Innovative practical (CCN)

Aim: Write a program for distance vector algorithm to find suitable path for transmission.

Theory:

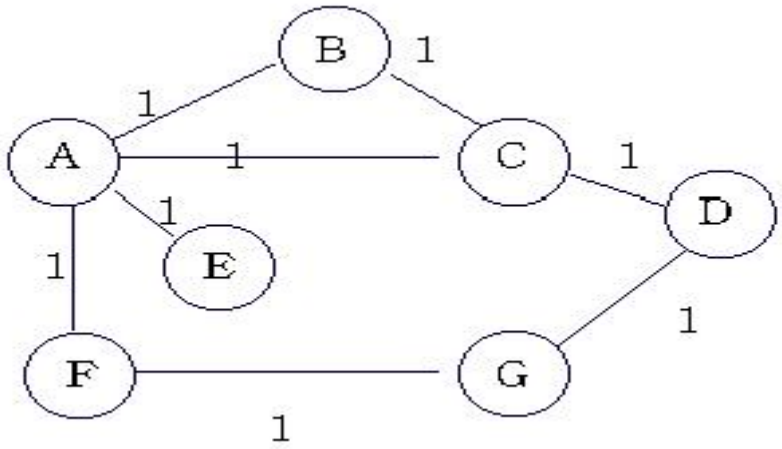
Distance Vector Algorithm:

Each node constructs a one-dimensional array containing the "distances"(costs) to all other nodes and distributes that vector to its immediate neighbors.

1. The starting assumption for distance-vector routing is that each node knows the cost of the link to each of its directly connected neighbors.

2. A link that is down is assigned an infinite cost.

Example.



Information Stored at Node	**Distance to Reach Node**						
	A	**B**	**C**	**D**	**E**	**F**	**G**
A	0	1	1	**?**	1	1	**?**
B	1	0	1	**?**	**?**	**?**	**?**
C	1	1	0	1	**?**	**?**	**?**

D	**?**	**?**	1	0	**?**	**?**	1
E	1	**?**	**?**	**?**	0	**?**	**?**
F	1	**?**	**?**	**?**	**?**	0	1
G	**?**	**?**	**?**	1	**?**	1	0

Table 1. Initial distances stored at each node(global view).

We can represent each node's knowledge about the distances to all other nodes as a table like the one given in Table 1.

Note that each node only knows the information in one row of the table.

1. Every node sends a message to its directly connected neighbors containing its personal list of distance. (for example, **A** sends its information to its neighbors **B,C,E**, and **F**.)
2. If any of the recipients of the information from **A** find that **A** is advertising a path shorter than the one they currently know about, they update their list to give the new path length and note that they should send packets for that destination through **A**. (node **B** learns from **A** that node **E** can be reached at a cost of 1; **B** also knows it can reach **A** at a cost of 1, so it adds these to get the cost of reaching **E** by means of **A**. **B** records that it can reach **E** at a cost of 2 by going through **A**.)
3. After every node has exchanged a few updates with its directly connected neighbors, all nodes will know the least-cost path to all the other nodes.
4. In addition to updating their list of distances when they receive updates, the nodes need to keep track of which node told them about the path that they used to calculate the cost, so that they can create their forwarding table. (for example, **B** knows that it was **A** who said " I can reach **E** in one hop" and so **B** puts an entry in its table that says " To reach **E**, use the link to **A**.)

Information Stored at Node	Distance to Reach Node						
	A	**B**	**C**	**D**	**E**	**F**	**G**
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

Table 2. final distances stored at each node (global view).

In practice, each node's forwarding table consists of a set of triples of the form:

(Destination, Cost, NextHop).

For example, Table 3 shows the complete routing table maintained at node B for the network in figure1.

Destination	Cost	NextHop
A	1	A
C	1	C
D	2	C
E	2	A

F	2	A
G	3	A

Table 3. Routing table maintained at node B.

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
int min(int a,int b)
{
return (a<b)?a:b;
}
void dist(int p[10][10],int n)
{
int i,j,k;
for(k=1;k<=n;k++)
for(j=1;j<=n;j++)
for(i=1;i<=n;i++)
p[i][j]=min(p[i][j],p[i][k]+p[k][j]);
}
```



```
int main()
{
int a[10][10],i,j,n,flag,temp=1;
printf("Enter number of nodes:\n");
scanf("%d",&n);
printf("Enter the matrix 0 for self loop and 999 for no path\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&a[i][j]);
for(i=1;i<=n;i++)
{
printf("Initial record of %d is\n",i);
```

```
for(j=1;j<=n;j++)
printf("%d to %d is %d \n",i,j,a[i][j]);
}
06CSL77
dist(a,n);
for(i=1;i<=n;i++)
{
printf("Updated record of %d is \n",i);
for(j=1;j<=n;j++)
printf("%d to %d is %d\n",i,j,a[i][j]);
}
while(temp)
{
do{
printf("Enter 0 to find shortest path and 1 to exit\n");
scanf("%d",&flag);
if(flag==0)
{
printf("Enter 2 nodes\n");
scanf("%d %d",&i,&j);
printf("%d to %d is %d\n",i,j,a[i][j]);
}
else
temp=0;
break;
}while(flag!=0);
}
exit(0);
}
```

OUTPUT:

```
[root@god ~]# vi dva.c
[root@god ~]# cc dva.c
[root@god ~]# ./a.out
Enter number of nodes:
5
```



```
Enter the matrix 0 for self loop and 999 for no path
0 1 2 3 999
1 0 999 4 0
2 999 0 7 999
3 4 7 0 5
999 0 999 5 0
Initial record of 1 is
```

1 to 1 is 0
1 to 2 is 1
1 to 3 is 2
1 to 4 is 3
1 to 5 is 999
Initial record of 2 is
2 to 1 is 1
2 to 2 is 0
2 to 3 is 999
2 to 4 is 4
2 to 5 is 0
Initial record of 3 is
3 to 1 is 2
3 to 2 is 999
3 to 3 is 0
3 to 4 is 7
3 to 5 is 999
Initial record of 4 is
4 to 1 is 3
4 to 2 is 4
4 to 3 is 7
4 to 4 is 0
4 to 5 is 5
Initial record of 5 is
5 to 1 is 999
5 to 2 is 0
5 to 3 is 999
5 to 4 is 5
5 to 5 is 0
Updated record of 1 is
1 to 1 is 0
1 to 2 is 1
1 to 3 is 2
1 to 4 is 3
1 to 5 is 1
Updated record of 2 is
2 to 1 is 1
2 to 2 is 0
2 to 3 is 3
2 to 4 is 4
2 to 5 is 0
Updated record of 3 is
3 to 1 is 2

3 to 2 is 3
3 to 3 is 0
3 to 4 is 5
3 to 5 is 3
Updated record of 4 is
4 to 1 is 3
4 to 2 is 4
4 to 3 is 5
4 to 4 is 0
4 to 5 is 4
Updated record of 5 is
5 to 1 is 1
5 to 2 is 0
5 to 3 is 3
5 to 4 is 4
5 to 5 is 0
Enter 0 to find shortest path and 1 to exit
0
Enter 2 nodes
4 5
4 to 5 is 4
Enter 0 to find shortest path and 1 to exit
0
Enter 2 nodes
2 5
2 to 5 is 0
Enter 0 to find shortest path and 1 to exit
1

Result: Successfully Implemented distance vector algorithm to find suitable path for transmission.

Viva Questions:

1) What is routing?
2) what are different types of routing?
3) What is Distance vector routing?
4) What is Link state routing?