VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

COMPUTER NETWORKS

Submitted by

PARAMESHWAR S (1BM21CS407)

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 "COMPUTER NETWORKS LAB" carried out by **PARAMESHWAR S (1BM121CS407),** who is 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. Latha N R
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

`

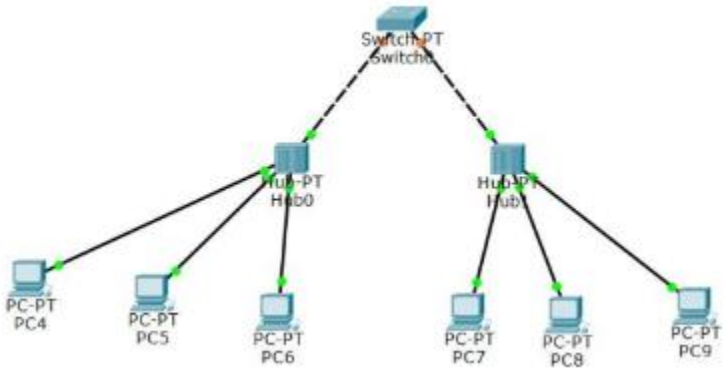
Index

Experiment 1

Cycle-1

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

Topology



Procedure

1. We have taken a switch and linked it to four end devices.
2. Link every device with the switch.
3. Provide the IP address to each device.
4. Transfer messages from one device to another and check the Table for Validation.

Output Screenshot

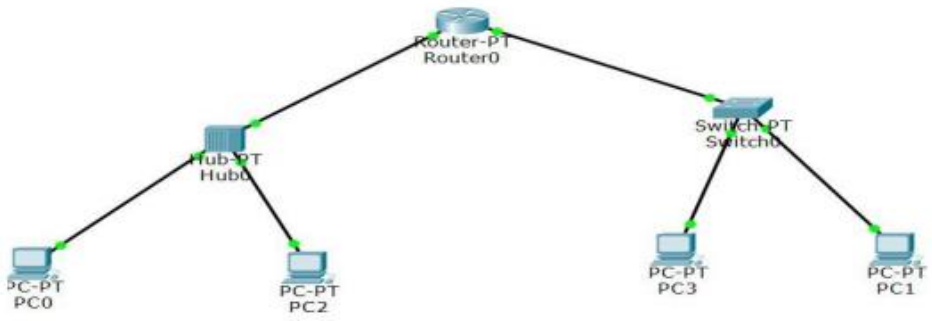
```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.4

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=1ms 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 = 1ms, Average = 0ms

PC>
```

Experiment 2

Cycle-1

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

Topology



Procedure

1. Select the router and Open CLI.

2. Press ENTER to start configuring Router1.

3. Type enable to activate the privileged mode.

4. Type config t(configure terminal) to access the configuration menu.

5. Configure interfaces of Router1:.

Output Screenshot

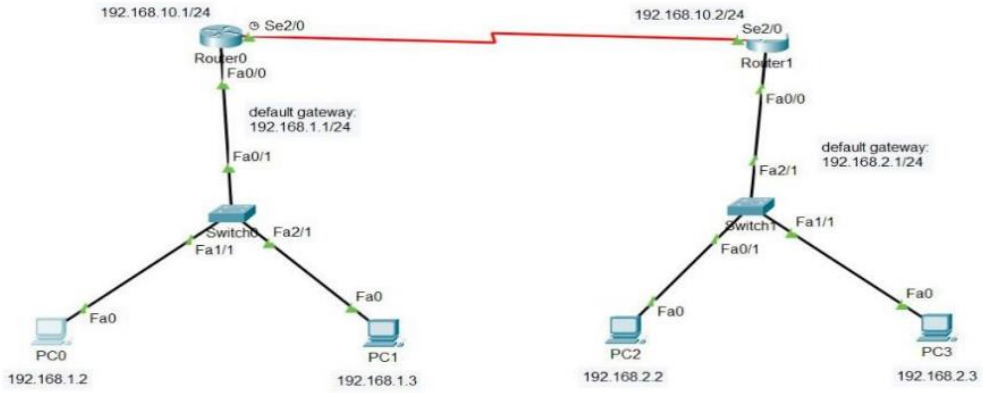
```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.4

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=1ms 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 = 1ms, Average = 0ms

PC>
```

Experiment 3

Cycle-1

Aim of the program: Configuring default route to the Router.

Topology



Procedure

1. First, create a network topology of these given devices listed below in the table.

2. Configuring Hosts (PCs) with IP addresses and Default Gateway using IP Addressing table given below.

3. Configuring the Interfaces (routers) with IP Addresses and Default gateways and assigning the default routes.

4. After configuring all the devices, the red indicator turns into green and the network is live so we can send and receive packets.

Output Screenshot

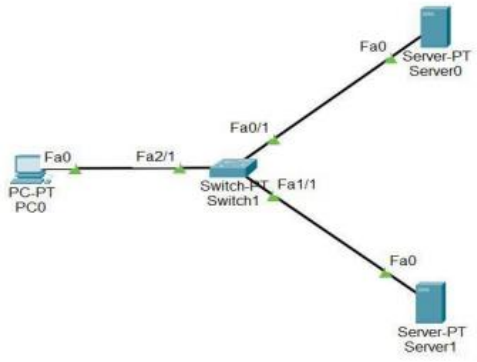
```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.2.3

Pinging 192.168.2.3 with 32 bytes of data:

Reply from 192.168.2.3: bytes=32 time=13ms TTL=126
Reply from 192.168.2.3: bytes=32 time=9ms TTL=126
Reply from 192.168.2.3: bytes=32 time=8ms TTL=126
Reply from 192.168.2.3: bytes=32 time=1ms TTL=126

Ping statistics for 192.168.2.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 13ms, Average = 7ms

C:\>
```

Experiment 4

Cycle-1

Aim of the program 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.

5. Configuring the PCs and changing the IP configuration

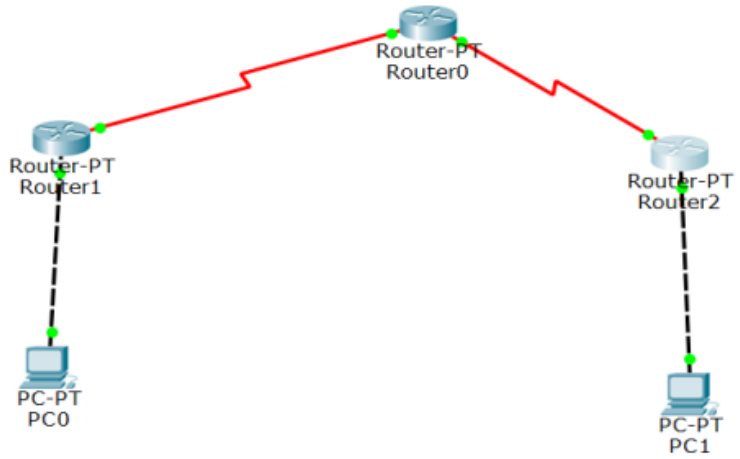
Output Screenshot

Experiment 5

Cycle-1

Aim of the program Configuring RIP Routing Protocol in Routers

Topology



Procedure

1. Configure the network interfaces. This example shows multiple loopback interface addresses to simulate attached networks. ...

2. Create the RIP group and add the interface. ...

3. Create the routing policy to advertise both direct and RIP-learned routes. ...

4. Apply the routing policy

Output Screenshot

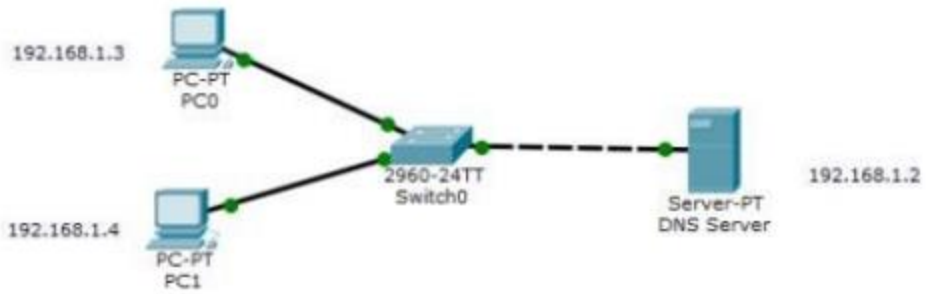
```
Packet Tracer PC Command Line 1.0
PC>ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

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

Experiment 6

Cycle-1

Aim of the program Demonstration of WEB server and DNS using Packet Tracer

Topology



Procedure

1. Configure the network interfaces. This example shows multiple loopback interface addresses to simulate attached networks. ...

2. Create the RIP group and add the interface. ...

3. Create the routing policy to advertise both direct and RIP-learned routes. ...

4. Apply the routing policy

Output Screenshot

```
Packet Tracer PC Command Line 1.0
C:\>ping pc1
Ping request could not find host pc1. Please check the name and try again.
C:\>ping pc1

Pinging 192.168.1.4 with 32 bytes of data:

Reply from 192.168.1.4: bytes=32 time=77ms TTL=128
Reply from 192.168.1.4: bytes=32 time<1ms TTL=128
Reply from 192.168.1.4: bytes=32 time<1ms TTL=128
Reply from 192.168.1.4: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
```

Cycle-2

Program 1

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

```
  return checkword
def encodeData(data, key):
  l_key = len(key)
  appended_data = data + '0'*(l_key-1)
  remainder = mod2div(appended_data, key)
  codeword = data + remainder
  print("Remainder : ", remainder)
  print("Encoded Data (Data + Remainder) : ",
     codeword)
data = "100100"
key = "10001000000100001"
encodeData(data, key)
```

Output Screenshot

```
Remainder :  0110010011100110
Encoded Data (Data + Remainder) :  1001000110010011100110

Process finished with exit code 0
```

Program 2

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

Code:

```
#include<stdio.h>
struct node
{
   unsigned dist[20];
   unsigned from[20];
}rt[10];
int main()
{
   int  costmat[20][20];
   int nodes,i,j,k,count=0;
   printf("\nEnter the number of nodes : ");
   scanf("%d",&nodes);//Enter the 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];//initialise the distance equal to cost matrix
         rt[i].from[j]=j;
      }
```

```
}
   do
   {
      count=0;
      for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the
direct distance from the node i to k using the cost matrix
      //and add the distance from k to node j
      for(j=0;j<nodes;j++)
      for(k=0;k<nodes;k++)
        if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
        {//We calculate the minimum distance
          rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
          rt[i].from[j]=k;
          count++;
        }
   }while(count!=0);
   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 via %d Distance %d
",j+1,rt[i].from[j]+1,rt[i].dist[j]);
      }
   }
 printf("\n\n");
```

Output Screenshot

```
Enter the number of nodes : 7

Enter the cost matrix :
0 2 99 3 99 99 99
2 0 5 99 4 99 99
99 5 0 99 99 4 3
3 99 99 0 5 99 99
99 4 99 5 0 2 99
99 99 4 99 2 0 1
99 99 3 99 99 1 0


 For router 1

node 1 via 1 Distance 0
node 2 via 2 Distance 2
node 3 via 2 Distance 7
node 4 via 4 Distance 3
node 5 via 2 Distance 6
node 6 via 2 Distance 8
node 7 via 2 Distance 9

 For router 2

node 1 via 1 Distance 2
node 2 via 2 Distance 0
node 3 via 3 Distance 5
node 4 via 1 Distance 5
node 5 via 5 Distance 4
node 6 via 5 Distance 6
node 7 via 5 Distance 7
```

```
 For router 3

node 1 via 2 Distance 7
node 2 via 2 Distance 5
node 3 via 3 Distance 0
node 4 via 2 Distance 10
node 5 via 6 Distance 6
node 6 via 6 Distance 4
node 7 via 7 Distance 3

 For router 4

node 1 via 1 Distance 3
node 2 via 1 Distance 5
node 3 via 1 Distance 10
node 4 via 4 Distance 0
node 5 via 5 Distance 5
node 6 via 5 Distance 7
node 7 via 5 Distance 8

 For router 5

node 1 via 2 Distance 6
node 2 via 2 Distance 4
node 3 via 6 Distance 6
node 4 via 4 Distance 5
node 5 via 5 Distance 0
node 6 via 6 Distance 2
node 7 via 6 Distance 3
```

```
 For router 6

node 1 via 5 Distance 8
node 2 via 5 Distance 6
node 3 via 3 Distance 4
node 4 via 5 Distance 7
node 5 via 5 Distance 2
node 6 via 6 Distance 0
node 7 via 7 Distance 1

 For router 7

node 1 via 6 Distance 9
node 2 via 6 Distance 7
node 3 via 3 Distance 3
node 4 via 6 Distance 8
node 5 via 6 Distance 3
node 6 via 6 Distance 1
node 7 via 7 Distance 0


Process returned 0 (0x0)   execution time : 439.178 s
Press any key to continue.
```

Program 3

Aim: To implement Dijkstra'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 num 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 Screenshot

```
Enter the num of vertices:      7

Enter the cost matrix:
0 2 99 3 99 99 99
2 0 5 99 4 99 99
99 5 0 99 99 4 3
3 99 99 0 5 99 99
99 4 99 5 0 2 99
99 99 4 99 2 0 1
99 99 3 99 99 1 0

Enter the source node:  1

The shortest distance is:

1----->1 = 0
1----->2 = 2
1----->3 = 7
1----->4 = 3
1----->5 = 6
1----->6 = 8
1----->7 = 9
Process returned 7 (0x7)   execution time : 128.944 s
Press any key to continue.
```

Program 4

Aim: To write a program for congestion control using Leaky bucket algorithm.

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
void main()
{
int i,packets[10],content=0,newcontent,time,clk,bcktsize,oprate;
for(i=0;i<5;i++)
{
packets[i]=rand()%10;
if(packets[i]==0) --i;
}
printf("\n Enter output rate of the bucket: \n");
scanf("%d",&oprate);
printf("\n Enter Bucketsize\n");
scanf("%d",&bcktsize);
for(i=0;i<5;++i)
{
if((packets[i]+content)>bcktsize)
{
if(packets[i]>bcktsize)
printf("\n Incoming packet size %d greater than the size of the
bucket\n",packets[i]);
```

```
else
printf("\n bucket size exceeded\n");
}
else
{
newcontent=packets[i];
content+=newcontent;
printf("\n Incoming Packet : %d\n",newcontent);
printf("\n Transmission left : %d\n",content);
time=rand()%10;
printf("\n Next packet will come at %d\n",time);
for(clk=0;clk<time && content>0;++clk)
{
printf("\n Left time %d",(time-clk));
if(content)
{
printf("\n Transmitted\n");
if(content<oprate)
content=0;
else
content=content-oprate;
printf("\n Bytes remaining : %d\n",content);
}
else
printf("\n No packets to send\n");
```

Output Screenshot

```
 Enter output rate of the bucket:
4

 Enter Bucketsize
5

 Incoming Packet : 1

 Transmission left : 1

 Next packet will come at 8

 Left time 8
 Transmitted

 Bytes remaining : 0

 Incoming packet size 7 greater than the size of the bucket

 Incoming Packet : 4

 Transmission left : 4

 Next packet will come at 8

 Left time 8
 Transmitted

 Bytes remaining : 0

 Incoming packet size 7 greater than the size of the bucket

 Incoming Packet : 4

 Transmission left : 4

 Next packet will come at 8

 Left time 8
 Transmitted

 Bytes remaining : 0

 Incoming packet size 9 greater than the size of the bucket

 Incoming Packet : 4

 Transmission left : 4

 Next packet will come at 2

 Left time 2
 Transmitted

 Bytes remaining : 0
```

Program 5

Aim: Using TCP/IP sockets, to write a client-server program to make the client send the file name and the server to send back the contents of the requested file if present.

Code:

servertcp.py

```
from socket import *
serverName='LAPTOP-HATRKFO6'
serverPort = 12530
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()
```

clienttcp.py

```
from socket import *
serverName = 'LAPTOP-HATRKFO6'
serverPort = 12530
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 Screenshot

```
The server is ready to receive
```

```
Enter file nameservertcp.py
From Server: from socket import *
serverName='LAPTOP-HATRKFO6'
serverPort = 12530
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()
```

Program 6

Aim: Using UDP sockets, write a client-server program to make the client send the file name and the server to send back the contents of the requested file if present.

Code:

serverudp.py

```
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)

    file=open(sentence,"r")

        l=file.read(2048)

    serverSocket.sendto(bytes(l,"utf-8"),clientAddress)

        print("sent back to client",l)

        file.close()
```

clientudp.py

```
from socket import *

serverName = "127.0.0.1"

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 Screenshot

```
The server is ready to receive
sent back to client 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)

     file=open(sentence,"r")
     l=file.read(2048)

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

```
Enter file nameserverudp.py
From Server: b'from socket import *\nserverPort = 12000\nserverSocket = socket(AF_INET, SOCK_DGRAM)\nserverSocket.bind(("127.0.0.1",
serverPort))\nprint ("The server is ready to receive")\nwhile 1:\n     sentence,clientAddress = serverSocket.recvfrom(2048)\n     \n
   file=open(sentence,"r")\n     l=file.read(2048)\n  \n     serverSocket.sendto(bytes(l,"utf-8"),clientAddress)\n     print("sent ba
ck to client",l)\n     file.close()'
```