

# **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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



## **LAB RECORD**

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

*Submitted by*

**Harshith B (1BM23CS409)**

*in partial fulfillment for the award of the degree of*

**BACHELOR OF ENGINEERING  
in  
COMPUTER SCIENCE AND ENGINEERING**



**B.M.S. COLLEGE OF ENGINEERING**

**(Autonomous Institution under VTU)**

**BENGALURU-560019**

**Academic Year 2024-25 (odd)**

# B.M.S. College of Engineering

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

## Department of Computer Science and Engineering



### CERTIFICATE

This is to certify that the Lab work entitled “Computer Network (23CS5PCCON)” carried out by **Harshith B (1BM23CS409)** 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.

|   |  |
|---|--|
| Spoorthi D M<br>Assistant Professor<br>Department of CSE, BMSCE | Dr. Kavitha Sooda<br>Professor & HOD<br>Department of CSE, BMSCE |
|---|--|

## **Index**

### **(Cycle – 1)**

| <b>Sl.<br/>No.</b> | <b>Date</b> | <b>Experiment Title</b>   | <b>Page No.</b> |
|--------------------|-------------|---|-----------------|
| 1                  | 1-10-24     | Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message. | 1-4             |
| 2                  | 15-10-24    | Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply     | 5-8             |
| 3                  | 22-10-24    | Configure default route, static route to the Router   | 9-15            |
| 4                  | 12-11-24    | Configure DHCP within a LAN and outside LAN.  | 16-22           |
| 5                  | 18-11-24    | Configure RIP routing Protocol in Routers   | 23-26           |
| 6                  | 18-11-24    | Configure OSPF routing Protocol   | 27-32           |
| 7                  | 26-11-24    | Demonstrate the TTL/ Life of a Packet   | 33-36           |
| 8                  | 12-11-24    | Configure Web Server, DNS within a LAN.   | 37-39           |
| 9                  | 26-11-24    | To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)   | 40-42           |
| 10                 | 29-10-24    | To understand the operation of TELNET by accessing the router in server room from a PC in IT office.  | 43-45           |
| 11                 | 3-12-24     | To construct a VLAN and make the PC's communicate among a VLAN  | 46-49           |
| 12                 | 26-11-24    | To construct a WLAN and make the nodes communicate wirelessly   | 50-52           |

## **Index**

### **(Cycle – 2)**

| <b>Sl.<br/>No.</b> | <b>Date</b> | <b>Experiment Title</b>   | <b>Page No.</b> |
|--------------------|-------------|---|-----------------|
| 13                 | 17-12-24    | Write a program for error detecting code using CRC-CCITT (16-bits).   | 53-55           |
| 14                 | 17-12-24    | Write a program for congestion control using Leaky bucket algorithm.  | 56-58           |
| 15                 | 24-12-24    | 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. | 59-60           |
| 16                 | 24-12-24    | 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.    | 61-62           |
| 5                  | 18-11-24    | Configure RIP routing Protocol in Routers   | 63              |

Github Link: [GitHub Link](#)

## **Program 1**

To demonstrate the transmission of a simple PDU between 2 devices connected using a hub and a switch

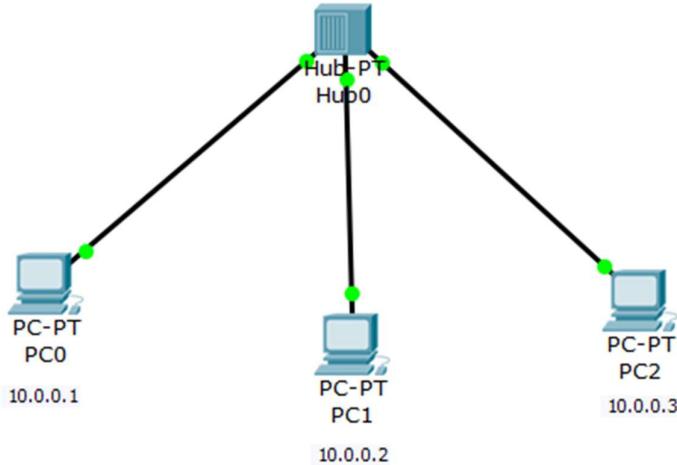


Figure 1: Using Hub

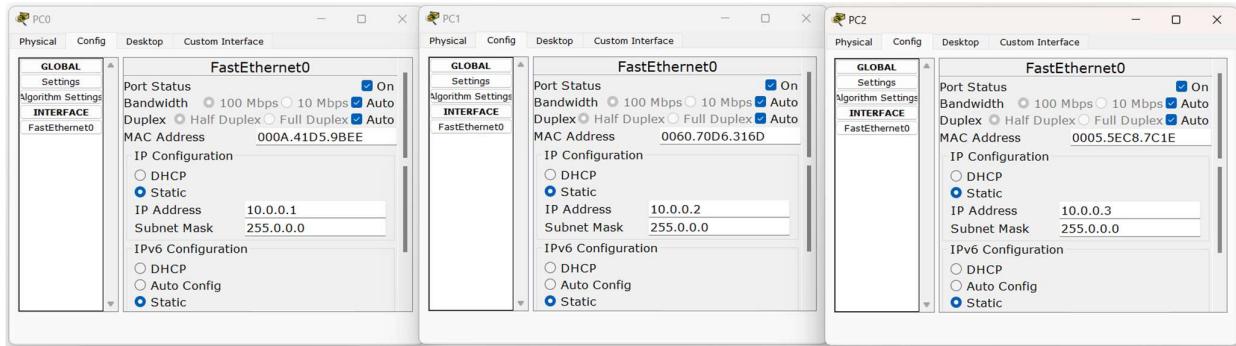


Figure 2: IP Addresses of the 3 PCs

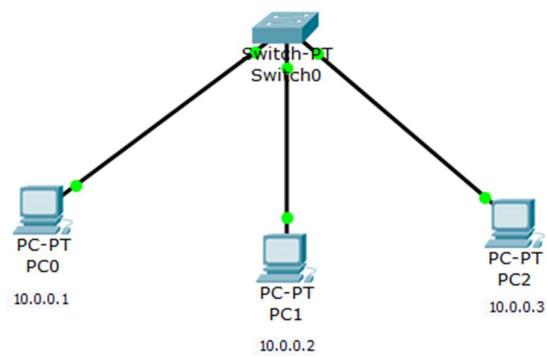


Figure 3: Using Switch

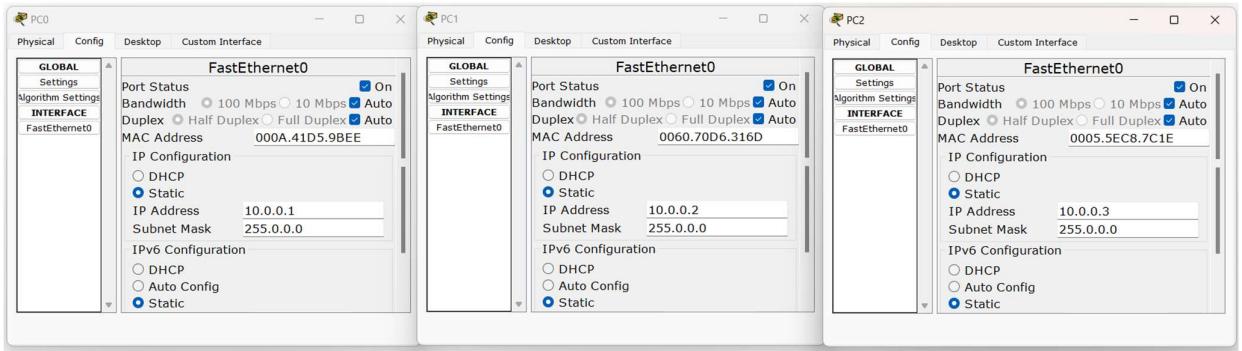


Figure 4: IP Addresses of the 3 PCs

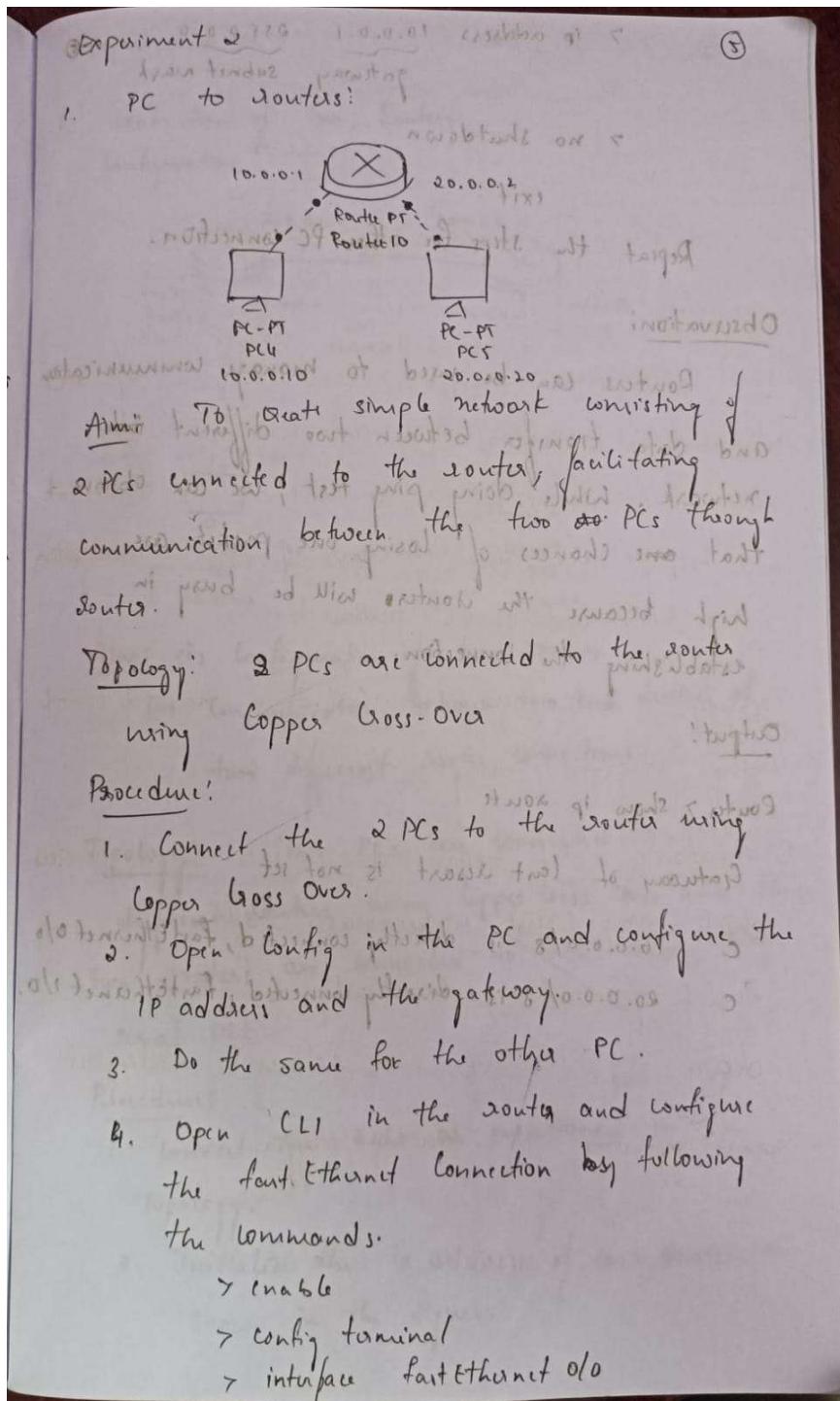
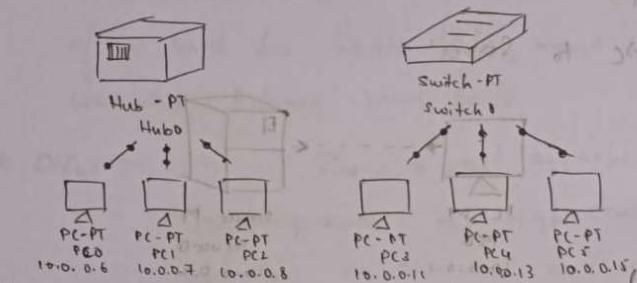


Figure 5: Observation Book 1

## 2. Hub & switch:



Aim: To create simple network consisting of 3 PCs connected to a central Hub and another network with 3 PCs connected to a switch. This connection will help observe the behaviour of data transmission using hub and switch devices.

Topology: 3 PCs are connected to a hub & switch using straight-through ethernet cables.

Observation: Hub broadcasts packets to all devices which may cause unnecessary traffic.  
Switch forwards packets only to appropriate device by learning MAC addresses, making it more efficient in reducing traffic.

Figure 6: Observation Book 2

## **Program 2**

To demonstrate configuration of IP addresses to the Routers and explore ping command.

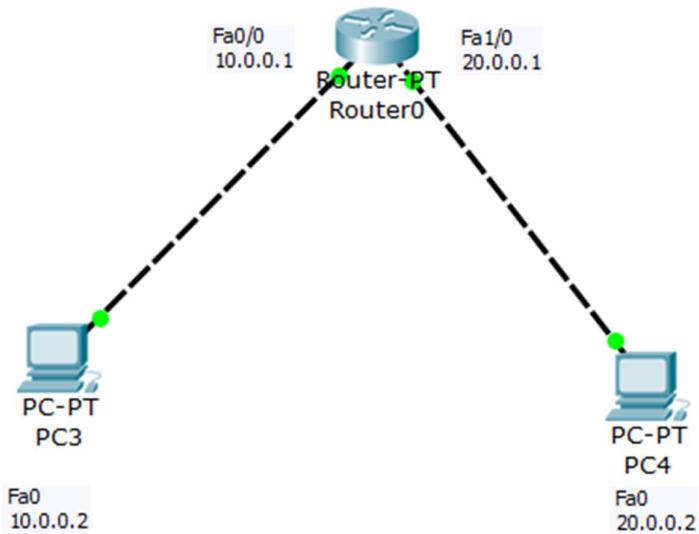


Figure 7: Topology

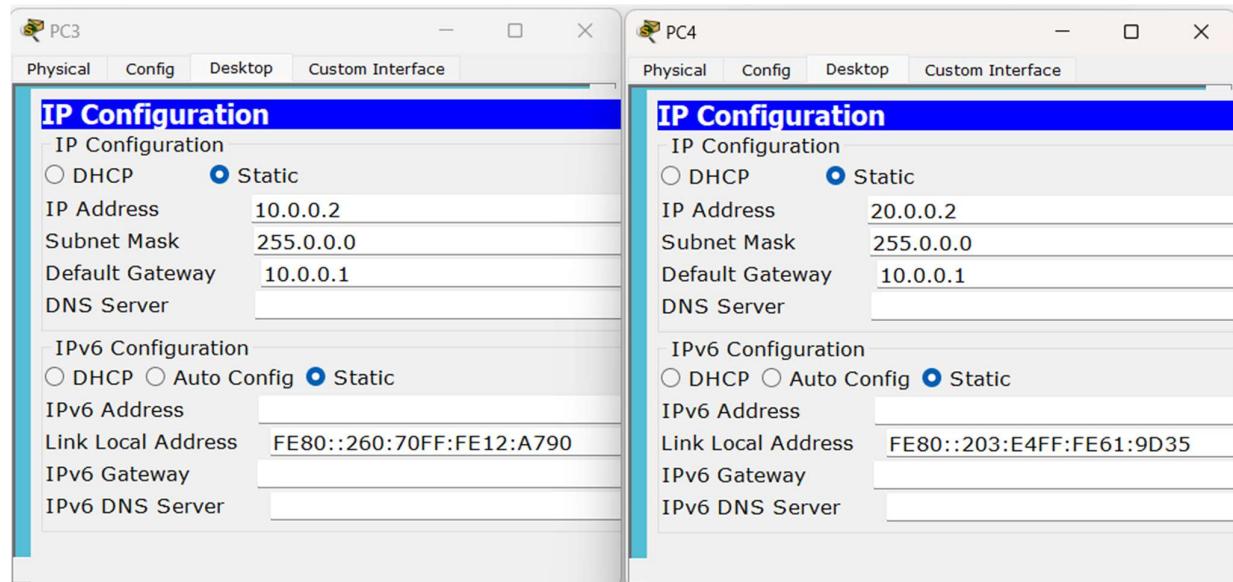


Figure 8: IP addresses

Figure 9: Router Fa0/0

Figure 10: Router Fa1/0

```

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

Pinging 10.0.0.2 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 10.0.0.2:
  Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>ping 10.0.0.2

Pinging 10.0.0.2 with 32 bytes of data:
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255

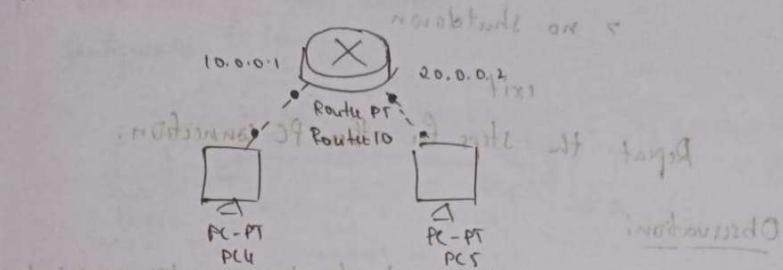
Ping statistics for 10.0.0.2:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
  Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
PC>

```

Figure 11: ping command output

Experiment 2 7.7.0 3.0.0.03 enables us  
to learn about routers

1. PC to routers:



Aim: To create a simple network consisting of 2 PCs connected to the router, facilitating communication between the two PCs through the router.

Topology: 2 PCs are connected to the router

using Copper Cross-over

Procedure:

1. Connect the 2 PCs to the router using Copper Cross over.
2. Open Config in the PC and configure the IP address and gateway.
3. Do the same for the other PC.
4. Open CLI in the router and configure the fast Ethernet connection by following the commands:
  - > enable
  - > config terminal
  - > interface fastEthernet 0/0

Figure 12: Observation book 1

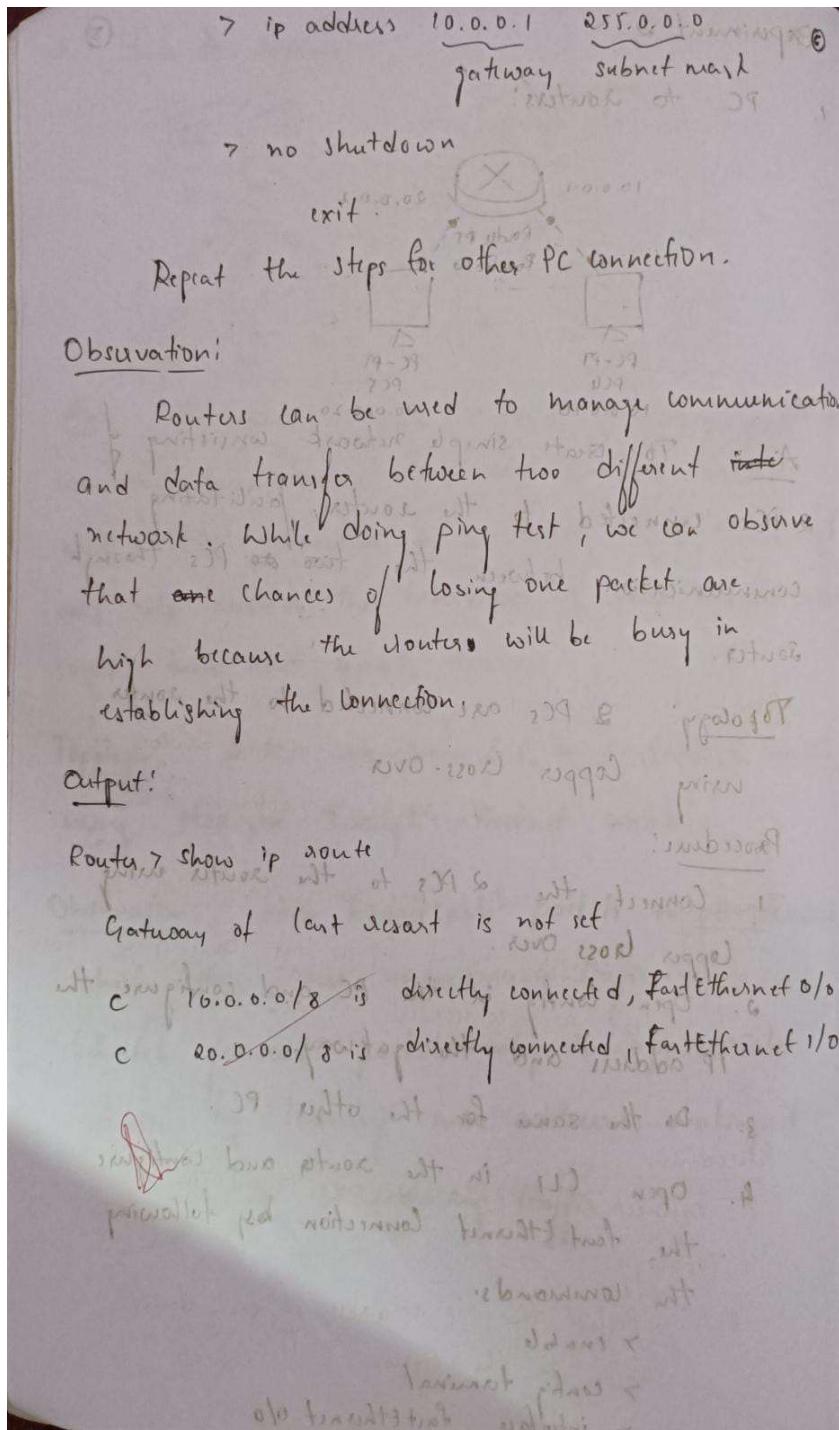


Figure 13: Observation Book 2

### **Program 3**

To demonstrate configuration of default and static routes through a connection of routers.

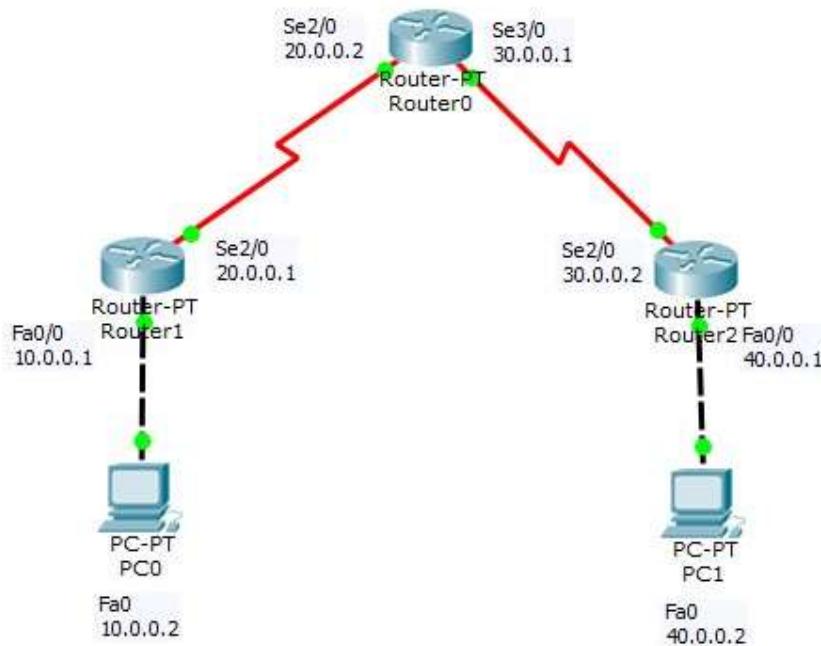


Figure 14: Topology for static route

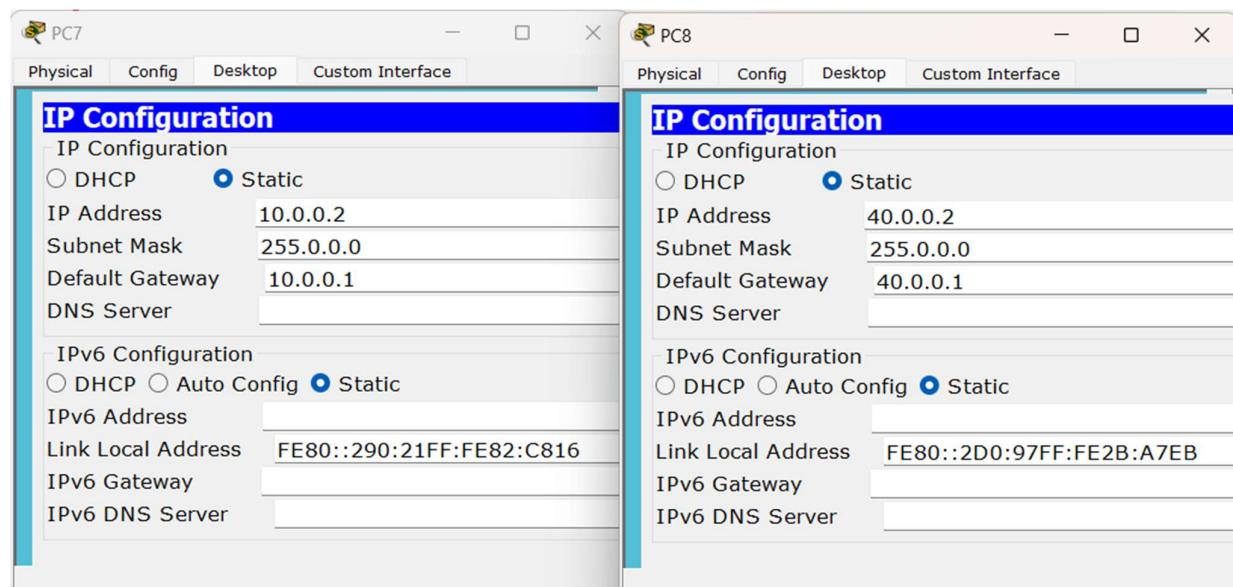


Figure 15: IP addresses

```

Router6# 
Router6# config t
Router6(config)# interface fastethernet0/0
Router6(config-if)# no shutdown
Router6(config-if)# ip address 20.0.0.1 255.0.0.0
Router6(config-if)# exit
Router6(config)# exit
Router6(config)# interface serial1/0
Router6(config-if)# no shutdown
Router6(config-if)# exit
Router6(config)# exit
Router6(config)# ip route 20.0.0.0 255.0.0.0 20.0.0.2
Router6(config)# ip route 30.0.0.0 255.0.0.0 30.0.0.2
Router6(config)# ip route 40.0.0.0 255.0.0.0 20.0.0.2
Router6(config)#

```

```

Router5# 
Router5# config t
Router5(config)# interface serial2/0
Router5(config-if)# no shutdown
Router5(config-if)# ip address 30.0.0.1 255.0.0.0
Router5(config-if)# exit
Router5(config)# exit
Router5(config)# interface serial3/0
Router5(config-if)# no shutdown
Router5(config-if)# exit
Router5(config)# exit
Router5(config)# ip route 10.0.0.0 255.0.0.0 20.0.0.1
Router5(config)# ip route 40.0.0.0 255.0.0.0 30.0.0.2
Router5(config)# ip route 20.0.0.0 255.0.0.0 20.0.0.1
Router5(config)#

```

```

Router7# 
Router7# config t
Router7(config)# interface fastethernet0/0
Router7(config-if)# no shutdown
Router7(config-if)# ip address 40.0.0.1 255.0.0.0
Router7(config-if)# exit
Router7(config)# exit
Router7(config)# interface serial0/0
Router7(config-if)# no shutdown
Router7(config-if)# exit
Router7(config)# exit
Router7(config)# ip route 30.0.0.0 255.0.0.0 30.0.0.1
Router7(config)# ip route 20.0.0.0 255.0.0.0 30.0.0.1
Router7(config)# ip route 40.0.0.0 255.0.0.0 30.0.0.1
Router7(config)# ip route 40.0.0.1 255.0.0.0 40.0.0.1
Router7(config)#

```

Figure 16: Routers CLI

```

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=lms 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 = lms, Average = 0ms

PC>ping 20.0.0.1
Pinging 20.0.0.1 with 32 bytes of data:
Reply from 20.0.0.1: bytes=32 time=0ms TTL=255

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

PC>ping 30.0.0.1
Pinging 30.0.0.1 with 32 bytes of data:
Reply from 30.0.0.1: bytes=32 time=4ms TTL=254
Reply from 30.0.0.1: bytes=32 time=lms TTL=254
Reply from 30.0.0.1: bytes=32 time=4ms TTL=254
Reply from 30.0.0.1: bytes=32 time=3ms TTL=254

Ping statistics for 30.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = lms, Maximum = 4ms, Average = 3ms

PC>ping 40.0.0.2
Pinging 40.0.0.2 with 32 bytes of data:
Request timed out.
Reply from 40.0.0.2: bytes=32 time=6ms TTL=125
Reply from 40.0.0.2: bytes=32 time=4ms TTL=125
Reply from 40.0.0.2: bytes=32 time=4ms TTL=125

Ping statistics for 40.0.0.2:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 4ms, Maximum = 6ms, Average = 4ms

PC>

```

Figure 17: Output

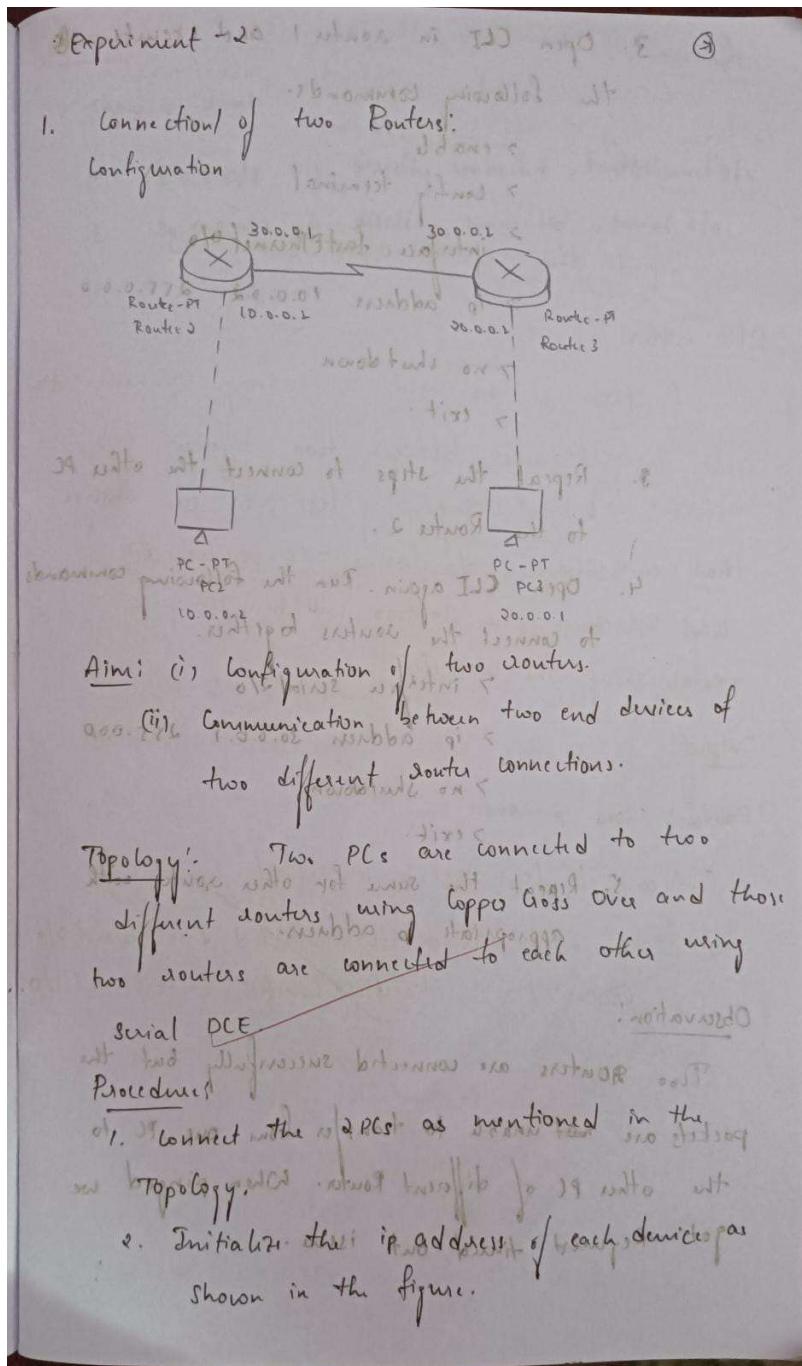


Figure 18: Observation book 1

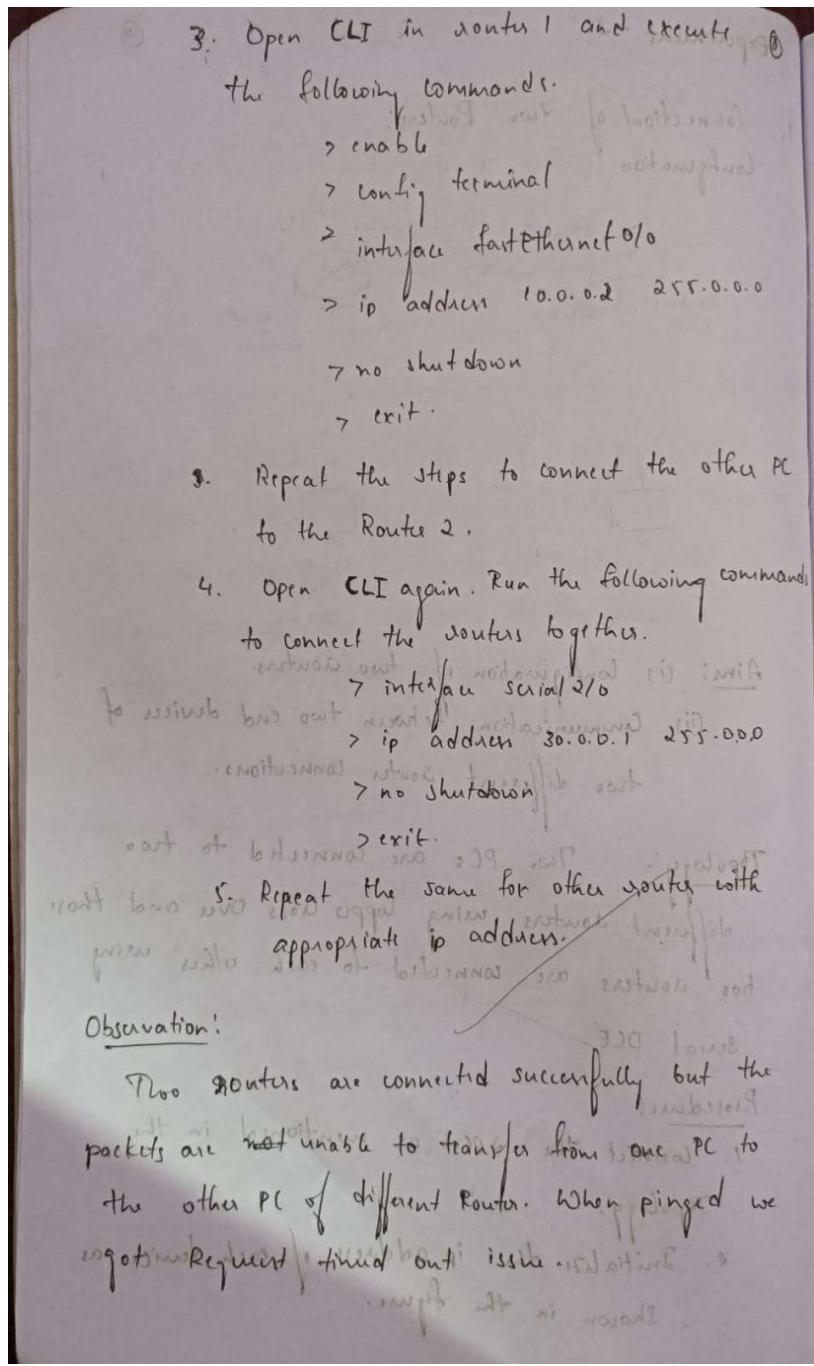
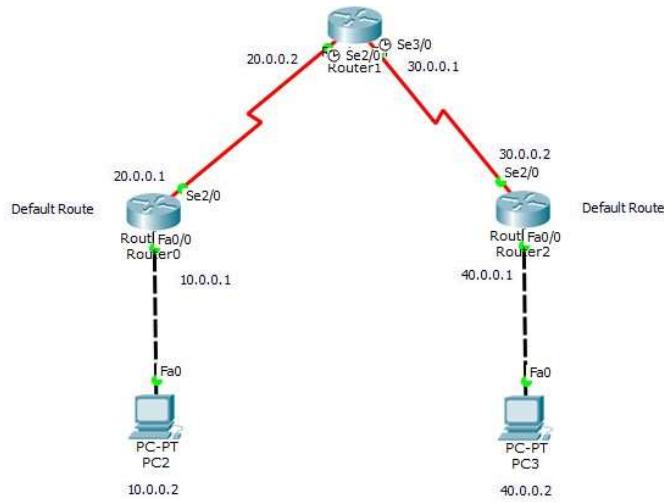
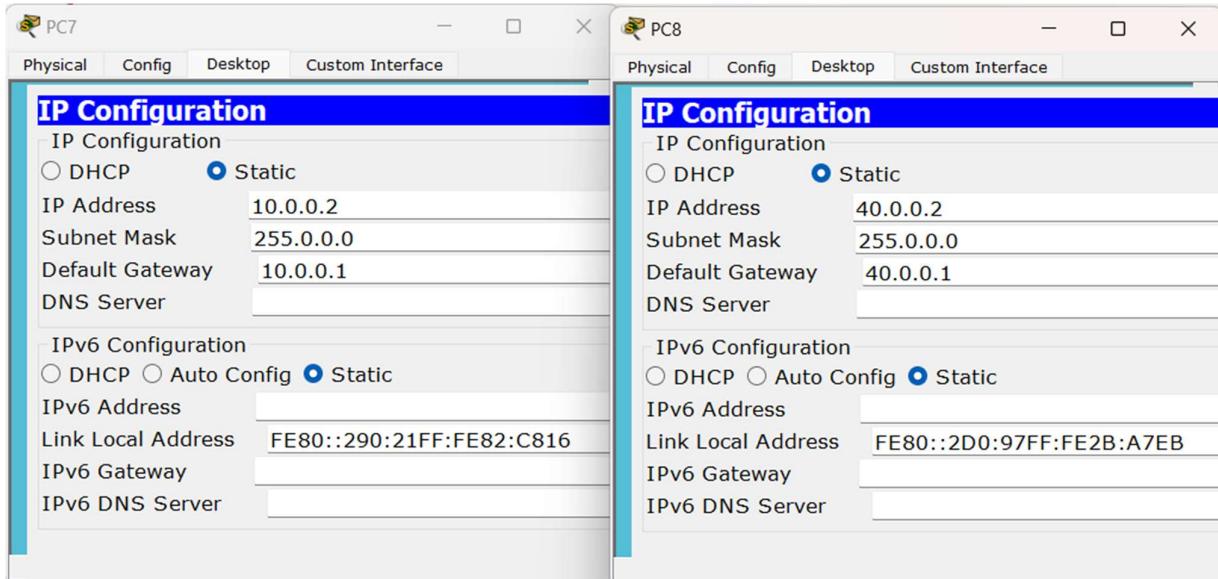


Figure 19: Observation book 2



*Figure 20: Topology for default route*



*Figure 21: IP addresses*

The figure displays three windows representing routers:

- Router9**: Shows configuration for FastEthernet0/0, including an IP address of 10.0.0.1 and a route to 0.0.0.0 via 20.0.0.2.
- Router8**: Shows configuration for Serial2/0, including an IP address of 20.0.0.2 and a route to 30.0.0.0 via 30.0.0.1.
- Router10**: Shows configuration for FastEthernet0/0, including an IP address of 30.0.0.1 and a route to 0.0.0.0 via 30.0.0.1.

Each window has tabs for Physical, Config, and CLI, and includes a Copy and Paste button at the bottom.

*Figure 22: Router CLI Commands*

## Command Prompt

```
PC>ping 40.0.0.2
Pinging 40.0.0.2 with 32 bytes of data:
Reply from 40.0.0.2: bytes=32 time=21ms TTL=123
Reply from 40.0.0.2: bytes=32 time=16ms TTL=123
Reply from 40.0.0.2: bytes=32 time=9ms TTL=123
Reply from 40.0.0.2: bytes=32 time=9ms TTL=123

Ping statistics for 40.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 9ms, Maximum = 21ms, Average = 13ms
```

Figure 23: ping command output

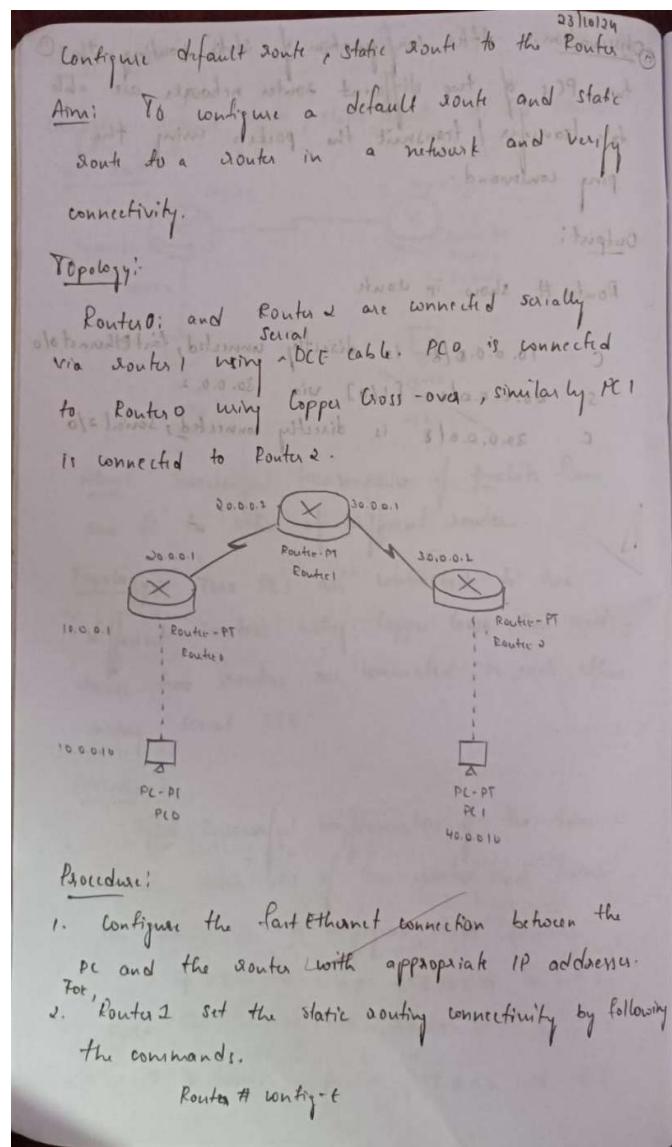


Figure 24: Observation book 1

```

④ Router(Config)# ip route 40.0.0.0 255.0.0.0 30.0.0.2
Router(Config)# ip route 10.0.0.0 255.0.0.0 20.0.0.1 (13)

These commands will set the static Routing of the
Router 01. 20.0.0.0 [6/1] 30.0.0.0

3. Set the default Routing of Router 0 and Router 2.

→ Open CLI on Router 0
Router# config-t
Router(Config)# ip route 0.0.0.0 0.0.0.0 20.0.0.2
Router(Config)# ip route 0.0.0.0 0.0.0.0 30.0.0.1

→ Open CLI on Router 2
Router# config-t
Router(Config)# ip route 0.0.0.0 0.0.0.0 30.0.0.1

When the network and the subnet mask are
set as 0.0.0.0. Any unknown network will be
transferred to the next hop address mentioned.

Observation:
After configuring the default route, PC0 could
communicate with external networks, including PC1
The static route ensured that packets followed
the specified path as given in the static routing.

Output:
Router!# show ip route
S 10.0.0.0/8 [1/0] via 20.0.0.1
C 20.0.0.0/8 is directly connected, Serial 2/0
C 30.0.0.0/8 is directly connected, serial 3/0
S 40.0.0.0/8 [1/0] via 30.0.0.2

```

Figure 25: Observation book 2

#### Program 4

To configure IP addresses of the host using DHCP server present within the LAN and present in a different LAN.

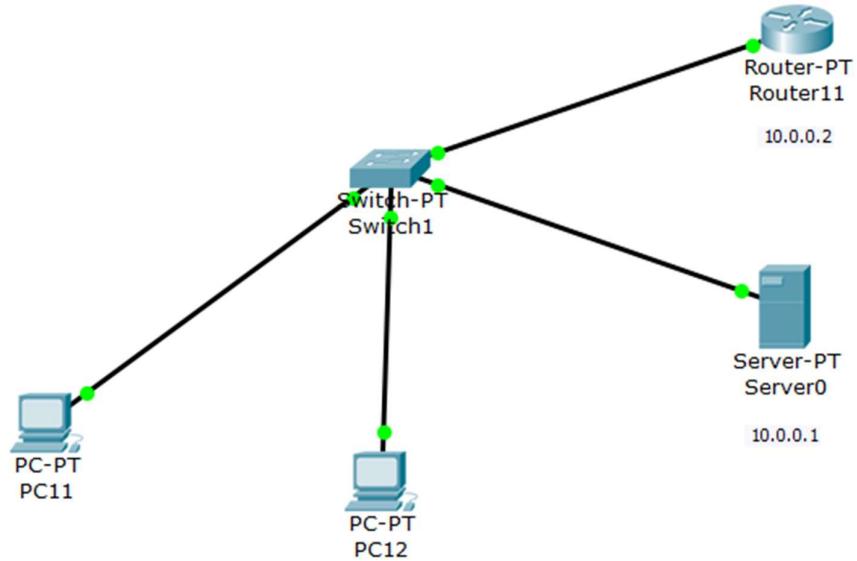


Figure 26: Topology

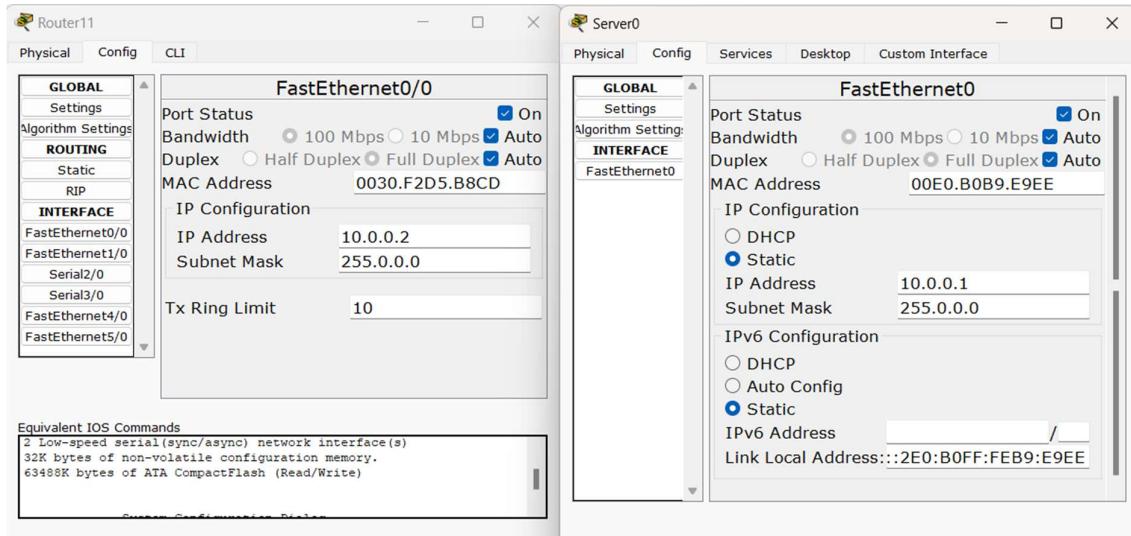


Figure 27: IP Addresses

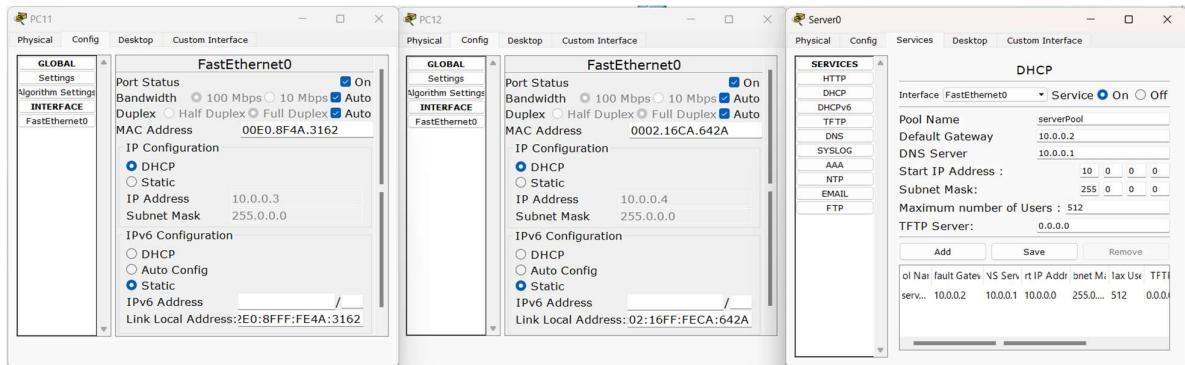


Figure 28: DHCP

```

Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.4

Pinging 10.0.0.4 with 32 bytes of data:

Reply from 10.0.0.4: bytes=32 time=0ms TTL=128

Ping statistics for 10.0.0.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

PC>

```

Figure 29: ping command output

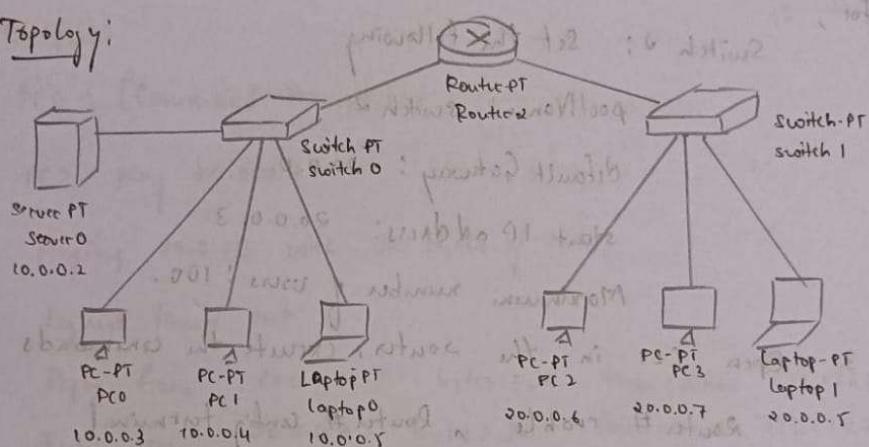
## Experiment 4:-

13-11-2024

Configure DHCP within a LAN and outside LAN.

Aim: To configure and verify the functionality of DHCP both within a LAN and in a network that extends beyond the local LAN.

### Topology:



### Procedure:

1. Connect the end devices and server with the router using copper straight through as shown in the topology.
2. In Server, go to Desktop > IP configuration and set the static address as.  
IP address: 10.0.0.2  
Subnet mask: 255.0.0.0  
Default gateway: 10.0.0.1  
Then go to Services > DHCP  
Configure the Pools like below.

Figure 30: Observation Book I

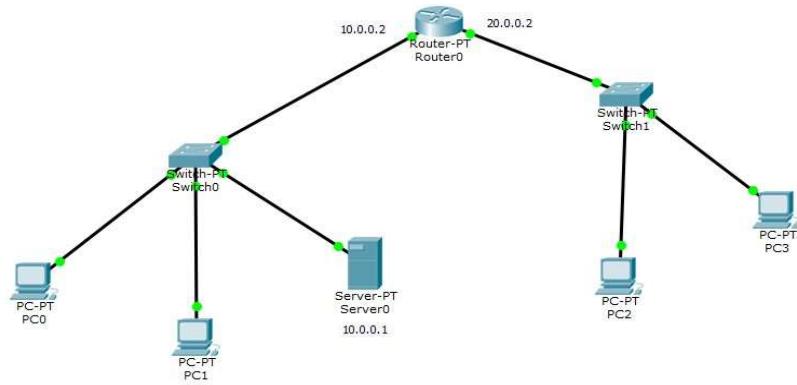


Figure 4: Topology for DHCP in different LAN

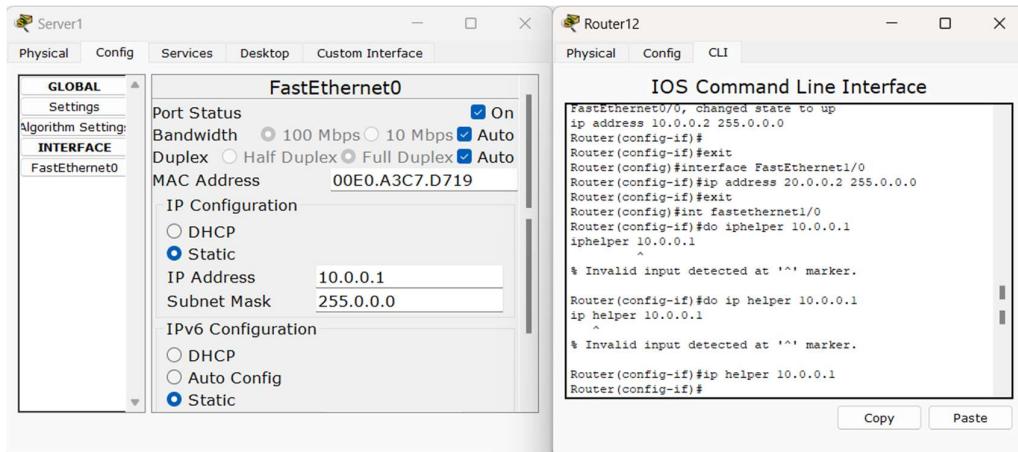


Figure 5: Router CLI and Server IP Address

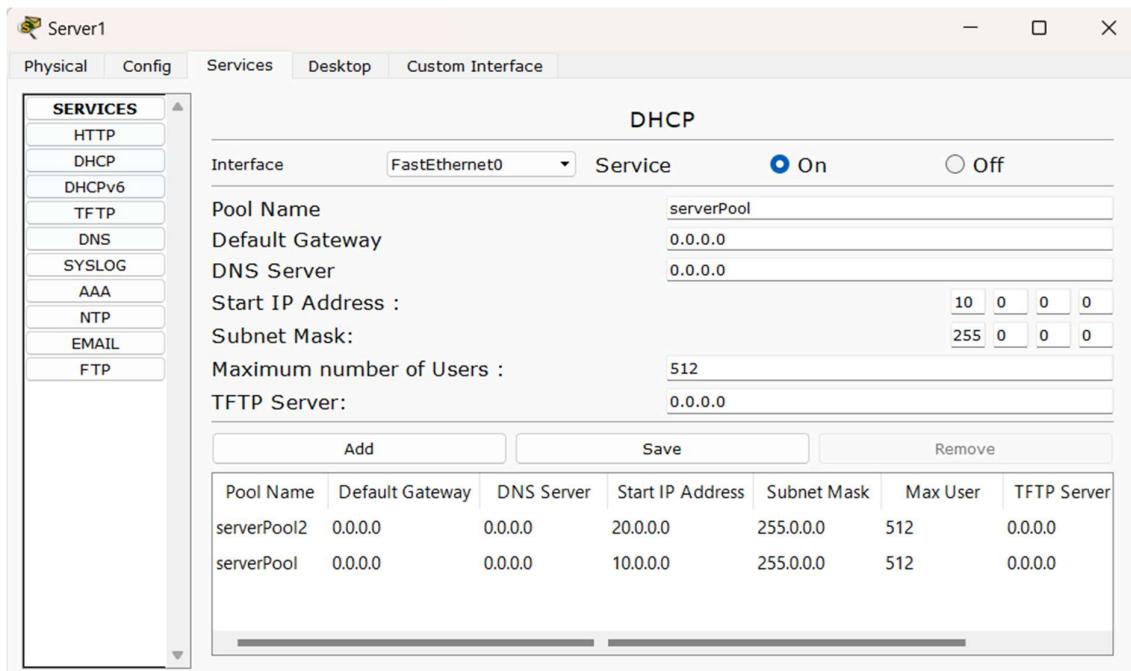


Figure 6: Server Pools of the DHCP Server

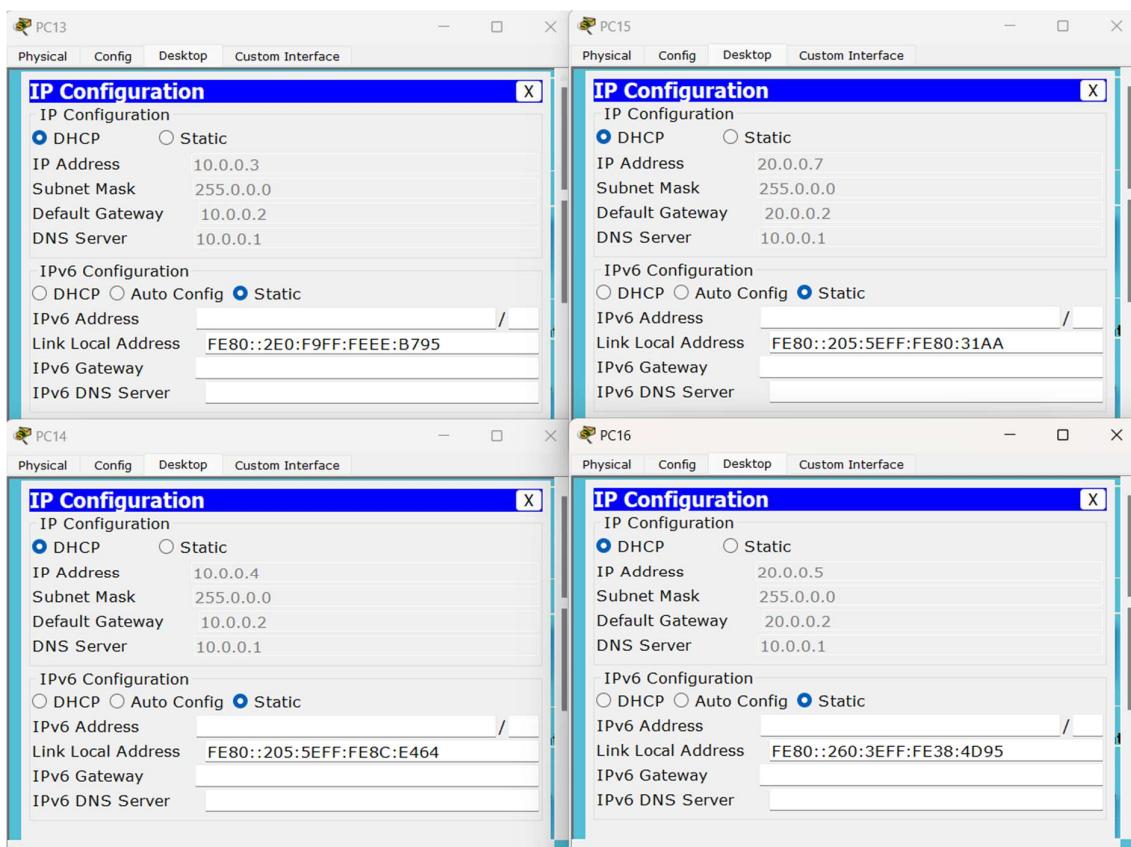


Figure 7: PC IP addresses automatically assigned by DHCP Server

```

PC>ping 20.0.0.3

Pinging 20.0.0.3 with 32 bytes of data:

Reply from 20.0.0.3: bytes=32 time=0ms TTL=127
Reply from 20.0.0.3: bytes=32 time=0ms TTL=127
Reply from 20.0.0.3: bytes=32 time=3ms TTL=127
Reply from 20.0.0.3: bytes=32 time=0ms TTL=127

Ping statistics for 20.0.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 3ms, Average = 0ms

PC>

```

Figure 8: ping command output

Switch 1: Set the following

poolName : switch1  
 default Gateway: 10.0.0.1  
 start IP address: 10.0.0.3  
 Maximum number of Users: 100

and add this.

for ,  
 Switch 2: Set the following

poolName 2: switch2  
 default Gateway: 20.0.0.1  
 start IP address: 20.0.0.3  
 Maximum number of users: 100.

3. Open CLI in the router, execute the commands

Router# enable , Router# config terminal  
 Router(config)# interface fastEthernet 0/0  
 Router(config-if)# ip address 10.0.0.1 255.0.0.0  
 Router(config-if)# ip helper-address 10.0.0.2  
 Router(config-if)# no shutdown.  
 exit.  
 Router(config)# interface fastEthernet 1/0  
 Router(config-if)# ip address 20.0.0.1 255.0.0.0  
 Router(config-if)# ip helper-address 10.0.0.2  
 Router(config-if)# no shutdown.  
 exit.

Figure 9: Observation book 1

Observation: Set up the Router with the fastethernet ⑦ cable connected to the two switches. Setting up the helper address which is the IP address of the server, the other network 20.0.0.0 can access the DHCP service which has been set in the pool service in the server.

Output:

```
PC0 : [Command Prompt]
PC> ping 20.0.0.7
pinging 20.0.0.7 with 32 bytes of data:
Request timed out
Reply from 20.0.0.7: bytes=32 time=0ms TTL=127
Reply from 20.0.0.7: bytes=32 time=0ms TTL=127
Reply from 20.0.0.7: bytes=32 time=0ms TTL=127
```

Ping statistics for 20.0.0.7

Packets: Sent = 4, Received = 3, Lost = 1 (25% loss).

Minimum = 0ms, Maximum = 2ms, Average = 0ms.

Qf the  
Buddy

18:00:09

After config has wt finished) configured

at a birectional work in no static condition

(adapter at all network in entire

0 interval of 100 ms with 6

100ms interval

Forward and Hostnet

Figure 10: Observation book 2

## Program 5

To configure RIP routing protocol in Routers.

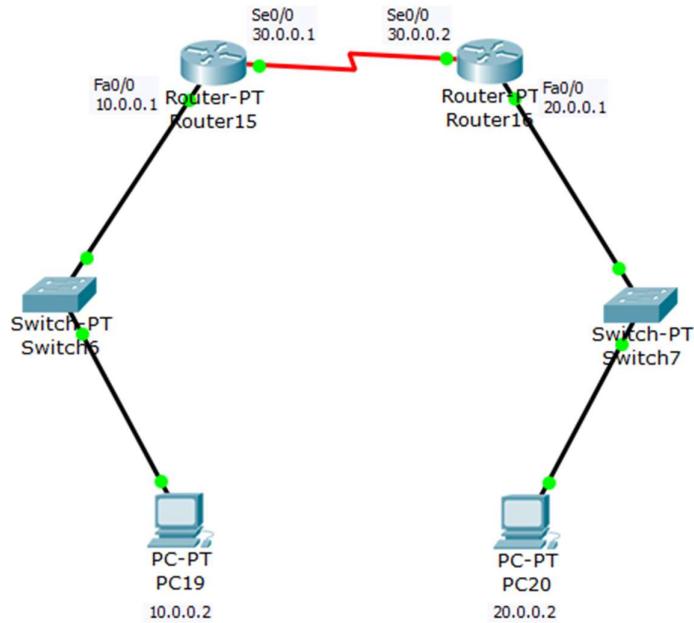


Figure 11: Topology

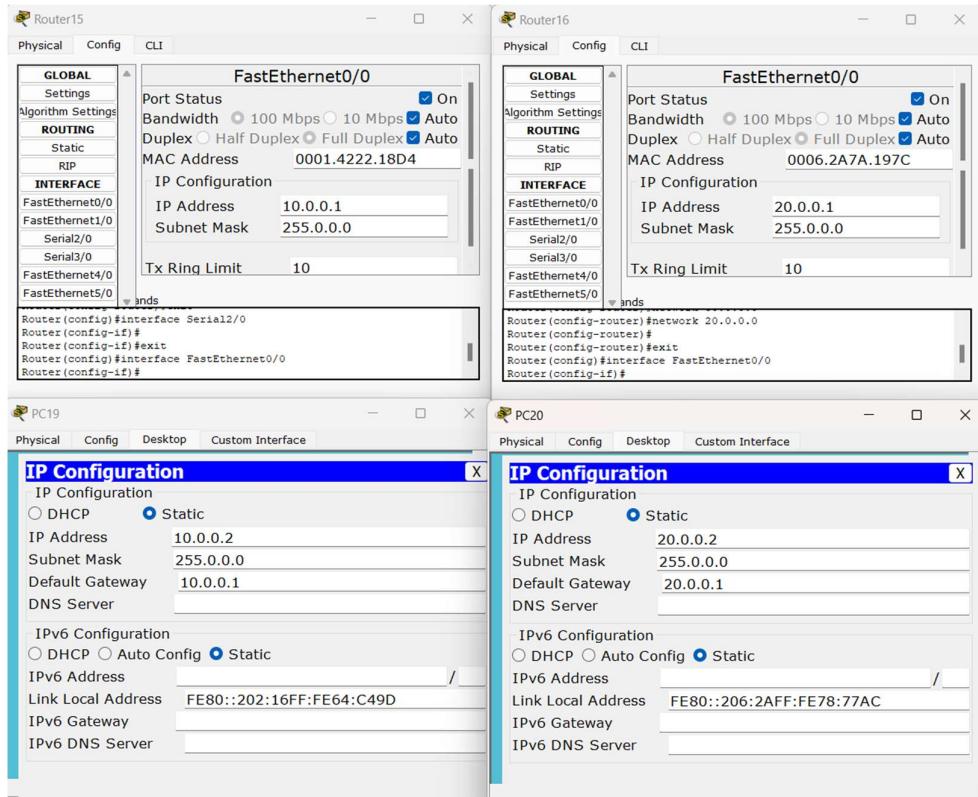


Figure 12: Router and PC IP addresses

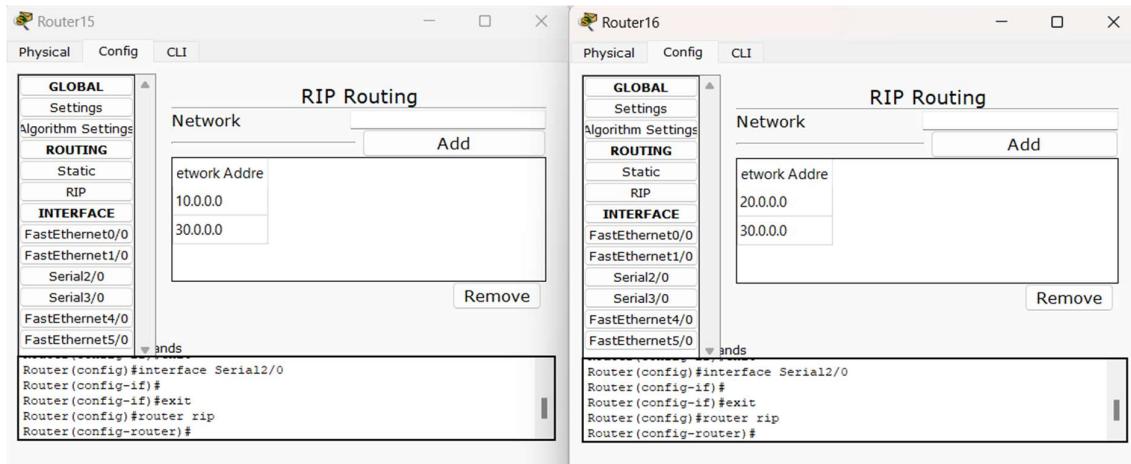


Figure 13: RIP networks

The image shows a 'Command Prompt' window on a PC named 'PC19'. The window title is 'Command Prompt'. The content of the window shows the output of several ping commands to the IP address 20.0.0.2. The output includes statistics for each ping and a summary at the end.

```

Reply from 20.0.0.2: bytes=32 time=9ms TTL=126
Reply from 20.0.0.2: bytes=32 time=1ms TTL=126
Reply from 20.0.0.2: bytes=32 time=5ms TTL=126
Reply from 20.0.0.2: bytes=32 time=1ms TTL=126

Ping statistics for 20.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 9ms, Average = 4ms

PC>ping 20.0.0.2

Pinging 20.0.0.2 with 32 bytes of data:

Reply from 20.0.0.2: bytes=32 time=2ms TTL=126
Reply from 20.0.0.2: bytes=32 time=1ms TTL=126
Reply from 20.0.0.2: bytes=32 time=3ms TTL=126
Reply from 20.0.0.2: bytes=32 time=1ms TTL=126

Ping statistics for 20.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 3ms, Average = 1ms

PC>

```

Figure 14: ping command output

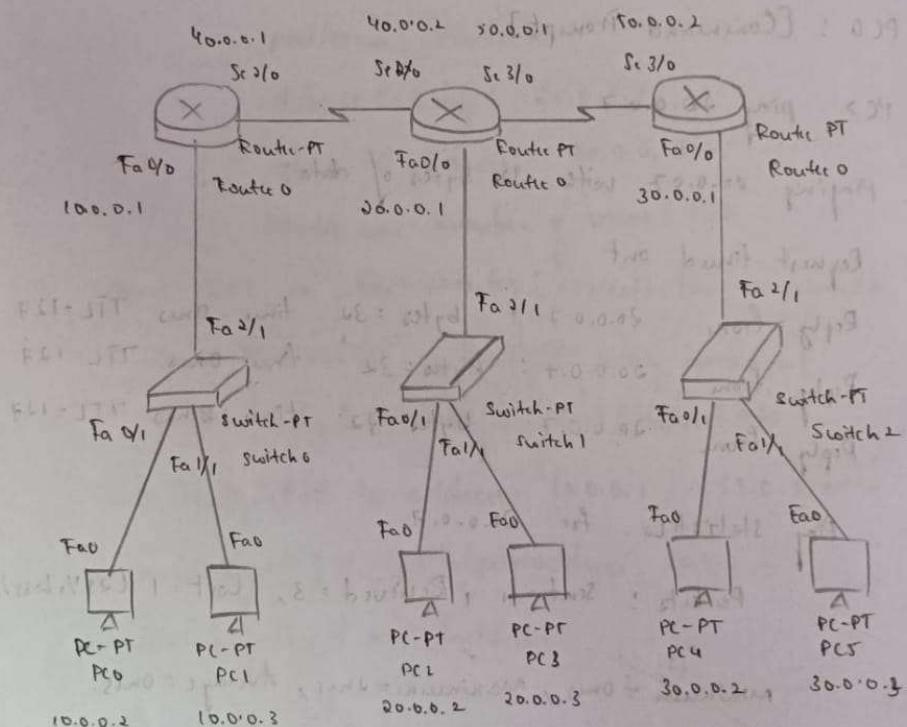
20-11-2024

### Experiment 5:

Configure RIP routing Protocol in Routers.

Aim: To configure RIP routing Protocol in Routers and verify connectivity.

### Topology:-



### Procedure:-

1. Configure Connect the end devices with the switches which are in ~~loop~~ connected to the routers as shown in the topology.

2. Now, Open CLI in Router 0.

Router> enable

Router# config terminal

Figure 15: Observation book 1

Router (config) # Router ip  
 Router (config-router) # network 10.0.0.0  
 Router (config-router) # network 40.0.0.0  
 Open CLI in Router 1.  
 Router > enable  
 Router # config terminal  
 Router (config) # Router ip (ws) & (ws) RIP  
 Router (config-router) # network 40.0.0.0  
 Router (config-router) # network 20.0.0.0  
 Router (config-router) # network 50.0.0.0  
 Open CLI in Router 2  
 Router > enable  
 Router # config terminal  
 Router (config) # Router ip  
 Router (config-router) # network 50.0.0.0  
 Router (config-router) # network 30.0.0.0  
 Give the IP address of the host to establish  
 the connection.  
Observation: After configuring the default or RIP  
 routing protocol, the end devices are able to communicate properly.  
Output: Router 0

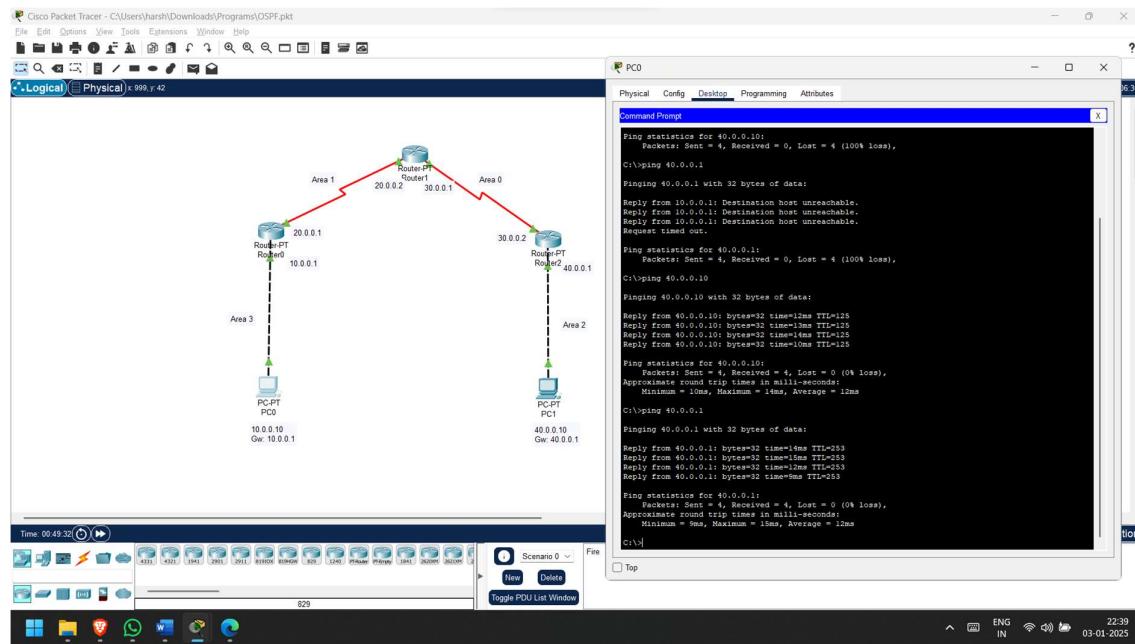
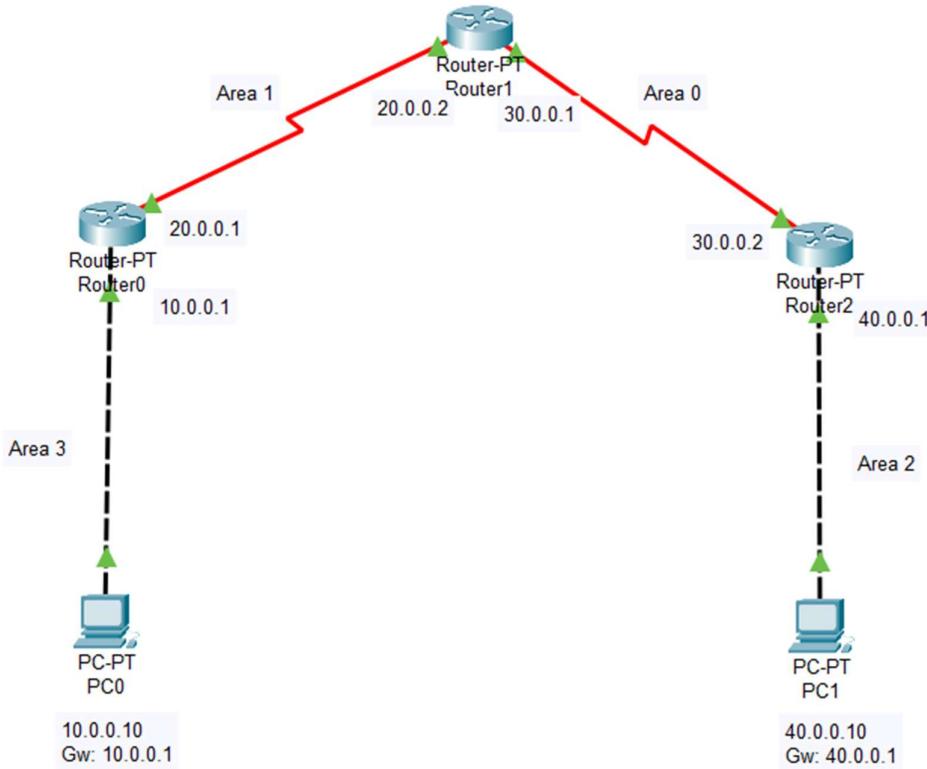
```

    Router > enable
    Router # show ip route
    C 10.0.0.0/8 is directly connected, FastEthernet 0/0
    C 20.0.0.0/8 [120/1] via 40.0.0.2, 00:00:24, Serial 1/0
    R 30.0.0.0/8 [120/1] via 40.0.0.2, 00:00:24, Serial 2/0
    C 40.0.0.0/8 [120/1] directly connected, Serial 2/0
    K 50.0.0.0/8 [120/1] via 40.0.0.2, 00:00:24, Serial 1/0
  
```

Figure 16: Observation book 2

## Program 6

Configure OSPF routing Protocol

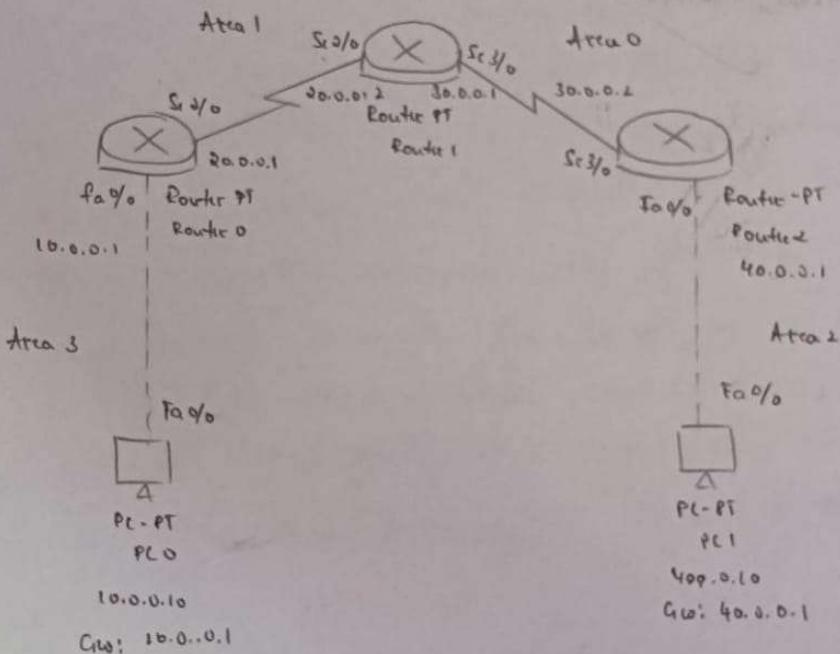


## Experiment No 6:

Configure OSPF routing protocol.

Aim: To configure OSPF routing Protocol in Routers and verify connectivity.

## Topology:



## Procedure:

1. Establish the network connection as shown in the topology.
2. Set the IP address of the PC and the gateway address respectively.
3. Open CLI in Router 0

```
R0(Config)# interface fastethernet 0/0
```

```
R0(config-if)# ip address 10.0.0.1 255.0.0.0
```

```
R0(config-if)# no shutdown
```

```
R0(config-if)# exit
```

R0(config) # interface serial 0/0 (23)

R0(config-if) # ip address 20.0.0.1 255.0.0.0

R0(config-if) # encapsulation ppp

R0(config-if) # clock rate 64000

Open CLI in router 01

R1(config) # interface serial 0/0

R1(config-if) # ip address 20.0.0.2 255.0.0.0

R1(config-if) # encapsulation ppp

R1(config-if) # no shutdown

R1(config-if) # exit

R1(config) # interface serial 3/0

R1(config-if) # ip address 30.0.0.1 255.0.0.0

R1(config-if) # encapsulation ppp

R1(config-if) # clock rate 64000

R1(config-if) # no shutdown

R1(config-if) # exit

Open CLI in router 2

R2(config) # interface fastethernet 0/0

R2(config-if) # ip address 40.0.0.1 255.0.0.0

R2(config-if) # no shutdown

R2(config-if) # exit

R2(config) # interface serial 3/0

R2(config-if) # ip address 30.0.0.2 255.0.0.0

R2(config-if) # encapsulation ppp

R2(config-if) # no shutdown

R2(config-if) # exit

Step 3: Enable ip routing by configuring ospf routing protocol.

In Router R0,

```
R0 Config # router ospf 1  
R0(Config-router) # router-id 1.1.1.1  
# network 10.0.0.0 0.255.255.255 area 3  
# network 20.0.0.0 0.255.255.255 area 1  
# exit.
```

In Router R1,

```
R1 Config # router ospf 1  
R1(Config-router) # router-id 2.2.2.2  
# network 20.0.0.0 0.255.255.255 area 1  
# network 30.0.0.0 0.255.255.255 area 0  
# exit.
```

In Router R2,

```
R2 Config # router ospf 1  
R2(Config-router) # router-id 3.3.3.3  
# network 30.0.0.0 0.255.255.255 area 0  
# network 40.0.0.0 0.255.255.255 area 2  
# exit.
```

Step 4: Check the routing table.

Router # show ip route.

You can see the code 0 which stands for OSPF connection, where Area 0 exists.

There must be one interface up to keep ospf process up. So its better to configure loopback address to ②

Routu

```
R0 (config) # interface loopback 0  
R0 (config-if) # ip add 172.16.1.252 255.255.0.0  
# no shutdown.
```

```
R1 (config) # interface loopback 0
```

```
R1 (config-if) # ip add 172.16.1.253 255.255.0.0  
# no shutdown.
```

```
R2 (config) # interface loopback 0
```

```
R2 (config-if) # ip add 172.16.1.254 255.255.0.0  
# no shutdown.
```

Step 5: Check Routing table of R3

You can see the node 0 here now.

Step 6: Create virtual link between R1, R2 and R2, R3

```
R1 (config) # router ospf 1
```

```
R1 (config-router) # area 1 virtual-link 2.2.2.2
```

```
R2 (config) # router ospf 1
```

```
R2 (config-router) # area 1 virtual-link 1.1.1.1  
# area 0 virtual-link 3.3.3.3
```

```
R3 (config) # router ospf 1
```

```
R3 (config-router) # area 0 virtual-link 2.2.2.2
```

Step 81 Check connectivity between host  
10.0.0.10 to 40.0.0.10.

Observation: After configuring the OSPF routing protocol and establishing the virtual link, the two end hosts can communicate successfully.

Output:

~~PC~~: A(0)

PC> ping 40.0.0.10

0 Broadcast packet to 40.0.0.10  
Ping to 40.0.0.10 with 32 bytes of data:

Reply from 40.0.0.10: bytes = 32 time = 9ms TTL = 125.  
(4 times)

Ping statistics for 40.0.0.10

Packet: Sent = 4 Received = 3 Lost = 1

RTT  
18/12/24

## Program 7

To understand the operation of TTL by sending a simple PDU from one network to different network

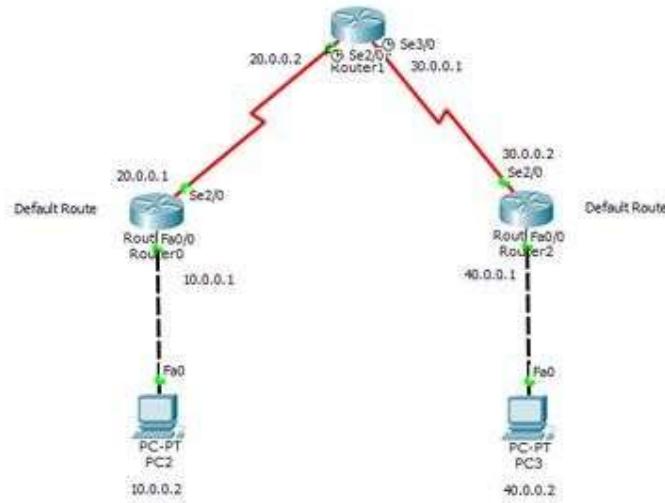


Figure 44: Topology

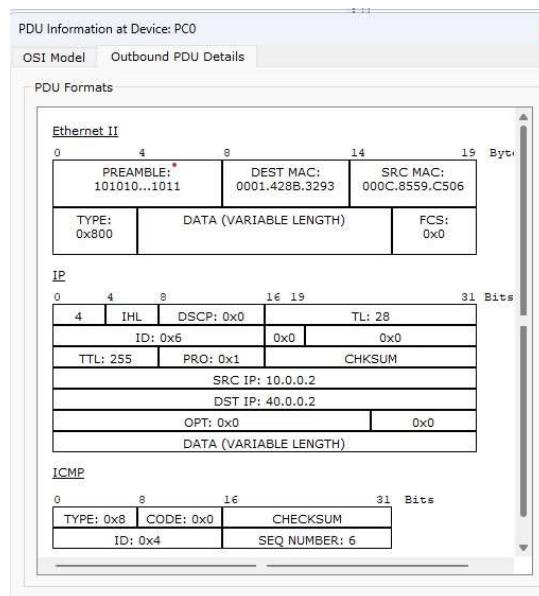


Figure 45: TTL of PC0

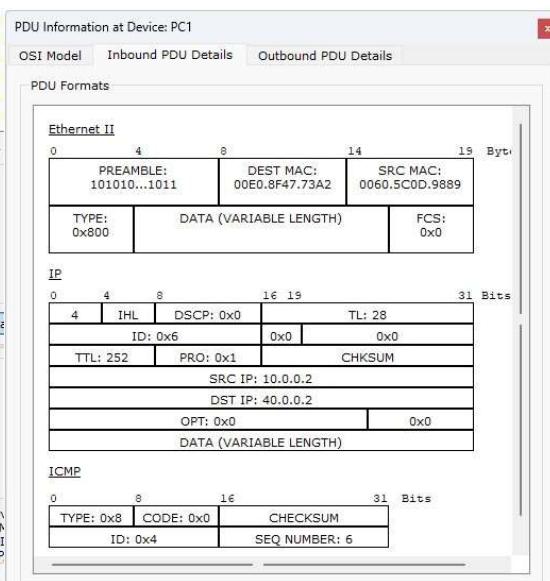


Figure 46: Inbound TTL of PC1

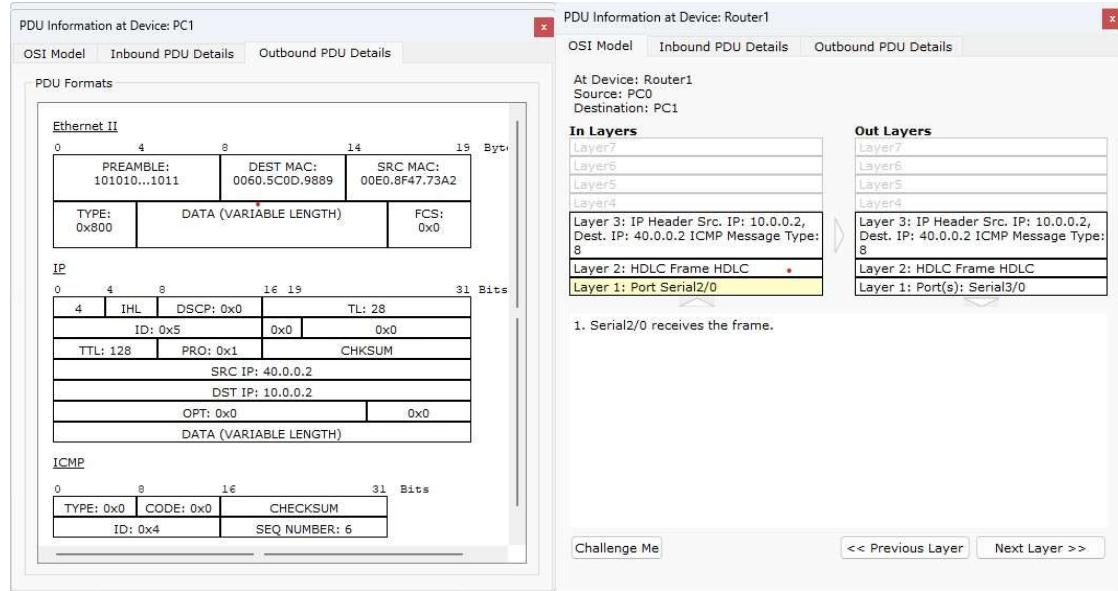


Figure 47: Outbound TTL of PC1

Figure 48: OSI Model of Router

## Experiment 7:

Demonstrate the TTL / Life of a packet.

### Observation:-

As a packet travels through a network, its TTL (Time to live) value decreases by 1 at each router it passes. Initially, the packet might start with a TTL of 255. When it reaches the first router, the TTL is reduced to 254. As the packet continues its journey through each successive router, the TTL continues to decrement - 253 after the second router, 252 after the third, and so on. If the TTL reaches 0 before reaching its destination, the packet is discarded and an ICMP "Time Exceeded" message is sent back to the sender.

### Topology:

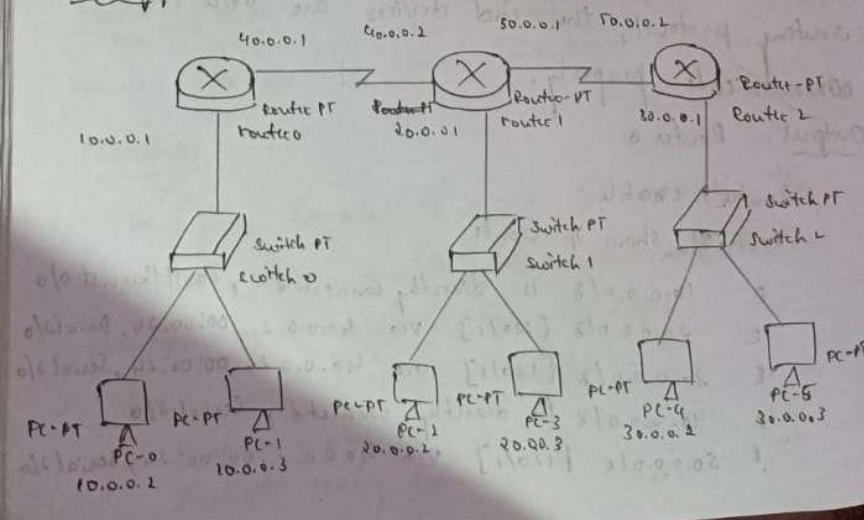


Figure 49: Observation book 1

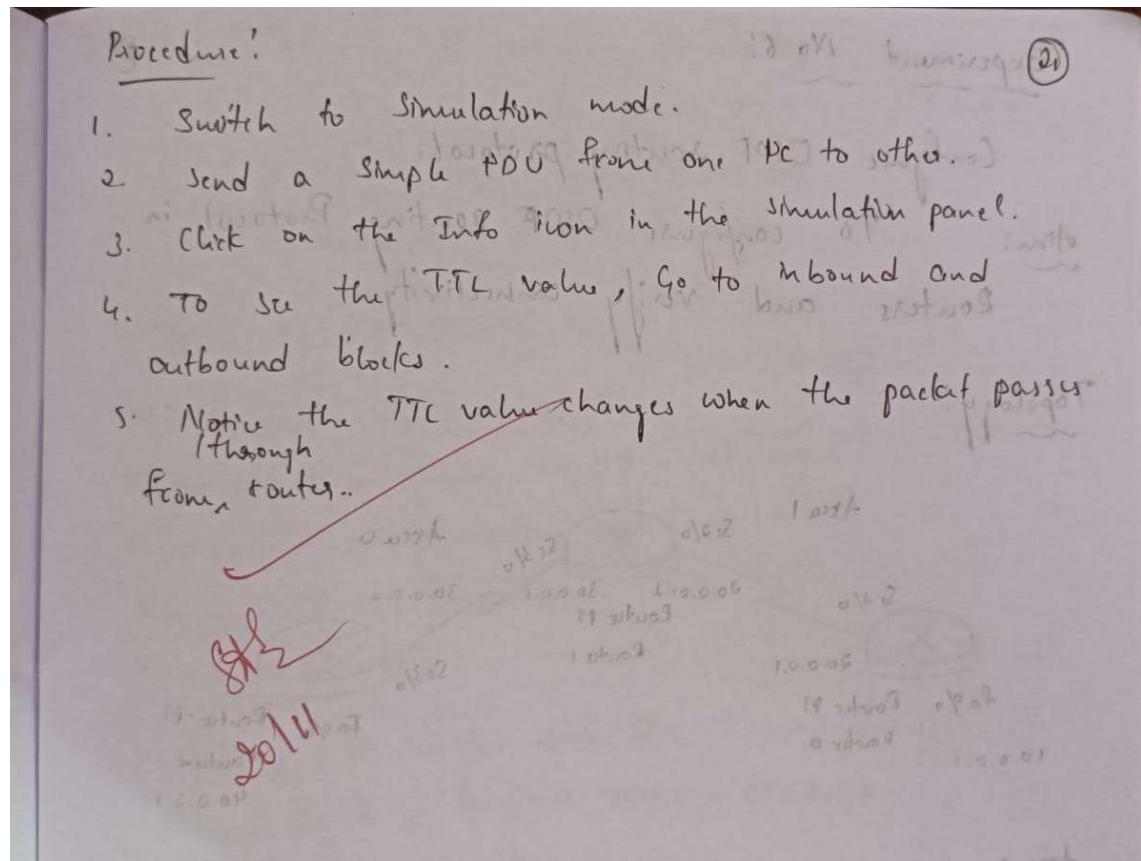


Figure 50: Observation book 2

## Program 8

To configure DNS server to demonstrate mapping of IP addresses and domain names.

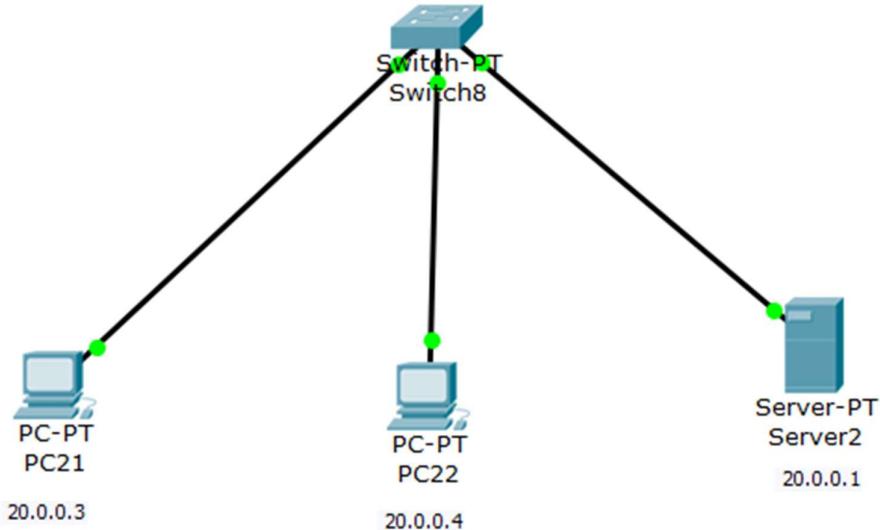


Figure 51: Topology

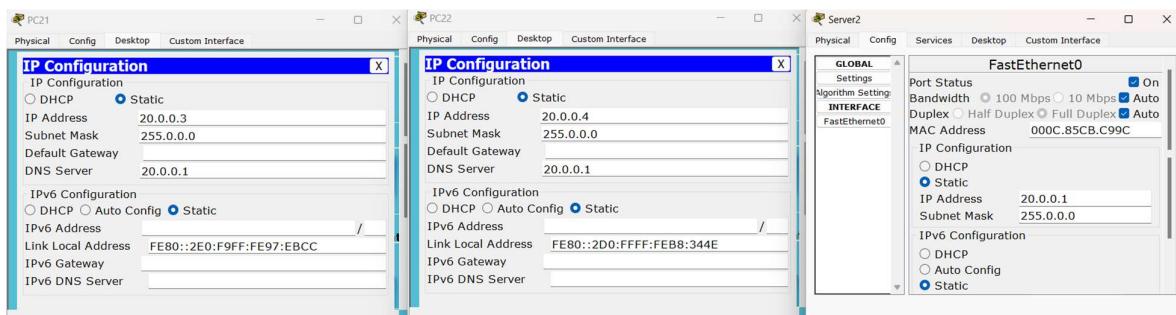


Figure 52: IP Addresses



Figure 53: Server and the website



Figure 54: DNS and domain names

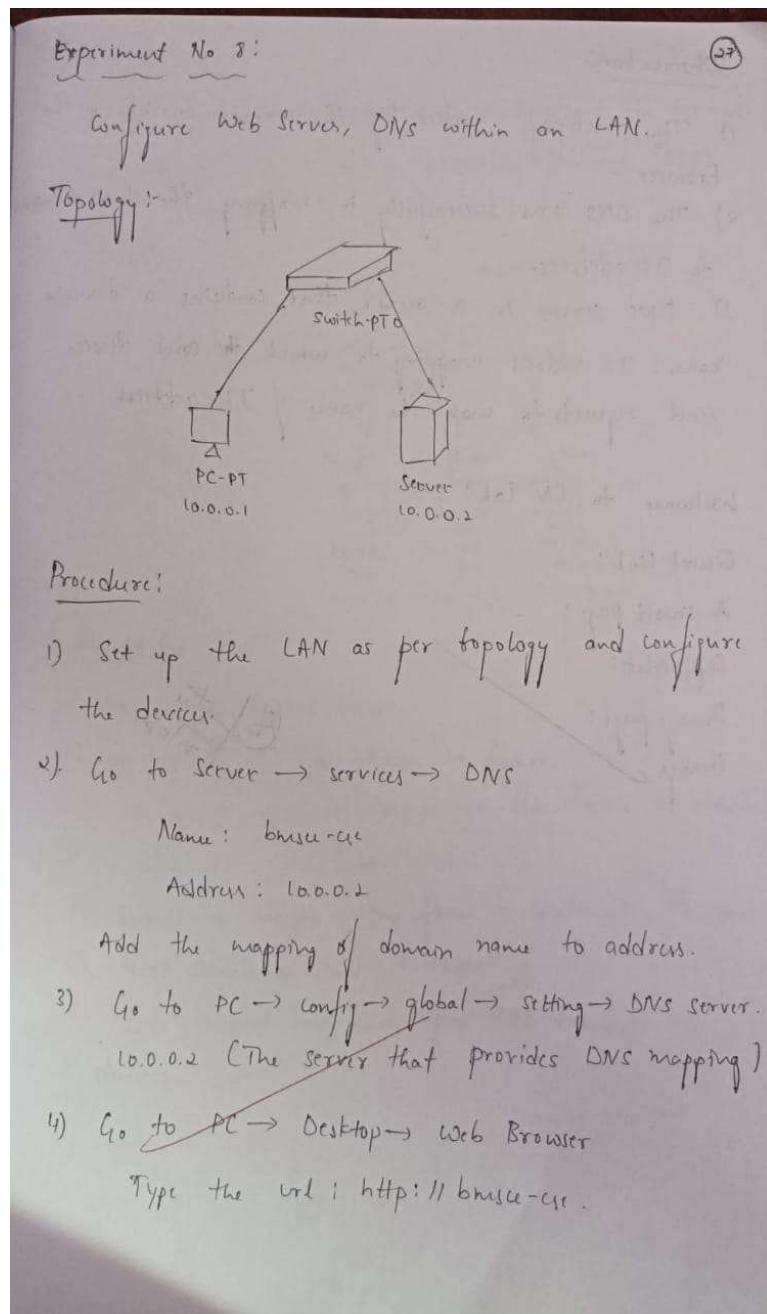


Figure 55: Observation Book I

Observation:

- 1) The webpage hosted by the server visible on the browser.
- 2) The DNS was successfully in mapping the domain name to IP address.
- 3) DNS server is a server that contains a domain name: IP address mapping to which the end device send requests to map the name of IP address.

Welcome to CN lab!

Quick link:

A small page:

Copyright:

Image page:

Image:

By the  
author

Figure 56: Observation Book 2

### **Program 9**

To demonstrate the working of Address Resolution Protocol for communication with the LAN.

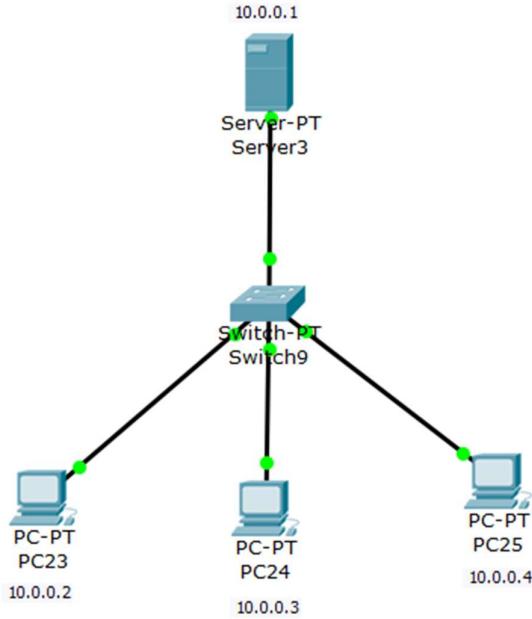


Figure 57: Topology

Four separate windows show the ARP tables for the Server and three PCs. Each window has columns for IP Address, Hardware Address, and Interface, with FastEthernet0 as the interface for all entries.

| IP Address | Hardware Address | Interface     |
|------------|------------------|---------------|
| 10.0.0.2   | 0001.C9A3.2E34   | FastEthernet0 |
| 10.0.0.3   | 0001.96E7.5E01   | FastEthernet0 |
| 10.0.0.4   | 0006.2A70.55DC   | FastEthernet0 |

| IP Address | Hardware Address | Interface     |
|------------|------------------|---------------|
| 10.0.0.1   | 0030.A3B6.E2D2   | FastEthernet0 |
| 10.0.0.3   | 0001.96E7.5E01   | FastEthernet0 |
| 10.0.0.4   | 0006.2A70.55DC   | FastEthernet0 |

| IP Address | Hardware Address | Interface     |
|------------|------------------|---------------|
| 10.0.0.1   | 0030.A3B6.E2D2   | FastEthernet0 |
| 10.0.0.2   | 0001.C9A3.2E34   | FastEthernet0 |
| 10.0.0.4   | 0006.2A70.55DC   | FastEthernet0 |

| IP Address | Hardware Address | Interface     |
|------------|------------------|---------------|
| 10.0.0.1   | 0030.A3B6.E2D2   | FastEthernet0 |
| 10.0.0.2   | 0001.C9A3.2E34   | FastEthernet0 |
| 10.0.0.3   | 0001.96E7.5E01   | FastEthernet0 |

Figure 58: ARP Tables of all PCs and Server

The screenshot shows a Cisco Packet Tracer window titled "PC23". At the top, there are tabs: Physical, Config, Desktop, and Custom Interface. Below the tabs is a toolbar with icons for various functions. A window titled "Command Prompt" is open, displaying the following text:

```

Packet Tracer PC Command Line 1.0
PC>ARP -A
No ARP Entries Found
PC>ARP -A
  Internet Address      Physical Address      Type
  10.0.0.1                0030.a3b6.e2d2    dynamic
  10.0.0.3                0001.96e7.5e01    dynamic
  10.0.0.4                0006.2a70.55dc    dynamic
PC>

```

Figure 59: ARP command output of PC0

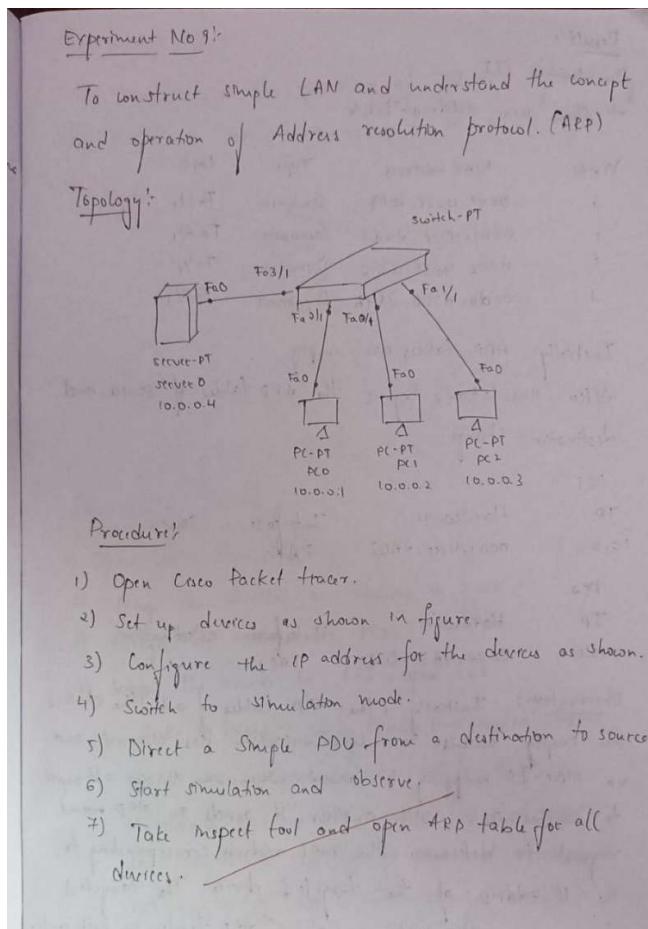


Figure 60: Observation Book I

Result:

In Switch CLI

→ Shows mac address table

| VLAN | MAC address    | Type    | Ports |
|------|----------------|---------|-------|
| 1    | 0001.4265.6c08 | Dynamic | Fa3/1 |
| 1    | 0001.43ee.2ad3 | Dynamic | Fa2/1 |
| 1    | 0002.4a35.d3c2 | Dynamic | Fa1/1 |
| 1    | 0000.d35b.384b | Dynamic | Fa0/1 |

Initially ARP tables are empty.

After simulation, begins the ARP tables of source and destination change.

PC1

| IP       | Hardware       | Interface | Source |
|----------|----------------|-----------|--------|
| 10.0.0.3 | 0001.43ee.2ad3 | Fa1/1     |        |

PC2

| IP       | Hardware       | Interface | Destination |
|----------|----------------|-----------|-------------|
| 10.0.0.2 | 0002.4a35.d3c2 | Fa1/1     |             |

Observation: Initially, the ARP tables of all the devices are empty because no communication has occurred, and no Mac-IP mapping is cached. When one device attempts to communicate with another, it sends an ARP request to determine the MAC address corresponding to the IP address of the targeted device. The targeted device with ARP reply, updating ARP tables on both ends.

The switch builds its MAC address table by mapping MAC address to ports based on receiving.

~~Only 1st time~~

Figure 61: Observation book 2

## **Program 10**

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

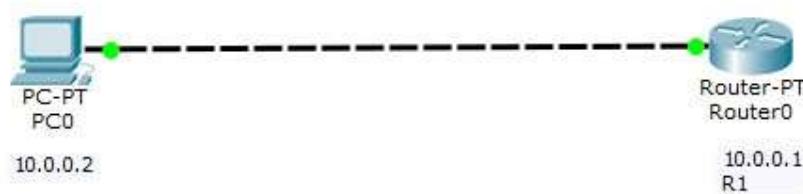


Figure 62: Topology

The screenshot shows the "IOS Command Line Interface" for Router0. The title bar includes tabs for "Physical", "Config", and "CLI", with "CLI" being the active tab. The main window displays the following configuration session:

```
-- System Configuration Dialog --
Continue with configuration dialog? [yes/no]: n

Press RETURN to get started!

Router>enable
Router>configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface FastEthernet0/0
Router(config-if)#exit
Router(config)#interface FastEthernet0/0
Router(config-if)#no shutdown

Router(config-if)#
*LINEPROTO-0-UPDOWN: Interface FastEthernet0/0, changed state to up
*LINKPROTO-0-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
no ip address
Router(config-if)#no ip address
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/0
Router(config-if)#ip address 10.0.0.1 255.0.0.0
Router(config-if)#post name R1

* Invalid input detected at '^' marker.
Router(config-if)#postname R1
^
* Invalid input detected at '^' marker.

Router(config-if)#hostname R1
R1(config)#
R1(config)#
R1(config)#router rip
R1(config-router)#enable config
^
* Invalid input detected at '^' marker.

R1(config-router)#enable secret R0
R1(config)#line vty 0 5
R1(config-line)#login
* Login disabled on line 130, until 'password' is set
* Login disabled on line 133, until 'password' is set
* Login disabled on line 134, until 'password' is set
* Login disabled on line 135, until 'password' is set
* Login disabled on line 136, until 'password' is set
* Login disabled on line 137, until 'password' is set
R1(config-line)#password P1
R1(config-line)#exit
R1(config)#
R1#
*TS-5-CONFIG_I: Configured from console by console
nr
Building configuration...
[OK]
R1#
```

Figure 63: Router CLI

```

Command Prompt

Packet Tracer PC Command Line 1.0
PC>PING 10.0.0.1

Pinging 10.0.0.1 with 32 bytes of data:
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 = 0ms, Average = 0ms

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=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 = 0ms, Average = 0ms

PC>telnet 10.0.0.1
Trying 10.0.0.1 ...Open

User Access Verification

Password:
R1>enable
Password:
R1#
```

Figure 64: Output

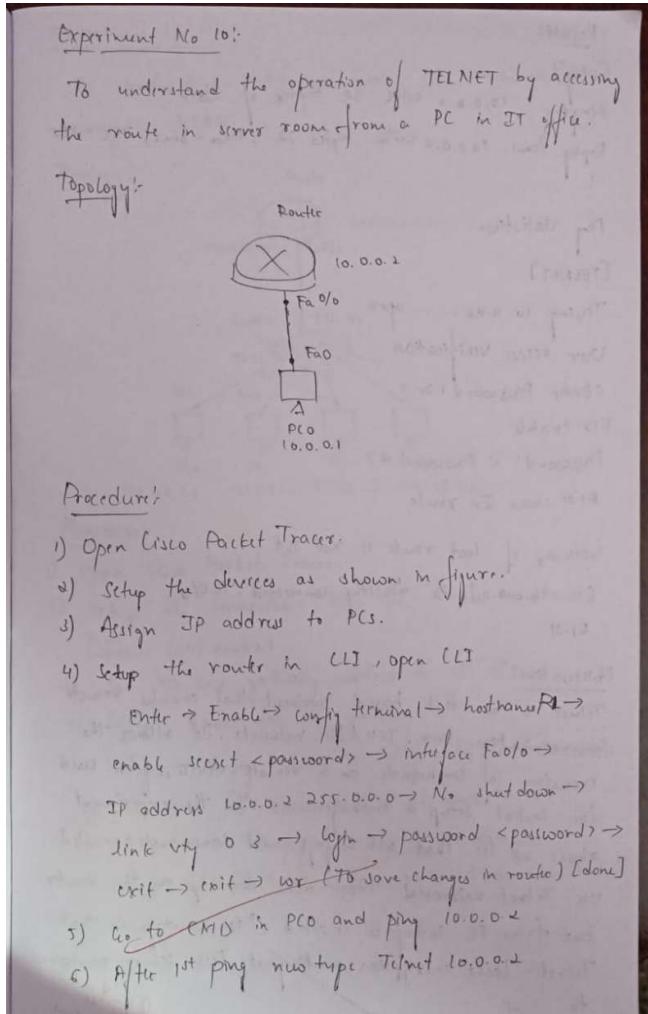


Figure 65: Observation Book I

Results

[ping]

Pinging 10.0.0.2 with 32 bytes of data  
 Reply from 10.0.0.2 with bytes 32, time = 0ms, TTL = 255

;

Ping statistics.

[TELNET]

Trying 10.0.0.2... open

User access Verification

<Enter Password>

R1# enable

Password: < Password >

R1# show IP route

Gateway of last route is not set

C: 10.0.0.0/8 is directly connected, Fa0/0

R1#

Observation:

Telnet is a text based protocol that enables remote communication over TCP/IP networks. It allows the execution of commands on a remote device, often used for initial setup or management. In the experiment above we see that all configs and commands executed via Telnet mirrored those done directly on the router but from PC interface instead. Disadvantage is that Telnet lacks encryption making it less secure compared to SSH.

*Redacted*

Figure 66: Observation Book 2

## Program 11

To create a virtual LAN on top of the physical LAN and enable communication between Physical LAN and Virtual LAN.

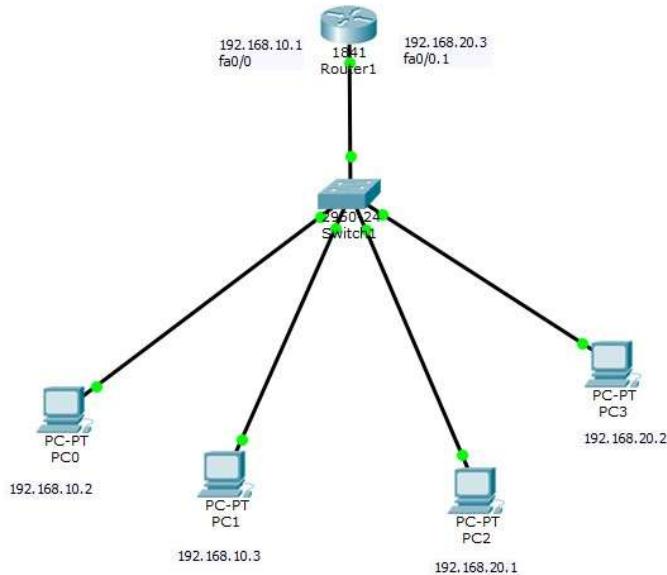


Figure 67: Topology

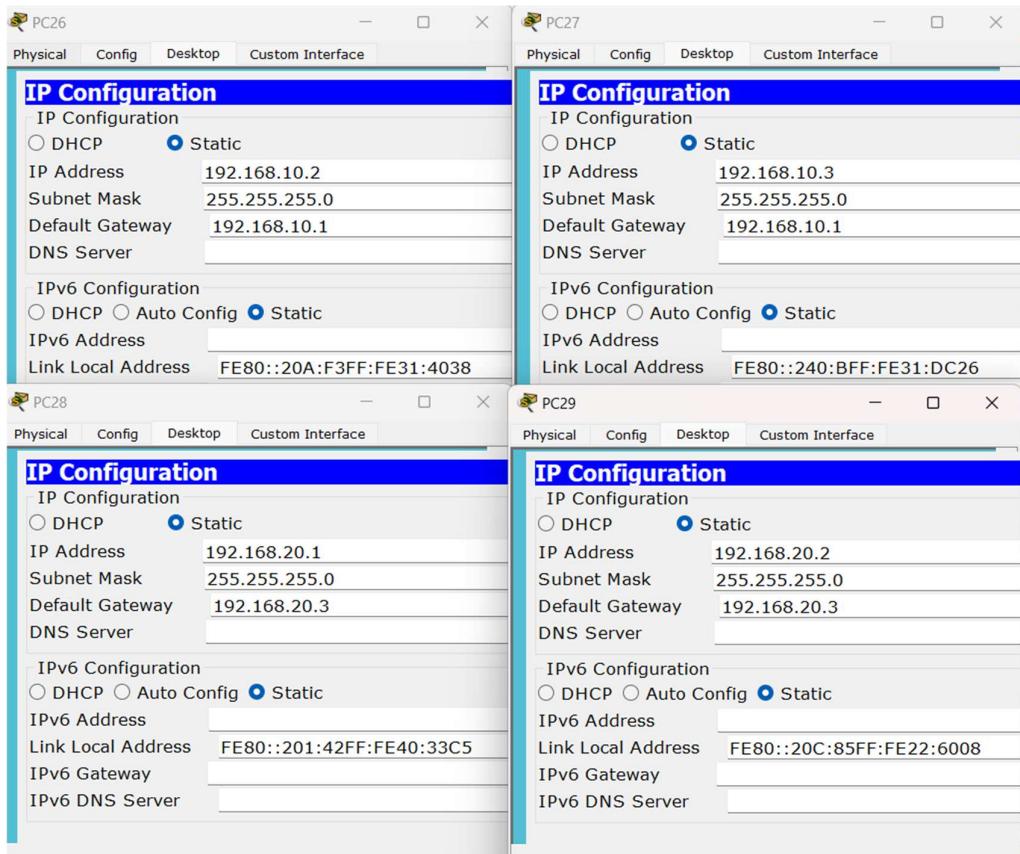
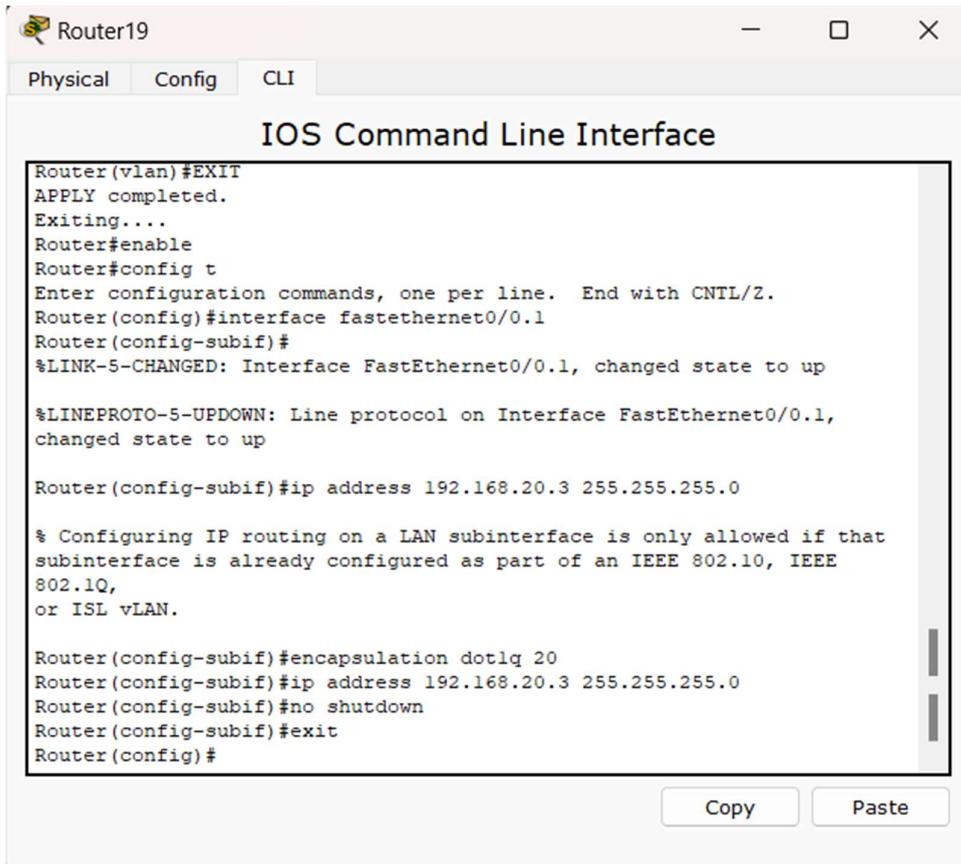


Figure 68: IP Addresses of the PCs



The screenshot shows the Router19 CLI interface. The title bar says "Router19". Below it are tabs for "Physical", "Config", and "CLI", with "CLI" being active. The main window title is "IOS Command Line Interface". The terminal window displays the following configuration commands:

```

Router(vlan)#EXIT
APPLY completed.
Exiting....
Router#enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface fastethernet0/0.1
Router(config-subif)#
%LINK-5-CHANGED: Interface FastEthernet0/0.1, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0.1,
changed state to up

Router(config-subif)#ip address 192.168.20.3 255.255.255.0

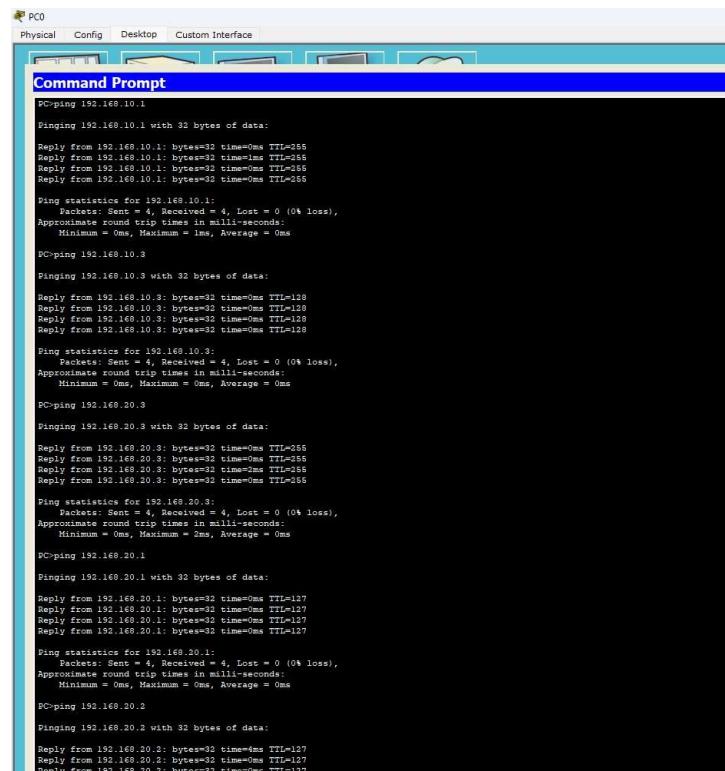
% Configuring IP routing on a LAN subinterface is only allowed if that
subinterface is already configured as part of an IEEE 802.10, IEEE
802.1Q,
or ISL VLAN.

Router(config-subif)#encapsulation dot1q 20
Router(config-subif)#ip address 192.168.20.3 255.255.255.0
Router(config-subif)#no shutdown
Router(config-subif)#exit
Router(config)#

```

At the bottom of the terminal window are "Copy" and "Paste" buttons.

Figure 69: Router CLI



The screenshot shows a "Command Prompt" window titled "PCO". The tabs at the top are "Physical", "Config", "Desktop", and "Custom Interface", with "Config" being active. The terminal window displays the results of multiple ping operations:

```

Command Prompt
PCping 192.168.10.1
Pinging 192.168.10.1 with 32 bytes of data:
Reply from 192.168.10.1: bytes=32 time=0ms TTL=255

Ping statistics for 192.168.10.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

PCping 192.168.10.3
Pinging 192.168.10.3 with 32 bytes of data:
Reply from 192.168.10.3: bytes=32 time=0ms TTL=129

Ping statistics for 192.168.10.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

PCping 192.168.20.3
Pinging 192.168.20.3 with 32 bytes of data:
Reply from 192.168.20.3: bytes=32 time=0ms TTL=255

Ping statistics for 192.168.20.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

PCping 192.168.20.1
Pinging 192.168.20.1 with 32 bytes of data:
Reply from 192.168.20.1: bytes=32 time=0ms TTL=127

Ping statistics for 192.168.20.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

PCping 192.168.20.2
Pinging 192.168.20.2 with 32 bytes of data:
Reply from 192.168.20.2: bytes=32 time=0ms TTL=127

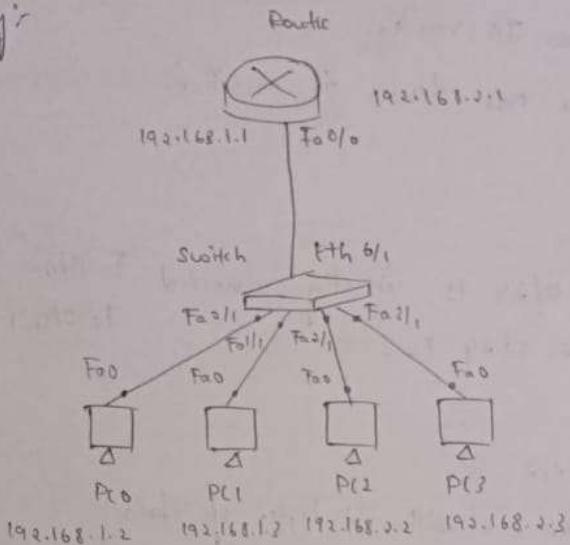
```

Figure 70: Output

### Experiment No. 11:

To construct a VLAN and make PCs communicate among a VLAN.

#### Topology:



#### Procedure:

- 1) Open Cisco Packet Tracer.
- 2) Setup the connections b/w the devices as shown.  
(choose 1841 router)
- 3) Set IP and gateway address to all PCs.
- 4) Setup Router for one gateway  
enable → config terminal → interface Fa0/0 → IP address  
192.168.1.1 255.255.255.0 → no shut → exit.
- 5) Go to switch config → select VLAN database.  
(create a new VLAN (VLAN no 2, VLAN name brs)  
and add.)
- 6) Go to interface Ethernet 6/1.

Figure 71: Observation Book I

- 7) VLAN database  $\rightarrow$  VLAN 2, name ~~bans~~  $\rightarrow$  exit.
- 8) config terminal  $\rightarrow$  interface Fa 0/0.1  $\rightarrow$  encapsulation dot1q  
2  $\rightarrow$  ip address 192.168.2.1 255.255.255.0  $\rightarrow$  no shut  
exit  $\rightarrow$  exit
- 9) Enter show IP route.
- 10) Ping from one device to another.

### Results

show IP route

C. 192.168.1.0/24 is directly connected Fa 0/0  
C. 192.168.2.0/24  $\cdots \cdots \cdots$  Fa 0/0.1

PC 0

Ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data.

Request timed out

Reply from 192.168.2.2 bytes=32 time=3ms TTL=127

ping stats.

### Observation:

The VLAN experiment involves creating and configuring VLAN to segment a network, assigning IPs to devices for seamless intra VLAN communication and using dot1q encapsulation for inter VLAN connectivity to communicate through a single trunk link. This experiment highlights the importance of VLANs in optimizing and managing modern networks effectively.

Redacted  
21/07/2023

Figure 72: Observation Book 2

## **Program 12**

To demonstrate communication between two devices using a wireless LAN.

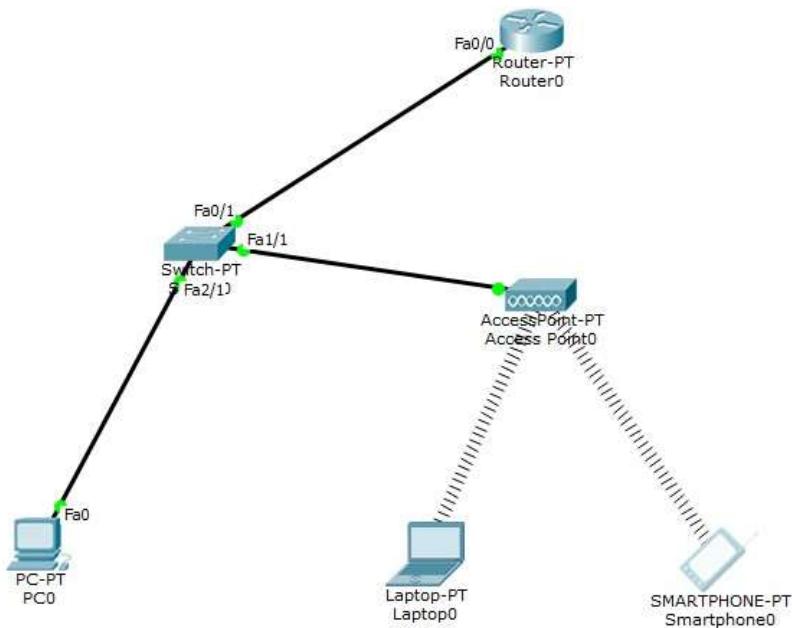


Figure 73: Topology

A screenshot of a Windows-style window titled 'Command Prompt'. The window shows the output of a ping command issued from 'PC-PT PC0'. The command 'ping 10.0.0.3' was entered, followed by the output of the ping request and response. The output is as follows:

```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.3

Pinging 10.0.0.3 with 32 bytes of data:

Reply from 10.0.0.3: bytes=32 time=3ms TTL=128
Reply from 10.0.0.3: bytes=32 time=15ms TTL=128
Reply from 10.0.0.3: bytes=32 time=21ms TTL=128
Reply from 10.0.0.3: bytes=32 time=9ms TTL=128

Ping statistics for 10.0.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 9ms, Maximum = 31ms, Average = 19ms

PC>|
```

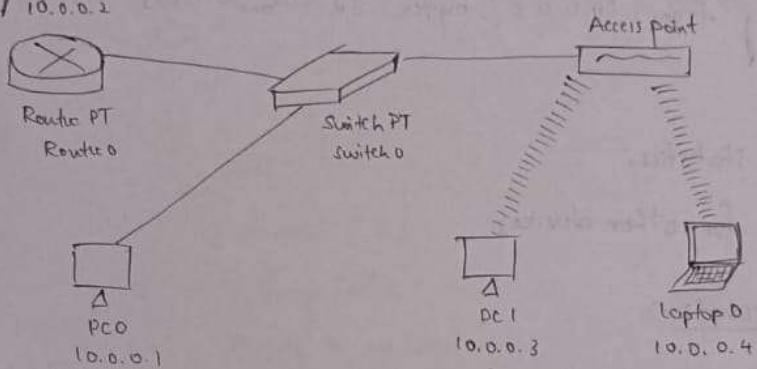
Figure 74: Output

Experiment no 12:

To construct a WLAN and make the nodes communicate wirelessly.

Aim: To show WLAN is effective for wireless communications.

Topology:



Procedure:

- 1) Open Cisco Packet Tracer.
- 2) Construct the above topology.
- 3) Configure access point: Port 1 → SSID name - any non  
SSID → code select WEP : 10 digit (0 1 2 3 4 5 6 7 8 9)
- 4) Configure PC1 and Laptop with wireless standards.
- 5) Switch off the device, drag the existing PT-host NM1-1A to  
the component listed in LHS. Drag the WMP 200  
wireless interface to the empty port switch on the device.
- 6) In config tab a new wireless interface would have  
been added. Now configure SSID → code. select WEP →  
10 digit WEP key. IP address and gateway to the  
device.

Figure 75: Observation book I

- 7) Ping from either devices
- 8) Setup PCo , route as normally done.

Result:

PCo:

l-ping 10.0.0.3

pinging 10.0.0.3 with 32 bytes of data

Reply from 10.0.0.3 , bytes = 32 time = 19ms TTL = 128

ping statistics.

Some for other devices.

Observation:

The experiment demonstrates the creation of a wireless network using an access point configured with an SSID, WEP encryption and a 10 digit key. Devices like PCs and laptops were configured with wireless adapter, IP addresses and gateways to enable communication.

The success of ping test between devices verify the setup, highlighting the simplicity and efficiency of WLAN connections for wireless communications.

Ritu  
21/12/24

Figure 76: Observation book 2

### **Program 13**

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

Code:

```
#include <stdio.h>
#include <stdint.h>
#define CRC_POLY 0x11021
#define INITIAL_CRC 0xFFFF

uint16_t compute_crc(uint8_t *data, size_t length) {
    uint16_t crc = INITIAL_CRC;
    for (size_t i = 0; i < length; i++) {
        crc ^= (data[i] << 8);
        for (int j = 0; j < 8; j++) {
            if (crc & 0x8000) {
                crc = (crc << 1) ^ CRC_POLY;
            } else {
                crc <<= 1;
            }
        }
    }
    return crc & 0xFFFF;
}

int check_crc(uint8_t *data, size_t length, uint16_t expected_crc) {
    uint16_t computed_crc = compute_crc(data, length);
    return (computed_crc == expected_crc);
}

int main() {
    uint8_t data[] = "Hello, World!";
    size_t data_length = sizeof(data) - 1;
    printf("Data: %s\n", data);
    uint16_t crc = compute_crc(data, data_length);
    printf("Computed CRC-CCITT: 0x%04X\n", crc);
    uint8_t received_data[] = "Hello, World!";
    size_t received_length = sizeof(received_data) - 1;
    if (check_crc(received_data, received_length, crc)) {
        printf("Data received correctly with no errors.\n");
    } else {
        printf("Error detected in received data!\n");
    }
    return 0;
}
```

Output:

```
Data: Hello, World!
Computed CRC-CCITT: 0x67DA
Data received correctly with no errors.
```

Figure 77: Output of CRC-CCIT

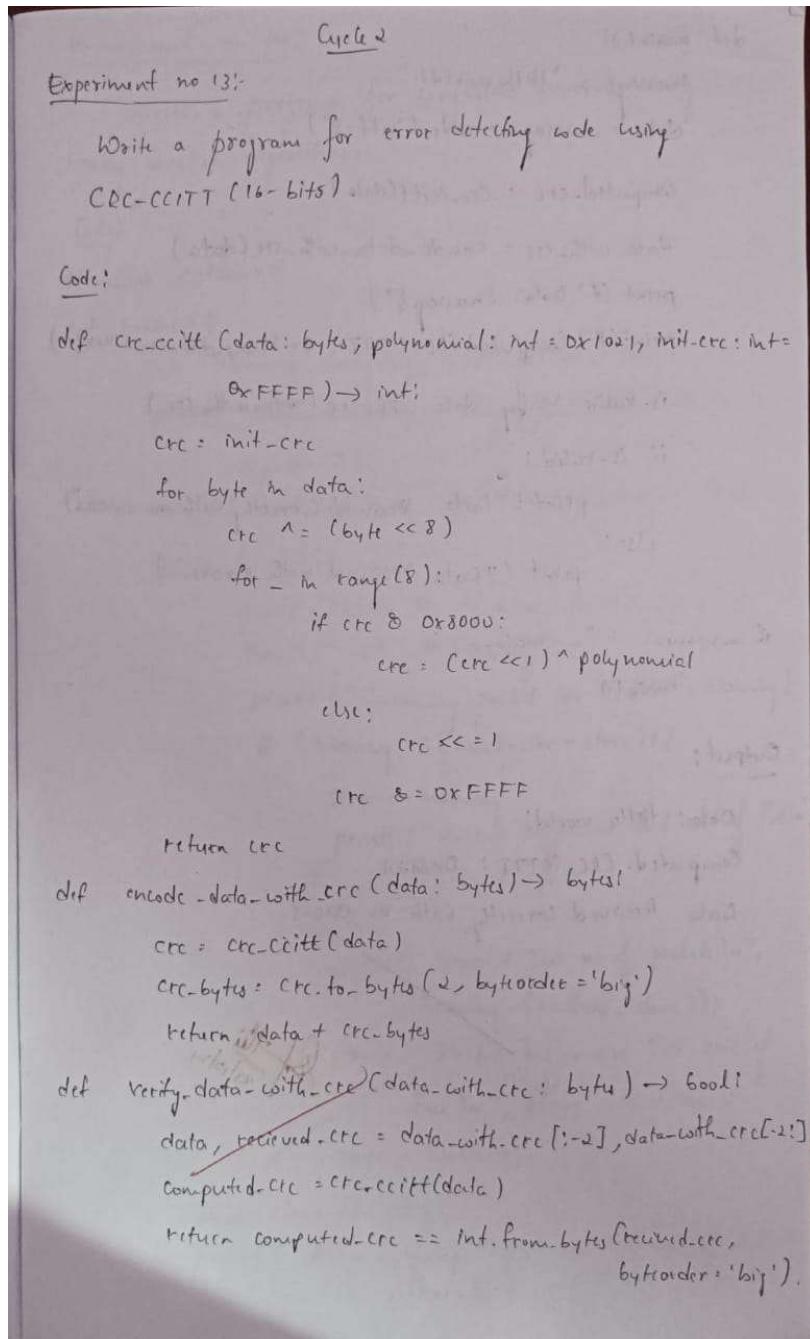


Figure 78: Observation Book I

```

def main():
    message = "Hello world!"
    data = message.encode('utf-8')
    computed_crc = crc_ccitt(data)
    data_with_crc = encode_data_with_crc(data)
    print(f"Data: {message}")
    print(f"Computed CRC-CCITT: 0x{computed_crc:04X}")
    is_valid = verify_data_with_crc(data_with_crc)
    if is_valid:
        print("Data received correctly with no errors")
    else:
        print("Data received with errors")

if __name__ == "__main__":
    main()

```

Output:

```

Data: Hello world!
Computed. CRC-CCITT : 0x882A
Data Received correctly with no errors.

```

✓  
Rithika  
11/12/24

Figure 79: Observation Book 3

### **Program 14**

Write a program for congestion control using Leaky Bucket algorithm

Code:

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

```
int main(){
    int incoming, outgoing, buck_size, n, store = 0;
    printf("Enter bucket size, outgoing rate and no of inputs: ");
    scanf("%d %d %d", &buck_size, &outgoing, &n);

    while (n != 0) {
        printf("Enter the incoming packet size : ");
        scanf("%d", &incoming);
        printf("Incoming packet size %d\n", incoming);
        if (incoming <= (buck_size - store)){
            store += incoming;
            printf("Bucket buffer size %d out of %d\n", store, buck_size);
        } else {
            printf("Dropped %d no of packets\n", incoming - (buck_size - store));
            printf("Bucket buffer size %d out of %d\n", store, buck_size);
            store = buck_size;
        }
        store = store - outgoing;
        printf("After outgoing %d bytes left out of %d in buffer\n", store, buck_size);
        n--;
    }
}
```

Output:

```
Enter bucket size, outgoing rate and no of inputs: 10 3 3
Enter the incoming packet size : 5
Incoming packet size 5
Bucket buffer size 5 out of 10
After outgoing 2 bytes left out of 10 in buffer
Enter the incoming packet size : 5
Incoming packet size 5
Bucket buffer size 7 out of 10
After outgoing 4 bytes left out of 10 in buffer
Enter the incoming packet size : 7
Incoming packet size 7
Dropped 1 no of packets
Bucket buffer size 4 out of 10
After outgoing 7 bytes left out of 10 in buffer
```

Figure 80: Output for Leaky Bucket algorithm

Experiment - no 14:

Write a program for congestion control using  
Leaky bucket algorithm.

Code:

```
#include <stdio.h>

int main() {
    int incoming, outgoing, buck-size, n, store = 0;
    printf("Enter bucket size, outgoing rate and no of\n");
    inputs: ");
    scanf("%d %d %d", &buck-size, &outgoing, &n);

    while (n != 0) {
        printf("Enter the incoming packet size: ");
        scanf("%d", &incoming);
        printf("Incoming packet size %d in", incoming);
        if (incoming <= buck-size - store) {
            store += incoming;
            printf("Bucket buffer size %d out of %d in", store, buck-size);
        } else {
            printf("Dropped %d no of packets in", incoming - (buck-size - store));
            printf("Bucket buffer size %d out of %d in", store, buck-size);
            store = buck-size;
        }
        store = store - outgoing;
        printf("After outgoing %d bytes left out of %d in", store, buck-size);
        n--;
    }
}
```

Figure 81: Observation Book I

Output  
 Enter bucket-size, outgoing rate and no of inputs : 10 3  
 Enter the incoming packet size : 5  
 Incoming packet size 5  
 Bucket buffer size 5 out of 10  
 After outgoing 2 bytes left out of 10 in buffer.  
 Enter the incoming packet size : 15  
 Incoming packet size : 5  
 Bucket buffer size 1 out of 10  
 After outgoing 4 bytes left out of 10 in buffer.  
 Enter the incoming packet size : 7  
 Incoming packet size 7  
 Dropped 1 no of packet  
 Bucket buffer size 4 out of 10.  
 After outgoing 7 bytes left out of 10 in buffer.

(X) the  
 31/10/13

Figure 82: Observation Book 2

### **Program 15**

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

#### Client.py

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

#### Server.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
print ("The server is ready to receive")
while 1:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    file = open(sentence, "r")
    l = file.read(1024)
    connectionSocket.send(l.encode())
    file.close()
    connectionSocket.close()
```

#### Output

```
= RESTART: C:/Users/Dell/Desktop/5th sem/CN/LAB/Client.py
Enter file name: example.txt
From Server: Hello World!!
|
```

Figure 83: Output of Client.py

```
===== RESTART: C:/Users/Dell/Desktop/5th sem/CN/LAB/Server.py =====
The server is ready to receive
```

Figure 84: Output of Server.py

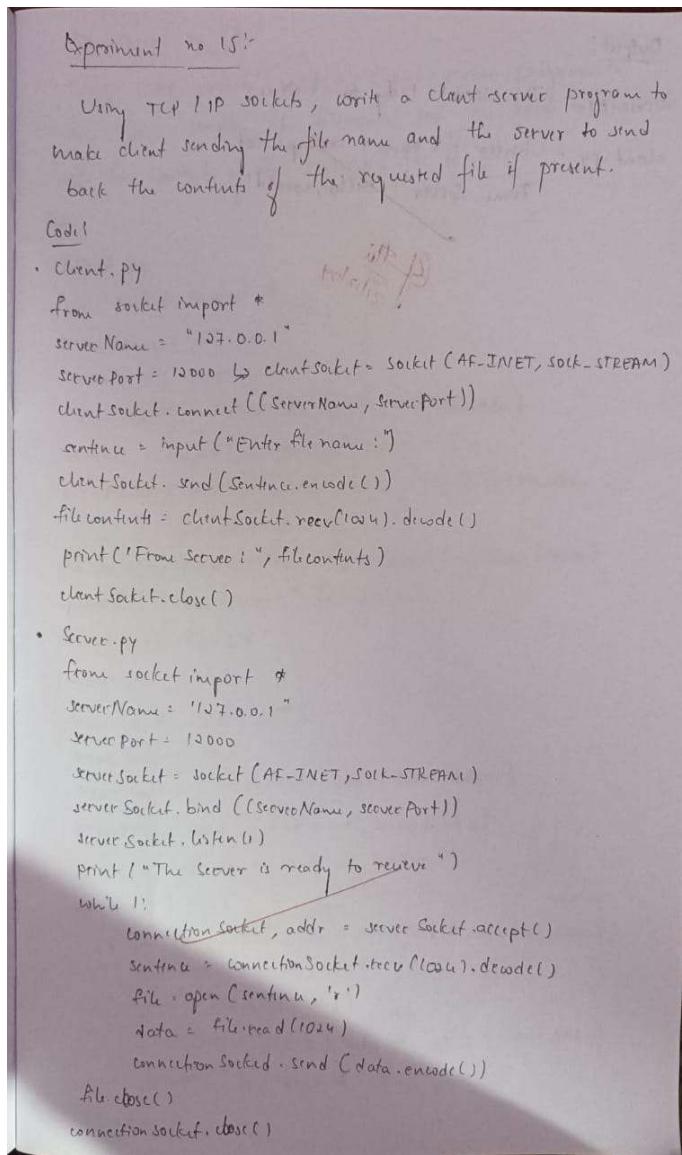


Figure 85: Observation Book

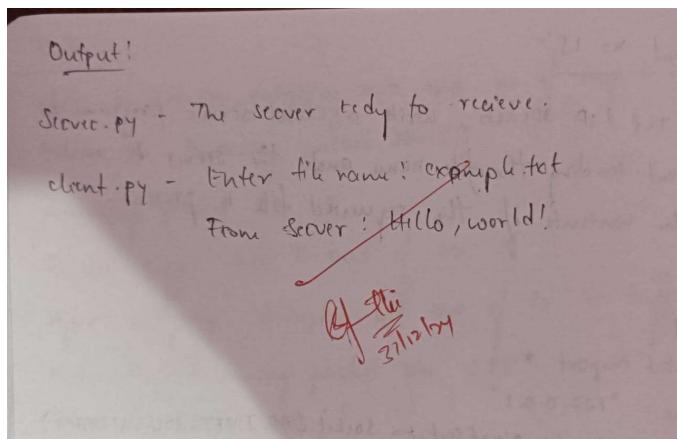


Figure 86: Observation Book

### **Program 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.

#### ClientUDP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("Enter file name")
clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))
filecontents,serverAddress = clientSocket.recvfrom(2048)
print ('From Server:', filecontents)
clientSocket.close()
```

#### ServerUDP.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print ("The server is ready to receive")
while 1:
    sentence,clientAddress = serverSocket.recvfrom(2048)
    file=open(sentence,"r")
    l=file.read(2048)
    serverSocket.sendto(bytes(l,"utf-8"),clientAddress)
    print("sent back to client",l)
    file.close()
```

#### Output

```
= RESTART: C:/Users/Dell/Desktop/5th sem/CN/LAB/ClientUDP.py
Enter file name: example.txt
From Server: b'Hello World!!'
```

Figure 87: Output of ClientUDP.py

```
= RESTART: C:/Users/Dell/Desktop/5th sem/CN/LAB/ServerUDP.py
The server is ready to receive
|sent back to client Hello World!!
```

Figure 88: Output of ServerUDP.py

Experiment no 16:

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

Code:

• Client.py

```
from socket import *  
serverName = '127.0.0.1'  
serverPort = 12000  
clientSocket = socket(AF_INET, SOCK_DGRAM)  
sentence = input("Enter file name : ")  
clientSocket.sendto(bytes(sentence, "utf-8"), (serverName,  
serverPort))  
fileContents, serverAddress = clientSocket.recvfrom(2048)  
print("From server:", fileContents)  
clientSocket.close()
```

• Server.py

```
from socket import *  
serverPort = 12000  
serverSocket = socket(AF_INET, SOCK_DGRAM)  
serverSocket.bind(( "127.0.0.1", serverPort ))  
print("The server is ready to receive")  
while 1:  
    sentence, clientAddr = serverSocket.recvfrom(2048)  
    file = open(sentence, 'r')  
    data = file.read(2048)  data  
    serverSocket.sendto(data, clientAddr)  
    print("sent back to client:", data)  
    file.close()
```

Figure 89: Observation Book

Output:

server.py - The server is ready to receive  
Sent back to client: Hello, world!

client.py - Enter file name: example.txt  
From server: Hello, world!

By the  
31/12/2014

Figure 90: Observation Book