19BCE2074

 DISTANCE VECTOR PROTOCOL

Enter the number of nodes : 3

Enter the cost matrix :
0 2 5
9 0 2
5 4 0


 For router 1

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

 For router 2

node 1 via 3 Distance 7
node 2 via 2 Distance 0
node 3 via 3 Distance 2

 For router 3

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

kulvir06@ubuntu:~/Desktop/net com$
```

Question 2:

Implement the Link State Routing Protocol

Aim :

To implement the link state routing protocol using C programming language and display the output accordingly.

Algorithm :

```
function dijkstra(G, S)
   for each vertex V in G
      distance[V] <- infinite
      previous[V] <- NULL
      If V != S, add V to Priority Queue Q
   distance[S] <- 0

   while Q IS NOT EMPTY
      U <- Extract MIN from Q
      for each unvisited neighbour V of U
         tempDistance <- distance[U] + edge_weight(U, V)
         if tempDistance < distance[V]
            distance[V] <- tempDistance
            previous[V] <- U
   return distance[], previous[]
```

Code Text :

```
#include <stdio.h>
#include <string.h>
int main()
{
int count,src_router,i,j,k,w,v,min;
int cost_matrix[100][100],dist[100],last[100];
int flag[100];
printf("\n Kulvir Singh \n 19BCE2074 \n Link State Protocol \n\n");
printf("\n Enter the no of routers:\t");
scanf("%d",&count);
printf("\n Enter the cost matrix values:");
for(i=0;i<count;i++)
{
for(j=0;j<count;j++)
{
printf("\n%d->%d:",i,j);
scanf("%d",&cost_matrix[i][j]);
if(cost_matrix[i][j]<0)cost_matrix[i][j]=1000;
```

```
}
}
printf("\n Enter the source router:");
scanf("%d",&src_router);
for(v=0;v<count;v++)
{
flag[v]=0;
last[v]=src_router;
dist[v]=cost_matrix[src_router][v];
}
flag[src_router]=1;
for(i=0;i<count;i++)
{
min=1000;
for(w=0;w<count;w++)
{
if(!flag[w])
if(dist[w]<min)
{
        v=w;
min=dist[w];
}
}
flag[v]=1;
for(w=0;w<count;w++)
{
if(!flag[w])
if(min+cost_matrix[v][w]<dist[w])
{
dist[w]=min+cost_matrix[v][w];
last[w]=v;
}
}
}
for(i=0;i<count;i++)
{
printf("\n%d==>%d:Path taken:%d",src_router,i,i);
w=i;
while(w!=src_router)
{
printf("\n<--%d",last[w]);w=last[w];
}
printf("\n Shortest path cost:%d",dist[i]);
}
}
```

Code Screenshots :

```c
#include <stdio.h>
#include <string.h>
int main()
{
int count,src_router,i,j,k,w,v,min;
int cost_matrix[100][100],dist[100],last[100];
int flag[100];
printf("\n Kulvir Singh \n 19BCE2074 \n Link State Protocol \n\n");
printf("\n Enter the no of routers:\t");
scanf("%d",&count);
printf("\n Enter the cost matrix values:");
for(i=0;i<count;i++)
{
for(j=0;j<count;j++)
{
printf("\n%d->%d:",i,j);
scanf("%d",&cost_matrix[i][j]);
if(cost_matrix[i][j]<0)cost_matrix[i][j]=1000;
}
}
printf("\n Enter the source router:");
scanf("%d",&src_router);
for(v=0;v<count;v++)
{
flag[v]=0;
last[v]=src_router;
dist[v]=cost_matrix[src_router][v];
}
flag[src_router]=1;
for(i=0;i<count;i++)
{
```

```
min=1000;
for(w=0;w<count;w++)
{
if(!flag[w])
if(dist[w]<min)
{
    v=w;
min=dist[w];
}
}
flag[v]=1;
for(w=0;w<count;w++)
{
if(!flag[w])
if(min+cost_matrix[v][w]<dist[w])
{
dist[w]=min+cost_matrix[v][w];
last[w]=v;
}
}
}
for(i=0;i<count;i++)
{
printf("\n%d==>%d:Path taken:%d",src_router,i,i);
w=i;
while(w!=src_router)
{
printf("\n<--%d",last[w]);w=last[w];
}
printf("\n Shortest path cost:%d",dist[i]);
}
}
```

Output Screenshots :

```
 Kulvir Singh
 19BCE2074
 Link State Protocol


 Enter the no of routers:       3

 Enter the cost matrix values:
0->0:0

0->1:2

0->2:3

1->0:1

1->1:0

1->2:5

2->0:1

2->1:1

2->2:0

 Enter the source router:2

2==>0:Path taken:0
<--2
 Shortest path cost:1

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

2==>2:Path taken:2
 Shortest path cost:0
kulvir06@ubuntu:~/Desktop/net com$
```

Question 3:

Show the Performance Evaluation between Distance Vector Routing Protocol and Link State Routing Protocol

Aim :

To implement the distance vector protocol and link state protocol and give a comparative study, display the output accordingly in any programming language.

Algorithm :

Implement the same algorithms as given in distance vector part and link state part and display results accordingly.

Code Text :

```
#include<stdio.h>
#include<string.h>
struct node
{
unsigned dist[20];
unsigned from[20];
}rt[10];
int main()
{
printf("\n Kulvir Singh \n 19BCE2074 \n Comparison of protocols \n\n\n Distance Vector\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];//initialise the distance equal to cost matrix
rt[i].from[j]=j;
}
}
do
{
count=0;
for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the direct distance from the node i to k using the cost matrix
//and add the distance from k to node j
```

```
for(j=0;j<nodes;j++)
for(k=0;k<nodes;k++)
if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
{//We calculate the minimum distance
rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
rt[i].from[j]=k;
count++;
}
}while(count!=0);
int sourceRouter = 0;
printf("\nEnter source router : \n");
scanf("%d",&sourceRouter);
for(j=0;j<nodes;j++)
{
printf("\t\nnode %d via %d Distance %d ",j+1,rt[sourceRouter].from[j]+1,rt[sourceRouter].dist[j]);
}
printf("\n\n");
//
//
//
//
//
printf("\n\nLink State Protocol");
count =0; i=0; j=0; k=0;
int src_router,w,v,min;
int cost_matrix[100][100],dist[100],last[100];
int flag[100];

printf("\n Enter the no of routers:\t");
scanf("%d",&count);
printf("\n Enter the cost matrix values:");
for(i=0;i<count;i++)
{
for(j=0;j<count;j++)
{
printf("\n%d->%d:",i,j);
scanf("%d",&cost_matrix[i][j]);
if(cost_matrix[i][j]<0)cost_matrix[i][j]=1000;
}
}
printf("\n Enter the source router:");
scanf("%d",&src_router);
for(v=0;v<count;v++)
{
flag[v]=0;
last[v]=src_router;
dist[v]=cost_matrix[src_router][v];
}
```

```
flag[src_router]=1;
for(i=0;i<count;i++)
{
min=1000;
for(w=0;w<count;w++)
{
if(!flag[w])
if(dist[w]<min)
{
            v=w;
min=dist[w];
}
}
flag[v]=1;
for(w=0;w<count;w++)
{
if(!flag[w])
if(min+cost_matrix[v][w]<dist[w])
{
dist[w]=min+cost_matrix[v][w];
last[w]=v;
}
}
}
for(i=0;i<count;i++)
{
printf("\n%d==>%d:Path taken:%d",src_router,i,i);
w=i;
while(w!=src_router)
{
printf("\n<--%d",last[w]);w=last[w];
}
printf("\n Shortest path cost:%d\n",dist[i]);
}
}
```

Code Screenshots :

```c
#include<stdio.h>
#include<string.h>
struct node
{
unsigned dist[20];
unsigned from[20];
}rt[10];
int main()
{
printf("\n Kulvir Singh \n 19BCE2074 \n Comparison of protocols \n\n\n Distance Vector\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];//initialise the distance equal to cost matrix
rt[i].from[j]=j;
}
}
do
{
count=0;
for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the direct distance from the no
//and add the distance from k to node j
for(j=0;j<nodes;j++)
for(k=0;k<nodes;k++)
if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
{//We calculate the minimum distance
rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
rt[i].from[j]=k;
count++;
}
}while(count!=0);
int sourceRouter = 0;
printf("\nEnter source router : \n");
scanf("%d",&sourceRouter);
for(j=0;j<nodes;j++)
{
```

```
for(j=0;j<nodes;j++)
{
printf("\t\nnode %d via %d Distance %d ",j+1,rt[sourceRouter].from[j]+1,rt[sourceRouter].dist[j]);
}
printf("\n\n");
//
//
//
//
//
printf("\n\nLink State Protocol");
count =0; i=0; j=0; k=0;
int src_router,w,v,min;
int cost_matrix[100][100],dist[100],last[100];
int flag[100];

printf("\n Enter the no of routers:\t");
scanf("%d",&count);
printf("\n Enter the cost matrix values:");
for(i=0;i<count;i++)
{
for(j=0;j<count;j++)
{
printf("\n%d->%d:",i,j);
scanf("%d",&cost_matrix[i][j]);
if(cost_matrix[i][j]<0)cost_matrix[i][j]=1000;
}
}
printf("\n Enter the source router:");
scanf("%d",&src_router);
for(v=0;v<count;v++)
{
flag[v]=0;
last[v]=src_router;
dist[v]=cost_matrix[src_router][v];
}
flag[src_router]=1;
for(i=0;i<count;i++)
{
min=1000;
for(w=0;w<count;w++)
{
if(!flag[w])
if(dist[w]<min)
```

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

Output Screenshots :

```
 Kulvir Singh
 19BCE2074
 Comparison of protocols


 Distance Vector

Enter the number of nodes : 3

Enter the cost matrix :
0 1 2
2 0 5
6 7 0

Enter source router :
1

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



Link State Protocol
 Enter the no of routers:       3

 Enter the cost matrix values:
0->0:0

0->1:1

0->2:2

1->0:2

1->1:0

1->2:5

2->0:6

2->1:7

2->2:0
```

```
 Enter the source router:1

1==>0:Path taken:0
<--1
 Shortest path cost:2

1==>1:Path taken:1
 Shortest path cost:0

1==>2:Path taken:2
<--0
<--1
 Shortest path cost:4
kulvir06@ubuntu:~/Desktop/net com$
```