Experiment No-07

AIM: Write a program to implement link state /Distance vector routing protocol to find suitable path for transmission.

OBJECTIVES:

- To study TCP protocol.
- To implement socket programming.
- To use TCP socket for wired network.

PROBLEM STATMENT:

Write a program to implement link state /Distance vector routing protocol to find suitable path for transmission.

SOFTWARE & HARDWARE REQUIREMENTS:

- Software: gcc compiler & wireshark.
- Hardware: Open source Linux operating system

THEORY:

A distance-vector routing (DVR) protocol requires that a router inform its neighbors of topology changes periodically. Historically known as the old ARPANET routing algorithm (or known as Bellman-Ford algorithm).

Bellman Ford Basics – 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.

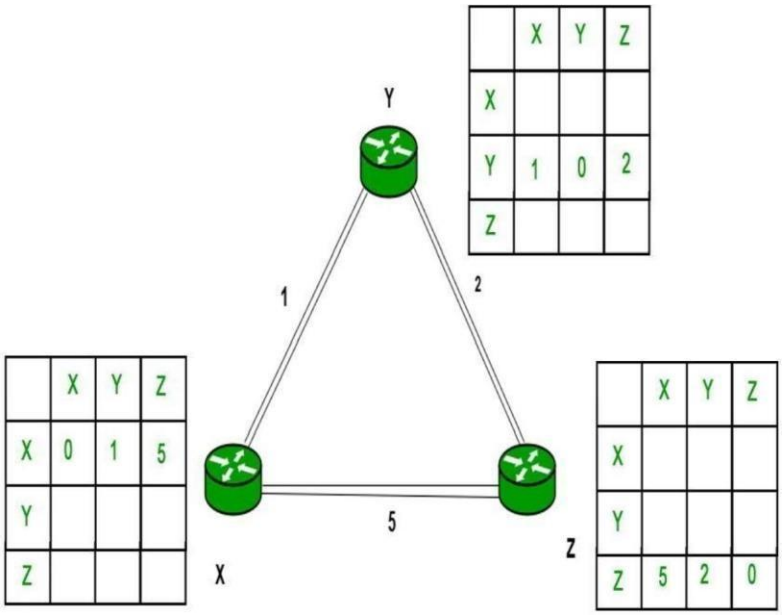
Distance Vector Algorithm –

1. A router transmits its distance vector to each of its neighbors in a routing packet.

2. Each router receives and saves the most recently received distance vector from each of its neighbors.
3. 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.

- From time-to-time, each node sends its own distance vector estimate to neighbors.
- When a node x receives new DV estimate from any neighbor v, it saves v's distance vector and it updates its own DV using B-F equation:
- $D_x(y) = \min \{ C(x,v) + D_v(y), D_x(y) \}$ for each node $y \in N$

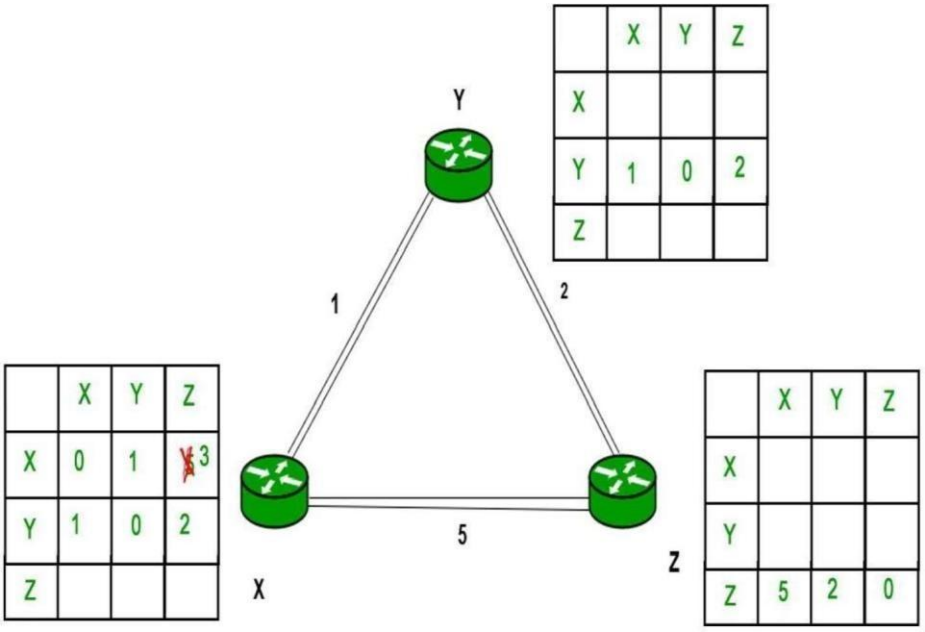
Example – Consider 3-routers X, Y and Z as shown in figure. Each router have their routing table. Every routing table will contain distance to the destination nodes.



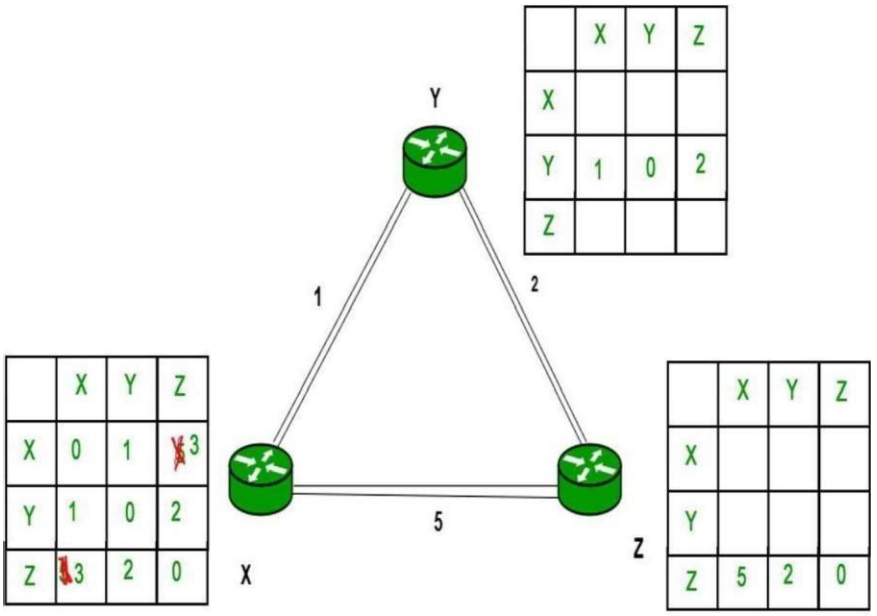
Consider router X , X will share it routing table to neighbors and neighbors will share it routing table to it to X and distance from node X to destination will be calculated using bellmen- ford equation.

$D_x(y) = \min \{ C(x,v) + D_v(y) \}$ for each node $y \in N$

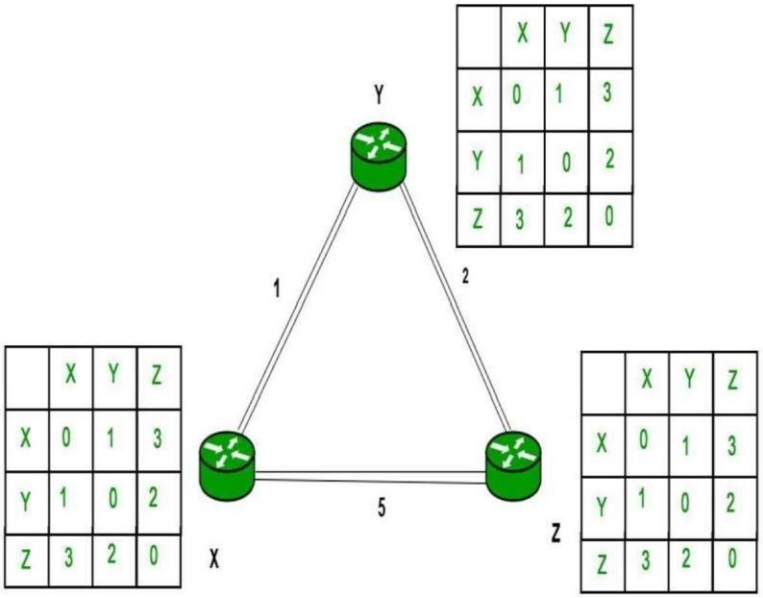
As we can see that distance will be less going from X to Z when Y is intermediate node(hop) so it will be update in routing table X.



Similarly for Z also –



Finally the routing table for all –



Advantages of Distance Vector routing –

It is simpler to configure and maintain than link state routing.

Disadvantages of Distance Vector routing –

It is slower to converge than link state.

- It is at risk from the count-to-infinity problem.
- It creates more traffic than link state since a hop count change must be propagated to all routers and processed on each router. Hop count updates take place on a periodic basis, even if there are no changes in the network topology, so bandwidth-wasting broadcasts still occur.
- For larger networks, distance vector routing results in larger routing tables than link state since each router must know about all other routers. This can also lead to congestion on WAN links.

Write a program to implement link state /Distance vector routing protocol to find suitable path for transmission.

```
#include<stdlib.h> #include<stdio.h>
#define NUL 1000 #define NODES 10 struct node
{
int t[NODES][3];
};
struct node n[NODES]; typedef struct node NOD; int main()
{
void init(int,int); void inp(int,int); void caller(int,int); void op1(int,int,int); void find(int,int);
int i,j,x,y,no; do{
printf("\n Enter the no of nodes required:"); scanf("%d",&no); }while(no>10||no<0); for(i=0;i<no;i++)
{ init(no,i); inp(no,i);
}
printf("\nThe configuration of the nodes after initalization is as follows:"); for(i=0;i<no;i++) op1(no,i,0); for(j=0;j<no;j++)
{
for(i=0;i<no;i++) caller(no,i);
}
printf("\nThe config of the nodes after the comp of the paths is as follows:"); for(i=0;i<no;i++) op1(no,i,1); while(1)
{ printf("\n Enter 0 to exit or any other key to find the shortest path:"); scanf("%d",&j); if(!j) break; Do{ printf("\n Enter the
    nodes btn which path is to be found:"); scanf("%d%d",&x,&y);
}while((x<0||x>no) && (y<0||y>no));
printf("\nThe most suitable route from node %d to %d is as follows\n",x,y); find(x,y); printf("%d",y); printf("\nThe length of
    the shortest path between node %d & %d is %d",x,y,n[x1].t[y-
1][2]); }
}
void init(int no,int x)
{
int i; for(i=0;i999); if(n[x].t[i][2]!=999) n[x].t[i][3]=i; }
} }
void caller(int no,int x) { void compar(int,int,int); int i; for(i=0;iz) { n[x].t[i][2]=z; n[x].t[i][3]=y; }
}
}
void op1(int no,int x,int z)
{ int i,j;
printf("\n The routing table for node no %d is as follows",x+1); printf("\n\n\t\t\tDESTINATION\tDISTANCE\tNEXT_HOP");
    for(i=0;i=999) ||(n[x].t[i][2]>=(999*no))) printf("\n\t\t\t %d \tNO LINK \t NO HOP",n[x].t[i][1]+1); else if(n[x].t[i][3]==NUL)
printf("\n\t\t\t %d \t\t %d \t\t NO HOP",n[x].t[i][1]+1,n[x].t[i][2]); else printf("\n\t\t\t %d \t\t %d
    \t\t%d",n[x].t[i][1]+1,n[x].t[i][2],n[x].t[i][3]+1); } }
```

void find(int x,int y) COMPUTER NETWORKS LABORATORY { int i,j; i=x-1; j=**y-1; printf("%d->",x);**

if(n[i].t[j][3]!=j) { find(n[i].t[j][3]+1,y); return; } }

```
 Enter the no of nodes required:3

Enter the dists from the nodes 1 to other node...
Pls enter 999 if there is no direct

 Enter dist to node 2=10

 Enter dist to node 3=999

Enter the dists from the nodes 2 to other node...
Pls enter 999 if there is no direct

 Enter dist to node 1=999

 Enter dist to node 3=15

Enter the dists from the nodes 3 to other node...
Pls enter 999 if there is no direct

 Enter dist to node 1=20

 Enter dist to node 2=25

The configuration of the nodes after initalization is as follows:
 The routing table for node no 1 is as follows

                        DESTINATION      DISTANCE          NEXT_HOP
                         1                0                1
                         2                10               2
                         3       NO LINK           NO HOP
 The routing table for node no 2 is as follows

                        DESTINATION      DISTANCE          NEXT_HOP
                         1       NO LINK           NO HOP
                         2                0                2
                         3                15               3
```

```
The routing table for node no 3 is as follows

                        DESTINATION      DISTANCE          NEXT_HOP
                         1                20               1
                         2                25               2
                         3                0                3
he config of the nodes after the comp of the paths is as follows:
 The routing table for node no 1 is as follows

                        DESTINATION      DISTANCE          NEXT_HOP
                         1                0                1
                         2                10               2
                         3                25               2
 The routing table for node no 2 is as follows

                        DESTINATION      DISTANCE          NEXT_HOP
                         1                35               3
                         2                0                2
                         3                15               3
 The routing table for node no 3 is as follows

                        DESTINATION      DISTANCE          NEXT_HOP
                         1                20               1
                         2                25               2
                         3                0                3
 Enter 0 to exit or any other key to find the shortest path:1

 Enter the nodes btn which path is to be found:1 3

he most suitable route from node 1 to 3 is as follows
-->2-->3
he length of the shortest path between node 1 & 3 is 25
 Enter 0 to exit or any other key to find the shortest path:0
```

CONCLUSION: -