

# **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**



## **LAB RECORD**

### **Computer Network Lab (23CS5PCCON)**

*Submitted by*

**Eashan Jain V (1BM23CS098)**

*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  
Academic Year 2024-25 (odd)**

# 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 Network (23CS5PCCON)” carried out by **Eashan Jain V(1BM23CS098)**, who is bonafide student of **B.M.S. College of Engineering**. It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements of the above-mentioned subject and the work prescribed for the said degree.

Sarala D V Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE
---	--

## Index

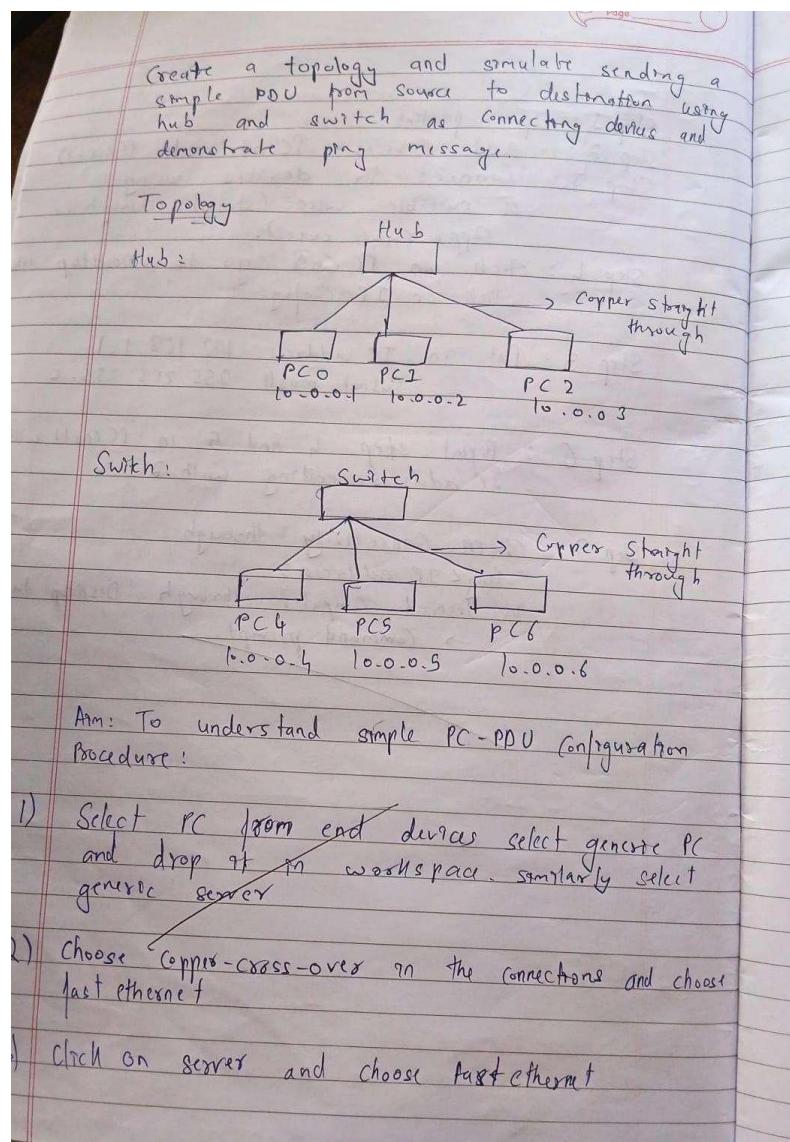
<b>Sl. No.</b>	<b>Date</b>	<b>Experiment Title</b>	<b>Page No.</b>
1	19/08/2025	Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.	1
2	03/09/2025	Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply	4
3	10/09/2025	Configure default route, static route to the Router	8
4	9/09/2025	Configure DHCP within a LAN and outside LAN.	11
5	23/10/2025	Configure RIP routing Protocol in Routers	14
6	14/10/2025	Configure OSPF routing protocol	17
7	11/11/2025	Demonstrate the TTL/ Life of a Packet	21
8	9/9/2025	Configure Web Server, DNS within a LAN.	23
9	18/11/2025	To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)	25
10	18/11/2025	To understand the operation of TELNET by accessing the router in server room from a PC in IT office.	28
11	14/10/2025	To construct a VLAN and make the PC's communicate among a VLAN	31
12	11/11/2025	To construct a WLAN and make the nodes communicate wirelessly	34
13	28/10/2025	Write a program for error detecting code using CRC-CCITT (16-bits).	37
14	28/10/2025	Write a program for congestion control using Leaky bucket algorithm.	42
15	17/11/2025	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.	45
16	17/11/2025	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.	47

Github Link : [github.com/eashanjn/CN](https://github.com/eashanjn/CN)

### Program 1

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

Observation:



- 4) Click on PC and go to config tab. Set IP address to 10.0.0.1 and click on subnetMask
- 5) Repeat same step and set IP address for server
- 6) In simulation mode in edit filters click only ICMP
- 7) Add simple PDU from PC to server
- 8) Click on autocapture/play

Hub -

Procedure -

- 1) Select the end devices and change their IP addresses suitably
- 2) Select hub as the connecting device
- 3) Select copper straight through as the connecting wire between end devices and hub
- 4) Connect the fastethernet to hub ports
- 5) Select the message and first click on source device and destination device
- 6) Observe the packet transmission and acknowledgement receiving procedure.

Switch

Procedure -

- 1) Select the end devices and change their IP addresses suitably
- 2) Select switch as the connecting device
- 3) Select copper straight through as the connecting wire between end devices and hub.
- 4) Connect fastethernet to hub ports
- 5) Select the msg and first click on source device and then the destination device
- 6) Observe the packet transmission and acknowledgement receiving procedure.

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=1ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128

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 = 1ms, Average = 0ms

PC>
```

Command Prompt

```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.5

Pinging 10.0.0.5 with 32 bytes of data:

Reply from 10.0.0.5: bytes=32 time=1ms TTL=128
Reply from 10.0.0.5: bytes=32 time=0ms TTL=128
Reply from 10.0.0.5: bytes=32 time=0ms TTL=128
Reply from 10.0.0.5: bytes=32 time=0ms TTL=128

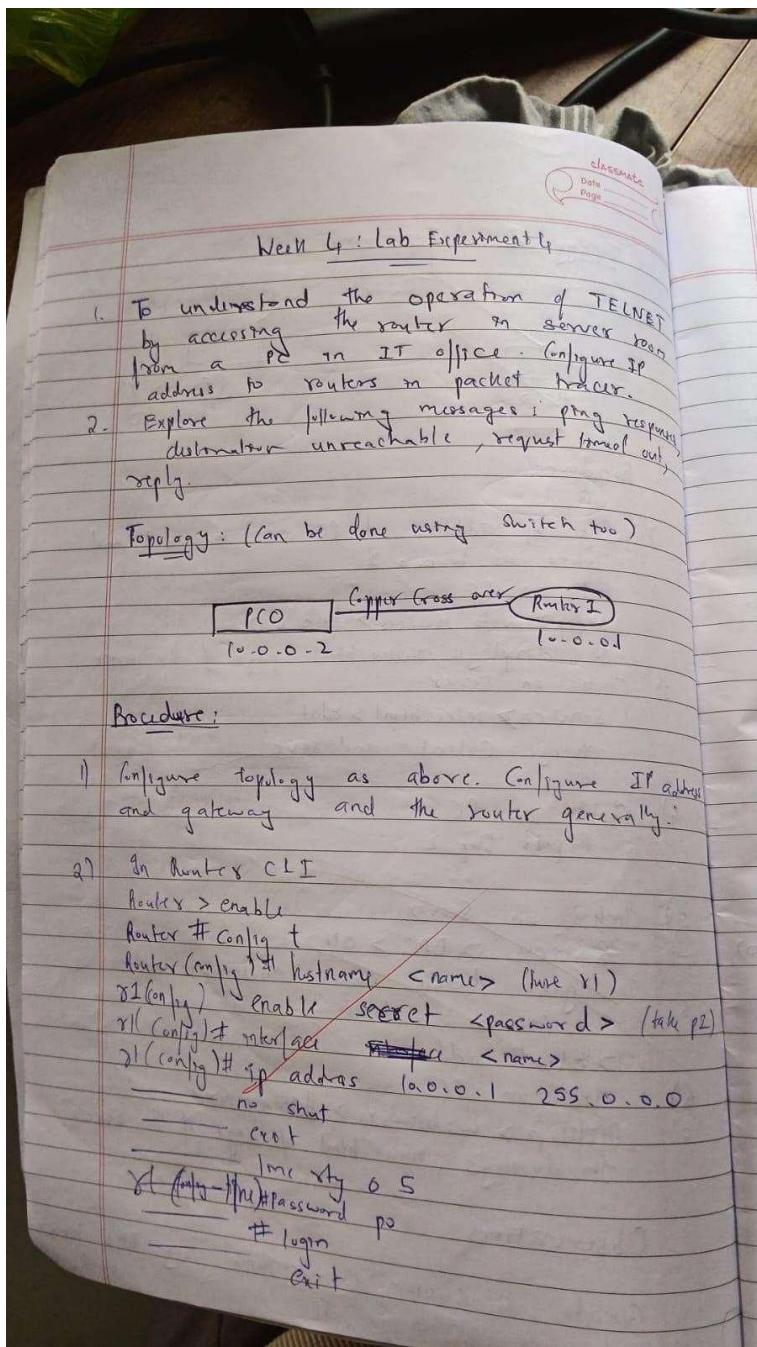
Ping statistics for 10.0.0.5:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

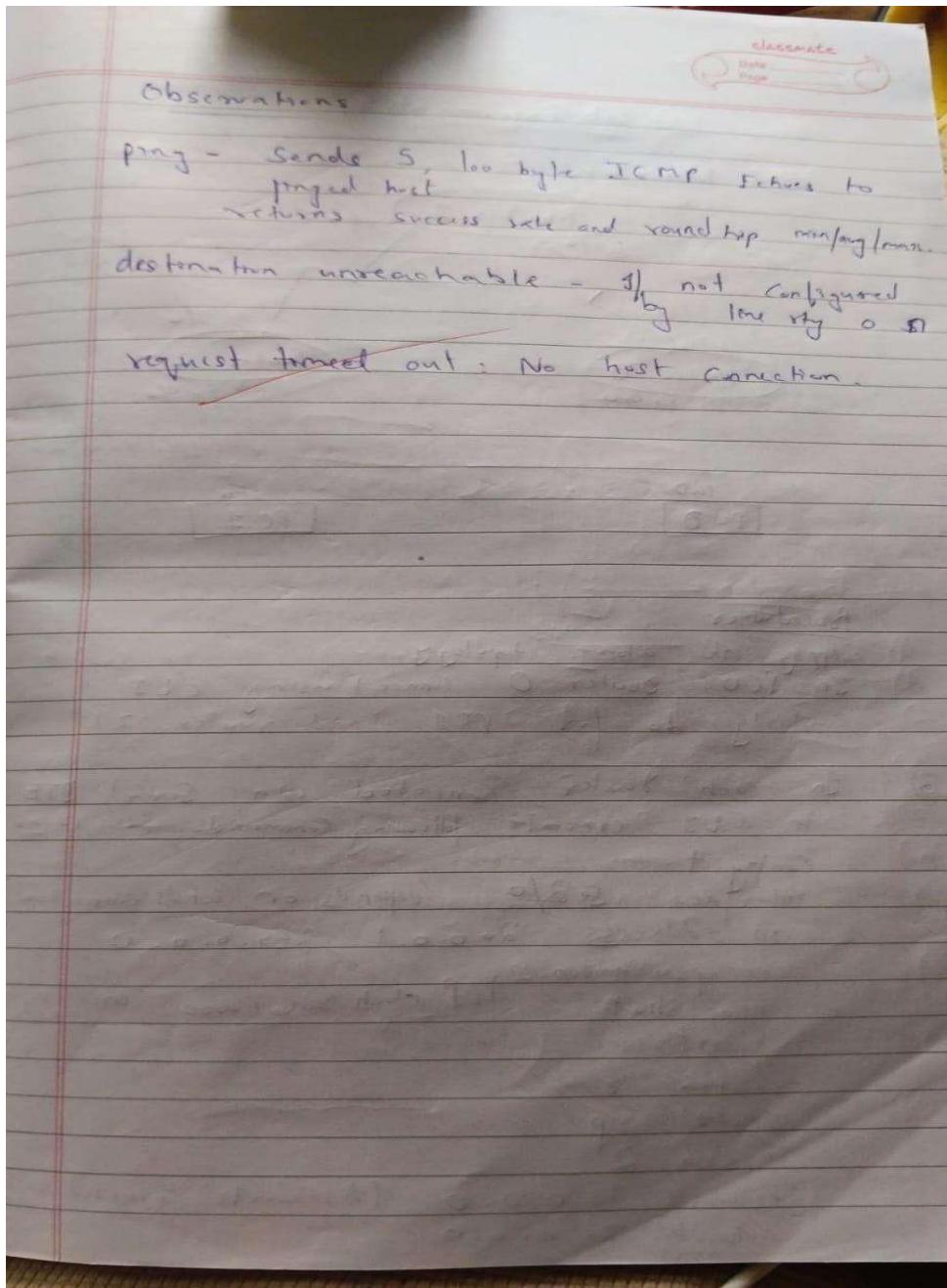
PC>
```

## Program 2

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

Observation:





Output:

## Command Prompt

```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.1

Pinging 10.0.0.1 with 32 bytes of data:

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

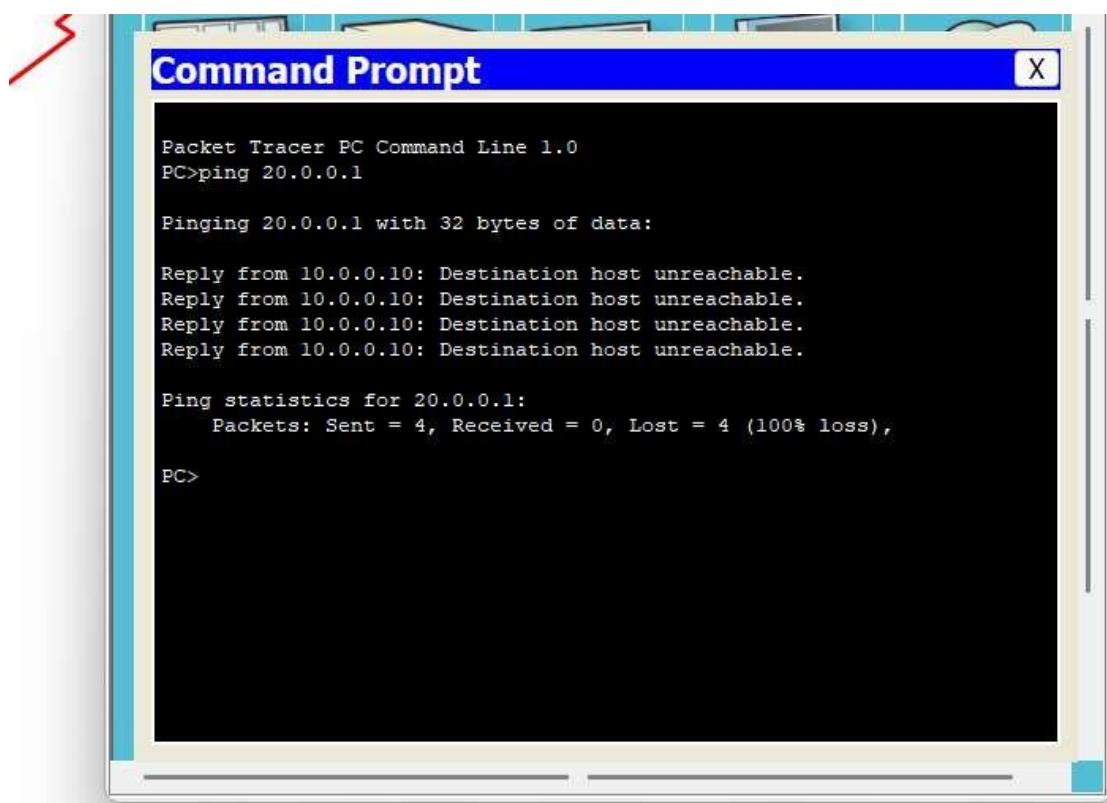
Ping statistics for 10.0.0.1:
  Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>
```

```
[connection to 10.0.0.1 closed by foreign host]
PC>ping 10.0.0.1

Pinging 10.0.0.1 with 32 bytes of data:

Reply from 10.0.0.1: bytes=32 time=3ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255

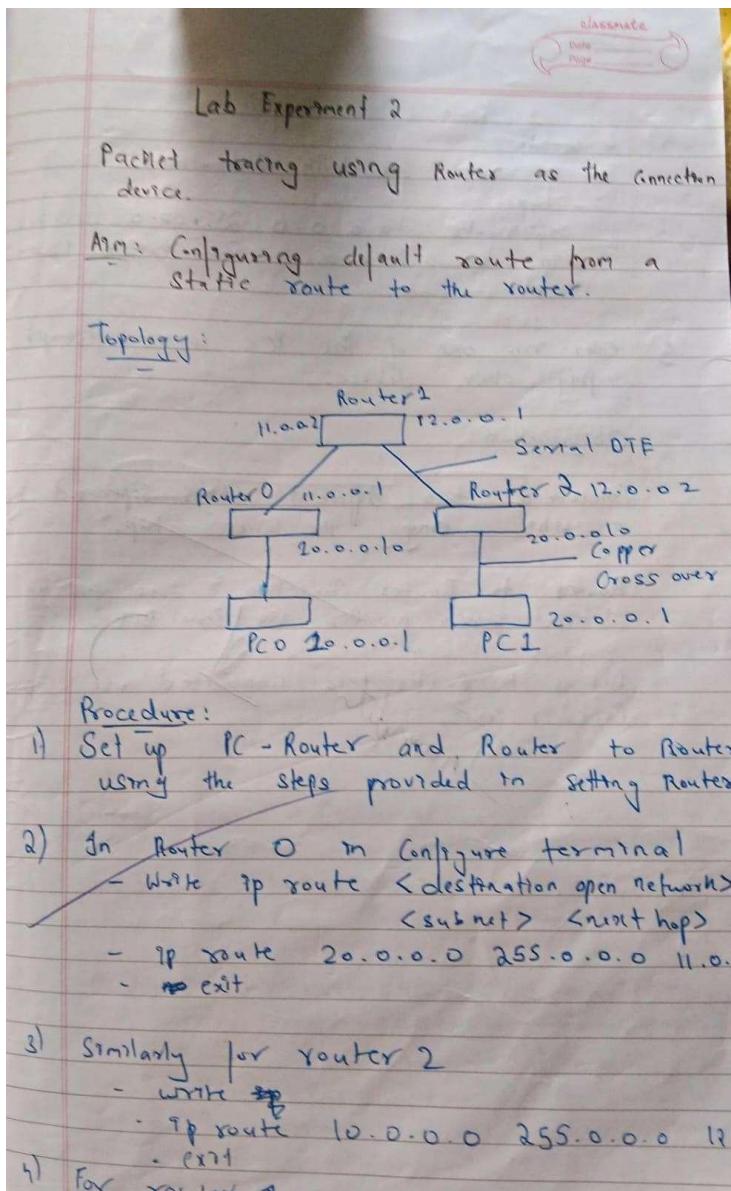
Ping statistics for 10.0.0.1:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
  Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 3ms, Average = 0ms
PC>
```

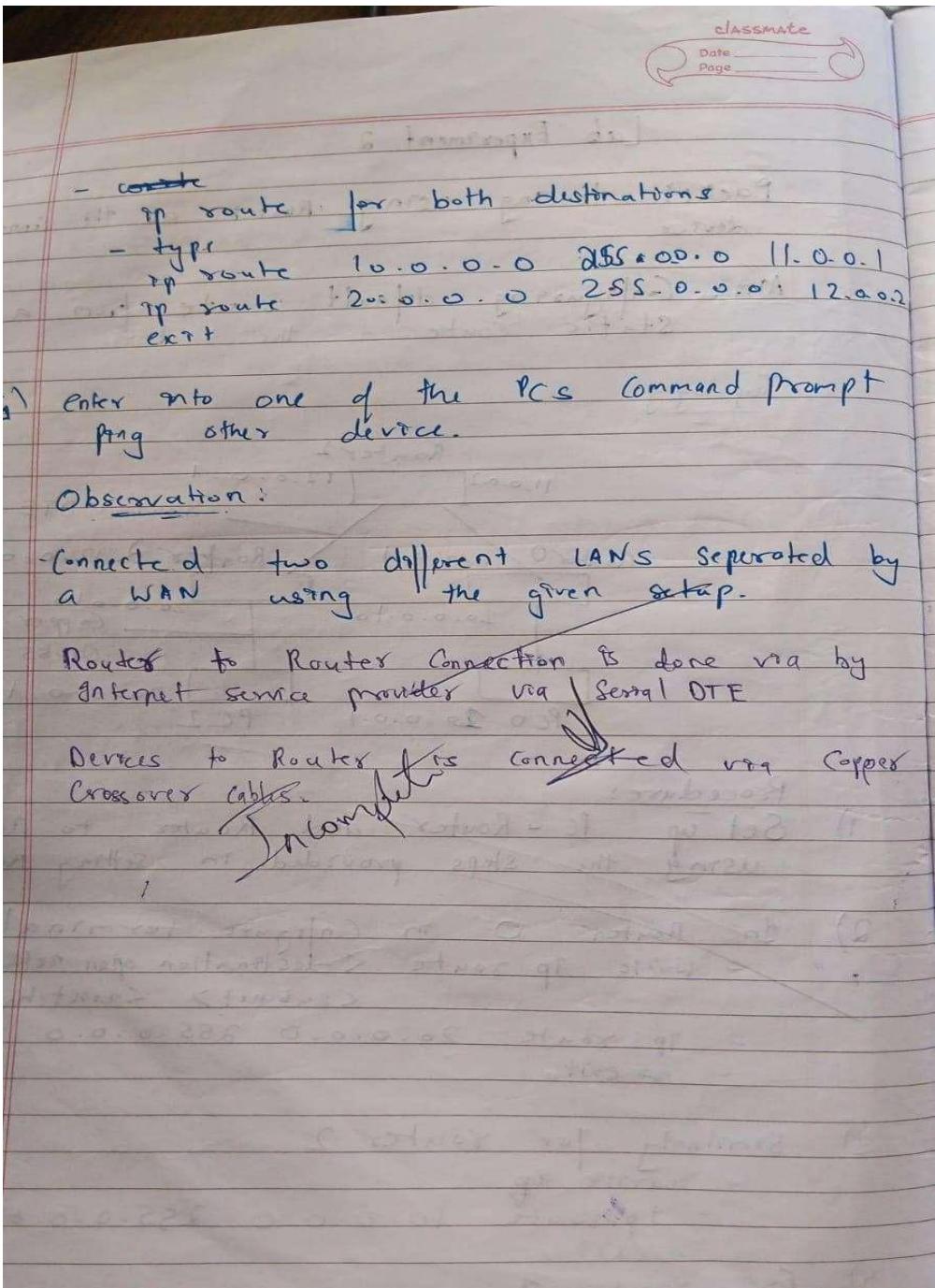


### Program 3

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

Observation:





## Output:

```
Command Prompt X
Pinging 20.0.0.1 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 20.0.0.1:
  Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
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=23ms TTL=125
Reply from 20.0.0.1: bytes=32 time=2ms TTL=125
Reply from 20.0.0.1: bytes=32 time=17ms TTL=125
Reply from 20.0.0.1: bytes=32 time=15ms TTL=125

Ping statistics for 20.0.0.1:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
  Minimum = 2ms, Maximum = 23ms, Average = 14ms
PC>
```

```
Command Prompt X
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.1

Pinging 10.0.0.1 with 32 bytes of data:

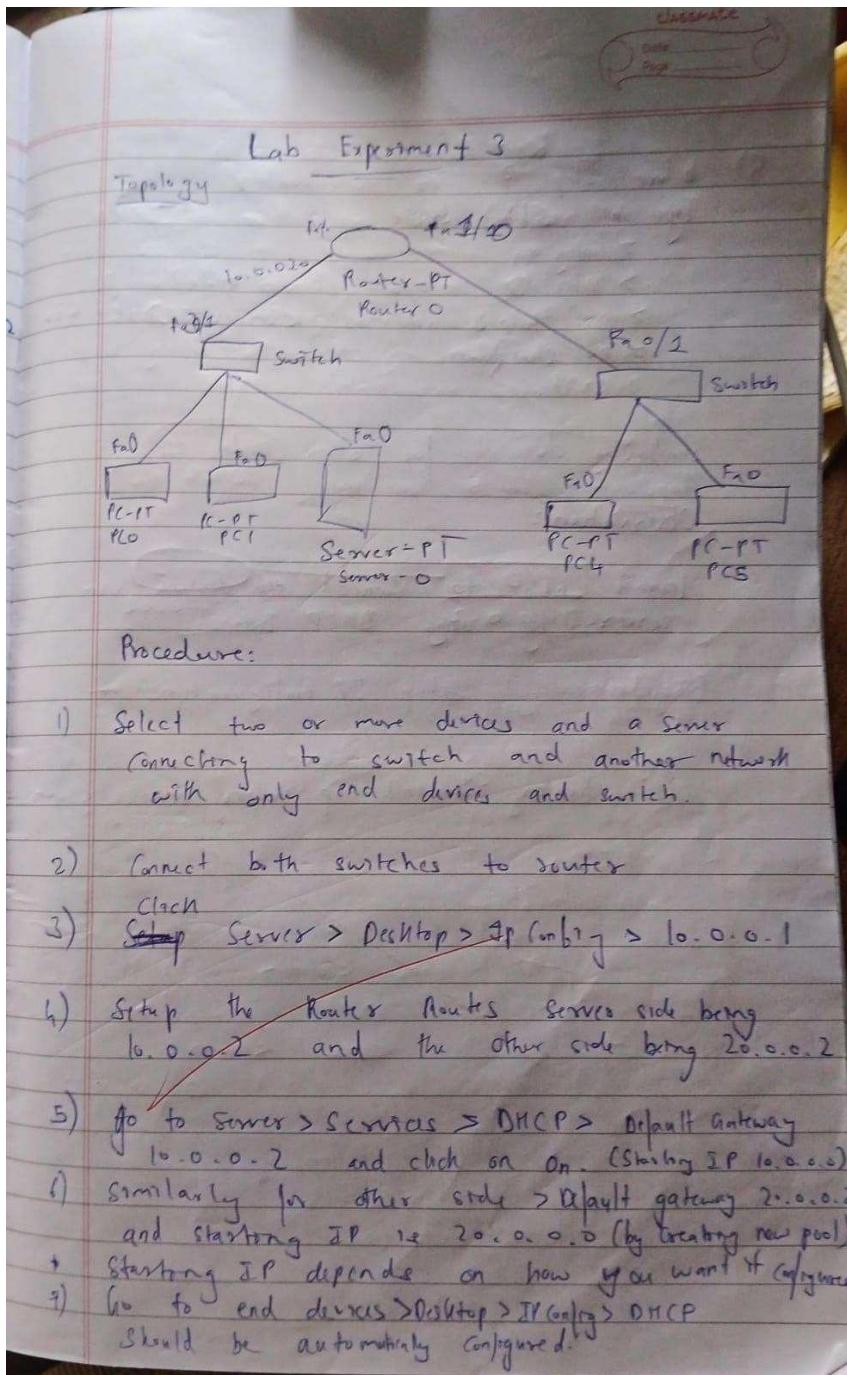
Reply from 10.0.0.1: bytes=32 time=20ms TTL=125
Reply from 10.0.0.1: bytes=32 time=27ms TTL=125
Reply from 10.0.0.1: bytes=32 time=3ms TTL=125
Reply from 10.0.0.1: bytes=32 time=2ms TTL=125

Ping statistics for 10.0.0.1:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
  Minimum = 2ms, Maximum = 27ms, Average = 13ms
PC>
```

## Program 4

Aim : Configure DHCP within a LAN and outside LAN.

Observation:



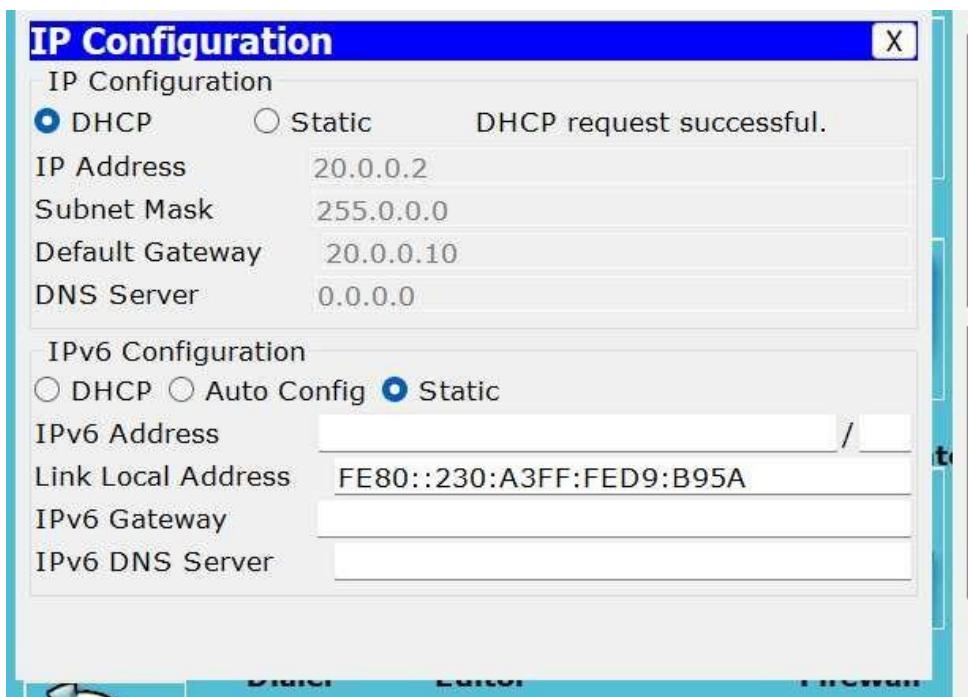
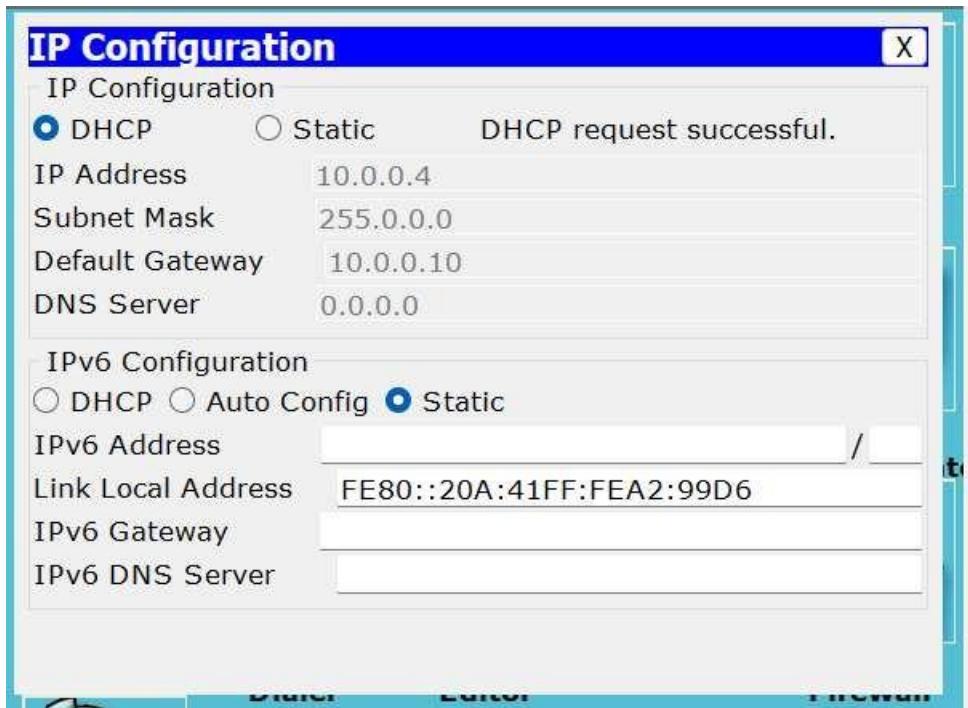
- 8) before step 7 go to router  
go to config write  
> ~~enable~~  
> config +  
> interface Fa 0/0 (server side)  
> ip helper-address <server IP> (here 192.0.0.1)  
> no shutdown  
> exit.

Similarly do it for other side.

Observation:

- learnt how to connect end devices via server by using DHCP Protocol.
- Dynamically setup IP configurations for end devices
- New Router Command ip helper address <server IP>

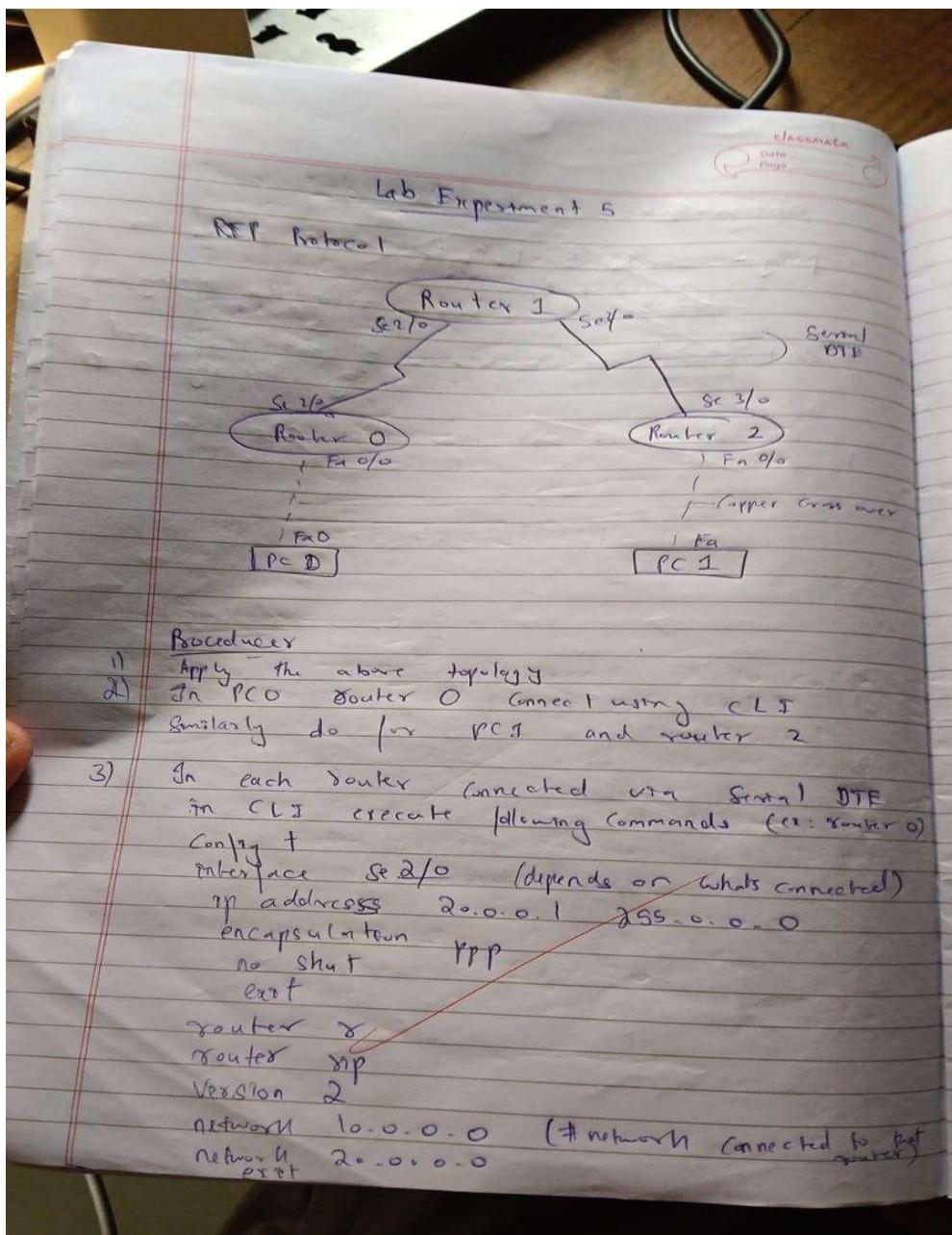
Output:

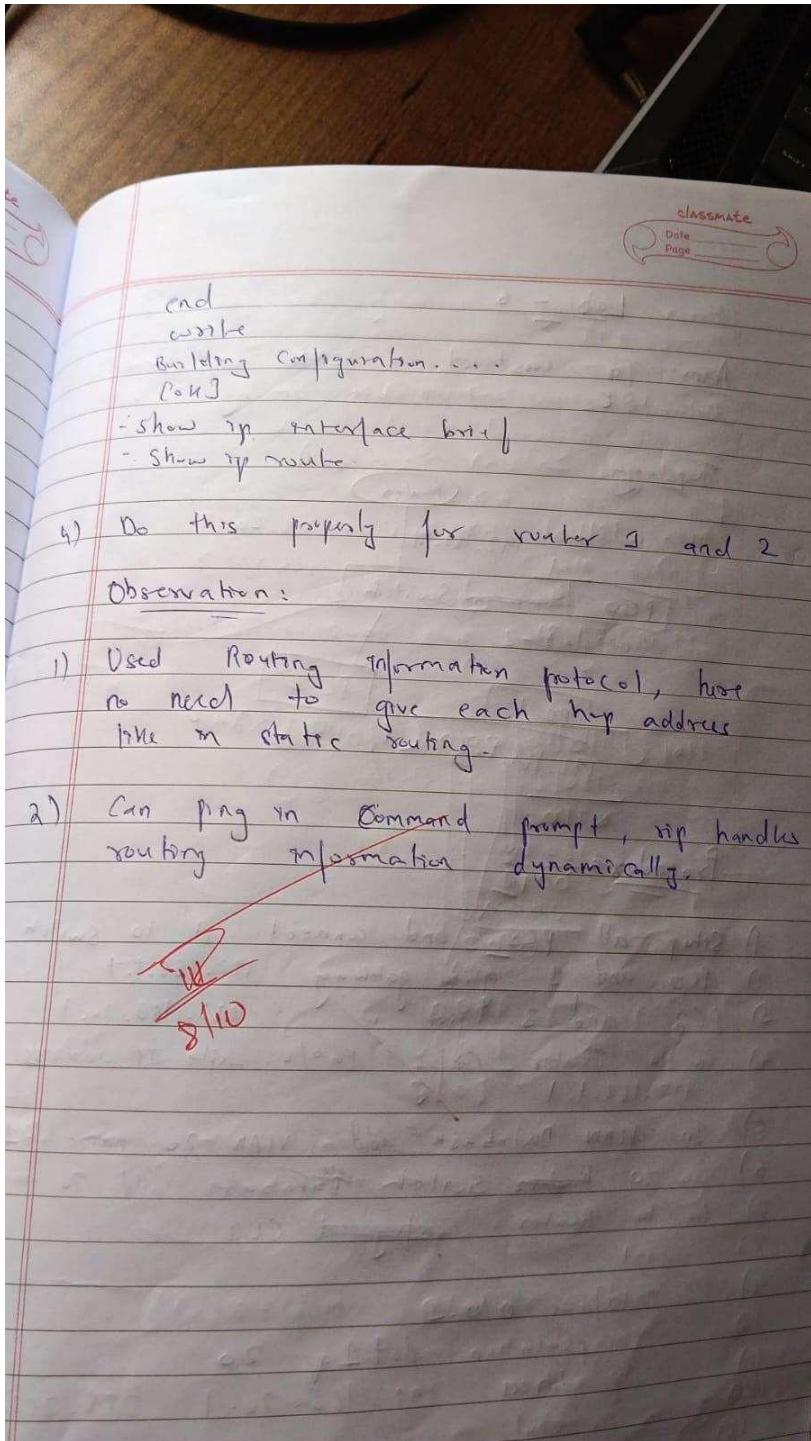


## Program 5

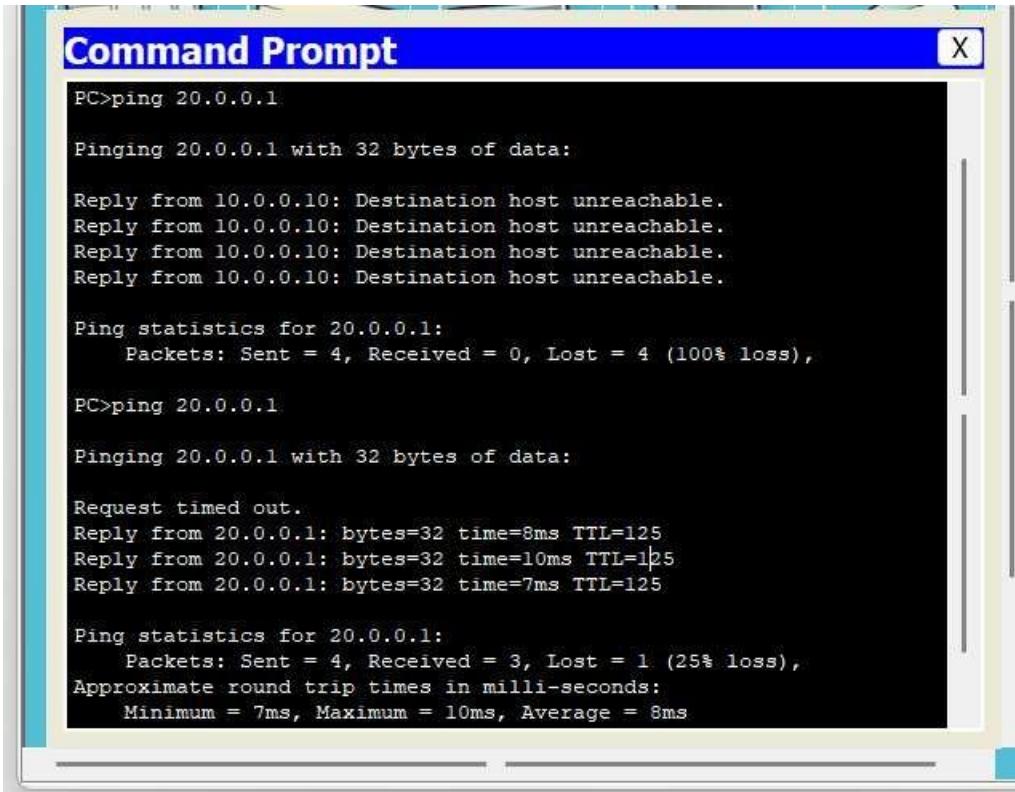
Aim : Configure RIP routing Protocol in Routers

Observations :





Output:



The screenshot shows a Windows Command Prompt window titled "Command Prompt". The window contains the following text output:

```
PC>ping 20.0.0.1
Pinging 20.0.0.1 with 32 bytes of data:
Reply from 10.0.0.10: Destination host unreachable.

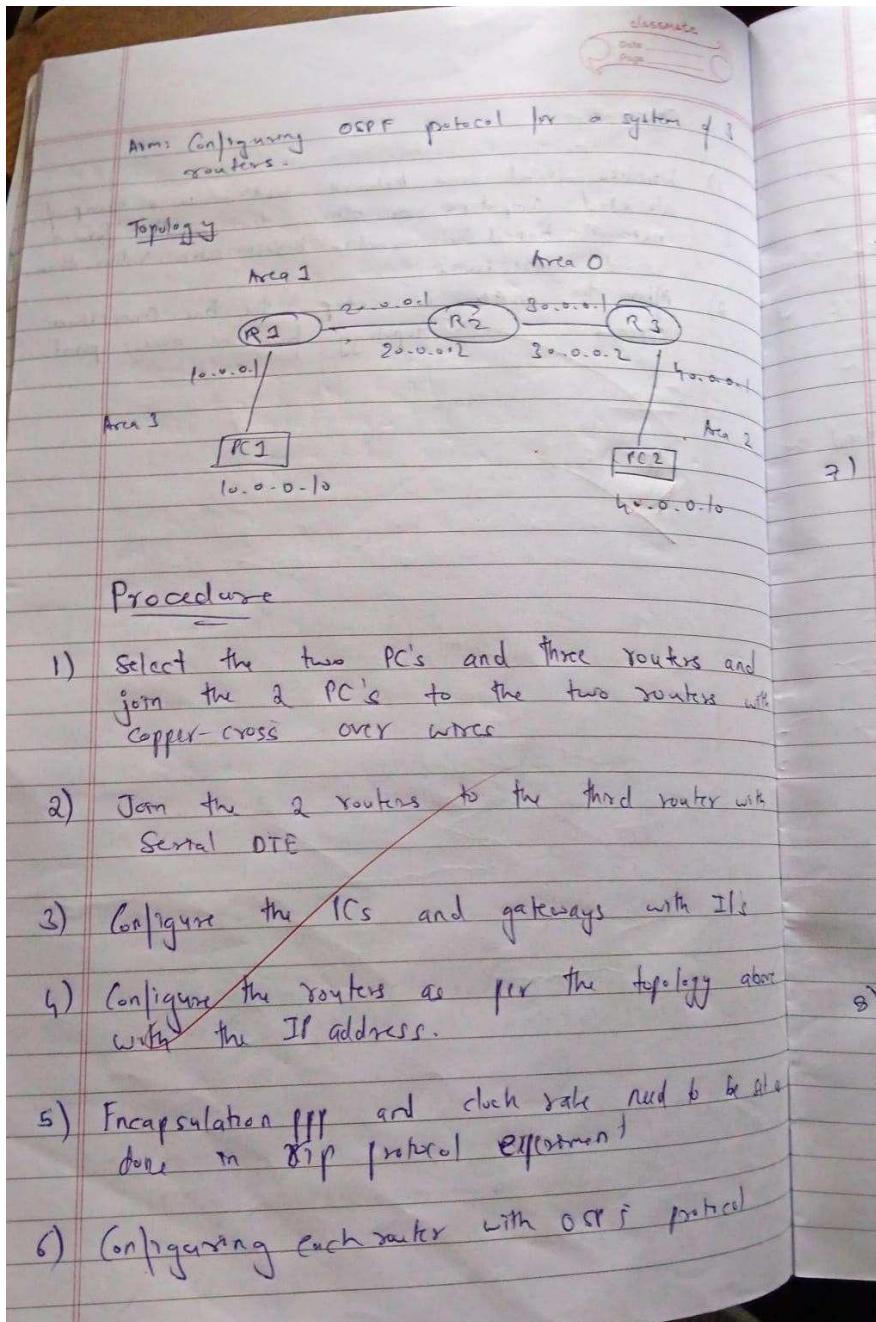
Ping statistics for 20.0.0.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>ping 20.0.0.1
Pinging 20.0.0.1 with 32 bytes of data:
Request timed out.
Reply from 20.0.0.1: bytes=32 time=8ms TTL=125
Reply from 20.0.0.1: bytes=32 time=10ms TTL=125
Reply from 20.0.0.1: bytes=32 time=7ms TTL=125

Ping statistics for 20.0.0.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 7ms, Maximum = 10ms, Average = 8ms
```

## Program 6

Aim : Configure OSPF routing Protocol

Observations:



For router 1

Config

router ospf 1

router-id 1.1.1.1

network 10.0.0.0 0.255.255.255 area 3

network 20.0.0.0 0.255.255.255 area 1

Similarly do for other routers with id 2 & 3

7) Configure the interface

(config)  
R0# interface loopback 0

ip address 192.16.1.252 255.255.0.0  
no shutdown

R1 interface loopback 0

253

R2

254

show ip route

8) Configure virtual link

R0 (config)# router ospf 1

area 2 virtual-link 2.2.2.2

config

R1 (config)

area 1

1.1.1.1

exit

Q) Show ip route configured in all routers

• Troubleshooting

show ip protocols

show ip ospf interface

#### Observation

(open shortest path first)

OSPF protocol uses Dijkstra's algorithm to find the shortest path to reach the destination.

Output:

```
Command Prompt X

Packet Tracer PC Command Line 1.0
PC>ping 40.0.0.10

Pinging 40.0.0.10 with 32 bytes of data:

Request timed out.
Reply from 40.0.0.10: bytes=32 time=8ms TTL=125
Reply from 40.0.0.10: bytes=32 time=7ms TTL=125
Reply from 40.0.0.10: bytes=32 time=7ms TTL=125

Ping statistics for 40.0.0.10:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 7ms, Maximum = 8ms, Average = 7ms

PC>`ping 10.0.0.1
Invalid Command.

PC>ping 20.0.0.2

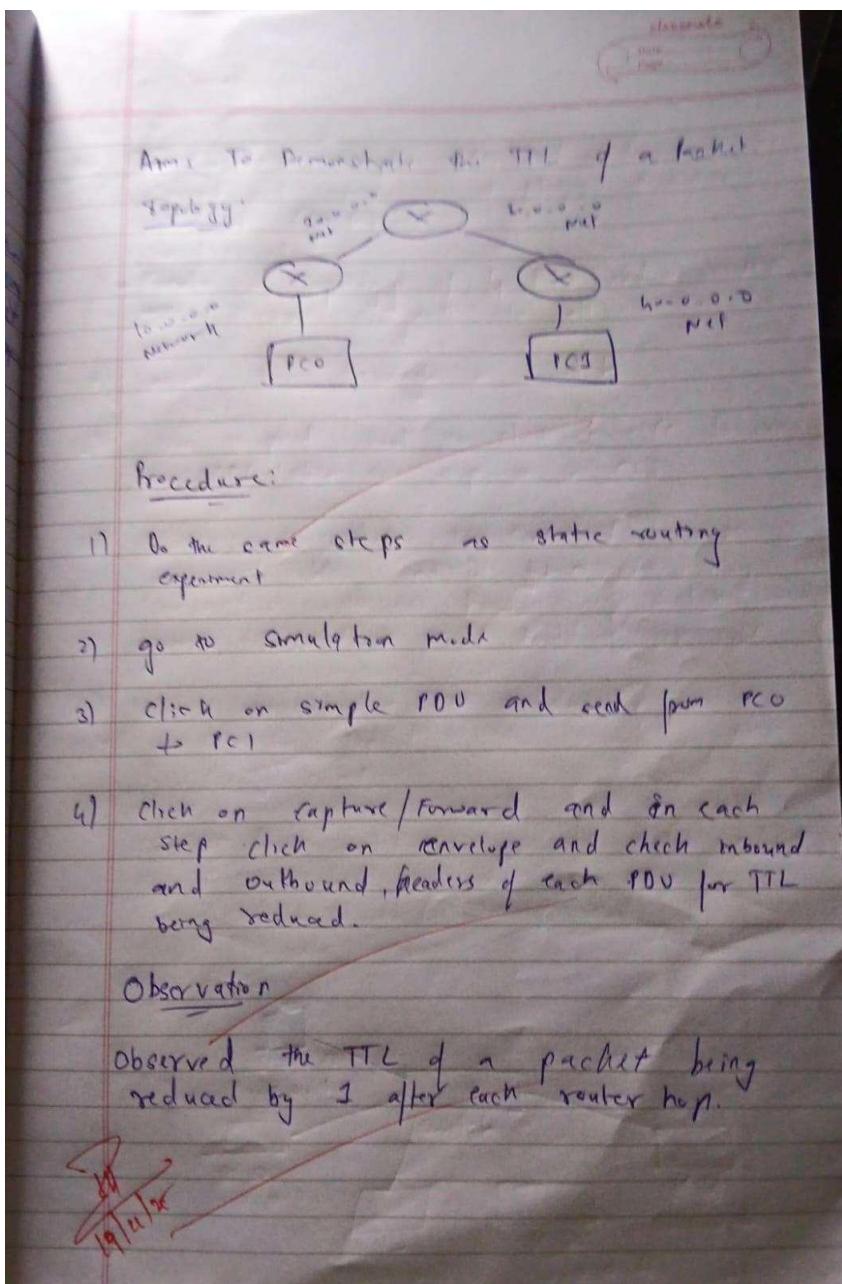
Pinging 20.0.0.2 with 32 bytes of data:

Reply from 20.0.0.2: bytes=32 time=8ms TTL=254
Reply from 20.0.0.2: bytes=32 time=5ms TTL=254
```

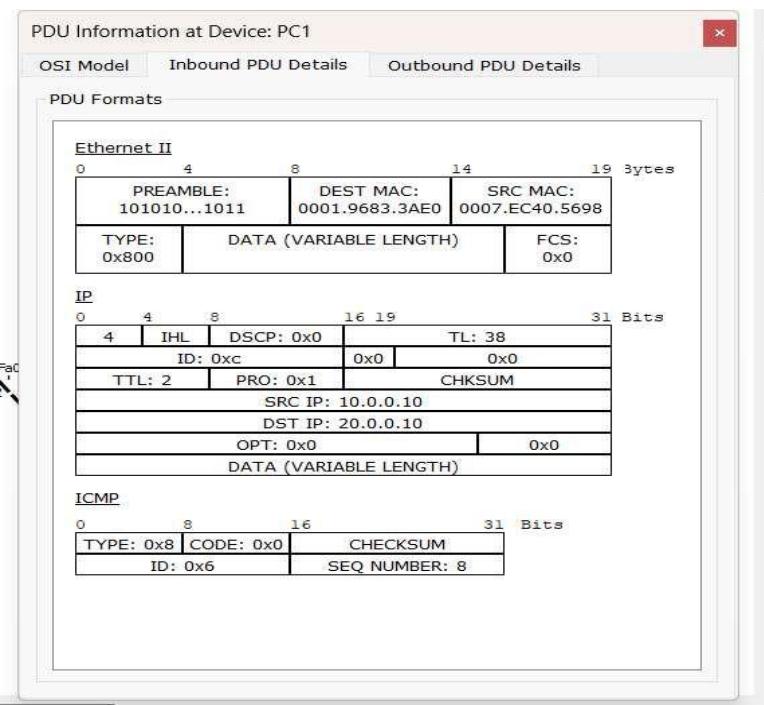
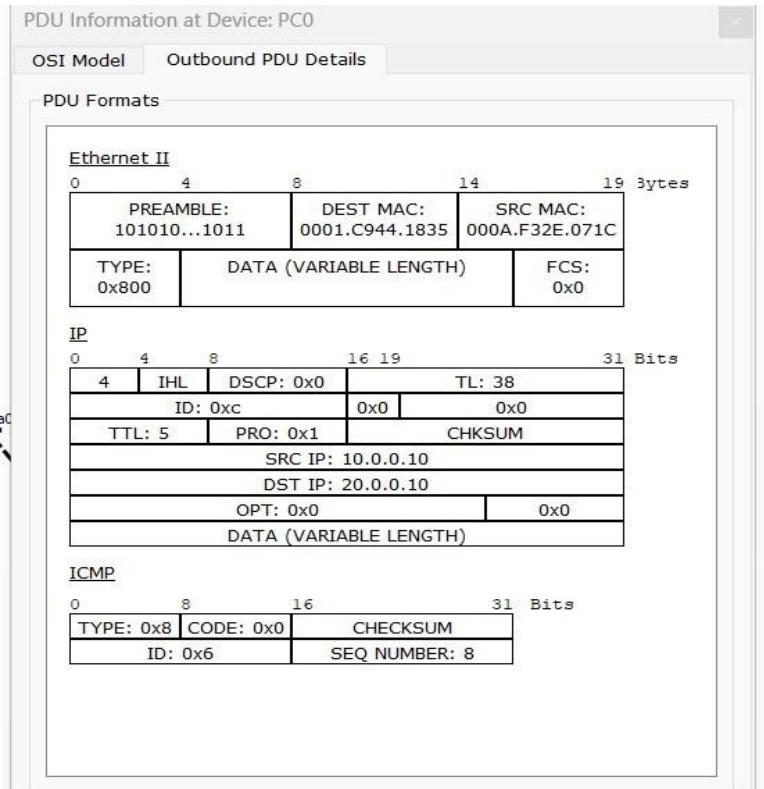
## Program 7

Aim : Demonstrate the TTL/ Life of a Packet

Observations:



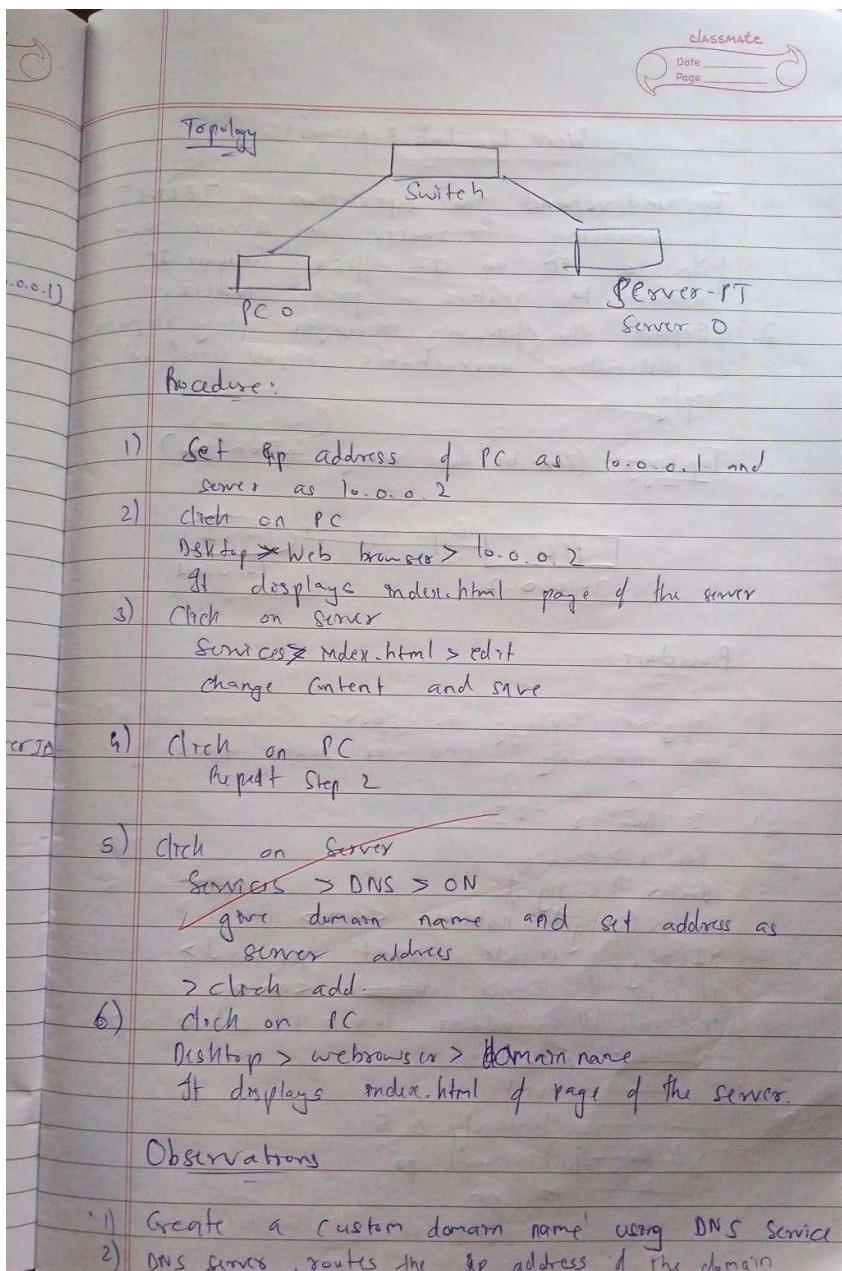
## Output:



## Program 8

Aim : Configure Web Server, DNS within a LAN.

Observation:



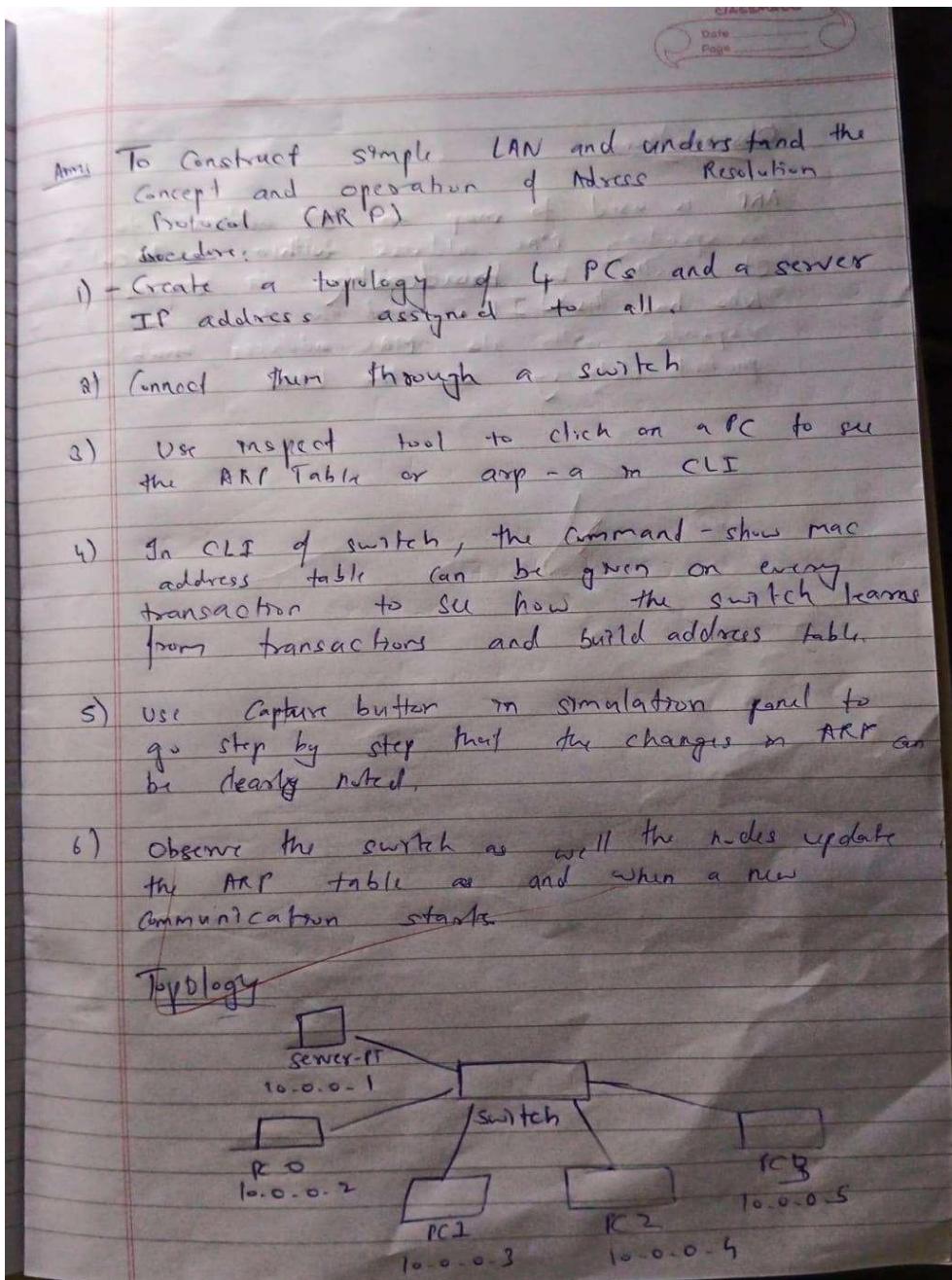
Output:

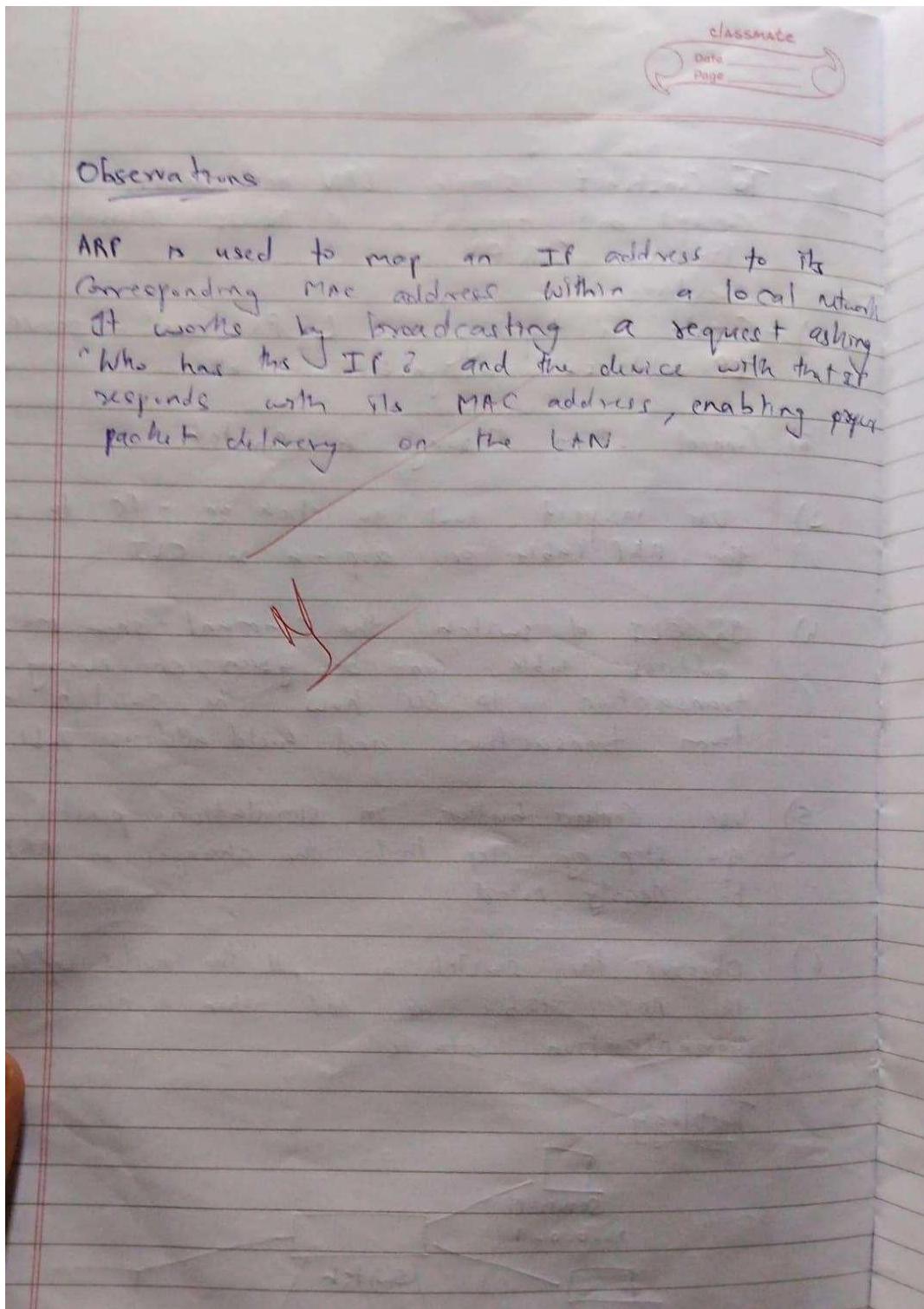


## Program 9

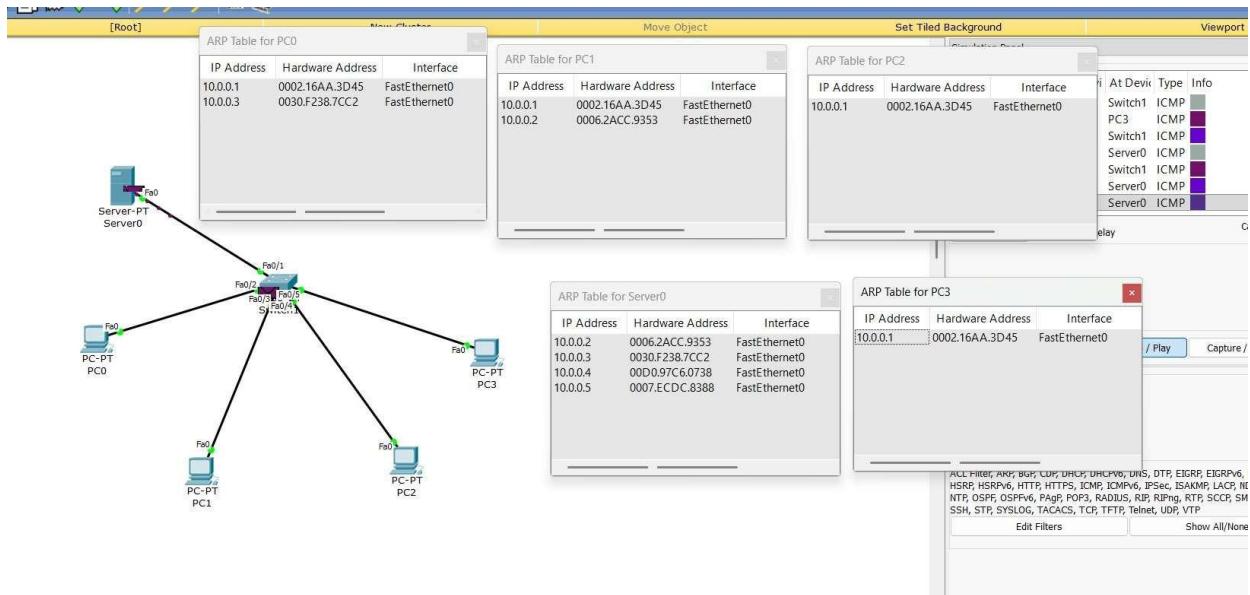
Aim : To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)

Observation:





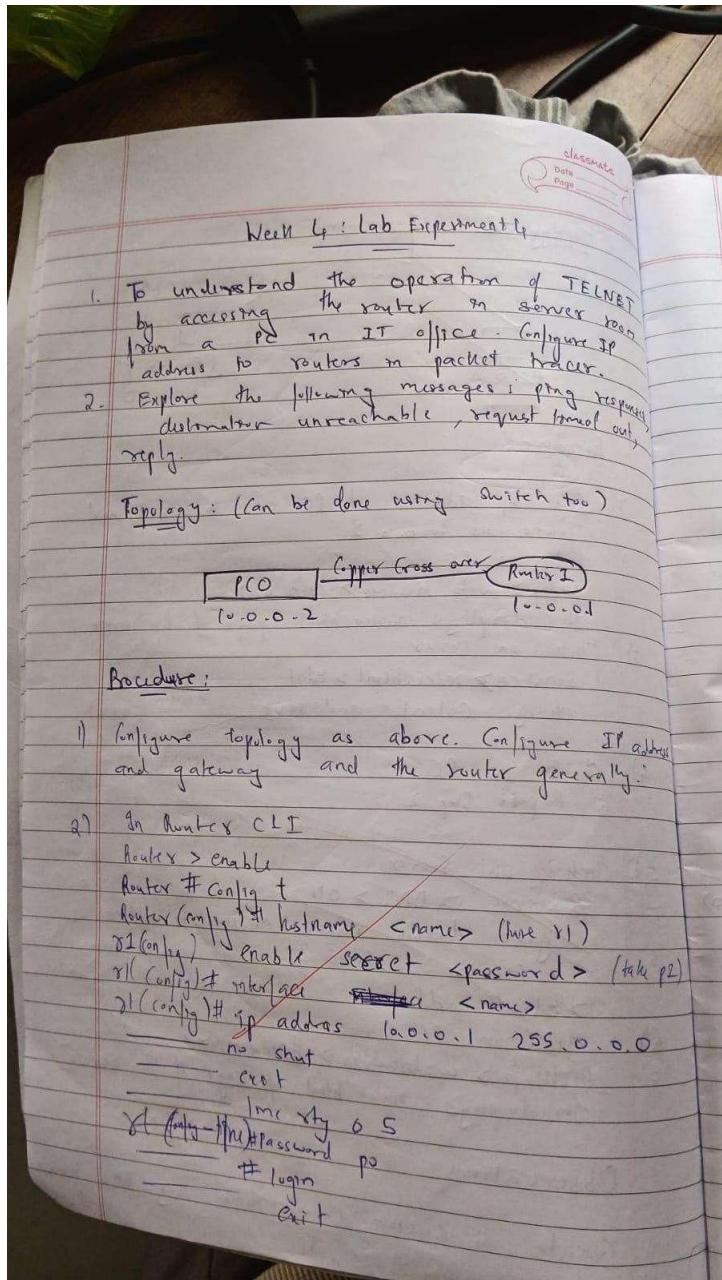
## Output:



## Program 10

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

Observation:



Observations

classmate

Date \_\_\_\_\_

Page \_\_\_\_\_

ping - Sends 5, 100 byte ICMP packets to  
pinged host  
returns success rate and round trip min/avg/max.

destination unreachable - If not configured  
by trying to a

~~request timed out: No host connection.~~

## Output:

```
PC>telnet 10.0.0.3  
Trying 10.0.0.3 ...  
% Connection timed out; remote host not responding
```

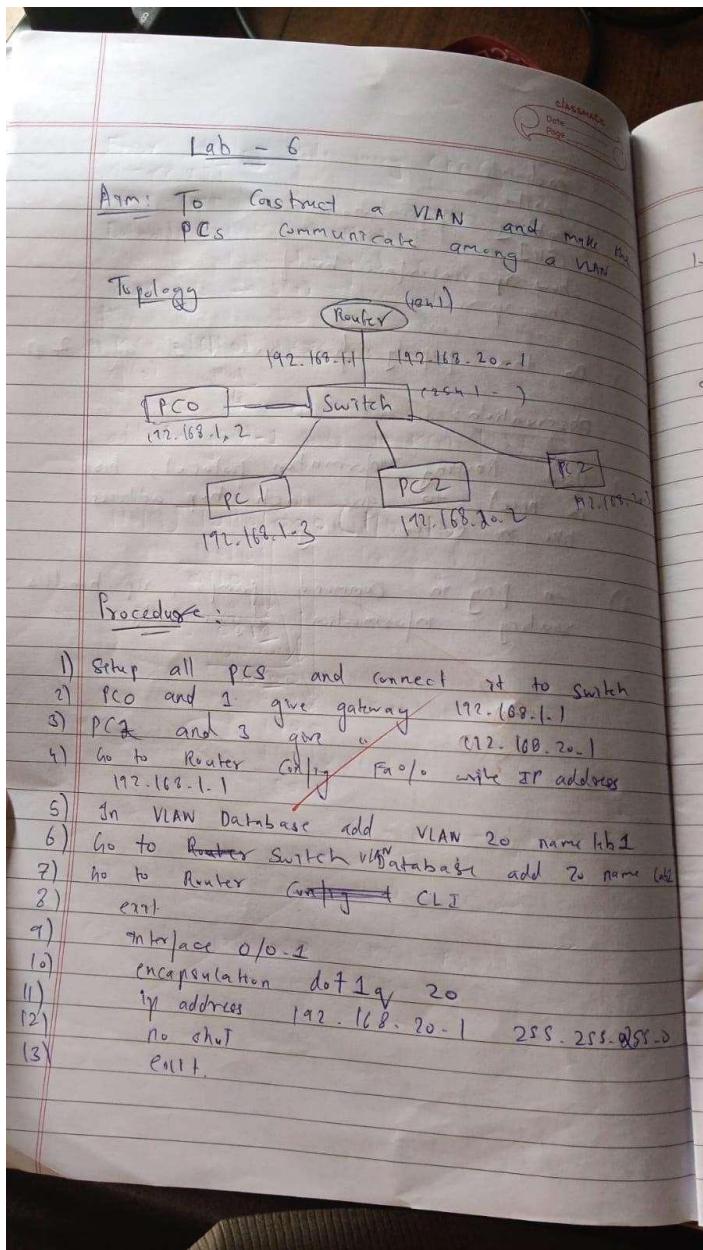
```
PC>telnet 10.0.0.1  
Trying 10.0.0.1 ...Open  
  
[Connection to 10.0.0.1 closed by foreign host]  
r1>
```

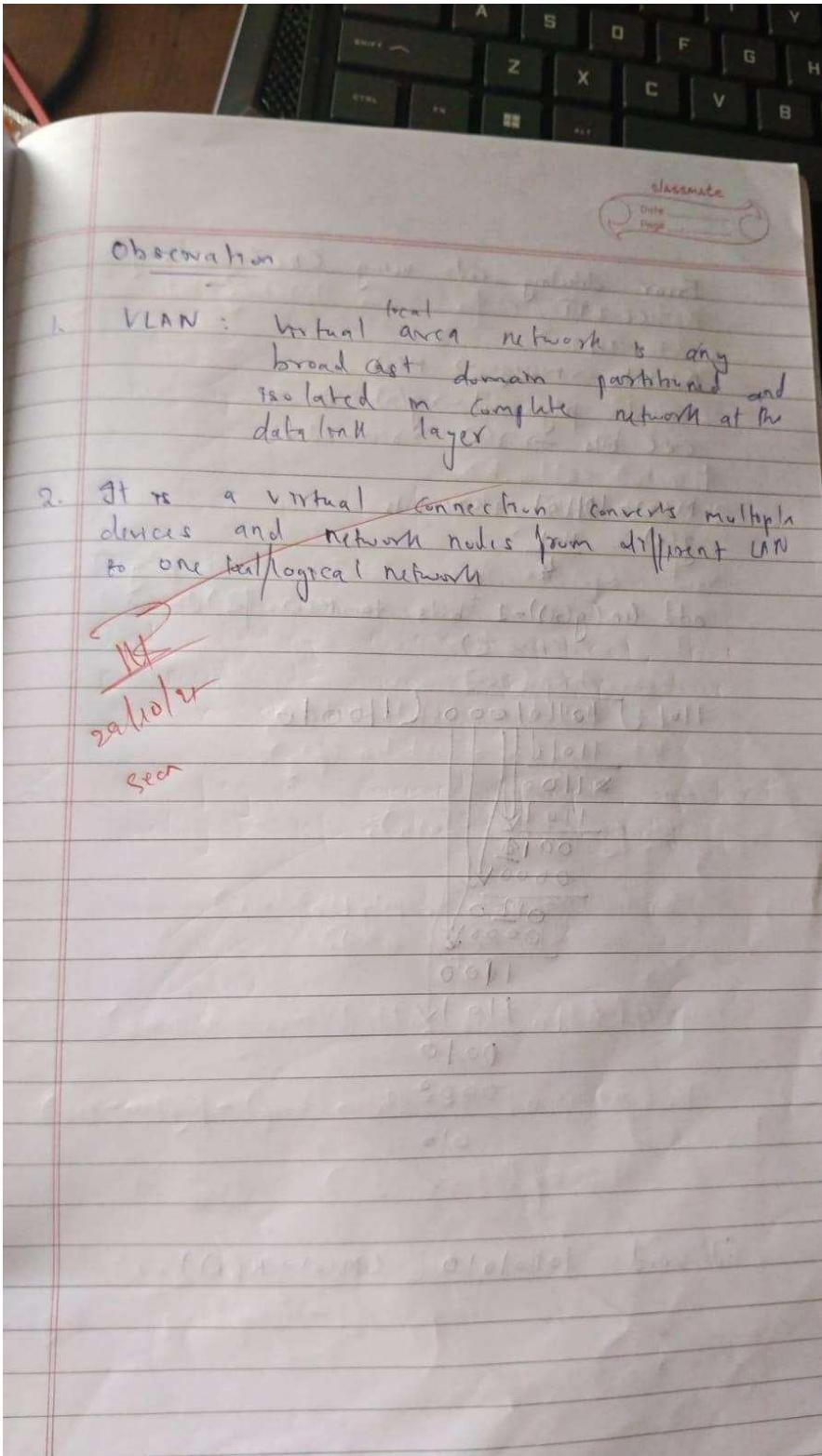
```
[Connection to 10.0.0.1 closed by foreign host]  
PC>telnet 10.0.0.1  
Trying 10.0.0.1 ...Open  
  
User Access Verification  
  
Password:  
r1>ping 10.0.0.2  
  
Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echos to 10.0.0.2, timeout is 2 seconds:  
!!!!!  
Success rate is 100 percent (5/5), round-trip min/avg/max = 0/0/1  
ms  
  
r1>
```

## Program 11

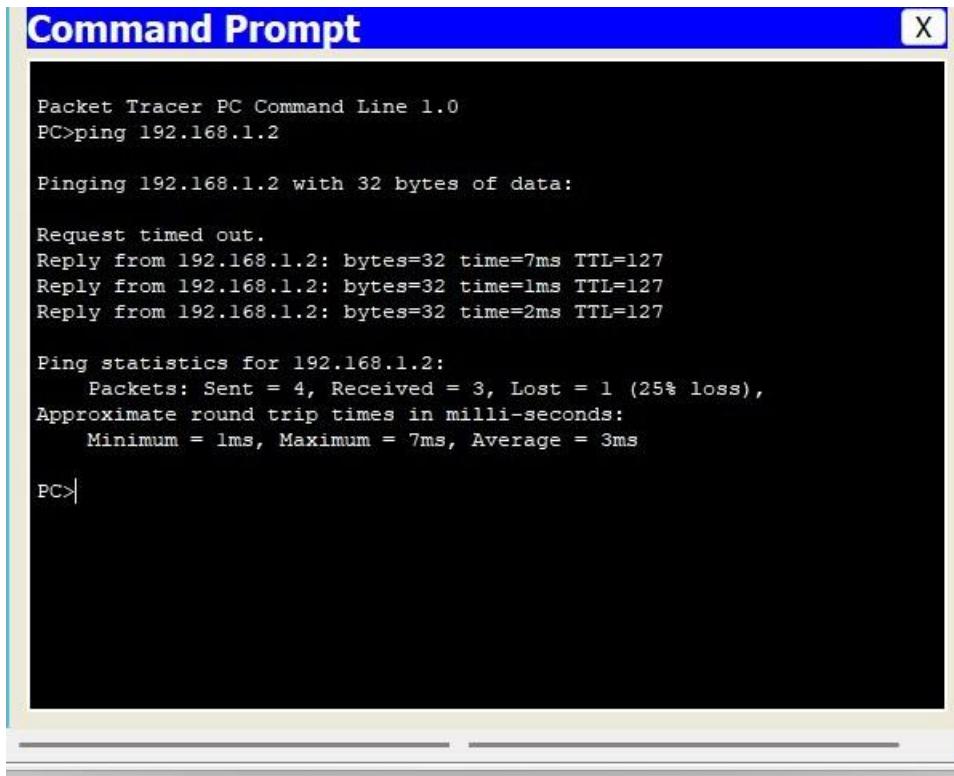
Aim : To construct a VLAN and make the PC's communicate among a VLAN

Observation:





## Output:



```
Command Prompt X

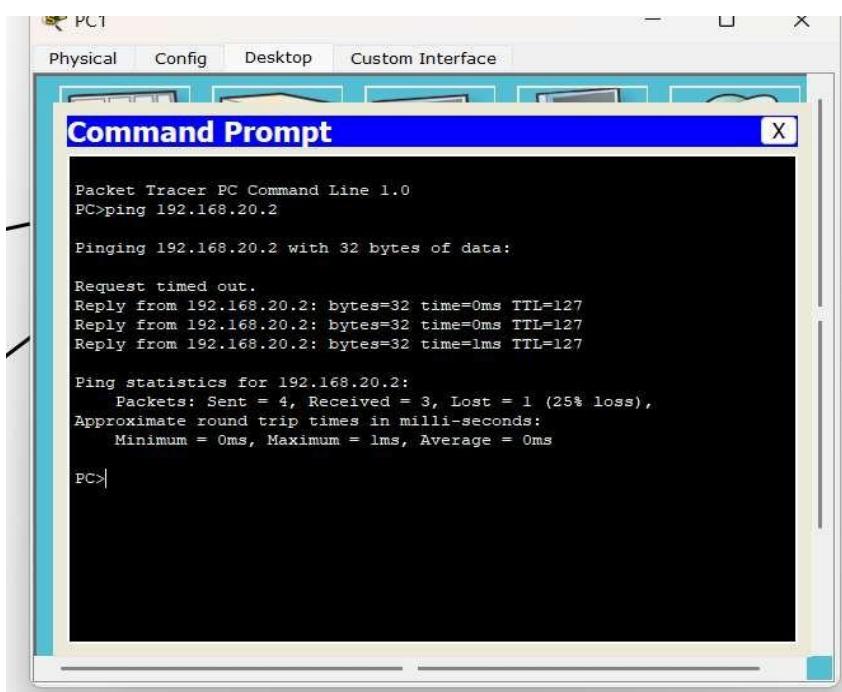
Packet Tracer PC Command Line 1.0
PC>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Request timed out.
Reply from 192.168.1.2: bytes=32 time=7ms TTL=127
Reply from 192.168.1.2: bytes=32 time=1ms TTL=127
Reply from 192.168.1.2: bytes=32 time=2ms TTL=127

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 7ms, Average = 3ms

PC>
```



```
PC1 Physical Config Desktop Custom Interface X

Command Prompt X

Packet Tracer PC Command Line 1.0
PC>ping 192.168.20.2

Pinging 192.168.20.2 with 32 bytes of data:

Request timed out.
Reply from 192.168.20.2: bytes=32 time=0ms TTL=127
Reply from 192.168.20.2: bytes=32 time=0ms TTL=127
Reply from 192.168.20.2: bytes=32 time=1ms TTL=127

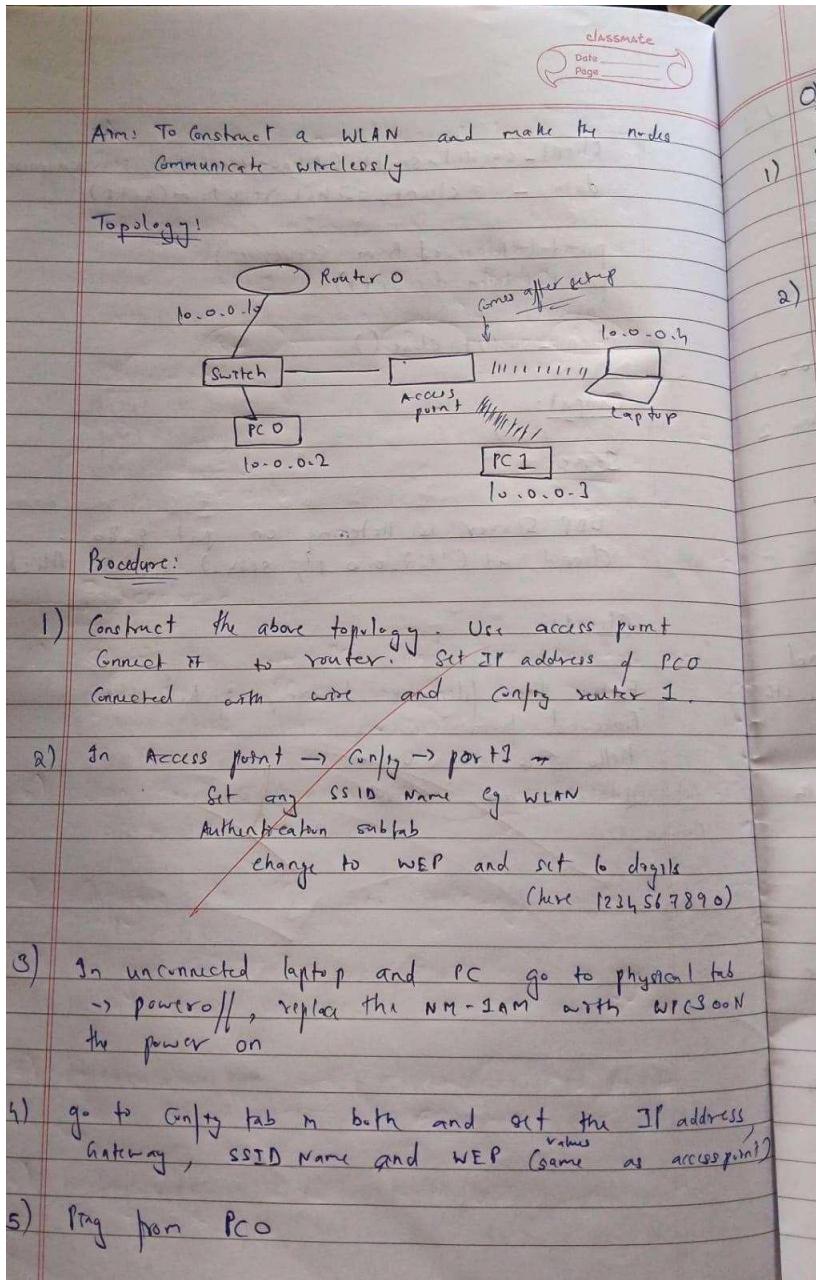
Ping statistics for 192.168.20.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

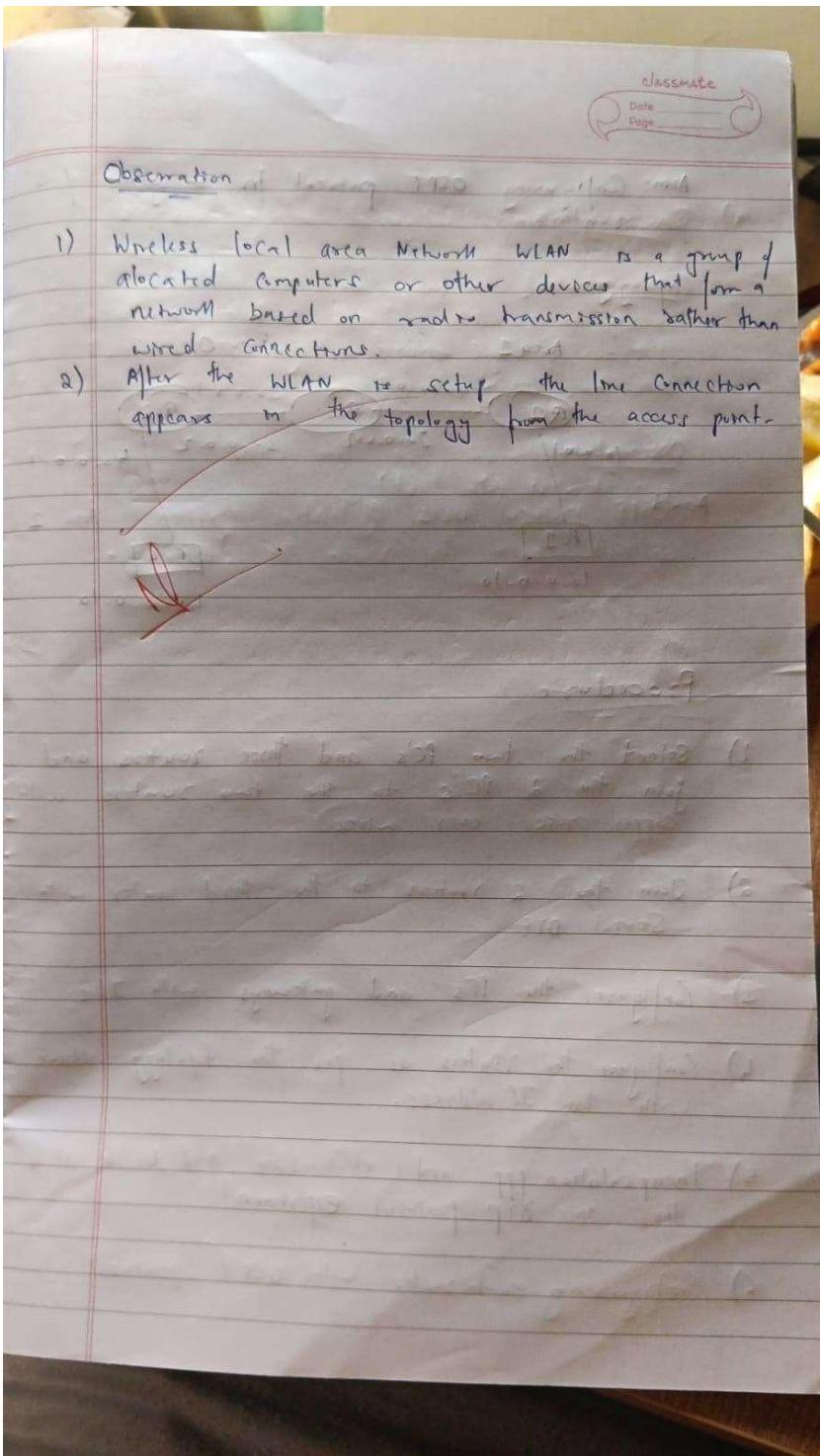
PC>
```

## Program 12

Aim : To construct a WLAN and make the nodes communicate wirelessly

Observation:





Output:

```
Command Prompt
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=22ms TTL=128
Reply from 10.0.0.4: bytes=32 time=7ms TTL=128
Reply from 10.0.0.4: bytes=32 time=9ms TTL=128
Reply from 10.0.0.4: bytes=32 time=6ms 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 = 6ms, Maximum = 22ms, Average = 11ms

PC>ping 10.0.0.5

Pinging 10.0.0.5 with 32 bytes of data:

Reply from 10.0.0.5: bytes=32 time=22ms TTL=128
Reply from 10.0.0.5: bytes=32 time=7ms TTL=128
Reply from 10.0.0.5: bytes=32 time=13ms TTL=128
Reply from 10.0.0.5: bytes=32 time=6ms TTL=128
```

### Program 13

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

Observation:

Error detecting code using CRC (Cyclic Redundancy Check)  
- CCITT

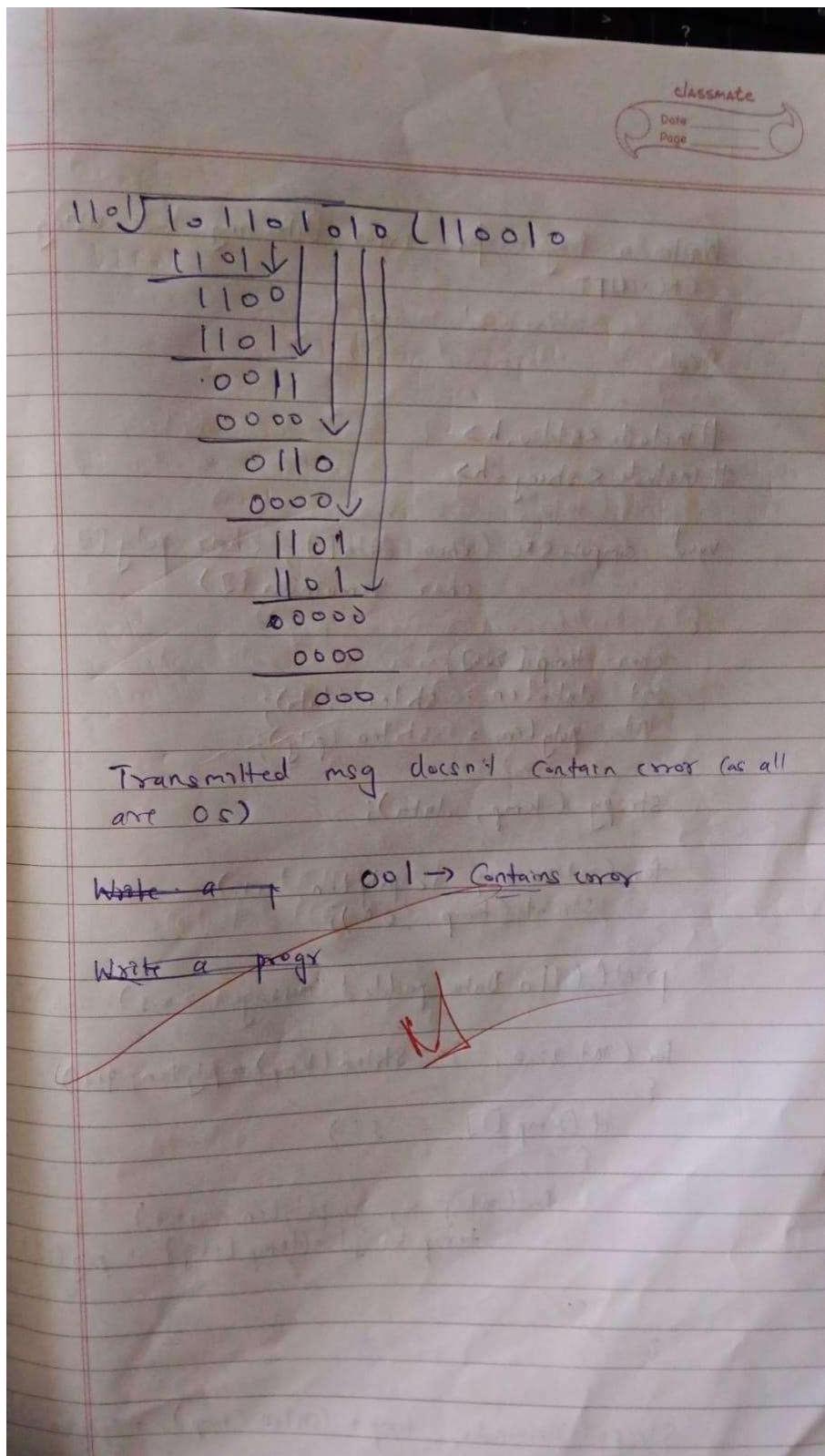
Databit  $\rightarrow$  message  $\rightarrow M(x)$   
Generator polynomial  $\rightarrow g(x) \rightarrow$  Divisor  
CRC bits  $\rightarrow r(x) \rightarrow$  remainder

$M(x) = 101101 + \text{CRC bits}$   
 $\underline{g(x)} = 1101$

add len( $g(x)$ ) - 2 bits to  $M(x)$  for division  
(here 3)

$$\begin{array}{r} 1101 \sqrt{101101000} (110010 \\ 1101 \downarrow | \\ \times 1100 \quad | \\ 1101 \downarrow \quad | \\ 0010 \quad | \\ \underline{0000} \quad | \\ 0100 \quad | \\ 0000 \quad | \\ \hline 1000 \quad | \\ 1101 \downarrow \\ 0010 \\ 0000 \\ \hline 010 \end{array}$$

Code word: 101101010 ( $M(x) + R(x)$ )



Code:

```
#include <stdio.h>
#include <string.h>

void computeCRC(char data[], char poly[], char remainder[])
{
    char temp[200];
    int dataLen = strlen(data);
    int polyLen = strlen(poly);

    strcpy(temp, data);

    // Append (polyLen - 1) zeros to data
    for (int i = 1; i < polyLen; i++)
        strcat(temp, "0");

    printf("\nData padded (message + zeros): %s\n", temp);

    // Perform binary division
    for (int i = 0; i <= strlen(temp) - polyLen; i++)
    {
        if (temp[i] == '1')
        {
            for (int j = 0; j < polyLen; j++)
                temp[i + j] = (temp[i + j] == poly[j]) ? '0' : '1';
        }
    }

    // CRC = last (polyLen-1) bits
    strcpy(remainder, temp + (strlen(temp) - (polyLen - 1)));
}

int checkReceived(char received[], char poly[])
{
    char temp[200];
```

```

int polyLen = strlen(poly);

strcpy(temp, received);

// Perform division on received message
for (int i = 0; i <= strlen(temp) - polyLen; i++)
{
    if (temp[i] == '1')
    {
        for (int j = 0; j < polyLen; j++)
            temp[i + j] = (temp[i + j] == poly[j]) ? '0' : '1';
    }
}

// Check any '1' in remainder portion
for (int i = strlen(temp) - (polyLen - 1); i < strlen(temp); i++)
    if (temp[i] == '1')
        return 0; // Error detected

return 1; // No error
}

int main()
{
    char message[100], poly[30], crc[30], transmitted[150], received[150];

    printf("Enter the message bits      : ");
    scanf("%s", message);

    printf("Enter the polynomial (g(x)) : ");
    scanf("%s", poly);

    computeCRC(message, poly, crc);

    printf("CRC value (remainder)      : %s\n", crc);
}

```

```

// Form transmitted message
strcpy(transmitted, message);
strcat(transmitted, crc);

printf("Message to be transmitted : %s\n", transmitted);

// Receiver side
printf("\nEnter received message bits (can modify 1 bit to test error): ");
scanf("%s", received);

if(checkReceived(received, poly))
    printf("\n+ ERROR detected in received message!\n");

return 0;
}

```

Output:

Enter the message bits 101101

Enter the polynomial (g(x)) : 1101

Data padded (message + zeros): 101101000

CRC value (remainder) 101

Message to be transmitted 101101101

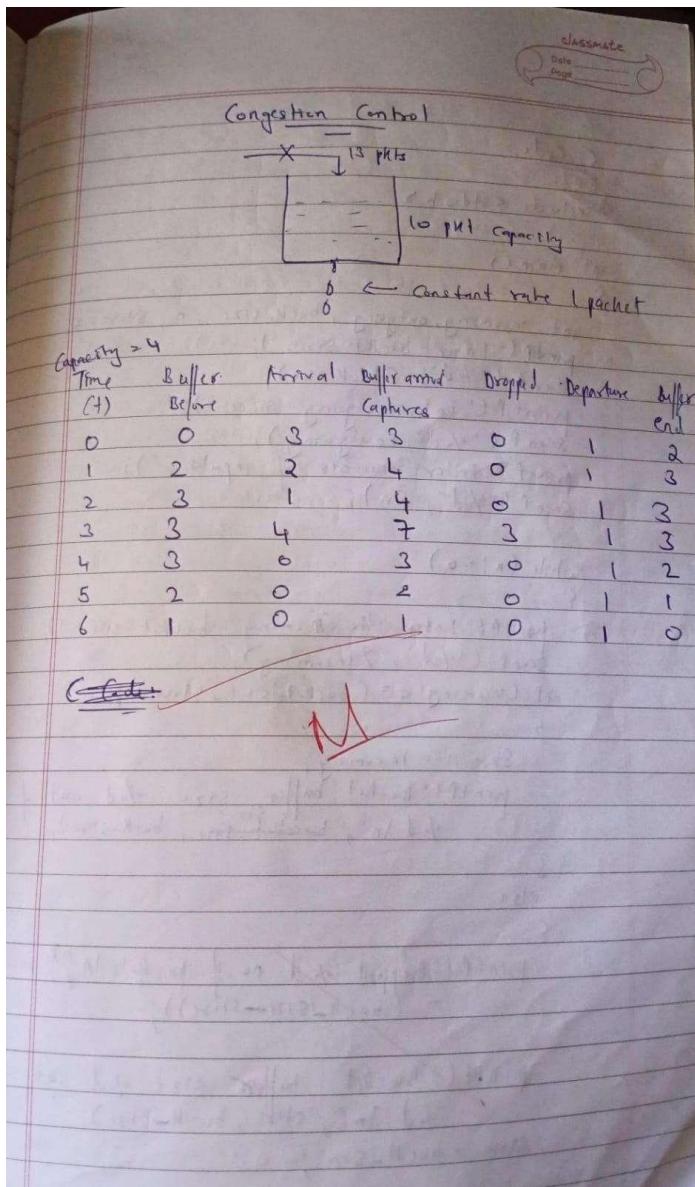
Enter received message bits (can modify 1 bit to test error): 101101101

■ No Error: Message received correctly.

## Program 14

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

Observation:



C-Code:

```
#include <stdio.h>

int main()
{
    int incoming, outgoing, buck-size, n, store=0;
    printf("Enter bucket size: ");
    scanf("%d", &buck-size);
    printf("Enter outgoing size: ");
    scanf("%d", &outgoing);
    printf("Enter number of inputs: ");
    scanf("%d", &n);

    while (n != 0)
    {
        printf("Enter the incoming bucket size: ");
        scanf("%d", &incoming);
        if (incoming <= (buck-size-store))
        {
            store += incoming;
            printf("Bucket buffer size %d out of
                  %d\n", store, buck-size);
        }
        else
        {
            printf("Dropped %d no of jackets in %d, incoming
                  - (%d)\n", incoming - (buck-size-store));
            printf("Bucket buffer size %d out of
                  %d\n", store, buck-size);
            store = buck-size;
        }
    }
}
```

classmate  
Date \_\_\_\_\_  
Page \_\_\_\_\_

Store = Store - outgoing;  
printf ("After outgoing %d packets left out  
of %d in buffer (%d, store, buck-size))  
n--;

3  
3

Output:

Output:

Enter bucket size: 5000  
Enter outgoing rate: 2000  
Enter no of inputs: 2  
Enter the incoming packet size: 3000  
Bucket buffer size 3000 out of 5000  
After outgoing 1000 packets left out of 5000 in buffer  
Enter the incoming packet size: 1000  
Bucket buffer size 2000 out of 5000  
After outgoing 0 packets left out of 5000 in buffer

## Program 15

Aim: 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.

Observation:

The image shows handwritten notes for a TCP socket program. At the top, it says "TCP Socket Programming". Below that, under "Code Server run ①", is the Python code for the server. It starts with "import socket" and defines a "server\_socket" using "socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)". It then binds the socket to "127.0.0.1" on port "8080" and listens for connections. It prints a message when listening and accepts a connection from a client. It then prints a message that a connection has been established and receives data from the client. If the file exists, it reads the file and sends its contents back to the client. If the file does not exist, it sends an error message. Finally, it closes the connection and the server socket. Under "Code Client run ②", the client code is shown, which creates a socket, connects to the server at "127.0.0.1" on port "8080", prompts the user to enter a filename, and then sends that filename to the server.

```
TCP Socket Programming  
Code Server run ①  
import socket  
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
server_socket.bind('127.0.0.1', 8080)  
server_socket.listen(1)  
print("Server is listening on port 8080...")  
conn, addr = server_socket.accept()  
print(f"Connection from {addr} has been established!")  
filename = conn.recv(4096).decode()  
print(f"Client requested file: {filename}")  
  
try:  
    with open(filename, 'r') as file:  
        data = file.read()  
        conn.sendall(data.encode())  
except FileNotFoundError:  
    conn.send(f"Error: File '{filename}' not found.".encode())  
  
conn.close()  
server_socket.close()  
  
Code Client run ②  
import socket  
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
client_socket.connect(('127.0.0.1', 8080))  
filename = input("Enter filename to request: ")
```

classmate  
Date \_\_\_\_\_  
Page \_\_\_\_\_

```
client_socket.sendall(filename.encode())
data = client_socket.recv(4096).decode()
print("Received from server:")
print(data)

client_socket.close()
```

### Output:

Output

After Running Server Script:  
Server is listening on port 8880..  
Connection from ('127.0.0.1', 58544) has been established!  
Client requested file: hello.txt

Client Script:

Enter the filename to request: hello.txt  
Received from server:  
Hello this is Umar  
I am an alien.

\* Need hello.txt file to be in same folder.

N

## Program 15

Aim: 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.

Observation:

Data Page

① Server code:

```
import socket
socket_server = socket.socket(socket.AF_INET,
                               socket.SOCK_DGRAM)
socket_server.bind('localhost', 8080)
print("UDP Server is listening on port 8080")
while True:
    filename, addr = socket_server.recvfrom(4096)
    filename = filename.decode()
    print(f"Client at {addr} requested file: {filename}")
    try:
        with open(filename, 'r') as file:
            data = file.read()
        socket_server.sendto(data.encode(), addr)
    except FileNotFoundError:
        error_msg = f"Error: file '{filename}' not found"
        socket_server.sendto(error_msg.encode(), addr)
```

② client code:

```
import socket
client_socket = socket.socket(socket.AF_INET,
                               socket.SOCK_DGRAM)
server_address = ('localhost', 8080)
filename = input('Enter the file name to request:')
```

```
client-socket.sendto(filename.encode(), serveraddress)  
data, _ = client-socket.recvfrom(4096)
```

```
print("Received from Server :")  
print(data.decode())
```

```
client-socket.close()
```

Output:

Output:

Server :

UDP Server is listening on port 8080...  
Client at ('127.0.0.1', 51343) requested file: hello.txt

Client:

Enter the filename to request: hello.txt  
Received from server:

Hello This is User

I am an alien

N