KRISHNA KANTH 14BIT050

DISTANCE VECTOR ROUTING

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

SERVER:

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<netinet/in.h>

#include<sys/socket.h>

void dij(int n,int v,int cost[20][20],int dist[])

{

int i,u,count,w,flag[10],min;

for(i=1; i<=n; i++)

{

flag[i]=0;

dist[i]=cost[v][i];

}

count=2;

while(count<=n)

{

min=99;

for(w=1; w<=n; w++)

if((dist[w]<min) && !flag[w])

{

min=dist[w];

u=w;

}

flag[u]=1;

count++;

for(w=1; w<=n; w++)

if((dist[u]+cost[u][w] < dist[w]) && !flag[w])

dist[w]=dist[u]+cost[u][w];

}

}

int main()

{

int sd,clen,x;

struct sockaddr\_in ser;

struct sockaddr clt;

ser.sin\_family=AF\_INET;

ser.sin\_port=htons(8989);

ser.sin\_addr.s\_addr=htonl(INADDR\_ANY);

sd=socket(AF\_INET,SOCK\_DGRAM,0);

bind(sd,&ser,sizeof(ser));

clen=sizeof(clt);

recvfrom(sd,&x,sizeof(x),0,&clt,&clen);

int n,i,j,cost[20][20],dist[10];

printf("\nEnter the number of nodes:");

scanf("%d",&n);

printf("\nEnter the cost matrix");

for(i=1; i<=n; i++)

{

for(j=1; j<=n; j++)

{

scanf("%d",&cost[i][j]);

//if(cost[i][j]==0)

//cost[i][j]=999;

}

}

dij(n,x,cost,dist);

sendto(sd,dist,sizeof(dist),0,&clt,clen);

close(sd);

}

CLIENT:

#include<stdio.h>

#include<stdlib.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<string.h>

int main()

{

int i,n,sd,len,x,dist[10];

struct sockaddr\_in ser;

ser.sin\_family=AF\_INET;

ser.sin\_port=htons(8989);

ser.sin\_addr.s\_addr=htonl(INADDR\_ANY);

sd=socket(AF\_INET,SOCK\_DGRAM,0);

printf("\nEnter the Node");

scanf("%d",&x);

sendto(sd,&x,sizeof(x),0,&ser,sizeof(ser));

recvfrom(sd,&dist,sizeof(dist),0,NULL,NULL);

for(i=0; i<7; i++)

printf("\n%d-->%d",i+1,dist[i]);

close(sd);

}

OUTPUT;

SERVER:

Enter the number of nodes:7

Enter the cost matrix

0 1 1 999 1 1 999

1 0 1 999 999 999 999

1 1 0 1 999 999 999

999 999 0 999 999 1

1 999 999 999 0 999 999

9999 999 999 1 999 1 0

1 999 0 0 1 999

0 1 0

CLIENT:

Enter the Node2

1-->1

2-->1

3-->0

4-->1

5-->2

6-->2

7-->2

--------------------------------------------------------------------------------------------------