

# **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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



## **LAB RECORD**

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

*Submitted by*

**MAKADIA RISHIT DILIPBHAI (1BM23CS177)**

*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**

**September 2025 – January 2026**

**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 **MAKADIA RISHIT DILIPBHAI (1BM23CS177)**, 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.

Praveen N. 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	9/9/25	Configure DHCP within a LAN and outside LAN.	1
2	9/9/25	Configure Web Server, DNS within a LAN.	3
3	18/11/25	To understand the operation of TELNET	5
4	9/9/25	Configure IP address to routers in packet tracer. Explore messages:	7
5	23/9/25	Configure default route, static route to the Router	9
6	23/9/25	Configure RIP routing Protocol in Routers	11
7	14/10/25	Configure OSPF routing protocol	13
8	11/11/25	Demonstrate the TTL/ Life of a Packet	14
9	14/10/25	To construct a VLAN and make the PC's communicate among a VLAN	16
10	11/11/25	To construct a WLAN and make the nodes communicate wirelessly	18
11	18/11/25	To understand the concept and operation of Address Resolution Protocol (ARP)	19
12	19/8/25	Create a topology and simulate sending a simple PDU using Hub and Switch	21
14	28/10/25	Congestion control using Leaky bucket algorithm.	22
15	28/10/25	Using TCP/IP sockets, write a client-server program	24
16	28/10/25	Using UDP sockets, write a client-server program	25
17	28/10/25	Error detecting code using CRC-CCITT	26

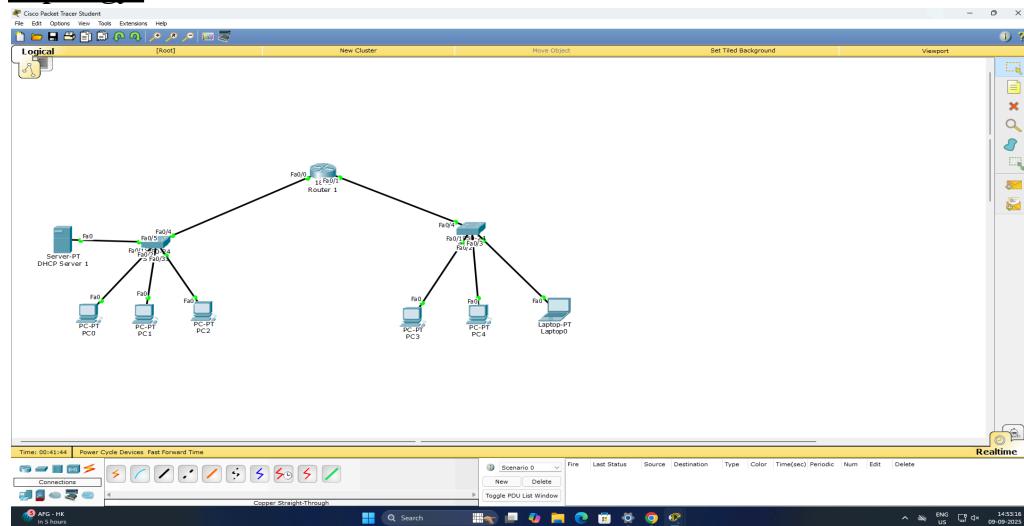
Github Link:

[https://github.com/rishitmakadia/1BM23CS177\\_CON\\_5](https://github.com/rishitmakadia/1BM23CS177_CON_5)

## Program – 1:

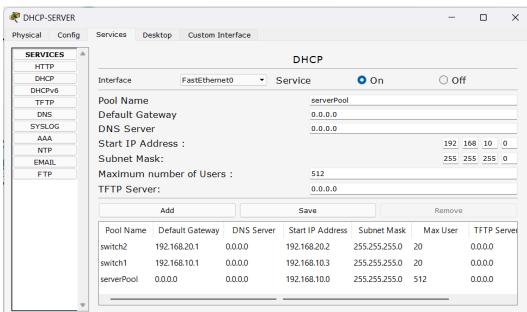
Aim: Configure DHCP within a LAN and outside LAN.

Topology:



Procedure:

1. Configure DHCP Server:  
in DHCP server go to Desktop>IP-Config, assign static IP – 192.168.10.2 and gateway 192.168.10.1
2. Open Services>DHCP and add following two dhcp pool:
  - (a) Pool Name: switch1  
Gateway: 192.168.10.1  
Start Ip: 192.168.10.3  
Subnet Mask: 255.255.255.0  
Max Users: 20
  - (b) Pool Name: switch2  
Gateway: 192.168.20.1  
Start Ip: 192.168.10.2  
Subnet Mask: 255.255.255.0  
Max Users: 20



3. Configure Router

Router>enable

Router#configure terminal

Router(config)# int fa0/0

Router(config-if)# ip address 192.168.10.1 255.255.255.0

Router(config-if)# ip helper-address 192.168.10.2

Router(config-if)# no shutdown

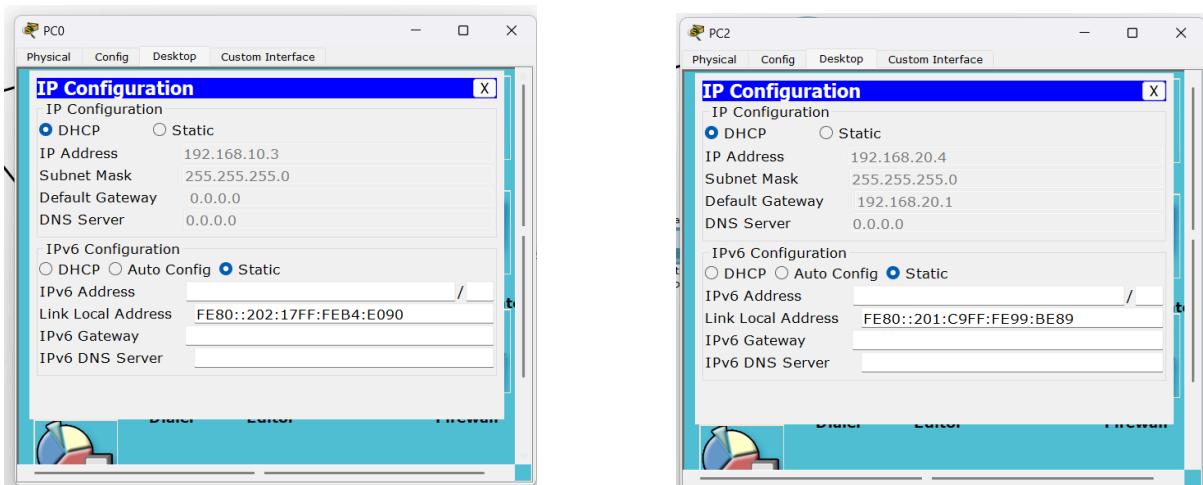
Router(config-if)# exit

```

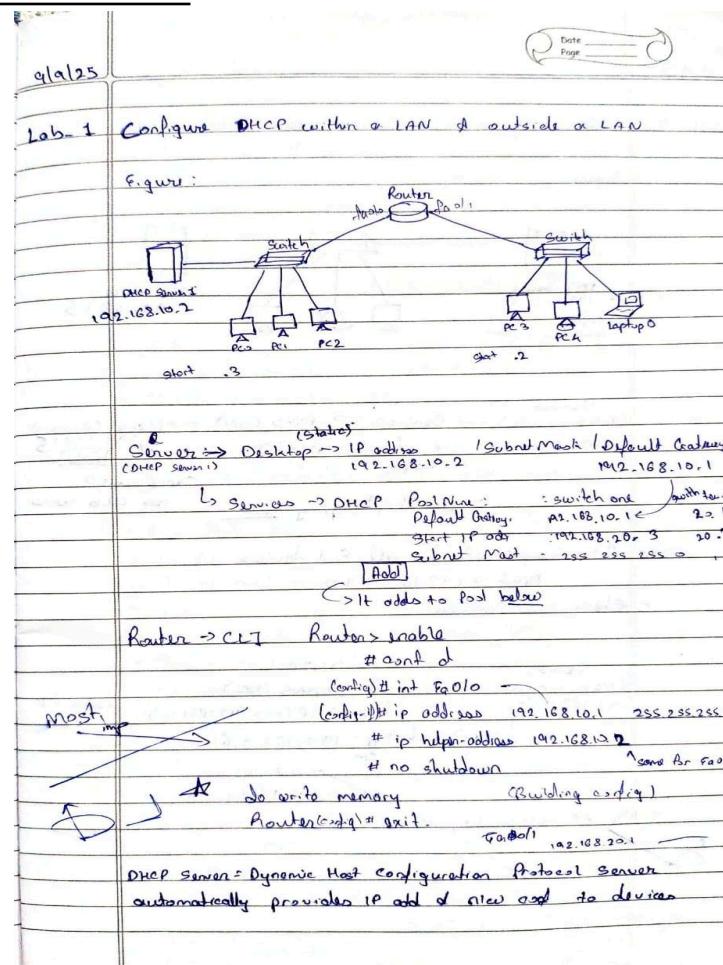
Router(config)# int fa0/1
Router(config-if)# ip address 192.168.20.1 255.255.255.0
Router(config-if)# ip helper-address 192.168.10.2
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# exit
Router# write memory

```

## Output



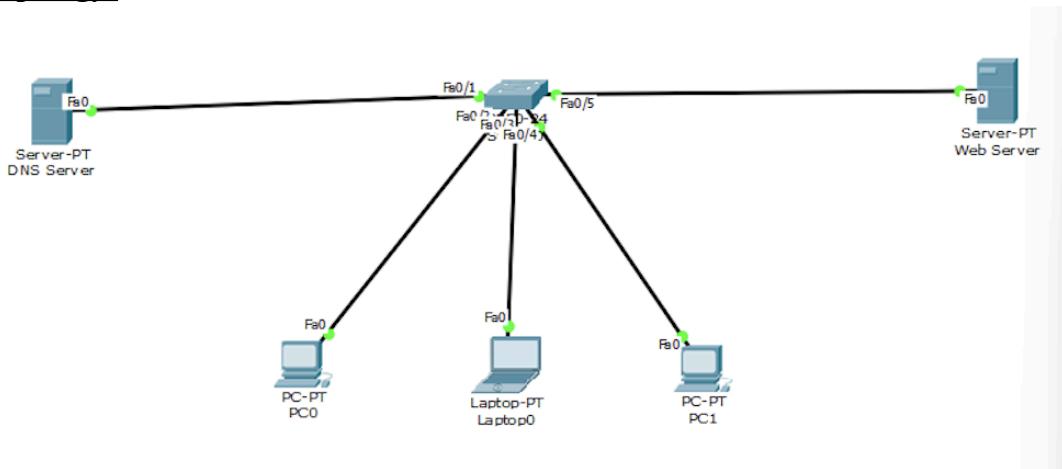
## Observation



## Program 2

**Aim:** Configure Web Server, DNS within a LAN.

**Topology:**



### Procedure:

#### 1. Configure Server

- Set static IP: 192.168.1.10/24
- Services > DNS: Enable, add A record:  
www.mysite.com → 192.168.1.10
- Services > HTTP: Enable, add simple HTML page

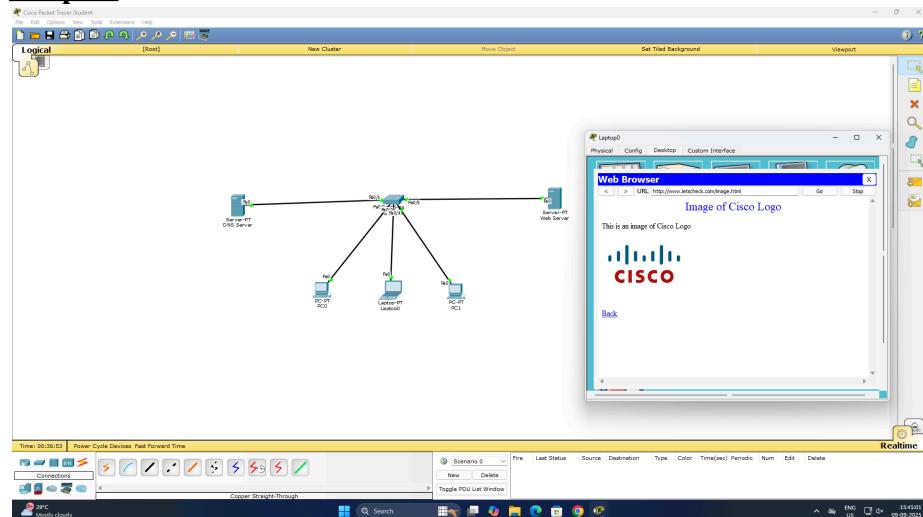
#### 2. Configure Clients

- Set IP: 192.168.1.x/24
- Set DNS: 192.168.1.10

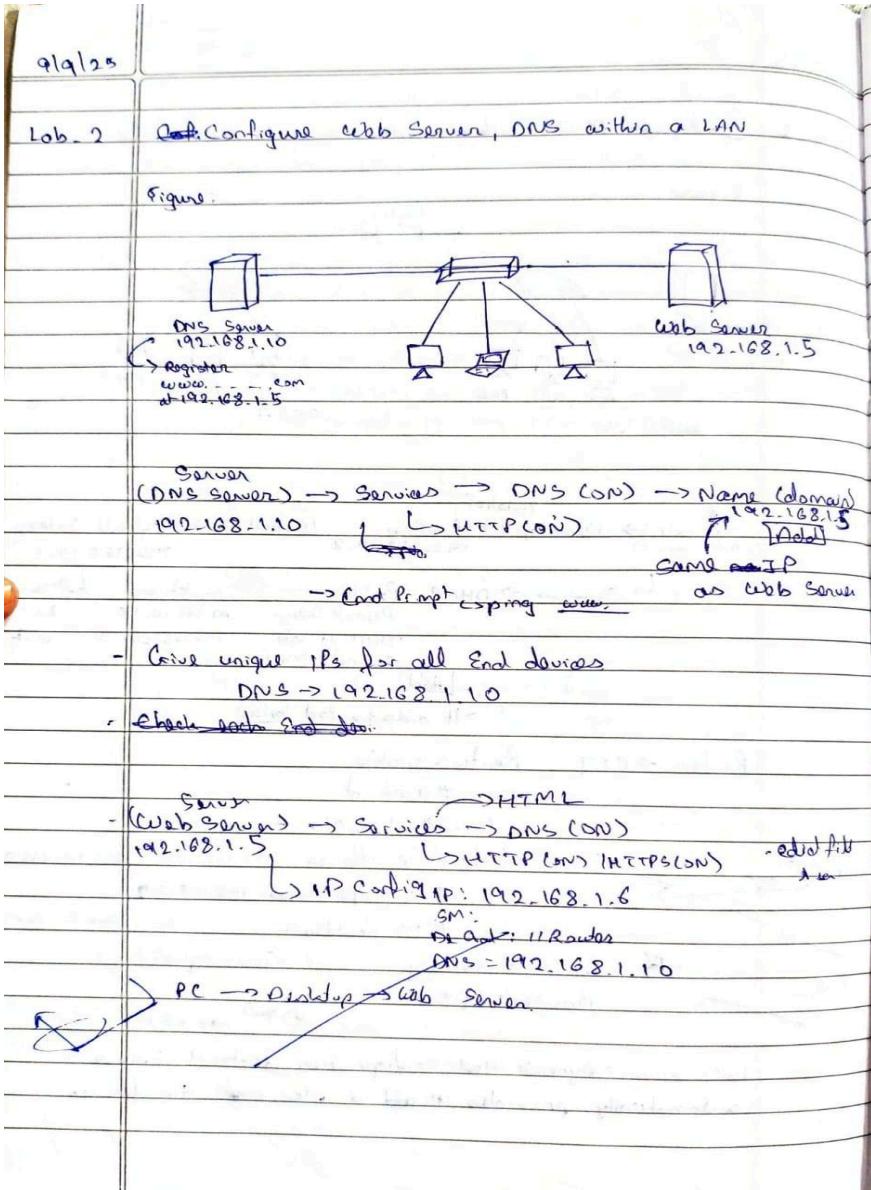
#### 3. Verification

- Command Prompt: ping www.mysite.com (should resolve to 192.168.1.10)
- Web Browser: Navigate to http://www.mysite.com (should load webpage)

### Output:



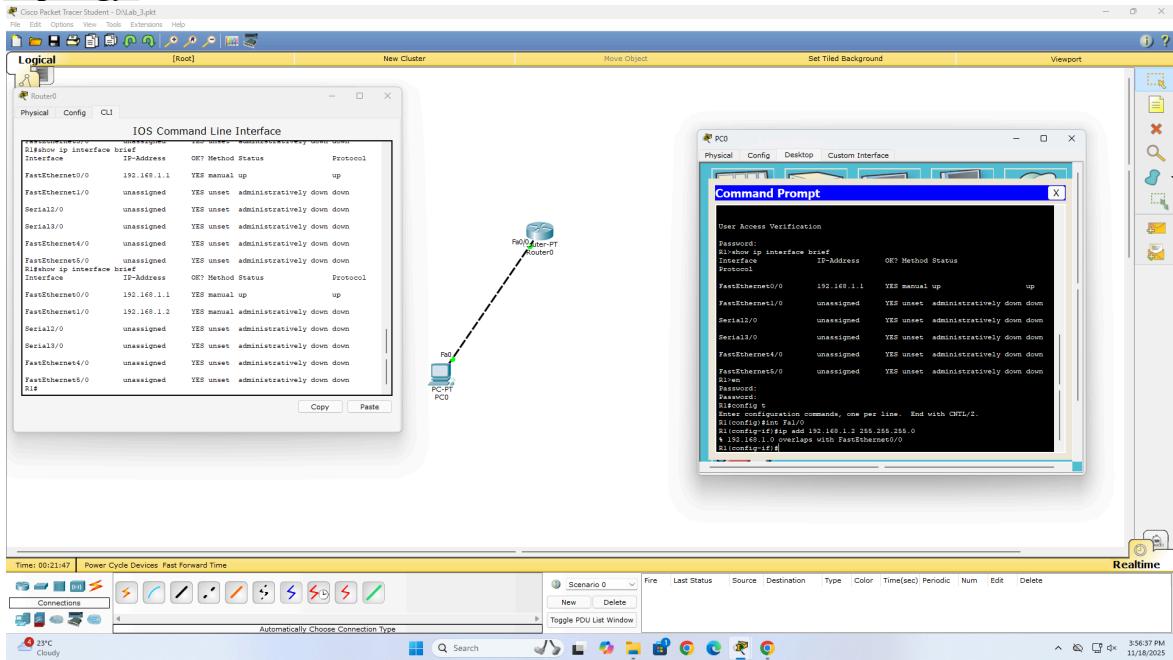
## Observation:



## Program 3

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

### **Topology:**



### **Procedure:**

#### 1. Configure Router

```
Router> enable
Router# configure terminal
Router(config)# hostname R1
R1(config)# interface fastethernet 0/0
R1(config-if)# ip address 192.168.1.1 255.255.255.0
R1(config-if)# no shutdown
R1(config-if)# exit
R1(config)# line vty 0 4
R1(config-line)# password tp
R1(config-line)# login
R1(config-line)# exit
R1(config)# enable secret class
R1(config)# exit
R1# write memory
```

#### 2. Configure PC

- Set static IP: 192.168.1.2/24
- Default gateway: 192.168.1.1

#### 3. Establish TELNET Session

ip: 192.168.1.2  
mask: 255.255.255.0

gateway: 192.168.1.1

# Test connectivity

ping 192.168.1.1

telnet 192.168.1.1

# Password: tp

# enable

# Secret: class

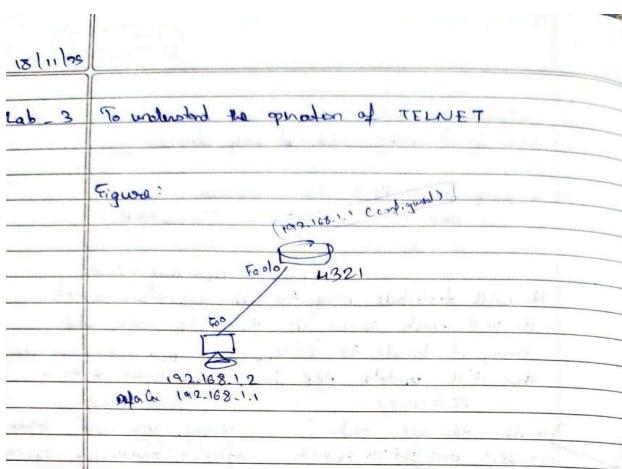
## Output:

```
C:\>ping 192.168.7.1
Pinging 192.168.7.1 with 32 bytes of data:
Reply from 192.168.7.1: bytes=32 time<1ms TTL=255

Ping statistics for 192.168.7.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>telnet 192.168.7.1 ...Open
User Access Verification
Password: tp
R1>en
R1#config-t
R1(config)#int Fa1/0
R1(config-if)#ip add 192.168.1.2 255.255.255.0
$ 192.168.1.0 overlaps with FastEthernet0/0
R1(config-if)#
R1#show ip interface brief
Interface          IP-Address      OK? Method Status      Protocol
FastEthernet0/0     192.168.1.1    YES manual up
FastEthernet1/0     unassigned      YES unset administratively down down
Serial2/0           unassigned      YES unset administratively down down
Serial3/0           unassigned      YES unset administratively down down
FastEthernet4/0     unassigned      YES unset administratively down down
FastEthernet5/0     unassigned      YES unset administratively down down
R1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#int GigabitEthernet0/0/1
R1(config-if)#ip add 192.168.8.1 255.255.255.0
R1(config-if)#exit
R1(config)#exit
R1#show ip interface brief
Interface          IP-Address      OK? Method Status      Protocol
GigabitEthernet0/0/0 192.168.7.1    YES manual up
GigabitEthernet0/0/1 192.168.8.1    YES manual administratively down down
Vlan1              unassigned      YES unset administratively down down
R1#exit
[Connection to 192.168.7.1 closed by foreign host]
C:\>
```

## Observation:



- It is used to access remote servers
- It is simple command tool that runs on comp. & allow to send cmd remotely to a server.
- TELNET is also used to manage other devices like router/switch to check if ports are open or close on a server.

Router → CLI

Router>en

# config t

Router(Config)# hostname R1

R1(Config)# enable secret rp

# int Fa0/0

Router(Config)# ip add 192.168.1.1 255.255.255.0

# no shut down

enable  
configure terminal  
line vty 0 4  
password: tp  
transport input telnet  
exit  
virtual terminal line 0 to 5  
config-vty 0 3  
login  
exit  
config-line # password: tp  
exit  
R1#  
R1# show ip interface brief

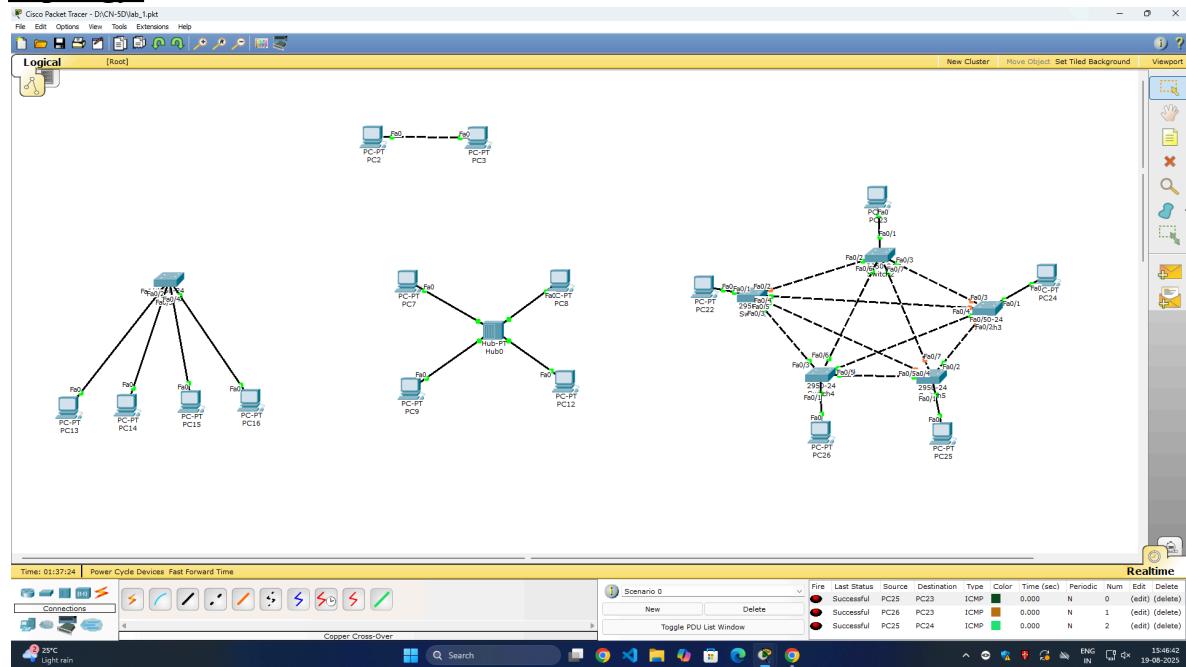
PC O → cmd prompt  
C:\> ping 192.168.1.1

> telnet 192.168.1.1  
Trying 192.168.1.1 ...open  
User Access Verification  
Password: tp  
R1>en  
R1> show ip interface brief (Same as route)  
R1>  
R1# config t  
R1# ip add 192.168.1.2 255.255.255.0  
\$ 192.168.1.0 overlaps with FastEthernet0/0  
R1(config)#  
C:→ check in Router CLI show ip interface brief

## Program 4

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

### Topology:



### Output:

#### Command Prompt

```
Packet Tracer PC Command Line 1.0
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),
```

```
[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
```

```
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>
```

## Observation:

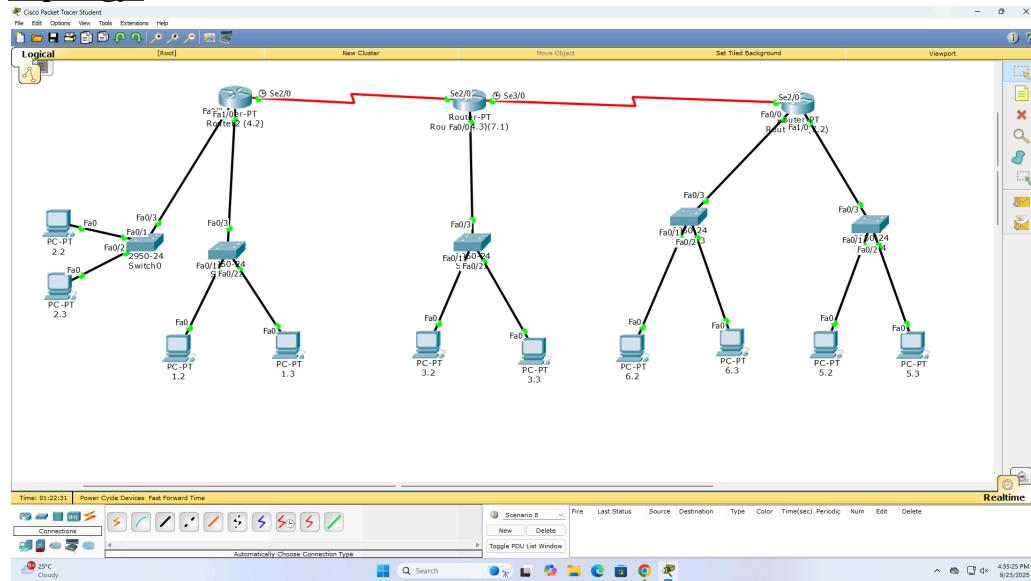
		Date _____ Page _____
Lab - h	Configure IP address to the Router in Packet tracer. Explore the following messages	
	<ul style="list-style-type: none"> <li>• ping response.</li> <li>• destination unreachable.</li> <li>• request time out.</li> <li>• reply</li> </ul>	
	<p>⇒ Ping:</p> <p>(It can be checked in any end device)</p> <p>Cmd Prompt <u>ping IP</u></p> <ul style="list-style-type: none"> <li>- The destination is reachable.</li> <li>- IP addresses are correct &amp; the network path works.</li> <li>- Both source &amp; destination are configured properly &amp; connected.</li> </ul>	
	<p>⇒ Destination Unreachable.</p> <ul style="list-style-type: none"> <li>- Router learned knows the destination network does not exist or no route to it.</li> </ul> <p>Causes:</p> <ul style="list-style-type: none"> <li>• Wrong IP address or subnet mask</li> <li>• Missing route in routers routing table</li> <li>• No interface configured for the network.</li> </ul>	
	<p>⇒ Request Time out</p> <ul style="list-style-type: none"> <li>- Packet was sent but no reply was received within the timeout.</li> </ul> <p>Causes:</p> <ul style="list-style-type: none"> <li>• Device is OFF or not responding</li> <li>• Firewall blocking ICMP ping</li> <li>• Wrong cabling or interface down.</li> </ul> <p>ICMP = Internet Control Message Protocol</p>	

	<p>⇒ Reply</p> <ul style="list-style-type: none"> <li>• This is the successful case where packets return correctly</li> <li>• It shows network connectivity is fine.</li> </ul>
	<p>Router with CLI</p> <p>Router → Config</p> <p>↳ Select interface (G0/0/0)</p> <p>Port Status: ON (= no shutdown)</p> <p>IP add</p> <p>Subnet mask</p> <p>* Assign IP to router interface</p> <p>enable</p> <p>config terminal</p> <p>interface fastethernet 0/0</p> <p>ip address 192.168.1.1 255.255.255.0</p> <p>no shutdown</p> <p>exit.</p>

## Program 5

**Aim:** Configure default route, static route to the Router

### **Topology:**



### **Procedure:**

```
Router> enable
Router# configure terminal
Router(config)# hostname Router3
Router3(config)# interface fastethernet 0/0
Router3(config-if)# ip address 192.168.4.1 255.255.255.0
Router3(config-if)# no shutdown
Router3(config-if)# exit
Router3(config)# interface fastethernet 1/0
Router3(config-if)# ip address 192.168.5.1 255.255.255.0
Router3(config-if)# no shutdown
Router3(config-if)# exit
Router3(config)# ip route 192.168.1.0 255.255.255.0 192.168.4.1
Router3(config)# ip route 192.168.2.0 255.255.255.0 192.168.4.1
Router3(config)# ip route 192.168.3.0 255.255.255.0 192.168.4.2
Router3(config)# ip route 192.168.6.0 255.255.255.0 192.168.5.2
# Configure default route (if needed)
Router3(config)# ip route 0.0.0.0 0.0.0.0 192.168.4.1
Router3(config)# exit
Router3# write memory
```

## Output:

Physical    **Config**    CLI    Attributes

GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0

Static Routes

Network:   
Mask:   
Next Hop:

Add

Network Address

192.168.3.0/24 via 192.168.4.2  
192.168.6.0/24 via 192.168.4.2

Remove

Physical    **Config**    CLI    Attributes

GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0

Static Routes

Network:   
Mask:   
Next Hop:

Add

Network Address

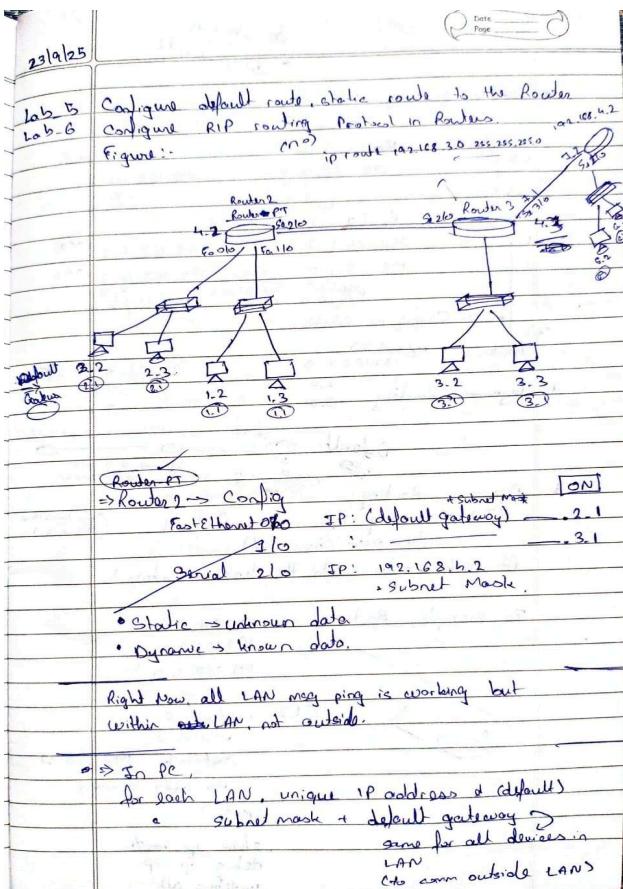
192.168.6.0/24 via 192.168.5.2  
192.168.2.0/24 via 192.168.4.1  
192.168.1.0/24 via 192.168.4.1

Remove

Equivalent IOS Commands

```
LINK-5-CHANGED: Interface Serial1/0, changed state to up
LINK-5-CHANGED: Line protocol on Interface FastEthernet0/0, changed state to up
LINK-5-CHANGED: Interface Serial3/0, changed state to up
LINK-5-CHANGED: Line protocol on Interface Serial3/0, changed state to up
LINK-5-CHANGED: Line protocol on Interface Serial2/0, changed state to up
Router#enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#
```

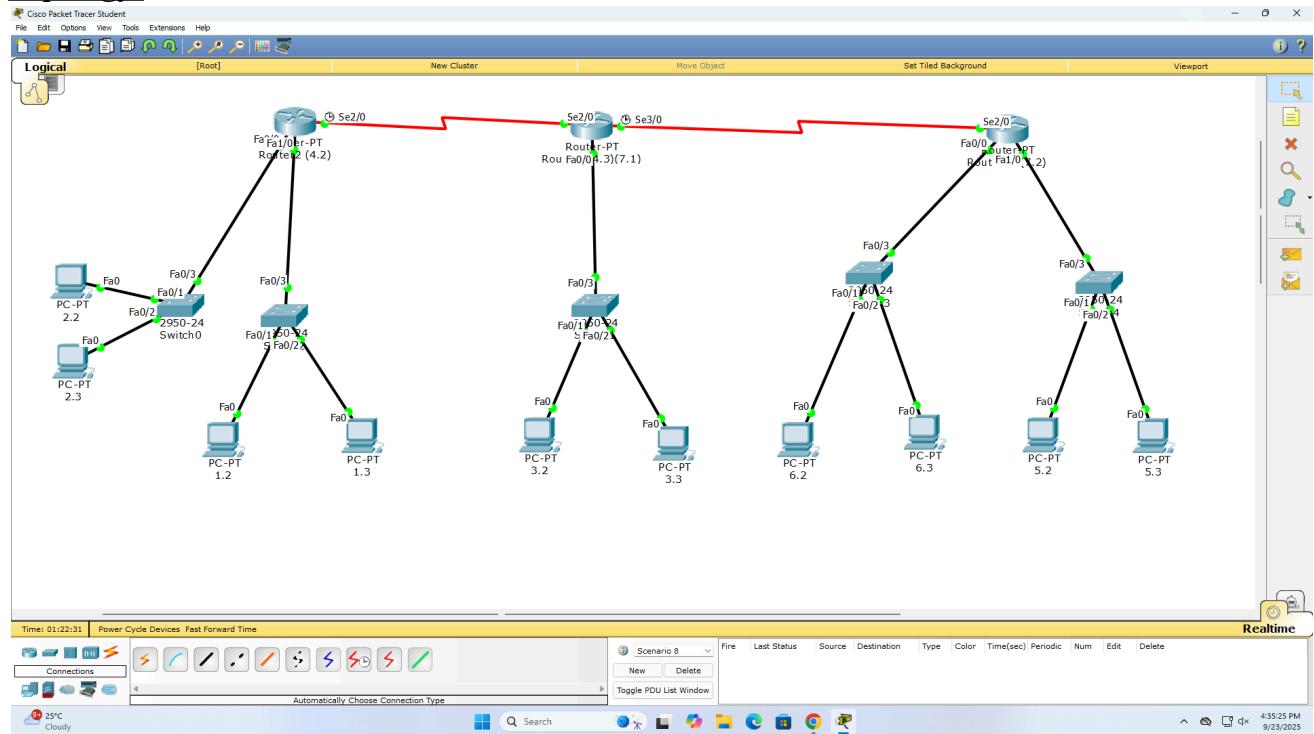
### Observation:



## Program 6

**Aim:** Configure RIP routing Protocol in Routers

### **Topology:**



### **Procedure:**

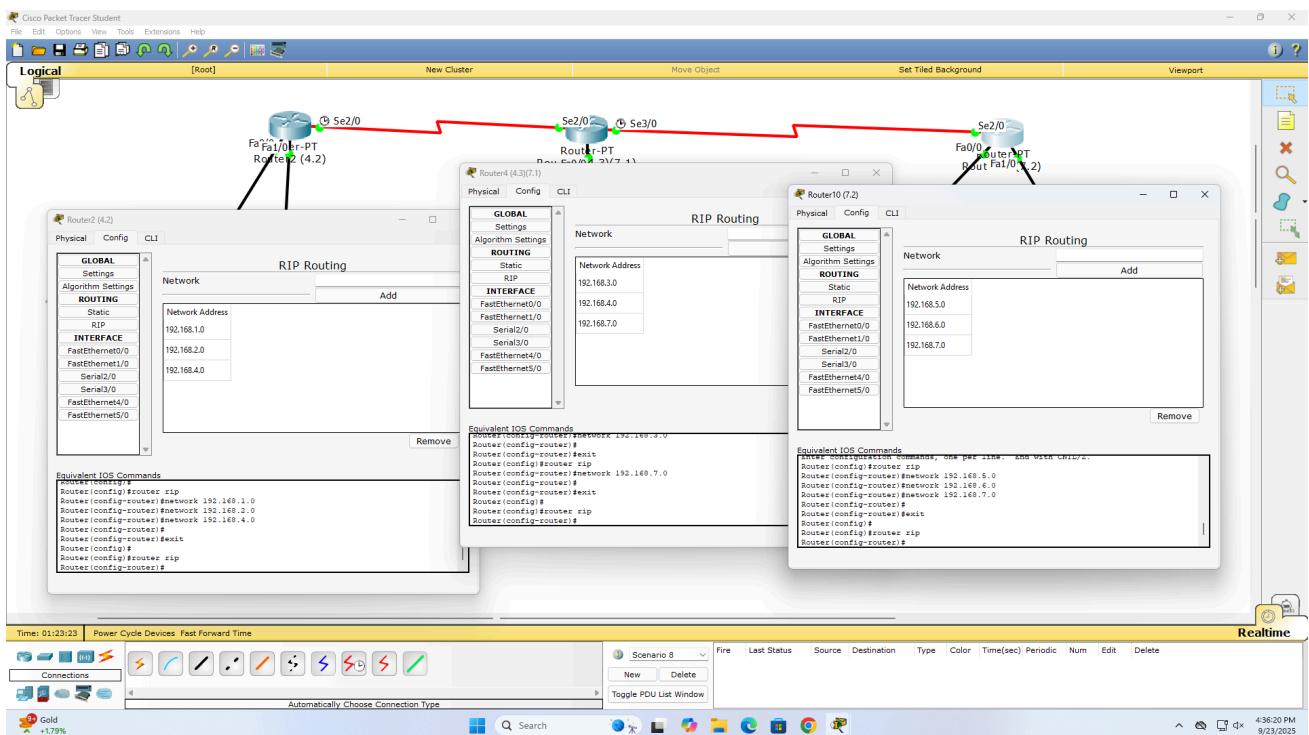
#### 1. Configure Interfaces:

- Go to: Config tab → INTERFACE
- FastEthernet0/0:
  - IP Address: 192.168.3.1
  - Subnet Mask: 255.255.255.0
- FastEthernet1/0:
  - IP Address: 192.168.4.1
  - Subnet Mask: 255.255.255.0
- Serial2/0:
  - IP Address: 192.168.7.1
  - Subnet Mask: 255.255.255.0

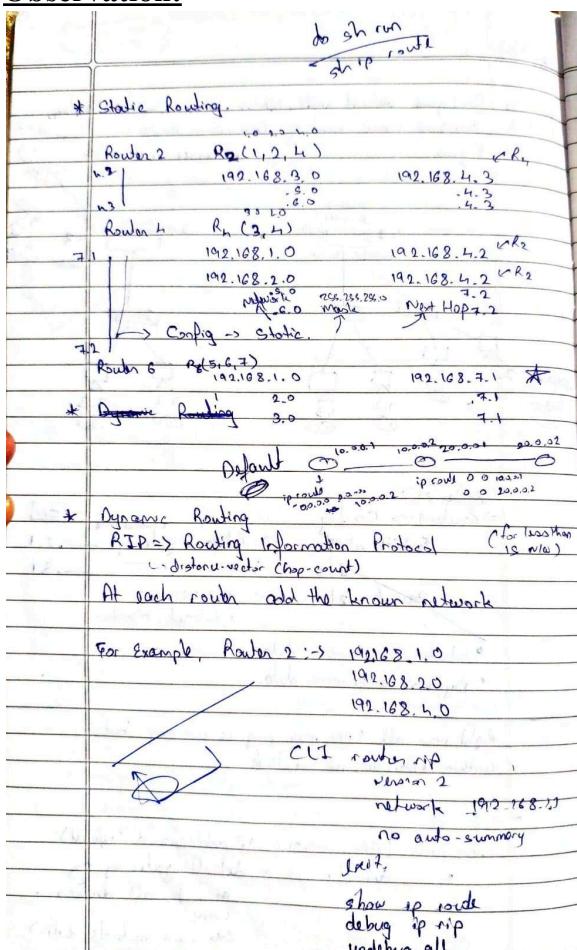
#### 2. Configure RIP Routing:

- Go to: Config tab → ROUTING → RIP
- Settings:
  - Version: RIP Version 2
- Network Entries:
  - Click Add → Network: 192.168.3.0
  - Click Add → Network: 192.168.4.0
  - Click Add → Network: 192.168.7.0

## Output:



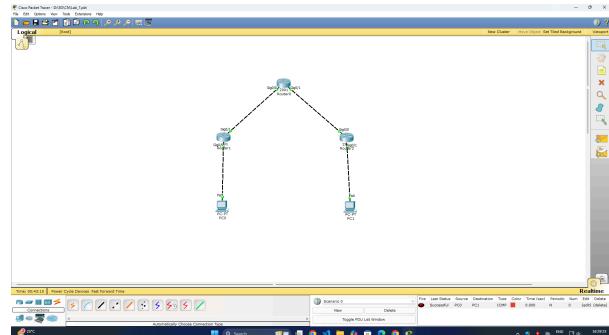
## Observation:



## Program 7

Aim: Configure OSPF routing protocol

### Topology:



### Procedure:

Router> enable

Router# conf t

Router1(config)# router ospf 1

Router1(config-router)# network 192.168.44.0 0.0.0.255 area 0

Router1(config-router)# exit

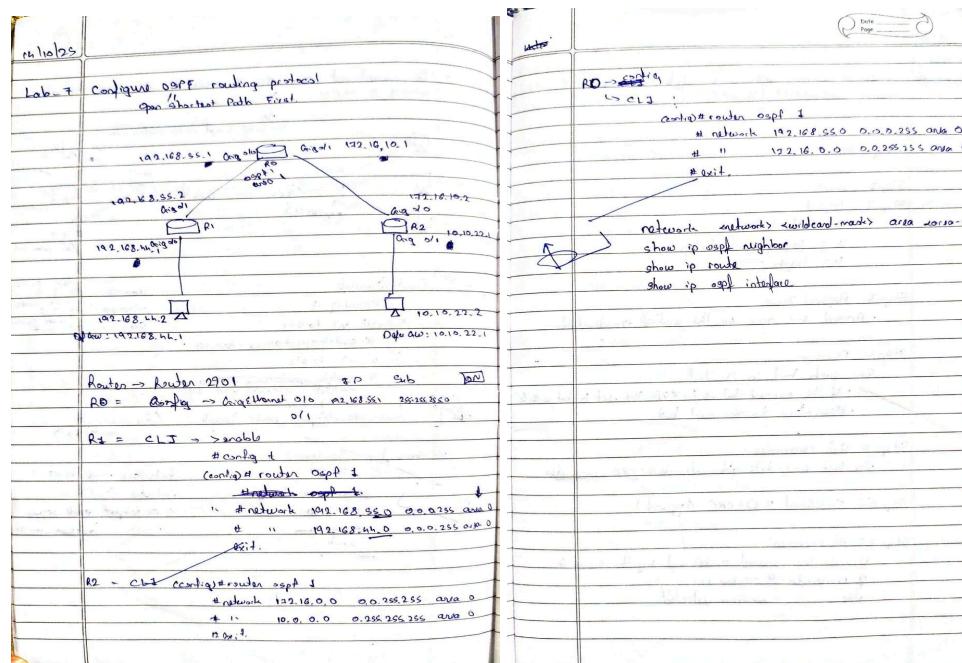
Router1(config)# exit

Router1# write memory

### Output:

Fire	Last Status	Source	Destination	Type	Color	Time/ser	Periodic	Num	Edit	Delete
Successful		PC0	PC1	ICMP	pink	0.000	N	0	(edit)	
Successful		PC0	Router2	ICMP	purple	0.000	N	1	(edit)	
Successful		PC0	Router0	ICMP	purple	0.000	N	2	(edit)	
Successful		Router0	PC1	ICMP	green	0.000	N	3	(edit)	

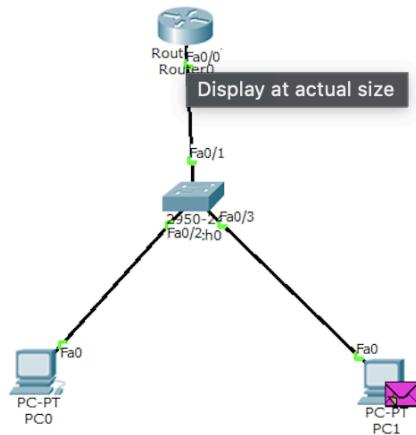
### Observation:



## Program 8

**Aim:** Demonstrate the TTL/ Life of a Packet

### Topology:



### Procedure:

1. Enable Packet Capture:
  - Right-click on a link between devices
  - Select "Capture / Forward"
  - Send ping from PC19 to PC20
2. Analyze TTL in Captured Packets:
  - Open "Event List" in Simulation mode
  - Double-click any ICMP packet
  - Go to "Inbound PDU Details" or "Outbound PDU Details"
  - Check "IP" section → "Time to Live" field

### Output:

PDU Information at Device: PC1

OSI Model   Inbound PDU Details

PDU Formats

Ethernet II

0	4	8	14	19	Bytes
PREAMBLE: 101010...1011		DEST MAC: 00E0.A32A.E118		SRC MAC: 0060.5CAE.11C2	
TYPE: 0x800		DATA (VARIABLE LENGTH)		FCS: 0x0	

IP

0	4	8	16	19	31 Bits
4 IHL DSCP: 0x0		TL: 28			
ID: 0x3		0x0	0x0		
TTL: 128	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: 0x0		CODE: 0x0	CHECKSUM

Vis.   Time(sec)   Last Device   At Device   Type   Info

Vis.	Time(sec)	Last Device	At Device	Type	Info
	0.003	PC0	Switch0	ARP	
	0.004	Switch0	PC0	ICMP	
	0.004	Switch0	PC1	ARP	
	0.004	--	PC1	ICMP	
	0.005	PC1	Switch0	ICMP	
	0.006	Switch0	PC0	ICMP	
	0.007	PC0	Switch0	ICMP	
	0.008	Switch0	PC1	ICMP	

## Observation:

11/12/25

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

Lab-8 Demonstrate the TTL / Life of a Packet

Figure:

~~Operating systems TTL's~~ → 8 bit IP header

Windows : 128      Once the TTL reach 0,  
Linux : 64            the packet is dropped  
Cisco : 255  
Solaris : 255

⇒ Go to Simulation Model  
Do Auto Capture/Play OR Capture/Erase

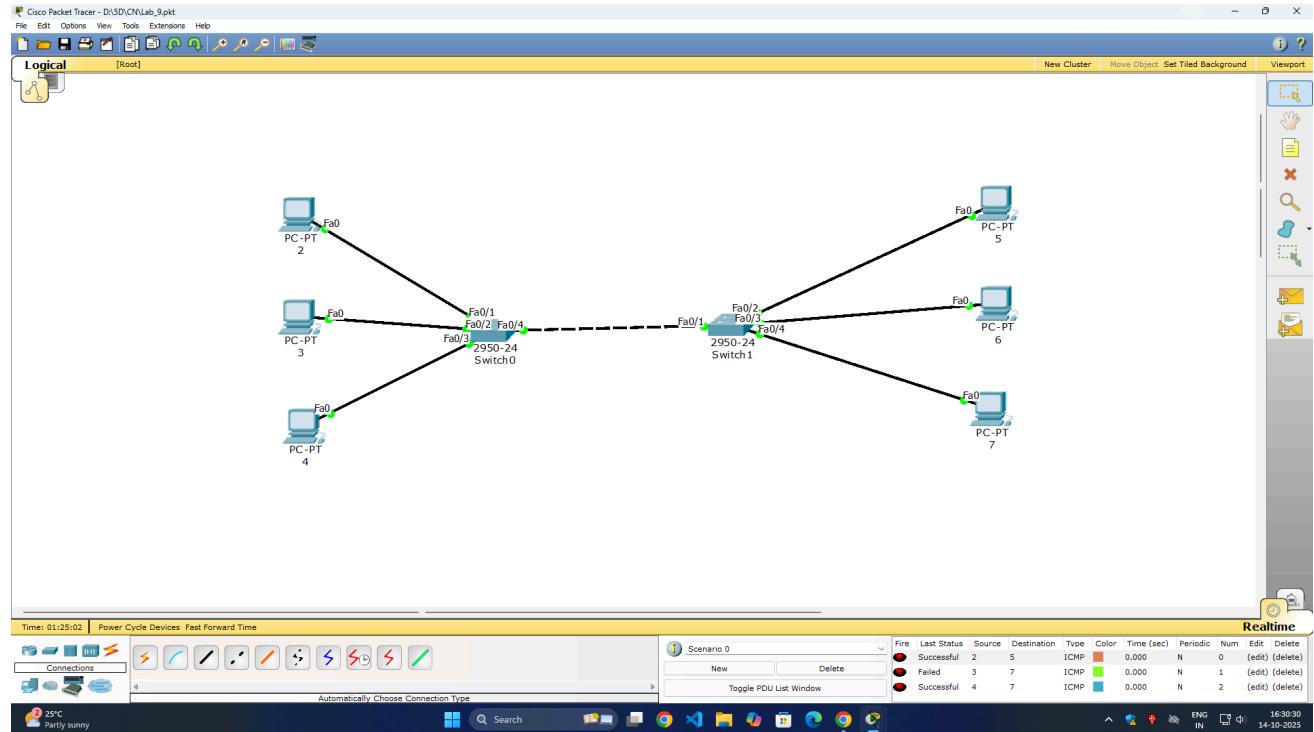
→ Click on message to see the PDU information at Daud.  
It includes OSI Model

Lifespan, Hop Count, OS

## Program 9

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

### Topology:



### Procedure:

```
Switch> enable
Switch# configure terminal
Switch(config)# interface fastethernet 0/1
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
Switch(config-if)# no shutdown
Switch(config-if)# exit
Switch(config)# interface fastethernet 0/2
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
Switch(config-if)# no shutdown
Switch(config-if)# exit
Switch(config)# interface fastethernet 0/4
Switch(config-if)# switchport mode trunk
Switch(config-if)# no shutdown
Switch(config-if)# exit
Switch(config)# exit
Switch# write memory
```

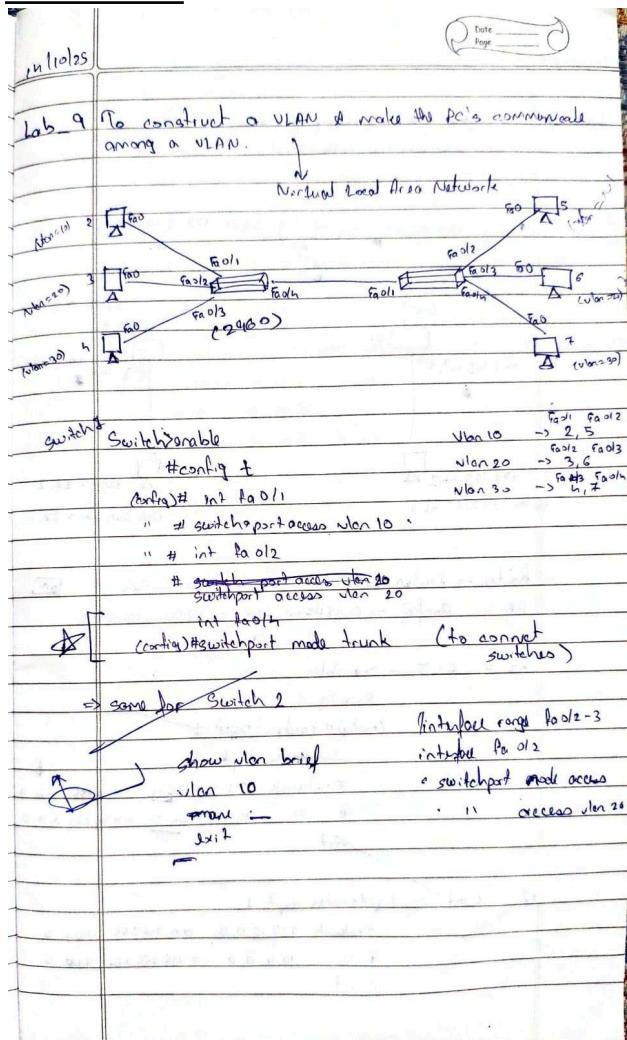
## Output:

```
Switch(config-if)#switchport access vlan 10
% Access VLAN does not exist. Creating vlan 10
Switch(config-if)#int fa0/2
Switch(config-if)#switchport access vlan 20
% Access VLAN does not exist. Creating vlan 20
Switch(config-if)#int fa0/3
Switch(config-if)#switchport access vlan 30
% Access VLAN does not exist. Creating vlan 30
Switch(config-if)#int fa0/4
Switch(config-if)#switchport mode trunk

Switch(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/4, changed state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/4, changed state to up

Switch(config-if)#exit
Switch(config)#
```

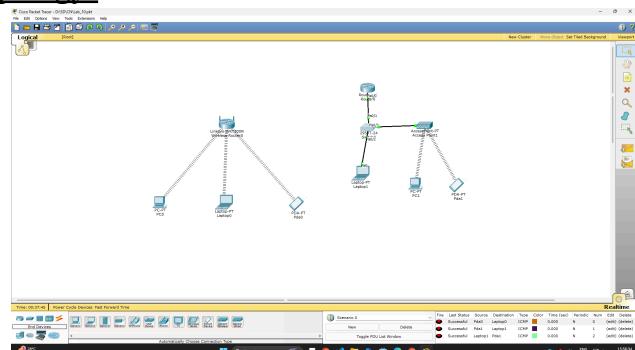
## Observation:



## Program 10

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

### Topology:



### Output:

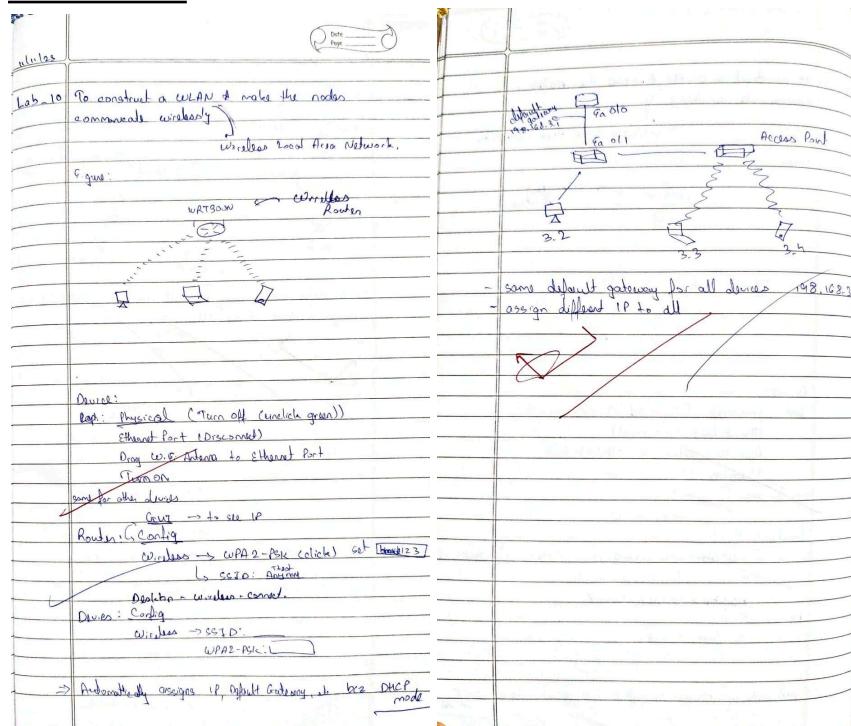
**Wireless Router Configuration (Left Window):**

- SSID:** Ttest
- Channel:** 6
- Authentication:** WPA2-PSK (selected)
- Pass Phrase:** bmscse123
- RADIUS Server Settings:** IP Address and Shared Secret fields are empty.
- Encryption Type:** AES

**Laptop0 Configuration (Right Window):**

- MAC Address:** 000D.BD93.391C
- SSID:** Ttest
- Bandwidth:** 300 Mbps
- Authentication:** WPA2-PSK (selected)
- Pass Phrase:** bmscse123
- User ID:** (empty)
- Password:** (empty)
- Encryption Type:** AES
- IP Configuration:** (empty)

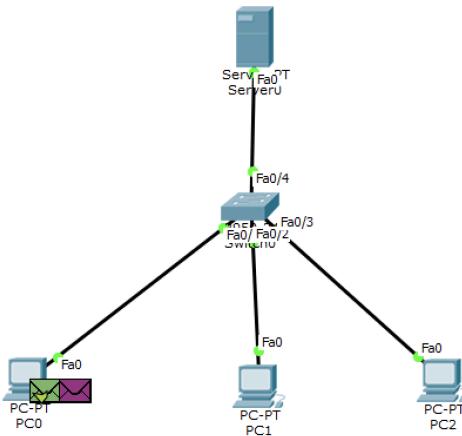
### Observation:



## Program 11

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

### Topology:



### Procedure:

ping 192.168.1.20

arp -a

Interface: 192.168.1.10

Internet Address	Physical Address	Type
192.168.1.20	00-50-56-xx-xx-xx	dynamic
192.168.1.30	00-50-56-yy-yy-yy	dynamic

### Output:

The screenshot shows the Cisco Packet Tracer interface with the following details:

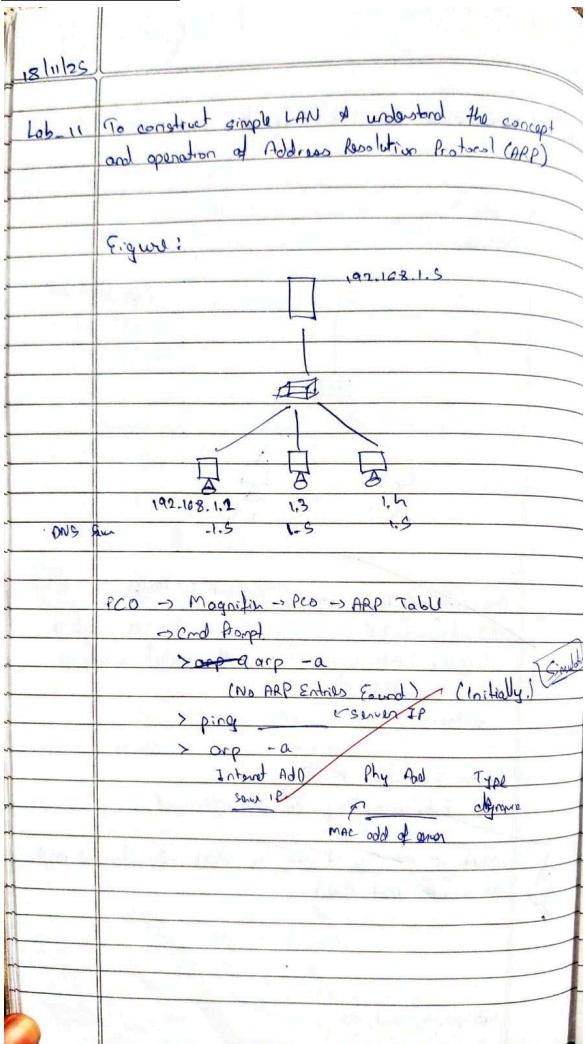
- Logical View:** Displays the network topology with the server, switch, and three PCs.
- Event List Panel:** Shows a timeline of events:

Vis.	Time(sec)	Last Device	At Device	Type	Info
0.000	--	PC0	PC0	ICMP	
0.000	--	PC0	PC0	ARP	
0.001	PC0	Switch0	ARP		
0.002	Switch0	PC1	ARP		
0.002	Switch0	PC2	ARP		
0.002	Switch0	Server0	ARP		
0.003	PC1	Switch0	ARP		
0.004	Switch0	PC0	ARP		
0.004	--	PC0	ICMP		
- ARP Table for Server0:**

IP Address	Hardware Address	Interface
192.168.1.2	0001:40C:CS2A	FastEthernet0
- ARP Table for PC0:**

IP Address	Hardware Address	Interface
192.168.1.3	0001:40C:CS2A	FastEthernet0
192.168.1.4	0010:1179:CEC8	FastEthernet0
192.168.1.5	0090:2BEE:5598	FastEthernet0
- Bottom Panel:** Shows the time as 00:13:58.938, power cycle devices, play controls (Back, Auto Capture / Play, Capture / Forward), scenario selection (Scenario 0), and a status bar indicating "Light rain At night" and the date/time 11/18/2025 3:29:31 PM.

## Observation:



PCO → Magazine → PCO → ARP Table

→ Cmd Prompt

> ~~arp -a~~

(No ARP Entries Found) (Initially)

> ping →

server IP

> arp -a

Internet Add

same IP

Phy Add

dynamic

## Observation:

To find MAC add of any device

- I ping from PCO to Server

ARP

ARP

It will distribute msg to all connection switch  
- It will reach server IP, take its MAC add  
bring it back to PCO  
then it'll update ARP for both server & PCO

PCO (ARP)

Server (ARP)

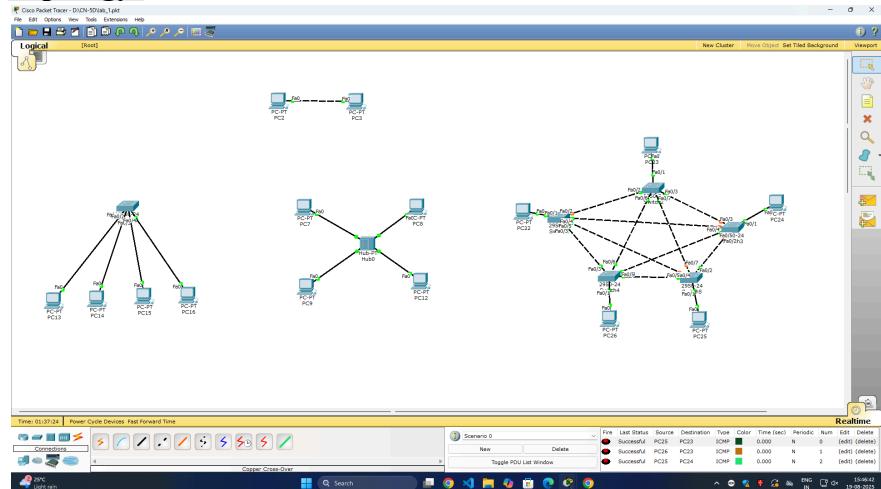
IP add H/w add reply IP add H/w add reply

192.168.1.5 00:0c:29:ff:ff:ff 192.168.1.2 00:0c:29:ff:ff:ff

## Program 12

**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.

### **Topology:**



### **Output:**

Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit	Delete
Successful		PC25	PC23	ICMP	Green	0.000	N	0	(edit) (delete)	
Successful		PC26	PC23	ICMP	Brown	0.000	N	1	(edit) (delete)	
Successful		PC25	PC24	ICMP	Green	0.000	N	2	(edit) (delete)	

### **Observation:**

Lab-12 Create a topology & simulate sending a simple PDU from source to destination using hub & switch as connecting devices & demonstrate a ping message.

1) Communication between PC

- Create two PC: PC0, PC1
- Connect them using copper wire (cross-over)
- PC - setup → IP add: 192.168.1.1 - 192.168.1.2
- Test it by adding sample PDU & check status.

2) Communication between PC via Hub. → Hub

- Create three PCs (PC0, PC1, PC2) & a Hub
- Connect PCs & Hub via copper wire (straight-through)
- Assign different IP address to each PC & subnet mask.
- To test, add a sample PDU from one PC (PC0) to another.

3) Communication between PCs via switch. → 2960

- Create two PCs (PC0, PC1, PC2) & a switch.
- Connect it by copper (straight-through)
- Set diff IP address for PCs & subnetmask
- Connect to switch on specific port.
- Ping any two PCs with PDU & check status.

Observations:

- PC - PC : Crossover Wires
- PC - Switch/Hub : Straight Wires
- Assign unique IP to each PC in network & general Subnet mask
- Switch has ports to connect device

Ping: (give unique IP to each PC)  
open cmd prompt of 1 PC > ping & IP (multiple)  
Protocol Data Unit  
Broadcast  
Edit with tempo  
Select msg  
Drag & drop

## Program 14

**Aim:** 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);
        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;
    }
    for (; count != 0; i++) {
        if (count > cap) {
            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:

```

Enter the bucket size:
10
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

```

## Observation:

CYCLE - 2  
 Lab - 14  
 Write a program for congestion control using token bucket algorithm.

Algorithm

Step 1 : Initials  
 - cap ← input bucket capacity  
 - process ← packet processing rate  
 - time ← simulation time.

Step 2 : For each second  
 - Add incoming packet : count += inpt[i].  
 if (count > cap)  
 compute dropped packet : drop = count - cap

Step 3 : Process packets  
 - send packets send = min(count, process)  
 - Subtract sent packet : count -= send.

Step 4 : Display statistics  
 For each second  
 print time, packet received, sent, left, & dropped.  
 and for

Step 5 : After all seconds.  
 - continue drawing bucket until count = 0.  
 - print final statistics for remaining seconds

$\Rightarrow$  drop = count - cap (when overflow occurs)  
 $\Rightarrow$  sent = min(count, process)  
 $\Rightarrow$  count = count - send

## Program 15

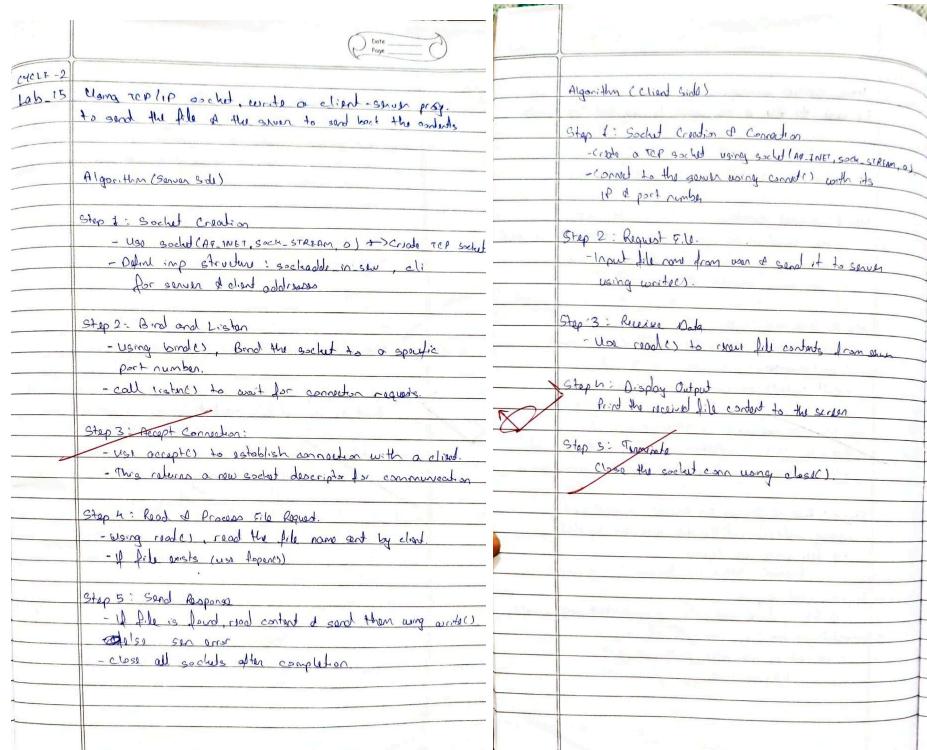
**Aim:** Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

### Algorithm:

<pre># tcp_client.py import socket # Step 1: Create TCP socket client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # Step 2: Connect to server client_socket.connect(('localhost', 8080)) # Step 3: Send filename filename = input("Enter filename to request: ") client_socket.send(filename.encode()) # Step 4: Receive file contents data = client_socket.recv(4096).decode() print("\n--- File Content ---\n") print(data) # Step 5: Close connection client_socket.close()</pre>	<pre># tcp_server.py import socket # Step 1: Create a TCP socket server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # Step 2: Bind to address and port server_socket.bind(('localhost', 8080)) # Step 3: Listen for client connections server_socket.listen(1) print("Server is listening on port 8080...") # Step 4: Accept connection conn, addr = server_socket.accept() print("Connected by:", addr) # Step 5: Receive file name filename = conn.recv(1024).decode().strip() try:     # Step 6: Open and read file with open(filename, 'r') as f:     data = f.read()     conn.send(data.encode()) # Send file contents except FileNotFoundError: conn.send(b"File not found on server.") # Step 7: Close connection conn.close() server_socket.close()</pre>
---	--

### Output:

### Observation:



### Algorithm (Client Side)

Step 1: Socket Creation & Connection

- Create a TCP socket using `socket(AF_INET, socket.SOCK_STREAM)`
- Connect to the server using `connect()` with its IP & port number

### Step 2: Request File

- Input file name from user & send it to server using `write()`.

### Step 3: Receive Data

- Use `read()` to read file contents from server.

### Step 4: Display Output

- Print the received file content to the screen.

### Step 5: Terminate

- Close the socket connection using `close()`.

## Program 16

**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.

### Algorithm:

<pre># tcp_client.py import socket # Step 1: Create TCP socket client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # Step 2: Connect to server client_socket.connect(('localhost', 8080)) # Step 3: Send filename filename = input("Enter filename to request: ") client_socket.send(filename.encode()) # Step 4: Receive file contents data = client_socket.recv(4096).decode() print("\n--- File Content ---\n") print(data) # Step 5: Close connection client_socket.close()</pre>	<pre># tcp_server.py import socket # Step 1: Create a TCP socket server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # Step 2: Bind to address and port server_socket.bind(('localhost', 8080)) # Step 3: Listen for client connections server_socket.listen(1) print("Server is listening on port 8080...") # Step 4: Accept connection conn, addr = server_socket.accept() print("Connected by:", addr) # Step 5: Receive file name filename = conn.recv(1024).decode().strip() try:     # Step 6: Open and read file with open(filename, 'r') as f:     data = f.read()     conn.send(data.encode()) # Send file contents except FileNotFoundError: conn.send(b"File not found on server.") # Step 7: Close connection conn.close() server_socket.close()</pre>
---	--

### Observation:

MCAE-2  
Lab-16. Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

Algorithm (Server Side)

Step 1: Socket Creation  
- Create a UDP socket using `socket(AF_INET, SOCK_DGRAM, 0)`  
- Initialize sockaddr\_in\_sau structure & bind it to a port.

Step 2: Wait for File Request:  
- Use a `recvfrom()` to receive the file name from the client.

Step 3: Process Request:  
- Open the file using `open()`.  
- If found, read contents & send using `sendto()`.

Step 4: Handle missing file:  
- If file not found send msg "file not found".

Step 5: Terminating  
- Close the socket after completing transmission.

Algorithm (Client Side)  
C4 L4

Step 1: Socket Creation  
- Create a UDP socket with `socket(AF_INET, SOCK_DGRAM, 0)`

Step 2: Send Request.  
- Input file name from user & send it to server using `sendto()`.

Step 3: Receive Response  
- Use `recvfrom()` to receive data packets from server.

Step 4: Display Output:  
- Print file contents received on the screen.

Step 5: Close connection  
- Terminate communication & close socket.

X X

## Program 17

**Aim:** 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:\n");
    gets(gen);
    printf("Generator polynomial is CRC-CCITT: %s\n", gen);
    genlen = strlen(gen);
    k = genlen - 1;
    printf("Enter the message:\n");
    n = 0;
    while ((c = getchar()) != '\n') {
        msj[n] = c;
        n++;
    }
    msj[n] = '\0';

    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);
    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++;
            }
        }
    }

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

    printf("Checksum (remainder):\n");
    puts(rem);

    printf("\nMessage 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);

    n = 0;
    printf("Enter the received message:\n");
    while ((c = getchar()) != '\n') {
        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';
    }

for (i = 0; i < k; i++)
    rem[i] = s[n + i];
rem[k] = '\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 contains error \n");

return 0;
}

```

## Output:

```

Enter the generation polynomial:
101
Generator polynomial is CRC-CCITT: 101
Enter the message:
1101010101010100

Message polynomial appended with zeros:
110101010101010000
Checksum (remainder):
11

Message with checksum appended:
110101010101010011
Enter the received message:
110101010101010011
Received polynomial is error-free

Process returned 0 (0x0)   execution time : 33.192 s

```

## Observation:

