

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



Lab Record

**Computer Networks – 23CS5PCCON**

*Submitted in partial fulfillment for the 5<sup>th</sup> 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 5<sup>th</sup> 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

<b>PART - A</b>	
<b>Serial No.</b>	<b>Name of Experiment</b>
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.
2.	Configure DHCP within a LAN and outside LAN.
3.	Configure Web Server, DNS within a LAN.
4.	Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.
5.	Configure default route, static route to the Router.
6.	Configure RIP routing Protocol in Routers.
7.	Configure OSPF routing protocol.
8.	To construct a VLAN and make the PC's communicate among a VLAN.
9.	To construct a WLAN and make the nodes communicate wireless.
10.	Demonstrate the TTL/ Life of a Packet.
11.	To understand the operation of TELNET by accessing the router in server room from a PC in IT office.
12.	To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP).

<b>PART – B</b>	
<b>Serial No.</b>	<b>Name of Experiment</b>
1.	Write a program for congestion control using Leaky bucket algorithm.
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.
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.
4.	Write a program for error detecting code using CRC-CCITT (16-bits).

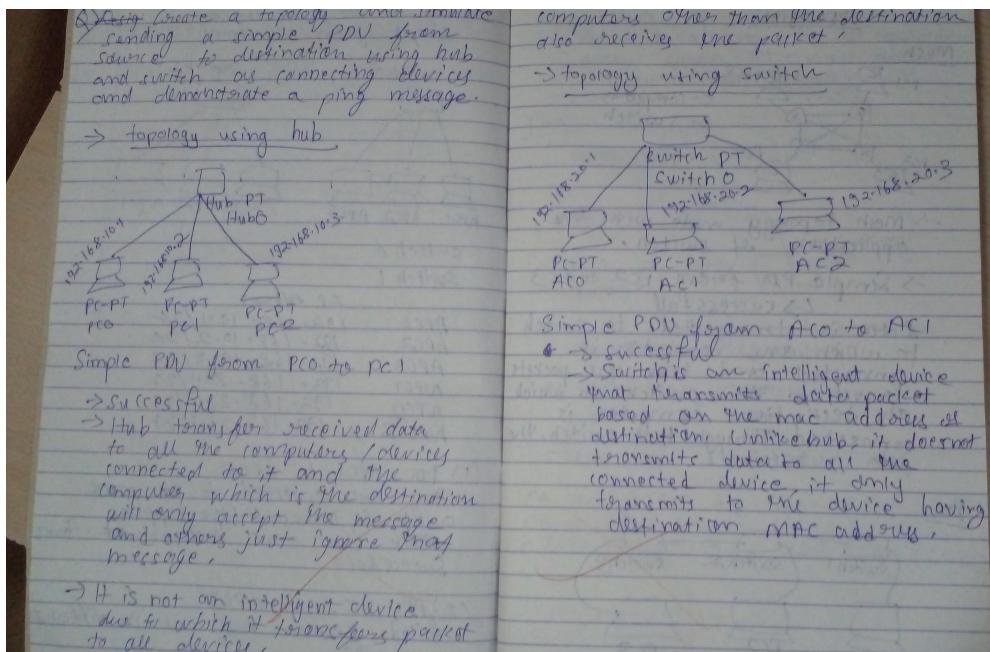
## PART - A

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:

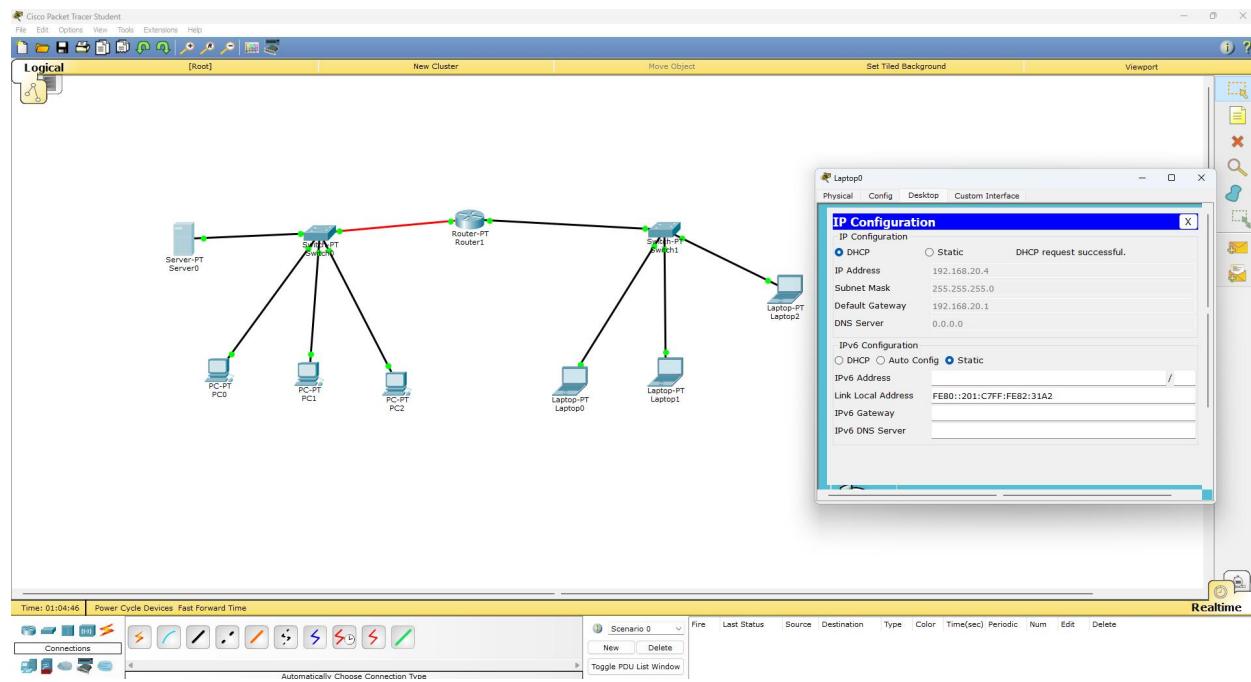


Configuration:



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

Network diagram:



Configuration:

Handwritten notes for configuring a DHCP server and router:

Configure DHCP server address the network

In DHCP Server  
 1) go to services  
 2) select DHCP  
 3) give ~~pool name~~ ~~subnet details~~  
 pool name Switch0 switch0  
 gateway 192.168.10.1 192.168.20.1  
 static IP address 192.168.10.3 192.168.20.2  
 subnet mask 255.255.255.0 255.255.255.0  
 max no of user 20 20

Router configuration

Router>enable  
 Router>conf t  
 Router>n 1  
 Router>ip address 192.168.10.1  
 Router>ip helper-address 192.168.10.2  
 Router>no shutdown

do write memory  
 #exit  
 #int FA1/0  
 #ip address 192.168.20.1  
 255.255.255.0  
 #ip helper-address 192.168.10.2  
 no shutdown  
 do write memory  
 exit  
 exit  
 write memory

Verification

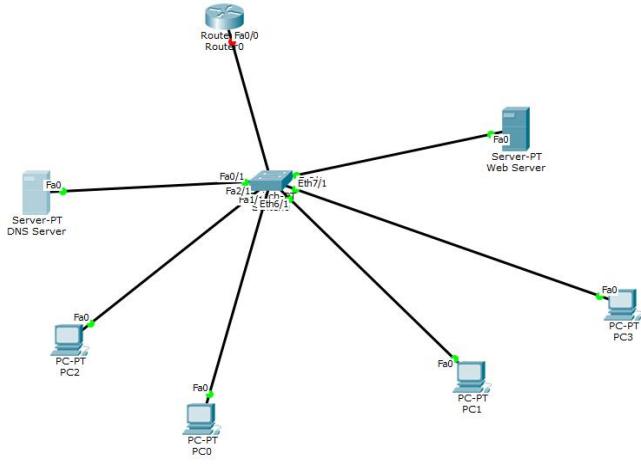
1) go to any desktop  
 2) goto desktop ip config  
 3) check whether the address is generated by the DHCP server or not  
 If yes the configuration is working

Observation

We can setup a DHCP server, for dynamically generating IP address among the networks connected through routers,

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

Network diagram:



Configuration:

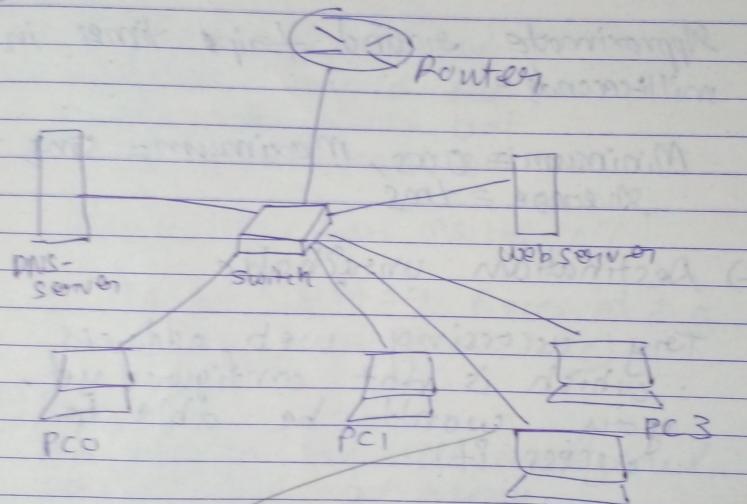
## LAB program 3

10/09

- 1) Configure webserver, DNS within a LAN
- 2) Configure IP addresses to router 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 pc which do not have DNS servers ('dn' 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 (A ping response)

Eg:- ping 192.168.1.5

O/P pinging 192.168.1.5 with 32 bytes  
of data

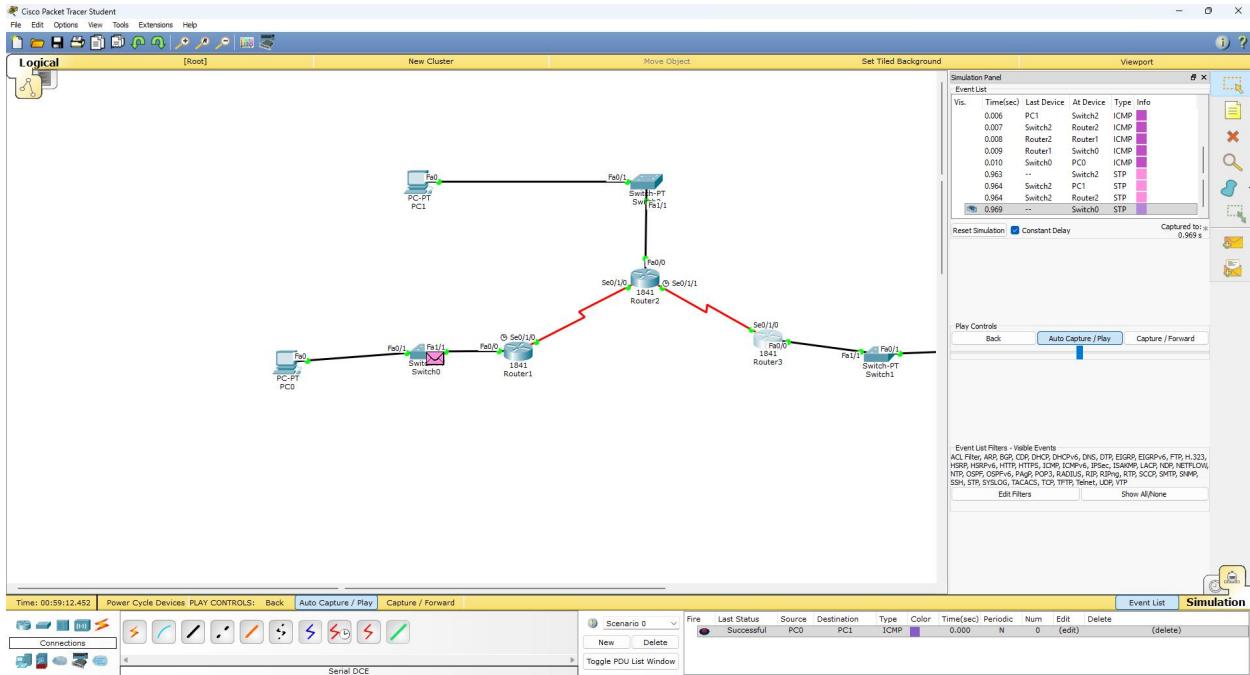
Packet : Sent = 9, Received = 9  
Lost = 0 (0% loss)

Approximate round trip times in milliseconds :

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

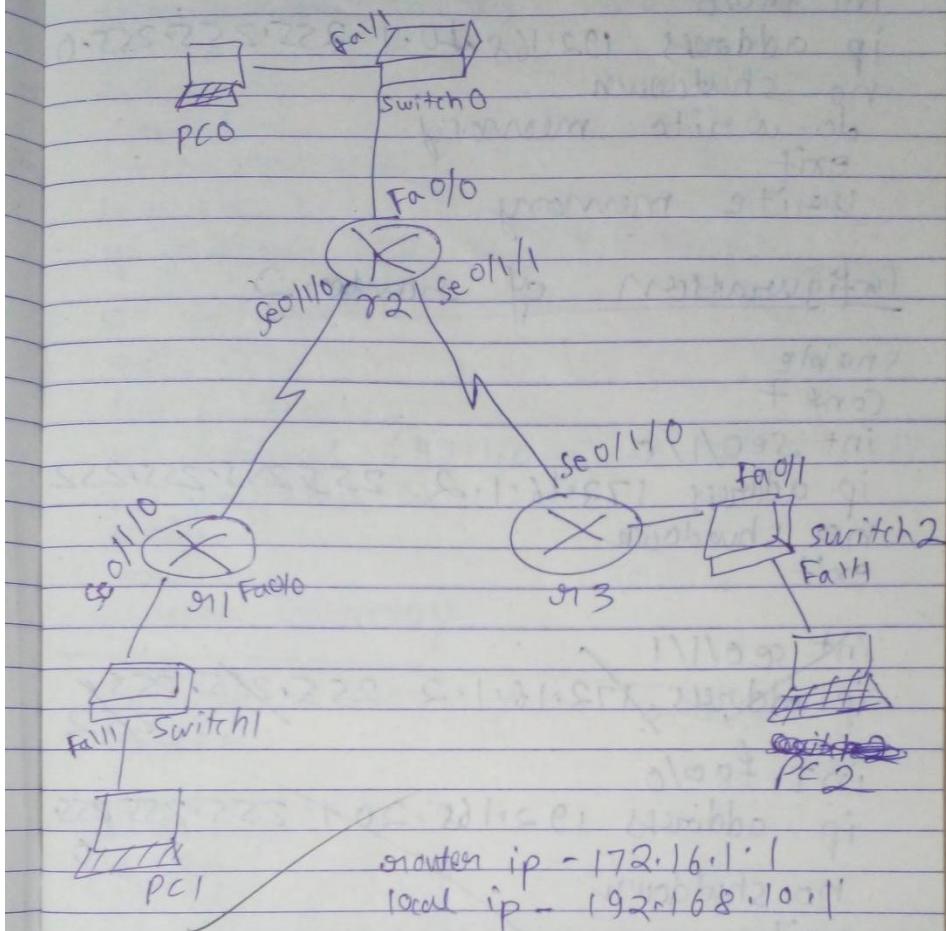
Program 4: Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.

Network diagram:



Configuration:

## LAB - 4 IPV4



configuration of router-1

enable

conf +

interface Se0/1/0

ip address 172.16.1.1 255.255.255.252

no shutdown

exit

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

### Configuration of router 2

enable  
conf +  
int se 0/1/0  
ip address 172.16.1.2 255.255.255.252  
no shutdown  
exit

int se 0/1/1  
ip address 172.16.1.2 255.255.255.252

int fa 0/0  
ip address 192.168.20.1 255.255.255.0  
no shutdown  
exit

int se 0/1/1  
ip address 172.16.2.1 255.255.255.252  
no shutdown

exit  
exit  
write memory

### Configuration of router 3

enable  
conf +  
hostname r3  
int se 0/1/1  
ip address 172.16.2.2 255.255.255.252  
no shutdown

exit  
int fa 0/0  
ip address 192.168.30.1 255.255.255.0  
no shutdown  
exit  
exit  
write memory

### Configuration for routing table

#### Router 1:-

enable  
conf +  
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 0.0.0.0 192.168.1.1

exit

write memory

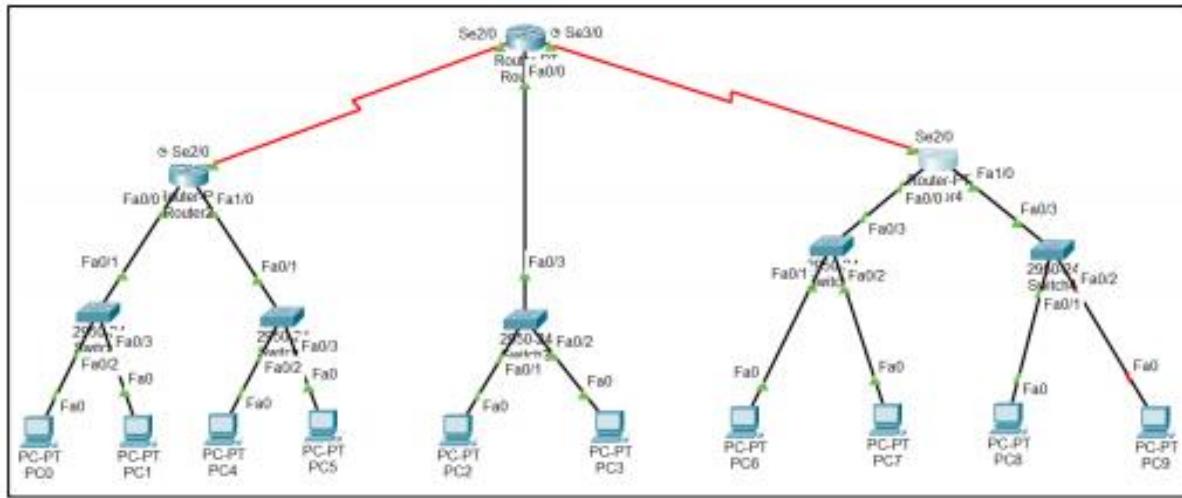
Simple PDU

PC1 to PC2 → successful

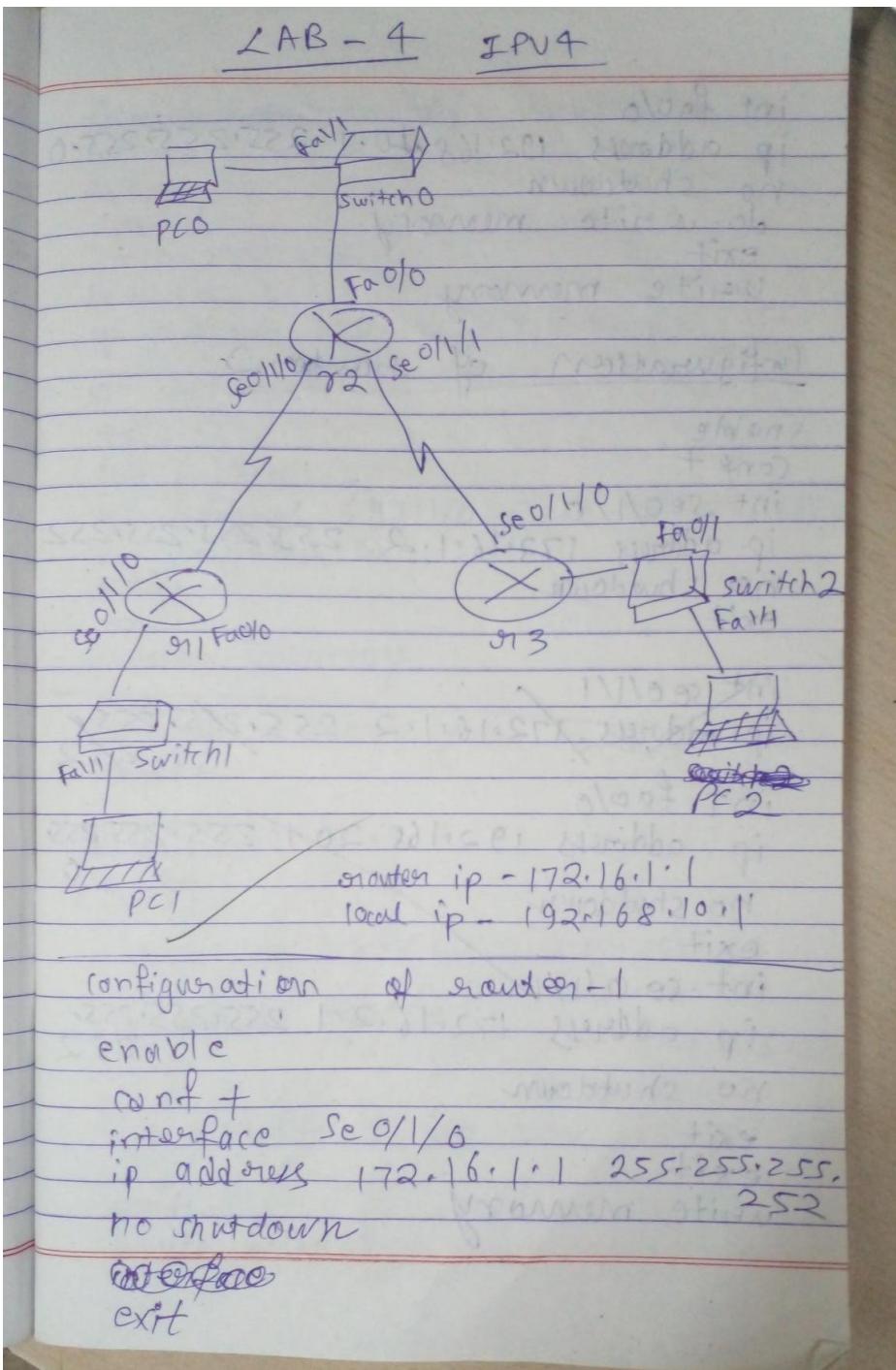
PC1 to PC0 → successful

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

Network diagram:



Configuration:



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

### Configuration of router 2

enable  
conf +  
int se 0/1/0  
ip address 172.16.1.2 255.255.255.252  
no shutdown  
exit

int se 0/1/1  
ip address 172.16.1.2 255.255.255.252

int fa 0/0  
ip address 192.168.20.1 255.255.255.0

no shutdown

exit

int se 0/1/1  
ip address 172.16.2.1 255.255.255.252

no shutdown

exit

exit

write memory

### Configuration of router 3

enable  
conf +  
hostname m3  
int se 0/1/1  
ip address 172.16.2.2 255.255.255.252

no shutdown

exit

int fa 0/0  
ip address 192.168.30.1 255.255.255.0

no shutdown

exit

exit  
write memory

### Configuration for routing table:-

ROUTER 1:-  
enable  
conf +  
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 0.0.0.0 192.168.1.1

exit

write memory

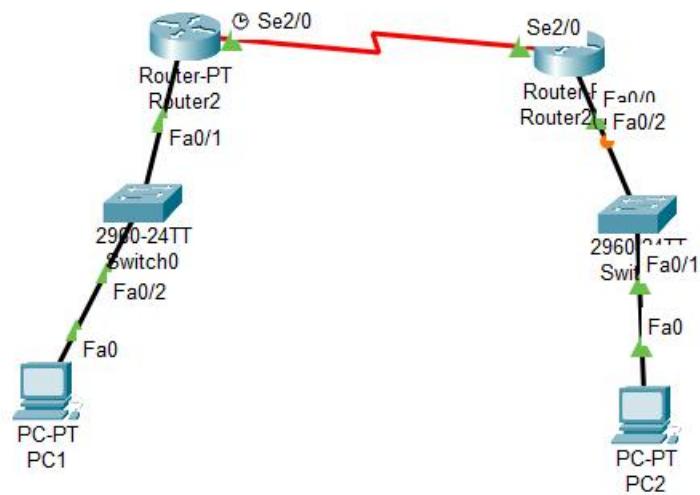
Simple PDU

PC1 to PC2 → successful

PC1 to PC0 → successful

## Program 6: Configure RIP routing Protocol in Routers.

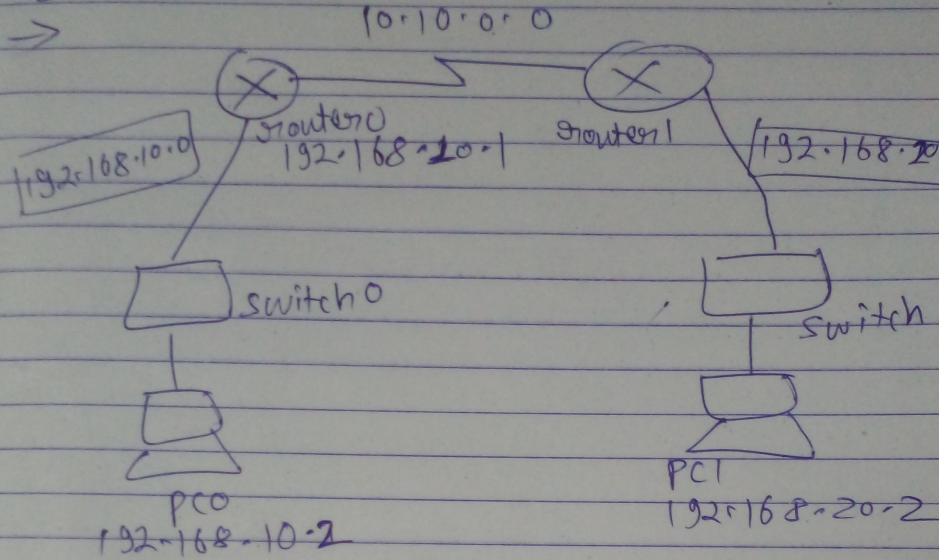
Network diagram:



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
10.10.0.0

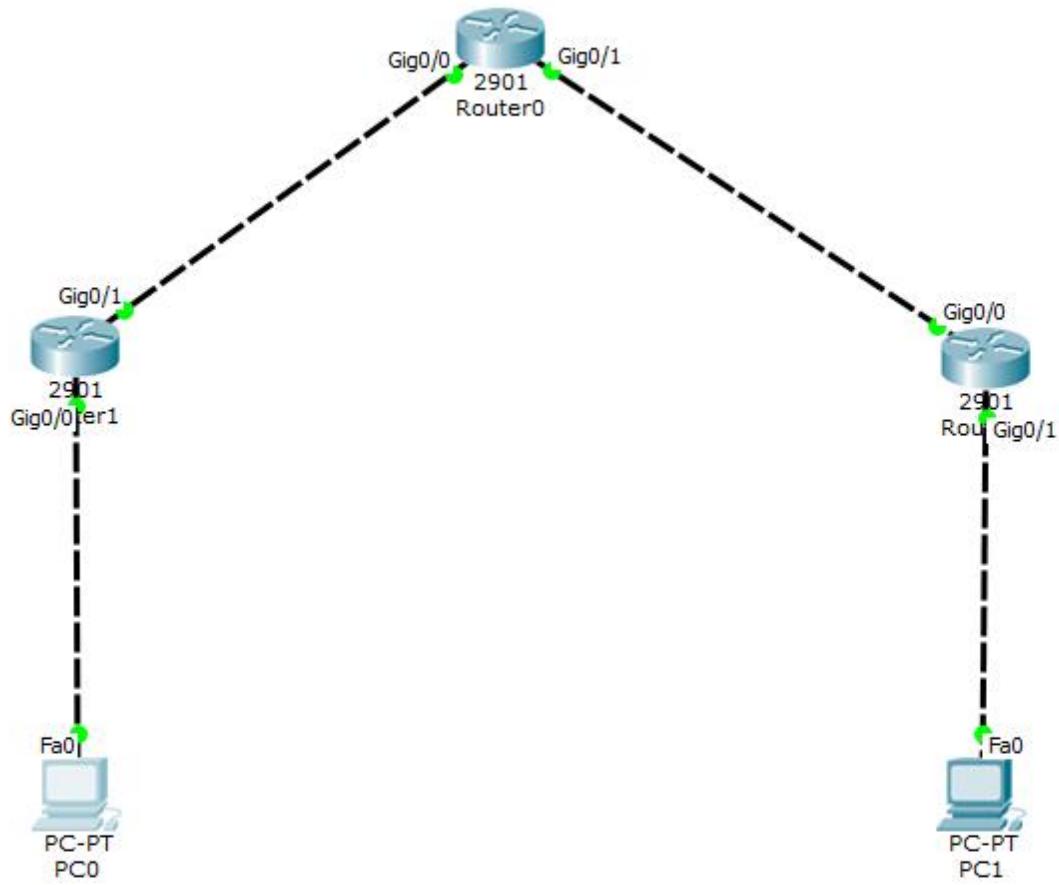
RIP table for R1 observation

~~10.10.0.0~~

Network
192.168.20.0
10.10.0.0

Program 7: Configure OSPF routing protocol.

Network diagram:



Configuration:

## Router CLI commands - R0

- 1) exit
- 2) exit
- 3) conf t
- 4) ~~conf t~~ 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
- 4) network 192.168.44.0 0.0.0.255

area 0

area 0

- 5) exit

for R2

- 1) conf t
- 2) ~~router ospf 1~~
- 3) network 176.16.0.0 0.0.0.255
- 4) network 10.0.0.6 0.25.25.255

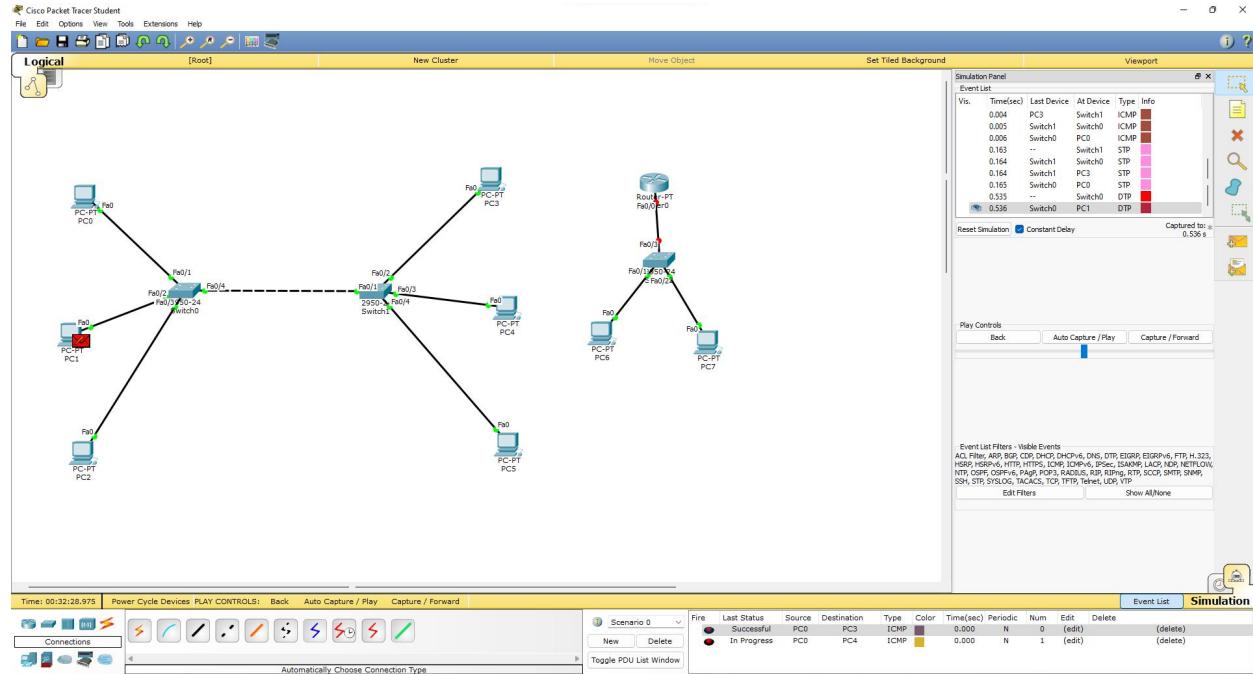
area 0

area 0

- 5) exit.

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

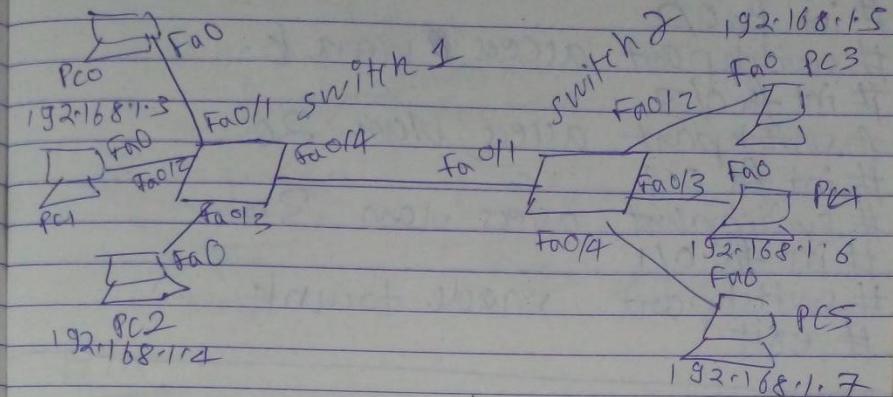
Network diagram:



Configuration:

## Lab - Program - 7

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



VLAN10	VLAN20	VLAN30
Fa0/1 PC0	Fa0/3 PC1	Fa0/3 PC2
Fa0/2 PC3	Fa0/1 PC4	Fa0/4 PCS

Virtual LAN configuration

~~commands for first switch~~

~~#enable~~

~~#config~~

~~#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 computers of different VLAN, we receive an error message !

from computer of VLAN  
viz: 192.168.1.2

ATI's ping 192.168.1.7  
cmd:

cmd -

ping 192.168.1.7

Pinging 192.168.1.7 with 32 bytes

request timed out

and time have

11 11 11  
11 11 11

Wkst. Pg. 1

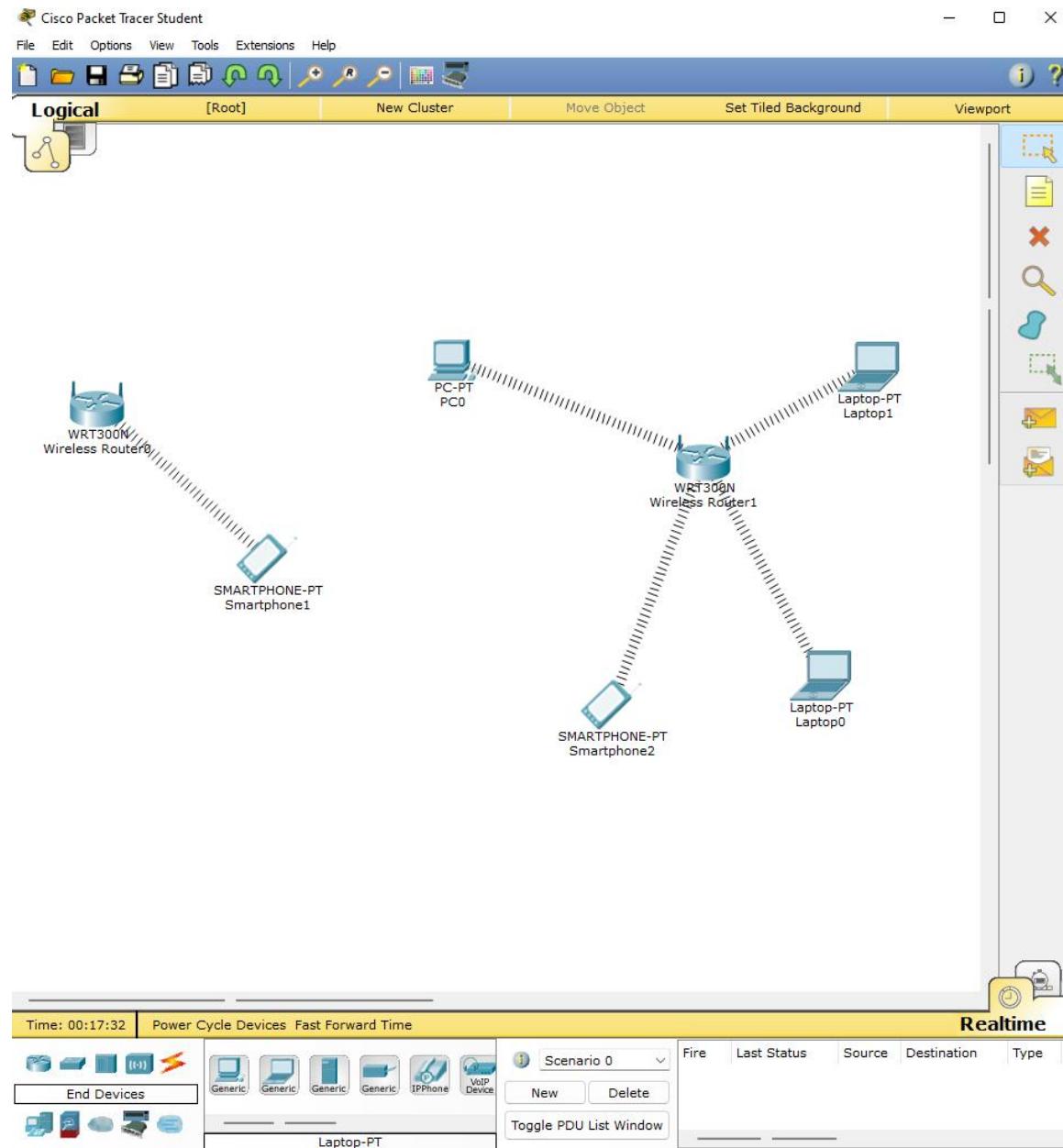
~~met: sent = 1, rec'd~~

~~Packet: sent = 4, received = 0, lost = 4~~

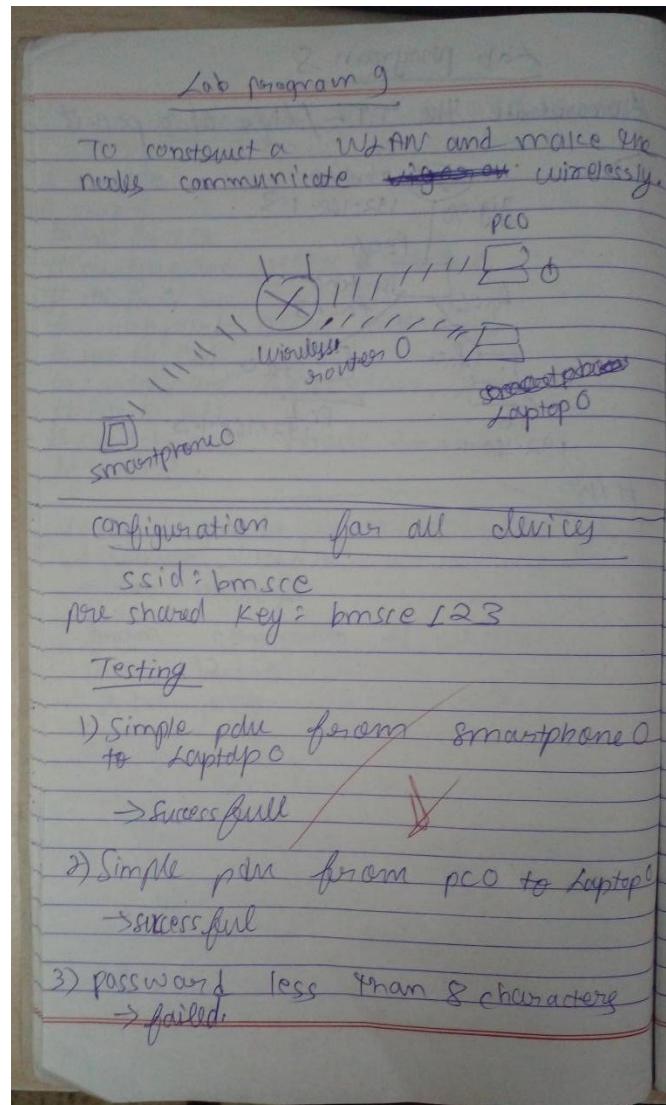
(100% loss)

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

Network diagram:

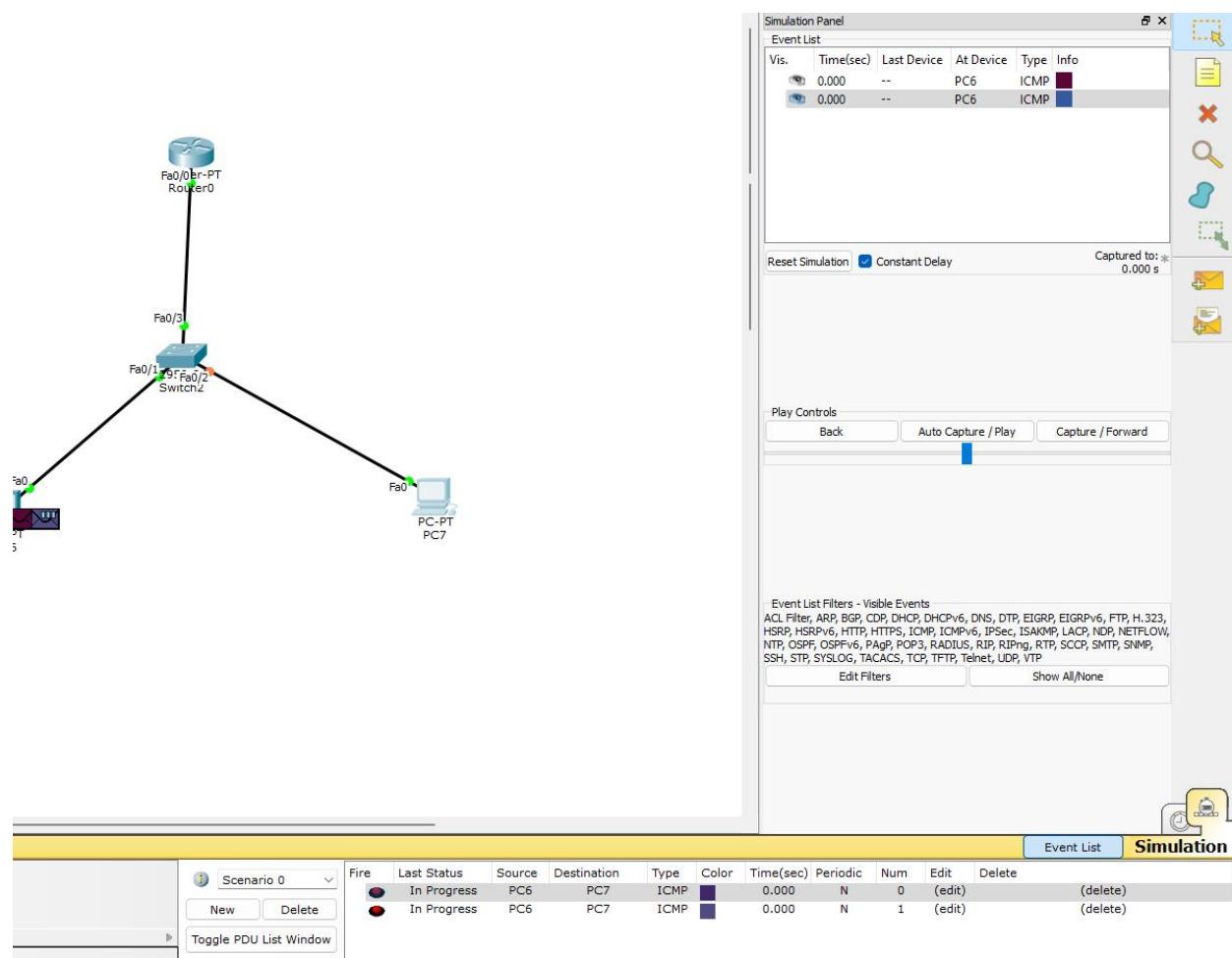


## Configuration:



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

Network diagram:



## Configuration:

Fa0/0 Er-PT  
Router0

PDU Information at Device: PC6

**OSI Model**   **Outbound PDU Details**

**PDU Formats**

Ethernet II

0	4	8	14	19	Bytes
PREAMBLE: 101010...1011		DEST MAC: 000A.4119.C9B0		SRC MAC: 0001.C931.D7A0	
TYPE: 0x800	DATA (VARIABLE LENGTH)			FCS: 0x0	

IP

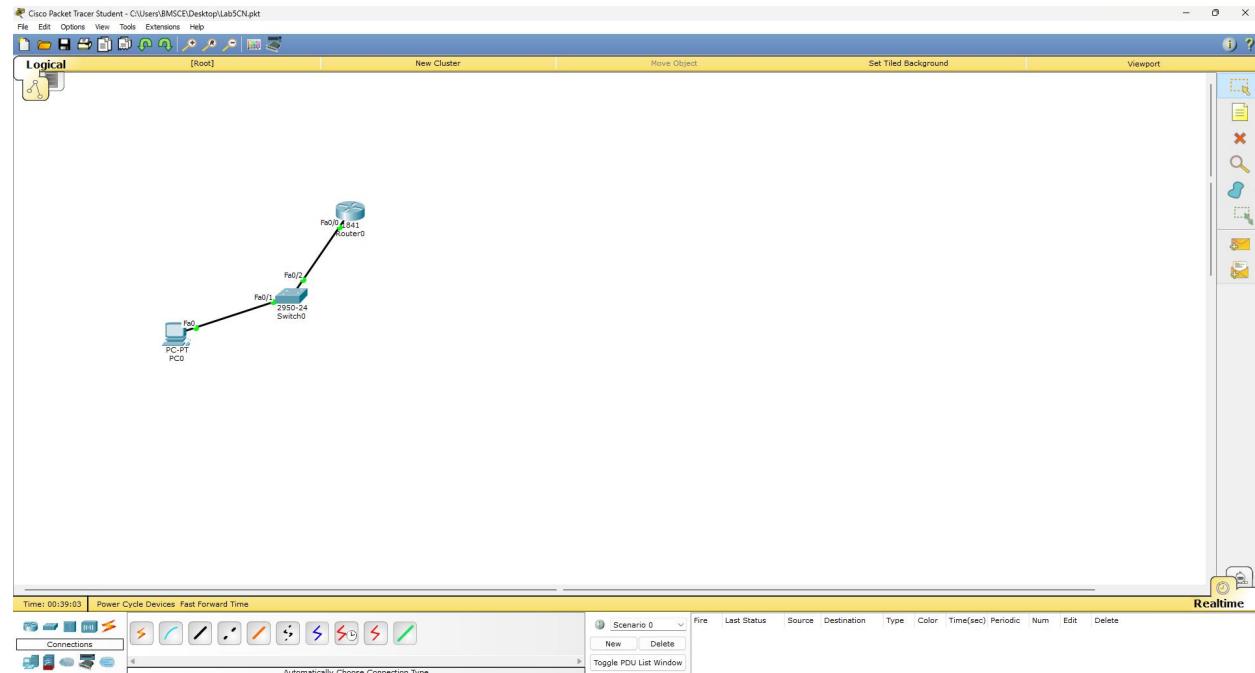
0	4	8	16	19	31	Bits
IHL		DSCP: 0x0		TL: 28		
ID: 0xb		0x0		0x0		
TTL: 255		PRO: 0x1		CHKSUM		
SRC IP: 192.168.1.2						
DST IP: 192.168.1.3						
OPT: 0x0				0x0		
DATA (VARIABLE LENGTH)						

ICMP

0	8	16	31	Bits
TYPE: 0x8		CODE: 0x0		CHECKSUM
ID: 0xc		SEQ NUMBER: 11		

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 routers remotely

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

vty → virtual terminal

Commands written in Router

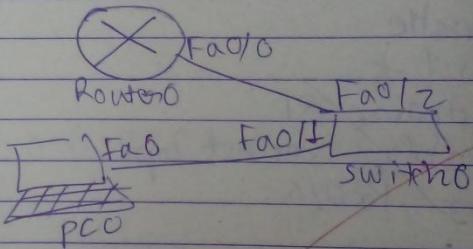
- 1) enable
- 2) conf t
- 3) hostname R1
- 4) enable secret sup
- 5) int fa0/0

- 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

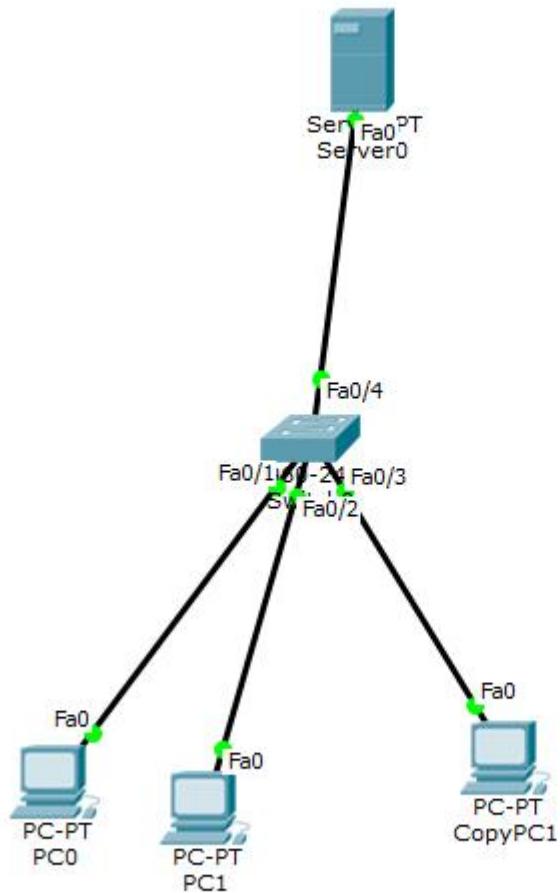
## Diagram

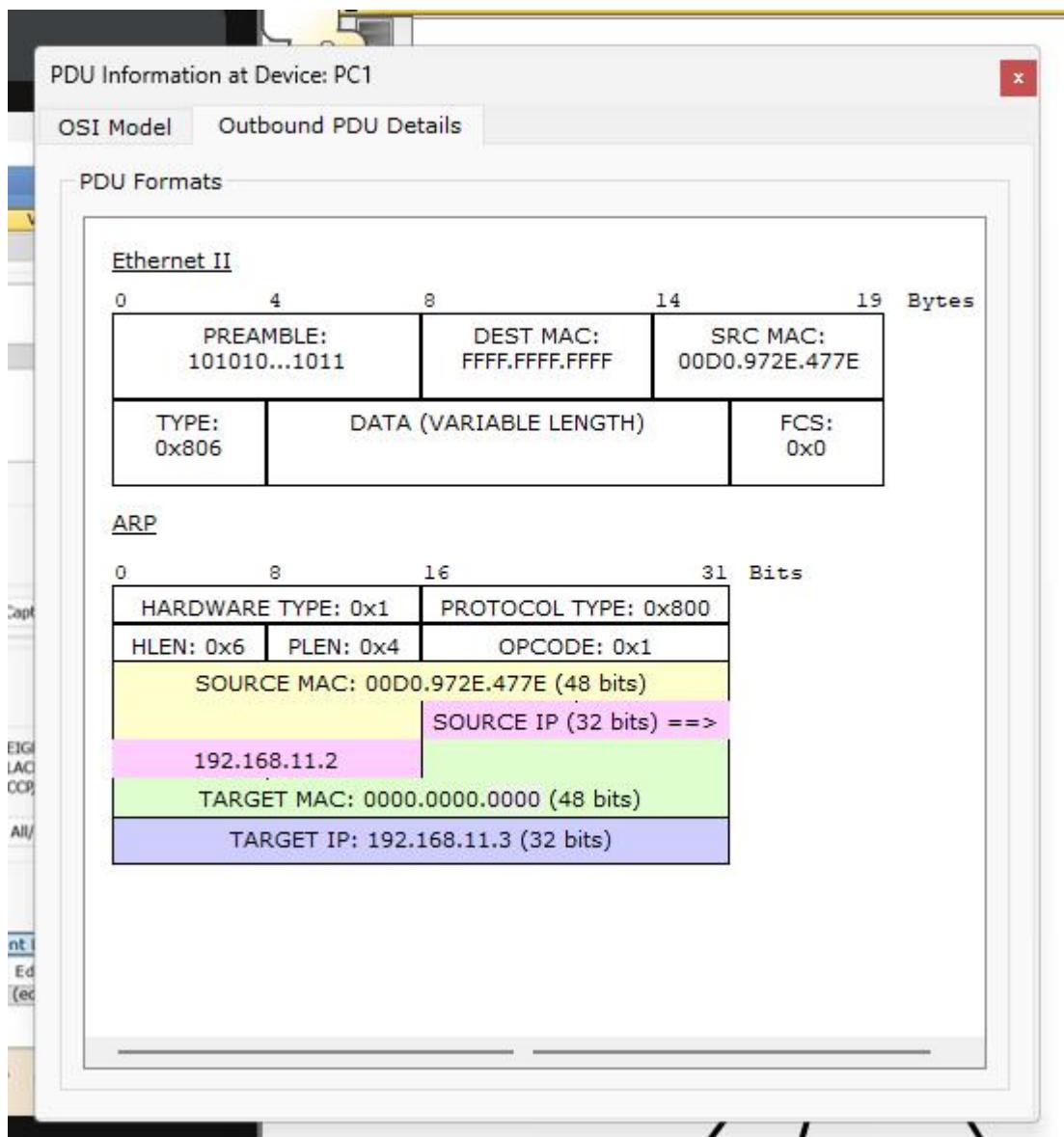


Outcome :- We can access the networking device remotely through PC using the Telnet protocol. The Telnet protocol can be used for configuring routers, switches and other networking devices.

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

Network diagram:





Configuration:

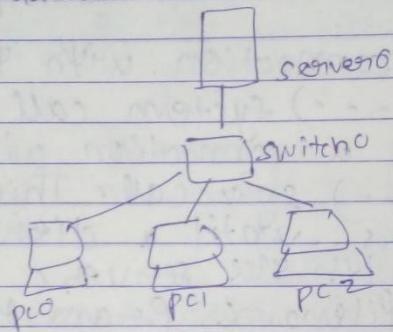
L AB-10

## ARP

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

0	8	16	31 Bits
Hardware type : 0x1		Protocol type : 0x800	
HLEN: 0x6   PLEN: 0xa		OPCODE : 0x1	
Source mac: 00:00:97:28:A7:78 (48 bits)		Source IP (32 bits)	
192.168.11.2			
Target mac : 00:00:00:00:00:00 (48 bits)			
Target IP : 192.168.11.3 (32 bits)			

## PART - B

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

Code:

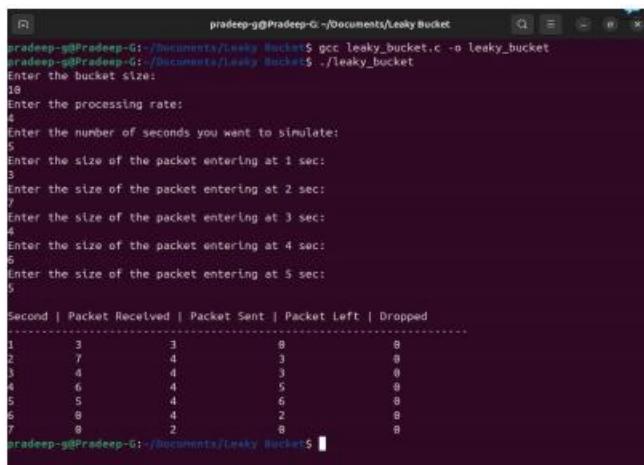
```
#include <stdio.h>
int min(int x, int y) {
if (x < y)
return x;
else
return y;
}
int main() {
int drop = 0, mini, nsec, cap, count = 0, i, inp[25],
process;
printf("Enter the bucket size:\n");
scanf("%d", &cap);
printf("Enter the processing rate:\n");
scanf("%d", &process);
printf("Enter the number 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");
-----\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("\t\t %d\n", count, drop);
    drop = 0;
}
return 0;
}

```

## Output:



```

pradeep-g@Pradeep-G:~/Documents/Leaky Bucket$ gcc leaky_bucket.c -o leaky_bucket
pradeep-g@Pradeep-G:~/Documents/Leaky Bucket$ ./leaky_bucket
Enter the bucket size:
18
Enter the processing rate:
4
Enter the number of seconds you want to simulate:
5
Enter the size of the packet entering at 1 sec:
3
Enter the size of the packet entering at 2 sec:
7
Enter the size of the packet entering at 3 sec:
4
Enter the size of the packet entering at 4 sec:
6
Enter the size of the packet entering at 5 sec:
5

Second | Packet Received | Packet Sent | Packet Left | Dropped
-----+
1     |      3          |      3       |      0       |      0
2     |      7          |      4       |      3       |      0
3     |      4          |      4       |      3       |      0
4     |      6          |      4       |      5       |      0
5     |      5          |      4       |      6       |      0
6     |      8          |      4       |      2       |      0
7     |      8          |      2       |      0       |      0
pradeep-g@Pradeep-G:~/Documents/Leaky Bucket$ 

```

## a) Lab program 102 II

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

1 " "

2 sec : 1

" "

3 sec : 2

" "

4 sec : 3

" "

5 sec : 4

" "

6 sec : 2

" "

7 sec : 1

" "

8 sec : 2

" "

9 sec : 3

" "

10 sec : 4

" "

For  $i = 1$  to  $n$ :
   
 $\text{count} = \text{count} + \text{packets}(i)$ 
  
 if  $\text{count} > \text{bucket\_size}$ :
   
 $\text{drop} = \text{count} - \text{bucket\_size}$ 
  
 $\text{count} = \text{bucket\_size}$ 
  
 $\text{send} = \min(\text{count}, \text{process\_rate})$ 
  
 $\text{count} = \text{count} - \text{send}$ 
  
 $\text{Point\_time} = i$ ,  $\text{received} = \text{packets}(i)$ 
  
 $\text{sent} = \text{send}$ ,  $\text{left} = \text{count}$ ,  $\text{dropped} = \text{drop}$ 
  
 While  $\text{count} > 0$ :
   
 $\text{Send} = \min(\text{count}, \text{process\_rate})$ 
  
 $\text{count} -= \text{Send}$ 
  
 Point remaining

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]);\n\n    mini = min(count, process);\n    printf(" %11d |", mini);\n\n    count -= mini;\n    printf(" %12d | %7d\\n", count, drop);\n    drop = 0;\n}\n\n// Continue processing remaining packets\nwhile (count != 0) {\n    if (count > cap) {\n        drop = count - cap;\n        count = cap;\n    }\n    printf("%6d | %15d |", ++i, 0);\n\n    mini = min(count, process);\n    printf(" %11d |", mini);\n\n    count -= mini;\n    printf(" %12d | %7d\\n", count, drop);\n    drop = 0;\n}\n\nreturn 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 10Z 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 "

Second	Packet received	Packet sent	Packet left	Dropped
1	1	1	0	0
2	2	2	0	0
3	1	1	0	0
4	5	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

~~X6 - program 11  
TCP/IP for socket programs~~

Algorithm for Leaky bucket

Algorithm Leaky-Bucket():

Input bucket-size, process-rate, n  
(numbers of seconds)

for i = 1 to n:

    count = count + packets[i]  
    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 = 1101011011

G: 10011

11000010

10011)11010110110000

10011↓

01001

10011

00000

10110

10011

0010100

10011

0011110

Data sent = 11010110110000

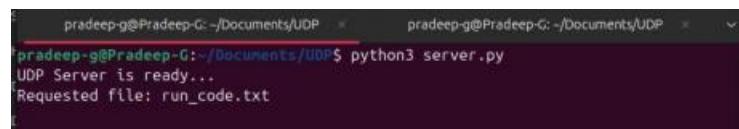
Program 3: Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

Code:

# udp_client.py	# udp_server.py
import socket	import socket
# Step 1: Create UDP socket	# Step 1: Create UDP socket
client_socket =	server_socket =
socket.socket(socket.AF_INET,	socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)	socket.SOCK_DGRAM)
server_address = ('localhost',	# Step 2: Bind to address and port
8081)	server_socket.bind(('localhost',
8081))	8081))
filename = input("Enter filename to request: ")	print("UDP Server is ready...")
# Step 2: Send filename to server	while True:
client_socket.sendto(filename.encode(), server_address)	# Step 3: Receive filename from client
# Step 3: Receive response	filename, addr =
data, addr =	server_socket.recvfrom(1024)
client_socket.recvfrom(4096)	filename =
print("\n--- File Content ---\n")	filename.decode().strip()
print(data.decode())	print(f'Requested file: {filename}')
# Step 4: Close socket	try:
client_socket.close()	# 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"File not found on server.", addr)

Output:

Server side Terminal:



```
pradeep-g@Pradeep-G:~/Documents/UDP$ python3 server.py
UDP Server is ready...
Requested file: run_code.txt
```

## Client side Terminal:

```
pradeep-g@Pradeep-G:~/Documents/UDP$ python3 client.py
Enter filename to request: run_code.txt
--- File Content ---
# How to Run in Ubuntu
Terminal 1: Start the server
python3 udp_server.py

Terminal 2: Run the client
python3 udp_client.py

Enter a filename
Example:
sample.txt
pradeep-g@Pradeep-G:~/Documents/UDPS
```

Client server program

Algorithm (Client side)

- 1)  $sfid = \text{create a socket with the socket}(\dots)$  system call
- 2) Connect the socket to the address of the server using the connect( $sfid, \dots$ ) sys call. The ip address of the servers machine and port no. of the servers service need to be provided.
- 3) Read file name from stdin by  $n = \text{read}(sfid, buffer, sizeof(buffer))$
- 4) Write filename to the socket using  $writen(sfid, buffer, n)$
- 5) Read file contents from the socket using  $m = \text{read}(sfid, buffer, sizeof(buffer))$
- 6) Display file contents to std op by  $writen(sfid, buffer, m)$
- 7) Go to step 5 if  $m > 0$
- 8) Close socket by  $\text{close}(sfid)$

Algorithm (Server Side)

- 1)  $sfid = \text{Create a socket}(\dots)$
- 2) Bind a socket to an address using the bind( $sfid, \dots$ ) sys call. If not sure of machine IP address, keep the structure member  $saddr$  to INADDRANY. Assign a port number like 5000 and 5000 to  $sin\_port$ .
- 3) Listen for connection with the listen( $sfid, \dots$ ) system call
- 4)  $sfid = \text{Accept a connection with the accept}(sfid, \dots)$  sys call. This call typically blocks until a client connects with the server.
- 5) Read the filename from the socket by  $n = \text{read}(sfid, buffer, sizeof(buffer))$
- 6) Open the file by  $fd = \text{open}(buffer)$
- 7) Read the contents of the file by  $m = \text{read}(fd, buffer, sizeof(buffer))$
- 8) Write the file contents of the file to socket by  $writen(sfid, buffer, m)$
- 9) Go to step 7 if  $m > 0$
- 10) close( $sfid$ )

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

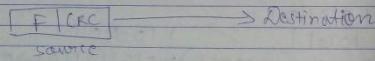
Message polynomial appended with zeros:
11010100
The checksum appended:
11

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

Verification in receiver side:

$$\begin{array}{r} 11000101 \\ 10011) 1101011011110 \\ \underline{10011} \\ 010011 \\ \underline{10011} \quad \downarrow \quad \downarrow \downarrow \downarrow \\ 0000010111 \\ \underline{10011} \quad \downarrow \\ 0010011 \\ \underline{10011} \quad \downarrow \\ \underline{\underline{000000}} \\ \text{zeroes} \end{array}$$

So, data is correct and not corrupted.



### CRC - Algorithm

Algorithm (CRC):

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

Let  $K = \text{length}(G) - 1$

$M \text{ append } (K + \text{zeros})$

$A = M \# \text{augmented message}$

for each 'bit' in  $A$  (upto length  $M$ ):

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

Extract the last  $K$  bits of  $A \rightarrow$   
remainder ( $R$ ).

Transmitted message ( $T$ ) =  $M \cdot \text{append}(R)$

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 C/C++:-

Enter the generator polynomial : 10011

Generator polynomial : 10011

Enter the message : 11010110110

Message polynomial appended with

zeros :

11010110110000

checksum (remainder) : 1110

Transmitted message with checksum:

11010110110110

Enter the received message : 11010110110110

Received polynomial is error free.

Verifying at receiver

$$F = 100100001$$

$$G = 1101$$

$$\begin{array}{r} 111101 \\ 1101) 100100001 \\ \underline{1101} \quad \downarrow \\ 1000 \\ \underline{1101} \quad \downarrow \\ 01010 \\ \underline{1101} \quad \downarrow \\ 01110 \\ \underline{1101} \quad \downarrow \\ 001100 \\ \underline{1101} \quad \downarrow \\ 00000 \\ \text{remainder} = 0 \end{array}$$

msg correct

②  $F = 100100$

$$G = m^3 + m^2 + 1 \quad (1101)$$

$$\begin{array}{r} 111101 \\ 1101) 100100000 \\ \underline{1101} \quad \downarrow \\ 1000 \\ \underline{1101} \quad \downarrow \\ 01010 \\ \underline{1101} \quad \downarrow \\ 01110 \\ \underline{1101} \quad \downarrow \\ 001100 \\ \underline{1101} \quad \downarrow \\ 00000 \\ \text{remainder} = 0 \end{array}$$

$$\begin{array}{r} 001100 \\ \underline{1101} \\ 00000 \end{array}$$

$$\begin{array}{r} 001100 \\ \underline{1101} \\ 00000 \\ \underline{1101} \\ 00000 \\ \underline{1101} \\ 00000 \end{array}$$