VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT on

COMPUTER NETWORKS

Submitted by

VIGNESH V (1BM20CS186)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING BENGALURU-560019 October-2022 to Feb-2023

(Autonomous Institution under VTU)

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "LAB COURSE **COMPUTER NETWORKS**" carried out by **VIGNESH V (1BM20CS186),** who is a bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a **Computer Networks - (20CS5PCCON)** work prescribed for the said degree.

Dr. Nandhini Vineeth
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

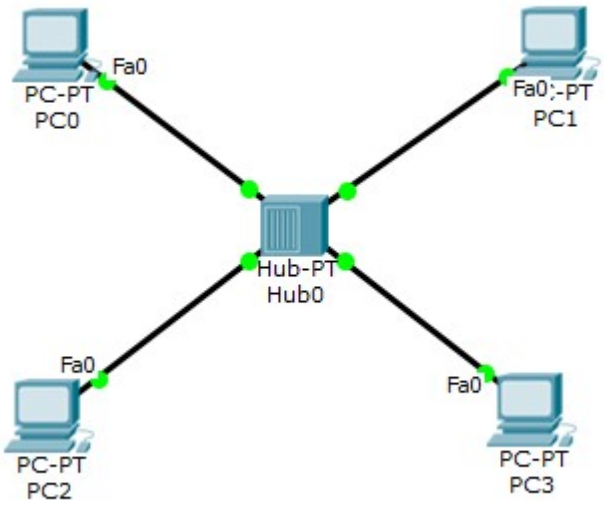
INDEX

CYCLE - 1

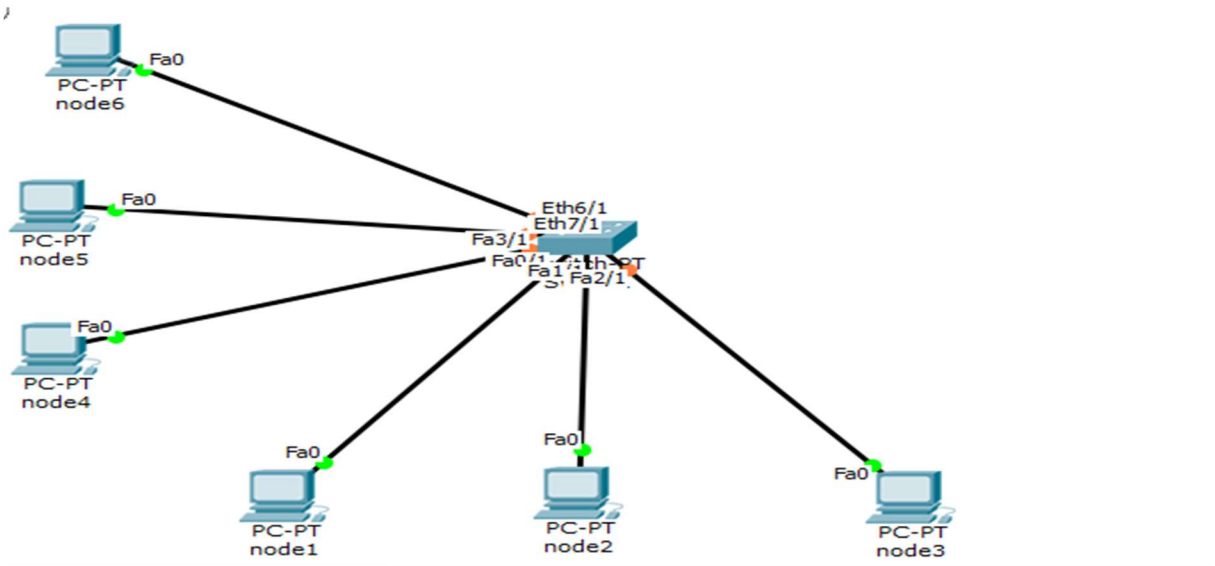
Experiment No-1

Aim : Creating a topology and simulating sending a simple PDU from source to destination using a hub and switch as connecting devices.

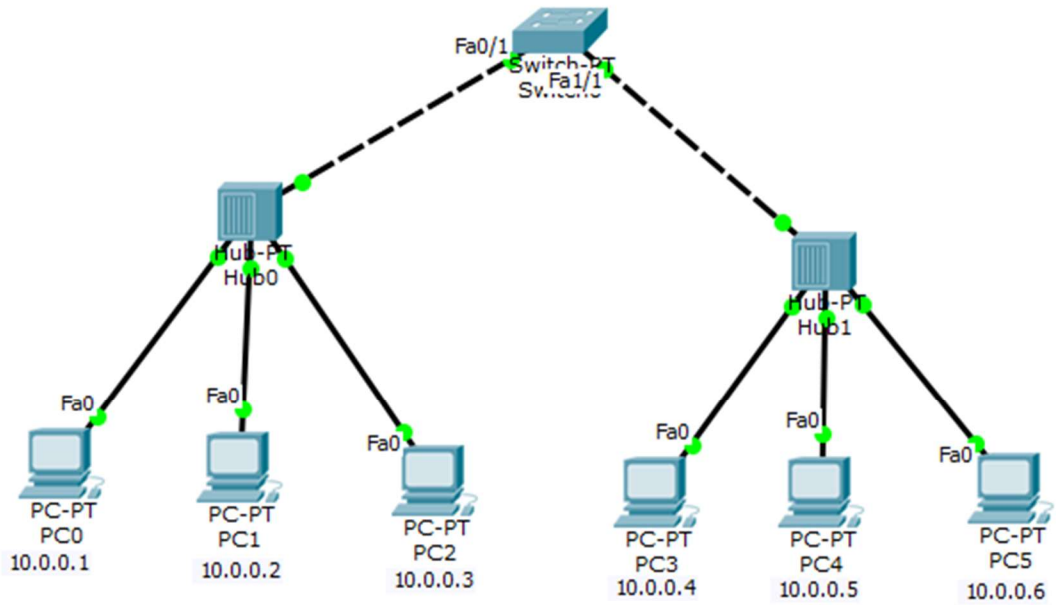
1. PC and Hub



2. Pc and Switch



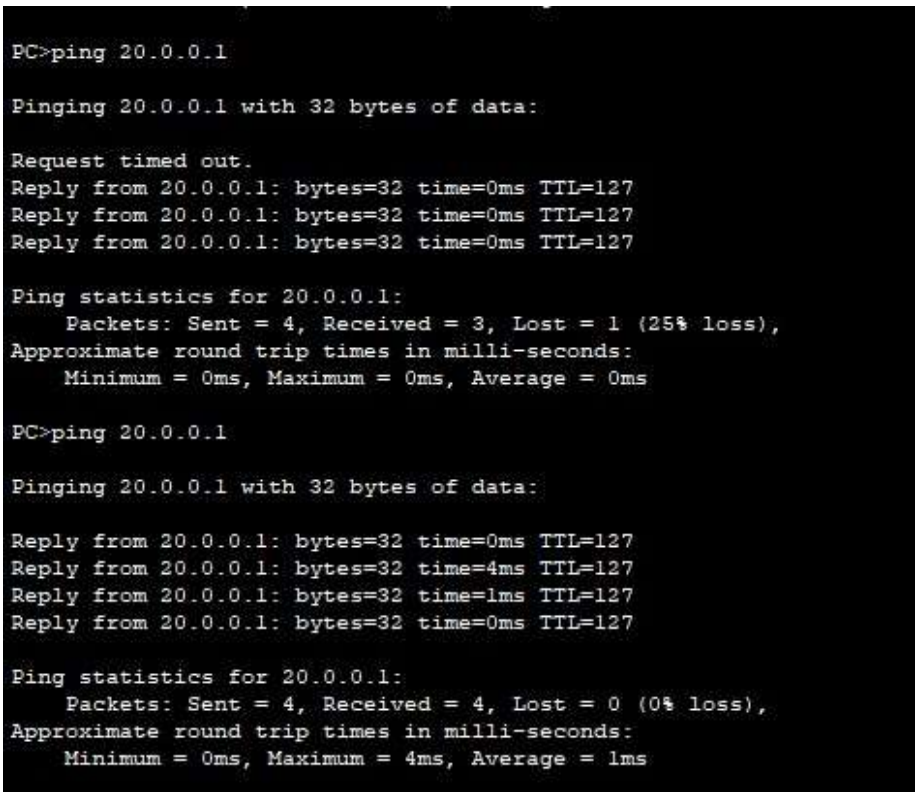
3. PCs with a combination of Switch and Hub



Procedure:

- Put all the devices(PCs, Hubs and Switches) needed for the experiment on the screen by looking at the topology.
- Choose the correct wire and make the Connection as shown in the topology
- Give ip address to all the devices
- Ping from one pc to all other pc in the network to make sure that the connection is correct.

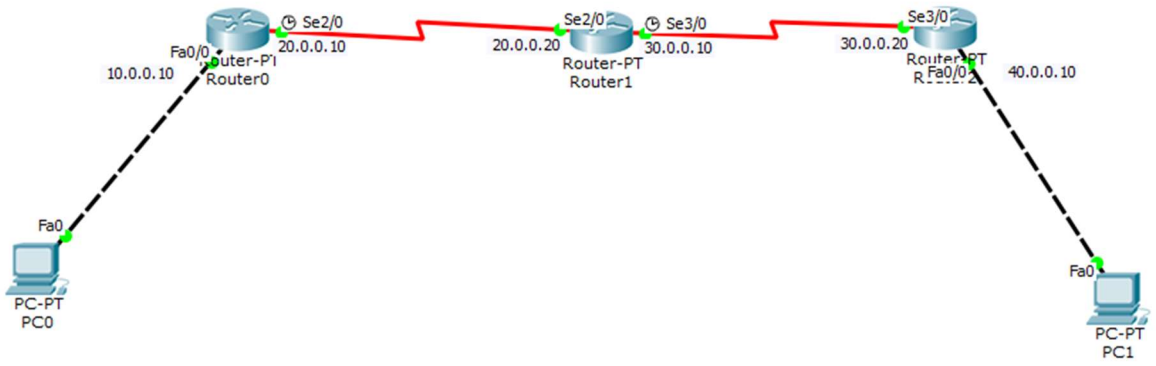
Output:



Experiment No-2

Aim : Configuring IP address to Routers in Packet Tracer. Explore the following messages: Ping Responses, Destination unreachable, Request timed out, Reply

Topology:



Procedure:

1. connect PC-0 with Router-0 using copper cross-over cable - fastethernet0/0
2. connect Router-0 to Router-1 using Serial DCE with the connection named as serial2/0, then connect Router1 to Router2 using serial DCE named serial3/0
3. connect Router2 to PC1 using copper cross-over cable - fastethernet1/0
4. set the IP addresses, subnet mask (255.0.0.0 for all PCs and routers) and gateways accordingly.

   a. PC0: IP address = 10.0.0.1 gateway = 10.0.0.10

   b. Router0: gateway1 = 10.0.0.10 gateway2 = 20.0.0.10

   c. Router1: gateway1 = 20.0.0.20 gateway2 = 30.0.0.10

   d. Router2: gateway1 = 30.0.0.20 gateway2 = 40.0.0.10

   e. PC1: IP address = 40.0.0.1 gateway = 40.0.0.10

5. for Router0, the first gateway is set to IP address of 10.0.0.10 which is as same as the gateway of PC0 then set up the connection between the

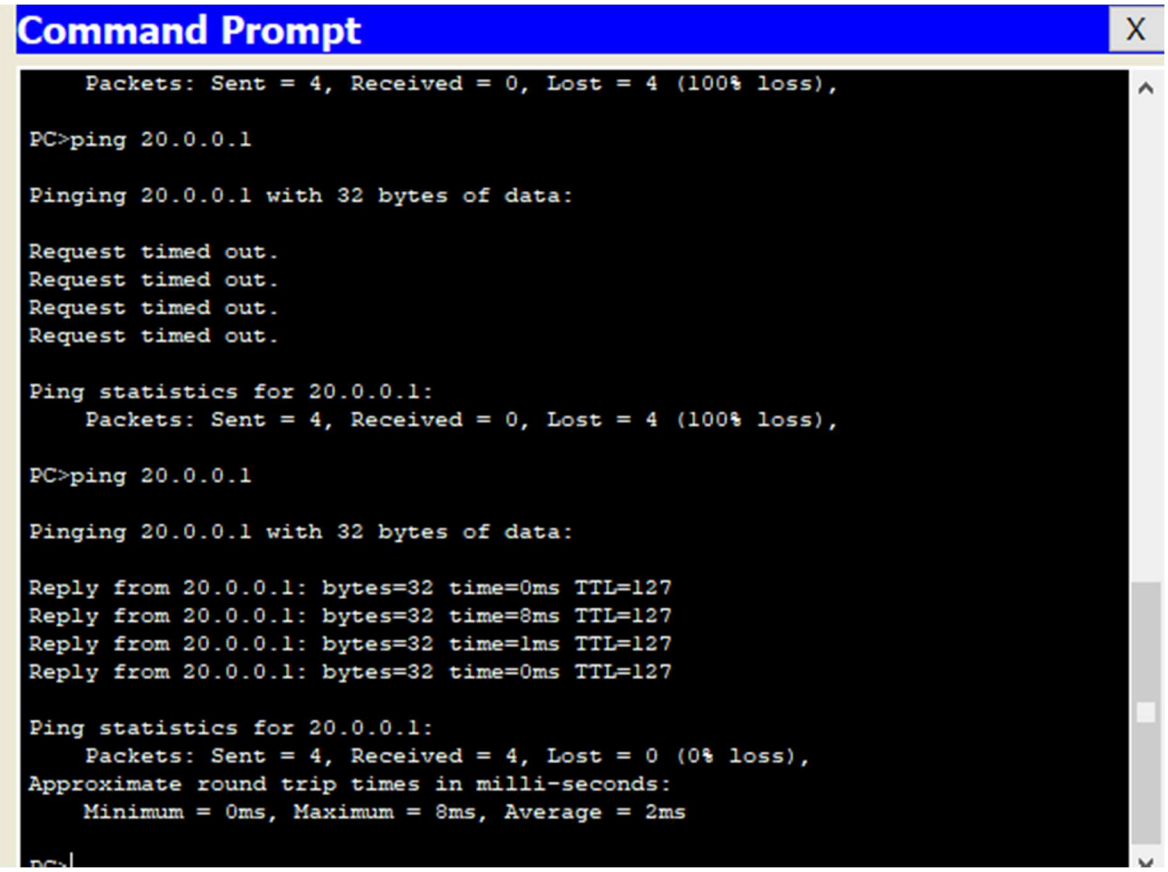
i. Router0 and the PC0 using the CLI.

ii. Router0 and Router1

iii. Router1 and Router2 iv. Router2 and PC1 using CLI

Do (config-if)#ip route {destination-network} {mask} {next-hop-address} for all the routers

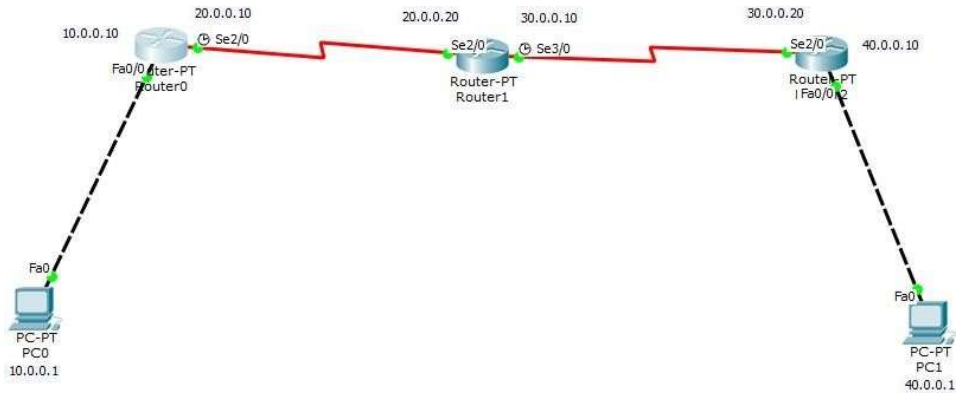
Output:



Experiment No-3

Aim : Configuring default route to the Router.

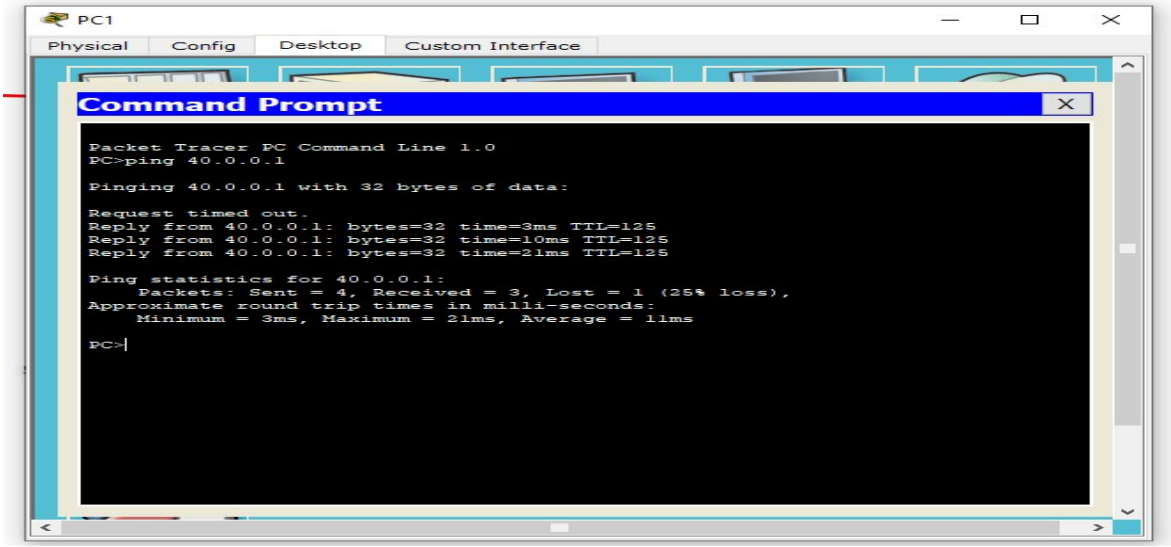
Topology:



Procedure:

- Do the connections as shown in the topology diagram.
- Assign an IP address to all the PCs.
- For router-to-router configuration do:
    - **(config)#ip route 0.0.0.0 0.0.0.0 {Next-hop-Address}**

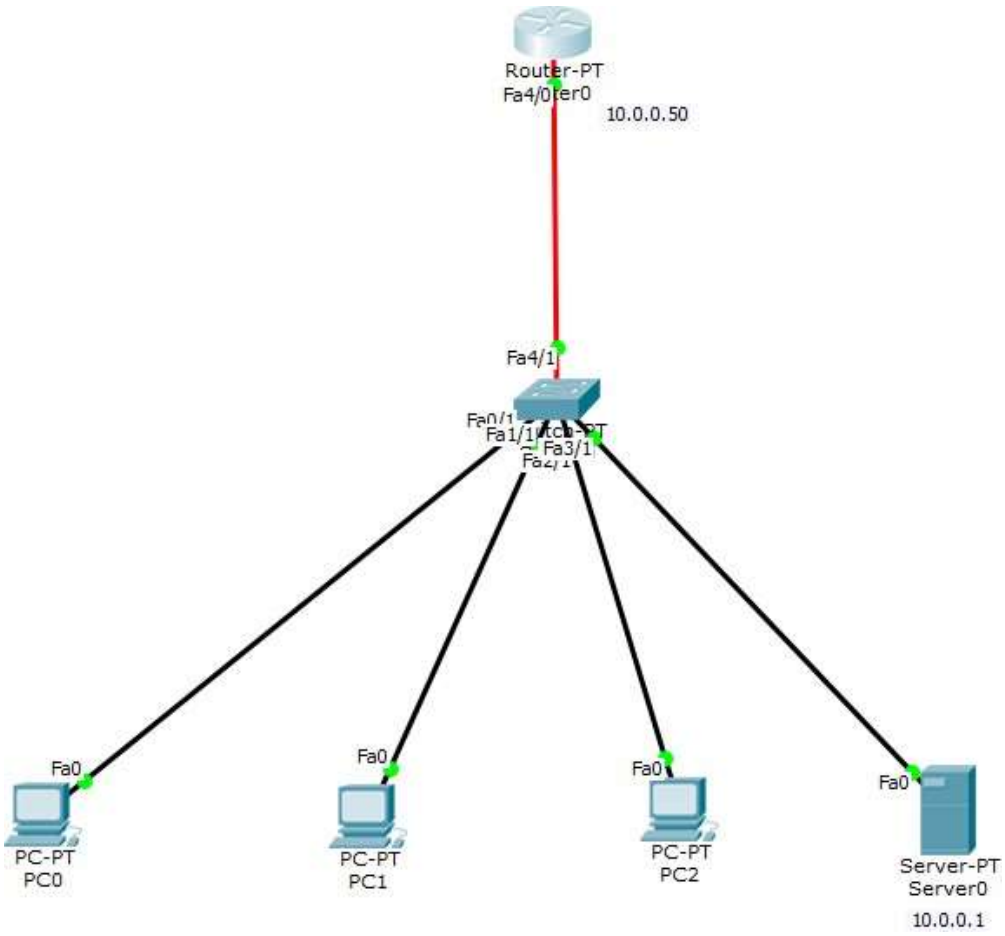
Output:



Experiment No-4

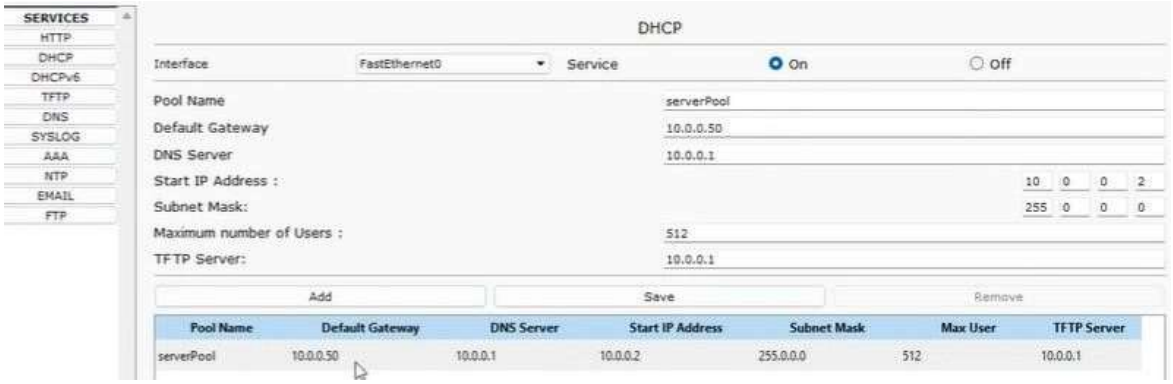
Aim : Configuring DHCP within a LAN in a packet Tracer

Topology:



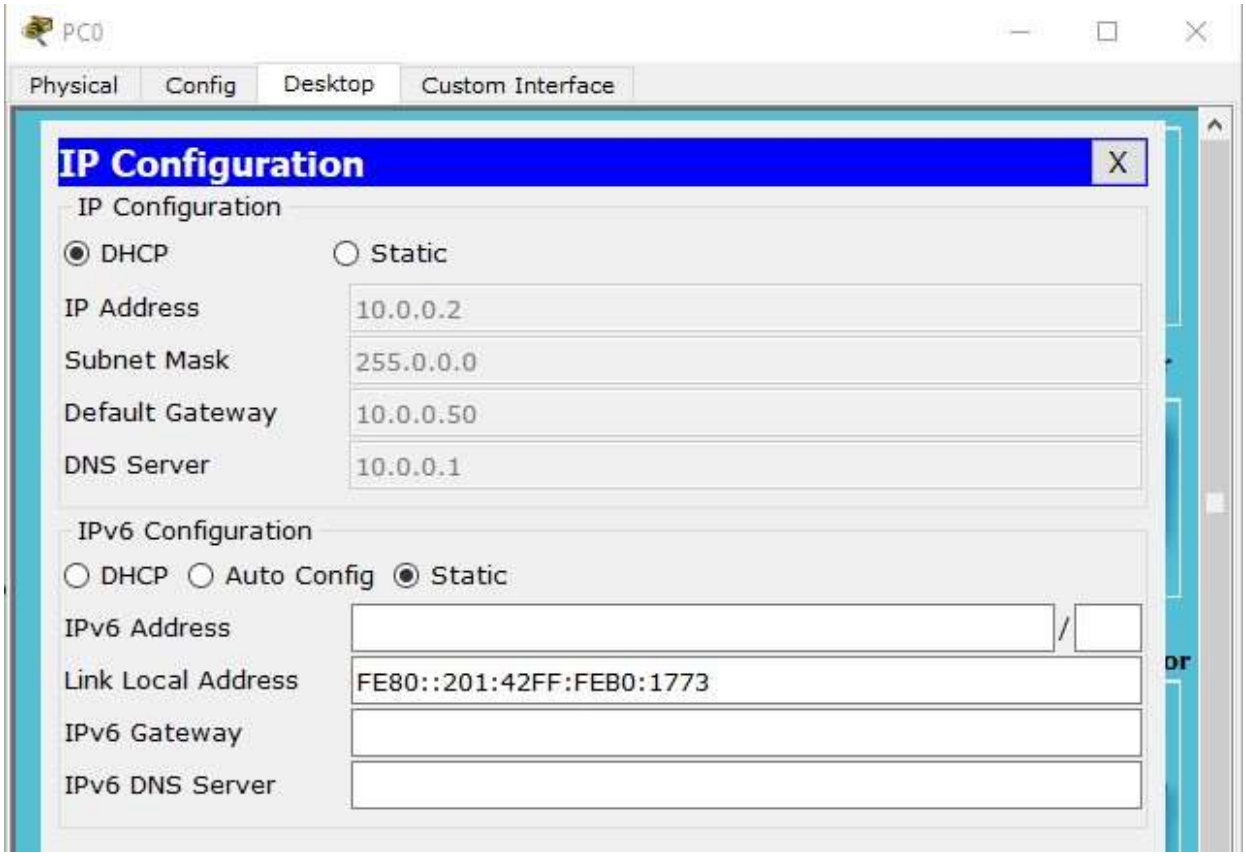
Procedure:

- Do the connection as shown in the topology diagram.
- For DHCP settings go to server and do the following

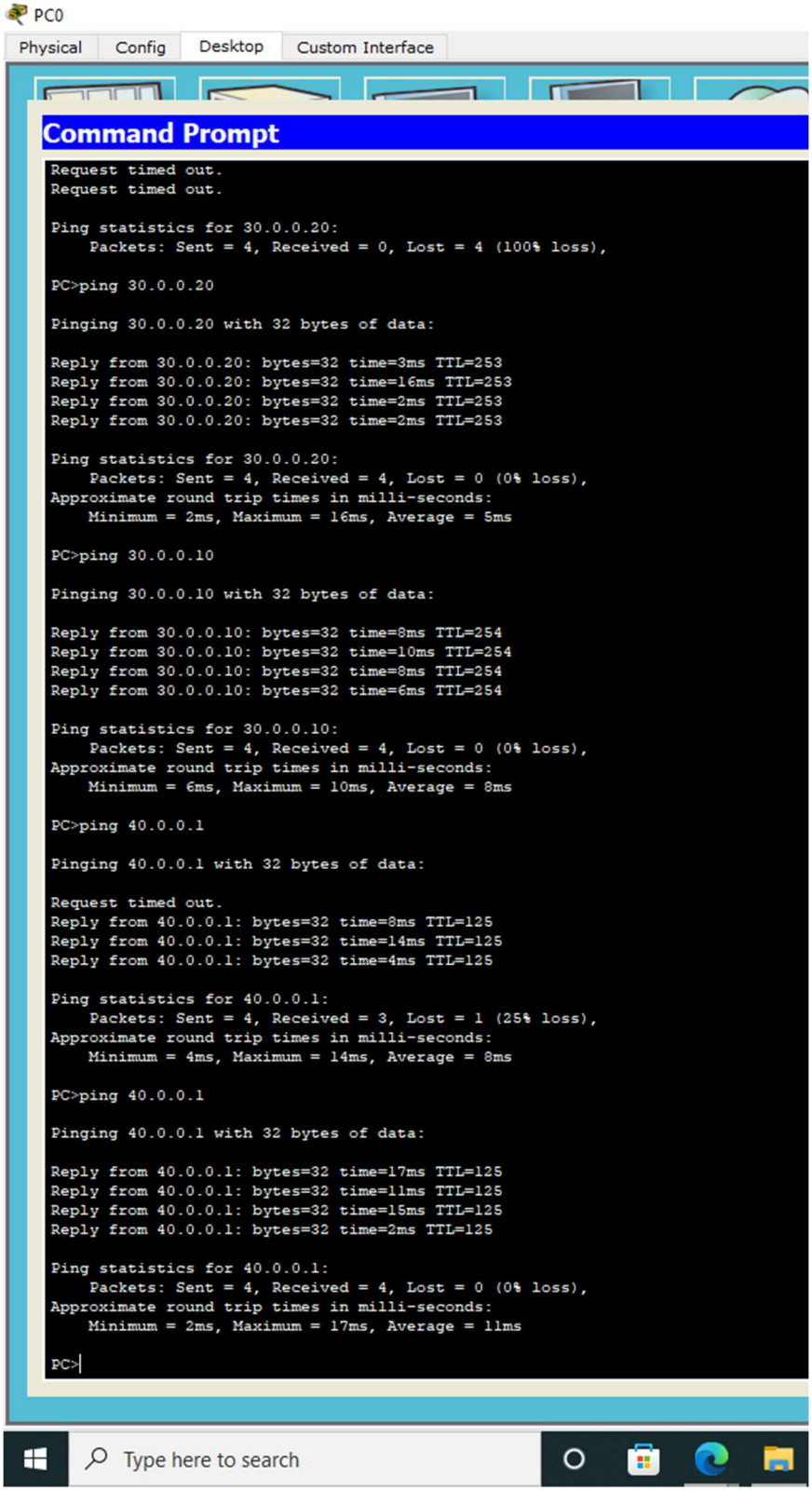


- For the PCs Go to ip configuration>Select DHCP.

=



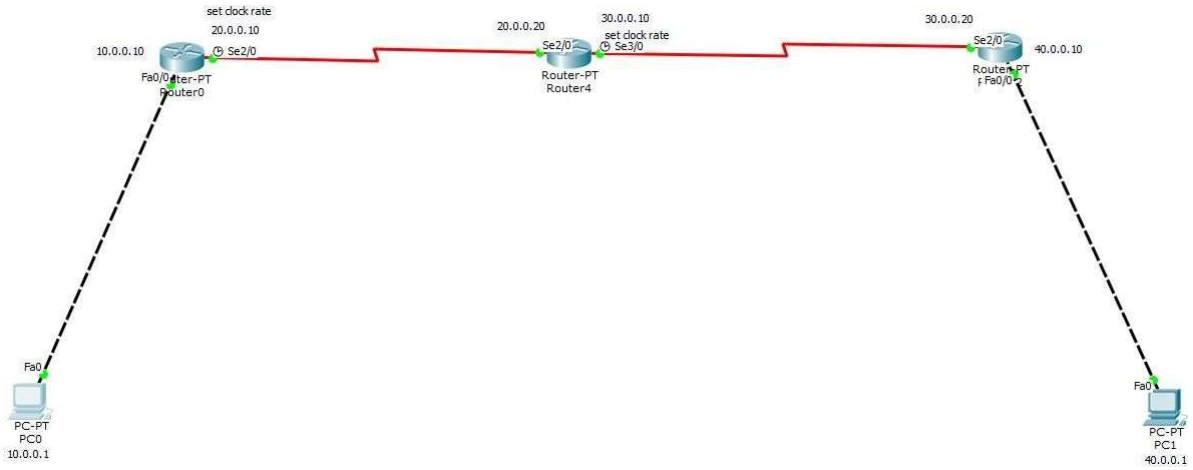
Output:


PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
Request timed out.
Request timed out.
Ping statistics for 30.0.0.20:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>ping 30.0.0.20
Pinging 30.0.0.20 with 32 bytes of data:
Reply from 30.0.0.20: bytes=32 time=3ms TTL=253
Reply from 30.0.0.20: bytes=32 time=16ms TTL=253
Reply from 30.0.0.20: bytes=32 time=2ms TTL=253
Reply from 30.0.0.20: bytes=32 time=2ms TTL=253
Ping statistics for 30.0.0.20:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 16ms, Average = 5ms
PC>ping 30.0.0.10
Pinging 30.0.0.10 with 32 bytes of data:
Reply from 30.0.0.10: bytes=32 time=8ms TTL=254
Reply from 30.0.0.10: bytes=32 time=10ms TTL=254
Reply from 30.0.0.10: bytes=32 time=8ms TTL=254
Reply from 30.0.0.10: bytes=32 time=6ms TTL=254
Ping statistics for 30.0.0.10:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 6ms, Maximum = 10ms, Average = 8ms
PC>ping 40.0.0.1
Pinging 40.0.0.1 with 32 bytes of data:
Request timed out.
Reply from 40.0.0.1: bytes=32 time=8ms TTL=125
Reply from 40.0.0.1: bytes=32 time=14ms TTL=125
Reply from 40.0.0.1: bytes=32 time=4ms TTL=125
Ping statistics for 40.0.0.1:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 4ms, Maximum = 14ms, Average = 8ms
PC>ping 40.0.0.1
Pinging 40.0.0.1 with 32 bytes of data:
Reply from 40.0.0.1: bytes=32 time=17ms TTL=125
Reply from 40.0.0.1: bytes=32 time=11ms TTL=125
Reply from 40.0.0.1: bytes=32 time=15ms TTL=125
Reply from 40.0.0.1: bytes=32 time=2ms TTL=125
Ping statistics for 40.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 17ms, Average = 11ms
PC>
Type here to search

Experiment No-5

Aim : Configuring RIP Routing Protocol in Routers.

Topology:



Procedure:

Router enable Router#config t

Router (config)#interface fastethernet0/0

Router (config-if)# ip address 10.0.0.10

255.0.0.0 Router (config-if)#no shut

Router (config-if)#exit

Router (config)#interface serial2/0

Router (config-if)#ip address 20.0.0.10 255.0.0.0 Router

(config-if)#encapsulation ppp

Router (config-if)#clock rate 6400 Unknown clock rate

Router (config-if)#clock rate 64000

Router (config-if)#no shut

Router (config) #interface serial2/0 Router

(config-if)#ip address 20.0.0.20 255.0.0.0

Router (config-if)#encapsulation ppp

Router (config-if)#no shut

Router (config) #interface serial 3/0

Router (config-if)# ip address 30.0.0.10 255.0.0.0 Router

(config-if)#encapsulation ppp

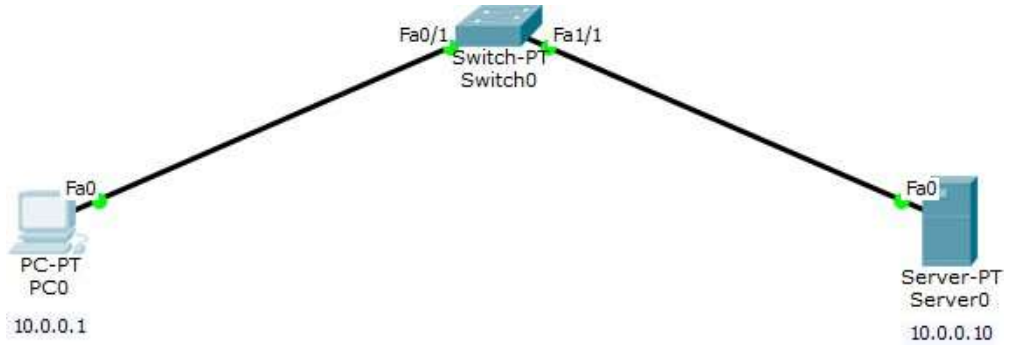
Router (config-if)#clock rate 64000 Router (config-if)#no shut

Output:

Experiment No-6

Aim: Demonstration of WEB server and DNS using Packet Tracer.

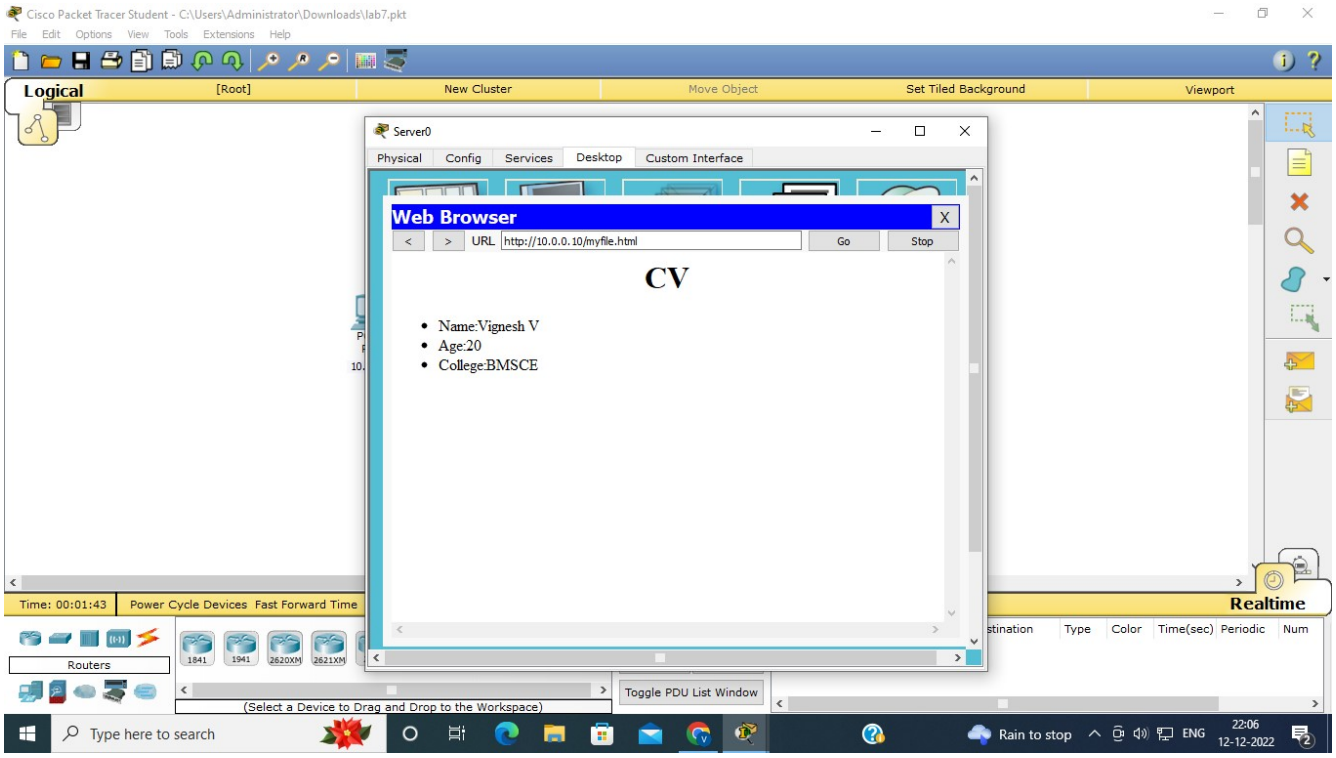
Topology:



Procedure:

- set up IP address for PC0 and server
- select PC, choose Desktop tab, choose Web Browser and enter 10.0.0.10 IP address, which displays the home page
- select server, choose Services tab, select HTTP and switch it on. CLick the edit button for ìndex.html and edit the file.
- switch the DNS on, and add a domain name - bmsce with the address 10.0.0.10
- search for the domain name in the web browser of the PC.

Output:


Cisco Packet Tracer Student - C:\Users\Administrator\Downloads\lab7.pkt
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Server0
Physical
Config
Services
Desktop
Custom Interface
Web Browser
URL
http://10.0.0.10/myfile.html
Go
Stop
CV
Name:Vignesh V
Age:20
College:BMSCE
Time: 00:01:43
Power Cycle Devices
Fast Forward Time
Realtime
Routers
Toggle PDU List Window
(Select a Device to Drag and Drop to the Workspace)
Type here to search
Rain to stop
ENG
22:06
12-12-2022

CYCLE - 2

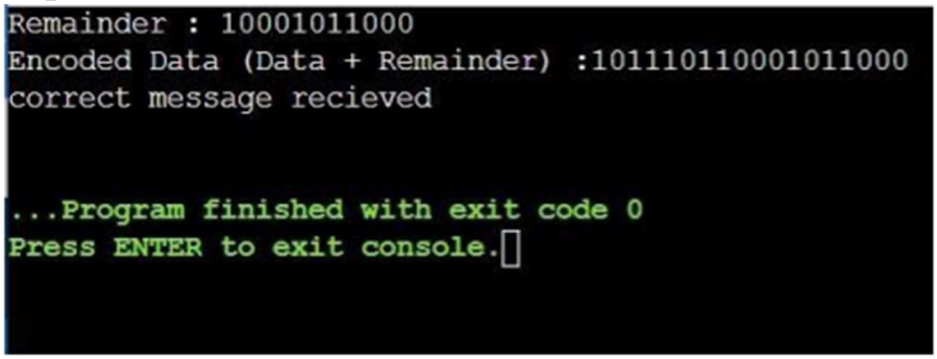
Program 1: Write a program for error-detecting code using CRC-CCITT (16-bits).

Code :

```
#include <iostream>
#include <string.h> using
namespace std;
int crc(char *ip, char *op, char *poly, int mode)
{ strcpy(op, ip); if (mode) { for (int i =
   1; i < strlen(poly); i++) strcat(op,
   "0");
   }
   /* Perform XOR on the msg with the selected polynomial */
   for (int i = 0; i < strlen(ip); i++) { if (op[i] == '1') { for (int j
   = 0; j < strlen(poly); j++) { if (op[i + j] == poly[j]) op[i + j]
   = '0';
            else op[i + j] =
               '1';
         }
      }
   }
   /* check for errors. return 0 if error detected
   */ for (int i = 0; i < strlen(op); i++) if (op[i]
   == '1') return 0;

   return 1;

}


int main(){ char ip[50], op[50],

   recv[50];

   char poly[] = "10001000000100001";

cout << "Enter the input message in binary"<< endl;
   cin >> ip; crc(ip,
   op, poly, 1);
   cout << "The transmitted message is: " << ip << op + strlen(ip) <<
   endl; cout << "Enter the recevied message in binary" << endl; cin >>
   recv; if (crc(recv, op, poly, 0))
```

```
    cout << "No error in data" << endl;
  else
    cout << "Error in data transmission has occurred" << endl;

  return 0;
}
```

Output :



Program 2 : Write a program for distance vector algorithm to find suitable path for transmission **Code :**

```
class Topology:
  def init (self, array_of_points):
    self.nodes = array_of_points
    self.edges = []

  def add_direct_connection(self, p1, p2, cost):
    self.edges.append((p1, p2, cost))
    self.edges.append((p2, p1, cost))

  def distance_vector_routing(self):
    import collections for node in
    self.nodes: dist =
    collections.defaultdict(int)
    next_hop = {node: node} for
    other_node in self.nodes:
        if other_node != node:
          dist[other_node] = 100000000 # infinity
      # Bellman Ford Algorithm for i
      in range(len(self.nodes)-1):
        for edge in self.edges: src, dest,
          cost = edge if dist[src] + cost
```

```
                < dist[dest]: dist[dest] =
                dist[src] + cost if src == node:
                    next_hop[dest] =dest
                elif src in next_hop:
                    next_hop[dest] = next_hop[src]

        self.print_routing_table(node, dist, next_hop)
        print()

    def print_routing_table(self, node, dist, next_hop):
        print(f'Routing table for {node}:') print('Dest
        \t Cost \t Next Hop') for dest, cost in
        dist.items(): print(f'{dest} \t {cost} \t
        {next_hop[dest]}') arr=[] l=int(input("Enter
        the number of nodes")) for _ in range (0,l):
    arr.append(input("Enter the name of the node"))
t=Topology(arr) edges=int(input('Enter no.
of connections')) for _ in range(edges):
    src,dest,cost=input('Enter [src][dest][cost]').split()
    t.add_direct_connection(src,dest,int(cost))
t.distance_vector_routing()
```

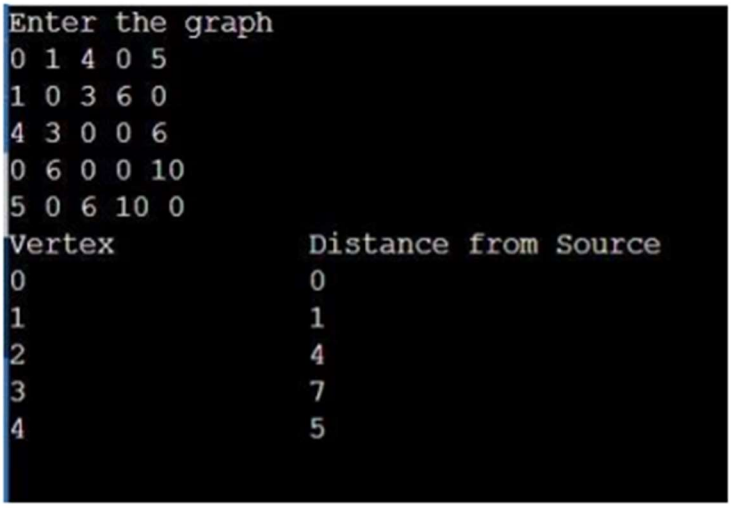
Output :

Program 3 : Implement Djikstra's algorithm to compute the shortest path for a given topology **Code :**

```
#include<stdio.h>
void dijkstras(); int
c[10][10],n,src;
void main()
{
int i,j;
printf("\nenter the no of vertices:\t");
scanf("%d",&n); printf("\nenter the
cost matrix:\n");
for(i=1;i<=n;i++)
{ for(j=1;j<=n;j++)
   scanf("%d",&c[i][j]);
}
printf("\nenter the source node:\t");
 scanf("%d",&src);
dijkstras(); getch();
}
void dijkstras()
```

```
{
int vis[10],dist[10],u,j,count,min;
for(j=1;j<=n;j++)
  {
   dist[j]=c[src][j];
  }
  for(j=1;j<=n;j++)
  { vis[j]=0;
  } dist[src]=0;
  vis[src]=1;
  count=1;
  while(count!=n
  )
  {
   min=9999;
   for(j=1;j<=n;j++) {
   if(dist[j]<min&&vis[j]!=1)
    { min=dist[j];
     u=j;
    } } vis[u]=1; count++;
   for(j=1;j<=n;j++) {
   if(min+c[u][j]<dist[j]&&vis[j]!=1)
    {
     dist[j]=min+c[u][j];
    }
   }
  } printf("\nthe shortest distance
  is:\n");
  for(j=1;j<=n;j++)
  { printf("\n%d ----
   >%d=%d",src,j,dist[j]);
  }
}
```

Output :



Program 4 : Write a program for congestion control using Leaky bucket algorithm.

Code :

```
#include <iostream>
#include <vector> #include

<bits/stdc++.h> using

namespace std;

int main()
{ int sum=0,pkt,leak = 10;
   int choice; vector <int>
   bucket; int cap = 50;
   while(true){
   cout<<"1. Add packet\n2. No packets\n3. Exit\nEnter choice :
   "; cin>>choice; switch(choice){ case 1 : cout<<"Enter pkt : ";
   cin>>pkt; if(pkt>cap-sum)
         cout<<"Bucket OverFlow"<<endl;
      else{
```

```
        bucket.push_back(pkt);
        sum = accumulate(bucket.begin(), bucket.end(), 0);
        cout<<"\nBefore leak"<<endl;
        cout<<"sum = "<<sum<<" leak = "<<leak<<endl;
        }
        bucket.push_back(-leak);
        sum = accumulate(bucket.begin(), bucket.end(), 0);
        if(sum<0) sum=0;
        cout<<"\nAfter leak"<<endl;
        cout<<"sum = "<<sum<<" leak = "<<leak<<endl;
        break;

        case 2:
        if(sum>leak){
        bucket.push_back(-leak); sum =
        accumulate(bucket.begin(), bucket.end(), 0);
        cout<<"sum = "<<sum<<" leak = "<<leak<<endl;
        } else
        if(sum<leak){
           sum = 0; cout<<"sum = "<<sum<<" leak =
           "<<leak<<endl;
        }
        else{
           bucket.push_back(-leak);
           sum = accumulate(bucket.begin(), bucket.end(), 0);
           cout<<"sum = "<<sum<<endl; cout<<"\nBucket
           Empty"<<endl;
        }break;

        case 3: cout<<"\nexit"; exit(0);
        break; default : cout<<"wrong
        choice\n";
    }
    }

    return 0;
}
```

Output :

```
Enter Bucket Size: 1000
Enter Output Rate: 200
Enter Input Packets: 300 200 100 400 450 550 400
Packet No: 0 Packet Size: 300
     200 bytes sent
     Last 100 bytes sent
     Bucket output successful

Packet No: 1 Packet Size: 200
     Last 200 bytes sent
     Bucket output successful

Packet No: 2 Packet Size: 100
     Last 100 bytes sent
     Bucket output successful

Packet No: 3 Packet Size: 400
     200 bytes sent
     Last 200 bytes sent
     Bucket output successful

Packet No: 4 Packet Size: 450
     200 bytes sent
     200 bytes sent
     Last 50 bytes sent
     Bucket output successful

Packet No: 5 Packet Size: 550
     200 bytes sent
     200 bytes sent
     Last 150 bytes sent
     Bucket output successful

Packet No: 6 Packet Size: 400
     200 bytes sent
     Last 200 bytes sent
     Bucket output successful

>
```

Program 5 : Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present **Code :**

Server :

```
from socket import * serverName="127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1) while 1: print ("The server is
ready to receive") connectionSocket, addr =
serverSocket.accept() sentence =
connectionSocket.recv(1024).decode()

file=open(sentence,"r") l=file.read(1024)

connectionSocket.send(l.encode())
print ('\nSent contents of ' + sentence)
file.close() connectionSocket.close()
```

Client :

```
from socket import * serverName
= '127.0.0.1' serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = input("\nEnter file name: ")
clientSocket.send(sentence.encode()) filecontents =
clientSocket.recv(1024).decode() print ('\nFrom
Server:\n') print(filecontents) clientSocket.close()
```

Output :

```
λ python server.py
------------------ Server ------------------
Connected by: ('127.0.0.1', 57673)
Received Filename: man.py
Sent: b'File man.py not found'

Received Filename: test.txt
Sent: b'this is a test.\nWelcome to Sockets!'
```

```
λ python client.py
------------------ Client ------------------
Enter file name: man.py
Sent: man.py
Received: File man.py not found

Enter file name: test.txt
Sent: test.txt
Received: this is a test.
Welcome to Sockets!

Enter file name:
```

Program 6 : Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present **Code :**

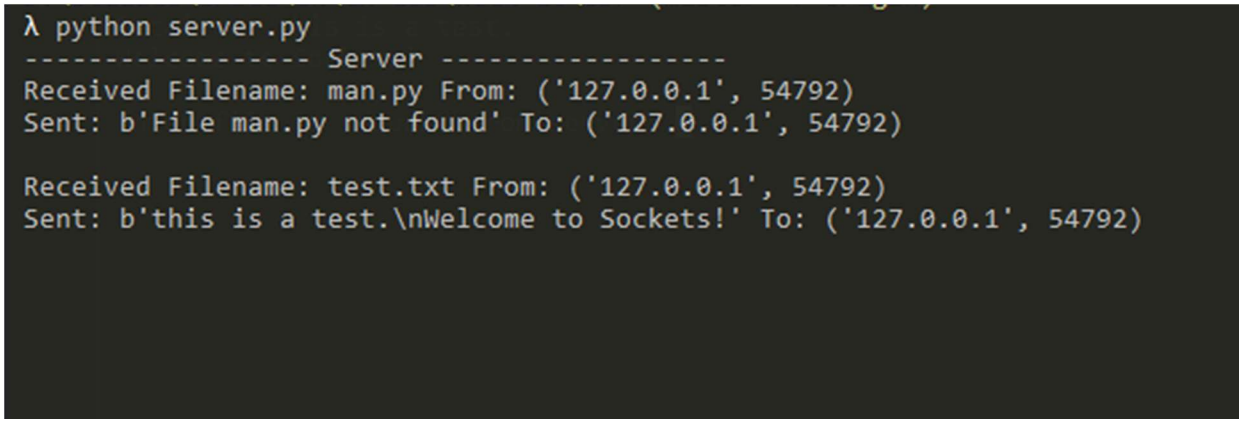
Server :

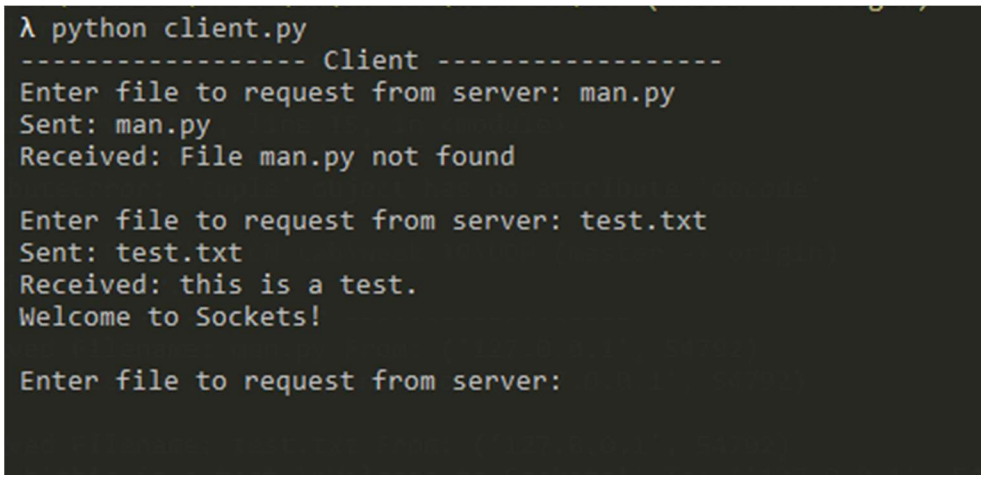
```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,
SOCK_DGRAM)
serverSocket.bind((gethostname(), serverPort))
print ("The server is ready to receive") while 1:
    sentence,clientAddress = serverSocket.recvfrom(2048)
file=open(sentence,"r")
    l=file.read(2048)

    serverSocket.sendto(bytes(l,"utf-8"),clientAddress)
    print("sent back to client",l) file.close()
```

Client :

```
from socket import * serverName
= gethostname() serverPort =
12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("Enter file name")
clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))
filecontents,serverAddress = clientSocket.recvfrom(2048) print
('From Server:', filecontents) clientSocket.close()
```

Output :




λ python client.py
------------------ Client ------------------
Enter file to request from server: man.py
Sent: man.py
Received: File man.py not found
Enter file to request from server: test.txt
Sent: test.txt
Received: this is a test.
Welcome to Sockets!
Enter file to request from server: