VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

COMPUTER NETWORKS

Submitted by

VARUN URS M S (1BM20CS182)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

October-2022 to Feb-2023

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 **VARUN URS M S (1BM20CS182),** 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. NANDINI VINEETH	**Dr. Jyothi S Nayak**
Assistant Professor	Professor and Head
Department of CSE	Department of CSE
BMSCE, Bengaluru	BMSCE, Bengaluru

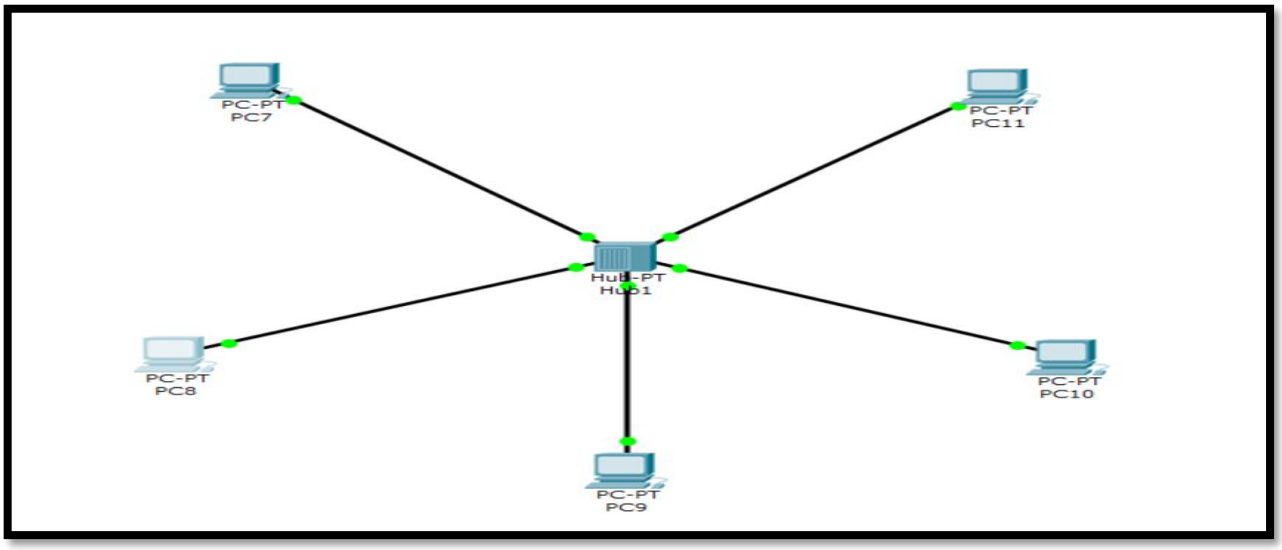
INDEX

CYCLE - 1

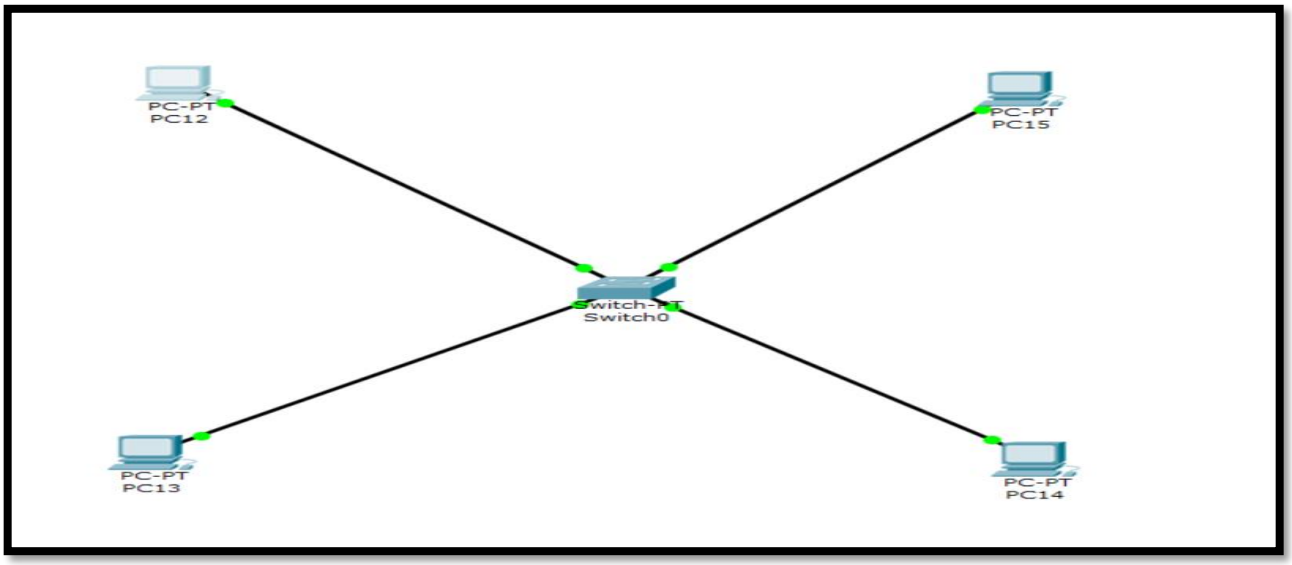
Experiment No-1

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

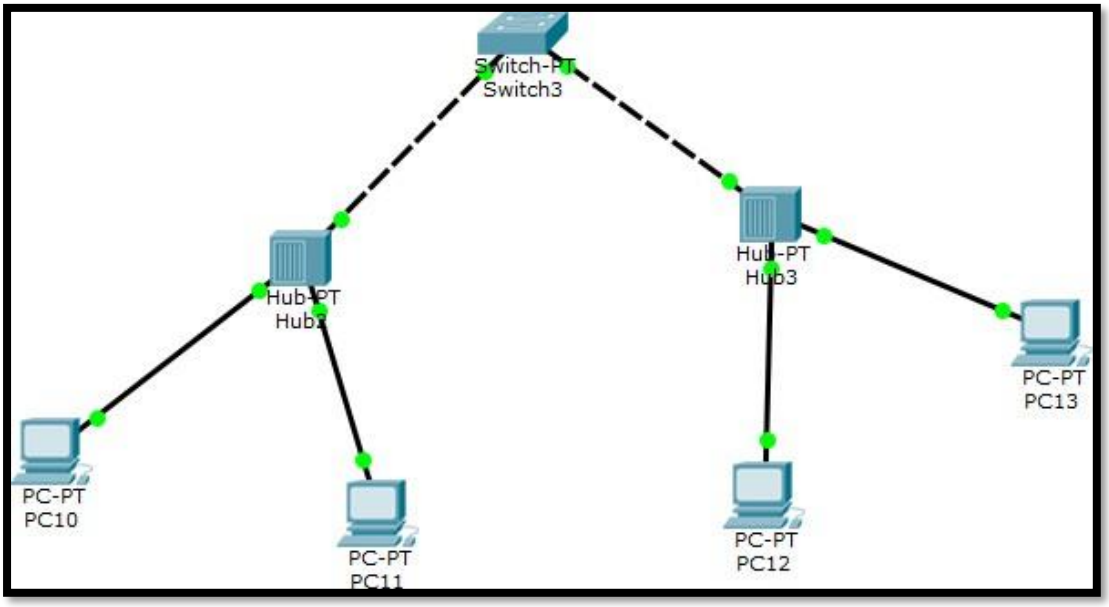
1. PC and Hub



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:

```
PC>ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data:

Request timed out.
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127

Ping statistics for 20.0.0.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

PC>ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data:

Reply from 20.0.0.1: bytes=32 time=0ms TTL=127
Reply from 20.0.0.1: bytes=32 time=4ms TTL=127
Reply from 20.0.0.1: bytes=32 time=1ms TTL=127
Reply from 20.0.0.1: bytes=32 time=0ms TTL=127

Ping statistics for 20.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 4ms, Average = 1ms
```

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:

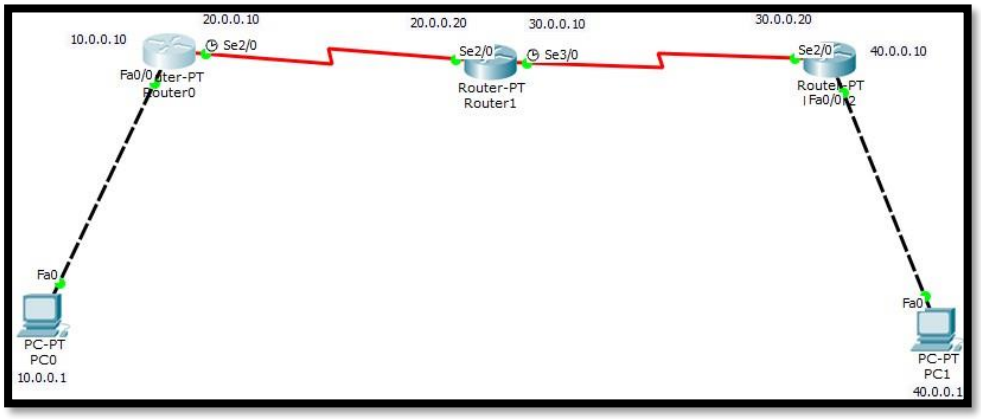
Command Prompt

```
Packet Tracer PC Command Line 1.0
PC>ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: Destination host unreachable.

Ping statistics for 40.0.0.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

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=3ms TTL=125
Reply from 40.0.0.1: bytes=32 time=2ms TTL=125
Reply from 40.0.0.1: bytes=32 time=14ms TTL=125
Reply from 40.0.0.1: bytes=32 time=8ms 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 = 14ms, Average = 6ms

PC>ping 40.0.0.10

Pinging 40.0.0.10 with 32 bytes of data:

Reply from 40.0.0.10: bytes=32 time=4ms TTL=253
Reply from 40.0.0.10: bytes=32 time=3ms TTL=253
Reply from 40.0.0.10: bytes=32 time=4ms TTL=253
Reply from 40.0.0.10: bytes=32 time=8ms TTL=253

Ping statistics for 40.0.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 3ms, Maximum = 8ms, Average = 4ms
```

Experiment No-3

Aim : Configuring default route to the Router.

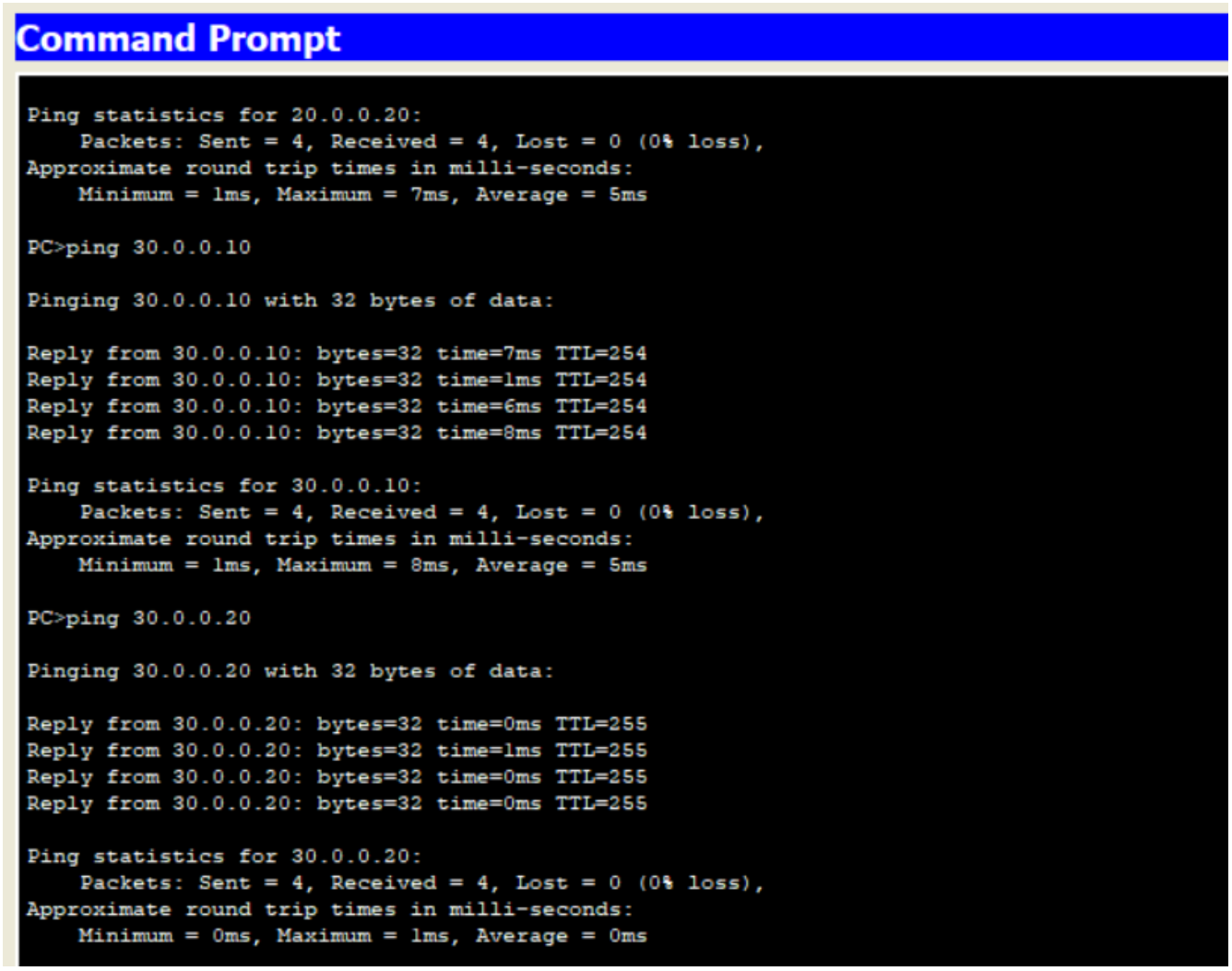
Topology:



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}**

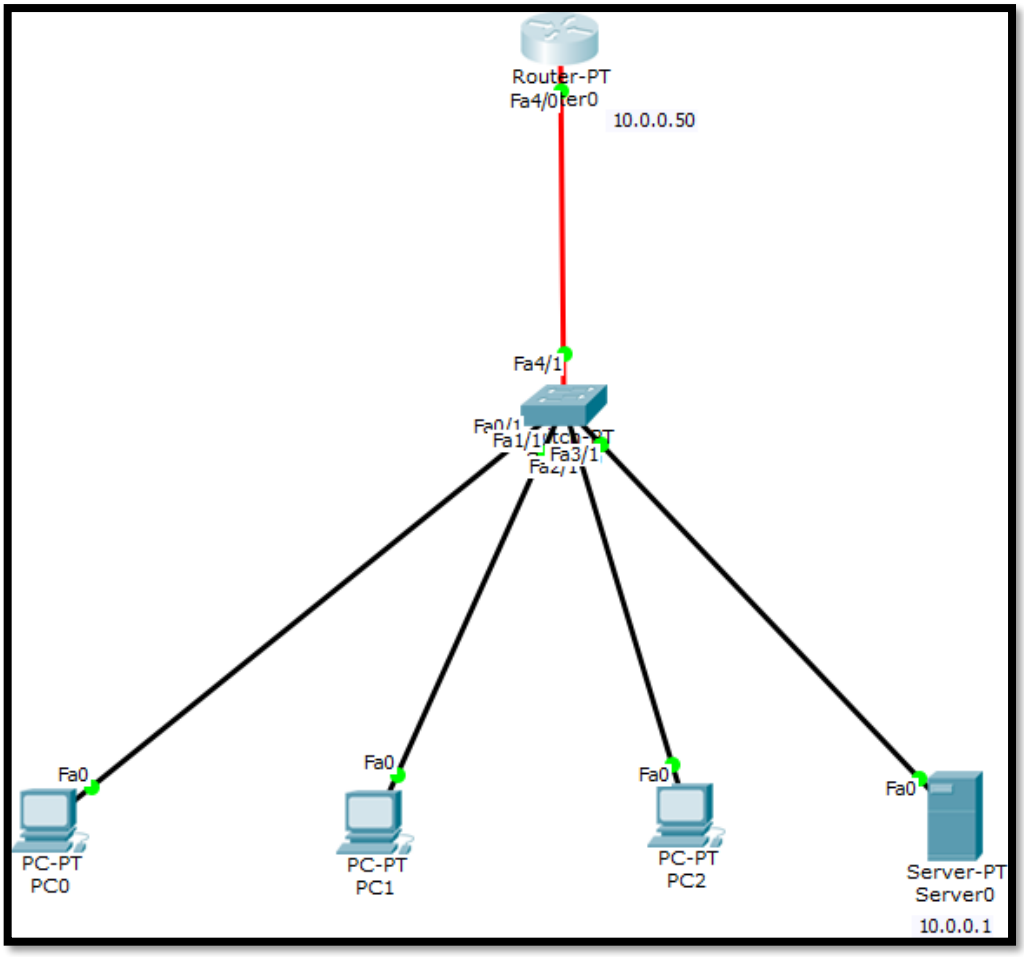
Output:


Command Prompt
Ping statistics for 20.0.0.20:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 1ms, Maximum = 7ms, 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=7ms TTL=254
Reply from 30.0.0.10: bytes=32 time=1ms TTL=254
Reply from 30.0.0.10: bytes=32 time=6ms TTL=254
Reply from 30.0.0.10: bytes=32 time=8ms 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 = 1ms, Maximum = 8ms, Average = 5ms
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=0ms TTL=255
Reply from 30.0.0.20: bytes=32 time=1ms TTL=255
Reply from 30.0.0.20: bytes=32 time=0ms TTL=255
Reply from 30.0.0.20: bytes=32 time=0ms TTL=255
Ping statistics for 30.0.0.20:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 1ms, Average = 0ms

Experiment No-4

Aim : Configuring DHCP within a LAN in a packet Tracer

Topology:



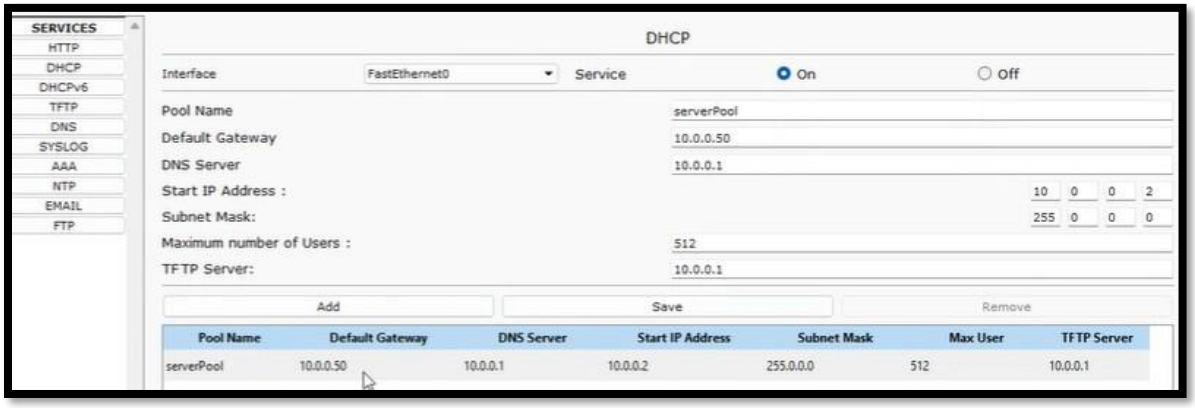
Procedure:

1. First, open the cisco packet tracer desktop and select the devices given below
2. Configure the Server with IPv4 address and Subnet Mask according to the Data given above.
3. Configuring the DHCP server.
4. Configuring Router with IPv4 Address and Subnet Mask.

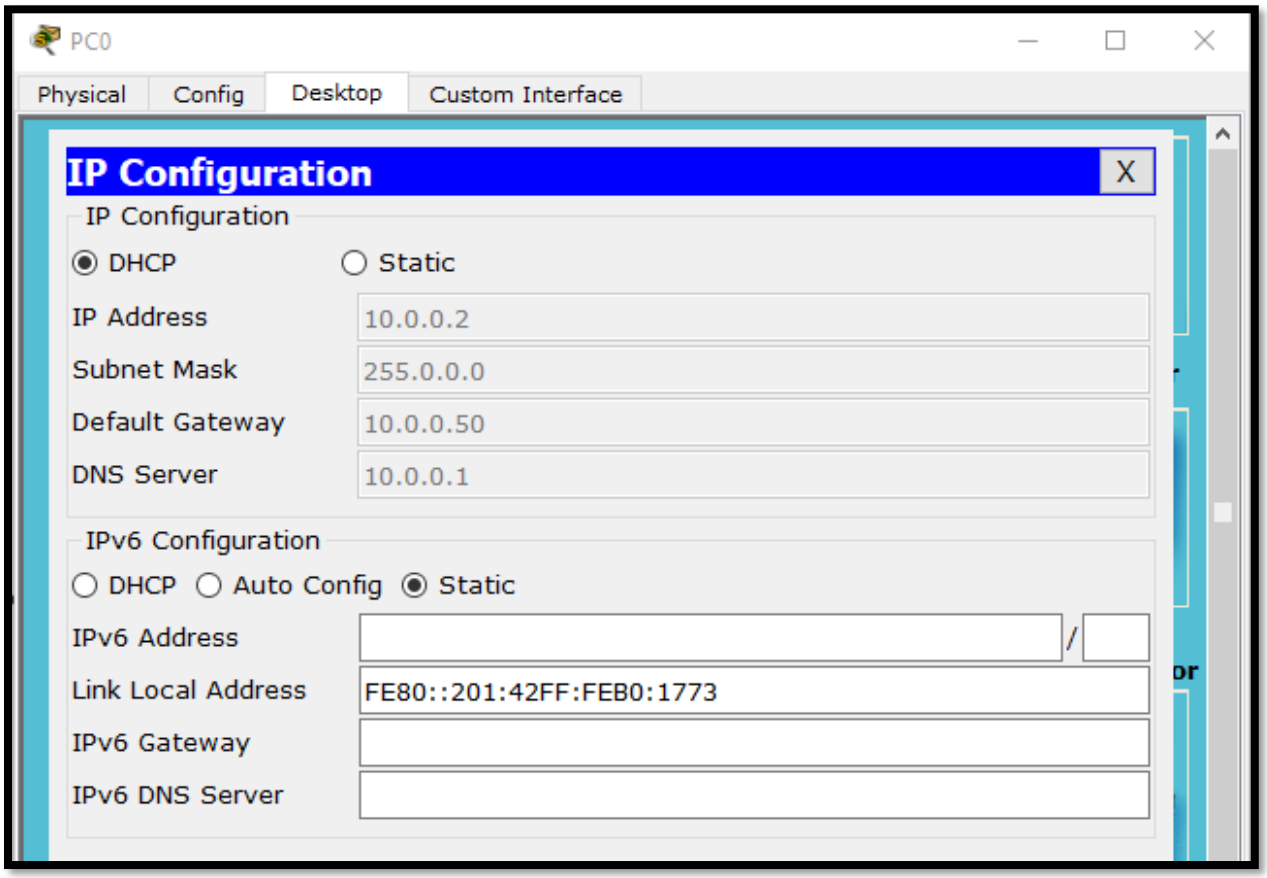
Configuring the PCs and changing the IP configuration.

- 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:

Packet Tracer PC Command Line 1.0

PC>ping 10.0.0.4

Pinging 10.0.0.4 with 32 bytes of data:

Reply from 10.0.0.4: bytes=32 time=0ms TTL=128 Reply from 10.0.0.4: bytes=32 time=0ms TTL=128 Reply from 10.0.0.4: bytes=32 time=0ms TTL=128 Reply from 10.0.0.4: bytes=32 time=0ms TTL=128

Ping statistics for 10.0.0.4:

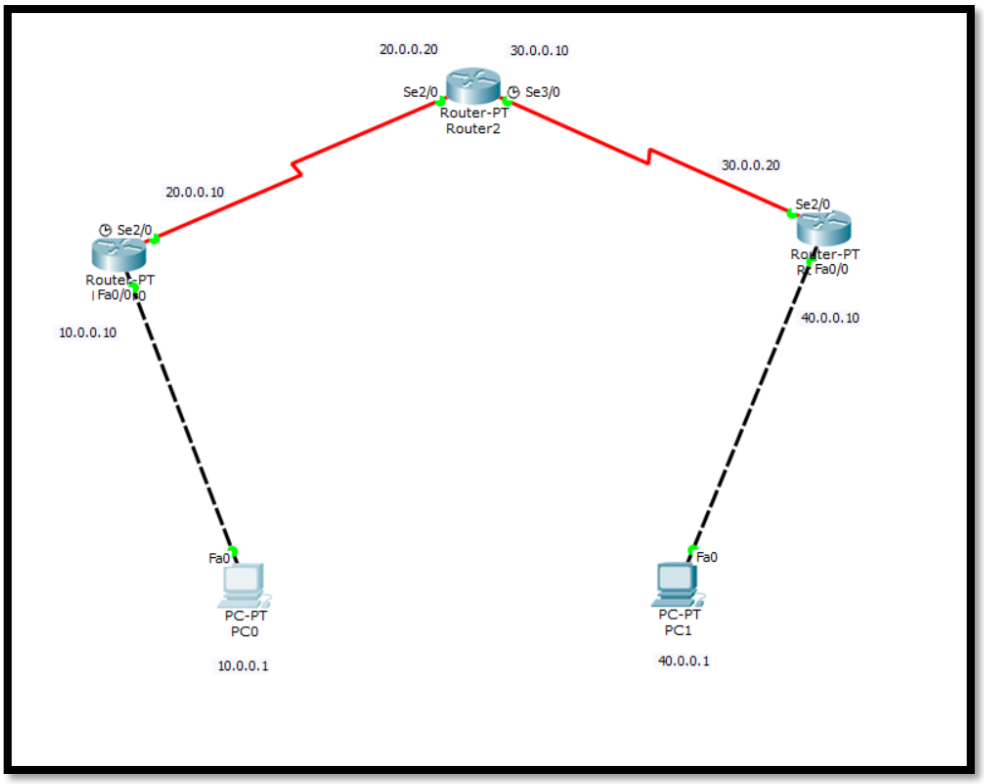
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss), Approximate round trip times in milli-seconds:

Minimum = 0ms, Maximum = 0ms, Average = 0ms

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

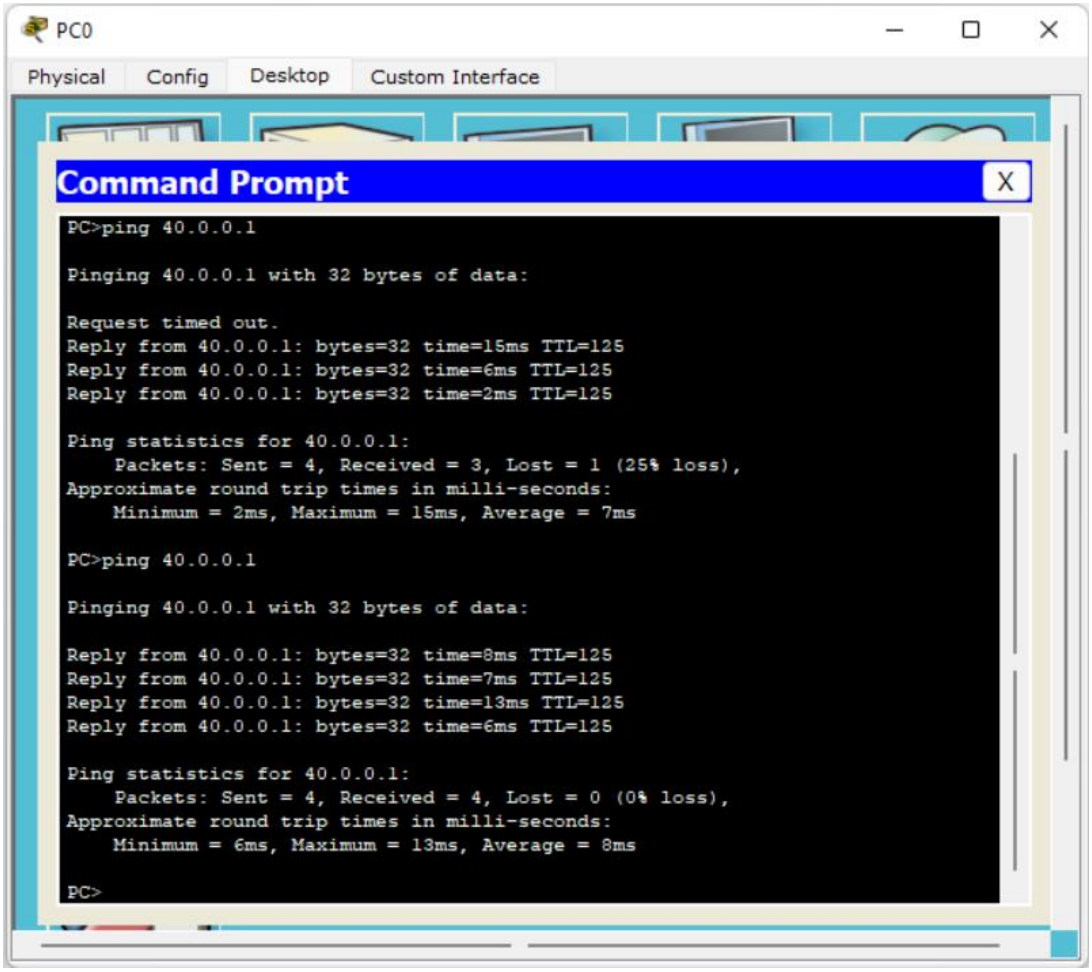
Router (config) #interface serial 3/0

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

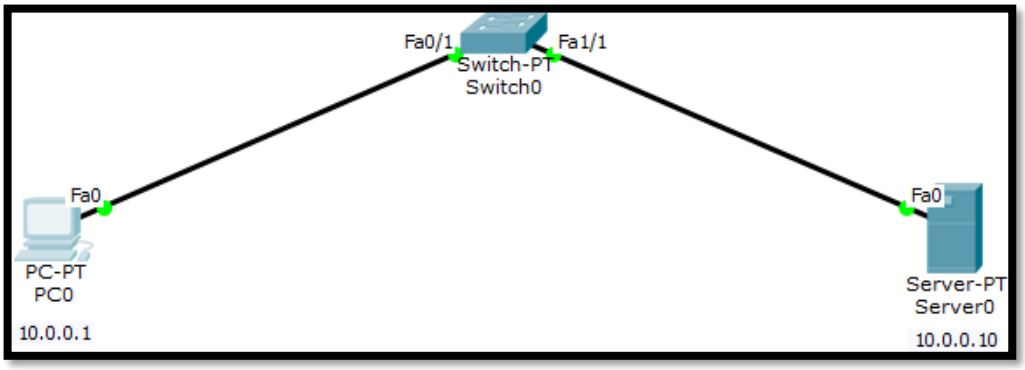
Output:



Experiment No-6

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

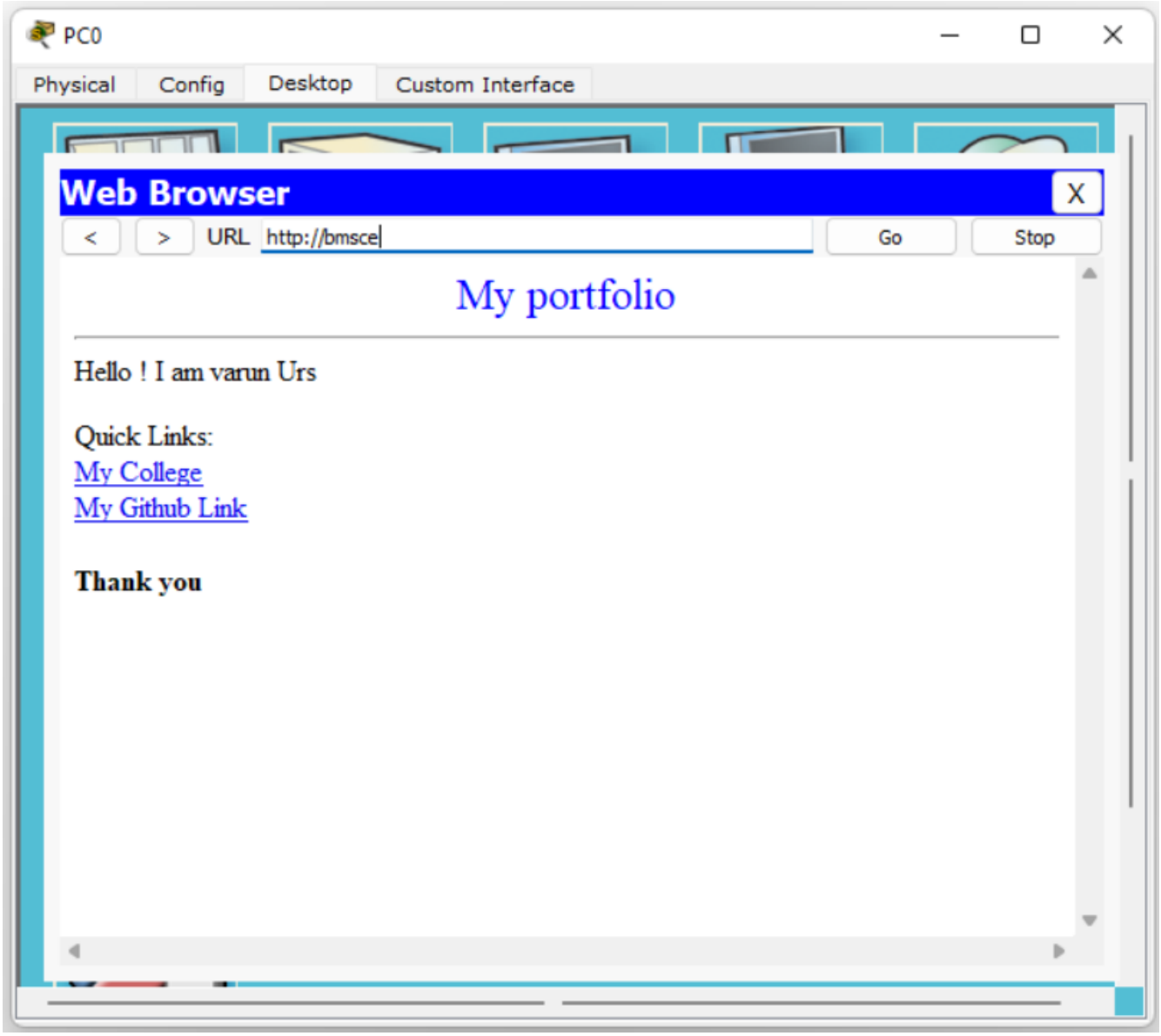
Topology:



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:


PC0
Physical
Config
Desktop
Custom Interface
Web Browser
X
<
>
URL
http://bmsce
Go
Stop
My portfolio
Hello ! I am varun Urs
Quick Links:
My College
My Github Link
Thank you

CYCLE - 2

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

Code :

```
def xor(a, b):
    result = []

    for i in range(1, len(b)):
        if a[i] == b[i]:
            result.append('0')
        else:
            result.append('1')
    return ''.join(result)


def mod2div(dividend, divisor):
    length = len(divisor)
    tmp = dividend[0: length]

    while length < len(dividend):
        if tmp[0] == '1':
            tmp = xor(divisor, tmp) + dividend[length]
        else:
            tmp = xor('0' * length, tmp) + dividend[length]

        length += 1

    if tmp[0] == '1':
        tmp = xor(divisor, tmp)
    else:
        tmp = xor('0' * length, tmp)

    checkword = tmp
    return checkword


def encodeData(data, key):
    keyLength = len(key)
    appended_data = data + '0' * (keyLength - 1)
    remainder = mod2div(appended_data, key)
    codeword = data + remainder
    return codeword,remainder


# Driver code
```

```
data = input("Enter the dataword : ")
key = input("Enter the generator : ")
encodedData,rem = encodeData(data, key)
print("Remainder of mod2 division is : ", rem)
print("Encoded Data (Data + Remainder) : ",encodedData)

newdata = input("Enter the transmitted data : ")
encodedData,rem = encodeData(newdata, key)
print("Remainder of mod2 division is : ", rem)
if int(rem) == 0:
    print("No error in transmitted data")
else:
    print("Error in transmitted data")
```

Output :

```
C:\Python310\python.exe C:/Users/VARUN-PC/PycharmProjects/pythonProject/AI_LAB/1.py
Enter the dataword : 1000100100
Enter the generator : 10001111
Remainder of mod2 division is :  0000111
Encoded Data (Data + Remainder) :  10001001000000111
Enter the transmitted data : 10001001000000111
Remainder of mod2 division is :  0000000
No error in transmitted data

Process finished with exit code 0
```

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

Code :

```
#include<stdio.h>

struct node{
   unsigned dist[20];
   unsigned from[20];
}rt[10];

void bellmanford(int nodes,int costmat[][nodes]){
   for(int i=0;i<nodes;i++)
      for(int j=0;j<nodes;j++)
         for(int k=0;k<nodes;k++)
            if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j]){
               rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
               rt[i].from[j]=k;
            }
}

int main(){
   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];
         rt[i].from[j]=j;
      }
   }

   bellmanford(nodes,costmat);
      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 \tDistance : %d \t Next Hop: %d ",j+1,rt[i].dist[j],rt[i].from[j]+1);
         }
      }
   printf("\n\n");
}
```

Output :

```
Enter the number of nodes : 3

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


 For router 1

node : 1        Distance : 0     Next Hop: 1
node : 2        Distance : 2     Next Hop: 2
node : 3        Distance : 1     Next Hop: 3

 For router 2

node : 1        Distance : 2     Next Hop: 1
node : 2        Distance : 0     Next Hop: 2
node : 3        Distance : 7     Next Hop: 3

 For router 3

node : 1        Distance : 7     Next Hop: 1
node : 2        Distance : 1     Next Hop: 2
node : 3        Distance : 0     Next Hop: 3
```

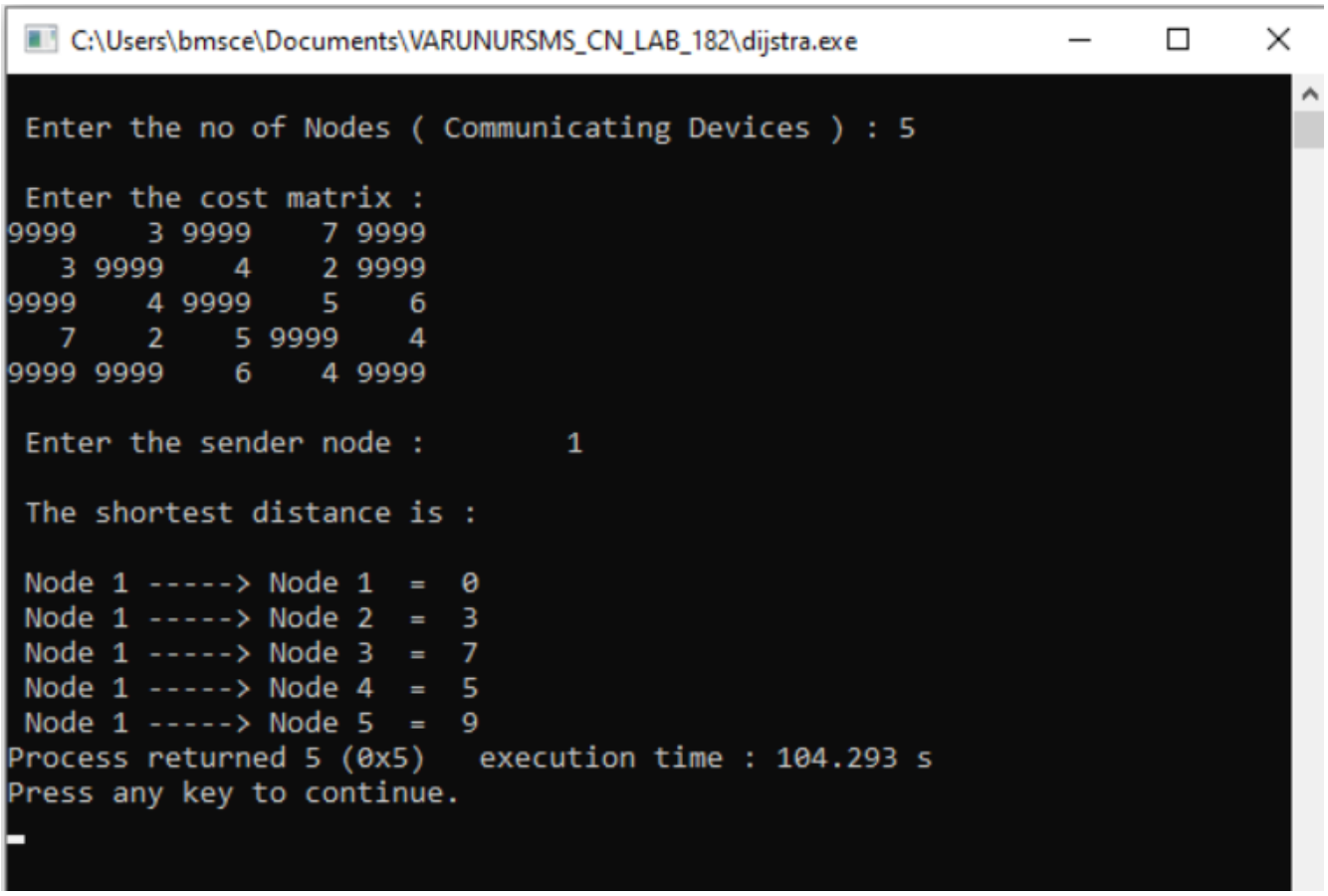
PROGRAM 3 : Implement Djikstra's algorithm to compute the shortest path for a given topology

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();
}

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 :

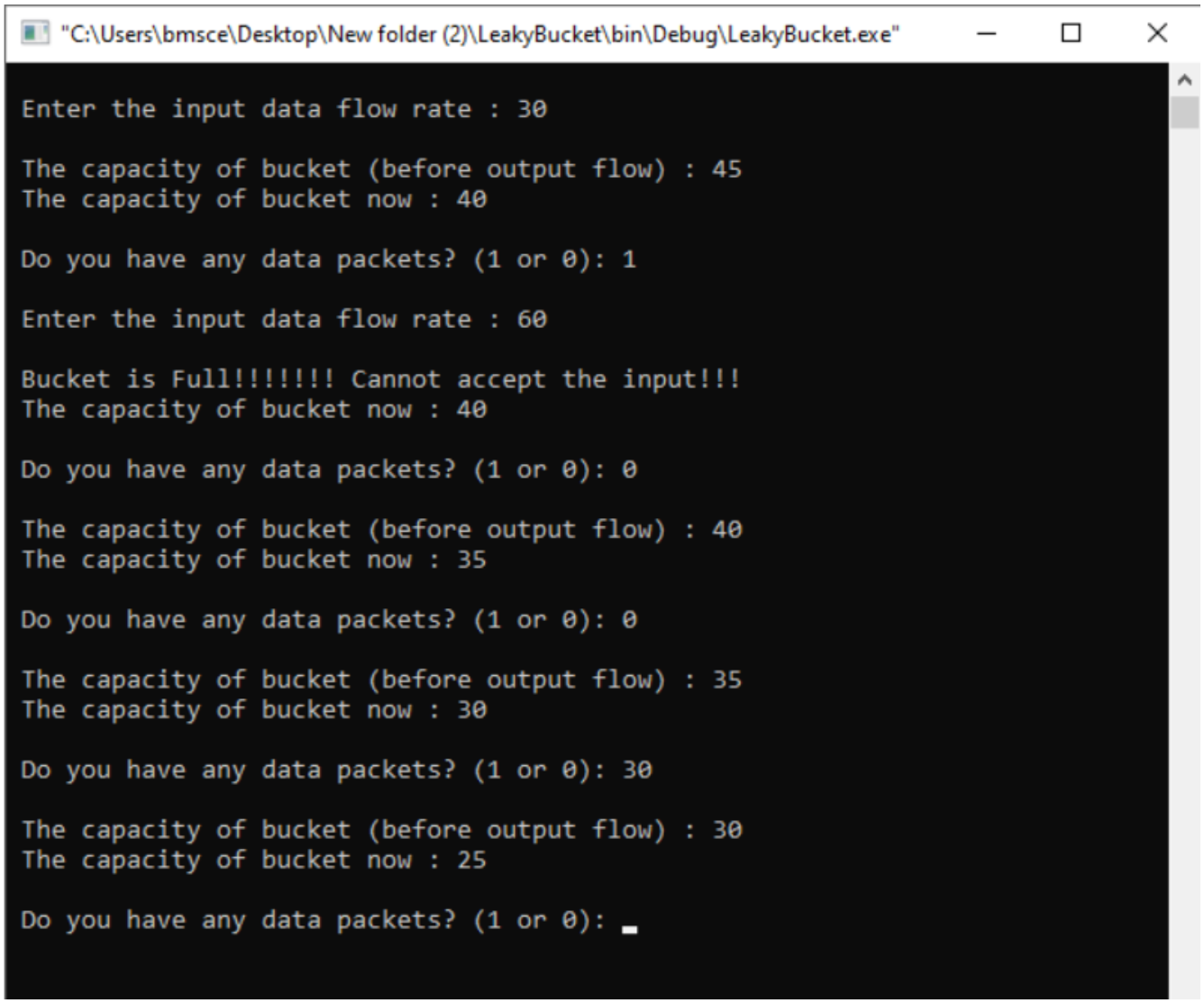


PROGRAM 4 : Write a program for congestion control using Leaky bucket algorithm.

Code :

```
#include<stdio.h>

void main(){
   int bucketSize = 60;
   int inputRate = 0;
   int outputRate = 0;
   int remainingSize = 0;
   int dataPresent = -1;

   printf("\n Enter the size of the bucket : ");
   scanf("%d",&bucketSize);

   printf("\n Enter the output flow rate : ");
   scanf("%d",&outputRate);

   while(1){
      printf("\n Do you have any data packets? (1 or 0): ");
      scanf("%d",&dataPresent);

      if(dataPresent == 1){
         printf("\n Enter the input data flow rate : ");
         scanf("%d",&inputRate);

         if((remainingSize + inputRate) <= bucketSize){
            remainingSize += inputRate;

            printf("\n The present size of bucket (before output flow) : %d",remainingSize);
            remainingSize -= outputRate;
            printf("\n The present size of bucket now : %d\n",remainingSize);
         }
         else{
            printf("\n Bucket is Full!!!!!!! Cannot accept the input!!!");
            printf("\n The present size of bucket now : %d\n",remainingSize);

         }
      }
      else{
         printf("\n The present size of bucket (before output flow) : %d",remainingSize);
         remainingSize -= outputRate;
         printf("\n The present size of bucket now : %d\n",remainingSize);
      }
   }
}
```

Output :


"C:\Users\bmsce\Desktop\New folder (2)\LeakyBucket\bin\Debug\LeakyBucket.exe"
Enter the input data flow rate : 30
The capacity of bucket (before output flow) : 45
The capacity of bucket now : 40
Do you have any data packets? (1 or 0): 1
Enter the input data flow rate : 60
Bucket is Full!!!!!!! Cannot accept the input!!!
The capacity of bucket now : 40
Do you have any data packets? (1 or 0): 0
The capacity of bucket (before output flow) : 40
The capacity of bucket now : 35
Do you have any data packets? (1 or 0): 0
The capacity of bucket (before output flow) : 35
The capacity of bucket now : 30
Do you have any data packets? (1 or 0): 30
The capacity of bucket (before output flow) : 30
The capacity of bucket now : 25
Do you have any data packets? (1 or 0):

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 :

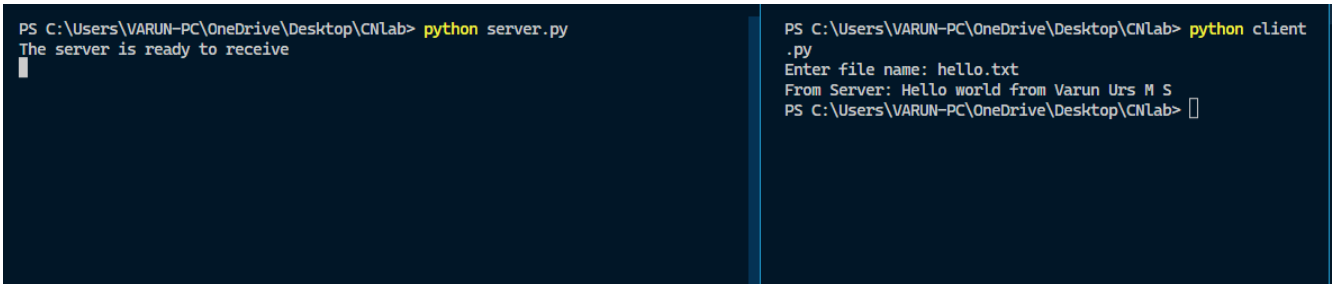
Server :

```
from socket import *
serverName='DESKTOP-KGIIO2U'
serverPort=14000
serverSocket=socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
print ("The server is ready to receive")
while 1:
    connectionSocket,addr=serverSocket.accept()
    sentence=connectionSocket.recv(1024).decode()
    file=open(sentence,"r")
    l=file.read(1024)
    connectionSocket.send(l.encode())
    file.close()
    connectionSocket.close()
```

Client :

```
from socket import *
serverName='DESKTOP-KGIIO2U'
serverPort=14000
clientSocket=socket(AF_INET,SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence=input("Enter file name: ")
clientSocket.send(sentence.encode())
filecontents=clientSocket.recv(1024).decode()
print('From Server:',filecontents)
clientSocket.close()
```

Output



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 :

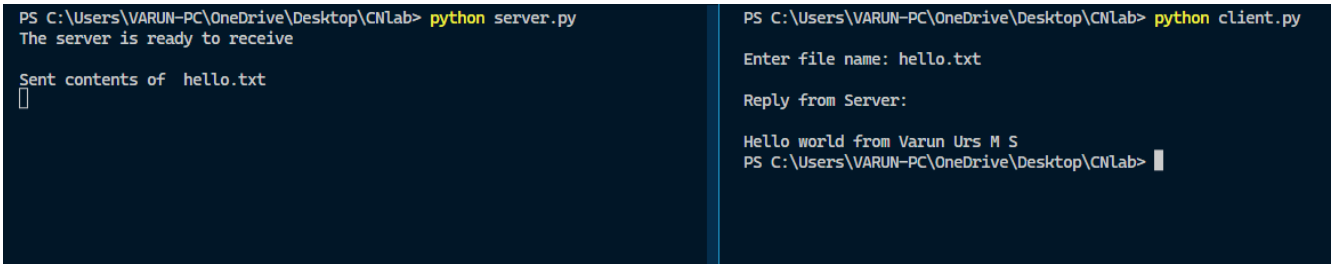
SERVER

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print ("The server is ready to receive")
while 1:
 sentence, clientAddress = serverSocket.recvfrom(2048)
 sentence = sentence.decode("utf-8")
 file=open(sentence,"r")
 l=file.read(2048)

 serverSocket.sendto(bytes(l,"utf-8"),clientAddress)
 print ('\nSent contents of ', end = ' ')
 print (sentence)
 # for i in sentence:
 # print (str(i), end = '')
 file.close()
```

CLIENT:

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("\nEnter file name: ")
clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))
filecontents,serverAddress = clientSocket.recvfrom(2048)
print ('\nReply from Server:\n')
print (filecontents.decode("utf-8"))
# for i in filecontents:
 # print(str(i), end = '')
clientSocket.close()
clientSocket.close()
```

Output :


PS C:\Users\VARUN-PC\OneDrive\Desktop\CNlab> python server.py
The server is ready to receive
Sent contents of hello.txt
PS C:\Users\VARUN-PC\OneDrive\Desktop\CNlab> python client.py
Enter file name: hello.txt
Reply from Server:
Hello world from Varun Urs M S
PS C:\Users\VARUN-PC\OneDrive\Desktop\CNlab>