VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

COMPUTER NETWORKS

Submitted by

PRASANNA KUMAR R (1BM21CS138)

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
June-2023 to September-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" carried out by **PRASANNA KUMAR R(1BM21CS138),** 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 academic semester June-2023 to September-2023. The Lab report has been approved as it satisfies the academic requirements in respect of a **COMPUTER NETWORKS (22CS4PCCON)** work prescribed for the said degree.

Swathi Sridharan
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

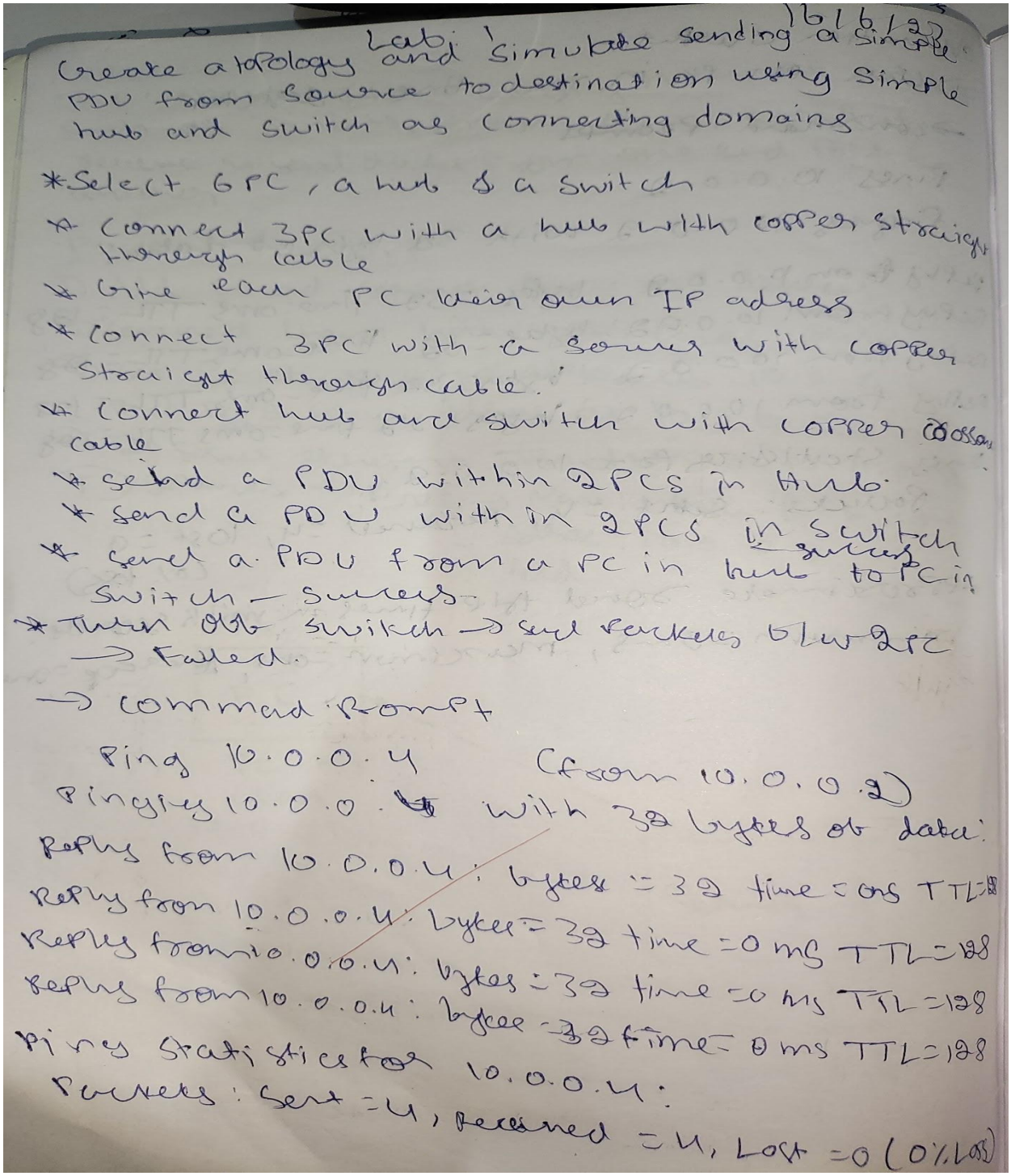
INDEX SHEET

Course Outcomes

CO1	Apply the fundamental concepts of communication in networking.
CO2	Analyze the various protocols, techniques in TCP/IP network architectur
CO3	Develop programs that demonstrate the functionalities of physical, Data Link, Network, Transport or Application layer

Experiment 1

Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.



Through switch

Ping 10.0.0.4 (from 10.0.0.2)

Pinging 10.0.0.4 with 32 bytes of data:

Request time out

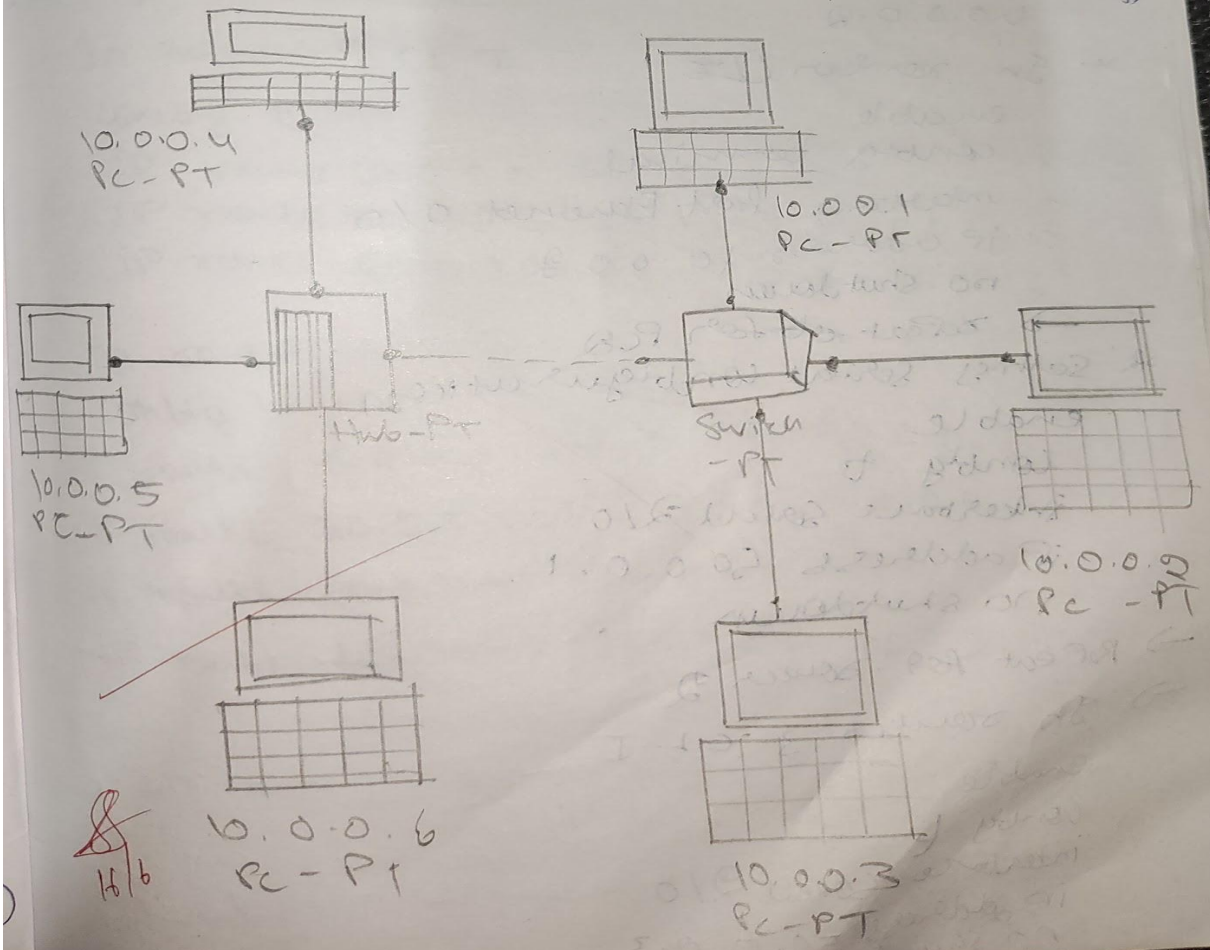
Request time out

Request time out

Request time out

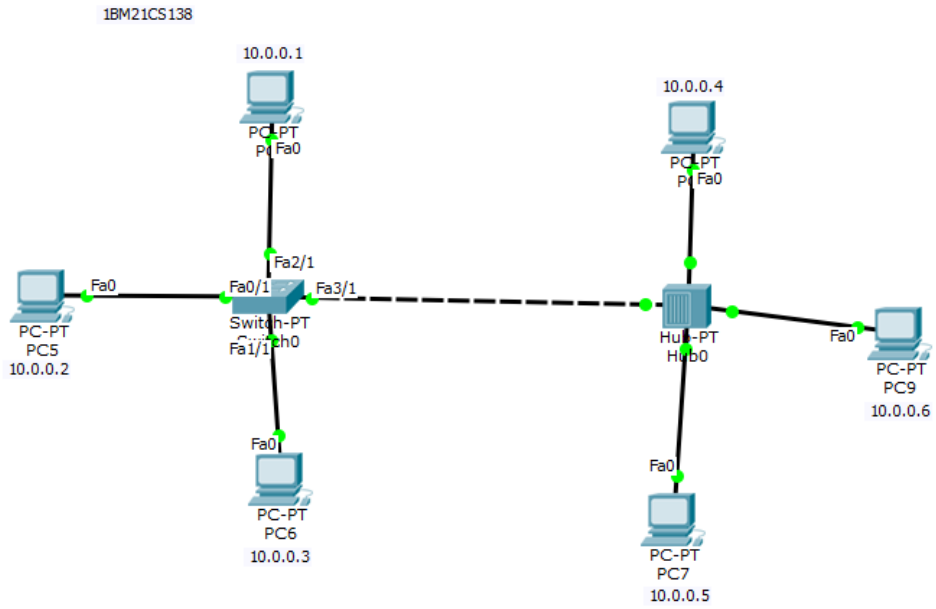
Ping Statistics for 10.0.0.4:

Packets: Sent = 4, Received = 0, Lost = 4 (100% loss)

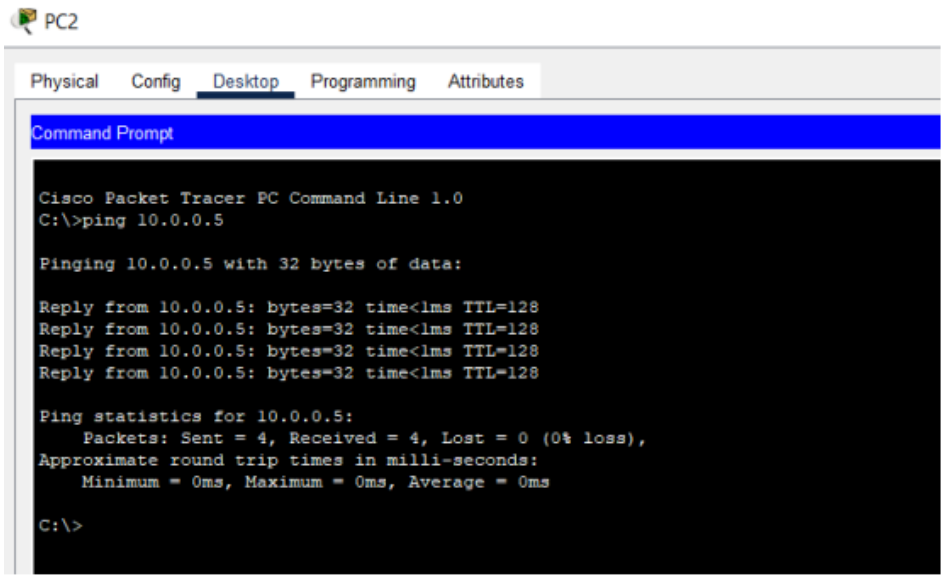


16/6

Topology:

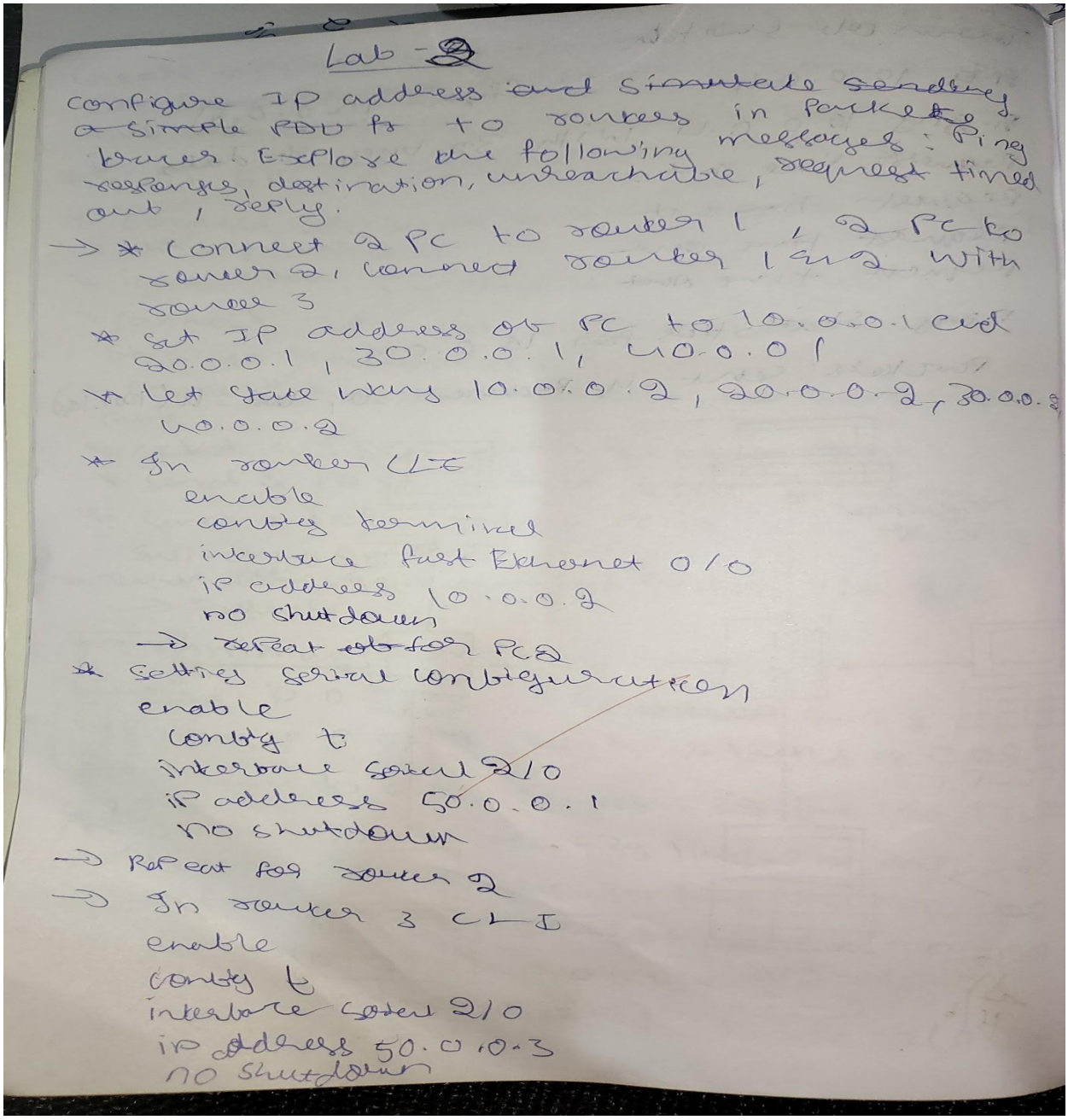


Output:



Experiment 2

Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply



Lab - 2

Configure IP address and simulate sending a simple PDU ~~fr~~ to routers in packet tracer. Explore the following messages: Ping responses, destination, unreachable, request timed out, reply.

→ * Connect a PC to router 1, a PC to router 2, connect router 1 and 2 with router 3
* Set IP address of PC to 10.0.0.1 and 20.0.0.1, 30.0.0.1, 40.0.0.1
* Let gate ways 10.0.0.2, 20.0.0.2, 30.0.0.2, 40.0.0.2
* In router CLI
  enable
  config terminal
  interface fast Ethernet 0/0
  ip address 10.0.0.2
  no shutdown
  → Repeat ~~of~~ for PC2
* Setting serial configuration
  enable
  config t
  interface serial 2/0
  ip address 50.0.0.1
  no shutdown
→ Repeat for router 2
→ In router 3 CLI
  enable
  config t
  interface serial 2/0
  ip address 50.0.0.3
  no shutdown

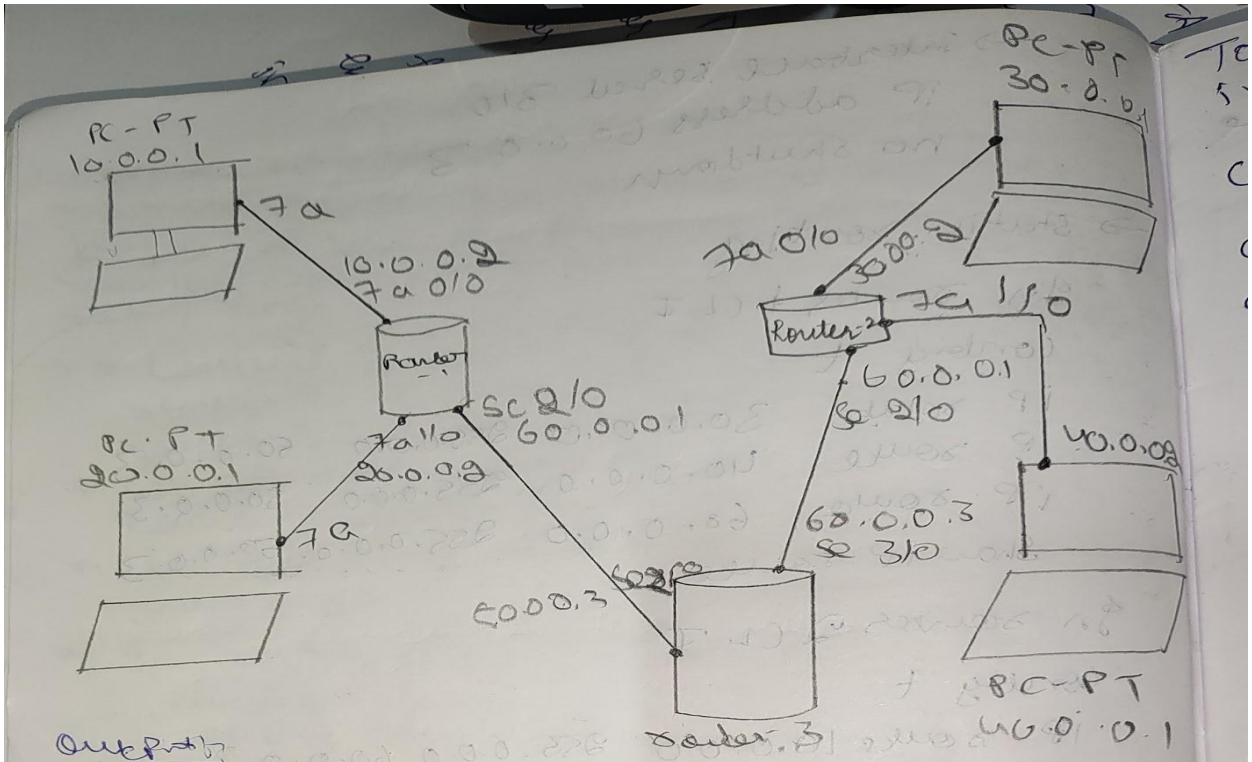
→ interface serial 3/0
ip address 60.0.0.3
no shutdown

- static routing

On router 1 CLI
config t
ip route 30.0.0.0 255.0.0.0 50.0.0.3
ip route 40.0.0.0 255.0.0.0 50.0.0.3
ip route 60.0.0.0 255.0.0.0 50.0.0.3
Show ip route

On router 2 CLI
config t
ip route 10.0.0.0 255.0.0.0 60.0.0.3
ip route 20.0.0.0 255.0.0.0 60.0.0.3
ip route 50.0.0.0 255.0.0.0 60.0.0.3

On router 3 CLI
config t
ip route 10.0.0.0 255.0.0.0 50.0.0.1
ip route 20.0.0.0 255.0.0.0 50.0.0.1
ip route 30.0.0.0 255.0.0.0 60.0.0.1
ip route 40.0.0.0 255.0.0.0 60.0.0.1



Output

In PC1 (10.0.0.1)

Ping 30.0.0.1

Pinging 30.0.0.1 with 32 bytes of data:

Reply from 10.0.0.2: Destination host unreachable

Reply from 10.0.0.2: Destination host unreachable

Reply from 10.0.0.2: Destination host unreachable

Reply from 10.0.0.2: Destination host unreachable

Ping Statistics for 30.0.0.1:

Packet: Sent = 4, Received = 0, Lost = 4 (100% loss)

Since 30.0.0.1 is not connected to PC1 directly it should be statically registered

To view ip route.

in PC1

show ip route

C 10.0.0.0/3 is directly connected, Fast ethernet 0/0

C 20.0.0.0/3 is directly connected, Fast ethernet 1/0

S 30.0.0.0/3 [1/0] via 50.0.0.3

S 40.0.0.0/8 [1/0] via 50.0.0.3

C 50.0.0.0/8 is directly connected, serial 2/0

S 60.0.0.0/8 [1/0] via 50.0.0.3

In PC1

Ping 30.0.0.1

Pinging 30.0.0.1 with 32 bytes of data

request timed out

Reply from 30.0.0.1 : bytes=32 time=5ms TTL=125

Reply from 30.0.0.1 : bytes=32 time=2ms TTL=125

Reply from 30.0.0.1 : bytes=32 time=10ms TTL=125

Ping statics for 30.0.0.1

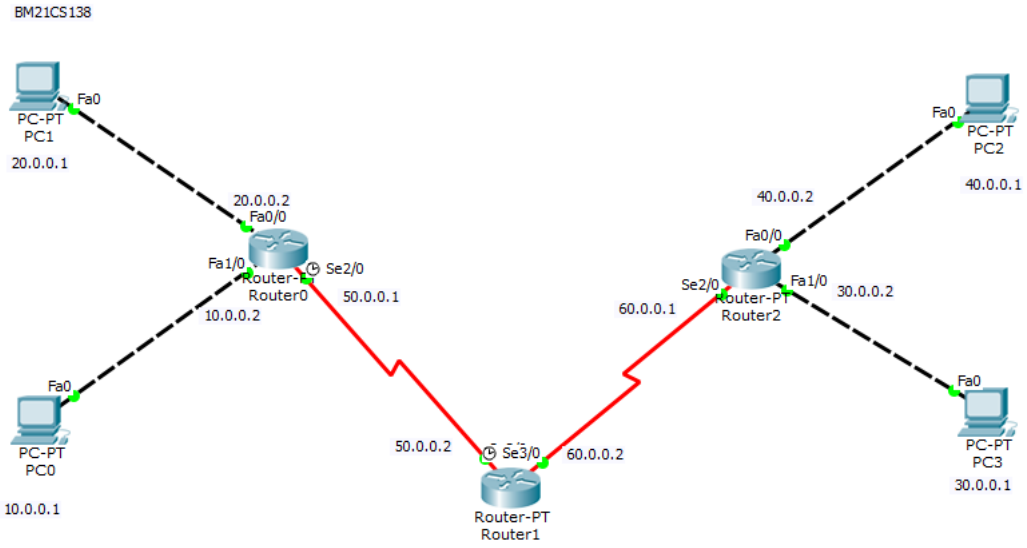
Packets: sent = 4 received = 3 lost = 1(25% loss)

Approximate round trip times in milli seconds

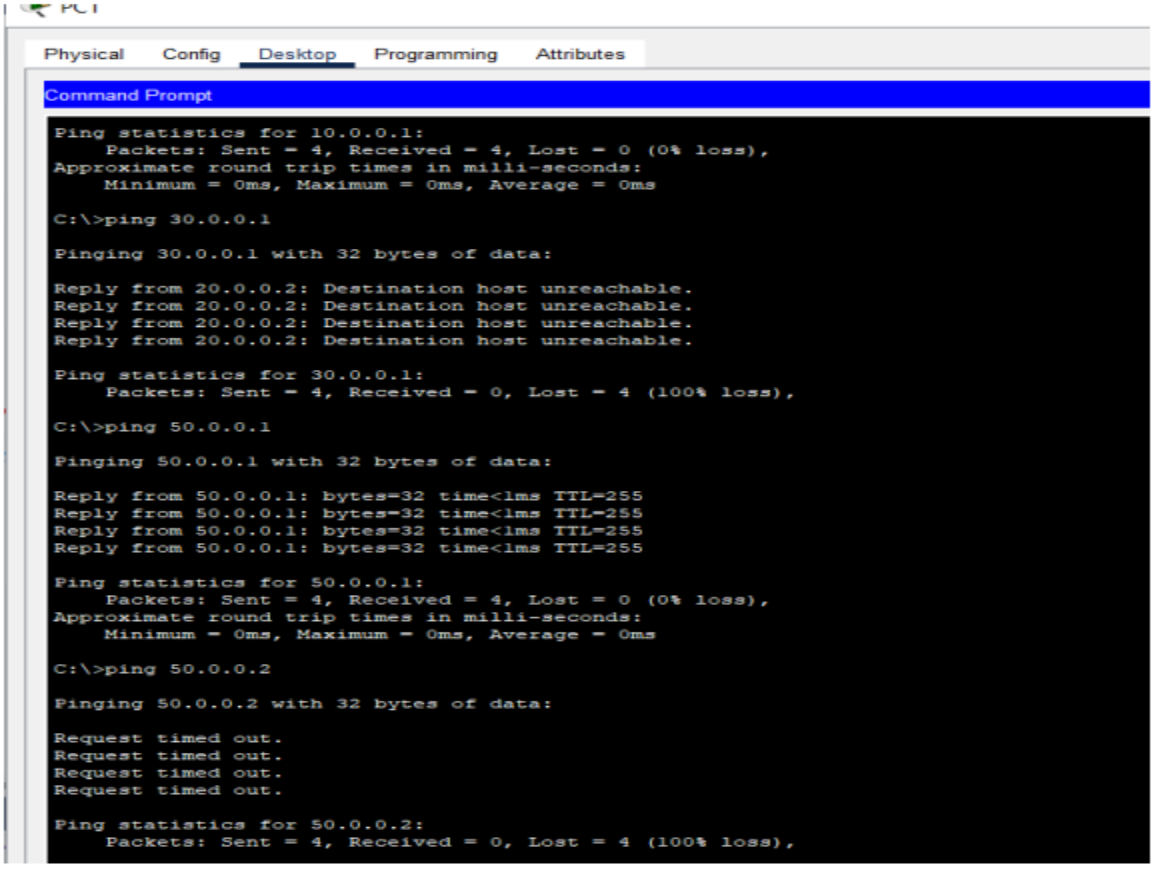
Minimum = 0ms, Maximum = 10ms, Average = 5ms

First is lost since it takes time to identify path.

Topology:

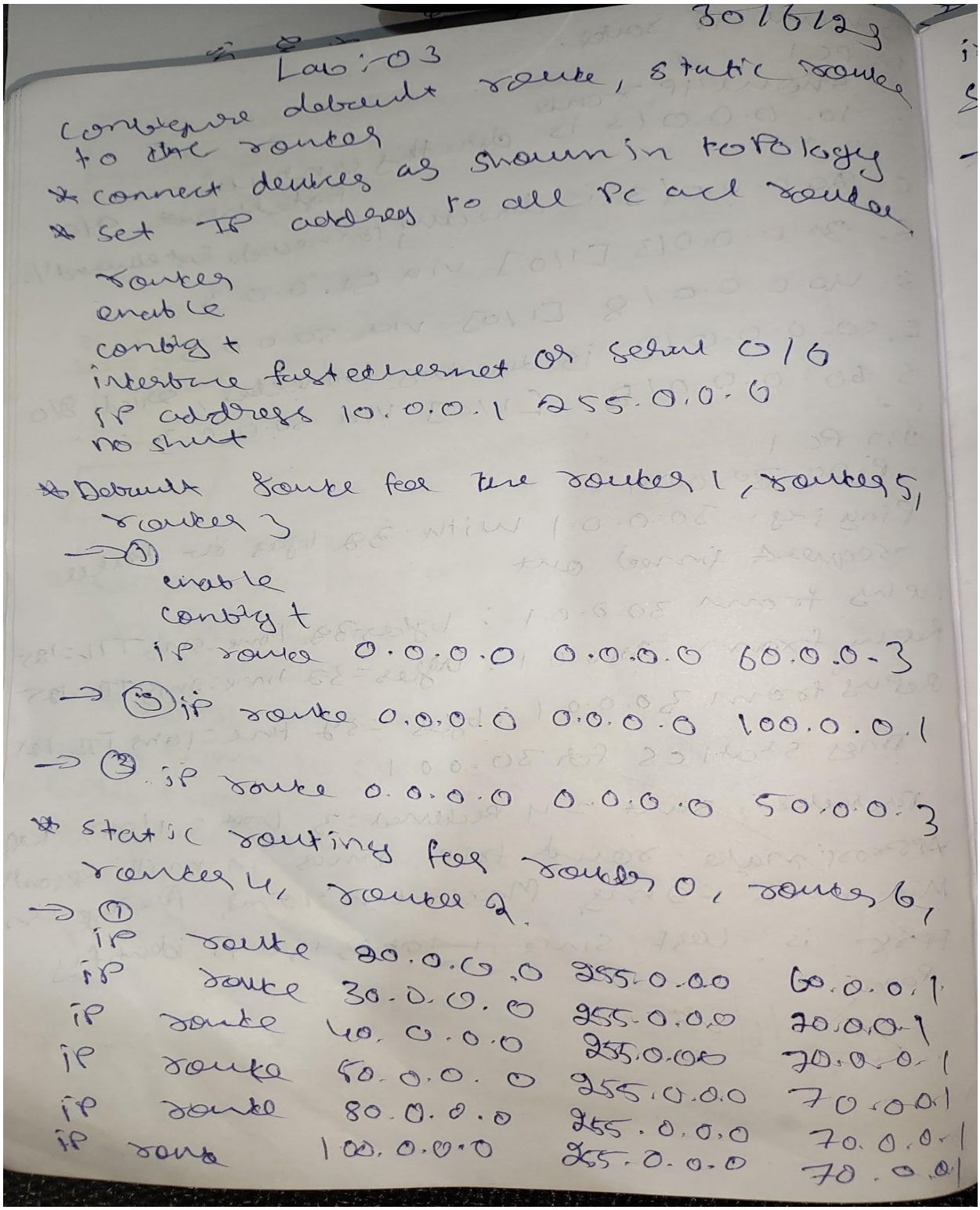


Output:



Experiment 3

Configure default route, static route to the Router



Lab:03

Configure default route, static route to the router

* connect devices as shown in topology
* set IP address to all PC and router.

router
enable
config t
interface fastethernet 0/0 or serial 0/0
IP address 10.0.0.1 255.0.0.0
no shut

* Default route for the router 1, router 5, router 3

→ (1)
enable
config t
ip route 0.0.0.0 0.0.0.0 60.0.0.3

→ (3) ip route 0.0.0.0 0.0.0.0 100.0.0.1

→ (3) ip route 0.0.0.0 0.0.0.0 50.0.0.3

* static routing for router 0, router 6, router 4, router 2.

→ (1)
ip route 20.0.0.0 255.0.0.0 60.0.0.1
ip route 30.0.0.0 255.0.0.0 70.0.0.1
ip route 40.0.0.0 255.0.0.0 70.0.0.1
ip route 50.0.0.0 255.0.0.0 70.0.0.1
ip route 80.0.0.0 255.0.0.0 70.0.0.1
ip route 100.0.0.0 255.0.0.0 70.0.0.1

if route 10.0.0.0 255.0.0.0 70.0.0.1
Similarly for rest routers

→ output :-

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 = 7ms TTL = 126
Reply from 20.0.0.1 : bytes = 32 time = 6ms TTL = 126
Reply from 20.0.0.1 : bytes = 32 time = 7ms TTL = 126

Ping statistics for 20.0.0.1 :

Packets : Sent = 4, Received = 3, Lost = 1 (25% loss)
minimum = 6ms, maximum = 7ms, Average = 6ms

9/10
N
LS 14/7/23

To view ip route.

in PC 1

show ip route

C 10.0.0.0/3 is directly connected — Fast ethernet 0/0

C 20.0.0.0/3 is directly connected, Fast ethernet 1/0

S 30.0.0.0/3 [1/0] via 50.0.0.3

S 40.0.0.0/8 [1/0] via 50.0.0.3

C 50.0.0.0/8 is directly connected, serial 2/0

S 60.0.0.0/8 [1/0] via 50.0.0.3

In PC 1

Ping 30.0.0.1

Pinging 30.0.0.1 with 32 bytes of data

request timed out

Reply from 30.0.0.1 : bytes=32 time=5ms TTL=125

Reply from 30.0.0.1 : bytes=32 time=2ms TTL=125

Reply from 30.0.0.1 : bytes=32 time=10ms TTL=125

Ping statics for 30.0.0.1

Packets: sent = 4 received = 3 lost = 1(25%) loss

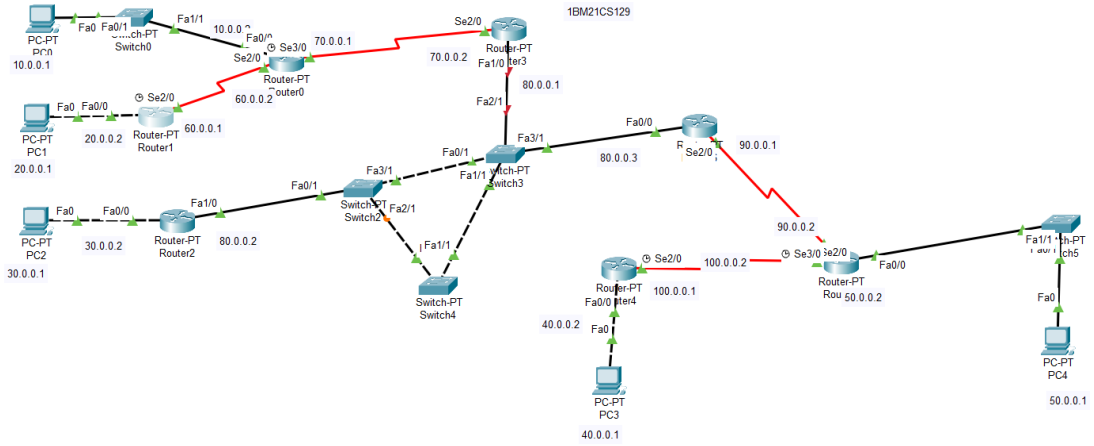
Approximate round trip times in milli seconds

Minimum = 0ms, Maximum = 10ms, Average = 5ms

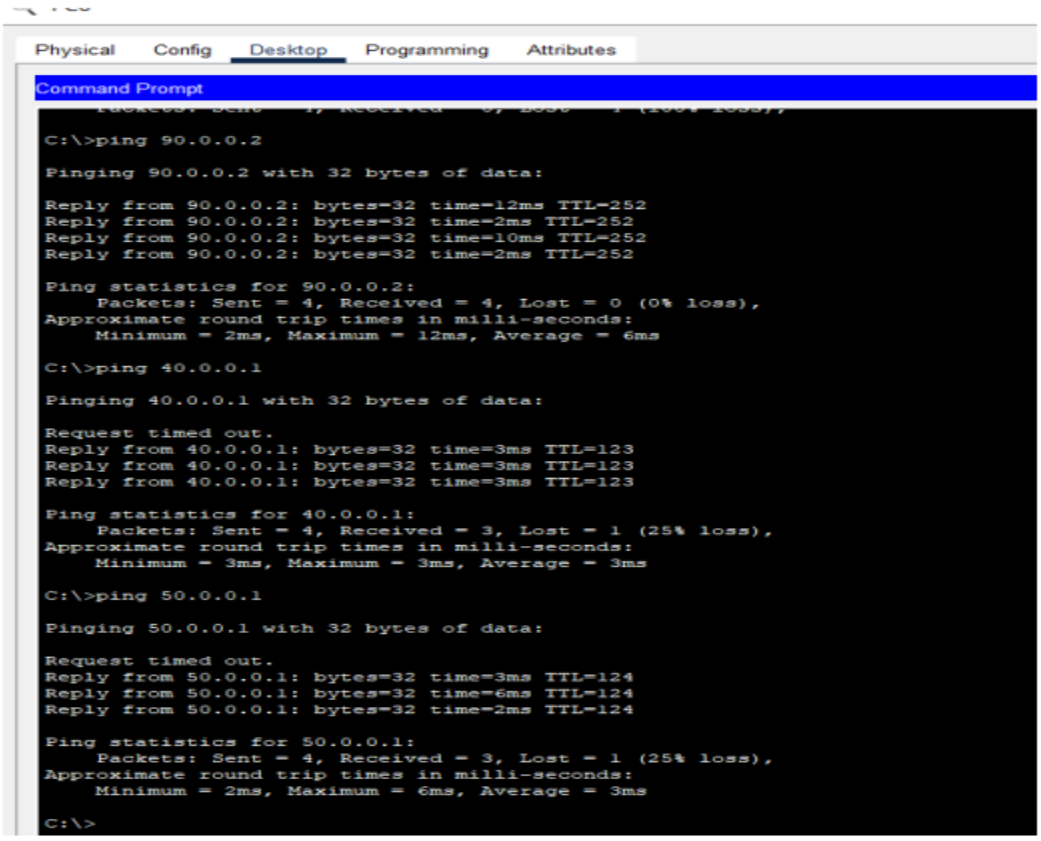
First is lost since it takes time to identify Path.

28/6/23

Topology:

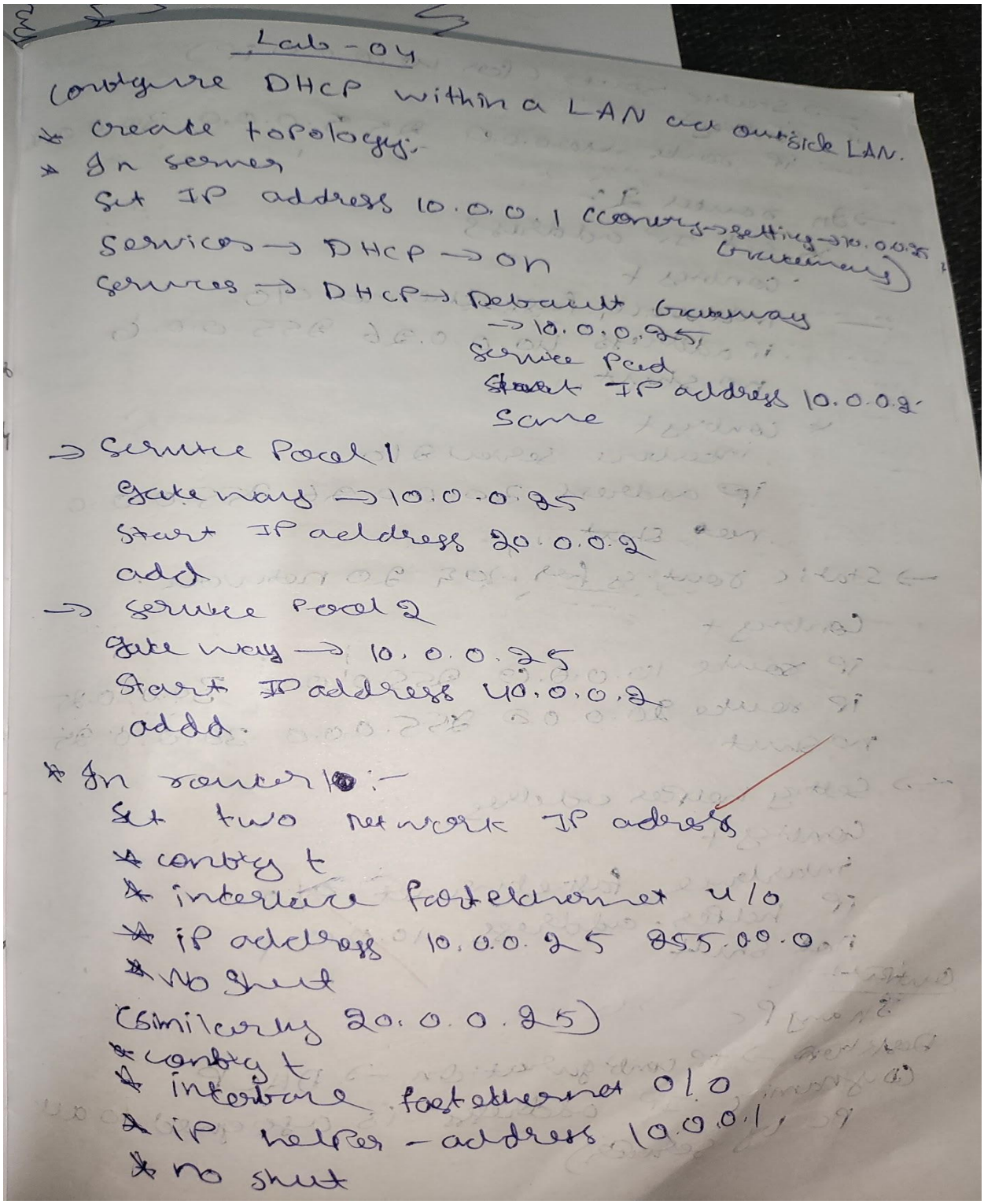


Output:



Experiment 4

Configure DHCP within a LAN and outside LAN



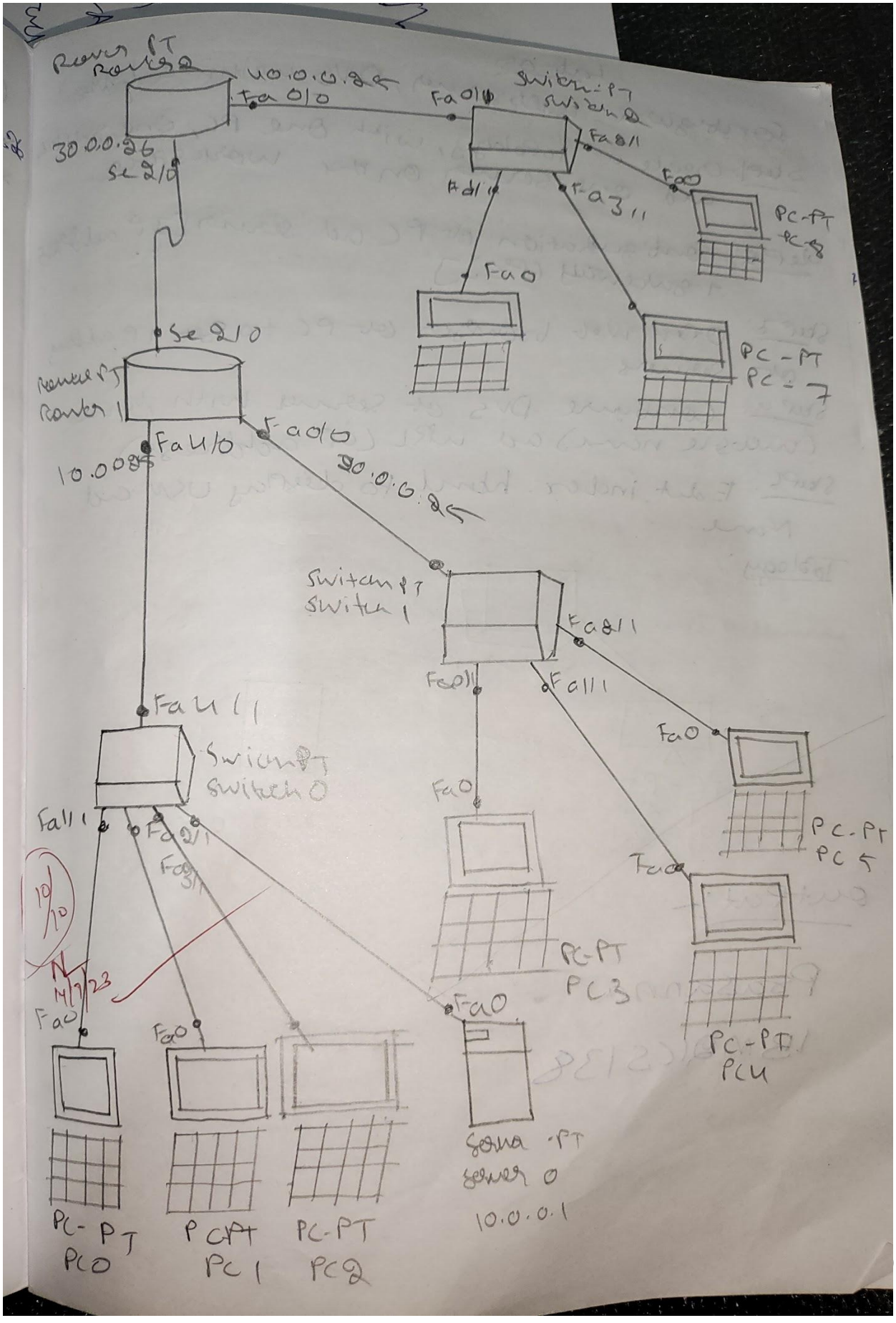
→ Static route (for 40.0.0.0 IP)
conf t
ip route 40.0.0.0 255.0.0.0 30.0.0.2

→ In router 2:
* Set IP address
conf t
interface fast ethernet 0/0
ip address 40.0.0.26 255.0.0.0
no shut
* conf t
interface serial 2/0
ip address 30.0.0.26 255.0.0.0
no shut

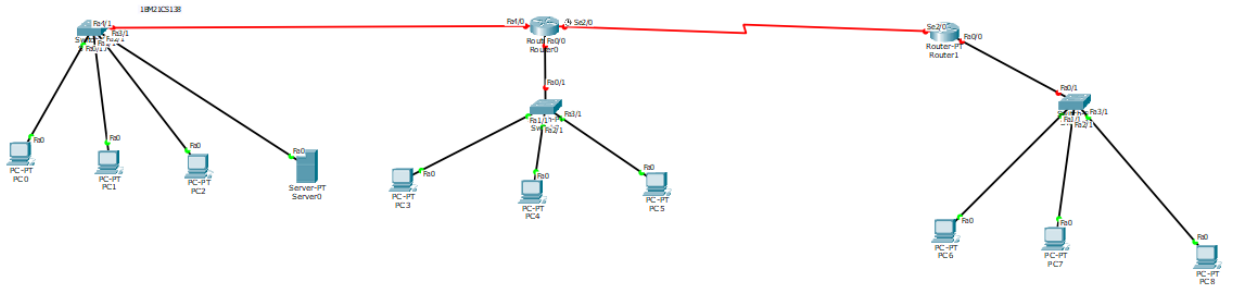
→ Static routing for 10 & 20 network.
Conf t
ip route 10.0.0.0 255.0.0.0 30.0.0.25
ip route 20.0.0.0 255.0.0.0 30.0.0.25
no shut

→ Setting helper address
Conf t
interface fastethernet 0/0
ip helper-address 10.0.0.1

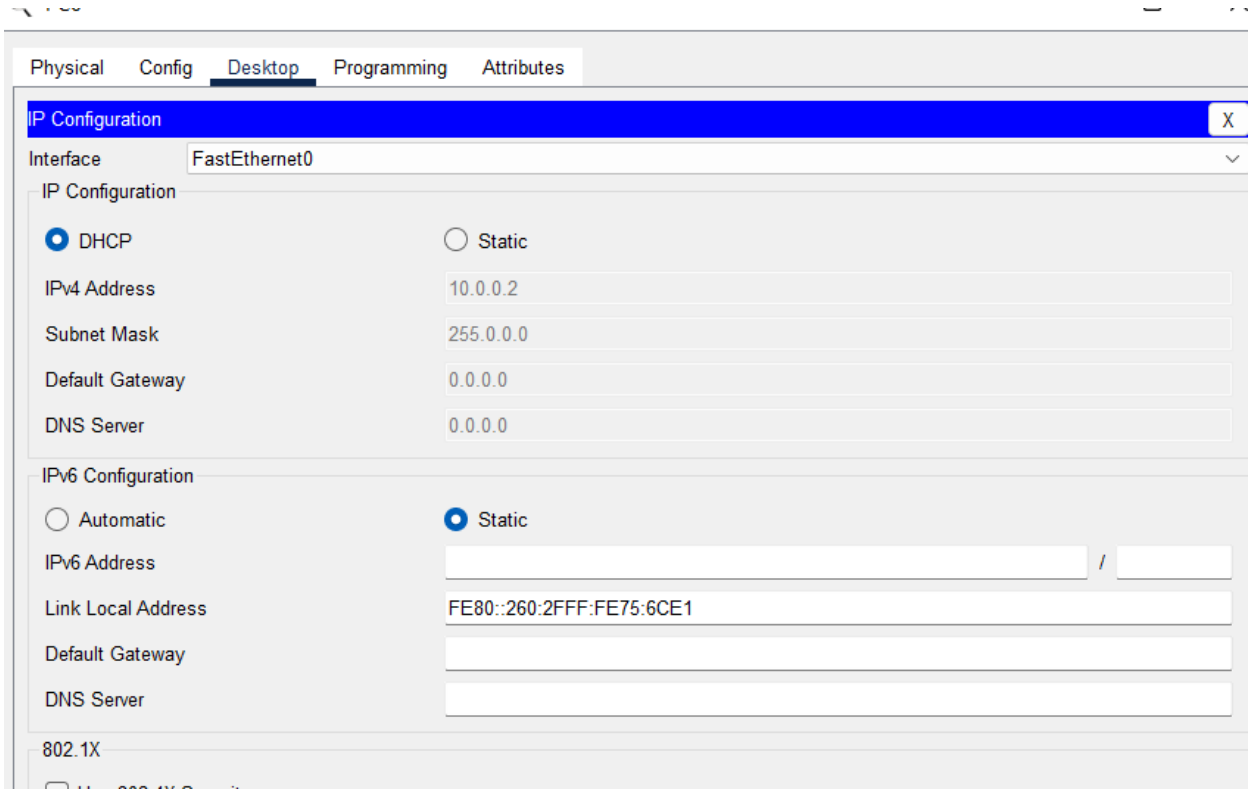
Output:
In any Pc
Desktop → IP configuration → DHCP
(Dynamic IP address is assigned to all Pc by server)


Router-PT
Router 2
40.0.0.2
Fa0/0
Fa0/0
Switch-PT
Switch 2
Fa0/1
30.0.0.2
Se2/0
Fa0/1
Fa0/2
Fa0
PC-PT
PC-8
Fa0
PC-PT
PC-7
Se2/0
Router-PT
Router 1
Fa4/0
10.0.0.2
Fa0/0
30.0.0.1
Switch-PT
Switch 1
Fa0/1
Fa0/1
Fa0/1
Fa4/1
Switch-PT
Switch 0
Fa0
PC-PT
PC5
Fa0
PC-PT
PC3
Fa0
PC-PT
PC4
Fa0/1
Fa0/2
Fa0/3
10/10
14/7/23
Fa0
Fa0
Fa0
Server-PT
Server 0
10.0.0.1
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2

Topology:

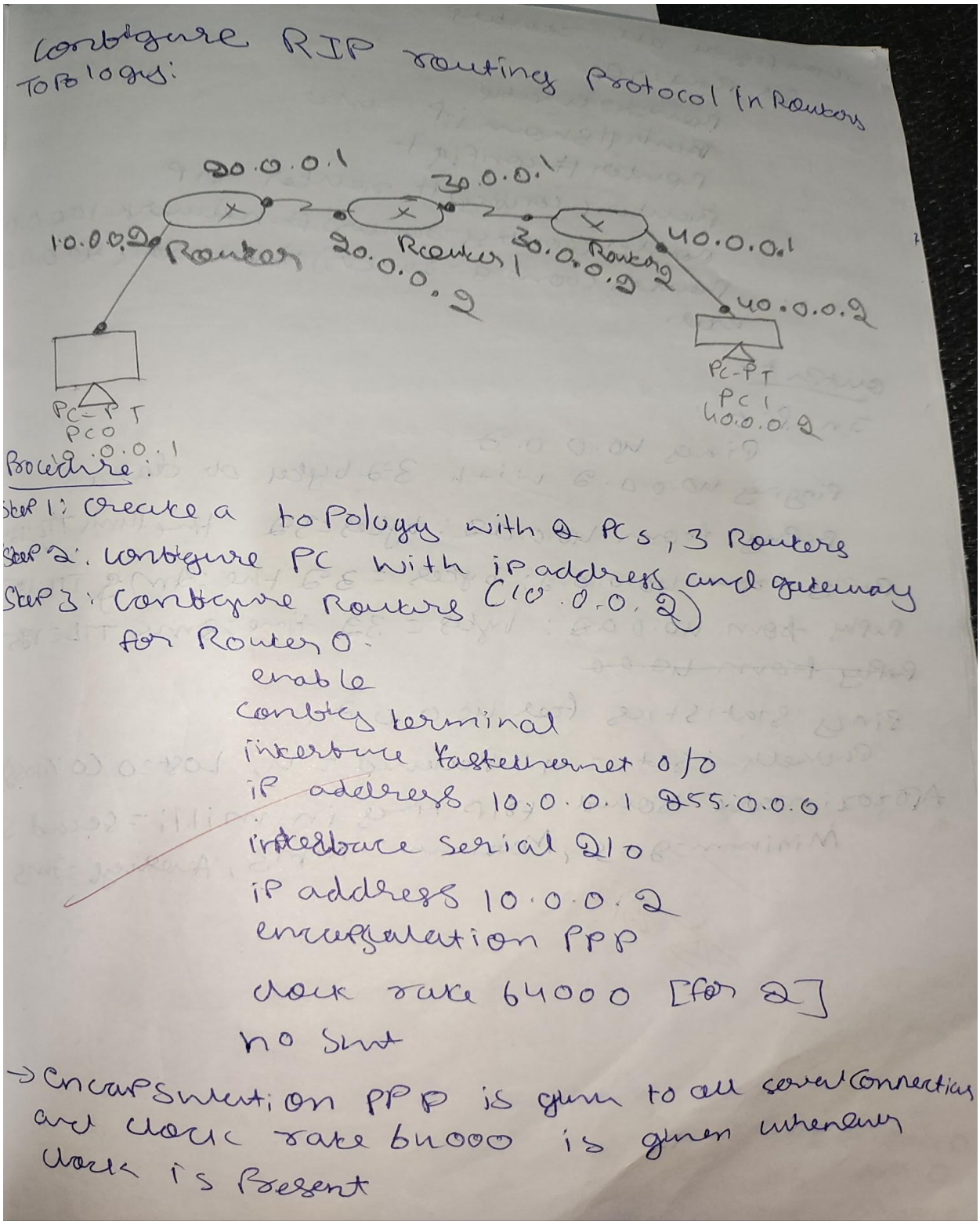


Output:



Experiment 5

Configure RIP routing Protocol in Routers



>configure all routers

for Router 0,

Router> enable

Router# show ip route

Router# config t

Router(config)# router rip

Router(config-router)# network 10.0.0.0

Router(config-router)# network 20.0.0.0

exit

output:-

In PC 0

Ping 40.0.0.2

Pinging 40.0.0.2 with 32 bytes of data:

Reply from 40.0.0.2: bytes=32 time=14ms TTL=128

Reply from 40.0.0.2: bytes = 32 time=2ms TTL: 125

Reply from 40.0.0.2: bytes = 32 time=2ms TTL = 125

~~Reply from 40.0.0.2~~

Ping statistics for 40.0.0.2:

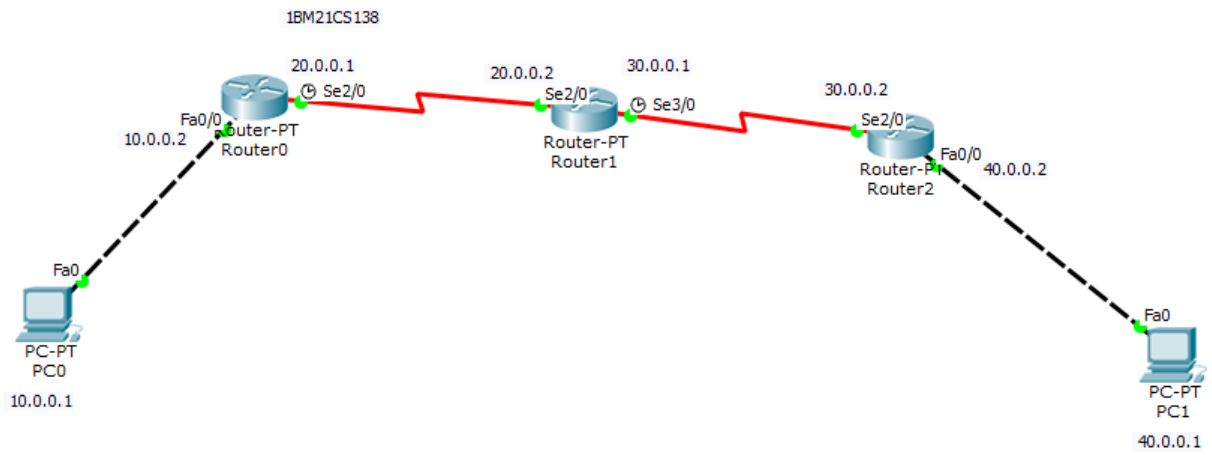
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

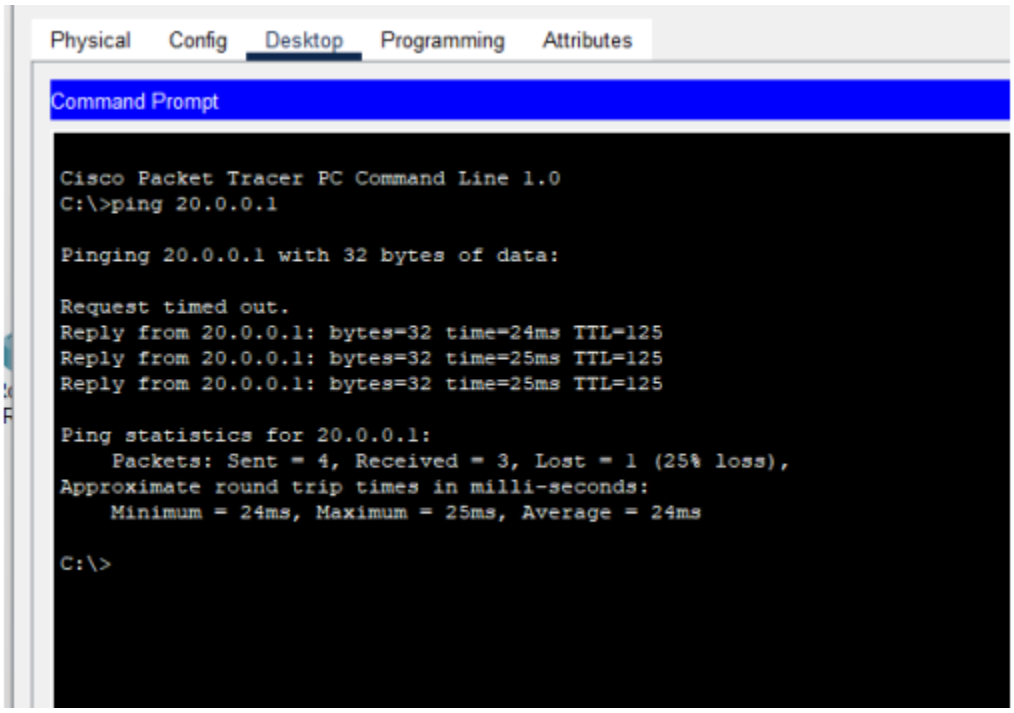
Minimum = 2ms, Maximum = 14ms, Average = 7ms

25/7

Topology:

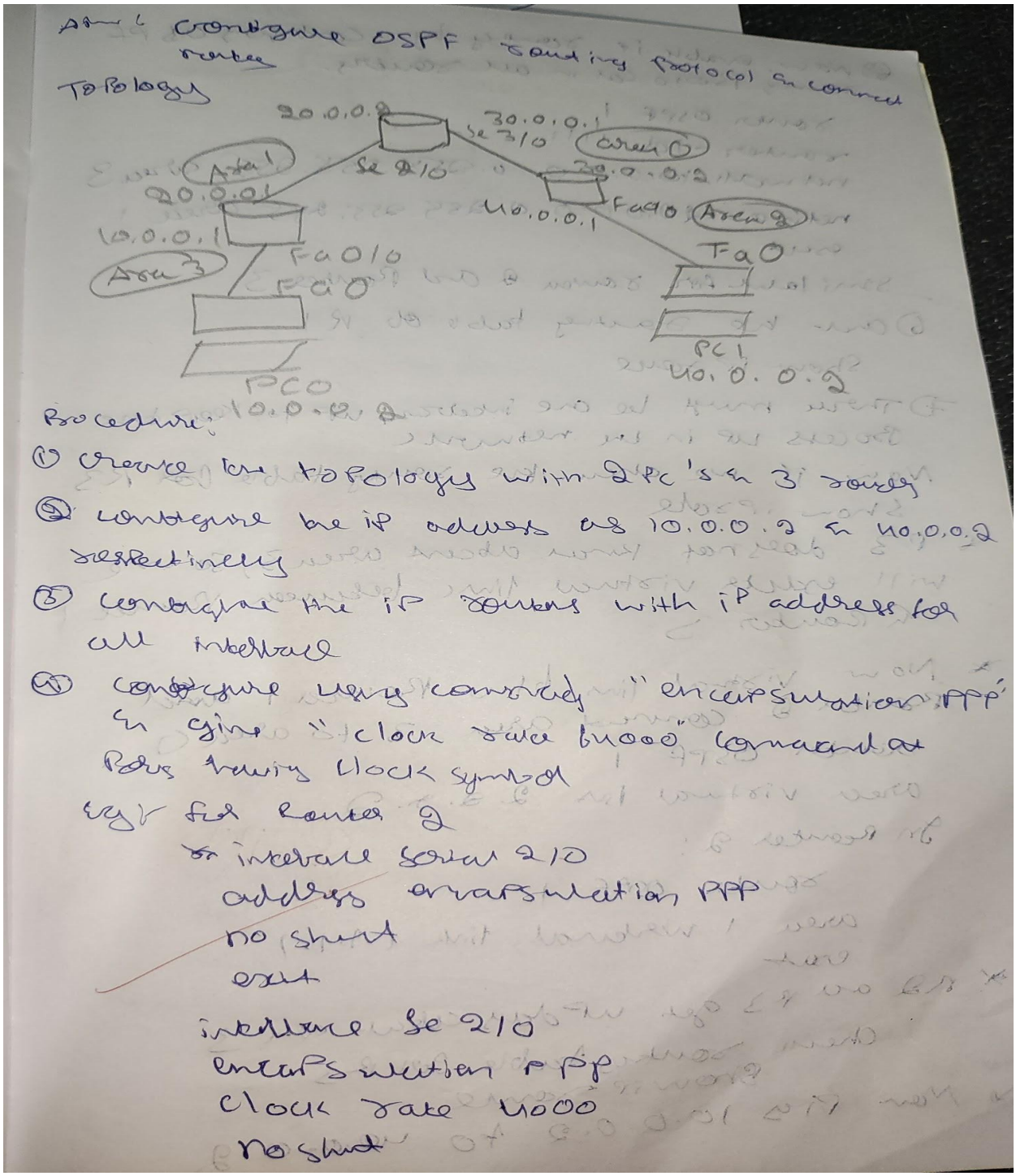


Output:



Experiment 6

Configure OSPF routing protocol



5) Now, enable ip routing by configuring OSPF routing Protocol on all routers.

router ospf 1
router-id 1111
network 10.0.0.0 0.255.255.255 area 3
network 20.0.0.0 0.255.255.255 area 1
exit

Similarly for router 2 and Router 3

6) Check the routing table of R1

Show ip route

7) There must be one interface up to keep the process up in the network

Now if we check the routing table for R3

Show ip route

* R3 doesn't know about area 3, so we will create virtual link between Router 1 & Router 3

* Now virtual link btw Router 1 and Router 2 connect area 3 to area 0

router OSPF 1
area virtual link 2.2.2.2

In Router 2:

router OSPF 1
area 1 virtual link 1.1.1.1
exit

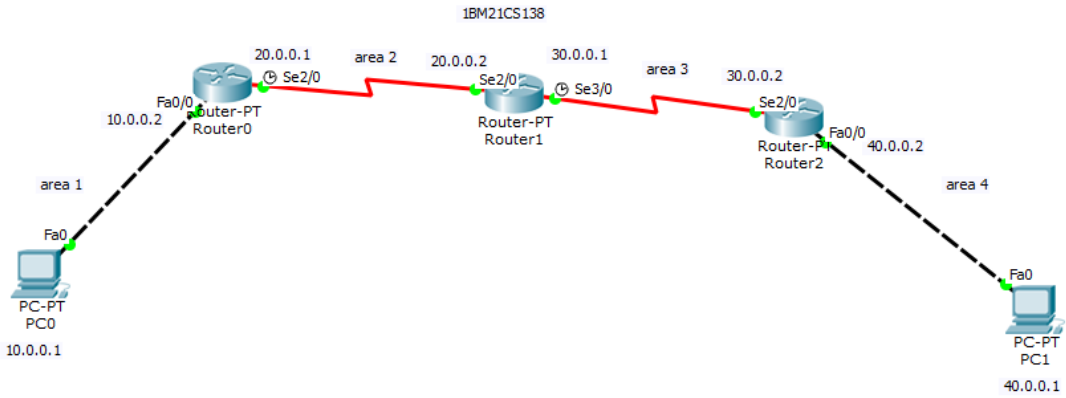
* R2 and R3 get updates about all 3

Check routing table for R3

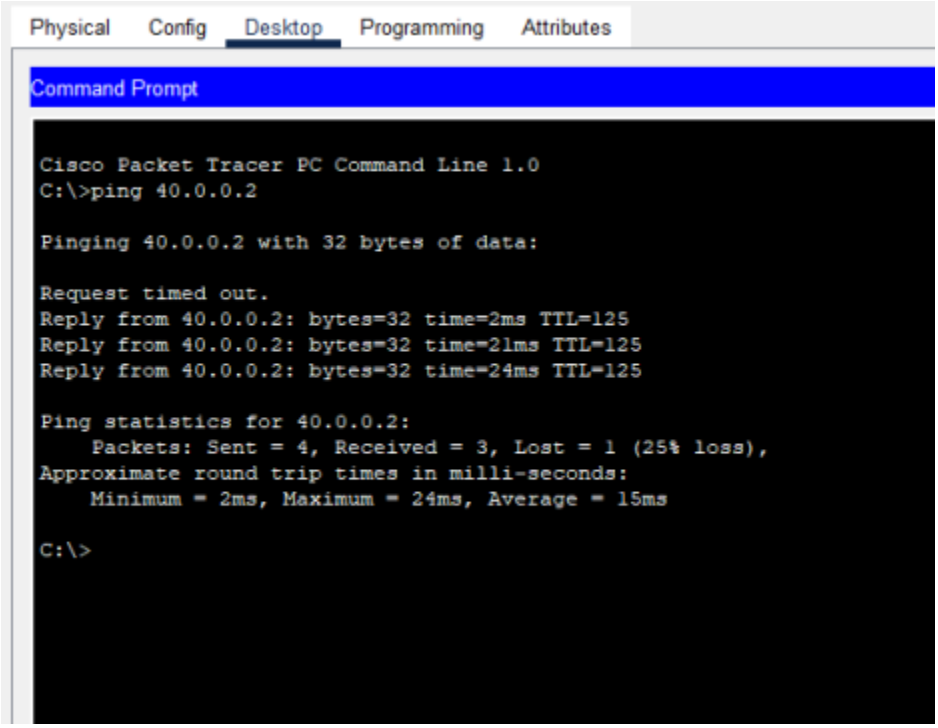
Show ip route

* Now Ping 10.0.0.2 to 40.0.0.2

Topology:

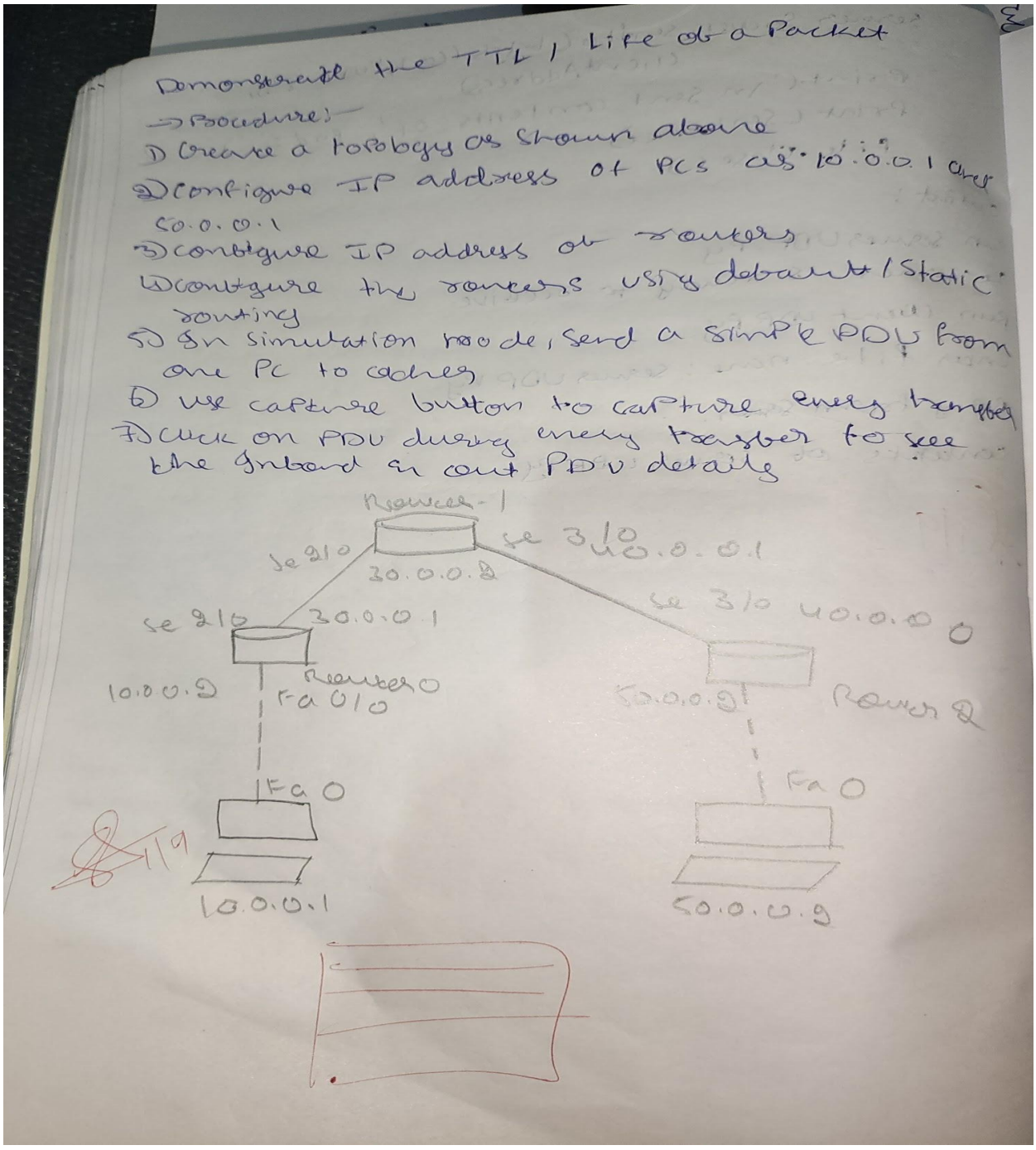


Output:

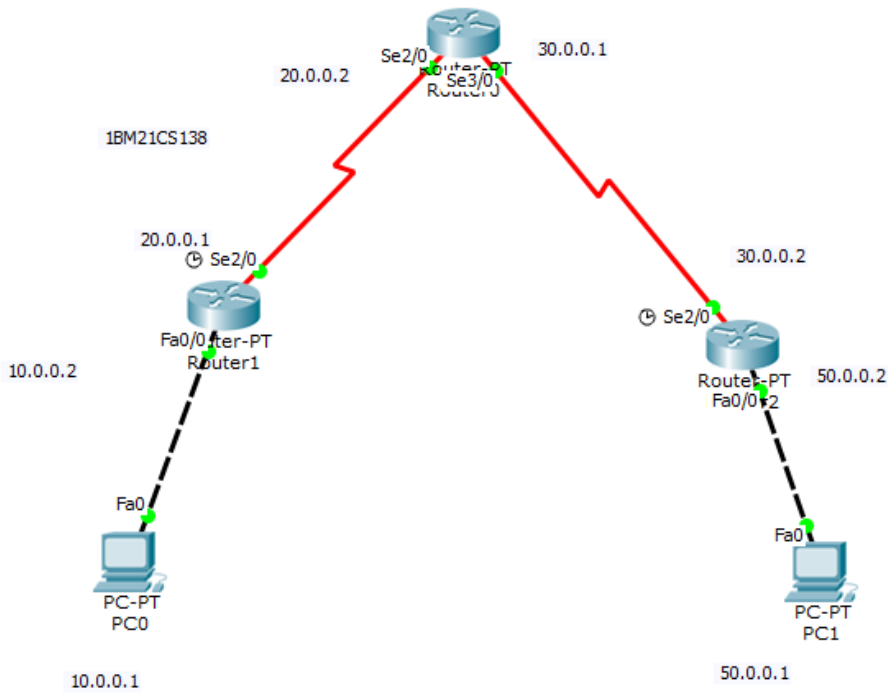


Experiment 7

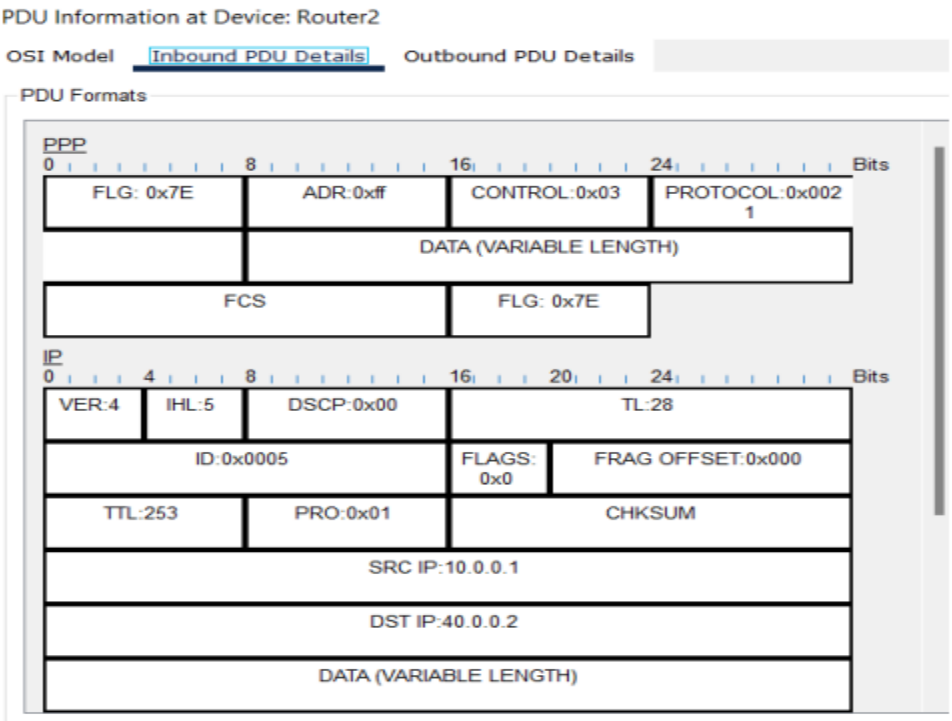
Demonstrate the TTL/ Life of a Packet



Topology:

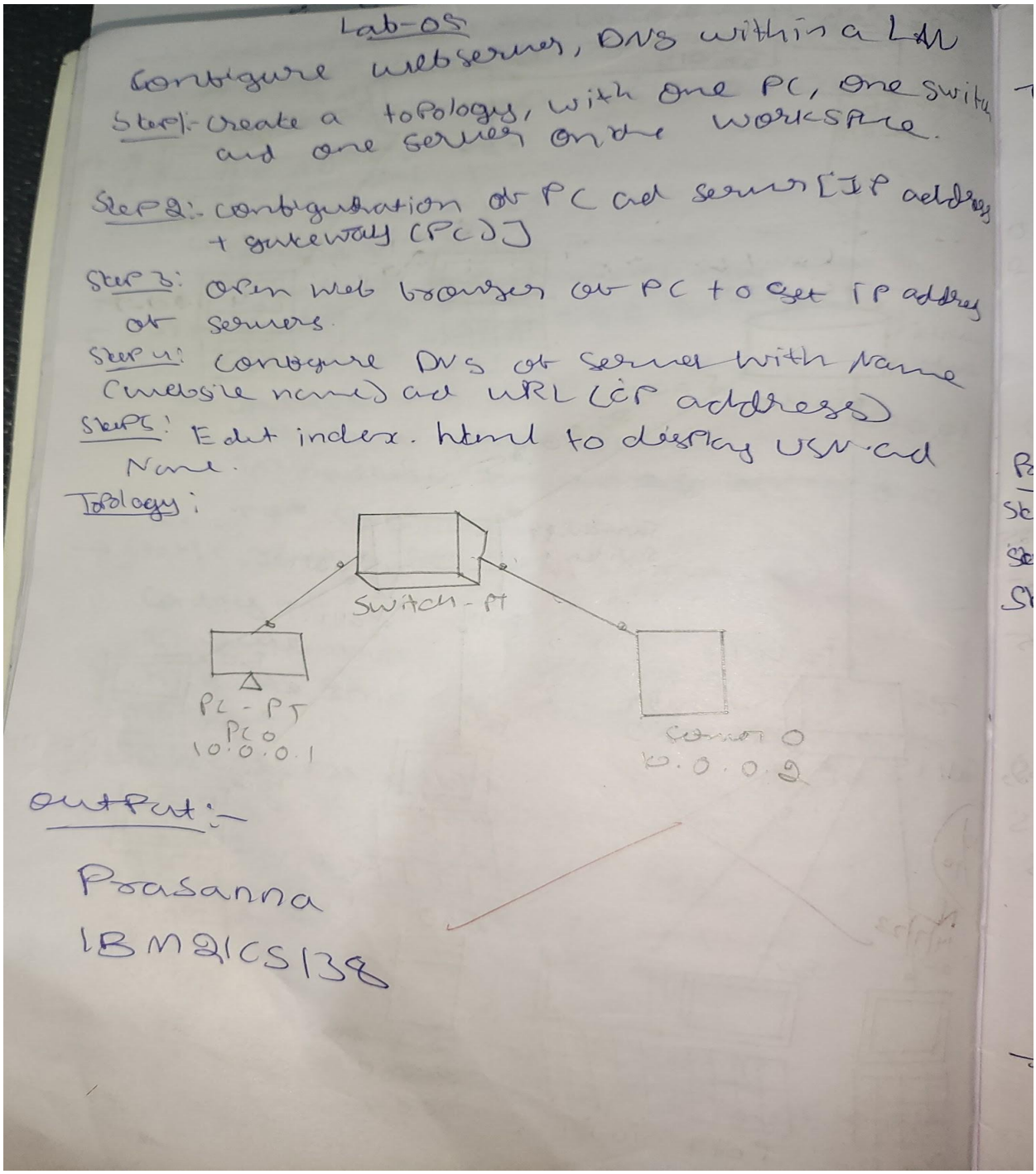


Output:

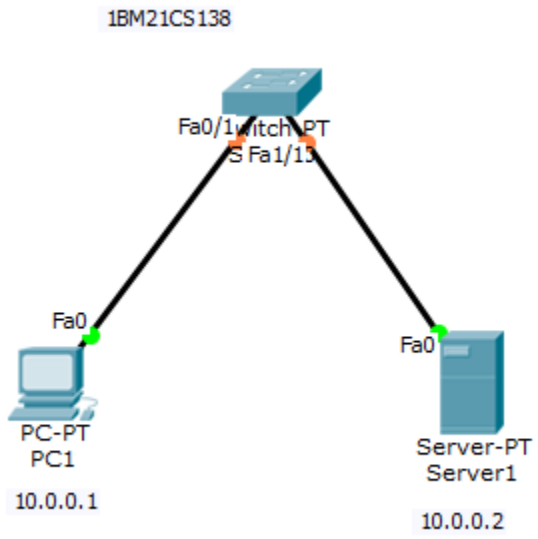


Experiment 8

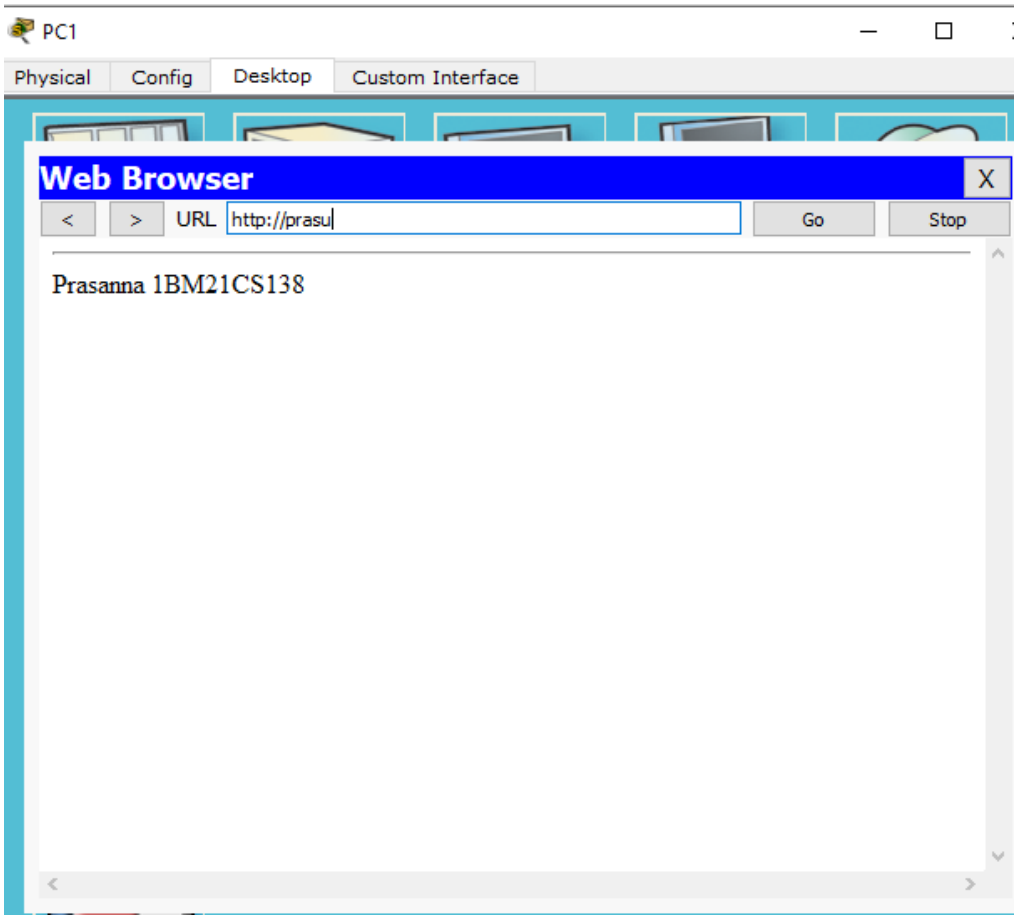
Configure Web Server, DNS within a LAN.



Topology:

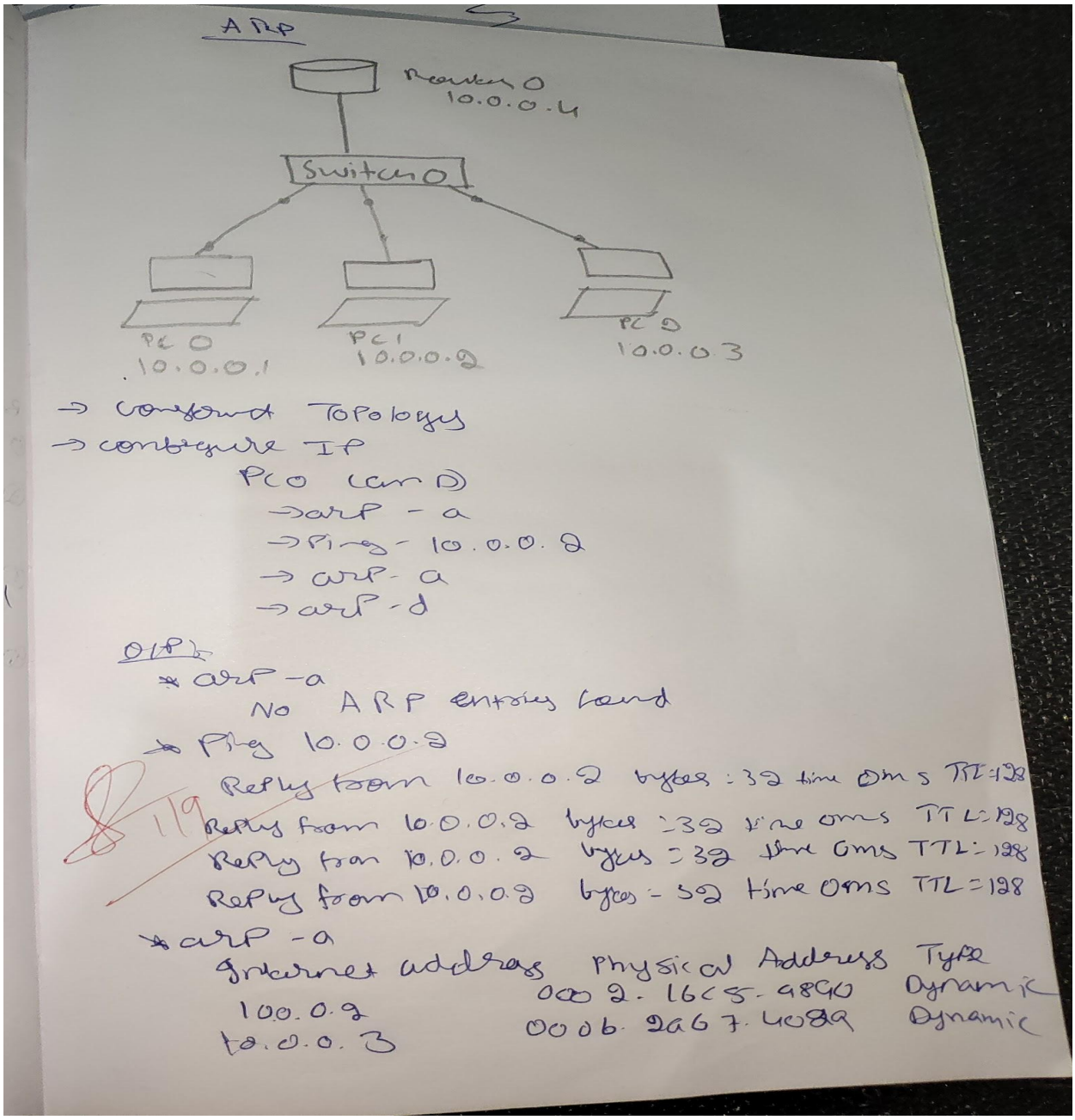


Output:

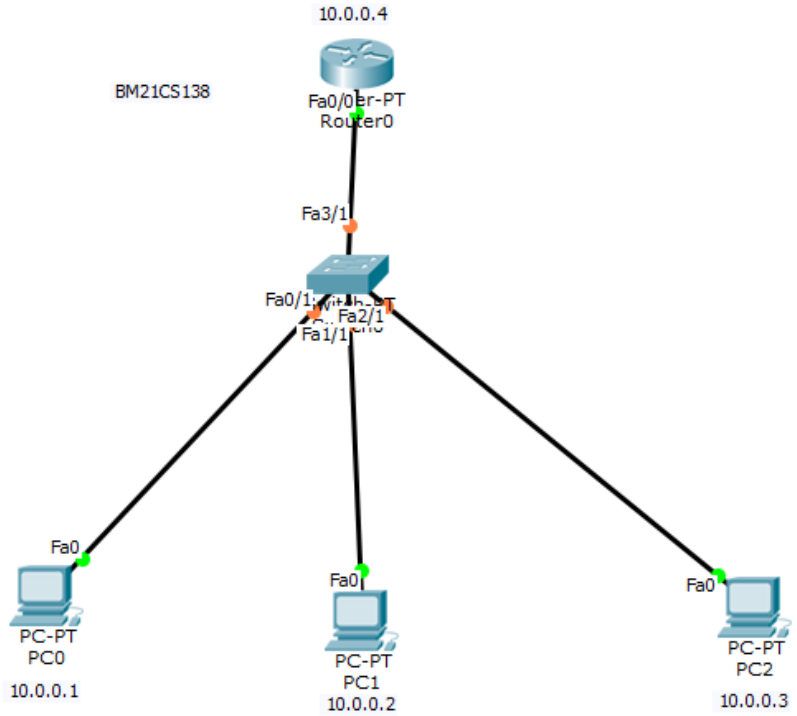


Experiment 9

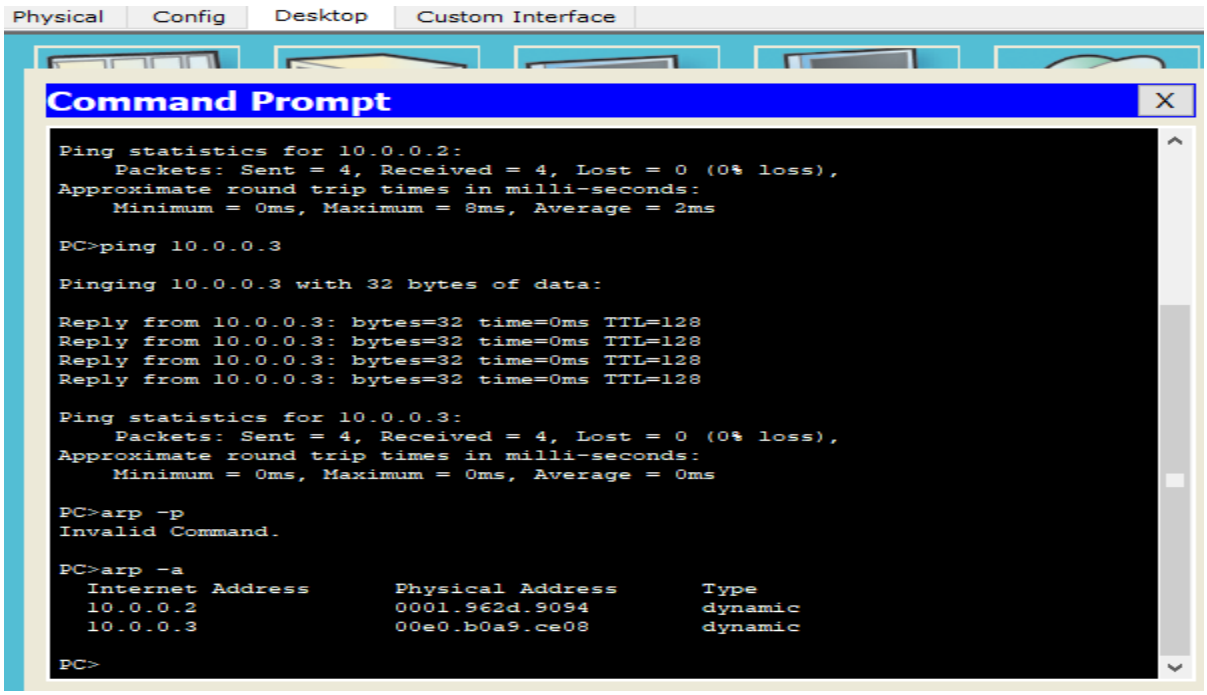
To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)



Topology:

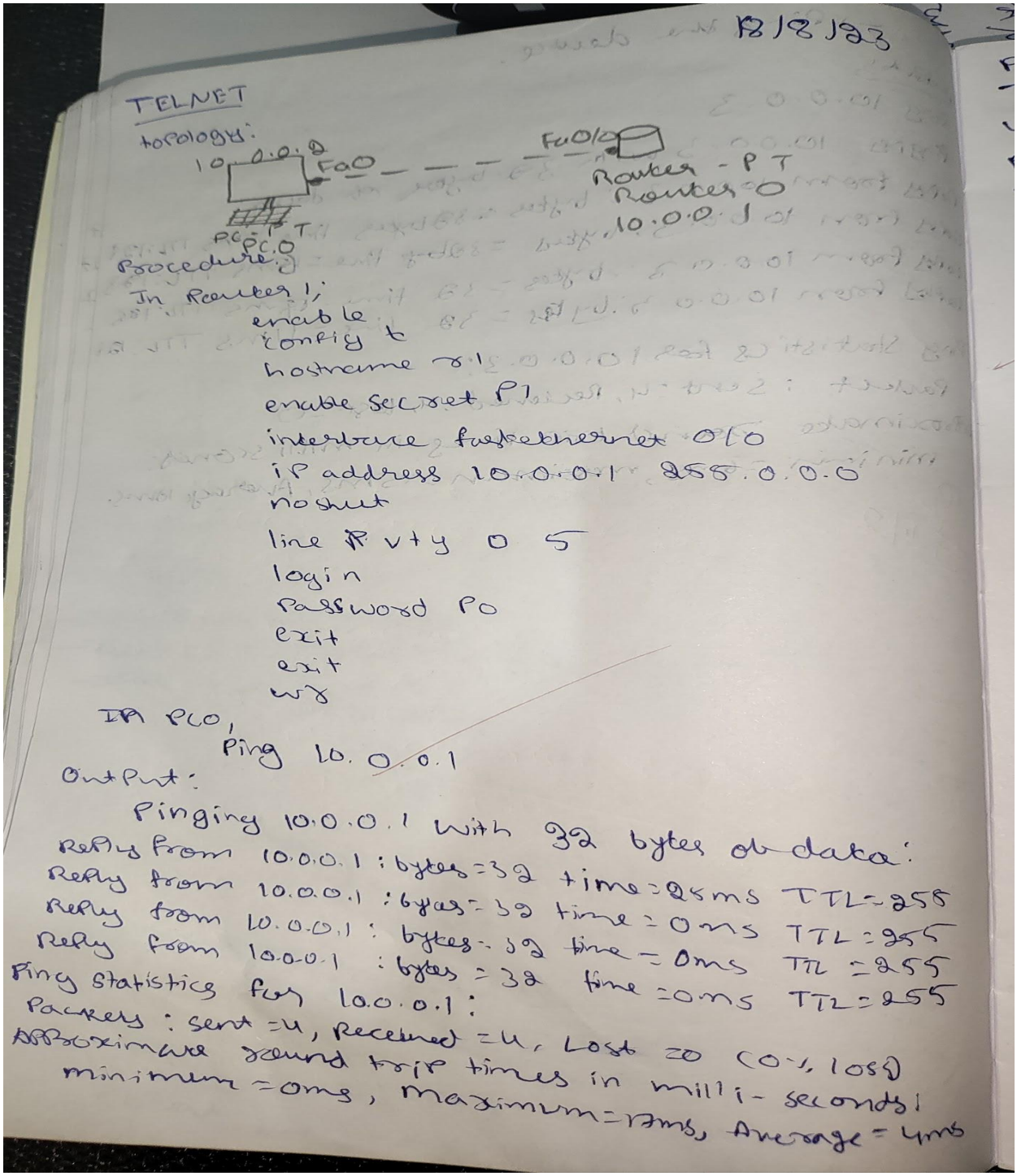


Output:



Experiment 10

To understand the operation of TELNET by accessing the router in server room from a PC in IT office.



PC > telnet 10.0.0.1
Trying 10.0.0.1 ... Open

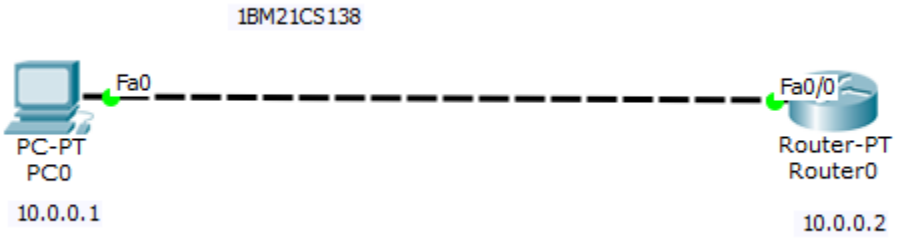
User Access Verification

Password : PO
R1 > enable
Password : P1
R1 #show ip route

10.0.0.0/8 is directly connected, FastEthernet 0/0

18/8

Topology:



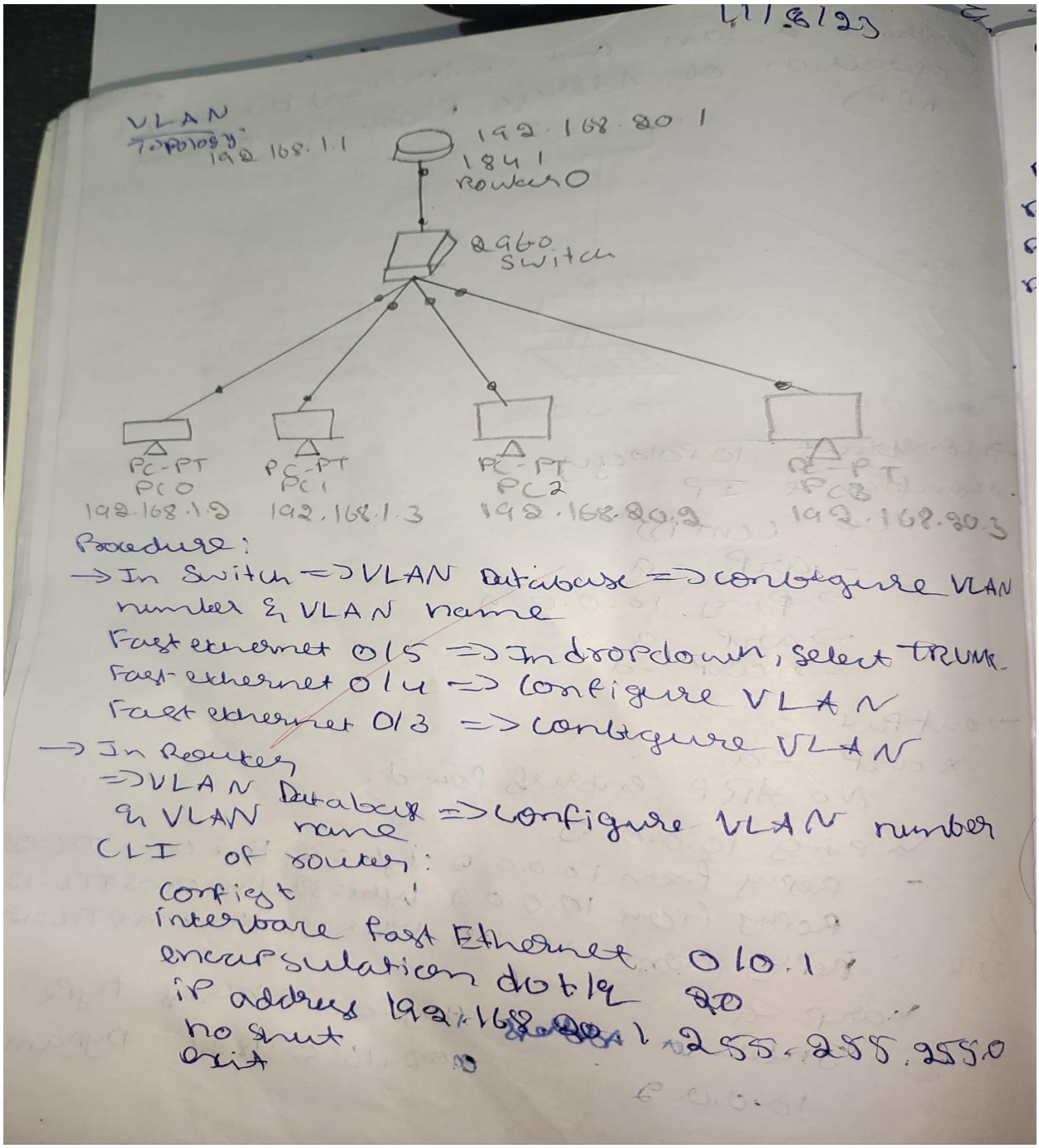
Output:

Command Prompt

```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.2

Pinging 10.0.0.2 with 32 bytes of data:

Reply from 10.0.0.2: bytes=32 time=96ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255

Ping statistics for 10.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 96ms, Average = 24ms

PC>telnet 10.0.0.2
Trying 10.0.0.2 ...Open


User Access Verification

Password:
prasu>enable
Password:
prasu#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

C    10.0.0.0/8 is directly connected, FastEthernet0/0
prasu#
```

Experiment 11

To construct a VLAN and make the PC's communicate among a VLAN



Output:

Ping 192.168.20.3

Pinging 192.168.20.3 with 32 bytes of data:

Reply from 192.168.20.3: bytes = 32 time=0ms TTL=127

Reply from 192.168.20.3: bytes = 32 time=0ms TTL=127

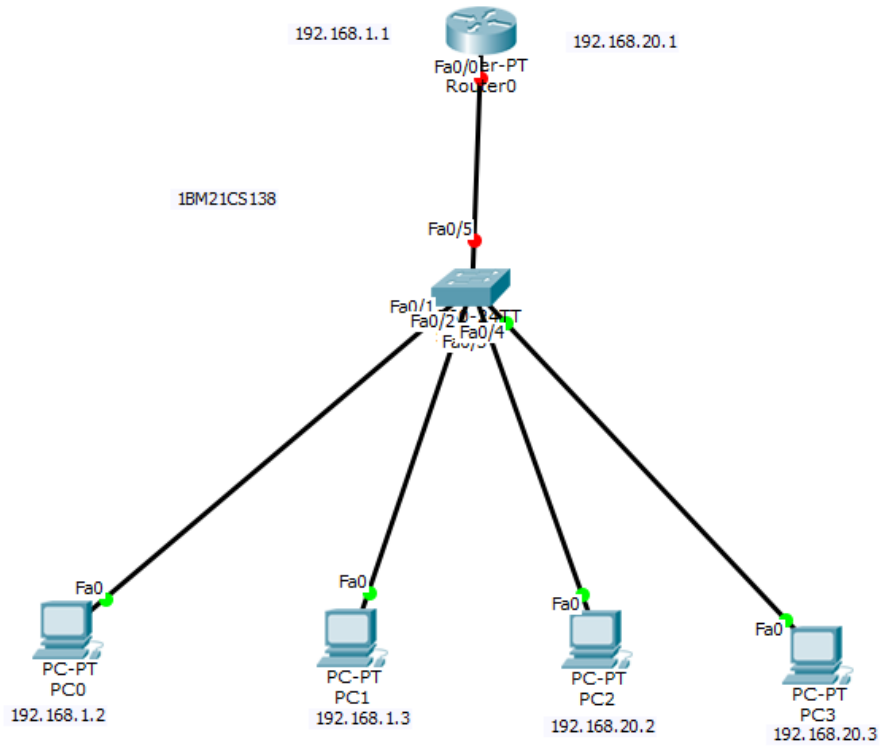
Reply from 192.168.20.3: bytes = 32 time=0ms TTL=127

Reply from 192.168.20.3: bytes = 32 time=0ms TTL=127

Ping statistics for 192.168.20.3:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)

Topology:

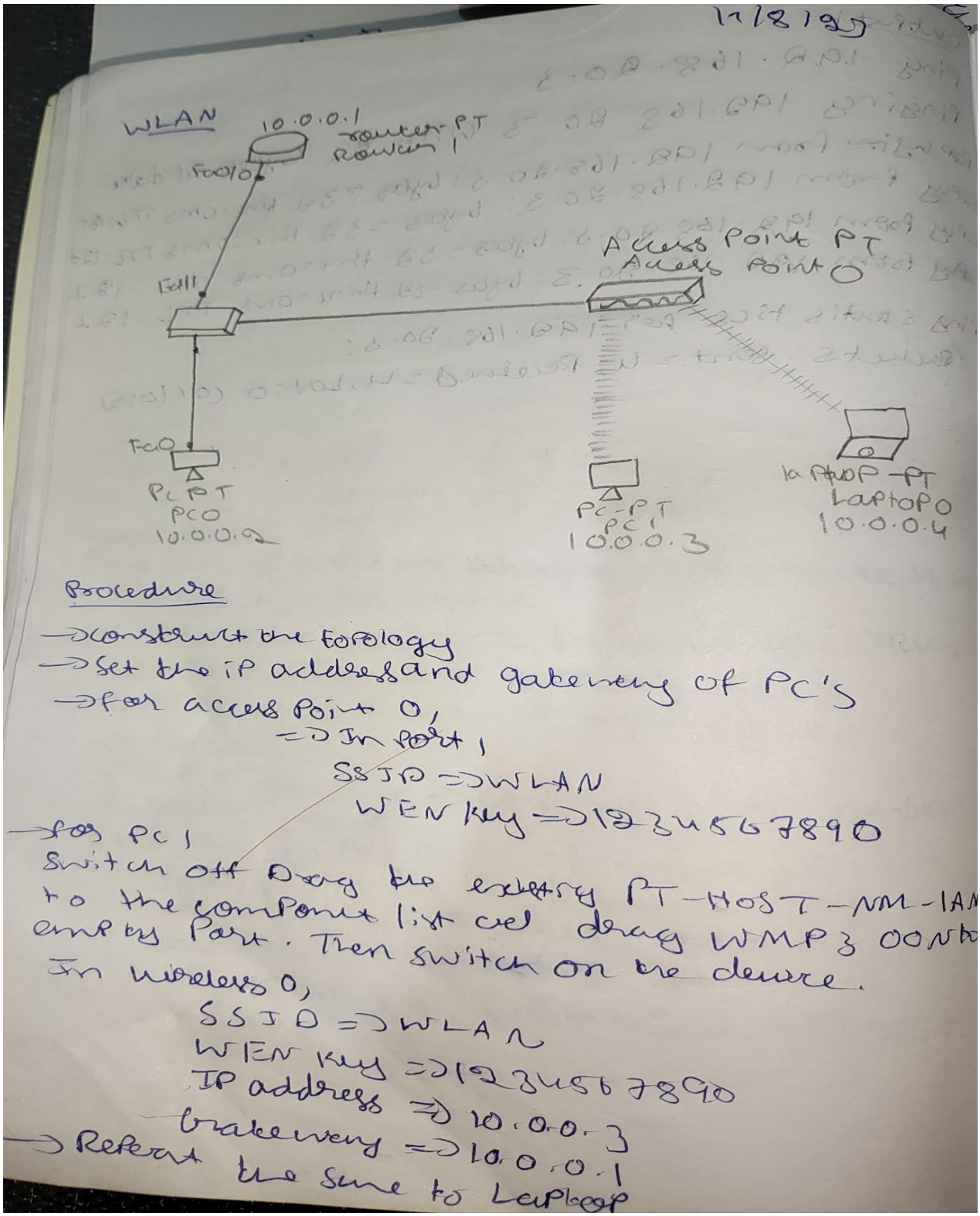


Output:

```
C:\>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

Request timed out.
Reply from 192.168.2.2: bytes=32 time<1ms TTL=127
Reply from 192.168.2.2: bytes=32 time<1ms TTL=127
Reply from 192.168.2.2: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
```

Experiment 12

To construct a WLAN and make the nodes communicate wirelessly



→ Pings the device.

Output:

Ping 10.0.0.3

Pinging 10.0.0.3 with 32 bytes of data:

Reply from 10.0.0.3 : bytes = 32 bytes time=15ms TTL=128

Reply from 10.0.0.3 : bytes = 32 bytes time=15ms TTL=128

Reply from 10.0.0.3 : bytes = 32 time=14ms TTL=128

Reply from 10.0.0.3 : bytes = 32 time=11ms TTL=128

Ping Statistics for 10.0.0.3:

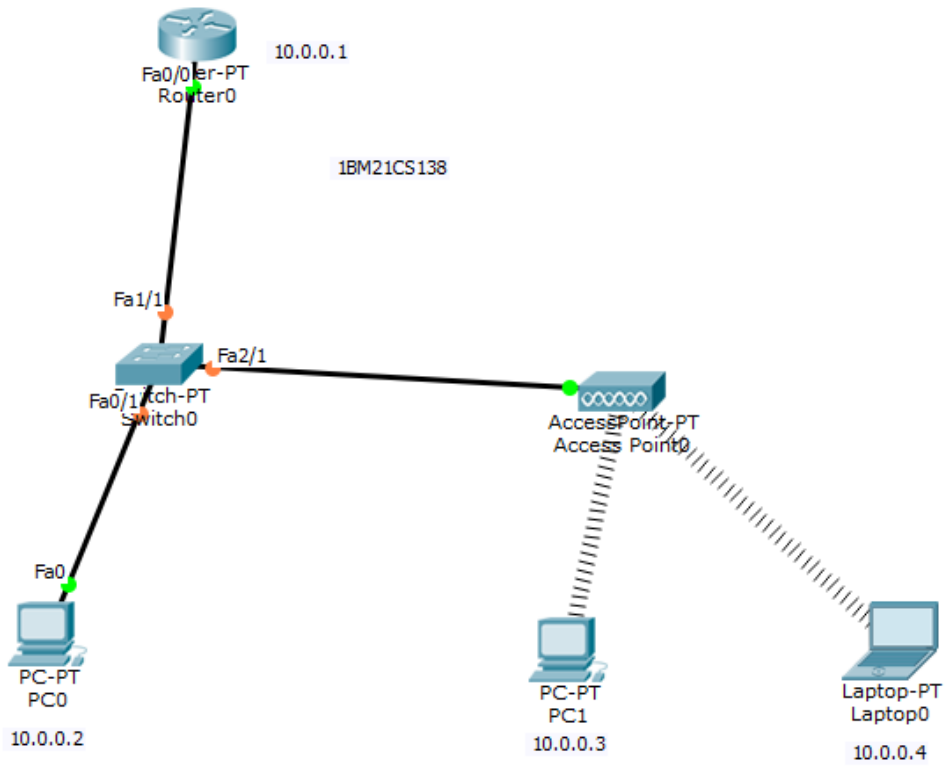
Packet : Sent = 4, Received = 4, Lost = 0

Approximate round trip times in milli seconds:

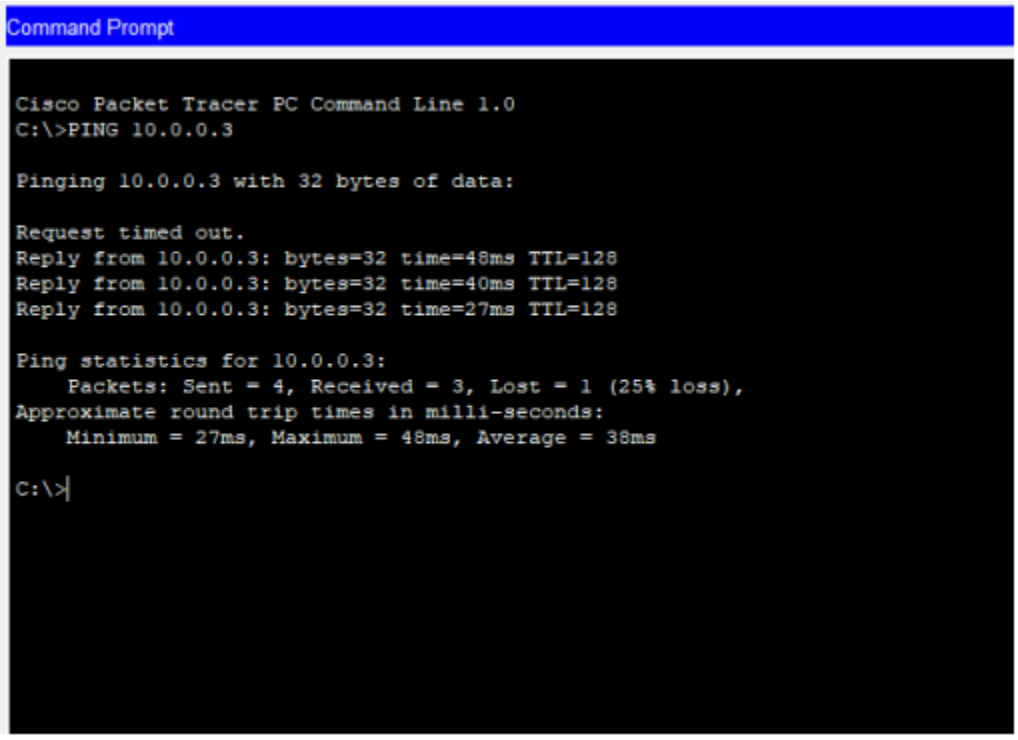
Minimum = 6ms, Maximum = 15ms, Average 10ms.

11/8

Topology:

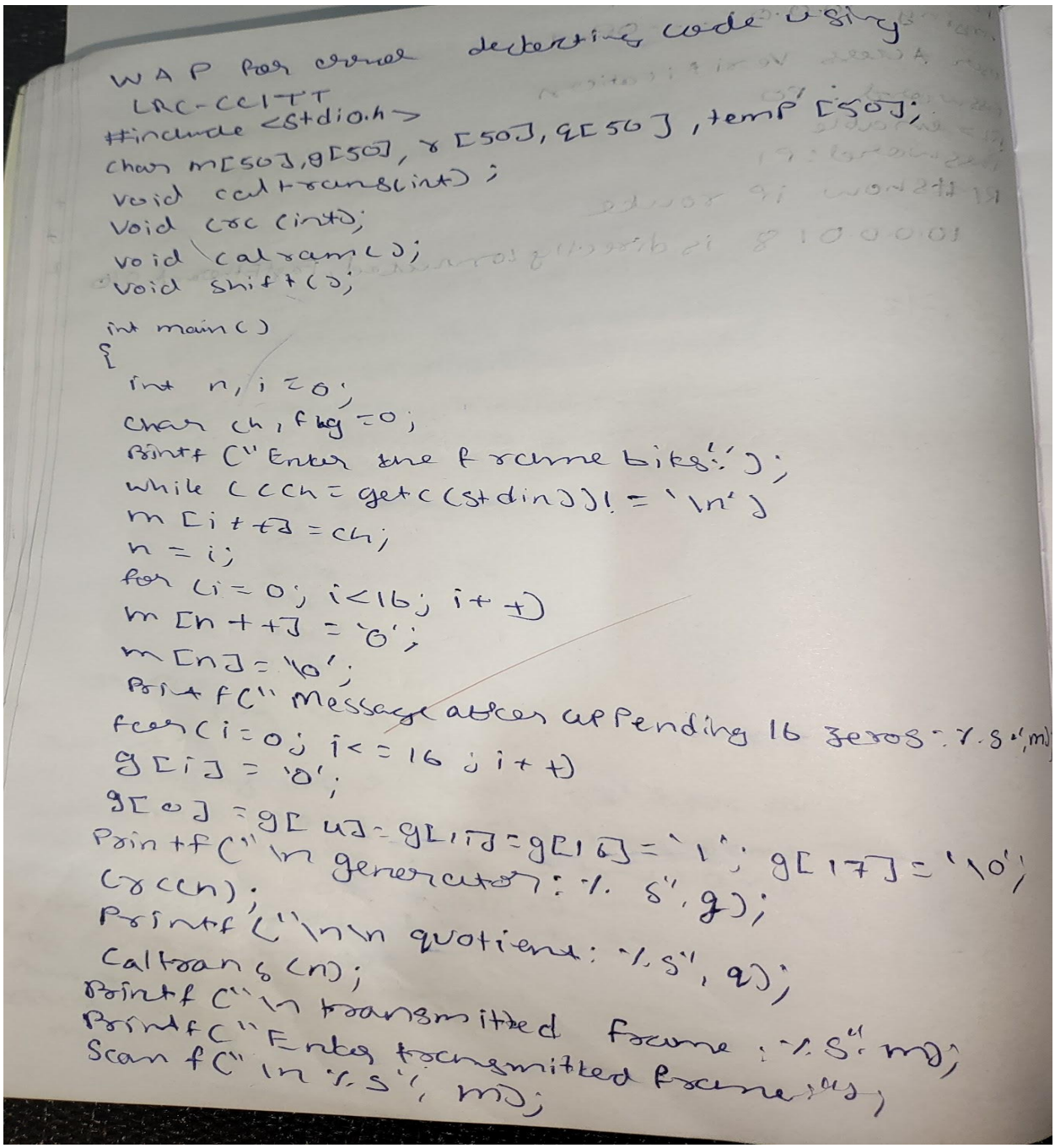


Output:

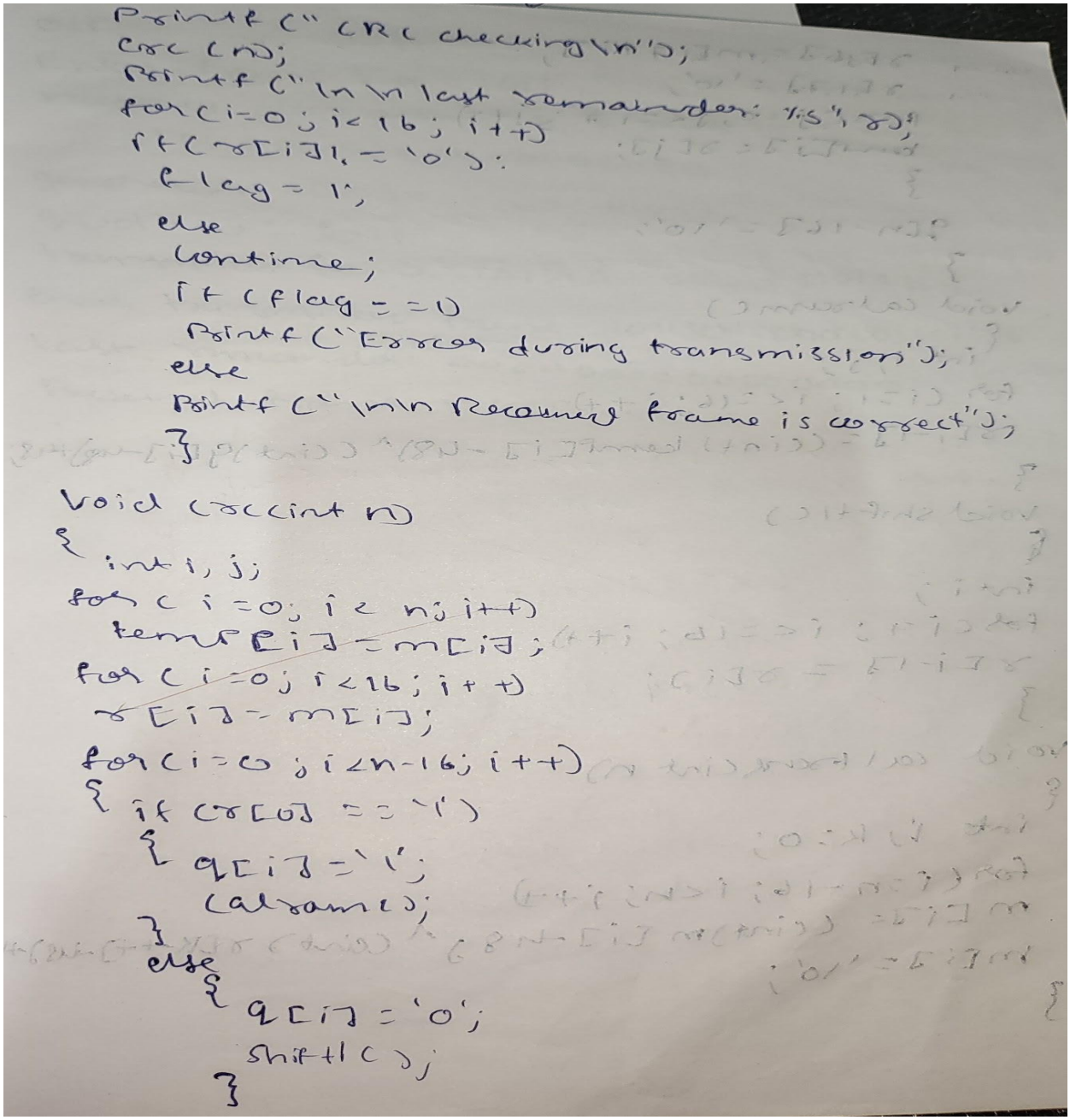


Experiment 13

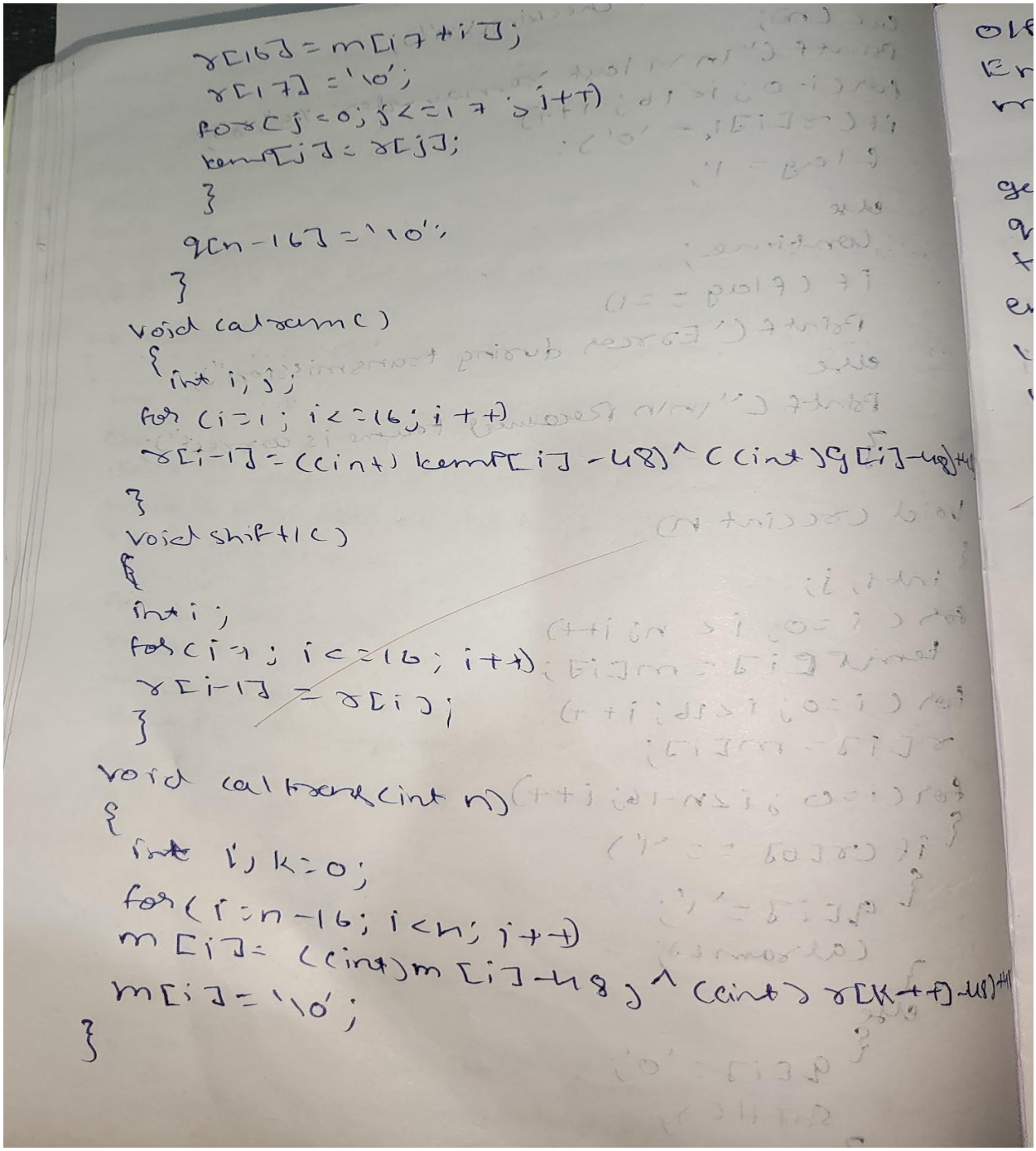
Write a program for error detecting code using CRC-CCITT (16-bits).



```
WAP for error detecting code using
CRC-CCITT
#include <stdio.h>
char m[50], g[50], r[50], q[50], temp[50];
void caltrans(int);
void crc(int);
void calram();
void shift();

int main()
{
  int n, i=0;
  char ch, flag=0;
  printf("Enter the frame bits:");
  while((ch=getc(stdin))!='\n')
  m[i++]=ch;
  n=i;
  for(i=0; i<16; i++)
  m[n++]='0';
  m[n]='\0';
  printf("Message after appending 16 zeros: %s",m);
  for(i=0; i<=16; i++)
  g[i]='0';
  g[0]=g[4]=g[11]=g[16]='1'; g[17]='\0';
  printf("\n generator: %s", g);
  crc(n);
  printf("\n\n quotient: %s", q);
  caltrans(n);
  printf("\n transmitted frame: %s", m);
  printf("Enter transmitted frame:");
  scanf("\n%s", m);
```



```
Printf(" CRC checking\n");
crc(n);
Printf("\n\n last remainder: %s", r);
for(i=0; i<16; i++)
if(r[i]!='0'):
  flag = 1;
  else
   continue;
   if(flag==1)
    Printf("Error during transmission");
   else
   Printf("\n\n Received frame is correct");
   }

void crc(int n)
{
  int i, j;
for(i=0; i<n; i++)
  temp[i] = m[i];
for(i=0; i<16; i++)
 r[i] = m[i];
for(i=0; i<n-16; i++)
{ if(r[0]=='1')
  { q[i]='1';
    calram();
  }
  else
    { q[i]='0';
      shift1();
    }
```



```
    r[16] = m[i7+i];
    r[17] = '\0';
    for(j=0; j<=17; j++)
    temp[j] = r[j];
    }
   q[n-16] = '\0';
  }
void calram()
 {int i, j;
 for(i=1; i<=16; i++)
 r[i-1] = ((int) temp[i] - 48) ^ ((int) q[i] - 48)
}
void shift1()
{
int i;
for(i=1; i<=16; i++)
r[i-1] = r[i];
}
void calkey(int n)
{
 int i, k=0;
 for(i=n-16; i<n; i++)
 m[i] = ((int)m[i] - 48) ^ ((int) r[k++] - 48)
 m[i] = '\0';
}
```

O/P:-

Enter frame bits : 1011

message after appending 16 zeros:

1011000 000 0000000000

generator: 10001000000100001

quotient: 1011

transmitted frame: 10111011000101110101 1

Error transmitted frame: 10111011000101110101 1

last remainder: 0000000000000000

Received frame is correct.

Write the algorithm & explain it

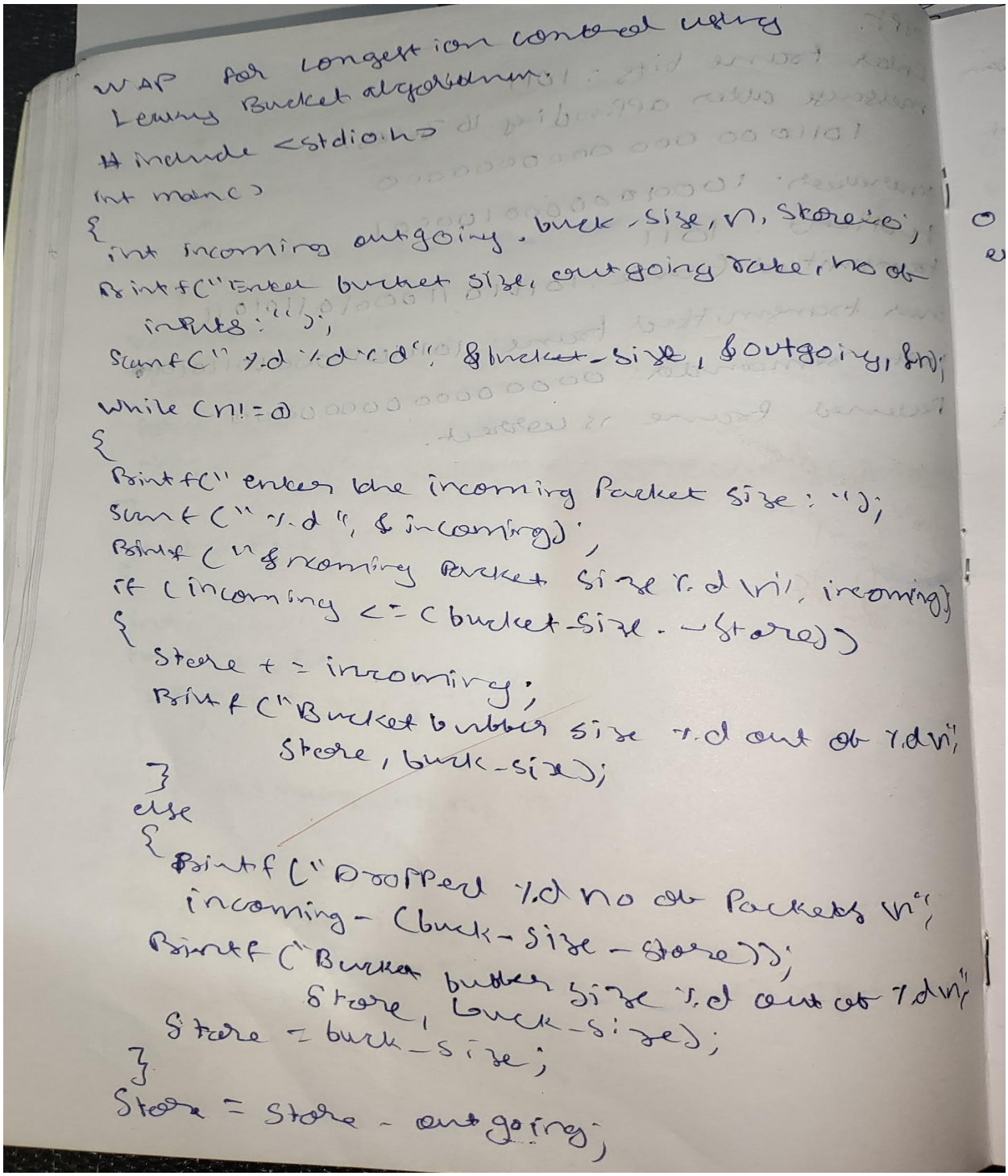
8/8

Output:

```
Enter the frame bits:1011
Message after appending 16 zeros:10110000000000000000
generator:10001000000100001


quotient:1011
transmitted frame:10111011000101101011
Enter transmitted freme:10111011000101101011
CRC checking


last remainder:0000000000000000

Received freme is correct
```

Experiment 14

Write a program for congestion control using Leaky bucket algorithm.



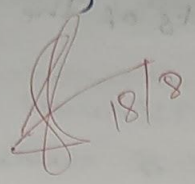
```
WAP for congestion control using
Leaky Bucket algorithm
#include <stdio.h>
int main()
{
int incoming, outgoing, buck_size, n, store=0;
Printf("Enter bucket size, outgoing rate, no of
  inputs:");
Scanf("%d %d %d", &bucket_size, &outgoing, &n);
while(n!=0)
{
Printf("enter the incoming Packet size:");
Scanf("%d", &incoming);
Printf("\nIncoming Packet size %d\n", incoming);
if(incoming <= (bucket_size - store))
{
  store += incoming;
  Printf("Bucket buffer size %d out of %d\n",
        store, buck_size);
}
else
{
  Printf("Dropped %d no of Packets\n",
   incoming - (buck_size - store));
  Printf("Bucket buffer size %d out of %d\n",
        store, buck_size);
  store = buck_size;
}
Store = Store - outgoing;
```

Printf("After outgoing %d Packets left out of %d in bubber\n", store, bucket_size);

n--;

}

}



O/P

Enter bucketSize, outgoing rate and no of inputs: 20 10 2

Enter the incoming Packet size: 30

Incoming Packet size: 30

Dropped 10 no of Packets

Bucket bubber size 0 out of 20

After outgoing 10 Packets left out of 20 in bubber

Enter the incoming Packet size: 10

Incoming Packet size 10

Bucket bubber size 20 out of 20

After outgoing 10 Packet left out of 20 in bubber

This is leaky bucket algo used to regulate network traffic. It takes input for bucket size, outgoing rate, no. of input for each Packet.

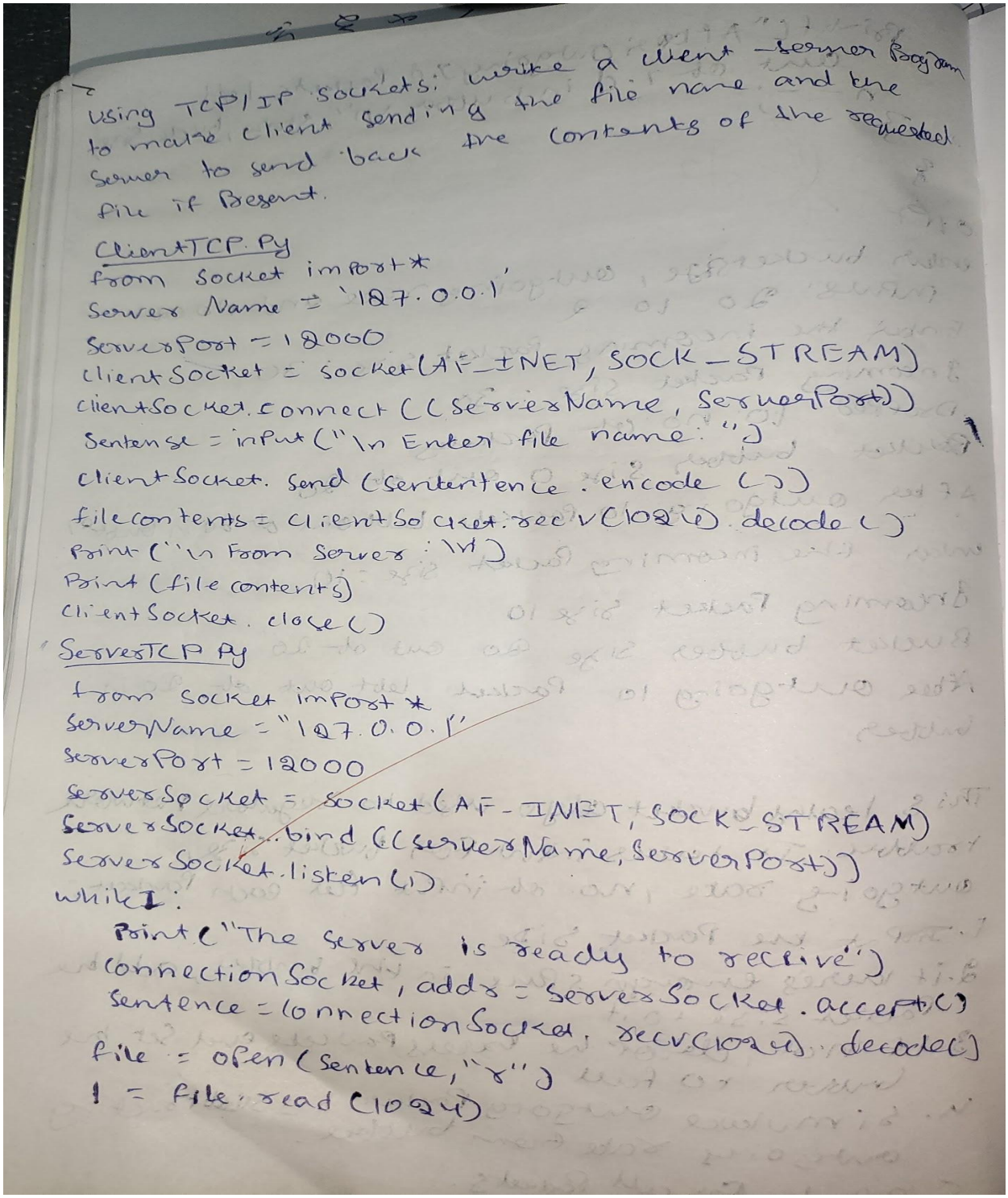
1. Input the Packet size.
2. if there is enough space in the bubber, add the Packet size to it.
3. if not, drop the excess Packets and set the bubber to full.
4. Simulate outgoing Packets by subtracting outgoing rate from bubber.
5. Repeat for all Packets.

Output:

```
Enter bucket size, outgoing rate and no of inputs: 10 10 2
Enter the incoming packet size : 30
Incoming packet size 30
Dropped 20 no of packets
Bucket buffer size 0 out of 10
After outgoing 0 packets left out of 10 in buffer
Enter the incoming packet size : 10
Incoming packet size 10
Bucket buffer size 10 out of 10
After outgoing 0 packets left out of 10 in buffer
```

Experiment 15

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



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.

ClientTCP.Py

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

ServerTCP.Py

```
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('In Sent contents of ' + sentence)
file.close()
connectionSocket.close()
```

Output:

Run the serverTCP.py

The server is ready to receive

Run the clientTCP.py

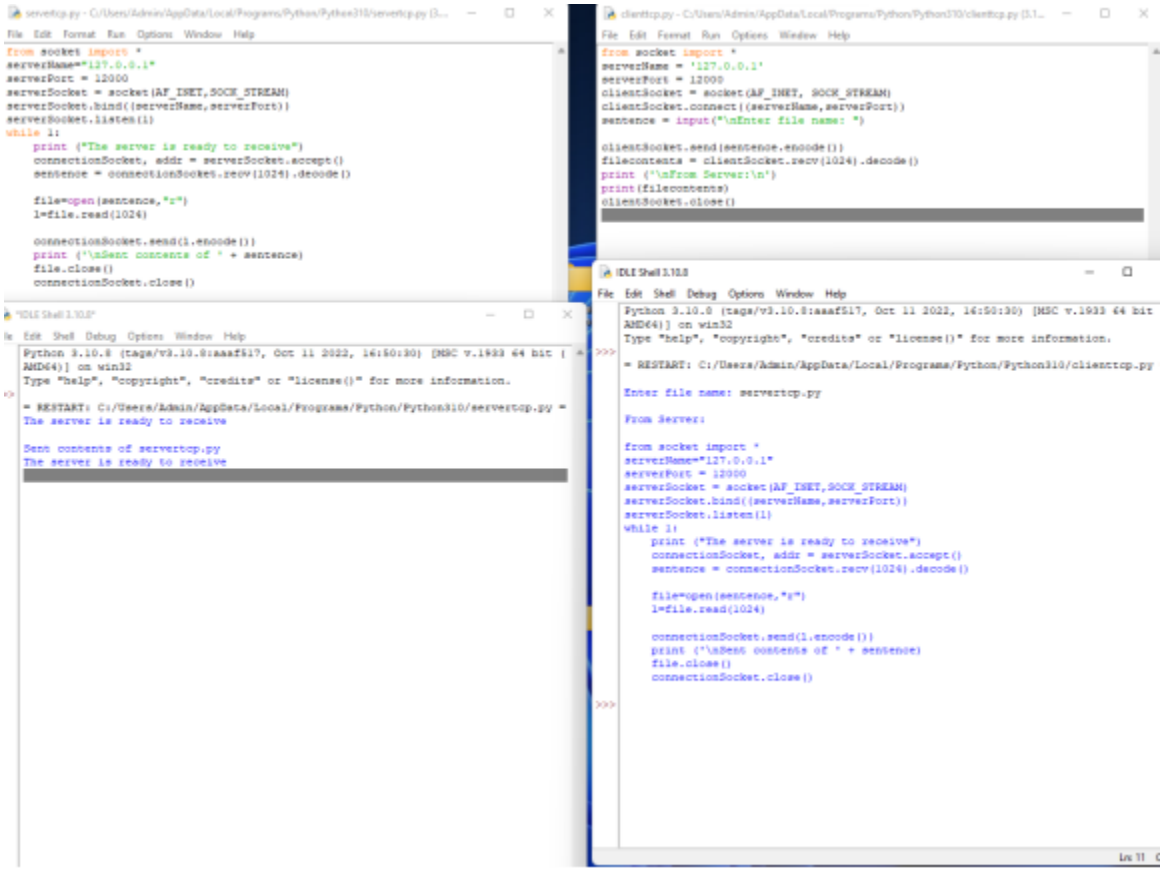
Enter file name: serverTCP.py

From server:

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK
```

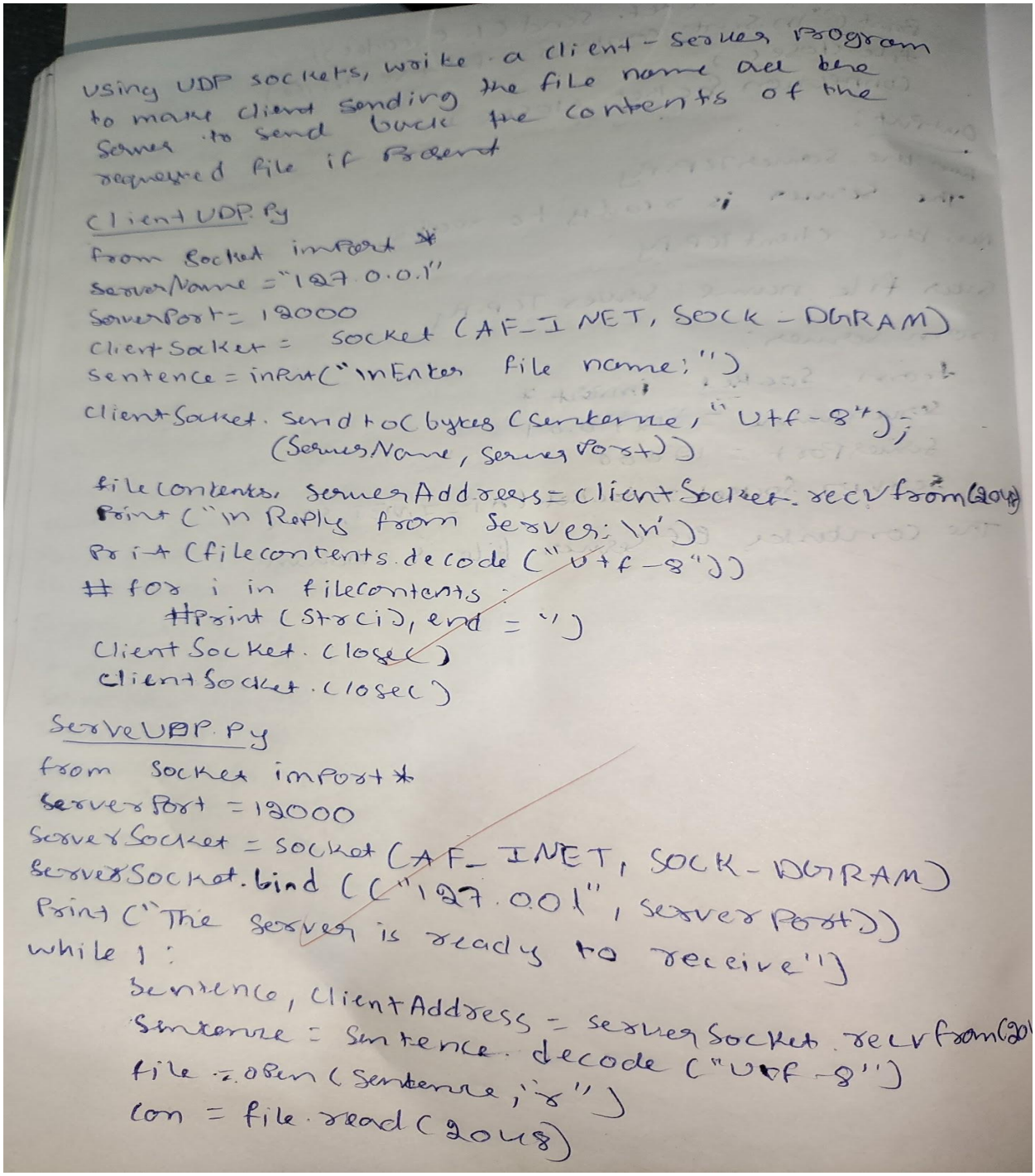
The contents of server file is shown.

Output:


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()
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()
Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2022, 16:50:30) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
= RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python310/servertcp.py =
The server is ready to receive
Sent contents of servertcp.py
The server is ready to receive
Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2022, 16:50:30) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
= RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python310/clienttcp.py
Enter file name: servertcp.py
From 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()

Experiment 16

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



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

ClientUDP.py

```
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("\n Reply from Server:\n")
print(fileContents.decode("utf-8"))
# for i in fileContents:
    #print(str(i), end = "")
clientSocket.close()
clientSocket.close()
```

ServeUDP.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(20
    sentence = sentence.decode("utf-8")
    file = open(sentence, "r")
    con = file.read(2048)
```

```
server.socket.sendto(bytes(con,"utf-8"),
                     clientAddress)
print('\n Sent contents of', end=' ')
print(sentance)
file.close()
```

Output :-

Run serverUDP.py

The server is ready to receive

Run clientUDP.py

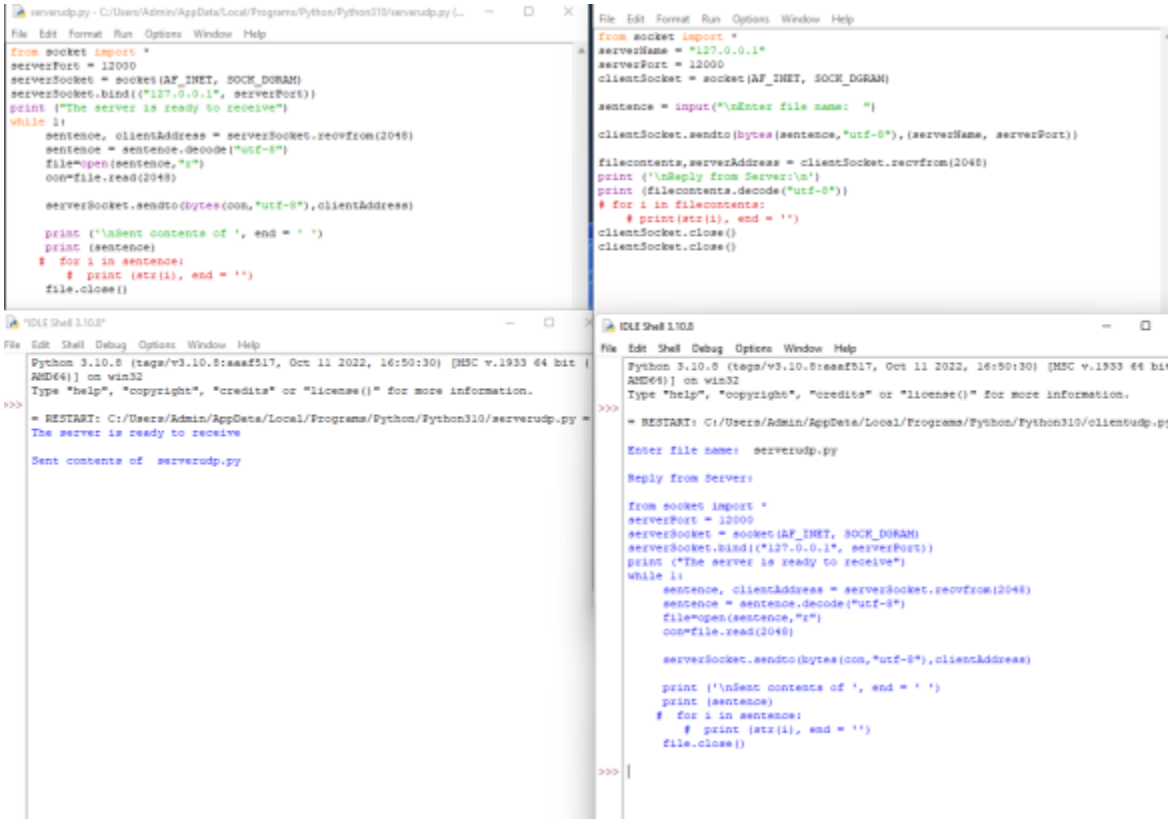
Enter file name: serverUDP.py

Reply from server:

Contents of serverUDP.py

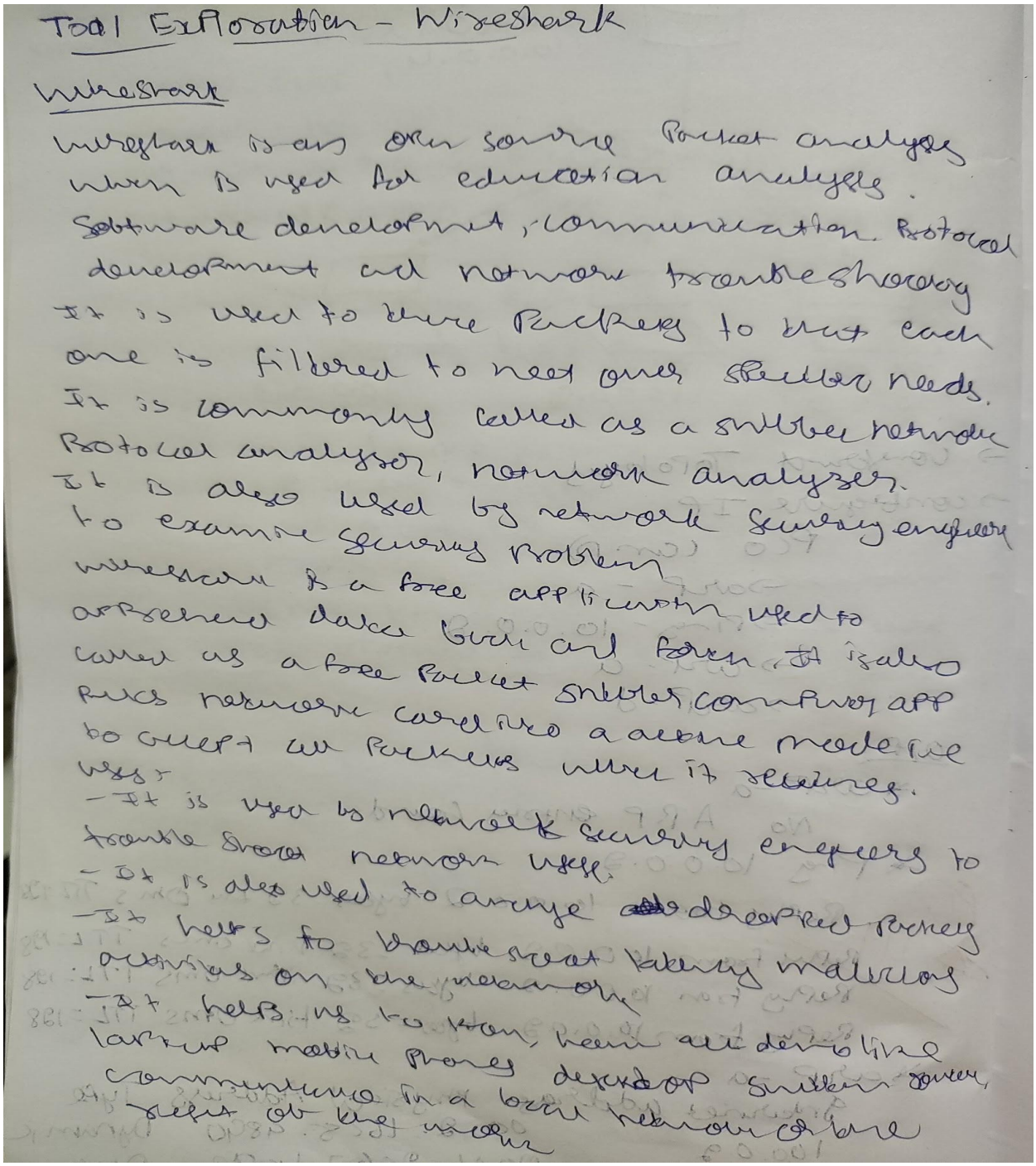
11/9

Output:


serverudp.py - C:/Users/Admin/AppData/Local/Programs/Python/Python310/serverudp.py (...
File Edit Format Run Options Window Help
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")
con=file.read(2048)
serverSocket.sendto(bytes(con,"utf-8"),clientAddress)
print ("\nSent contents of ", end = ' ')
print (sentence)
# for i in sentence:
# print (str(i), end = '')
file.close()
File Edit Format Run Options Window Help
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()
IDLE Shell 3.10.8
File Edit Shell Debug Options Window Help
Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2022, 16:50:30) [MSC v.1933 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python310/serverudp.py =
The server is ready to receive
Sent contents of serverudp.py
IDLE Shell 3.10.8
File Edit Shell Debug Options Window Help
Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2022, 16:50:30) [MSC v.1933 64 bit
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python310/clientudp.py
Enter file name: serverudp.py
Reply from 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")
con=file.read(2048)
serverSocket.sendto(bytes(con,"utf-8"),clientAddress)
print ("\nSent contents of ", end = ' ')
print (sentence)
# for i in sentence:
# print (str(i), end = '')
file.close()
>>>

Experiment 17

Tool Exploration -Wireshark



Tool Exploration - Wireshark

Wireshark

Wireshark is an open source Packet analyzer which is used for education analysis. Software development, communication Protocol development and network troubleshooting It is used to store Packets to that each one is filtered to meet our specific needs. It is commonly called as a sniffer network Protocol analyser, network analyzer. It is also used by network security engineers to examine security Problems

Wireshark is a free app it is widely used to interpret data both and form. It is also called as a free Packet sniffer computer app Puts network card into a promiscuous mode to capture all Packets which it receives. Uses:

- It is used by network security engineers to trouble shoot network issues.
- It is also used to analyze dropped Packets
- It helps to troubleshoot latency malicious activities on the network.
- It helps us to know, how all devices like laptop mobile phones desktop switches routers communicate in a local network or how packets on the move

Functionality of wireshark:

It is similar to a TCP Dump in networking
It is a graphic end user and filtering tool
It is also monitors the network traffic with
its NIC sent to network's MAC address interface.
The Port mirroring is a method to monitor the
network traffic. When it is enabled switch
sends copies of all network Packets present
at one Port to another Port.

Features of wireshark:

→ It is a multi Platform software i.e. it can
run on the Linux, windows, OSX, Free BSD,
NetBSD etc.
→ It is a standard three pane Packet browser.
→ It performs deep inspection of hundreds of Protocols
→ It can read standard filtering option which
makes easy to user to view the data
→ It can capture data USB traffic
→ It is used in IP analysis