**B.M.S. COLLEGE OF ENGINEERING BENGALURU**
Autonomous Institute, Affiliated to VTU

Lab Record

# Computer Networks – 23CS5PCCON

*Submitted in partial fulfillment for the 5th Semester Laboratory*

Bachelor of Engineering
in
Computer Science and Engineering

*Submitted by:*

**GHANSHYAM SHARMA**

(1BM23CS100)

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
August 2025-December 2025

# B.M.S. COLLEGE OF ENGINEERING

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING


## *CERTIFICATE*

This is to certify that the Computer Networks (23CS5PCCON) laboratory has been carried out by GHANSHYAM SHARMA (1BM23CS100) during the 5th Semester August 2025-December 2025

Signature of the Faculty Incharge:

**Sarala DV**
**Assistant Professor**

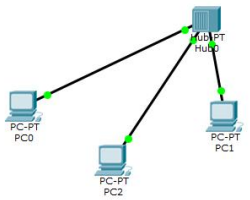
Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

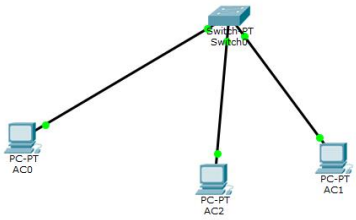
# Table of Contents

# PART - A

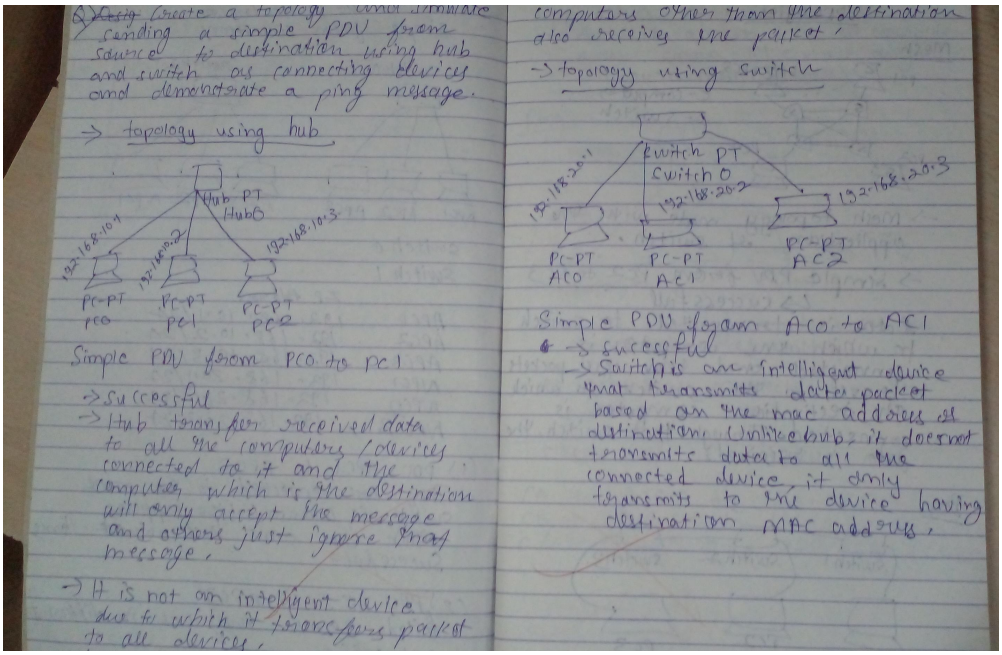
Program 1: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

Network diagram:



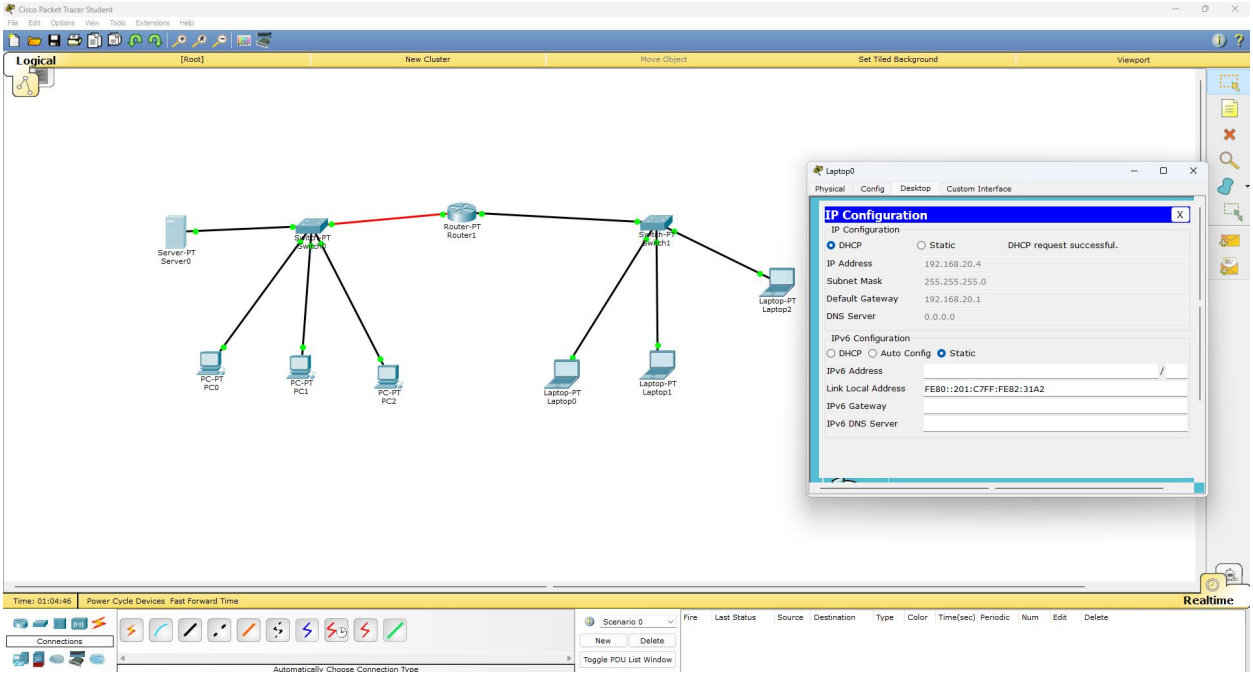




Configuration:

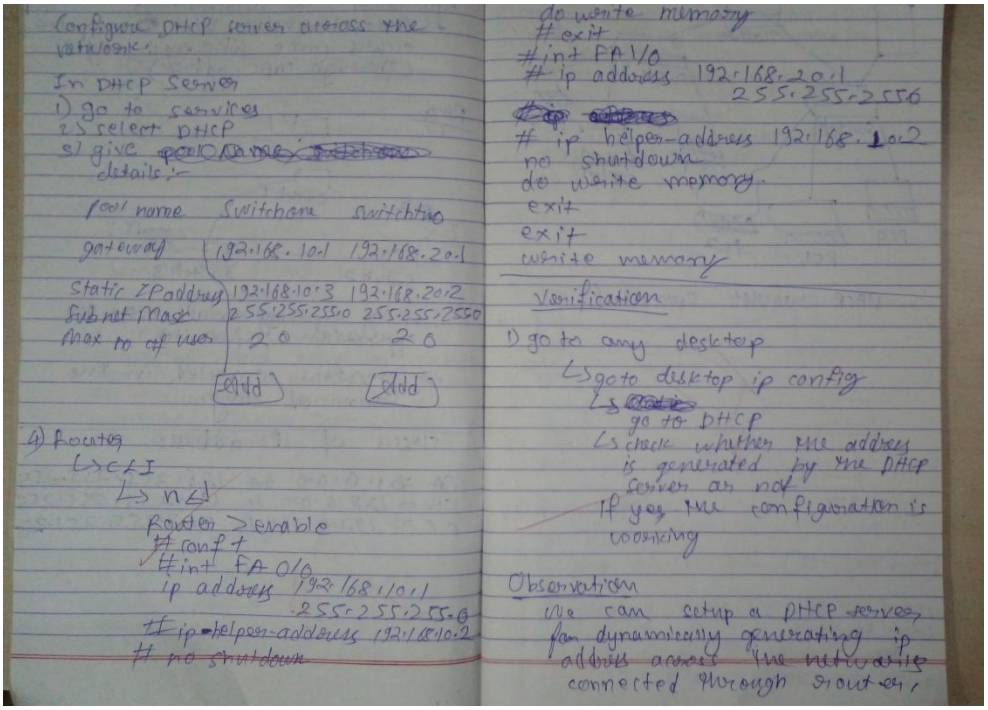


Program 2: Configure DHCP within a LAN and outside LAN.

Network diagram:

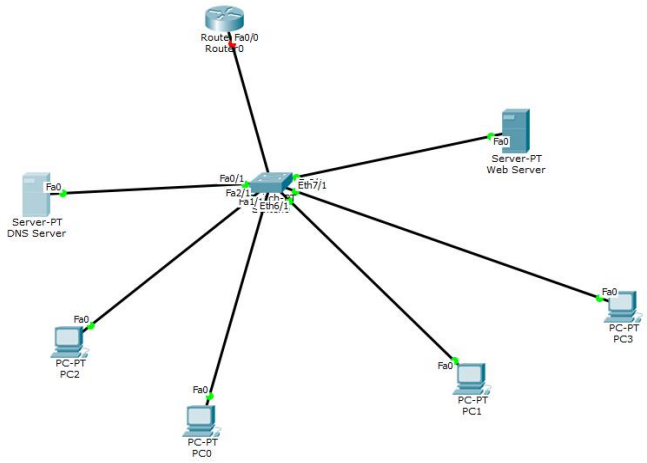



Configuration:



Program 3: Configure Web Server, DNS within a LAN.

Network diagram:

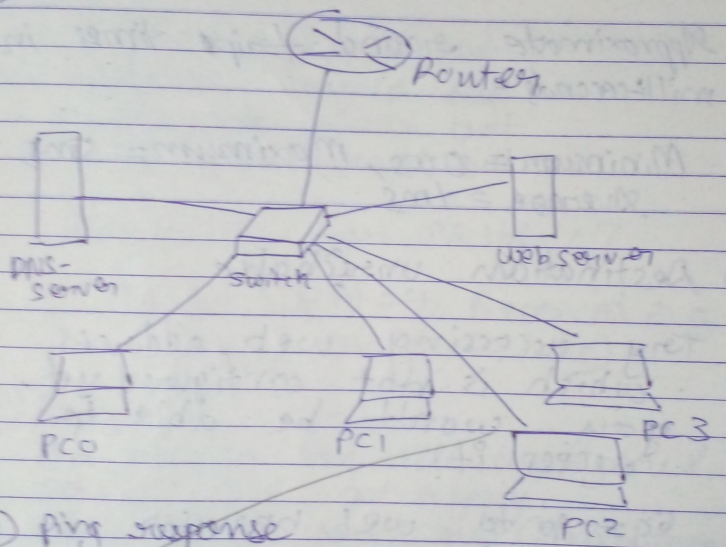



Configuration:

# LAB program 3 10/09

1) Configure webservers, DNS within a LAN
2) Configure IP addresses to routers in packet tracer.
Explore the following messages:
1) Ping response
2) Destination unreachable
3) Request timeout
4) Reply

→



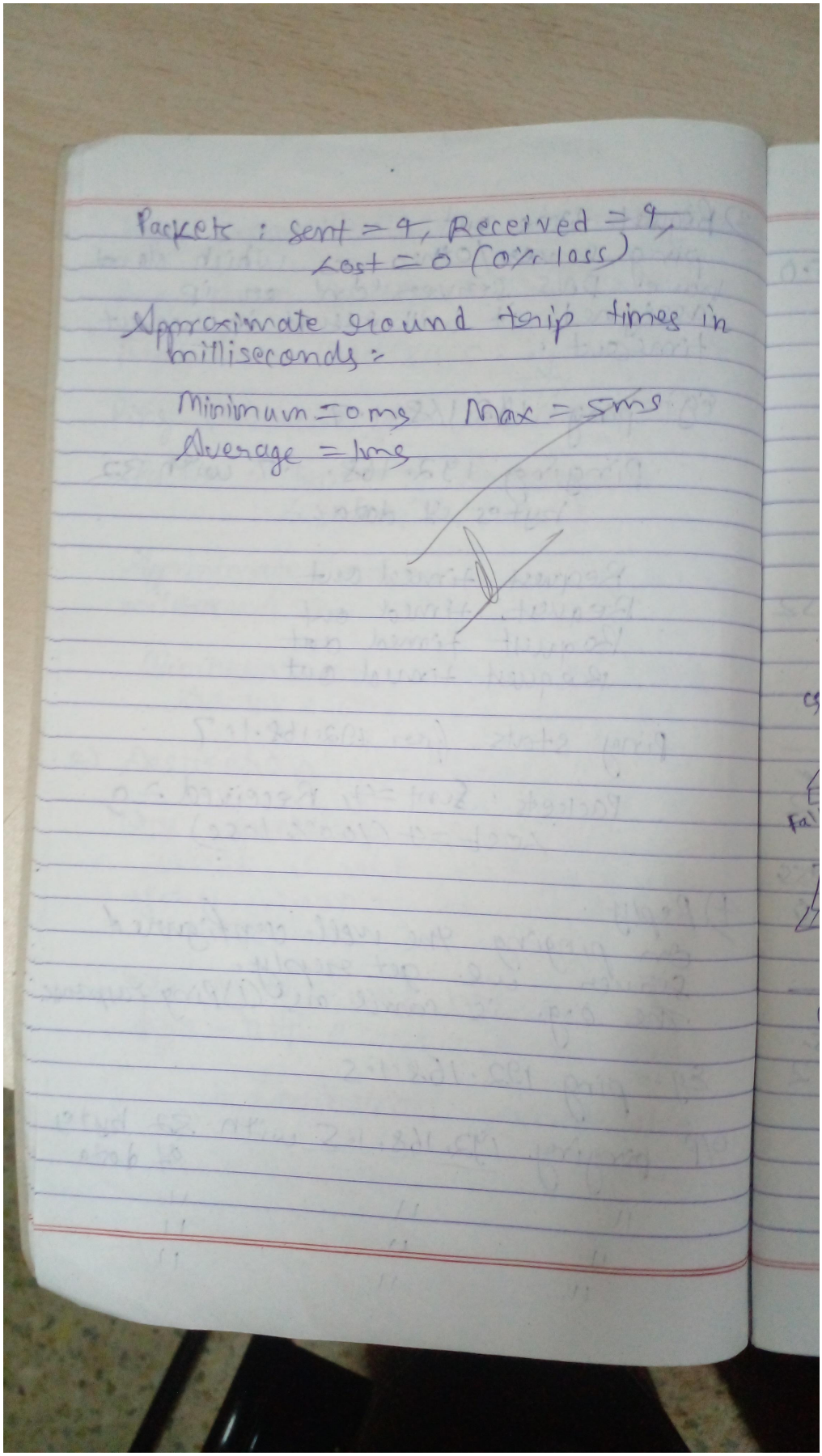

1) ping response
on PC0,
ping 192.168.1.5 (It's DNS server)
o/p:-
pinging 192.168.1.5 with 32 bytes of data:
Reply from 192.168.1.5:
bytes=32 time=0ms TTL=128

Reply from 192.168.1.5 : bytes=32 time=0ms TTL=128
Reply from 192.168.1.5 : bytes=32 time=0ms TTL=128
Reply from 192.168.1.5 : bytes=32 time=0ms TTL=128
Ping statistics for 192.168.1.5 :
Packets : Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milliseconds :
Minimum = 0ms, Maximum = 5ms, Average = 1ms

2) Destination unreachable
Try accessing web address which is not configured yet, you won't be able to access it.
Eg :- go to web browser
type :- http://www.linux.com
-> Destination unreachable

3) Request time out
ping server/other pcs which do not have DNS server 'on' or ip assigned, it will result in request timeout.
Eg :- ping 192.168.1.7
Pinging 192.168.1.7 with 32 bytes of data :
Request timed out
Request timed out
Request timed out
Request timed out
Ping stats for 192.168.1.7
Packets : Sent = 4, Received = 0, Lost = 4 (100% loss)

4) Reply
On pinging the well configured server we get reply.
The e.g. is same as (1) Ping response
Eg :- ping 192.168.1.5
O/P pinging 192.168.1.5 with 32 bytes of data



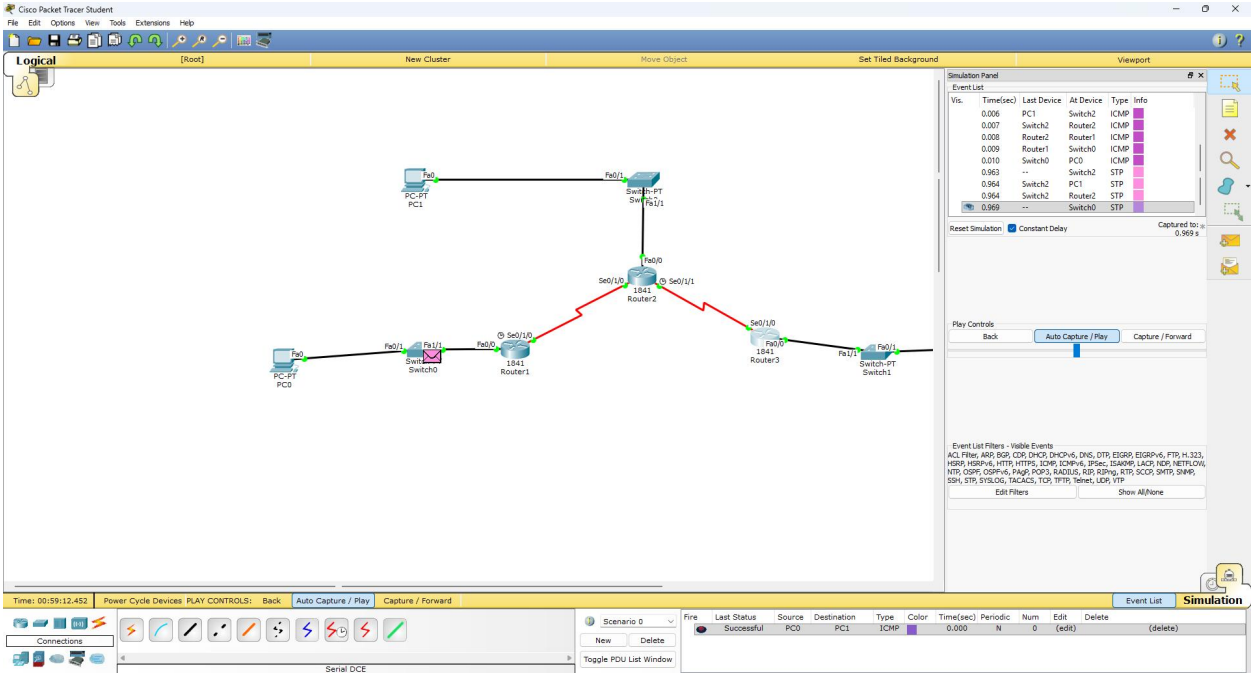
Packets : Sent = 4, Received = 4,
Lost = 0 (0% loss)

Approximate round trip times in milliseconds :

Minimum = 0ms Max = 5ms
Average = 1ms

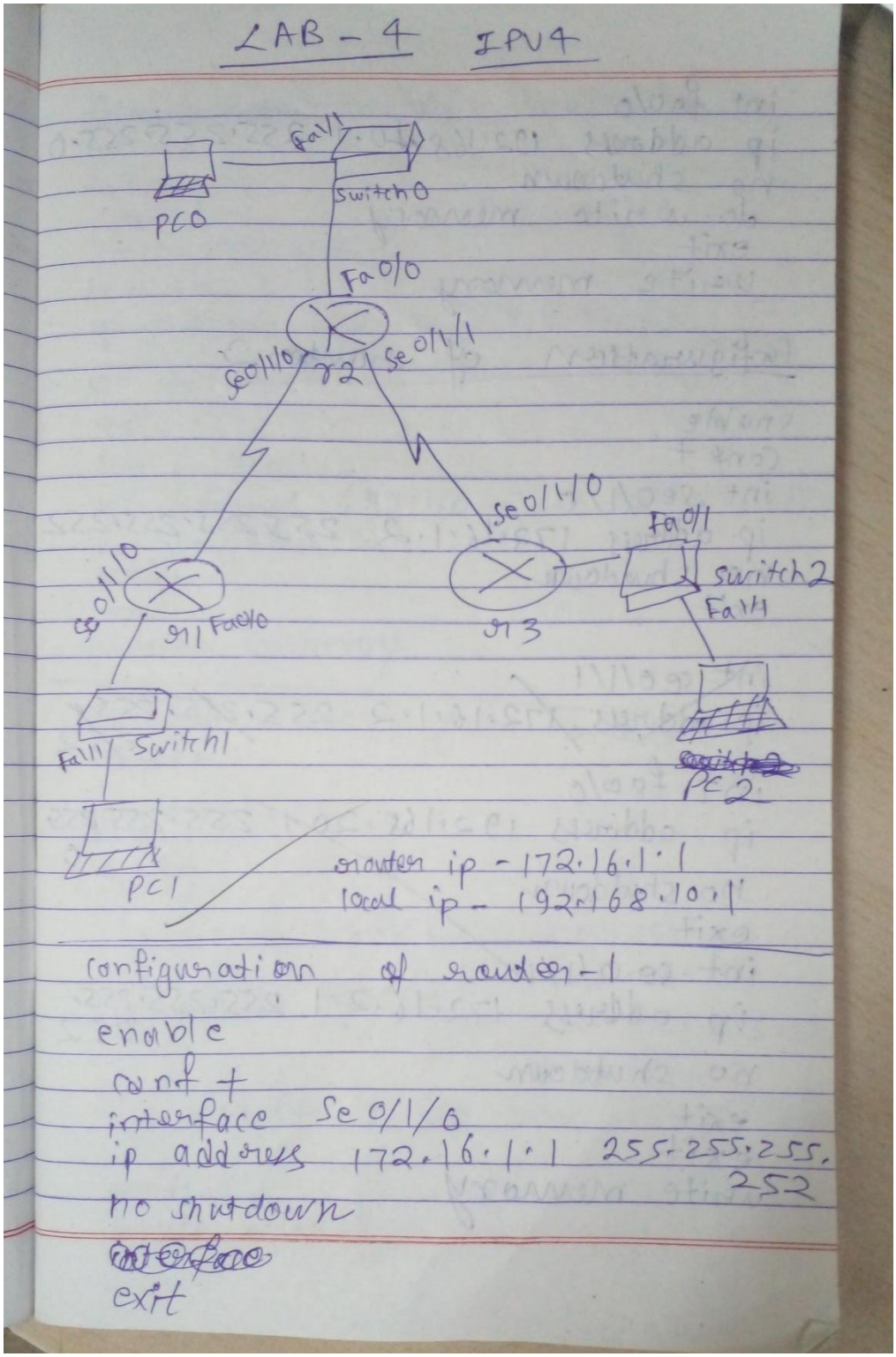
Program 4: Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.

Network diagram:




Configuration:

# LAB - 4 IPV4




configuration of router-1

enable
conf t
interface Se 0/1/0
ip address 172.16.1.1 255.255.255.252
no shutdown
exit

int fa0/0
ip address 192.168.10.1 255.255.255.0
no shutdown
do write memory
exit
write memory

Configuration of router 2

enable
conf t
int se0/1/0
ip address 172.16.1.2 255255.255.252
no shutdown
exit

int se0/1/1
ip address 172.16.1.2 255.255.255.252
int fa0/0
ip address 192.168.20.1 255.255.255.0
no shutdown
exit
int se0/1/1
ip address 172.16.2.1 255.255.255.252
no shutdown
exit
exit
write memory

Configuration of router 3

enable
conf t
hostname r3
int se0/1/1
ip address 172.16.2.2 255.255.255.252
no shutdown
exit
int fa 0/0
ip address 192.160.50.1 255.255.255.0
no shutdown
exit
exit
write memory

Configuration for routing table:-

router1:-
enable
conf t
ip route 192.168.20.0 255.255.255.0 172.16.1.2
ip route 172.16.2.0 255.255.255.252 172.16.1.2
ip route 192.168.30.0 255.255.255.0 172.16.1.2
exit

To check routing table type:-

show ip route

Router 2 :-

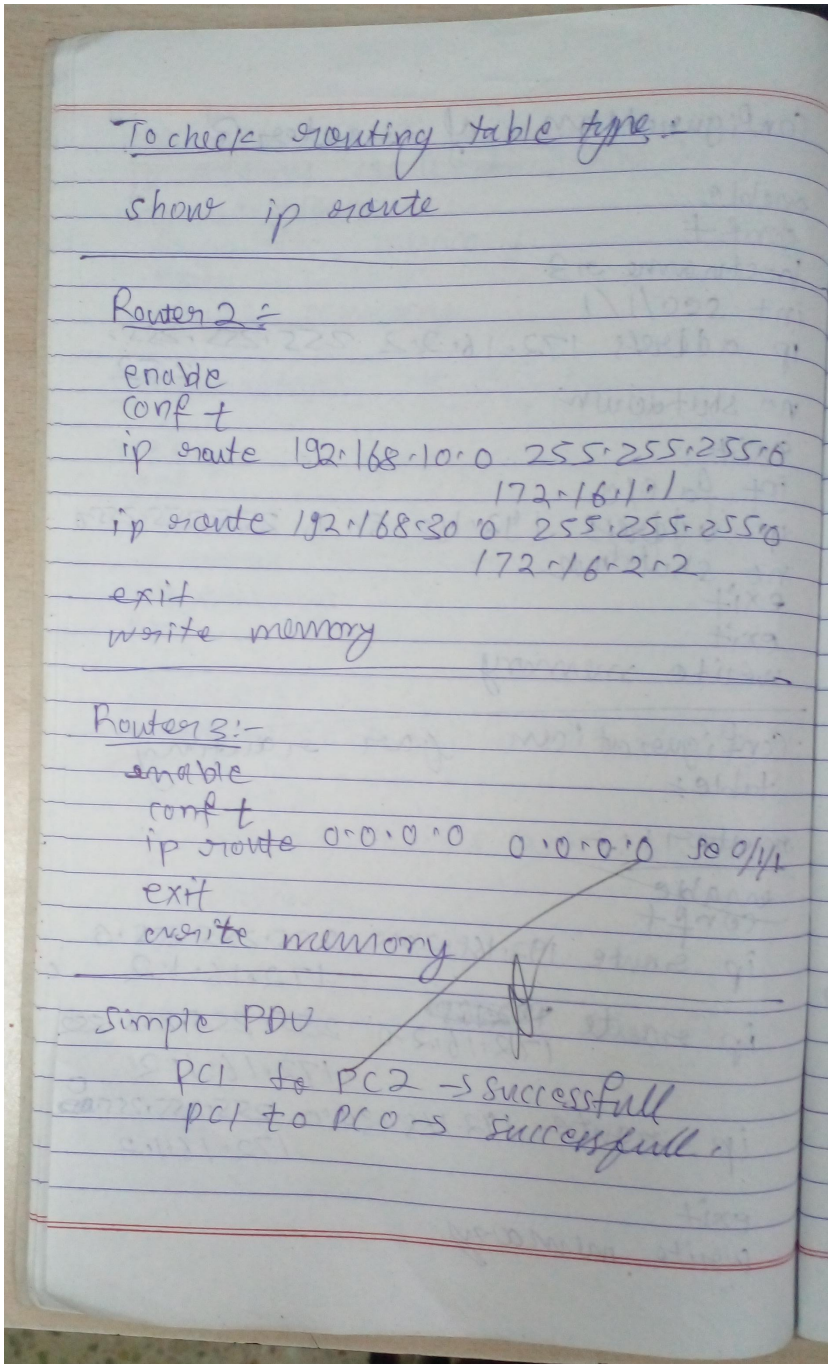
enable
conf t
ip route 192.168.10.0 255.255.255.0
172.16.1.1
ip route 192.168.30.0 255.255.255.0
172.16.2.2
exit
write memory

Router 3:-

enable
conf t
ip route 0.0.0.0 0.0.0.0 se 0/1/1
exit
write memory

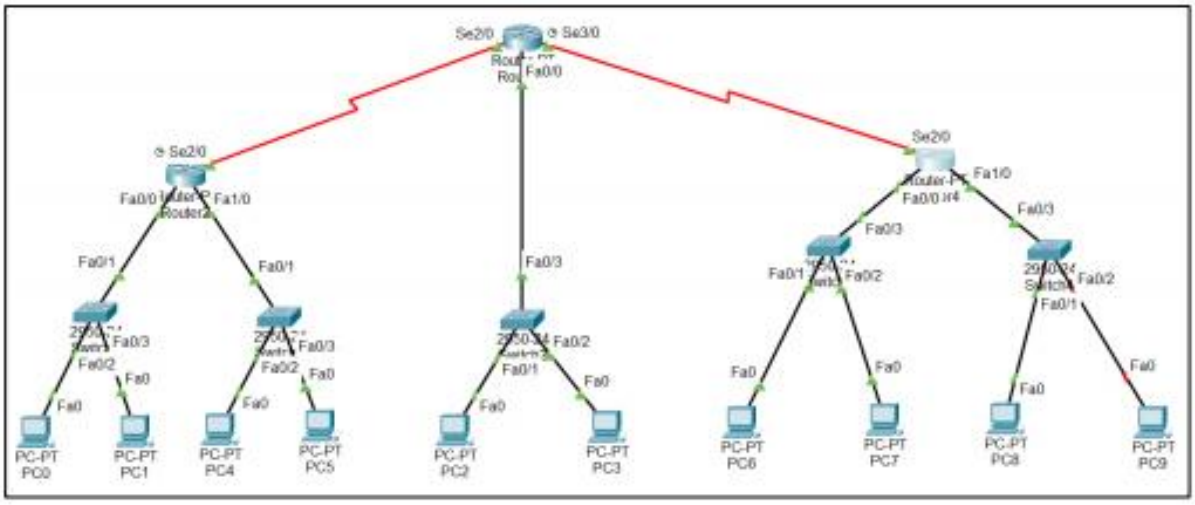
Simple PDU

PC1 to PC2 -> successfull
PC1 to PC0 -> successfull.

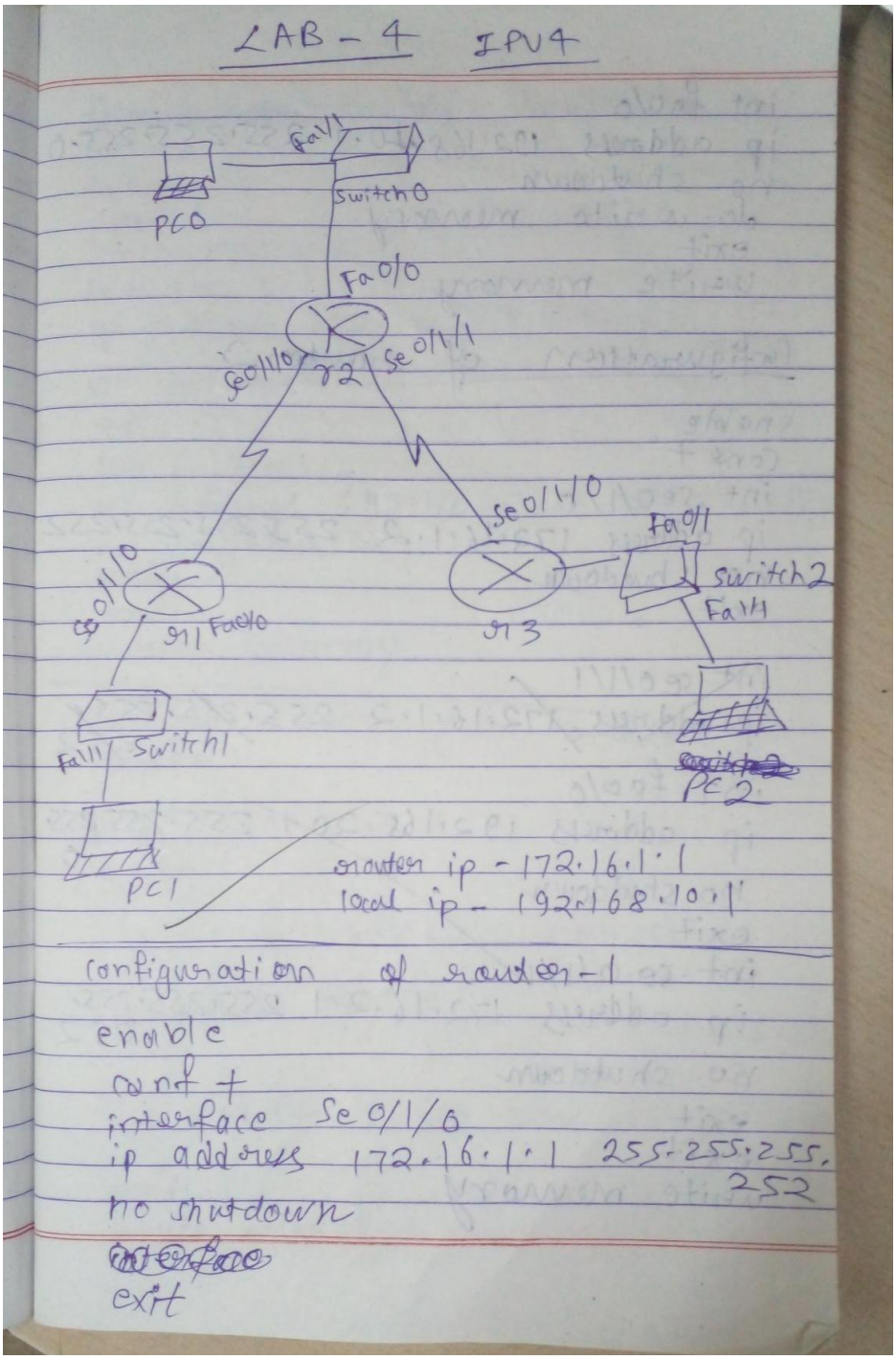


Program 5: Configure default route, static route to the Router.

Network diagram:



Configuration:



int fa0/0
ip address 192.168.10.1 255.255.255.0
no shutdown
do write memory
exit
write memory

Configuration of router 2

enable
conf t
int se0/1/0
ip address 172.16.1.2 255255.255.252
no shutdown
exit

int se0/1/1
ip address 172.16.1.2 255.255.255.252
int fa0/0
ip address 192.168.20.1 255.255.255.0
no shutdown
exit
int se0/1/1
ip address 172.16.2.1 255.255.255.252
no shutdown
exit
exit
write memory

Configuration of router 3

enable
conf t
hostname r3
int se0/1/1
ip address 172.16.2.2 255.255.255.252
no shutdown
exit
int fa 0/0
ip address 192.160.50.1 255.255.255.0
no shutdown
exit
exit
write memory

Configuration for routing table:-

router1:-
enable
conf t
ip route 192.168.20.0 255.255.255.0 172.16.1.2
ip route 172.16.2.0 255.255.255.252 172.16.1.2
ip route 192.168.30.0 255.255.255.0 172.16.1.2
exit

To check routing table type:-

show ip route

Router 2 :-

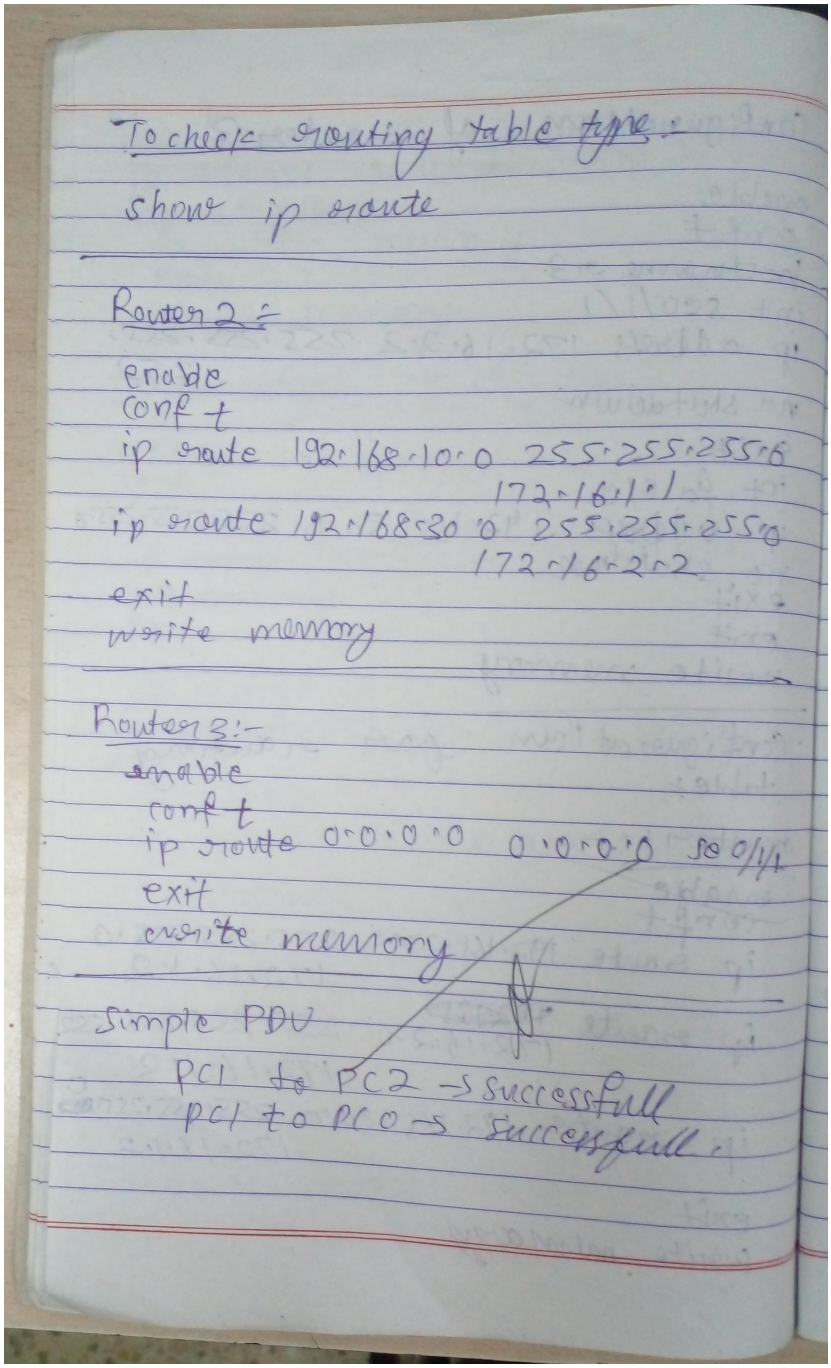
enable
conf t
ip route 192.168.10.0 255.255.255.0
172.16.1.1
ip route 192.168.30.0 255.255.255.0
172.16.2.2
exit
write memory

Router 3:-

enable
conf t
ip route 0.0.0.0 0.0.0.0 se 0/1/1
exit
write memory

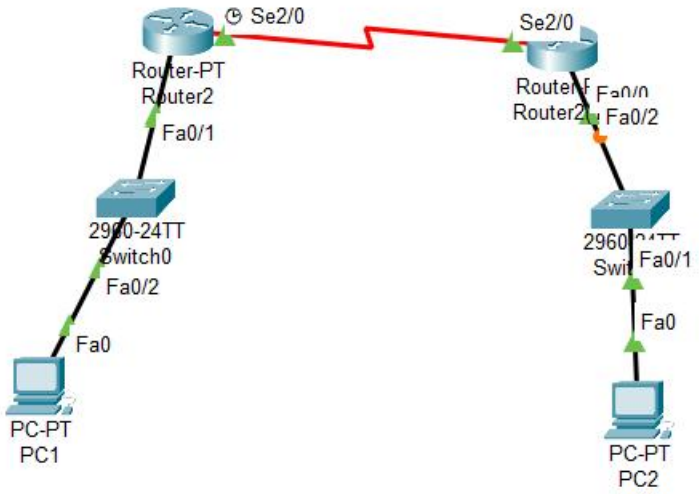
Simple PDU

PC1 to PC2 -> successfull
PC1 to PC0 -> successfull.



Program 6: Configure RIP routing Protocol in Routers.

Network diagram:

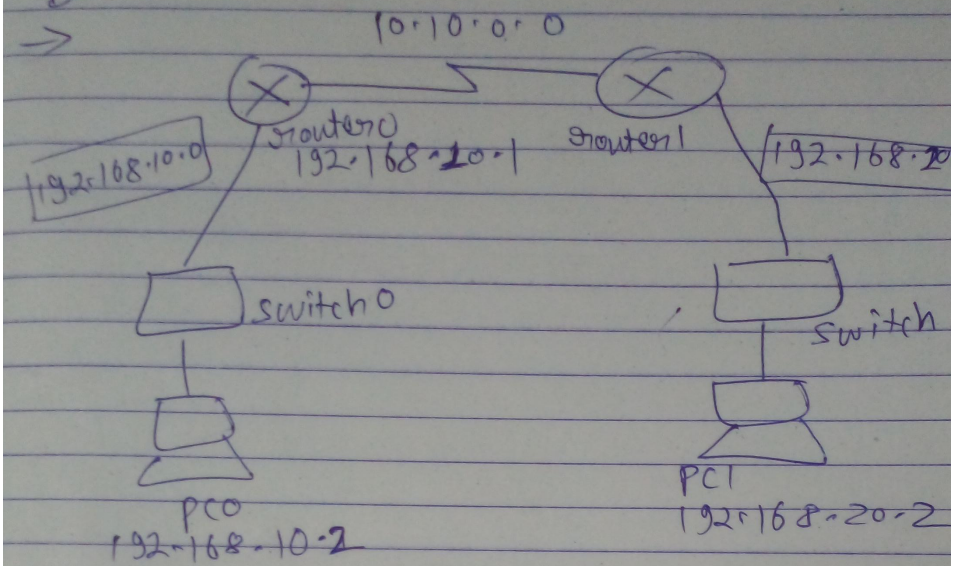



Configuration:

# LAB-10

Configure RIP routing protocol in routers to transfer packet from node A to node B.

→




RIP table for R0 observation

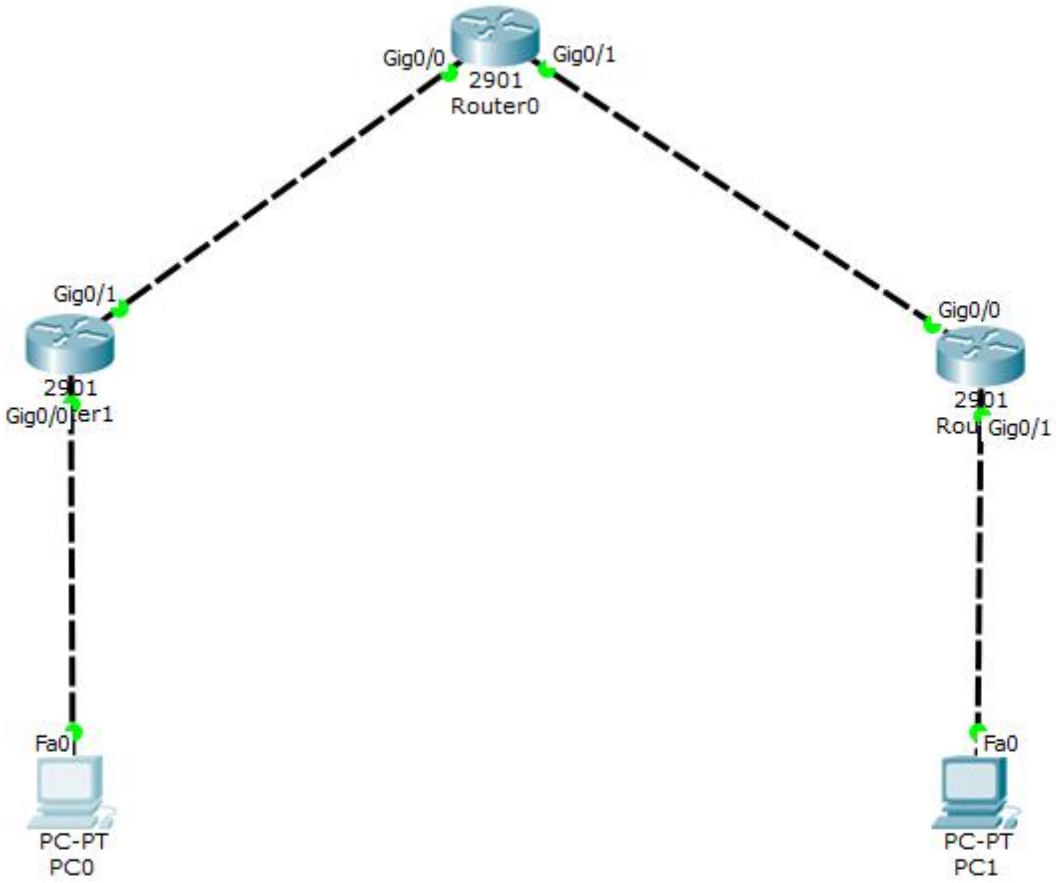
| Network |
| --- |
| 192.168.10.0 |
| 10.0.0.0 |

RIP table for R1 observation

| Network |
| --- |
| 192.168.20.0 |
| 10.0.0.0 |

Program 7: Configure OSPF routing protocol.

Network diagram:




Configuration:

Router CLI commands - R0

1) exit
2) exit
3) conf t
4) ~~ospf 1~~ router ospf 1
5) network 192.168.55.0
0.0.0.255 area 0
6) network 176.16.0.0
0.0.25.255 area 0
7) exit

for R1

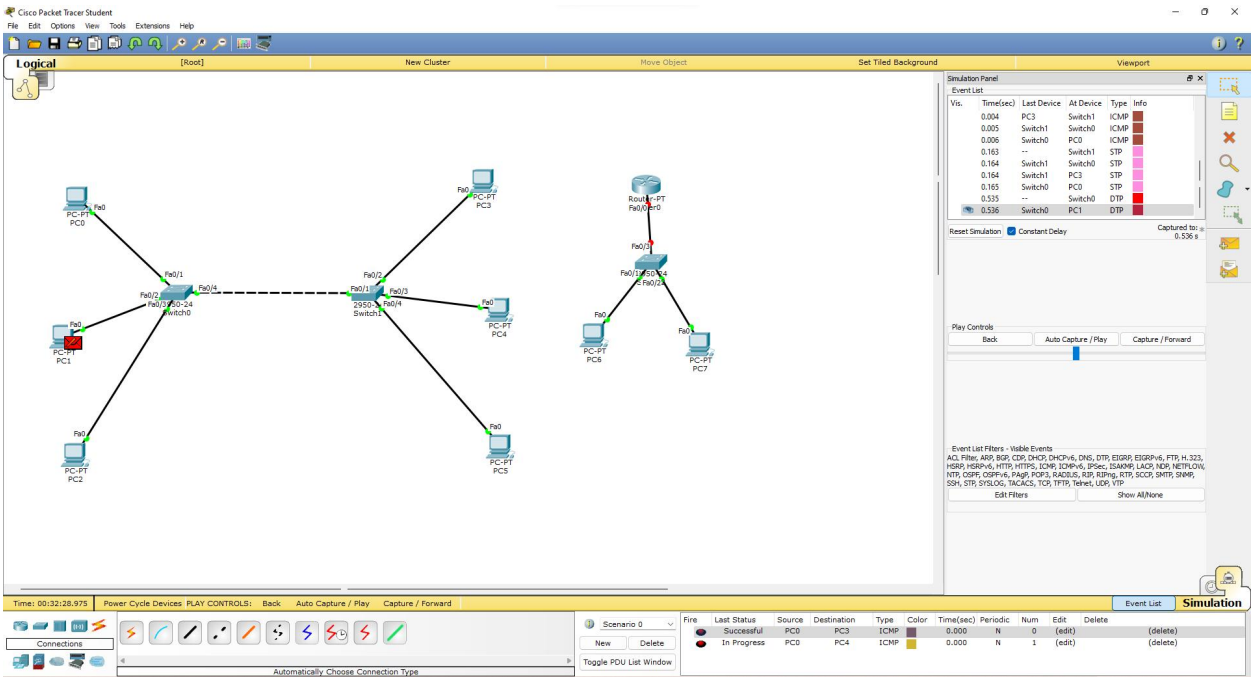
1) conf t
2) router ospf 1
3) network 192.168.55.0 0.0.0.255
area 0
4) network 192.168.44.0 0.0.0.255
area 0
5) exit

for R2

1) conf t
2) router ospf 1
3) network 176.16.0.0 0.0.255.255
area 0
4) network 10.0.0.0 0.255.255.255
area 0
5) exit.

Program 8: To construct a VLAN and make the PC's communicate among a VLAN.

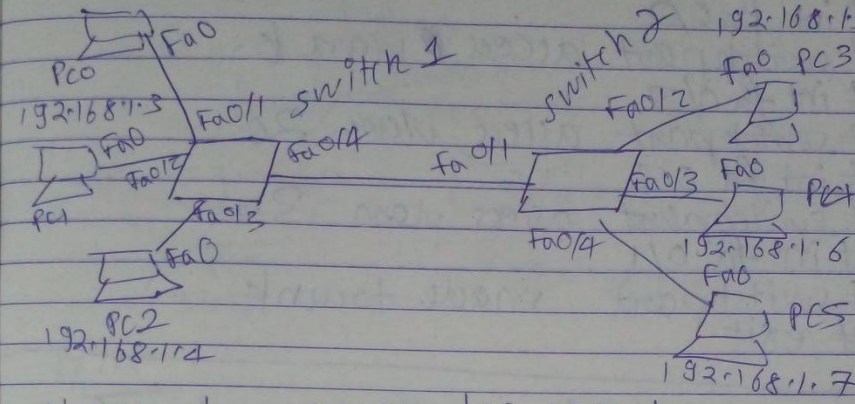
Network diagram:




Configuration:

# Lab - Program - 7

To construct a VLAN and make the PCs communicating among a VLAN 192.168.1.2




| VLAN 10 | VLAN 20 | VLAN 30 |
|---|---|---|
| fa0/1 PC0 | fa0/3 PC1 | fa0/3 PC2 |
| fa0/2 PC3 | fa0/1 PC4 | fa0/4 PC5 |

Virtual LAN configuration

Commands for first switch

>enable
# config t
# int fa 0/1
# switchport access vlan 10
# int fa 0/2
# switchport access vlan 20
# int fa 0/3
# switchport access vlan 30
# int fa 0/4
# switchport mode trunk
# exit

## Commands for second switch

```
# enable
# conf t
# int fa 0/2
# switchport access vlan 10
# int fa 0/3
# switchport access vlan 20
# int fa 0/4
# switchport access vlan 30
# int fa 0/1
# switchport mode trunk
# exit
```

---

Now, trying to ping the computer of different VLAN, we receive an error message.

from computer of VLAN 1
viz: 192.168.1.2.
Let's ping 192.168.1.7

cmd:-

ping 192.168.1.7

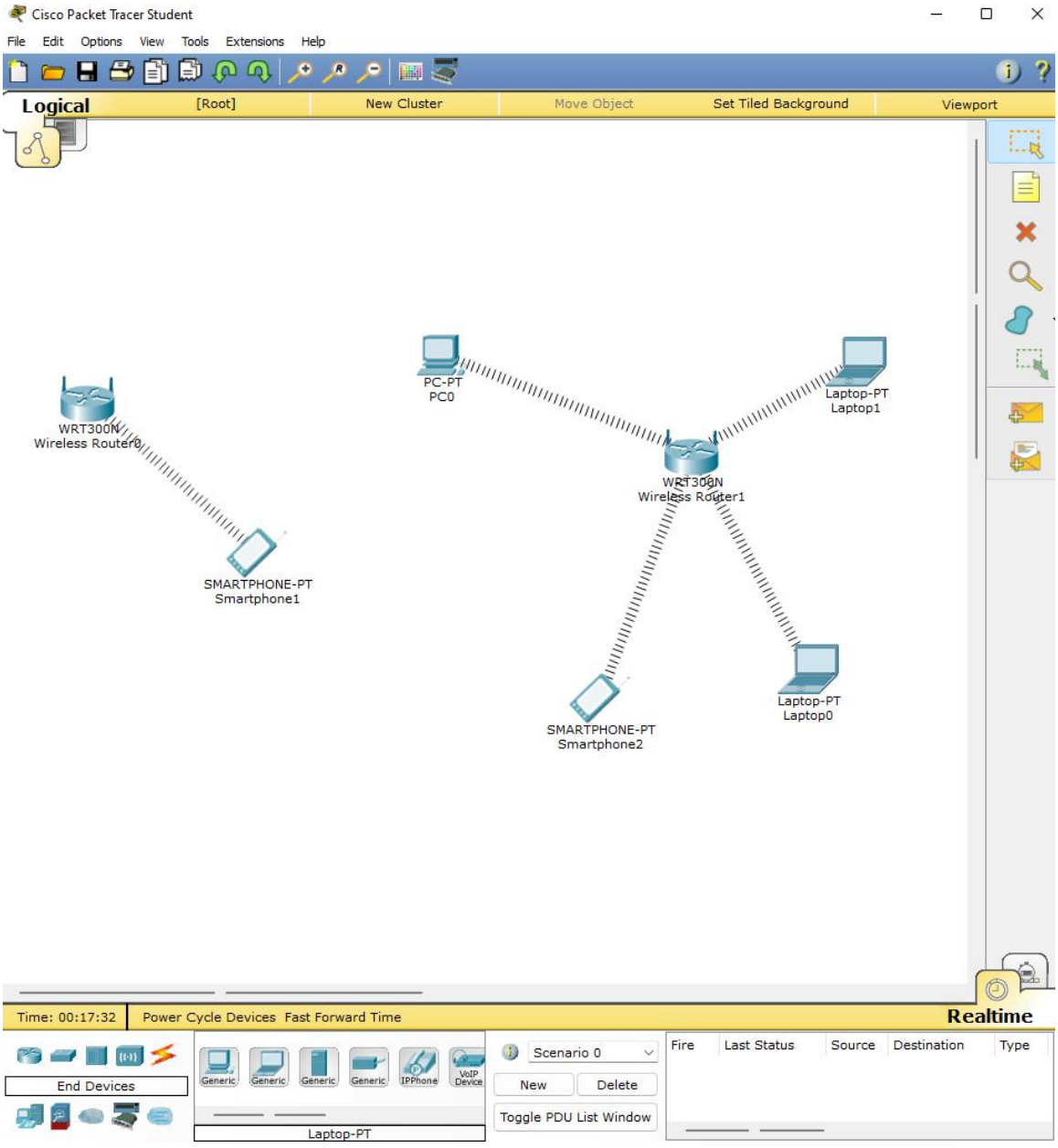
Pinging 192.168.1.7 with 32 bytes of data:

request timed out
request timed out
" " "
" " "

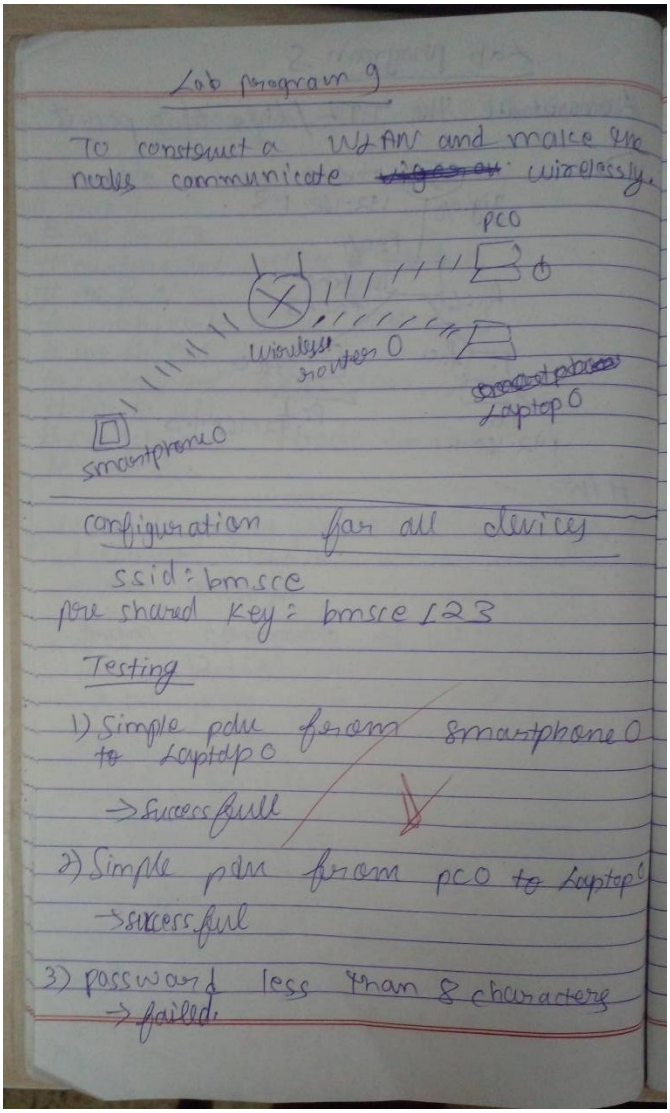
Packet: Sent = 4, received = 0, Lost = 4
(100% loss)

Program 9: To construct a WLAN and make the nodes communicate wirelessly.

Network diagram:

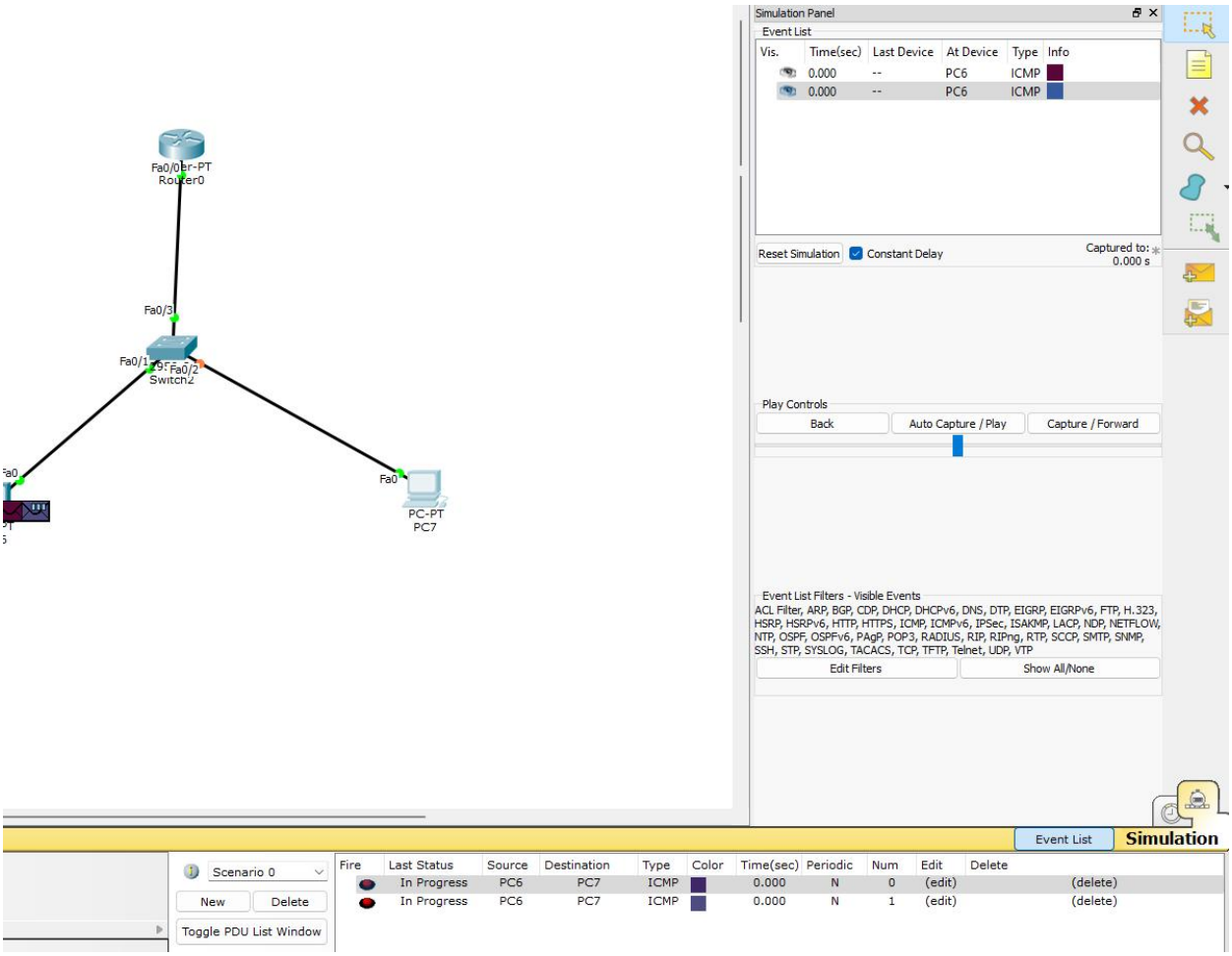


Configuration:

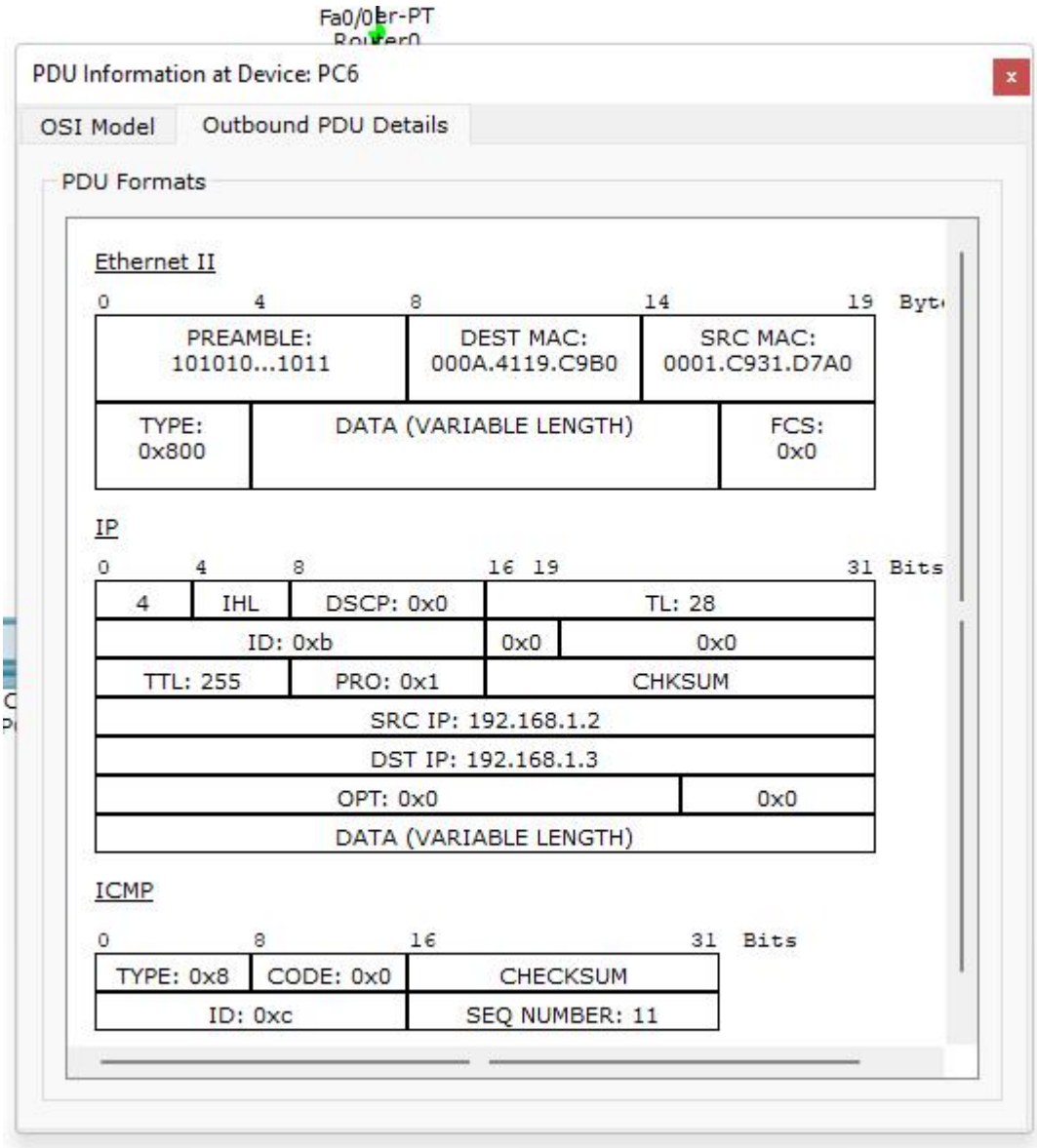


Program 10: Demonstrate the TTL/ Life of a Packet.

Network diagram:

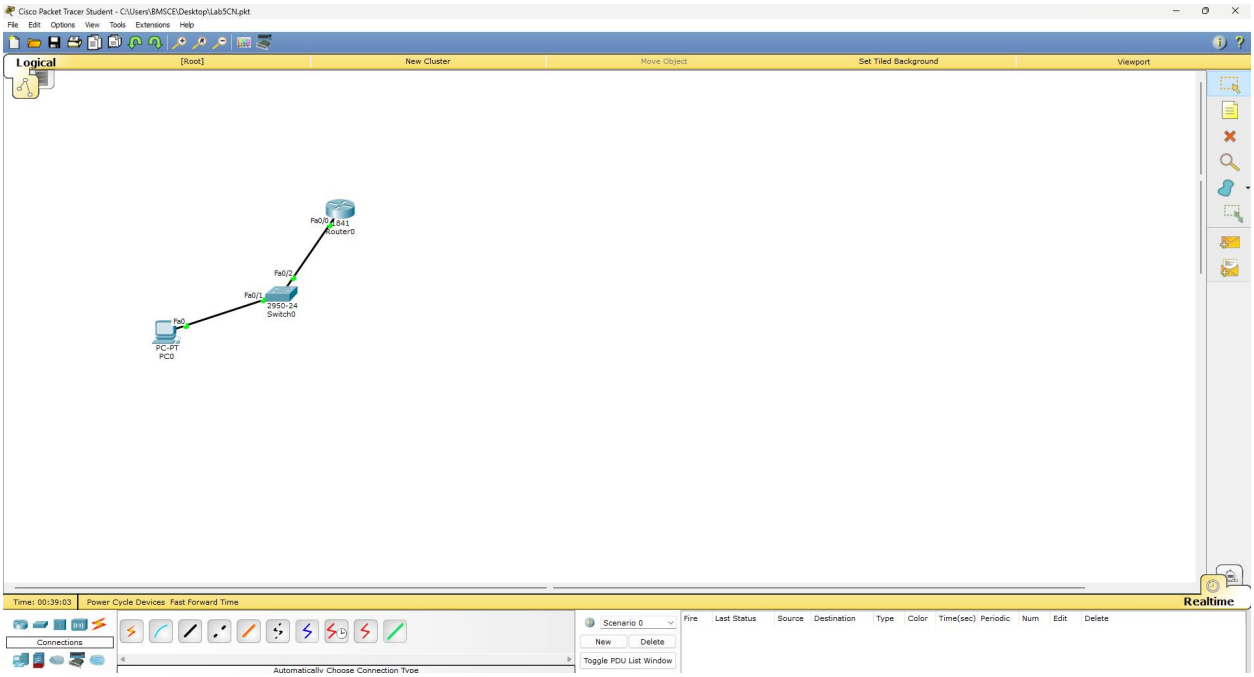


Configuration:



Program 11: To understand the operation of TELNET by accessing the router in server room from a PC in IT office.

Network diagram:




Configuration:

Lab Program-5 Telnet 8/10/025

Configure telnet to access router remotely

-> Telnet is used to access remote server.
-> It is a simple cmd tool that runs on your computer and it allow you to send command remotely to a server and administrator.
-> Telnet is also used to manage other devices like router, switches to check if ports are open or close on the servers.

vty -> virtual teletype

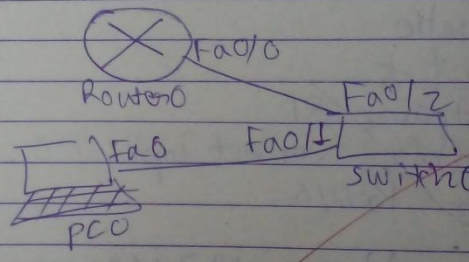
Commands written in Routers

1) enable
2) conf t
3) hostname R1
4) enable secret cp
5) int fa0/0
6) ip address 192.168.1.1 255.255.255.0
7) no shutdown
8) line vty 0 5
9) login
10) password tp
11) exit
12) exit
13) wr
14) show ip interface brief
15) —— x ——

CMD of PC commands

1) ping 192.168.1.1
2) telnet 192.168.1.1
3) User Access verification
   Password : tp
4) show ip interface brief
5) conf t
6) int Fa0/1
7) ip address 192.168.1.2 255.255.255.0
8) exit
9) exit
10) show ip interface brief
11) exit

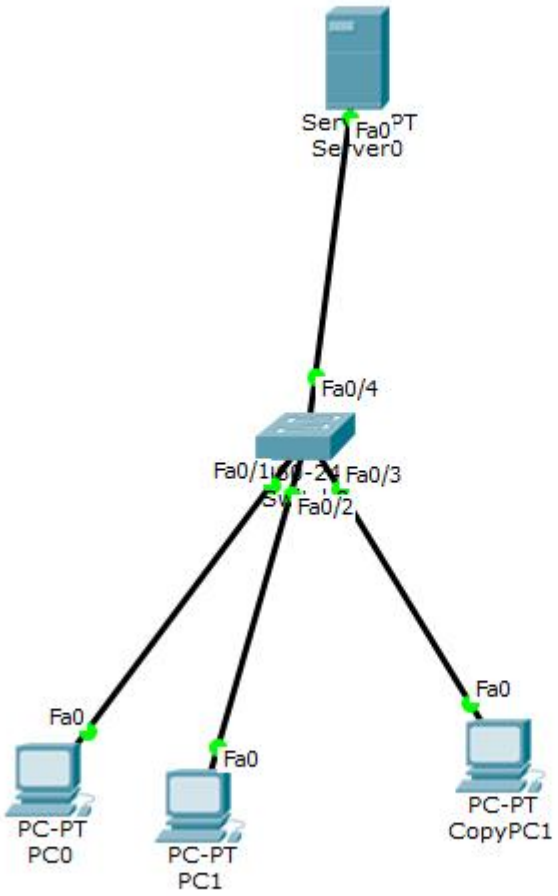
Diagram

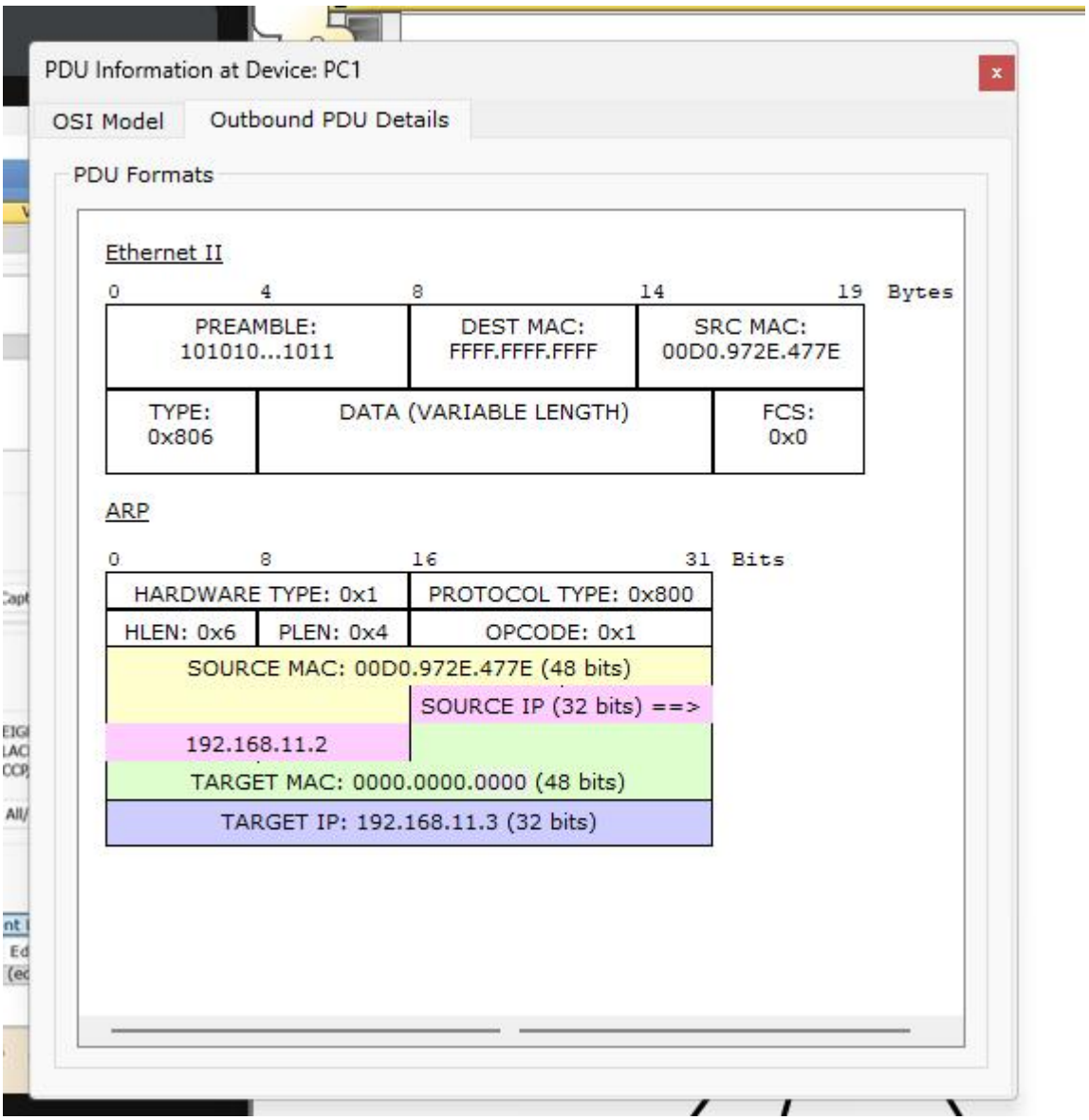



Outcome :- We can access the networking device remotely through PC using the Telnet protocol. The Telnet protocol can be used for configuring router switches and other networking devices.

Program 12: To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP).

Network diagram:






Configuration:

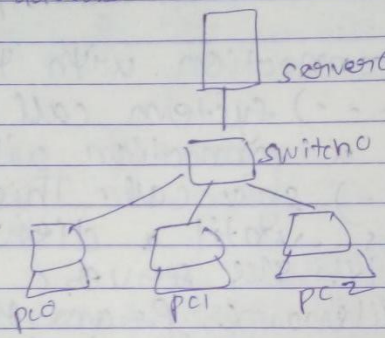
# ARP

AB-10

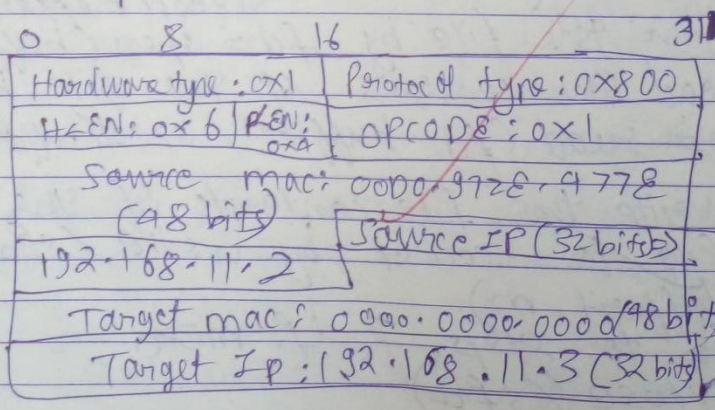
ARP stands for Address Resolution Protocol.

ARP is used to map an ip address to a mac address

ARP is used to get data link layer address, mac add. with the help of ip address.




ARP packet observation



# PART - B

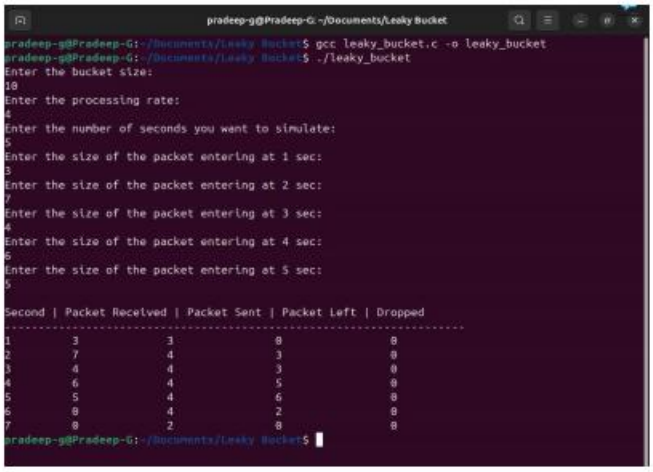
Program 1: Write a program for congestion control using Leaky bucket algorithm.

Code:

```
#include <stdio.h>
int min(int x, int y) {
if (x < y)
return x;
else
return y;
}
int main() {
int drop = 0, mini, nsec, cap, count = 0, i, inp[25],
process;
printf("Enter the bucket size:\n");
scanf("%d", ∩);
printf("Enter the processing rate:\n");
scanf("%d", &process);
printf("Enter the number of seconds you want to
simulate:\n");
scanf("%d", &nsec);
for (i = 0; i < nsec; i++) {
printf("Enter the size of the packet entering at %d
sec:\n", i + 1);32
scanf("%d", &inp[i]);
}
printf("\nSecond | Packet Received | Packet Sent | Packet
Left | Dropped\n");
printf("-------------------------------------------------
---------------\n");
for (i = 0; i < nsec; i++) {
count += inp[i];
if (count > cap) {
drop = count - cap;
count = cap;
}
printf("%d\t %d\t\t", i + 1, inp[i]);
mini = min(count, process);
printf("%d\t\t", mini);
count = count - mini;
printf("%d\t\t %d\n", count, drop);
drop = 0;
}
// Remaining packets after time ends
for (; count != 0; i++) {
if (count > cap) {33
drop = count - cap;
```

```
count = cap;
}
printf("%d\t 0\t\t", i + 1);
mini = min(count, process);
printf("%d\t\t", mini);
count = count - mini;
printf("%d\t\t %d\n", count, drop);
drop = 0;
}
return 0;
}
```

Output:



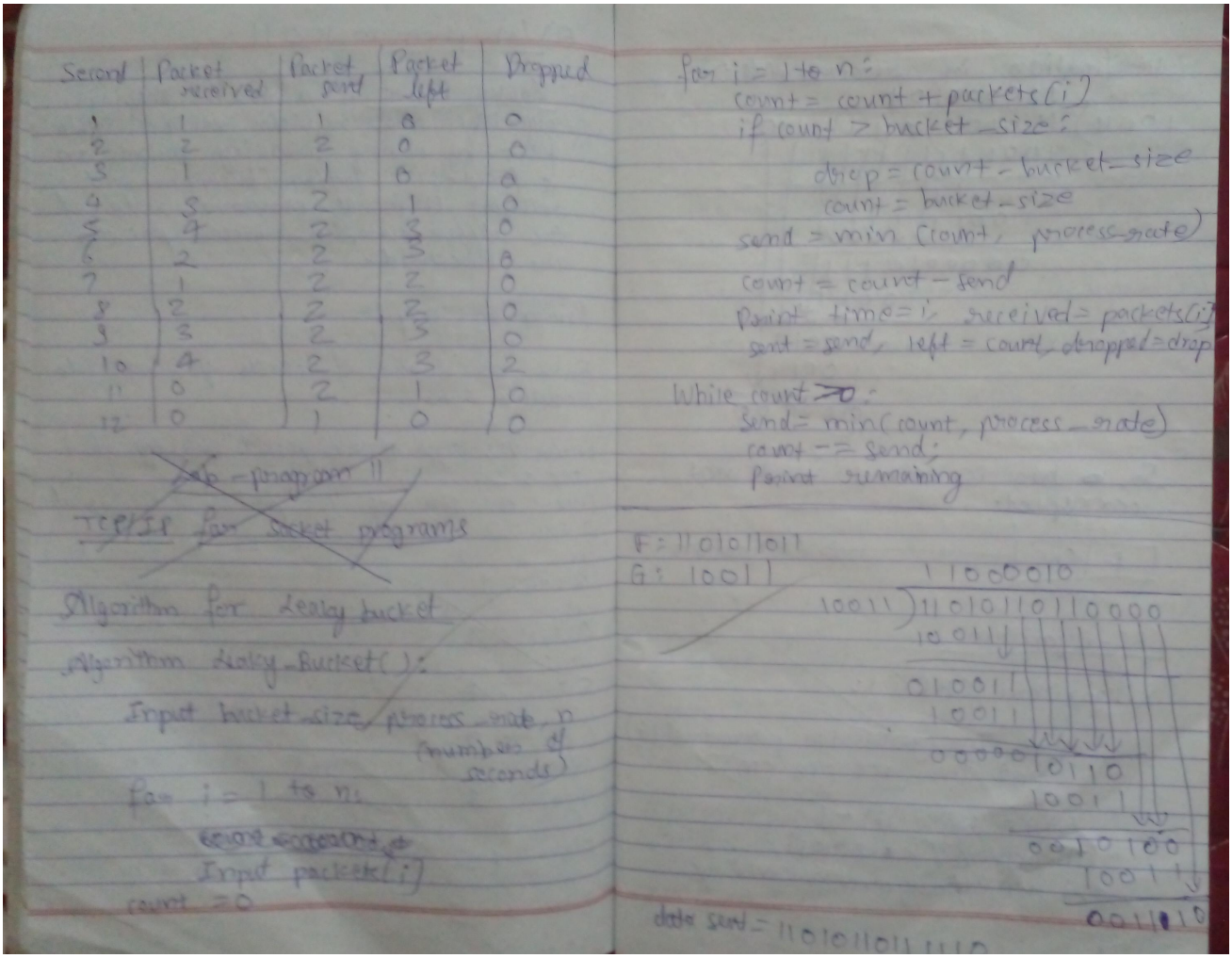
Q) Lab program 10 & 11

Leaky Bucket is a traffic shaping algorithm that regulates network data flow by storing incoming packets in a fixed-size buffer (bucket) and releasing them at constant rate; if the buffer overflows, excess packets are discarded.

Program o/p:-

Enter the number of seconds you want to simulate : 10

Enter the size of the packet entering at 1 sec : 1
Enter the size of the packet entering at 2 sec : 2
" " " " 3 sec : 1
" " " " 4 sec : 3
" " " " 5 sec : 4
" " " " 6 sec : 2
" " " " 7 sec : 1
" " " " 8 sec : 2
" " " " 9 sec : 3
" " " " 10 sec : 4




Program 2: 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:

```
#include <stdio.h>


int min(int x, int y) {
   return (x < y) ? x : y;
}
```

```
int main() {
    int drop = 0, mini, nsec, cap, count = 0, i, inp[25], process;

    printf("Enter the bucket size (capacity): ");
    scanf("%d", &cap);

    printf("Enter the processing rate (packets per second): ");
    scanf("%d", &process);

    printf("Enter the number of seconds you want to simulate: ");
    scanf("%d", &nsec);

    for (i = 0; i < nsec; i++) {
        printf("Enter the number of packets entering at %d sec: ", i + 1);
        scanf("%d", &inp[i]);
    }

    printf("\nSecond | Packet Received | Packet Sent | Packet Left | Dropped\n");
    printf("-----------------------------------------------------------------\n");

    for (i = 0; i < nsec; i++) {
        count += inp[i];
        if (count > cap) {
            drop = count - cap;
            count = cap;
        }
```

```
    printf("%6d | %15d |", i + 1, inp[i]);

    mini = min(count, process);
    printf(" %11d |", mini);

    count -= mini;
    printf(" %12d | %7d\n", count, drop);
    drop = 0;
}

// Continue processing remaining packets
while (count != 0) {
    if (count > cap) {
        drop = count - cap;
        count = cap;
    }
    printf("%6d | %15d |", ++i, 0);

    mini = min(count, process);
    printf(" %11d |", mini);

    count -= mini;
    printf(" %12d | %7d\n", count, drop);
    drop = 0;
}

return 0;
```

}

Output:

```
Enter the bucket size (capacity): 5
Enter the processing rate (packets per second): 2
Enter the number of seconds you want to simulate: 3
Enter the number of packets entering at 1 sec: 5
Enter the number of packets entering at 2 sec: 4
Enter the number of packets entering at 3 sec: 3

Second | Packet Received | Packet Sent | Packet Left | Dropped
---------------------------------------------------------------
     1 |               5 |           2 |           3 |       0
     2 |               4 |           2 |           3 |       2
     3 |               3 |           2 |           3 |       1
     4 |               0 |           2 |           1 |       0
     5 |               0 |           1 |           0 |       0
```

Q) Lab program 10 & 11

Leaky Bucket is a traffic shaping algorithm that regulates network data flow by storing incoming packets in a fixed-size buffer (bucket) and releasing them at constant rate; if the buffer overflows, excess packets are discarded.

Program o/p:-

Enter the numbers of seconds you want to simulate : 10

Enter the size of the packet entering at 1 sec : 1

Enter the size of the packet entering at 2 sec : 2

3 sec : 1

4 sec : 3

5 sec : 4

6 sec : 2

7 sec : 1

8 sec : 2

9 sec : 3

10 sec : 4

| Second | Packet received | Packet sent | Packet left | Dropped |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 2 | 2 | 2 | 0 | 0 |
| 3 | 1 | 1 | 0 | 0 |
| 4 | 3 | 2 | 1 | 0 |
| 5 | 4 | 2 | 3 | 0 |
| 6 | 2 | 2 | 3 | 0 |
| 7 | 1 | 2 | 2 | 0 |
| 8 | 2 | 2 | 2 | 0 |
| 9 | 3 | 2 | 3 | 0 |
| 10 | 4 | 2 | 3 | 2 |
| 11 | 0 | 2 | 1 | 0 |
| 12 | 0 | 1 | 0 | 0 |

~~Lab - program 11~~

~~TCP/IP for socket programs~~

Algorithm for Leaky bucket

```
Algorithm leaky_Bucket():
    Input bucket_size, process_rate, n (number of seconds)
    for i = 1 to n:
        ~~enter packet~~
        Input packets[i]
    count = 0
    for i = 1 to n:
        count = count + packets[i]
        if count > bucket_size:
            drop = count - bucket_size
            count = bucket_size
        send = min(count, process_rate)
        count = count - send
        Print time=i, received=packets[i]
        sent = send, left = count, dropped=drop
    While count > 0:
        send = min(count, process_rate)
        count -= send;
        Print remaining
```

F: 11010110 11

G: 1001 1

```
                1 1 0 0 0 0 1 0
        ------------------------
1 0 0 1 1 ) 1 1 0 1 0 1 1 0 1 1 0 0 0 0
            1 0 0 1 1 ↓
            ---------
            0 1 0 0 1 1
              1 0 0 1 1 ↓
              ---------
              0 0 0 0 1 0 1 1 0
                1 0 0 1 1 ↓
                ---------
                0 0 1 0 1 0 0
                  1 0 0 1 1 ↓
                  ---------
                  0 0 1 1 1 0
```

data sent = 11010110 11

Program 3: 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.

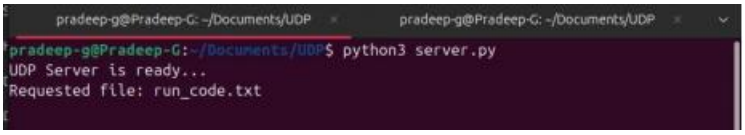
Code:

```python
# udp_client.py


import socket


# Step 1: Create UDP socket

client_socket =
socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)


server_address = ('localhost',
8081)


filename = input("Enter filename
to request: ")


# Step 2: Send filename to
server

client_socket.sendto(filename.en
code(), server_address)


# Step 3: Receive response

data, addr =
client_socket.recvfrom(4096)


print("\n--- File Content ---
\n")

print(data.decode())


# Step 4: Close socket

client_socket.close()
```

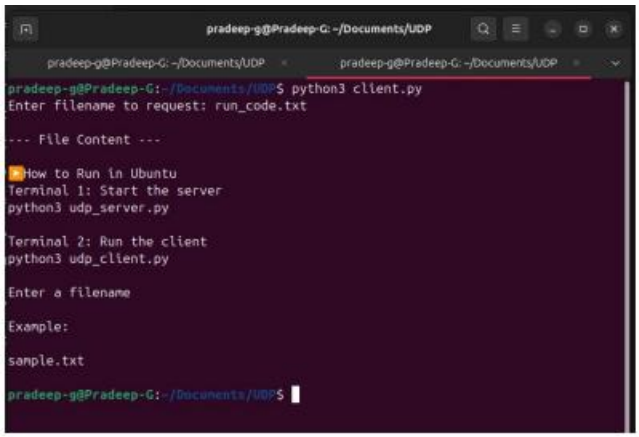
```python
# udp_server.py

import socket

# Step 1: Create UDP socket
server_socket =
socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)

# Step 2: Bind to address and port
server_socket.bind(('localhost',
8081))

print("UDP Server is ready...")

while True:
    # Step 3: Receive filename
from client
    filename, addr =
server_socket.recvfrom(1024)
    filename =
filename.decode().strip()

    print(f"Requested file:
{filename}")

    try:
        # Step 4: Open file and
send content
        with open(filename, 'r')
as f:
            data = f.read()

        server_socket.sendto(data.
encode(), addr)

    except FileNotFoundError:
        server_socket.sendto(b"Fil
e not found on server.", addr)
```

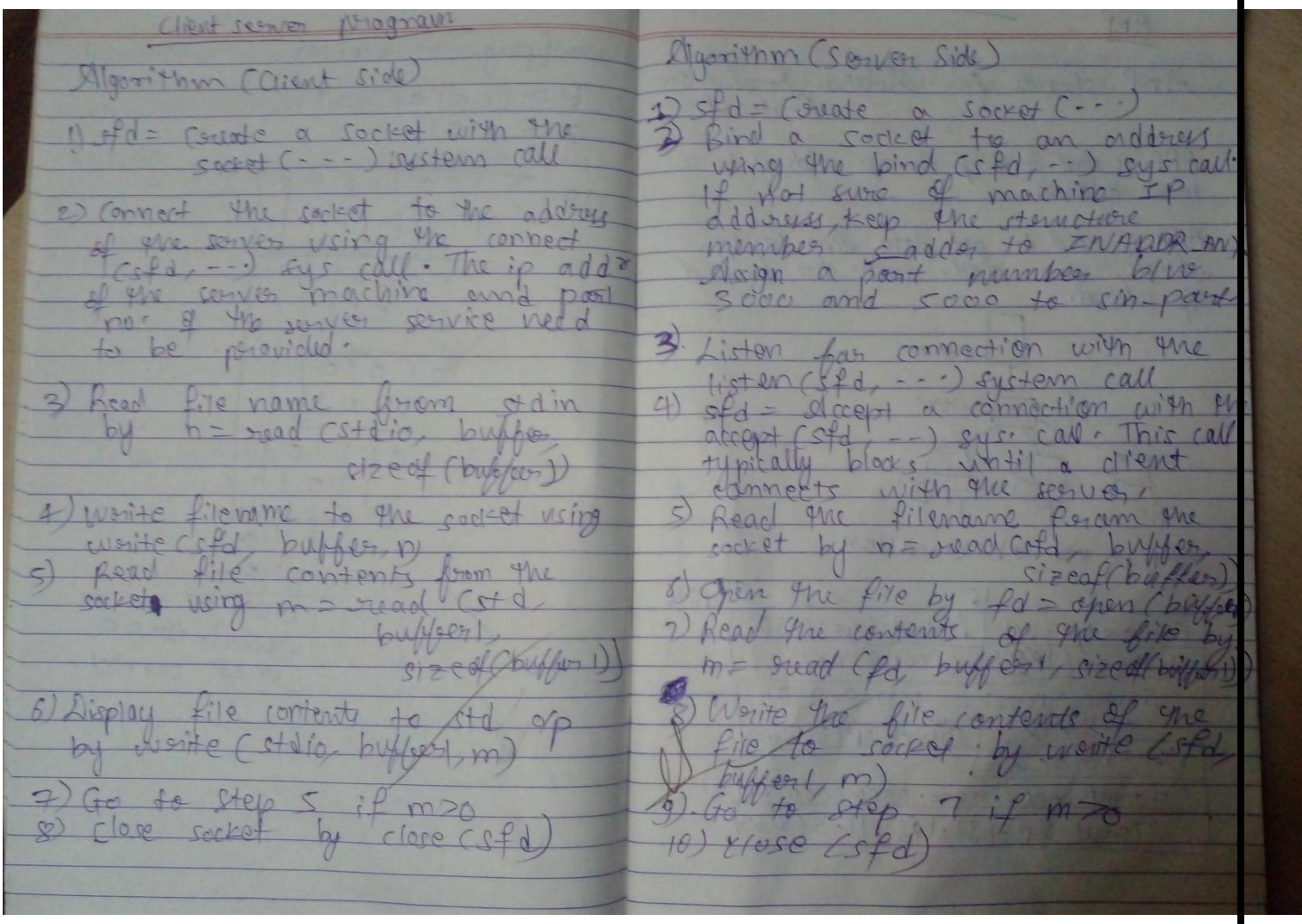
Output:

Server side Terminal:



Client side Terminal:



Program 4: Write a program for error detecting code using CRC-CCITT (16-bits).

Code:

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

int main()
{
   char rem[50], a[50], s[50], c, msj[50], gen[30];
   int i, genlen, t, j, flag = 0, k, n;

   printf("Enter the generation polynomial: ");
   fgets(gen, sizeof(gen), stdin);
   gen[strcspn(gen, "\n")] = '\0'; // remove newline if present
   printf("Generator polynomial is CRC :CCITT: %s\n", gen);

   genlen = strlen(gen);
   k = genlen - 1;

   printf("Enter the message: ");
   n = 0;
   while ((c = getchar()) != '\n' && c != EOF)
   {
      msj[n] = c;
      n++;
   }
   msj[n] = '\0';
```

```
// Append zeros
for (i = 0; i < n; i++)
   a[i] = msj[i];
for (i = 0; i < k; i++)
   a[n + i] = '0';
a[n + k] = '\0';

printf("\nMessage polynomial appended with zeros:\n");
puts(a);

// Division process
for (i = 0; i < n; i++)
{
   if (a[i] == '1')
   {
      t = i;
      for (j = 0; j <= k; j++)
      {
         if (a[t] == gen[j])
            a[t] = '0';
         else
            a[t] = '1';
         t++;
      }
   }
}
```

```
// Remainder (checksum)
for (i = 0; i < k; i++)
    rem[i] = a[n + i];
rem[k] = '\0';

printf("The checksum appended:\n");
puts(rem);

// Final transmitted message
printf("\nThe message with checksum appended:\n");
for (i = 0; i < n; i++)
    a[i] = msj[i];
for (i = 0; i < k; i++)
    a[n + i] = rem[i];
a[n + k] = '\0';
puts(a);

// Receiver side
n = 0;
printf("Enter the received message: ");
while ((c = getchar()) != '\n' && c != EOF)
{
    s[n] = c;
    n++;
}
s[n] = '\0';
```

```
for (i = 0; i < n; i++)
{
   if (s[i] == '1')
   {
      t = i;
      for (j = 0; j <= k; j++, t++)
      {
         if (s[t] == gen[j])
            s[t] = '0';
         else
            s[t] = '1';
      }
   }
}

// Check remainder
for (i = 0; i < k; i++)
   rem[i] = s[n + i];
rem[k] = '\0';

flag = 0;
for (i = 0; i < k; i++)
{
   if (rem[i] == '1')
      flag = 1;
}
```

```
    if (flag == 0)
        printf("Received polynomial is error-free\n");
    else
        printf("Received polynomial has error\n");

    return 0;
}
```

Output:

```
Enter the generation polynomial: 101
Generator polynomial is CRC :CCITT: 101
Enter the message: 110101

Message polynomial appended with zeros:
11010100
The checksum appended:
11

The message with checksum appended:
11010111
Enter the received message: 11010111
Received polynomial is error-free
```

Verification in receiver side:

```
        11000101
10011) 11010110111110
       10011
       -----
       01001110
        10011
        -----
        00000101
            1001
            ----
            00110011
             10011
             -----
             000000
```

zero

So, data is correct and not corrupted.

F | CRC → Destination
source

CRC - Algorithm

Algorithm CRC():

Input the generator polynomial (G) and the message (M).

Let K = length(G) - 1

M.append(K zeros)

A = M # augmented message

for each bit in A (up to length M):

if bit = 1, XOR the next length(G) bits of A with G.

Extract the last K bits of A → remainder (R).

Transmitted message (T) = M.append(R)

Input the received message (Rcvd).

Perform the same XOR division on Rcvd using G.

If remainder = all zeroes
→ No error
else
→ Error detected

Display transmitted message and result.

Program o/p:-

Enter the generator polynomial: 10011
Generator polynomial: 10011
Enter the message: 11010110 11

Message polynomial appended with zeros:
11010110110000
checksum (remainder): 1110

Transmitted message with checksum:
11010110111110
enter the received message: 11010110111110
Received polynomial is error free.

(2) T = 100100
G = $x^3 + x^2 + 1$ (1101)

```
      111101
1101) 100100000
      1101
      ----
      1000
      1101
      ----
      01010
       1101
       ----
       01110
        1101
        ----
        00111
         1101
         ----
         0010
```

remainder

verifying at receiver

T = 100100001
G = 1101

```
      111101
1101) 100100001
      1101
      ----
      1000
      1101
      ----
      01010
       1101
       ----
       01110
        1101
        ----
        00111
         1101
         ----
         0000
```

remainder = 0
msg correct.