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 Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index
(Cycle – 1)

Index
(Cycle – 2)

Github Link: GitHub Link

Program 1

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

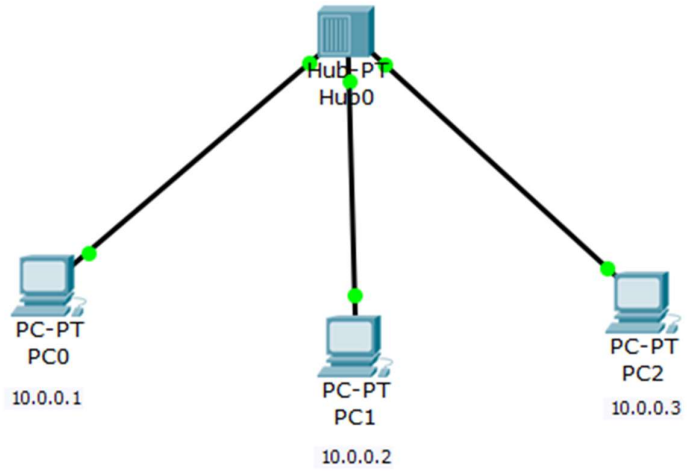


Figure 1: Using Hub

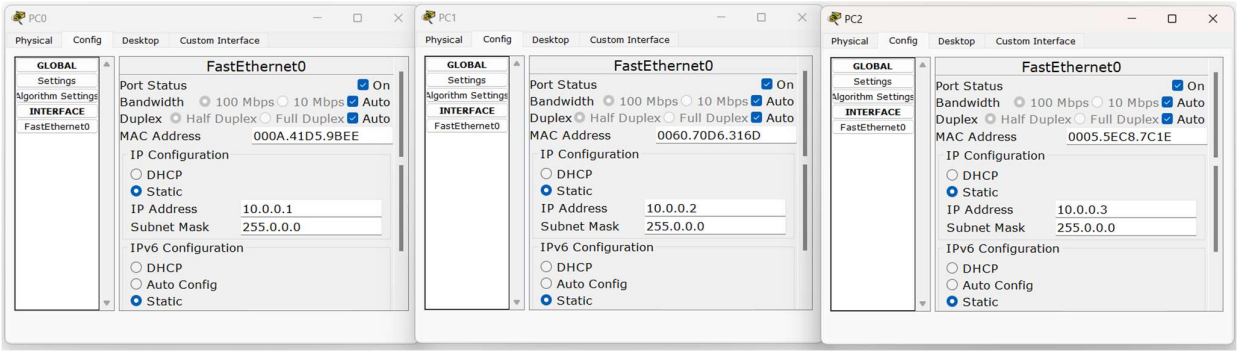


Figure 2: IP Addresses of the 3 PCs

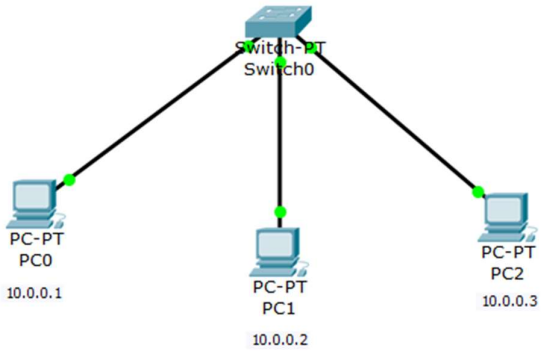


Figure 3: Using Switch

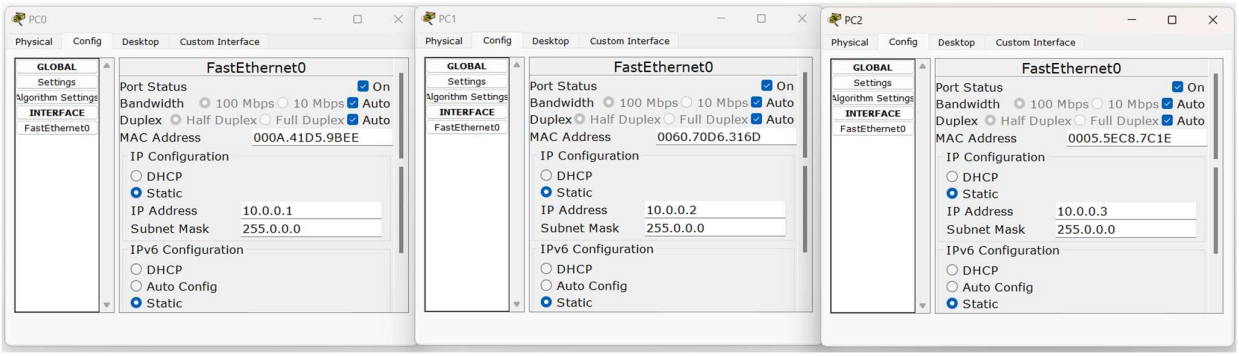
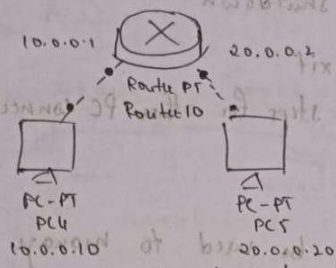


Figure 4: IP Addresses of the 3 PCs

Experiment 2

1. PC to routers:



Aim: To create simple network consisting of 2 PCs connected to the router, facilitating communication between the two PCs through 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 the 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 5: Observation Book 1

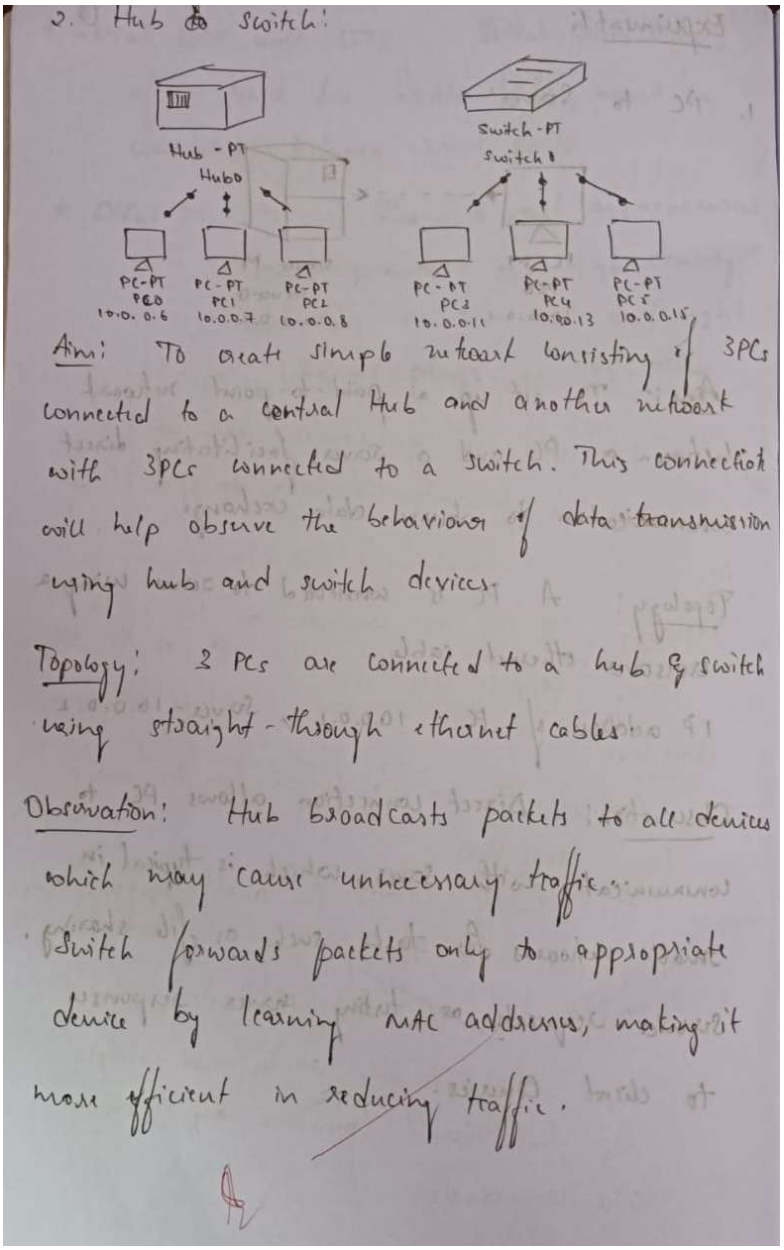


Figure 6: Observation Book 2

Program 2

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

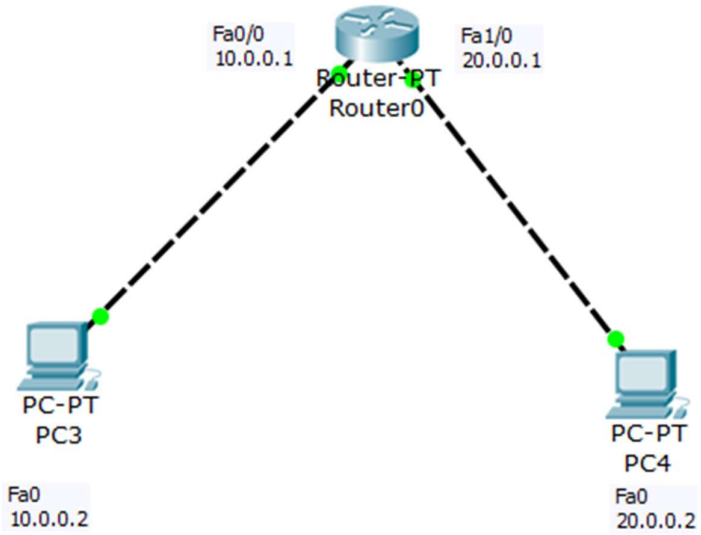


Figure 7: Topology

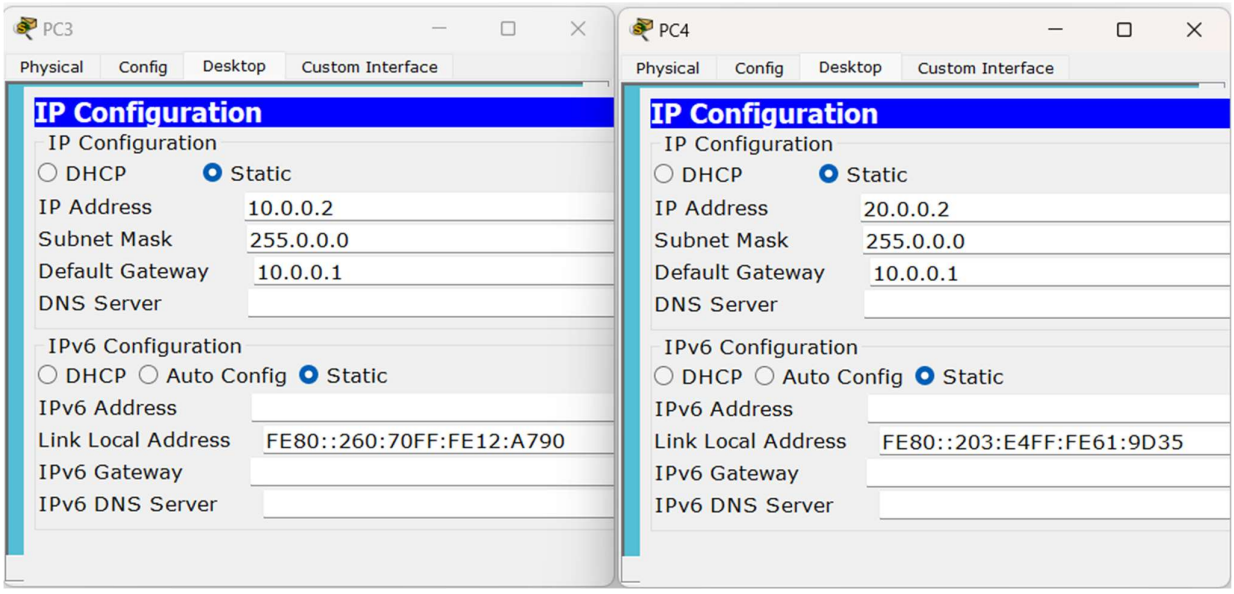


Figure 8: IP addresses

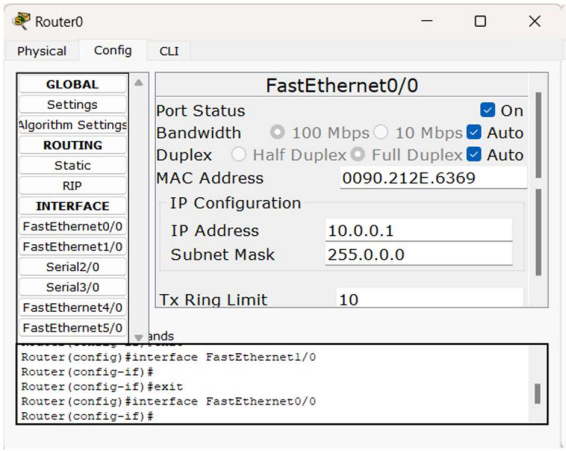


Figure 9: Router Fa0/0

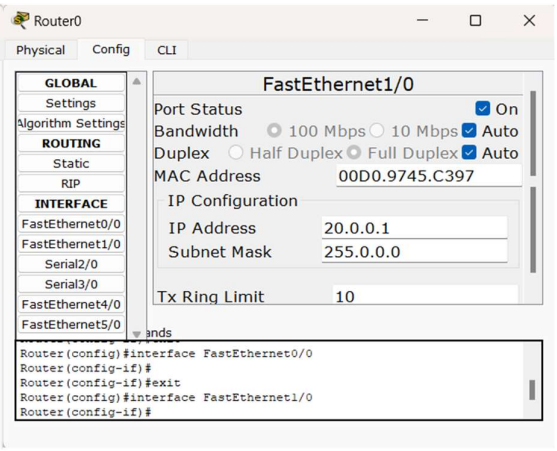


Figure 10: Router Fa1/0

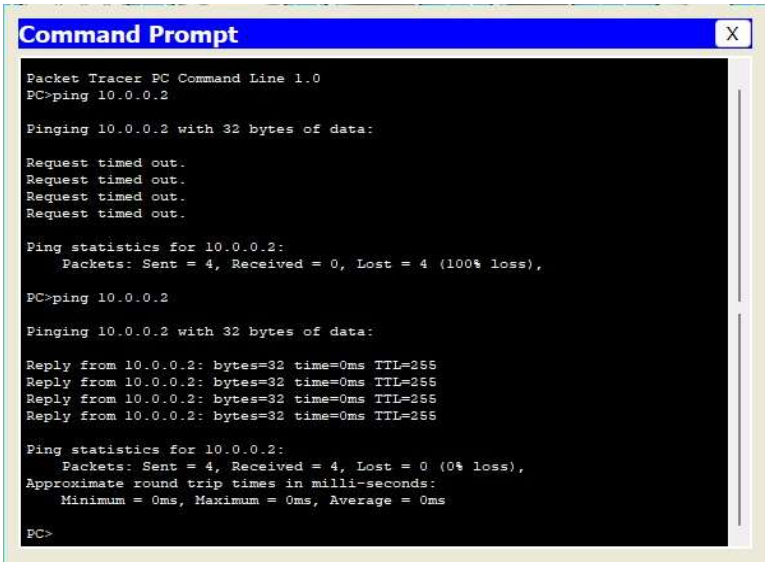
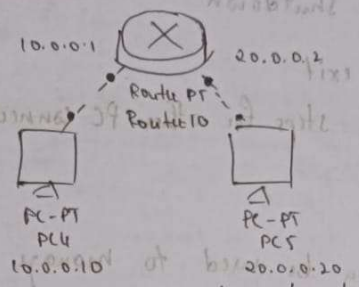


Figure 11: ping command output

Experiment 2 (5)

1. PC to routers:



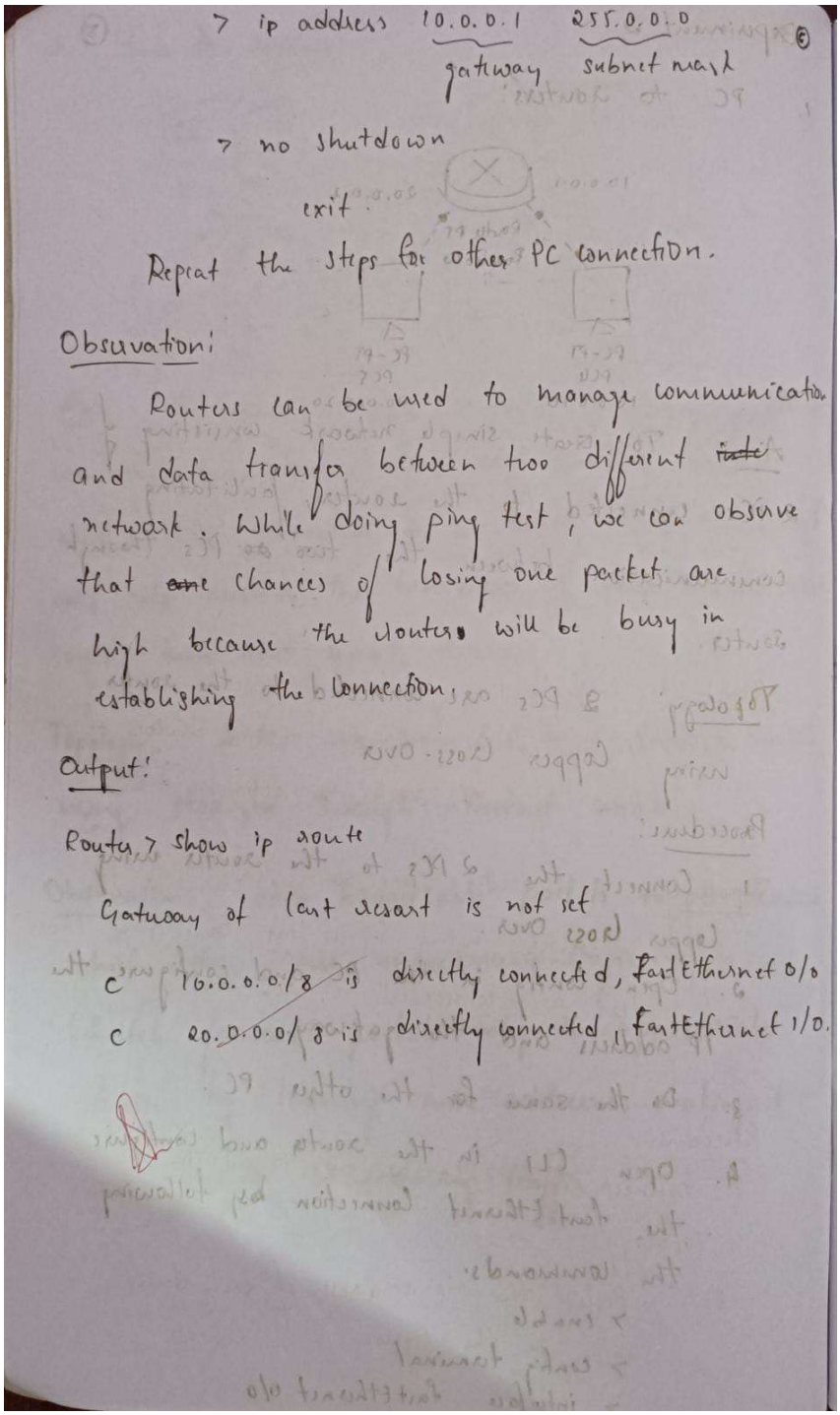
Aim: To create simple network consisting of 2 PCs connected to the router, facilitating communication between the two PCs through router.

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 the 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


> ip address 10.0.0.1 255.0.0.0
gateway subnet mask

> no shutdown

exit

Repeat the steps for other PC connection.

Observation:

Routers can be used to manage communication and data transfer between two different ~~route~~ network. While doing ping test, we can observe that ~~one~~ chances of losing one packet are high because the routers will be busy in establishing the connection.

Output:

Router > show ip route

Gateway of last resort is not set

C 10.0.0.0/8 is directly connected, FastEthernet 0/0

C 20.0.0.0/8 is directly connected, FastEthernet 1/0.

Figure 13: Observation Book 2

Program 3

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

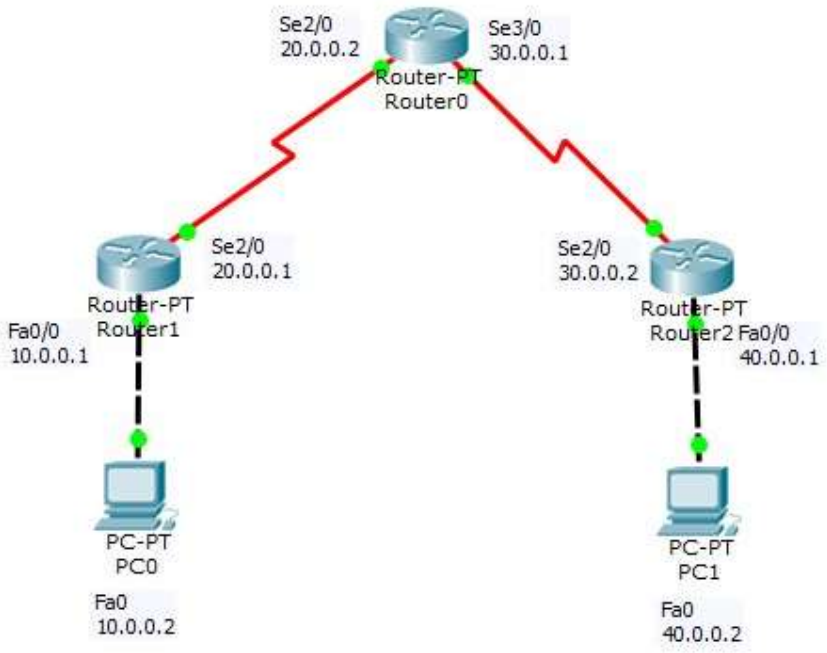


Figure 14: Topology for static route

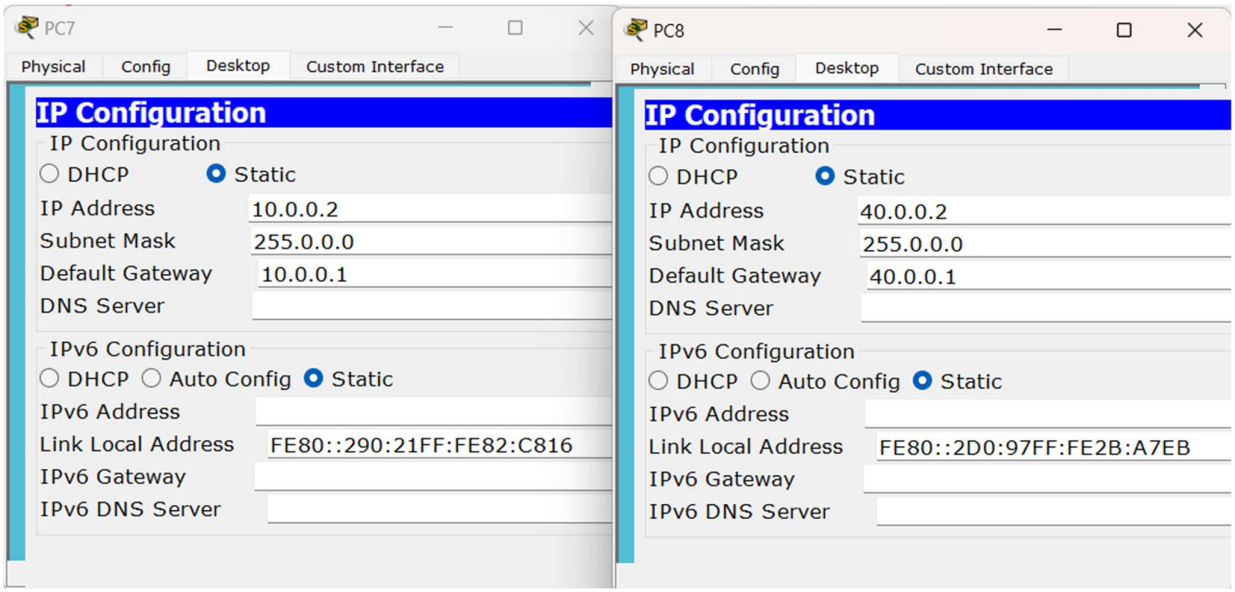


Figure 15: IP addresses

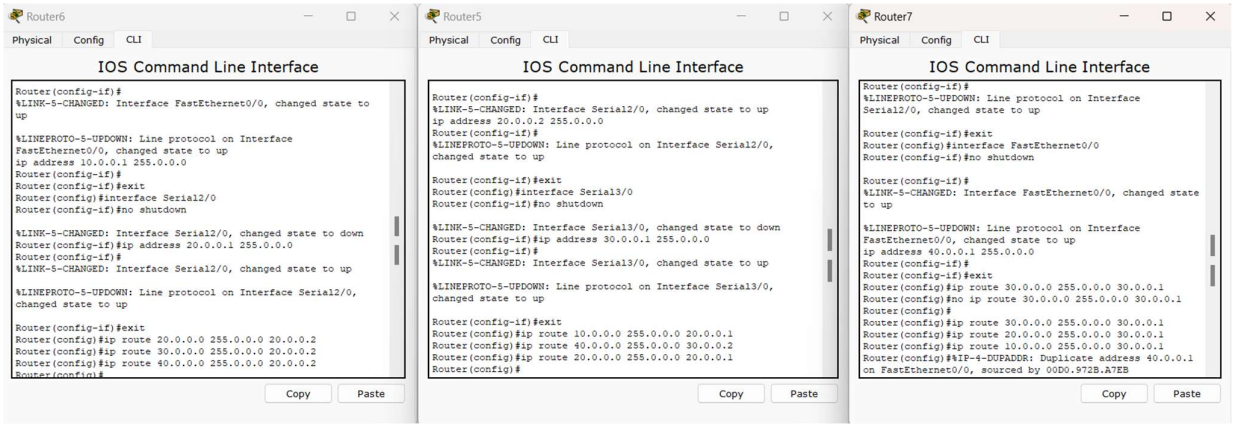


Figure 16: Routers CLI

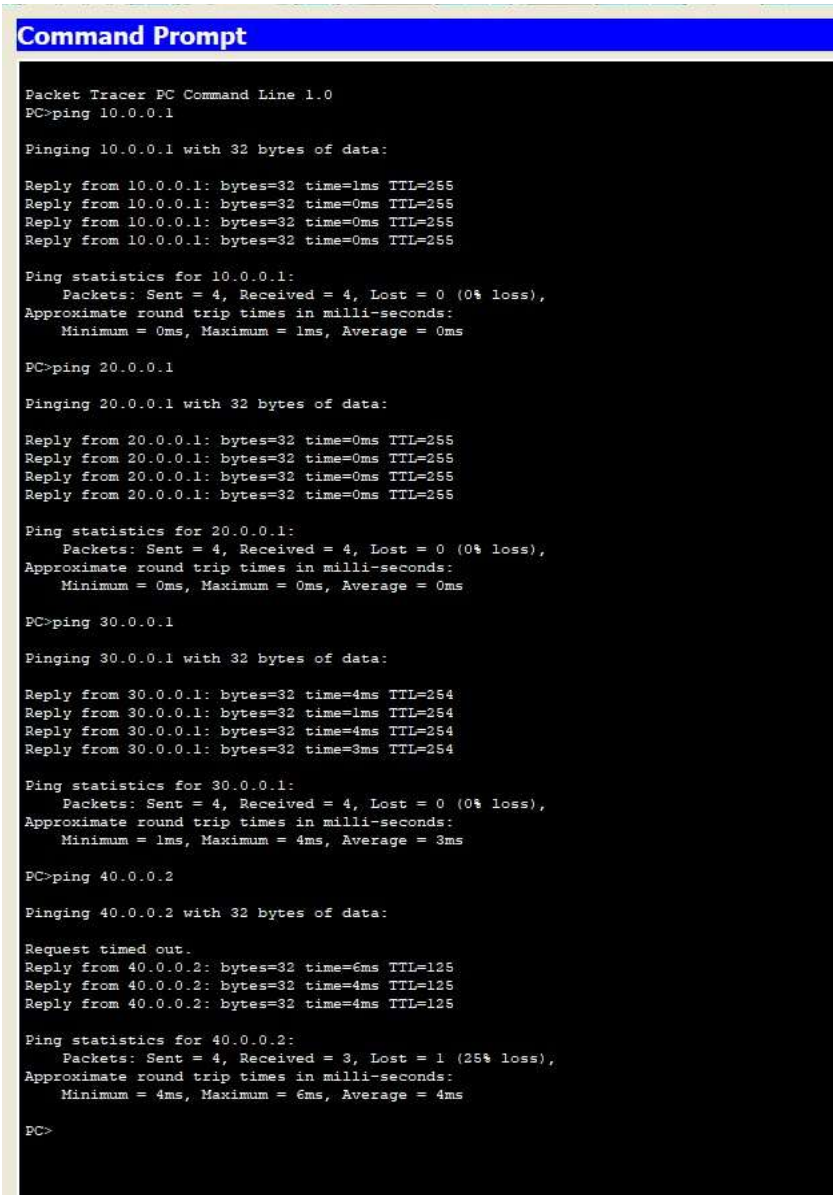
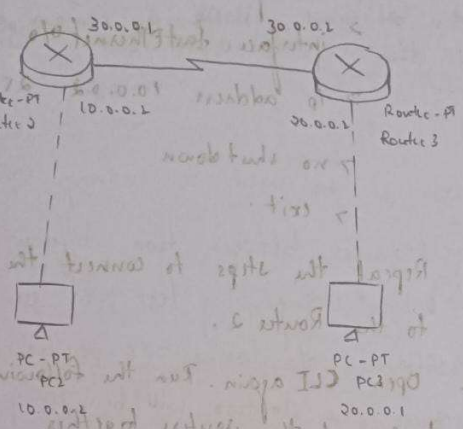


Figure 17: Output

Experiment - 2

1. Connection of two Routers:

Configuration



Aim: (i) Configuration of two routers.

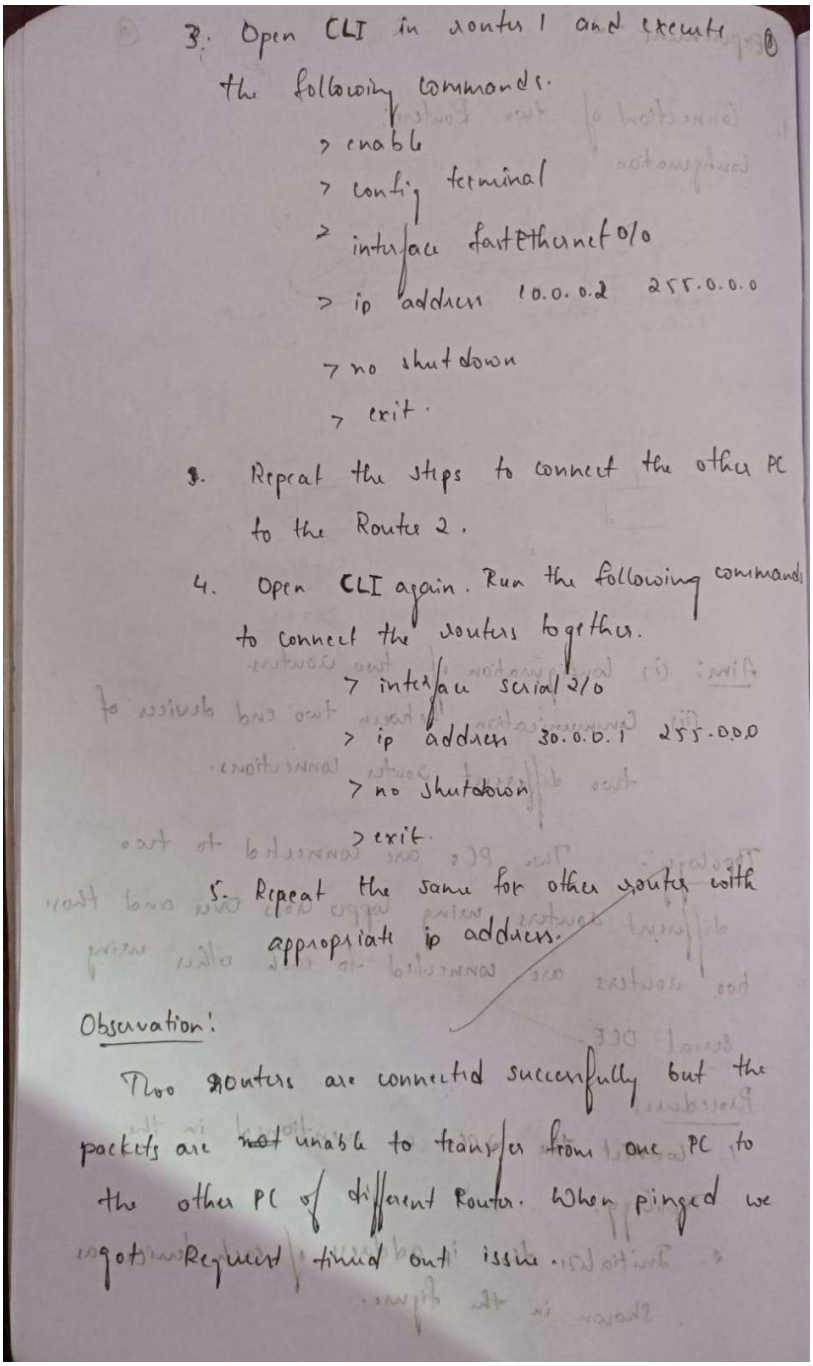
(ii) Communication between two end devices of two different router connections.

Topology: Two PCs are connected to two different routers using Copper Cross Over and those two routers are connected to each other using serial DCE.

Procedure:

1. Connect the 2 PCs as mentioned in the Topology.
2. Initialize the ip address of each devices as shown in the figure.

Figure 18: Observation book 1



3. Open CLI in router 1 and execute the following commands.
   - > enable
   - > config terminal
   - > interface fastethernet 0/0
   - > ip address 10.0.0.2 255.0.0.0
   - > no shutdown
   - > exit.
4. Repeat the steps to connect the other PC to the Router 2.
5. Open CLI again. Run the following commands to connect the routers together.
   - > interface serial 2/0
   - > ip address 30.0.0.1 255.0.0.0
   - > no shutdown
   - > exit.
6. Repeat the same for other routers with appropriate ip addresses.

Observation:

Two routers are connected successfully but the packets are ~~not~~ unable to transfer from one PC to the other PC of different Router. When pinged we got Request timed out issue.

Figure 19: Observation book 2

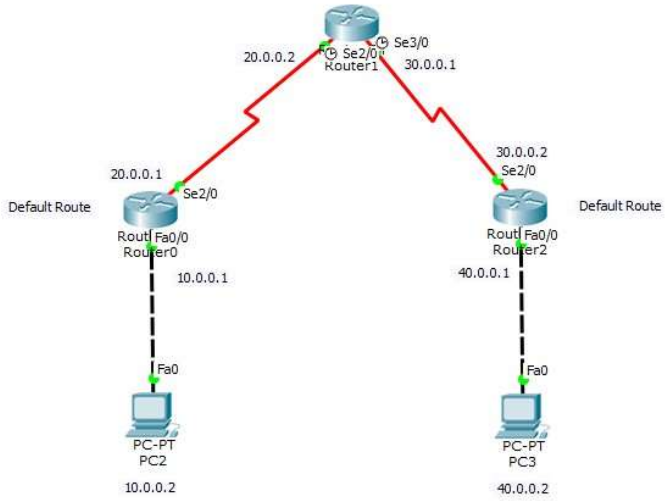


Figure 20: Topology for default route

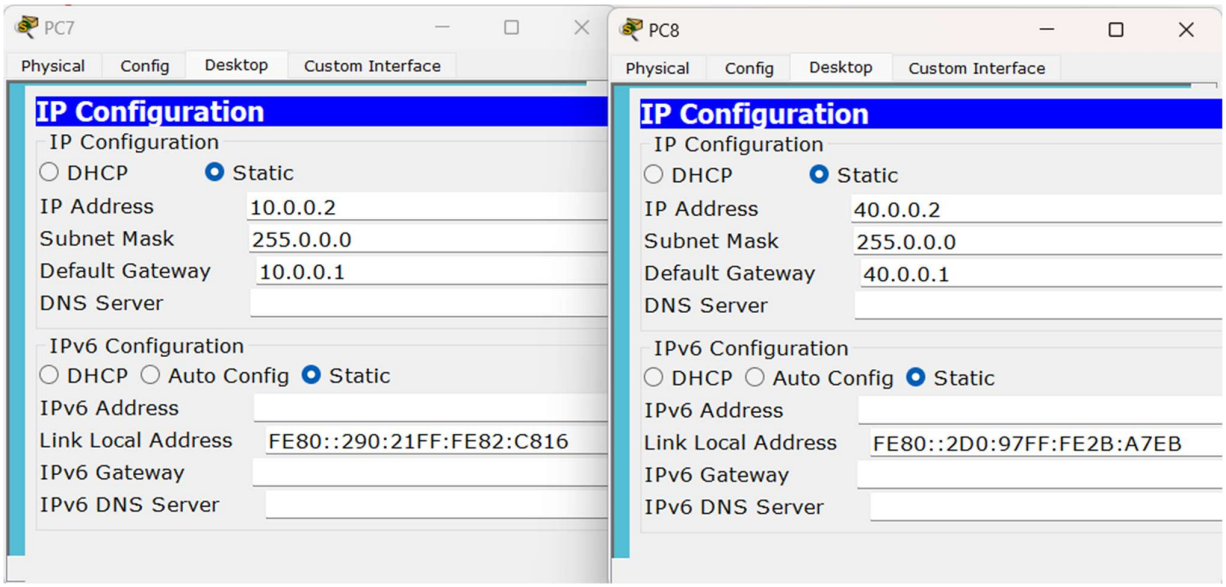


Figure 21: IP addresses

Figure 22: Router CLI Commands

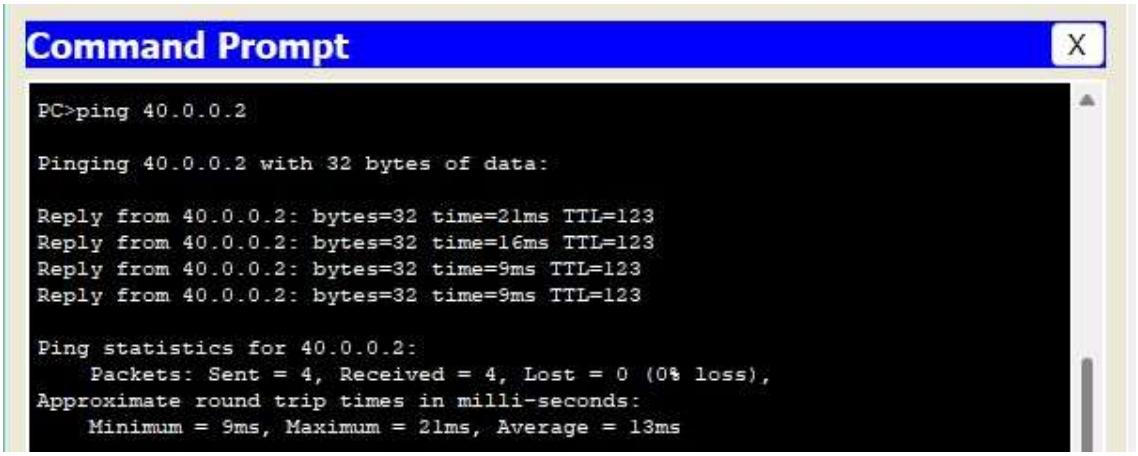


Figure 23: ping command output

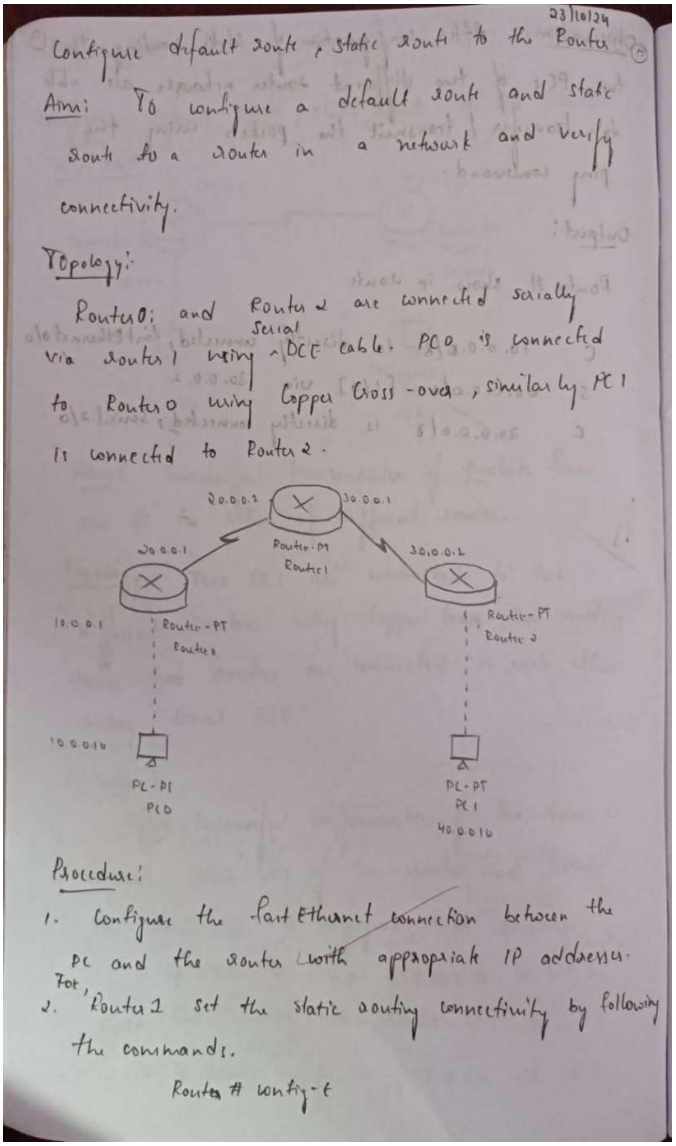
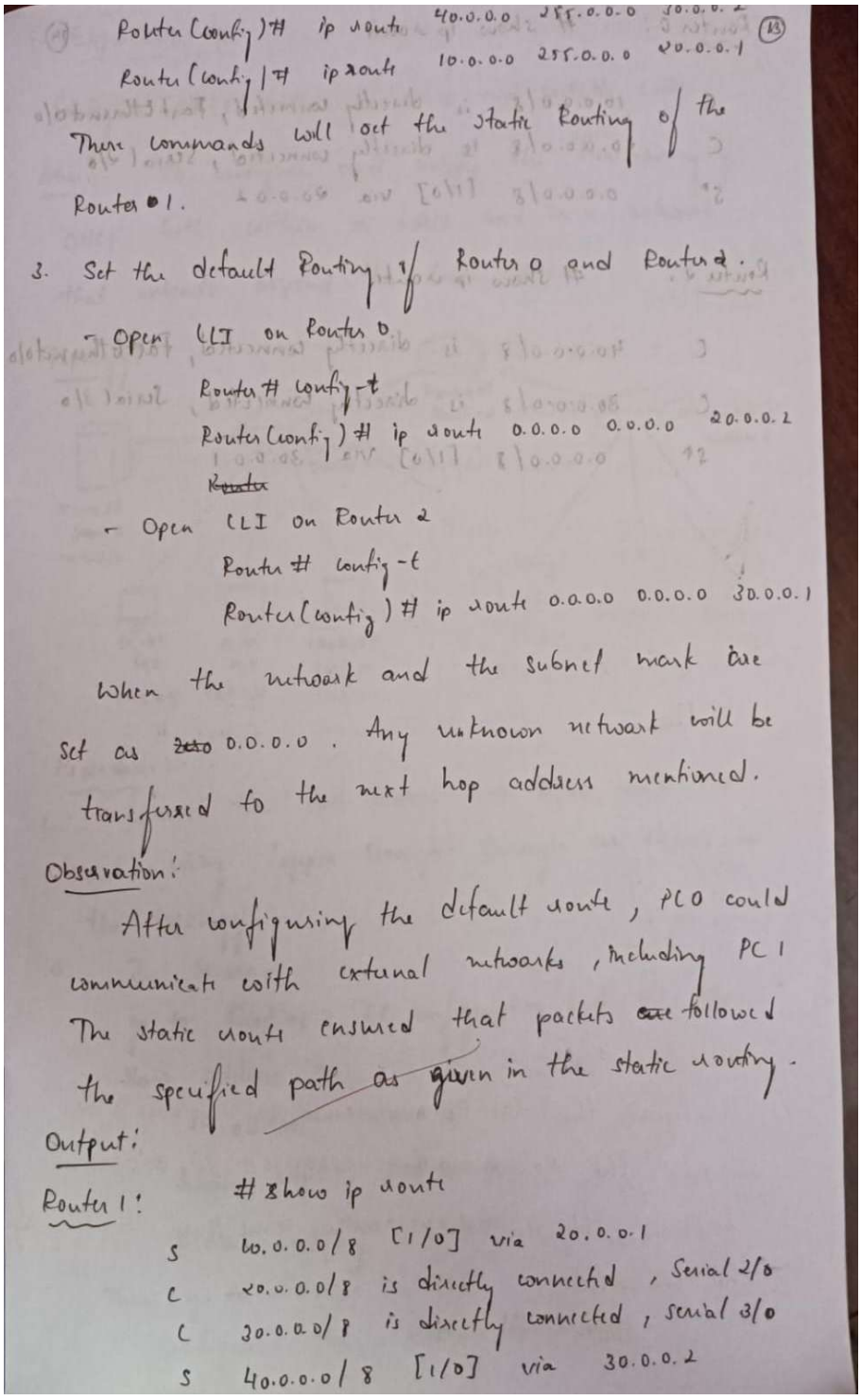


Figure 24: Observation book 1



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

These commands will set the static Routing of the Router 1.

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~~

- 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 ~~zero~~ 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 ~~are~~ followed the specified path as given in the static routing.

Output:

Router1: # 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.

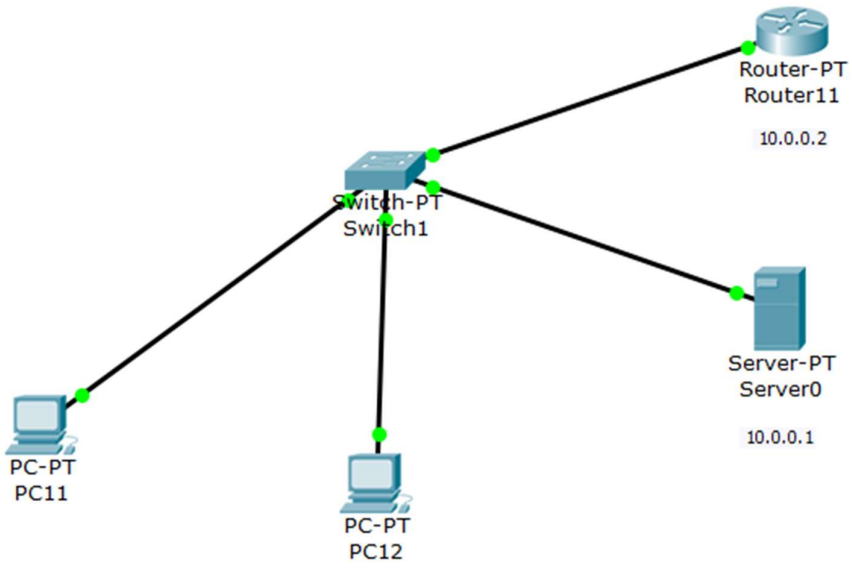


Figure 26: Topology

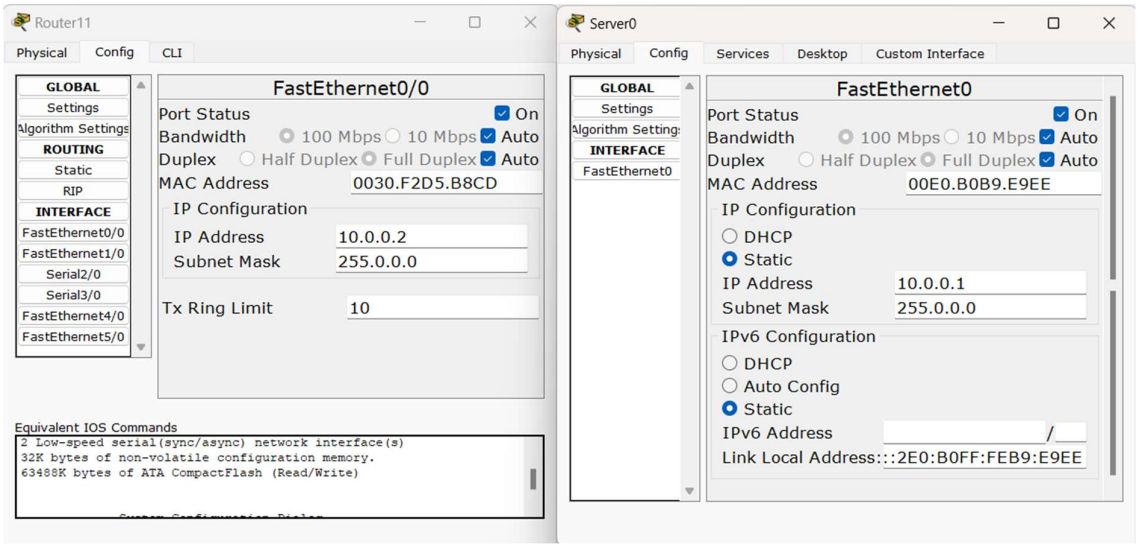


Figure 27: IP Addresses

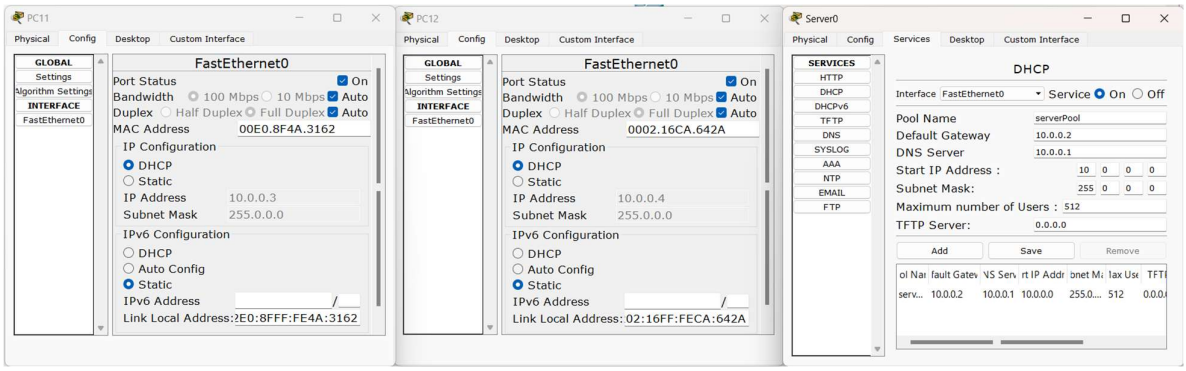


Figure 28: DHCP

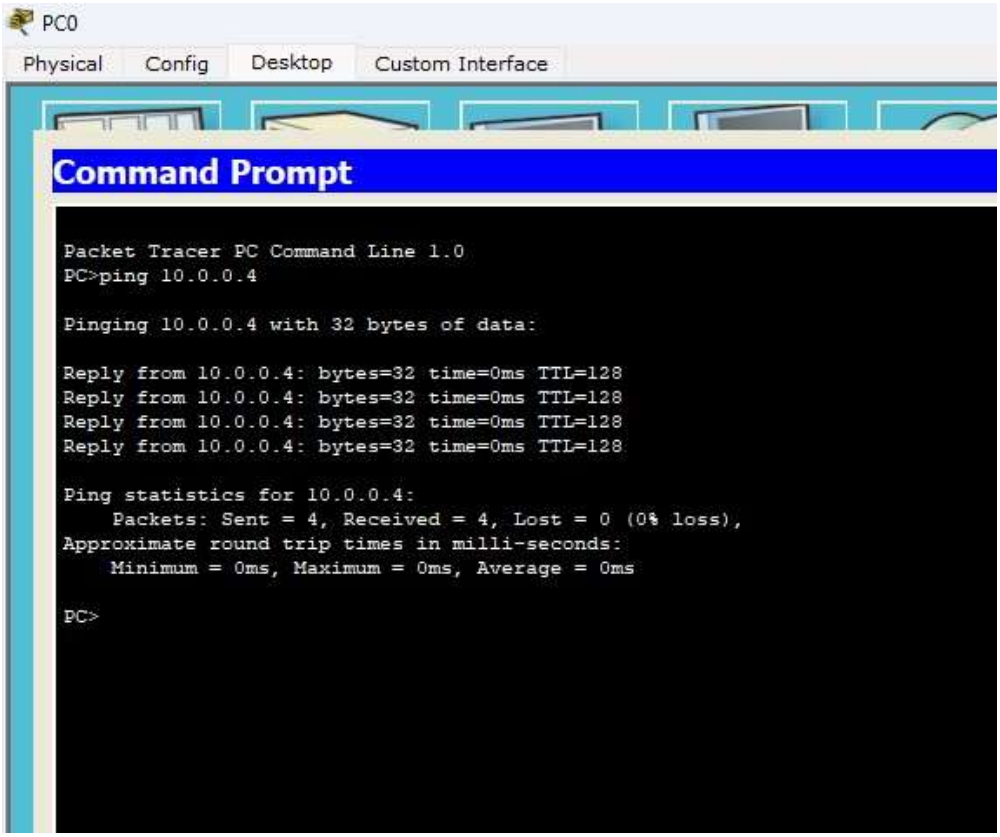


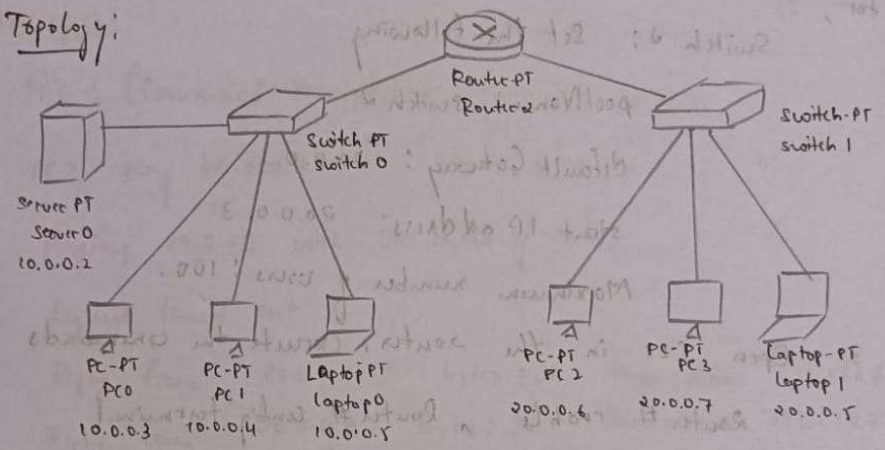
Figure 29: ping command output

Experiment 4:- 13-11-2024

Configure DHCP within a LAN and outside LAN.

Aim: To configure and verify the functioning 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 routers 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 1

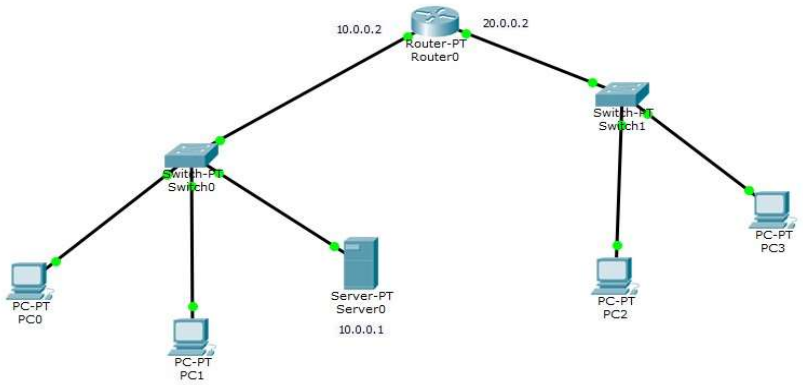


Figure 4: Topology for DHCP in different LAN

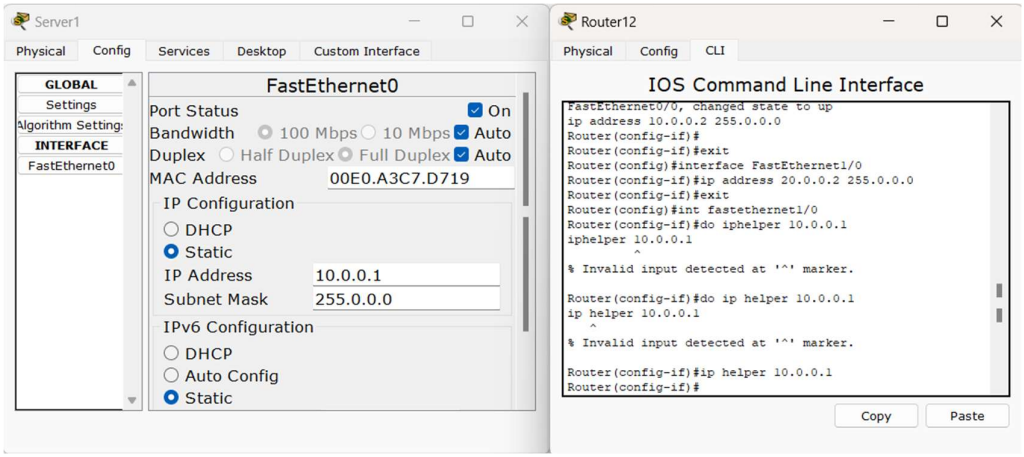


Figure 5: Router CLI and Server IP Address

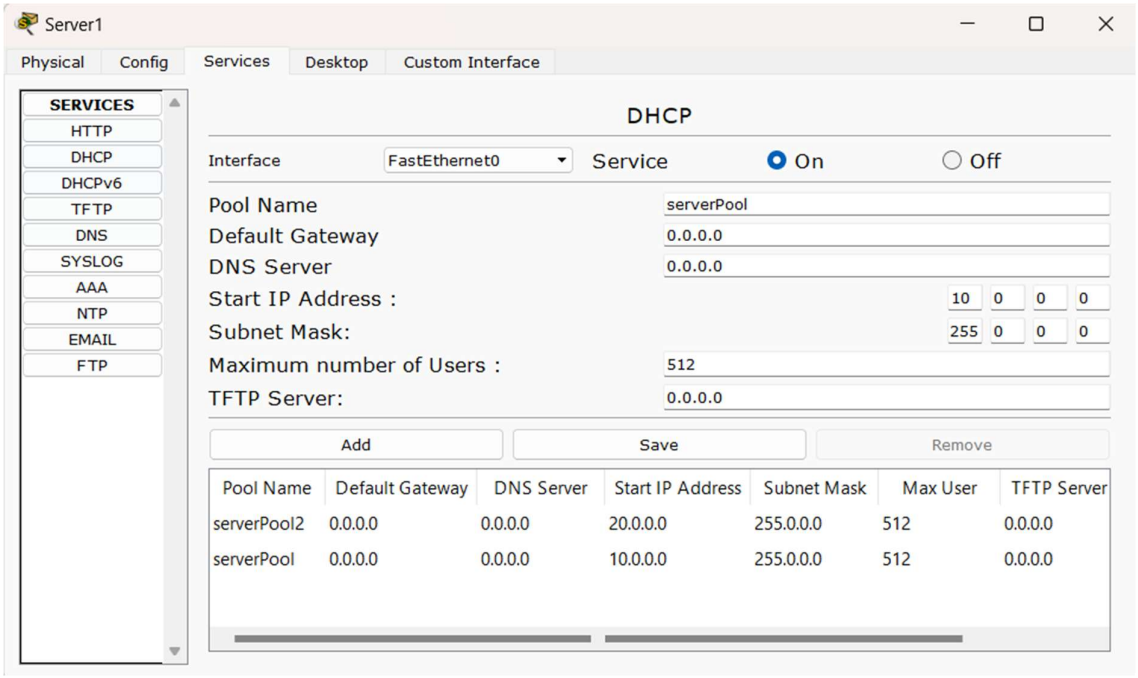


Figure 6: Server Pools of the DHCP Server

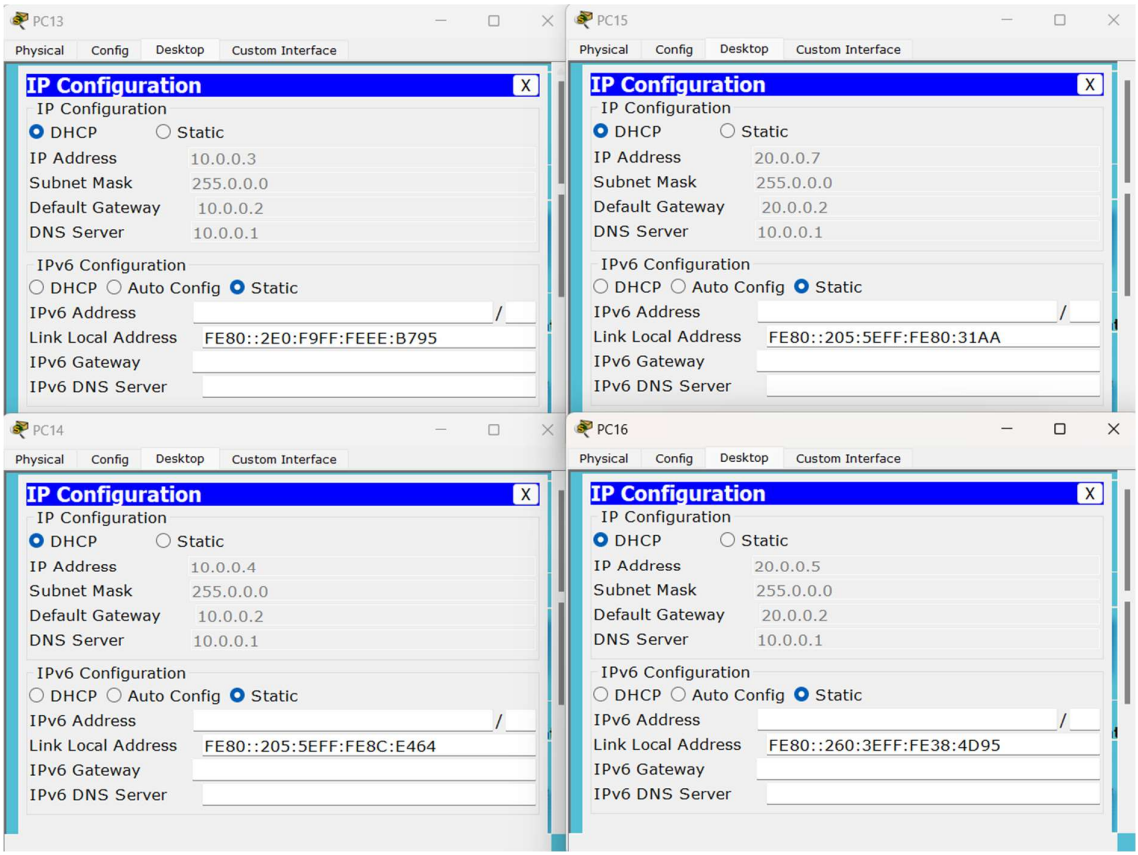


Figure 7: PC IP addresses automatically assigned by DHCP Server

```
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

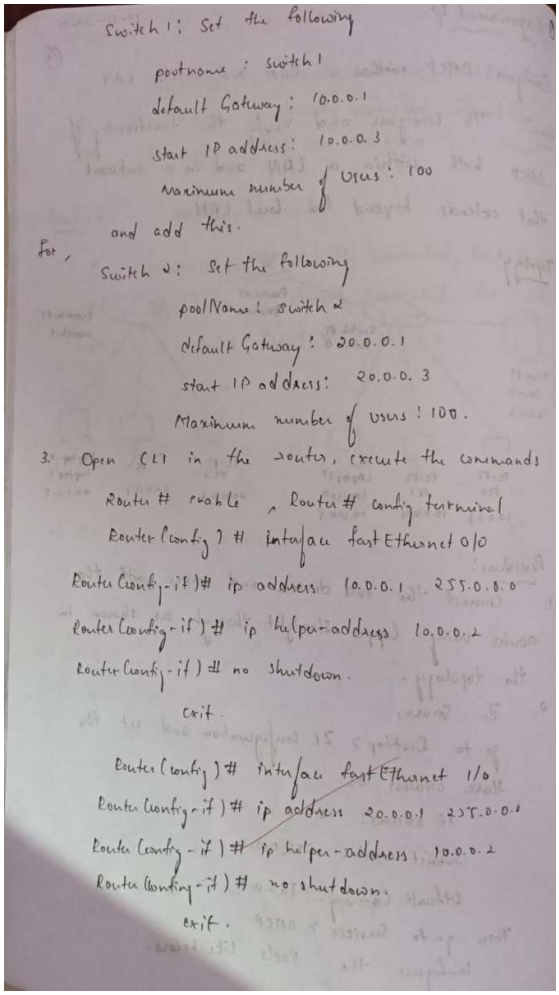


Figure 9: Observation book 1

Observation: Set up the Routes with the fastEthernet (17)
cable connected to the two switches. by 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 = 0ms, Average = 0ms.

13/1/24

Figure 10: Observation book 2

Program 5

To configure RIP routing protocol in Routers.

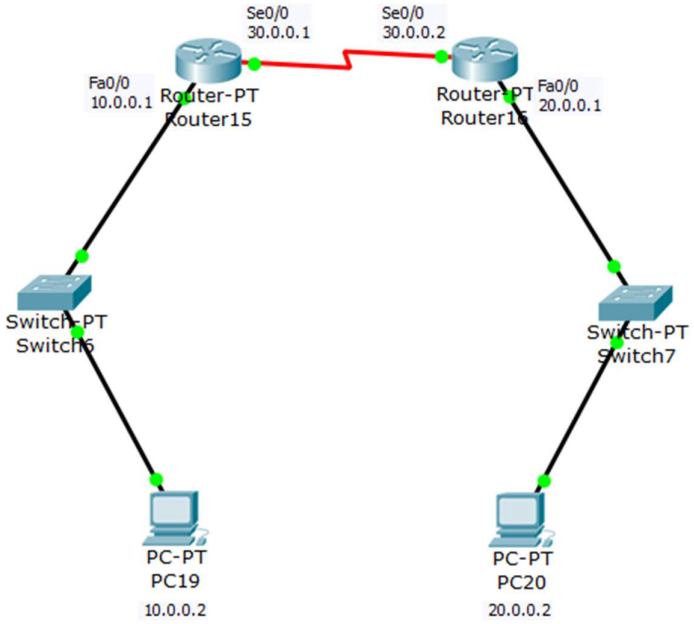


Figure 11: Topology

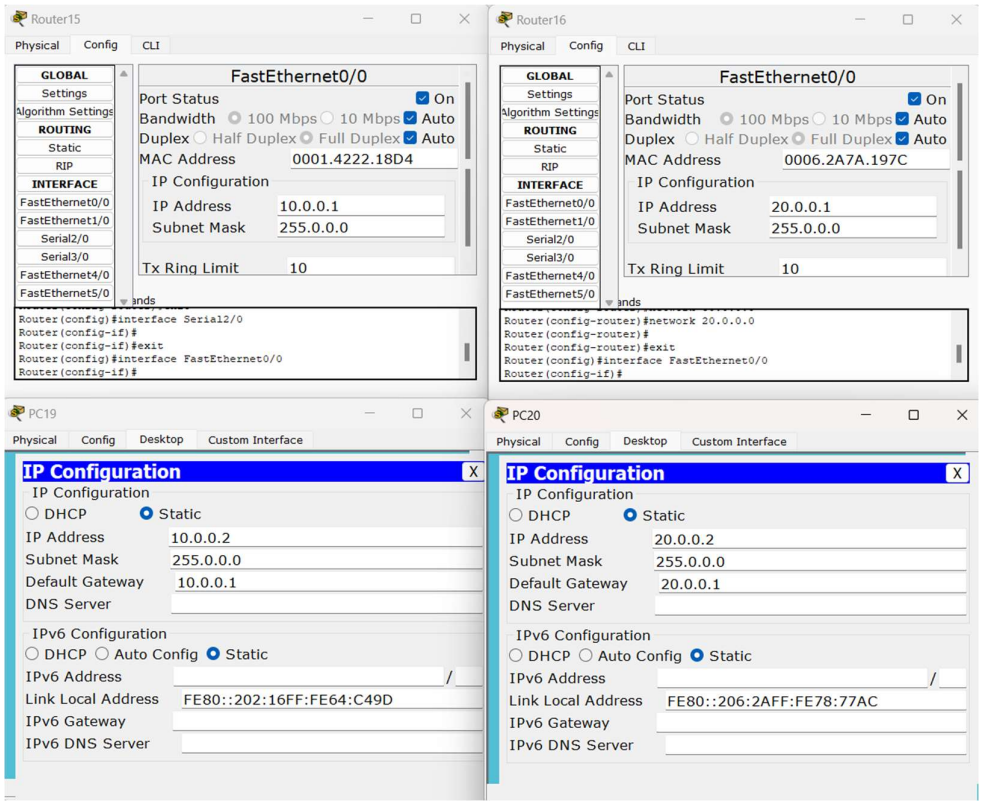


Figure 12: Router and PC IP addresses

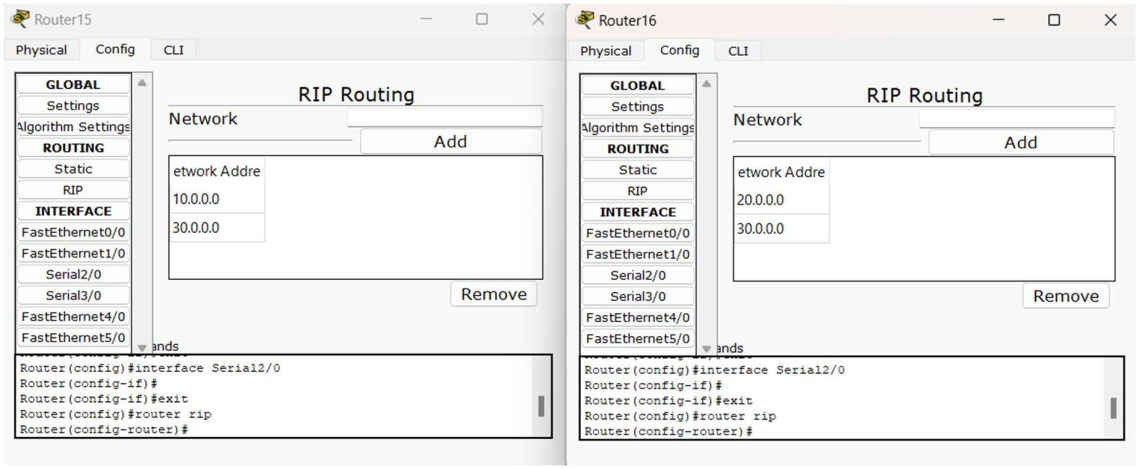


Figure 13: RIP networks

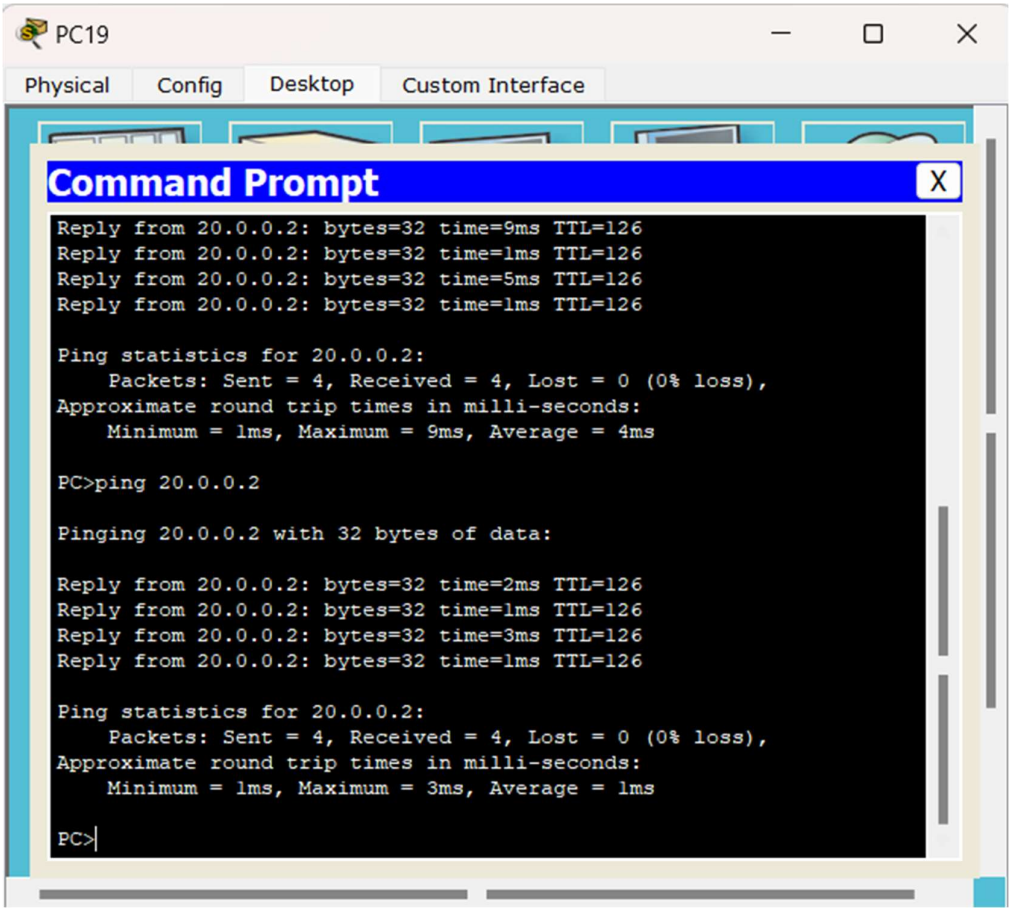


Figure 14: ping command output

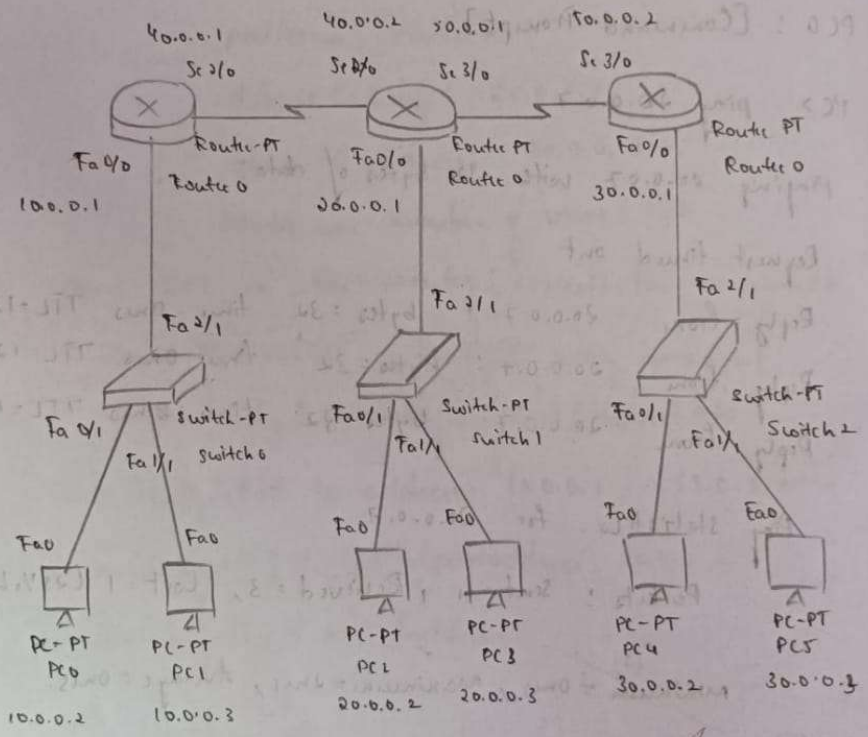
20-11-2024

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 turn 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

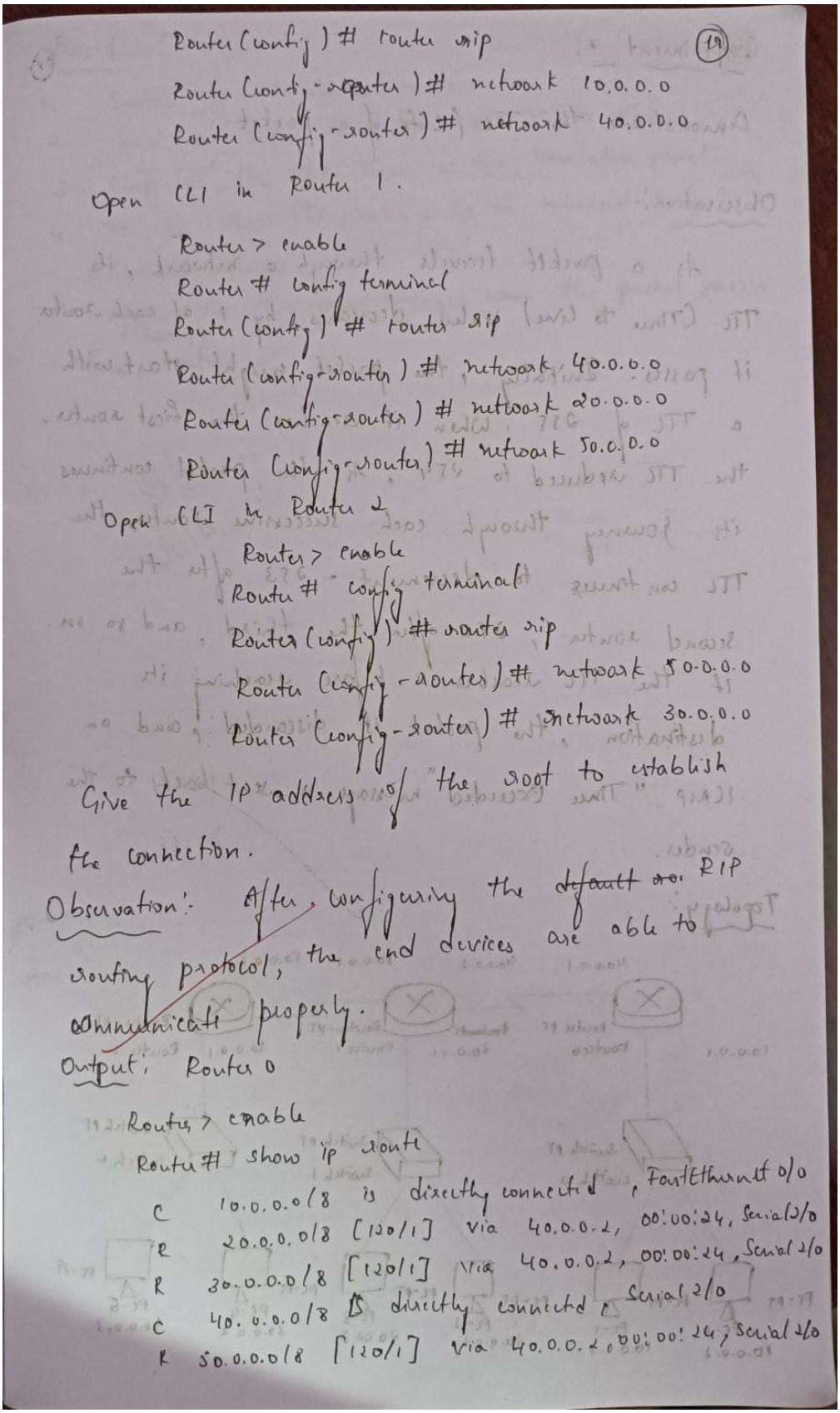
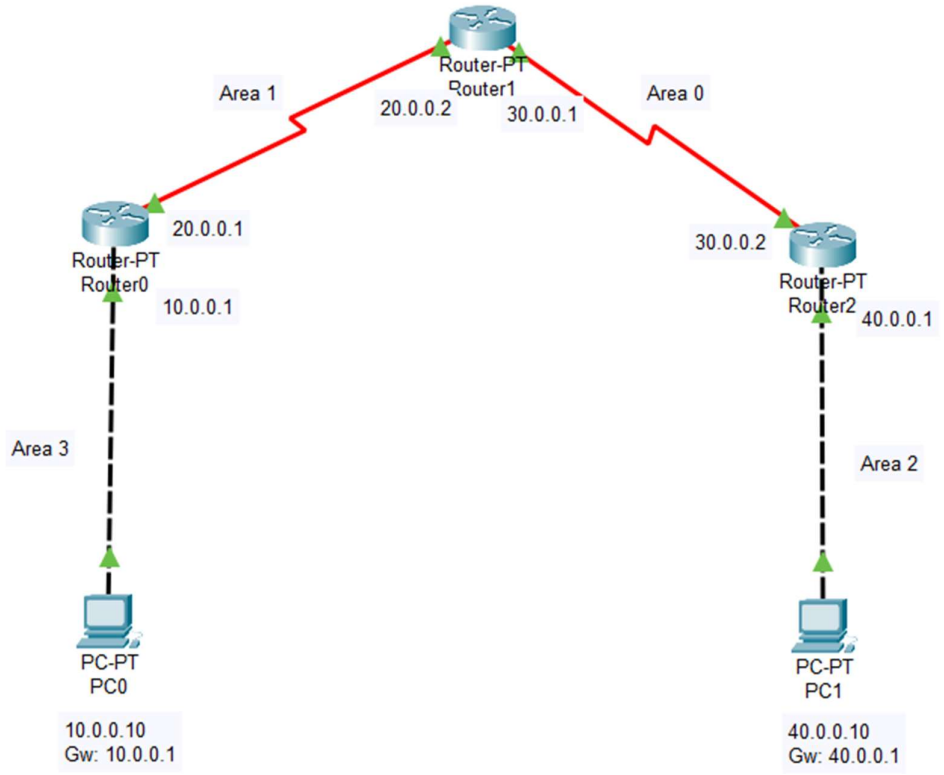
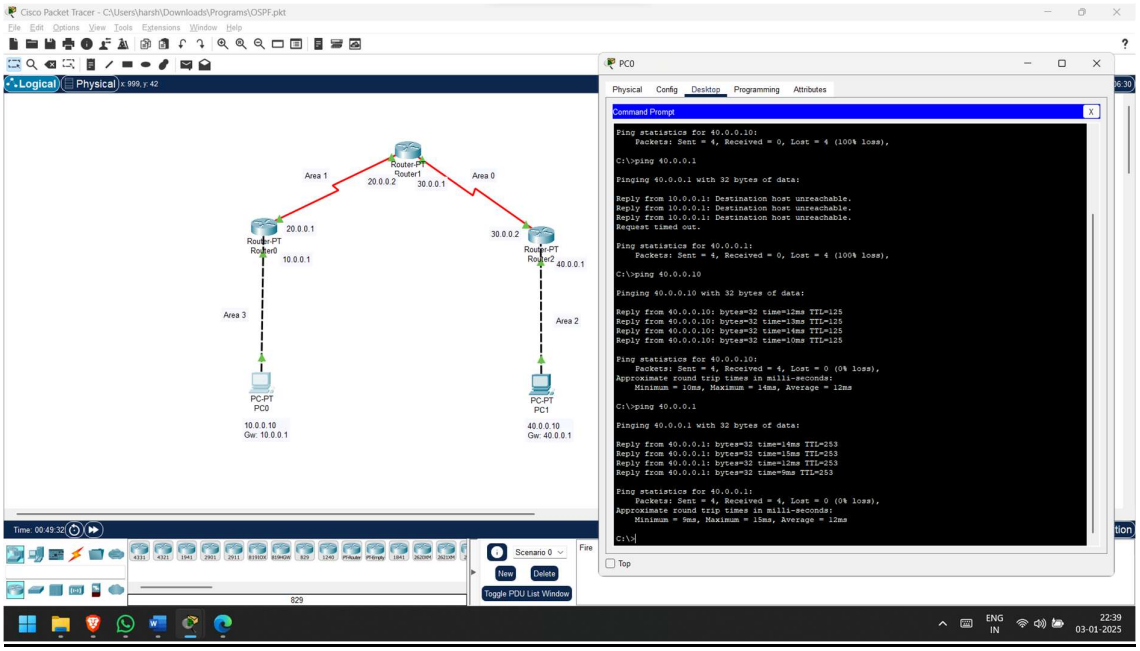


Figure 16: Observation book 2

Program 6

Configure OSPF routing Protocol



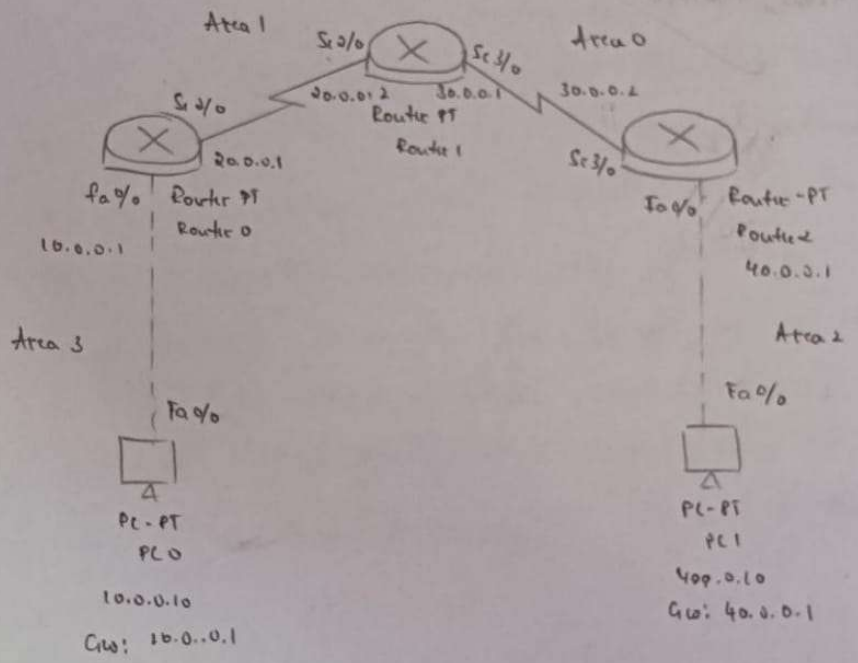


Experiment No 6:

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 2/0
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 2/0
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)
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 O which stands for OSPF connection. where Area 0 exist.

There must be one interface up to keep ospf process up. So its better to configure loopback address to (23) router.

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 code O 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 #1 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 seamlessly.

Output:

PC0:

PC> ping 40.0.0.10

Pinging 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

Packets: Sent=4 Received=3 Lost=1

15/12/24

Program 7

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

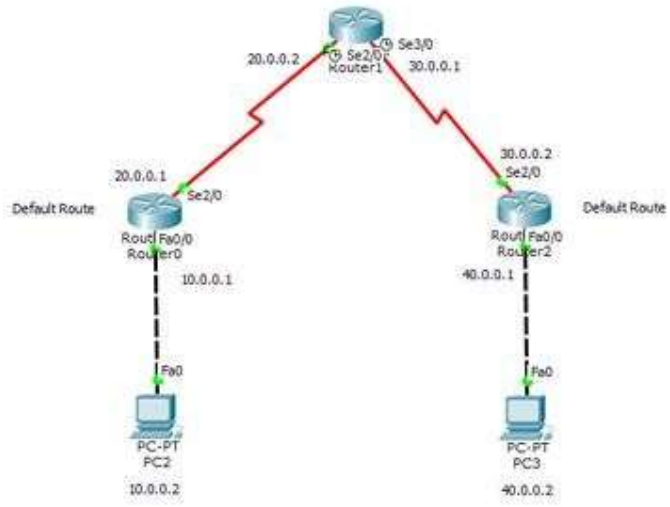


Figure 44: Topology

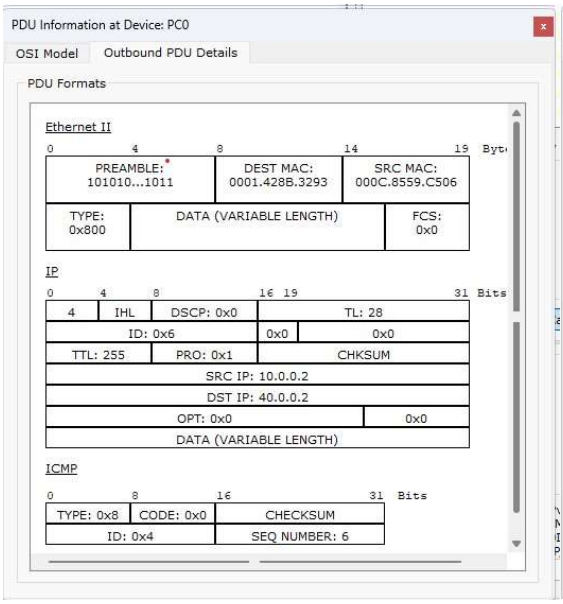


Figure 45: TTL of PC0

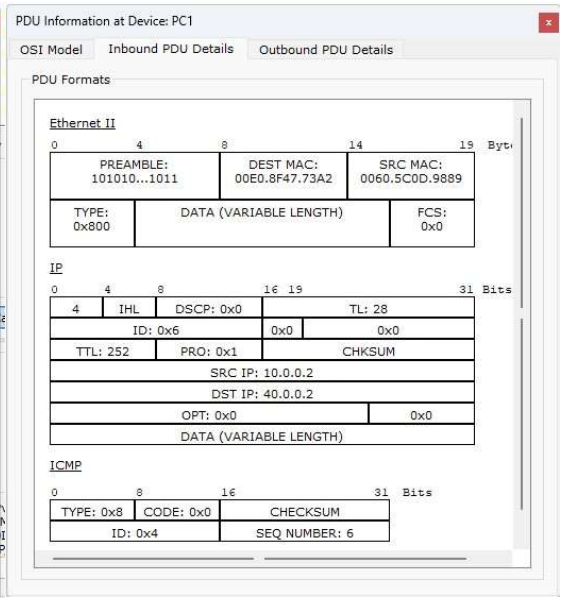


Figure 46: Inbound TTL of PC1

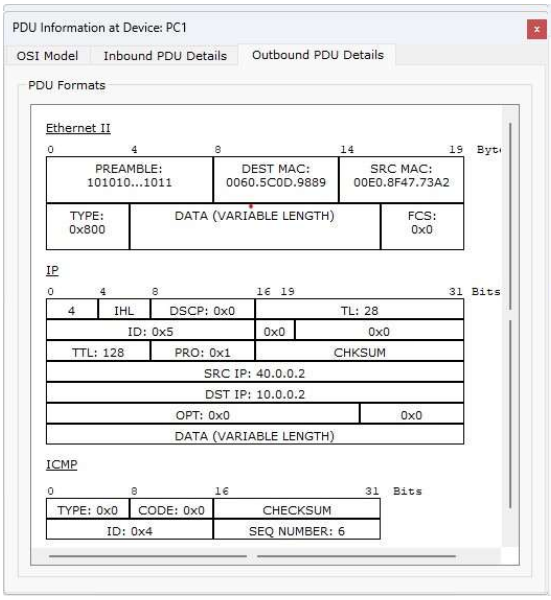


Figure 47: Outbound TTL of PC1

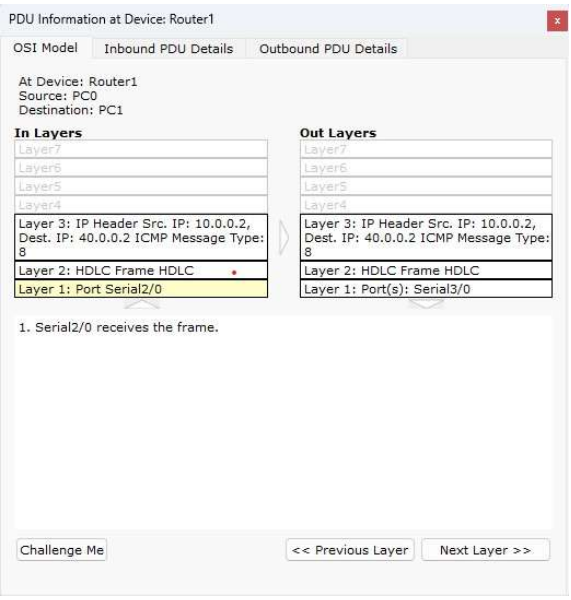


Figure 48: OSI Model of Router

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 reduced to 254. As the packet continues its journey through each successive routers, 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:

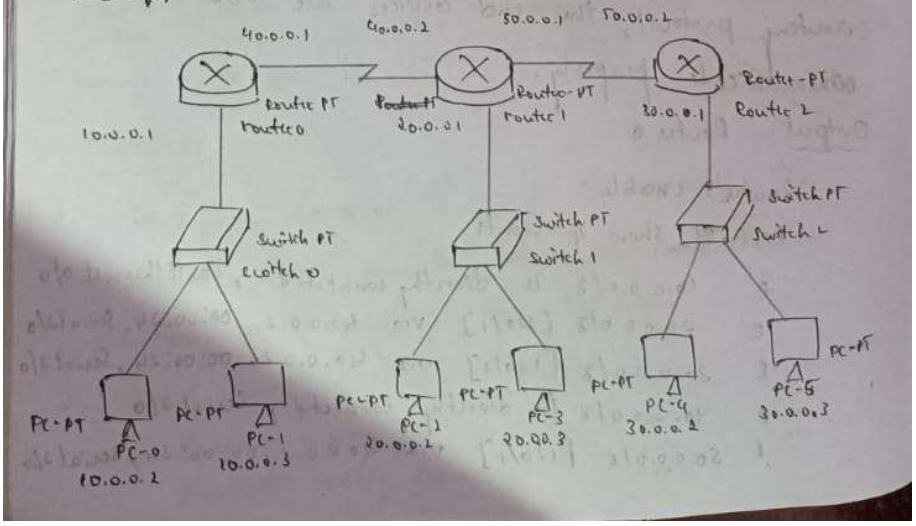
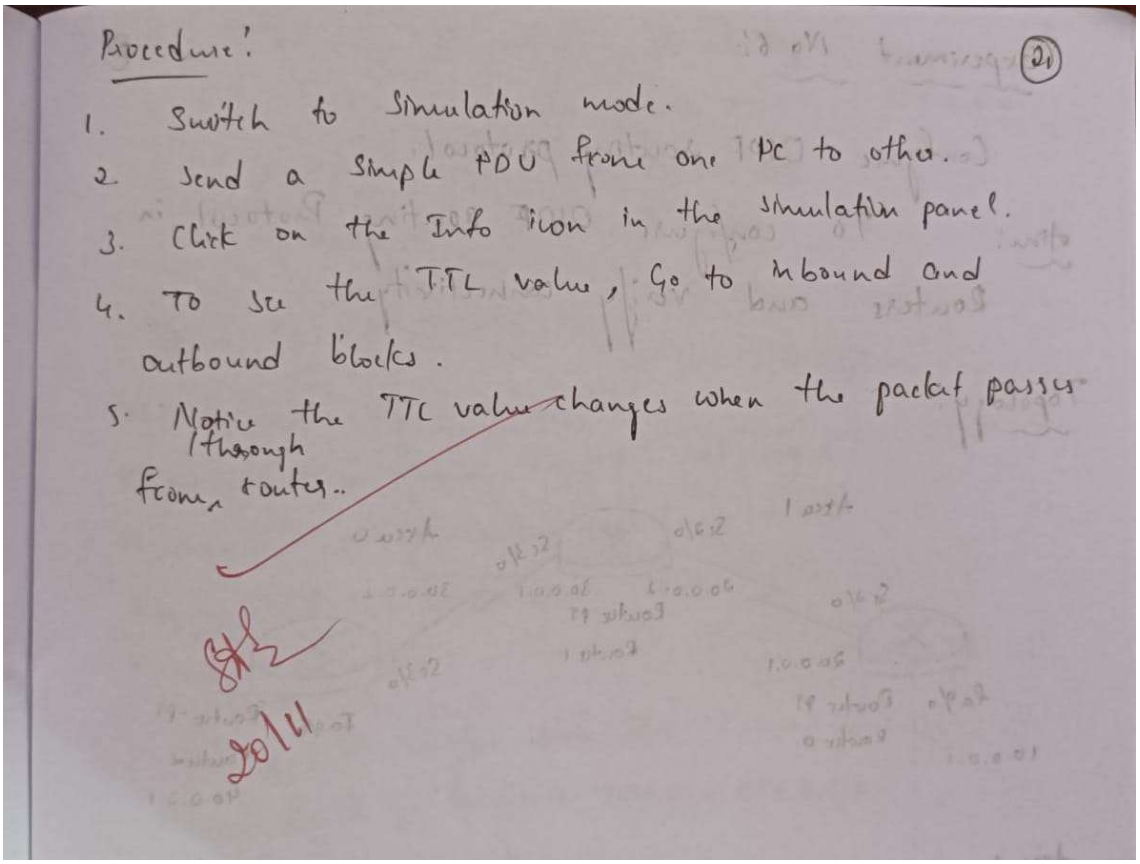


Figure 49: Observation book 1


Procedure:

(21)

1. Switch to Simulation mode.
2. Send a Simple PDU from one PC to other.
3. Click on the Info icon in the simulation panel.
4. To see the TTL value, Go to Inbound and outbound blocks.
5. Notice the TTL value changes when the packet passes through from routers.

20/11

Figure 50: Observation book 2

Program 8

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

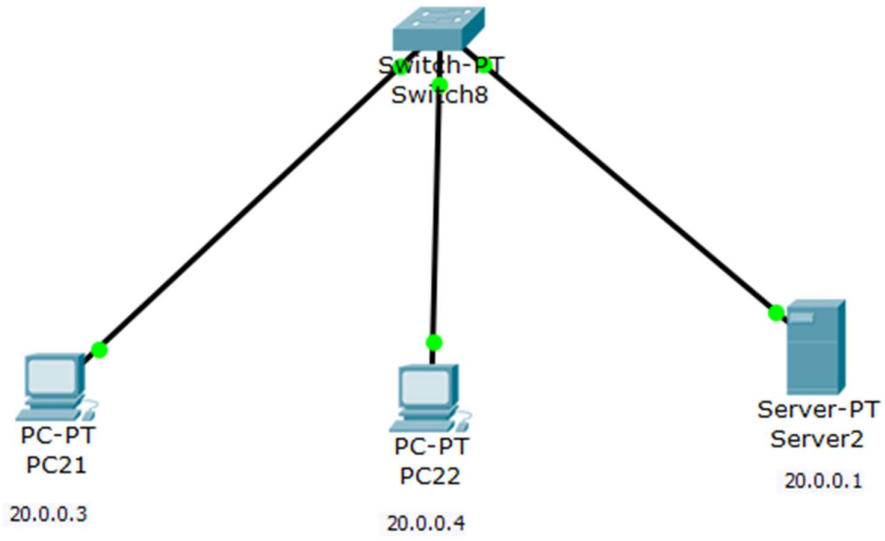


Figure 51: Topology

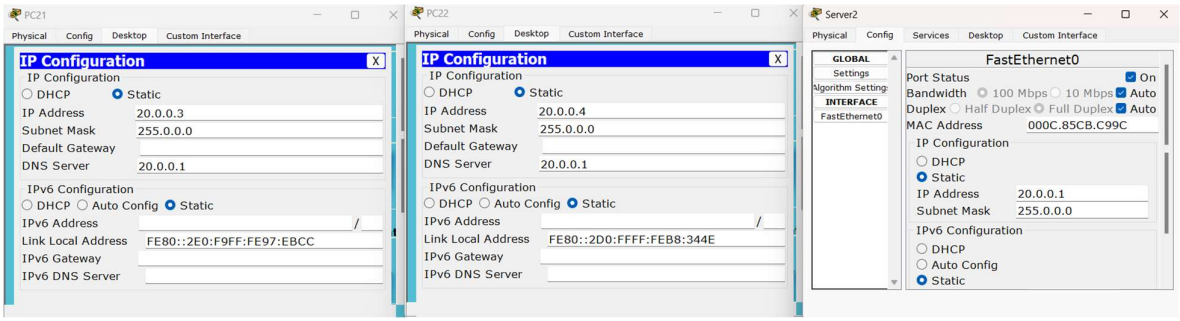


Figure 52: IP Addresses

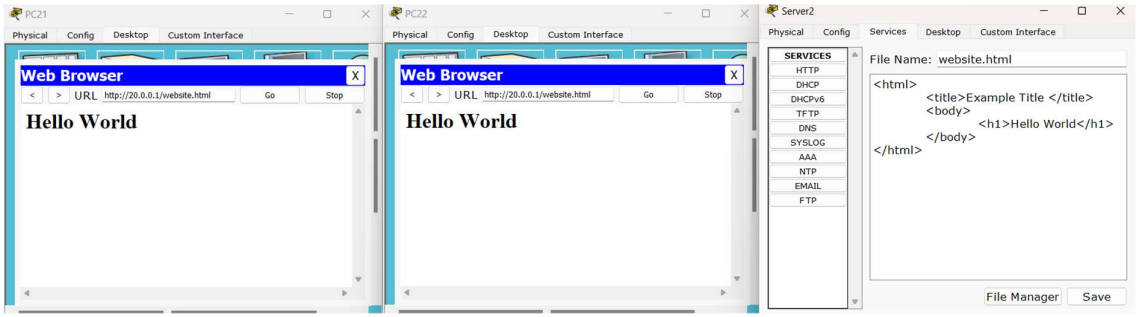


Figure 53: Server and the website

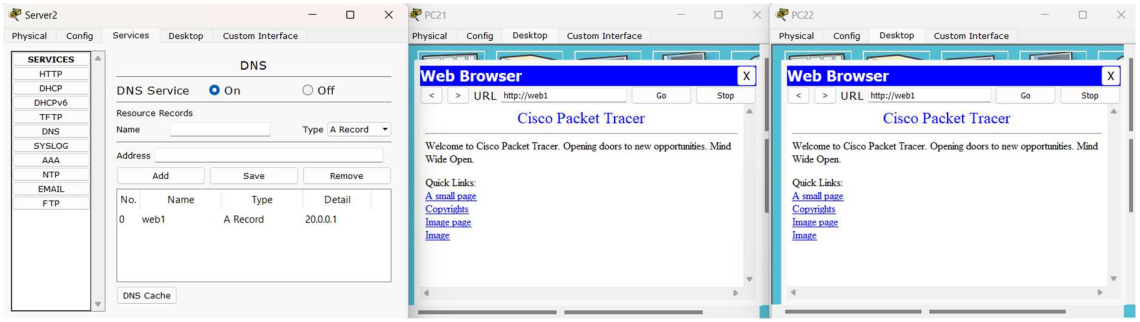


Figure 54: DNS and domain names

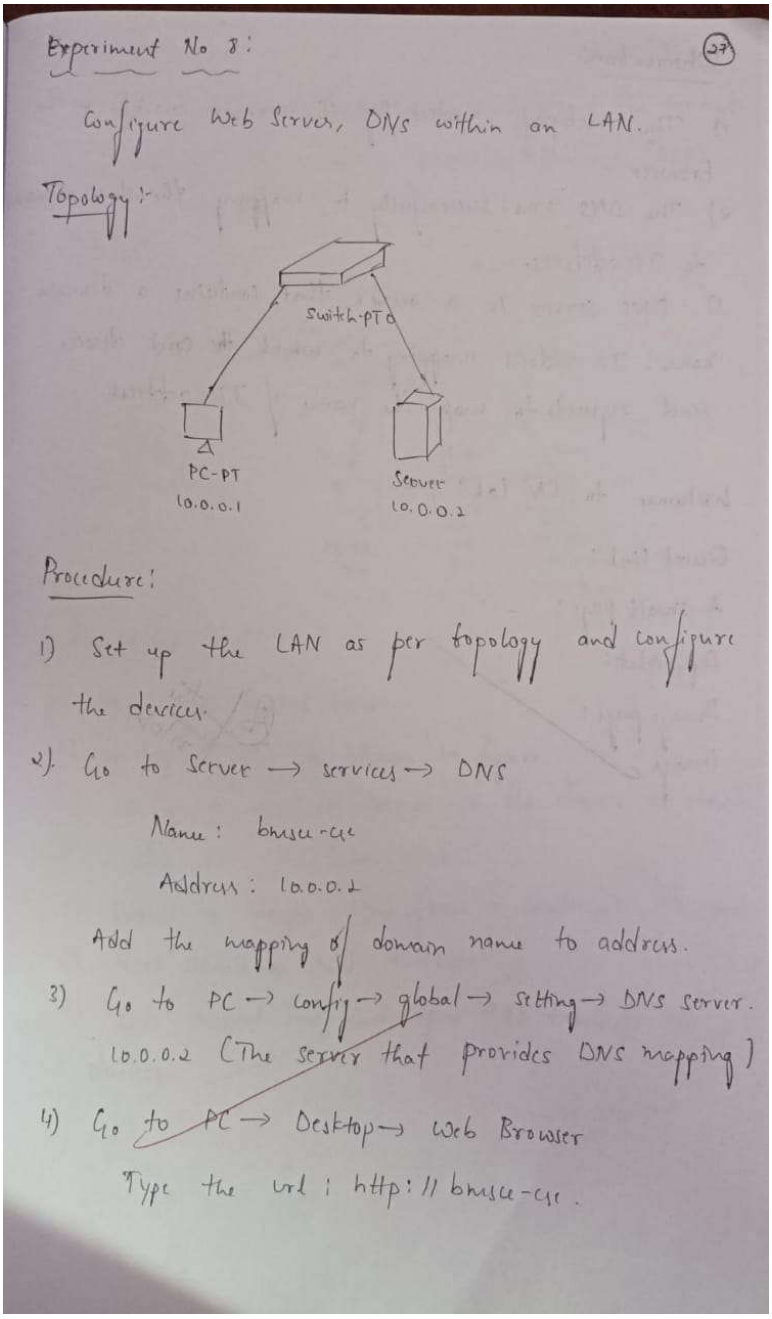
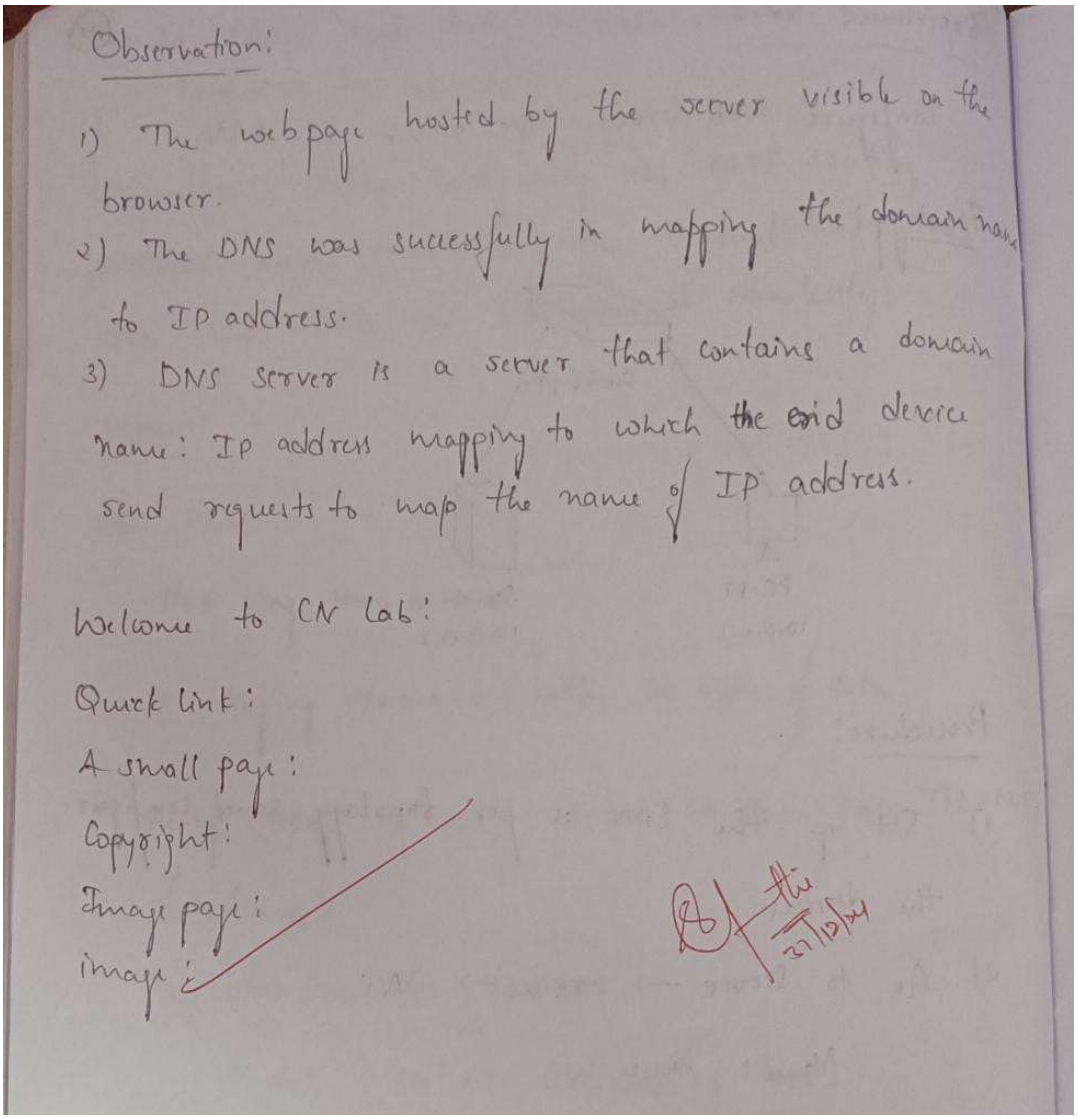


Figure 55: Observation Book 1


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:

Figure 56: Observation Book 2

Program 9

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

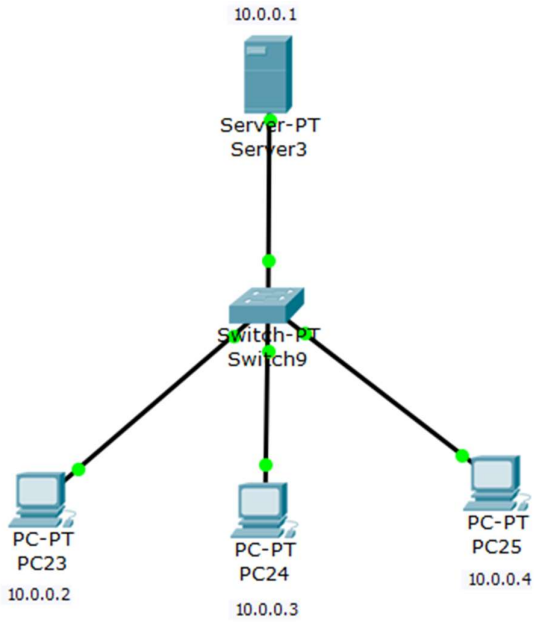


Figure 57: Topology

ARP Table for Server3

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

ARP Table for PC23

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

ARP Table for PC24

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

ARP Table for PC25

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

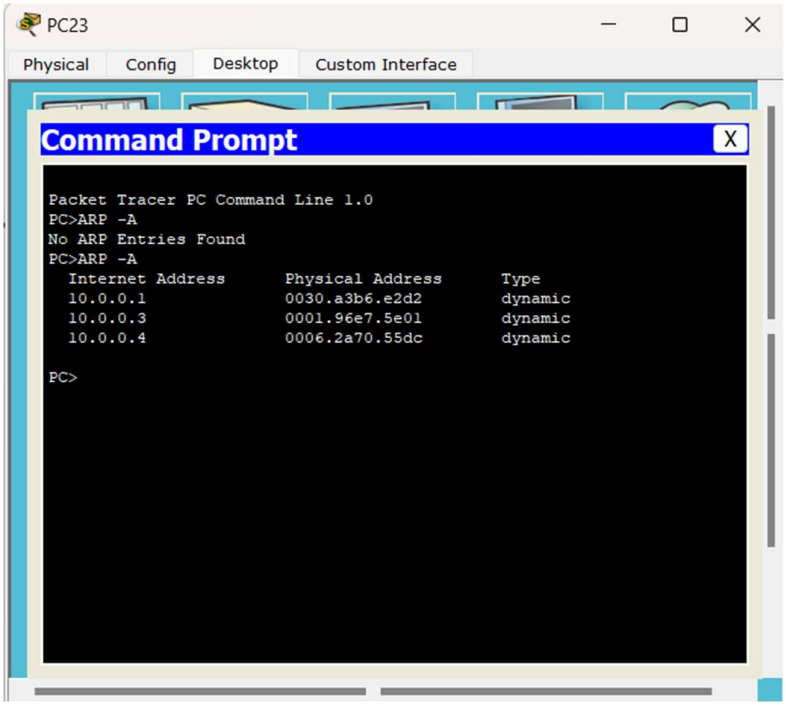


Figure 59: ARP command output of PC0

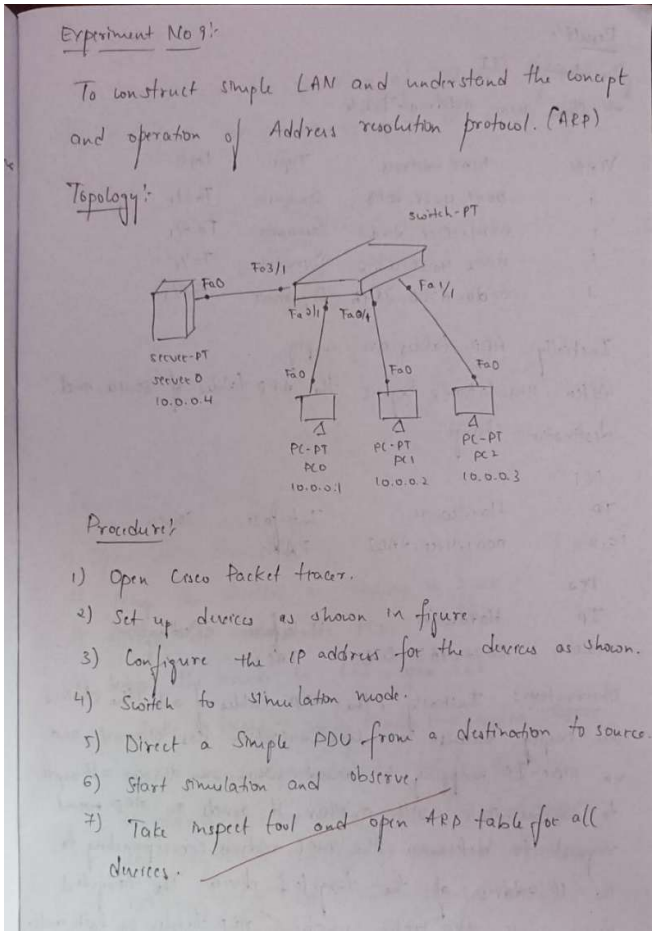
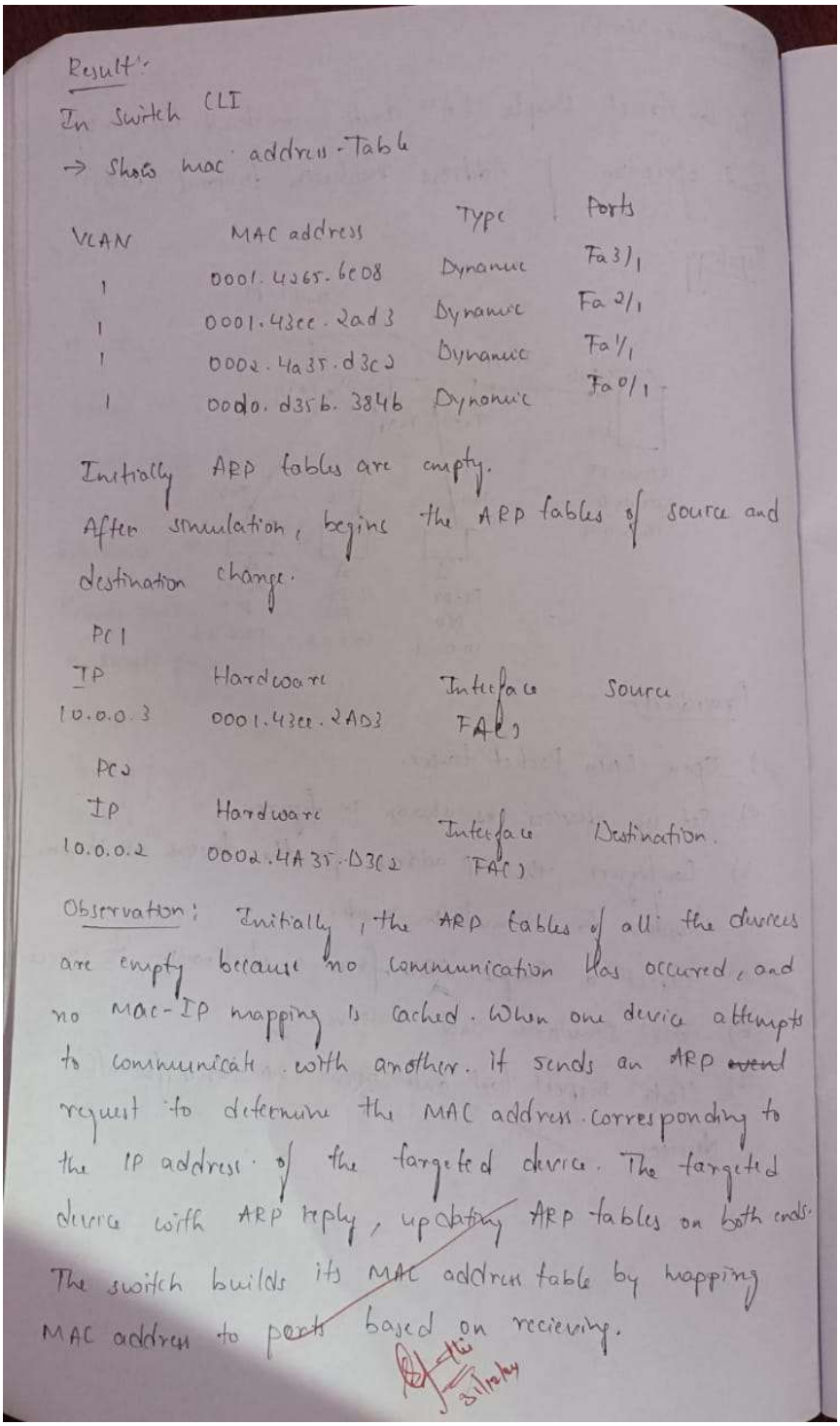


Figure 60: Observation Book 1



Result:

In Switch CLI

→ Shows mac address Table

VLAN	MAC address	Type	Ports
1	0001.4365.6e08	Dynamic	Fa3/1
1	0001.43ee.2ad3	Dynamic	Fa2/1
1	0002.4a35.d3c2	Dynamic	Fa1/1
1	00d0.d356.384b	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	FA0/1	

PC2

IP	Hardware	Interface	Destination.
10.0.0.2	0002.4A35.D3C2	FA0/1	

Observation: Initially, the ARP tables of all the devices are empty because no communication has occured, and no MAC-IP mapping is cached. When one device attempts to communicate with another. it sends an ARP ~~event~~ 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 port based on recieving.

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.

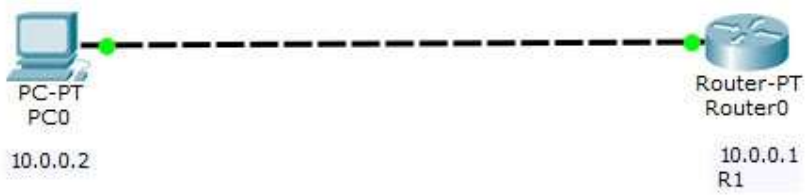


Figure 62: Topology

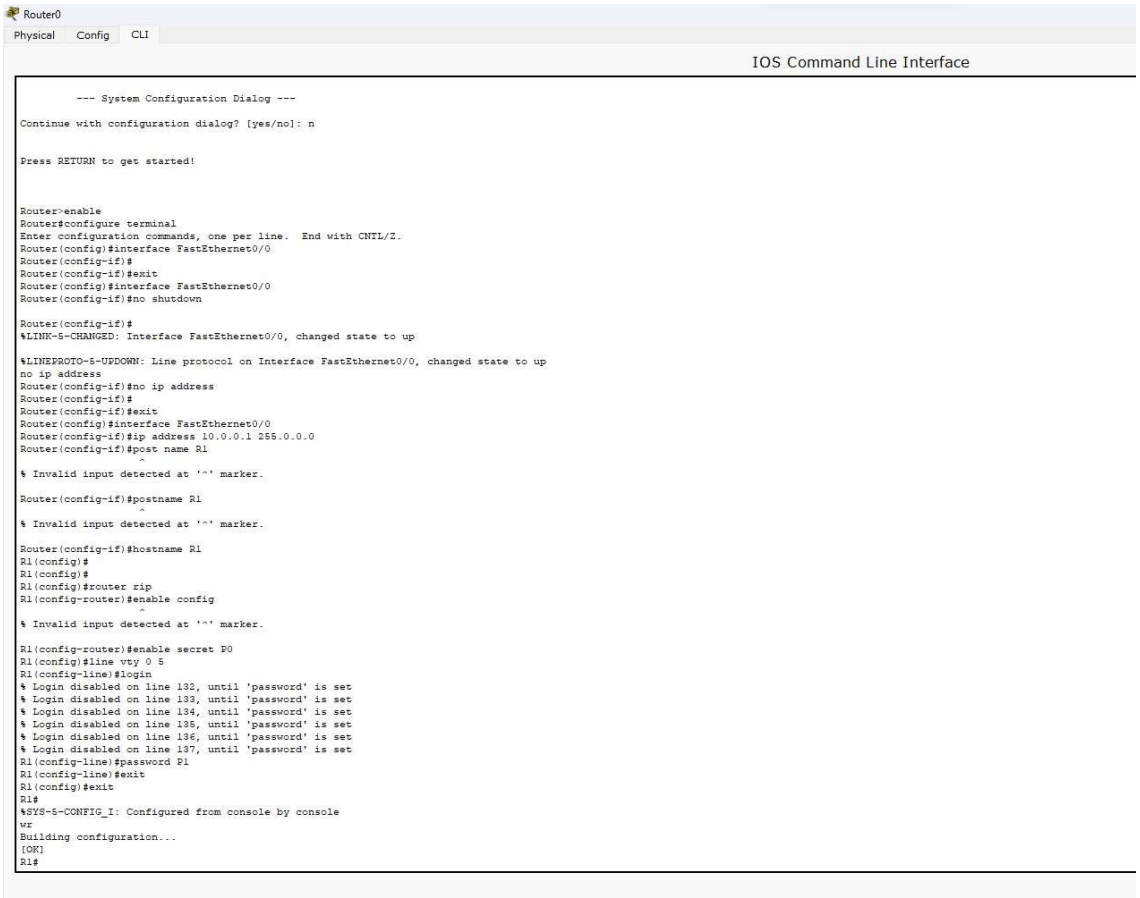


Figure 63: Router CLI

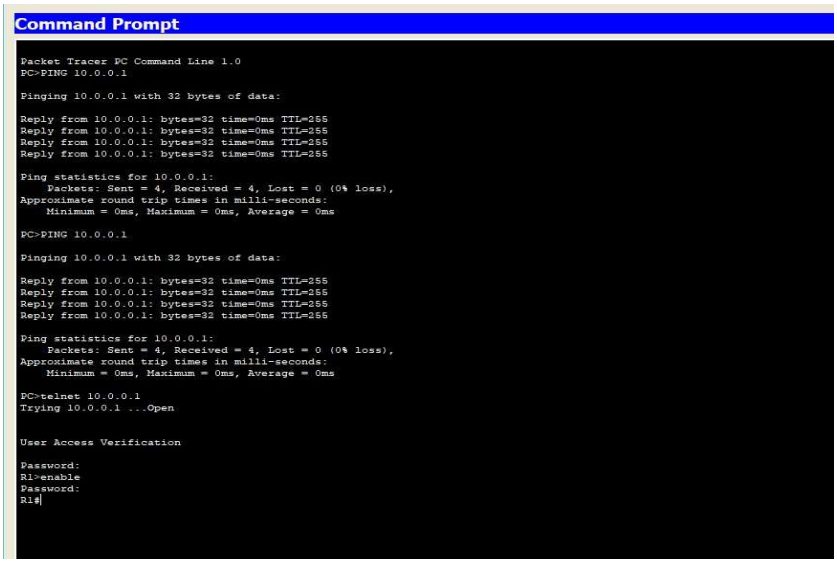


Figure 64: Output

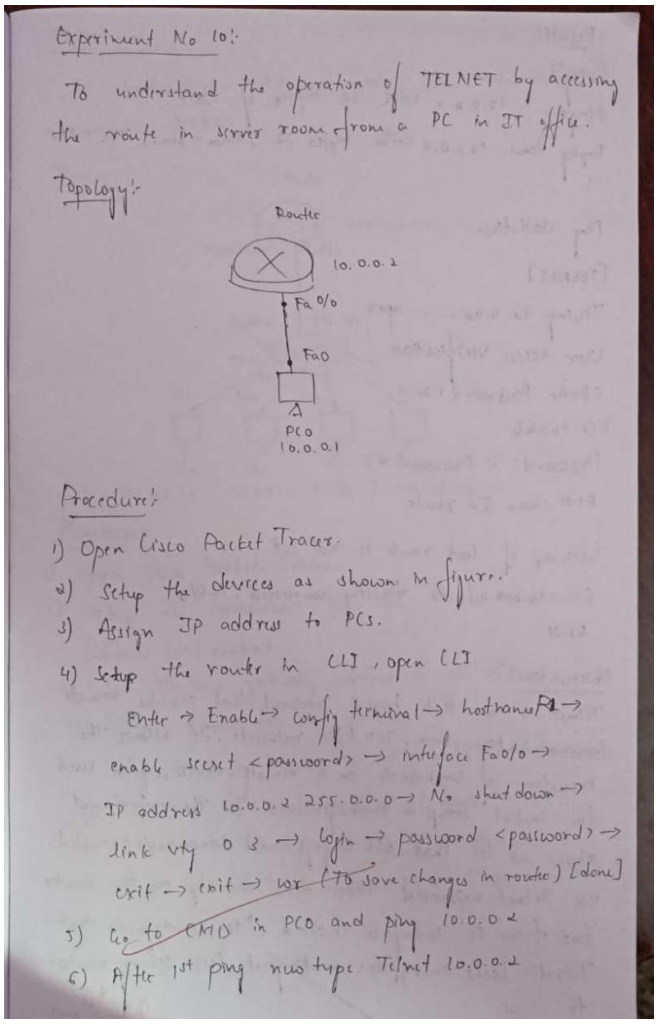
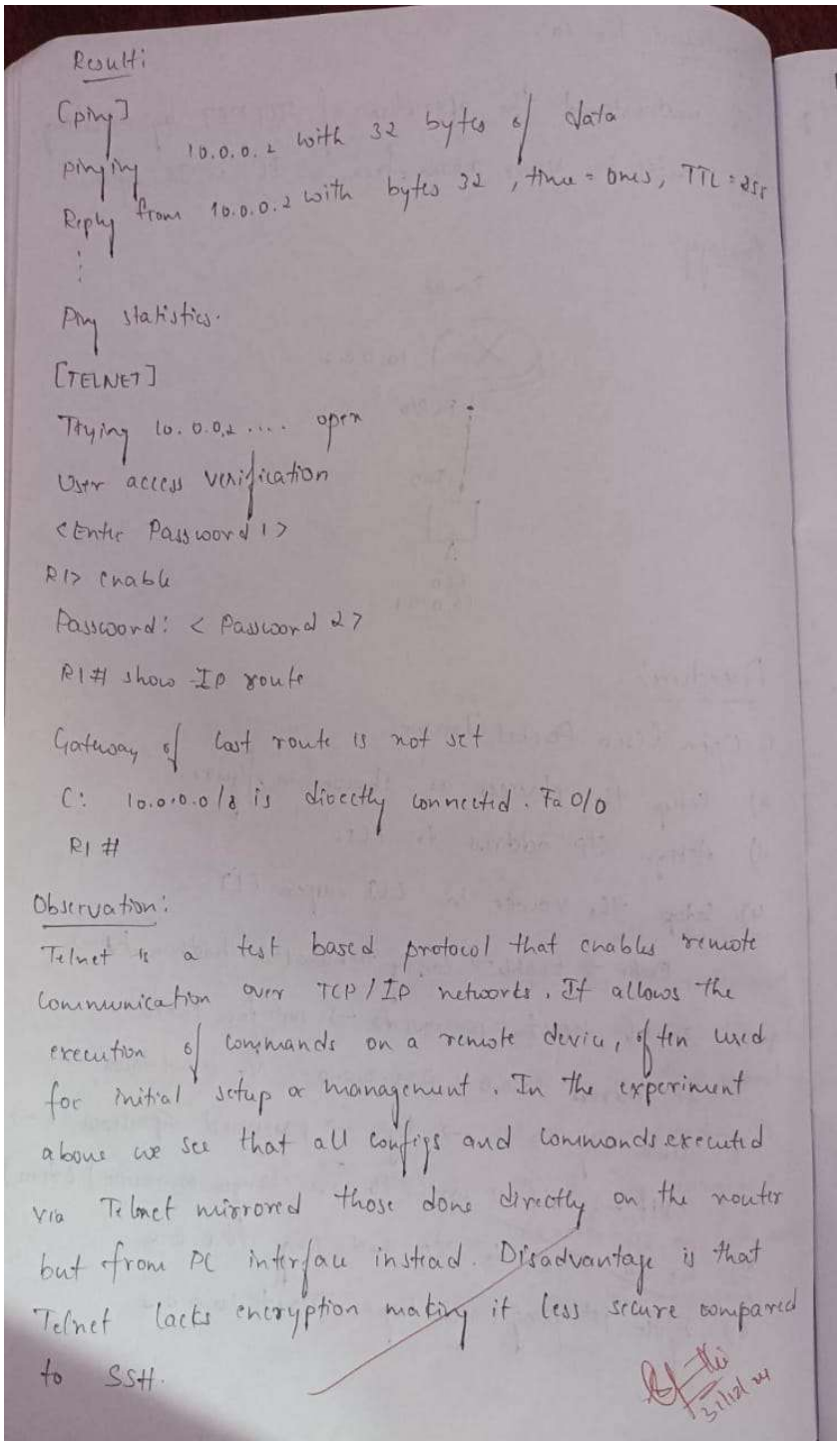


Figure 65: Observation Book 1


Result:

[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 2>

R1# show IP route

Gateway of last route is not set

C: 10.0.0.0/8 is directly connected. Fa 0/0

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.

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.

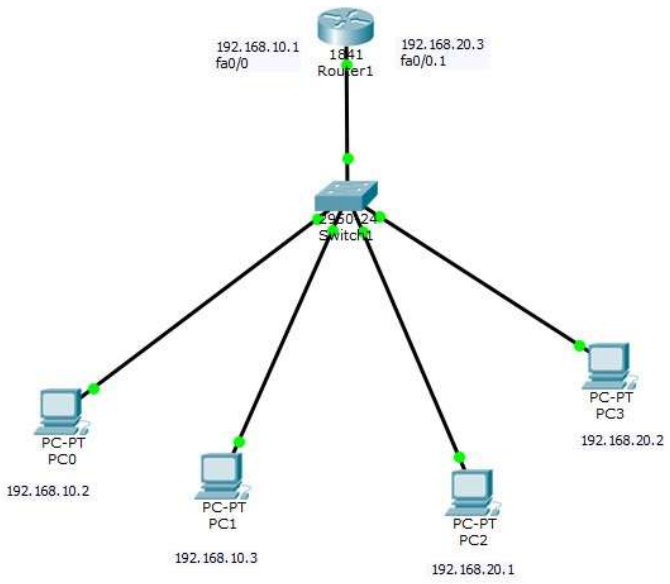


Figure 67: Topology

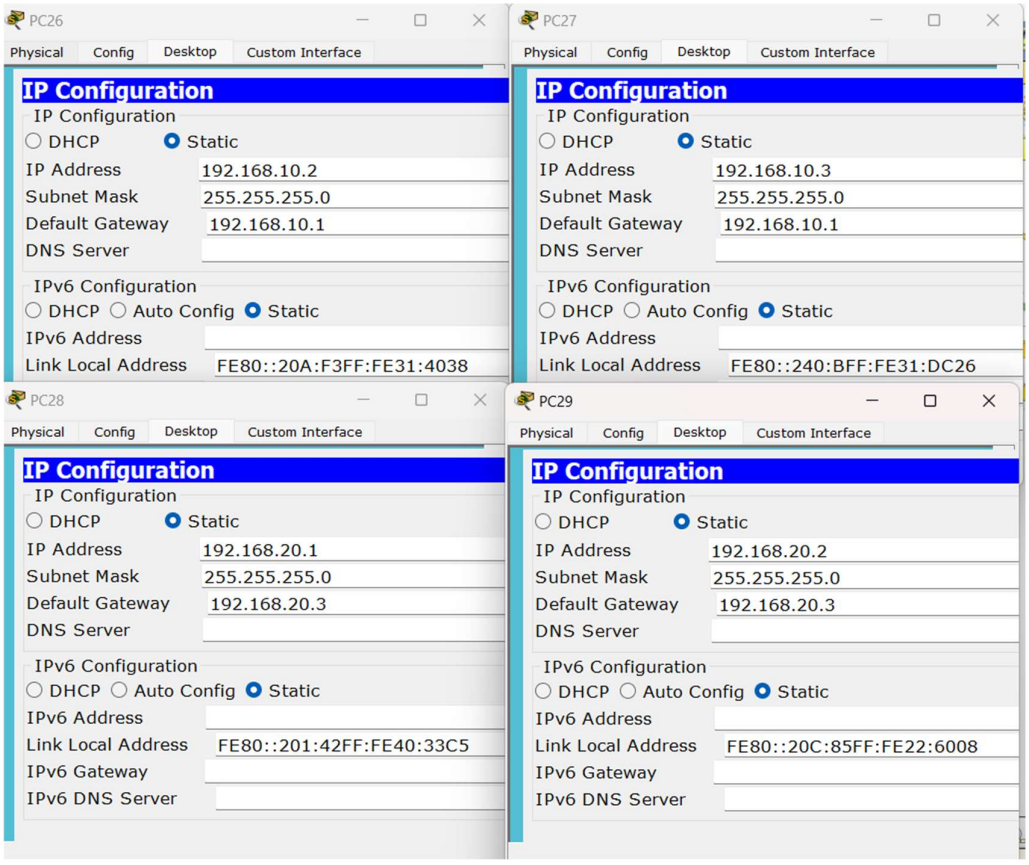


Figure 68: IP Addresses of the PCs

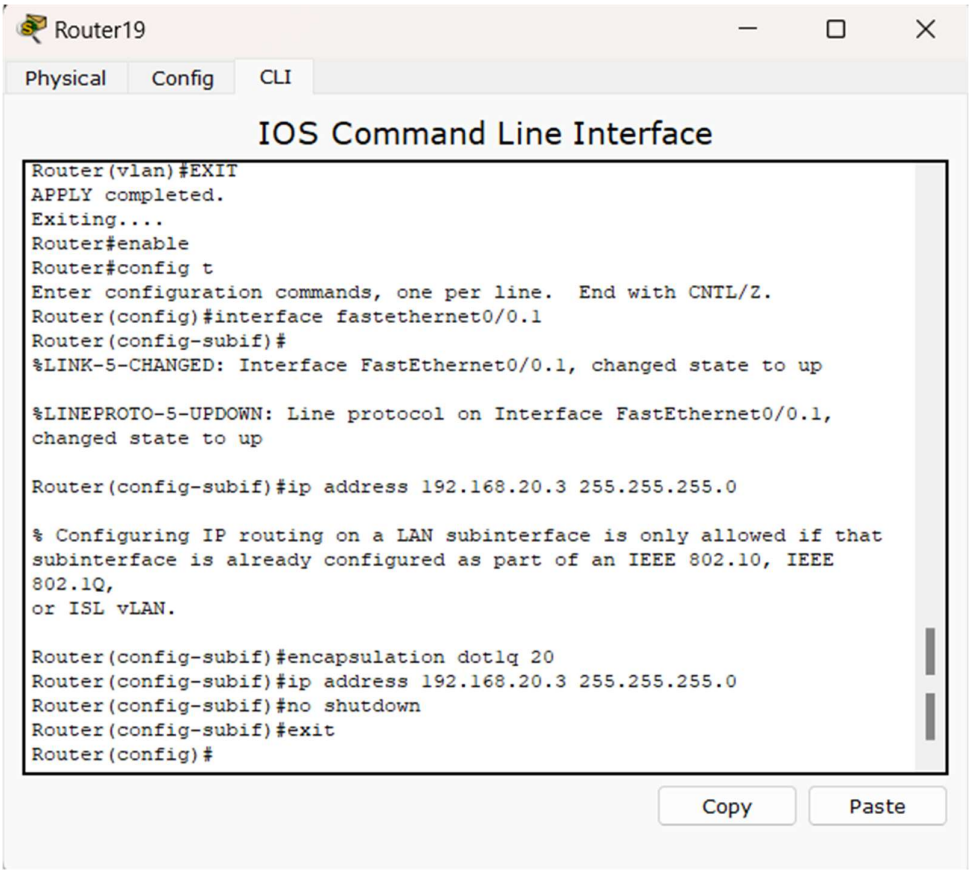


Figure 69: Router CLI

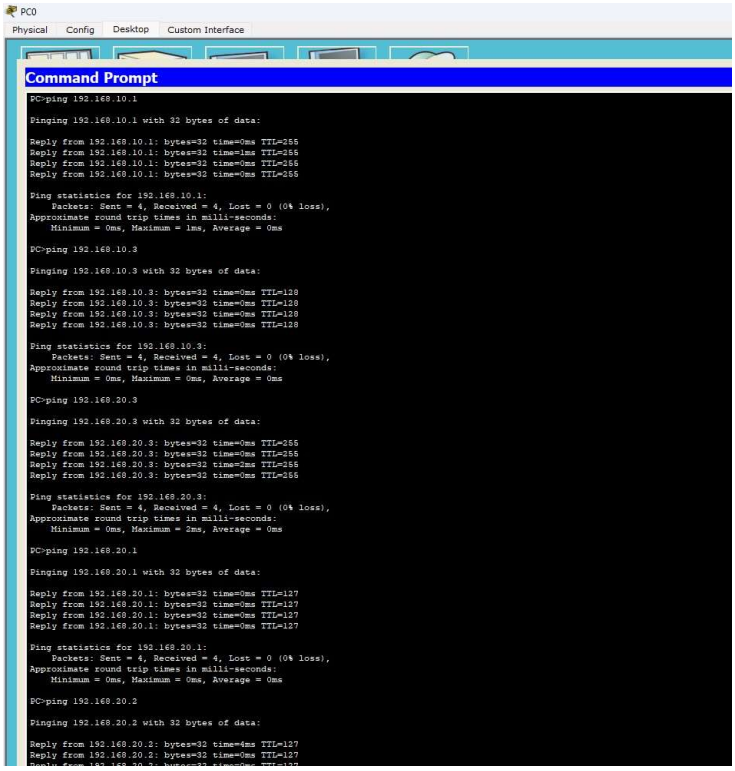
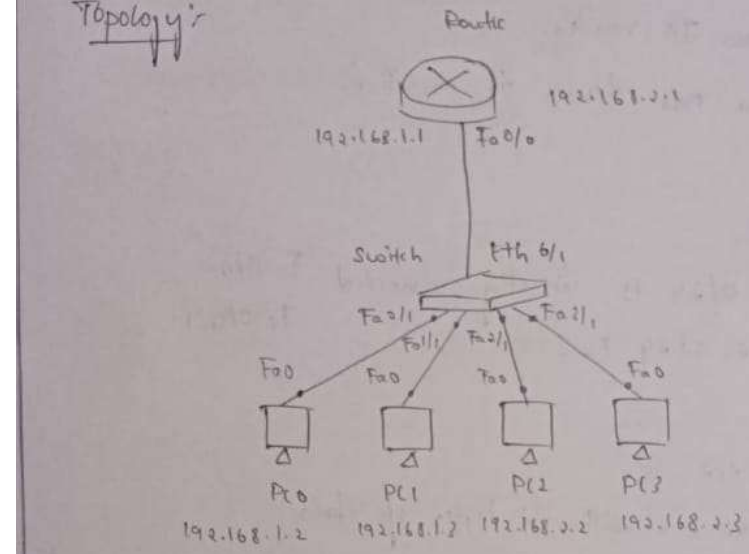


Figure 70: Output

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 Fa 0/0 → 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 bms) and add.
6) Go to interface ethernet 6/1.

Figure 71: Observation Book 1

7) VLAN database → VLAN 2, name bons → exit.

8) config terminal → interface Fa 0/0.1 → encapsulation dot1q
2 → ip address 192.168.2.1 255.255.255.0 → no shut
exit → exit

9) Enter show ip route.

10) Ping from one device to another.

Result:

show ip route
C. 192.168.1.0/24 is directly connected Fa 0/0
C. 192.168.2.0/24 Fa 0/0.1

PC0
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=2ms TTL=127
:
ping stats.

Observation:

The VLAN experiment involves creating and configuring VLAN to segment a network, assigning IPs to device 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 network effectively.

Thio 31/10/25

Figure 72: Observation Book 2

Program 12

To demonstrate communication between two devices using a witeless LAN.

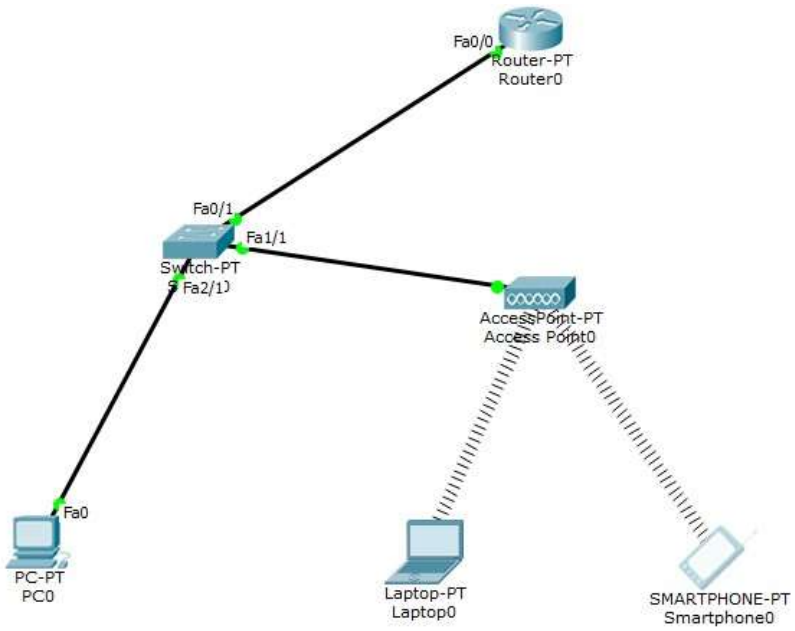


Figure 73: Topology

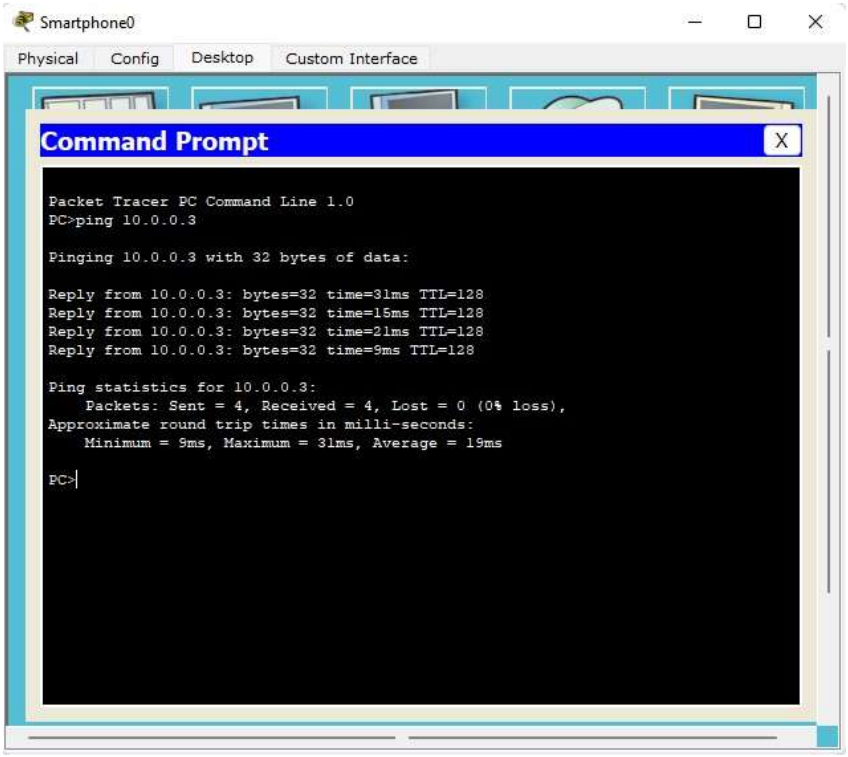


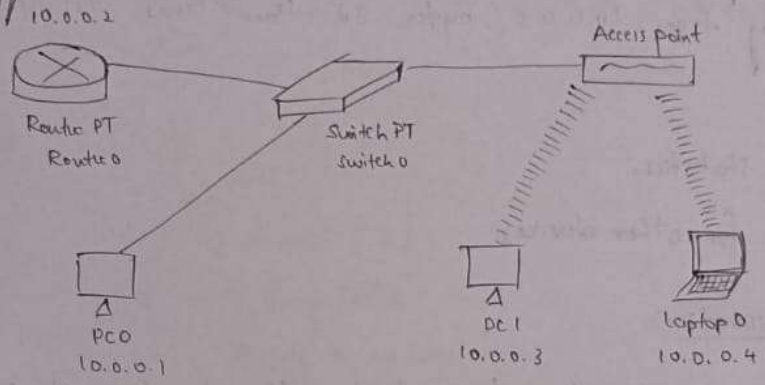
Figure 74: Output

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 digits (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-IAH 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 1

7) Ping from either devices

8) Setup PC0, router as normally done.

Result:

PC0:

C:\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.

Same for other devices.

Observation:

The experiment dimonstrates 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 tests between devices verify the setup; highlighting the simplicity and efficiency of WLAN connections for wireless communications.

21/12/24

Figure 76: Observation book 2

Program 13

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

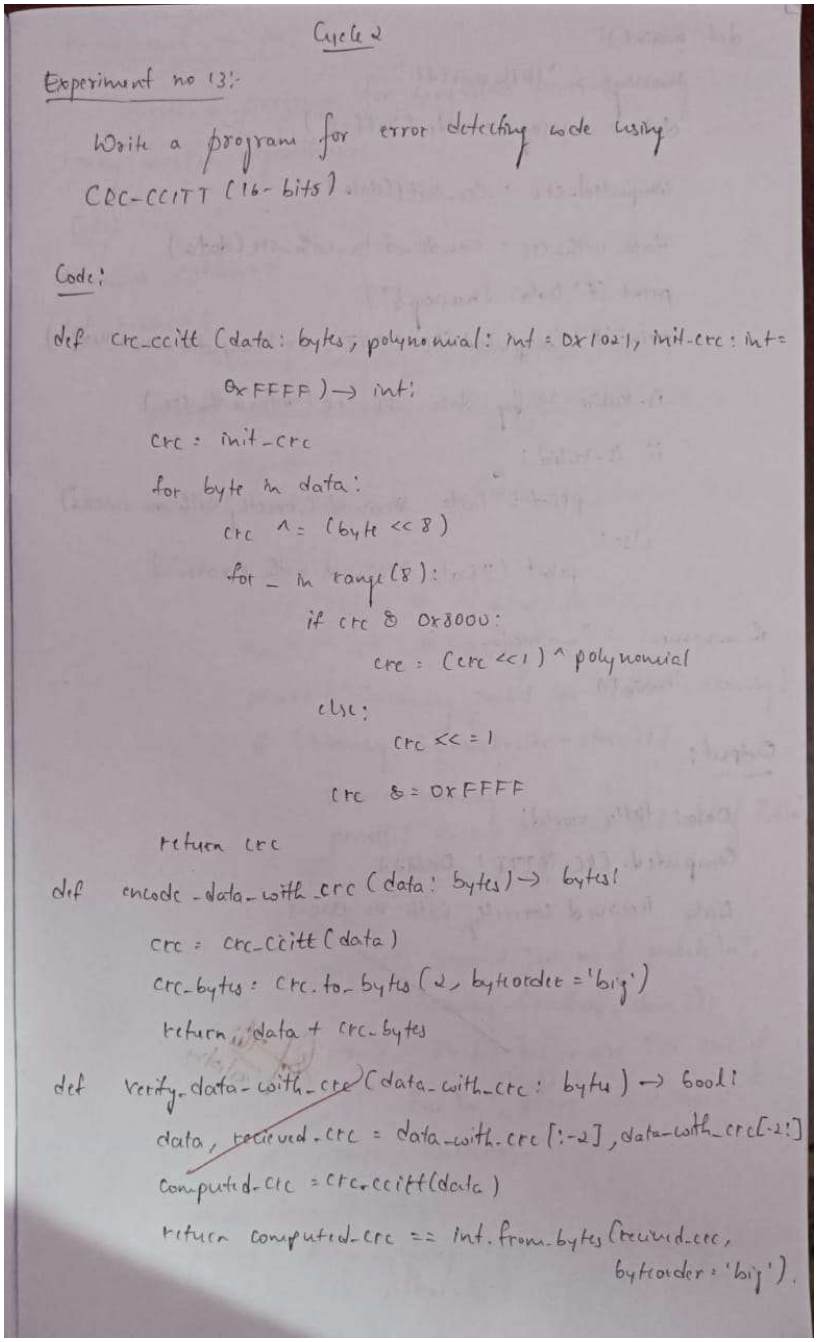
Code:

```
#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



Cycle 2

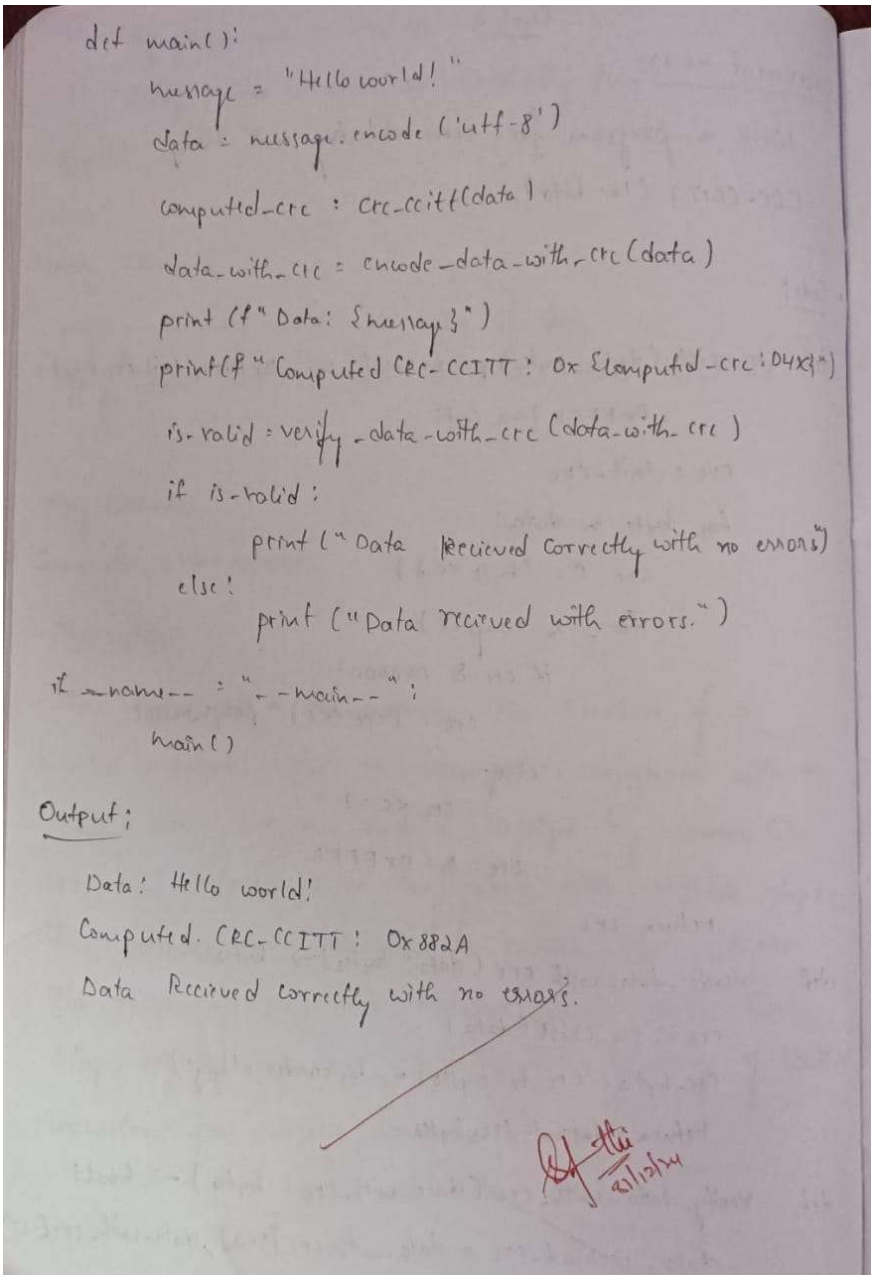
Experiment no 13:-

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

Code:

```
def crc_ccitt(data: bytes, polynomial: int = 0x1021, init_crc: int=
        0xFFFF) -> int:
    crc = init_crc
    for byte in data:
        crc ^= (byte << 8)
        for _ in range(8):
            if crc & 0x8000:
                crc = (crc << 1) ^ polynomial
            else:
                crc <<= 1
            crc &= 0xFFFF
    return crc
def encode_data_with_crc(data: bytes) -> bytes:
    crc = crc_ccitt(data)
    crc_bytes = crc.to_bytes(2, byteorder='big')
    return data + crc_bytes
def verify_data_with_crc(data_with_crc: bytes) -> bool:
    data, received_crc = data_with_crc[:-2], data_with_crc[-2:]
    computed_crc = crc_ccitt(data)
    return computed_crc == int.from_bytes(received_crc,
                                          byteorder='big').
```

Figure 78: Observation Book 1



```
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.
```

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

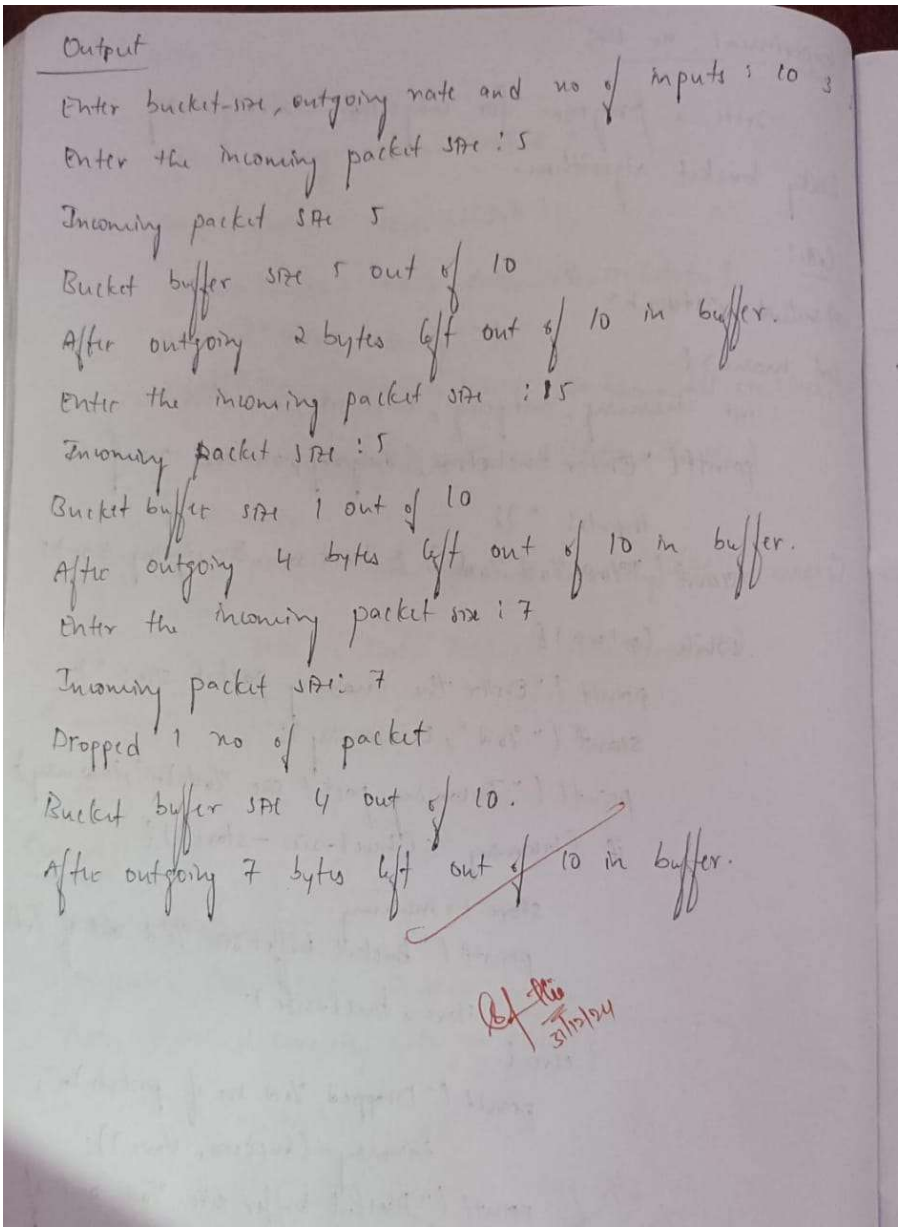
Experiment - no 14;

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
        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--; }}
```

Figure 81: Observation Book 1


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 : 5

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.

31/12/24

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

Experiment no 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.

Code:

- 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')
    data = file.read(1024)
    connectionSocket.send(data.encode())
file.close()
connectionSocket.close()
```

Figure 85: Observation Book

Output:

Server.py - The server ready to recieve.

client.py - Enter file name: example.txt

From Server: Hello, world!

31/12/24

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 recieve")
while 1:
    sentence, clientAddr = serverSocket.recvfrom(2048)
    file = open(sentence, 'r')
    data = file.read(2048)
    serverSocket.sendto(bytes(data, 'utf-8'), clientAddr)
    print("sent back to client:", data)
file.close()
```

Figure 89: Observation Book

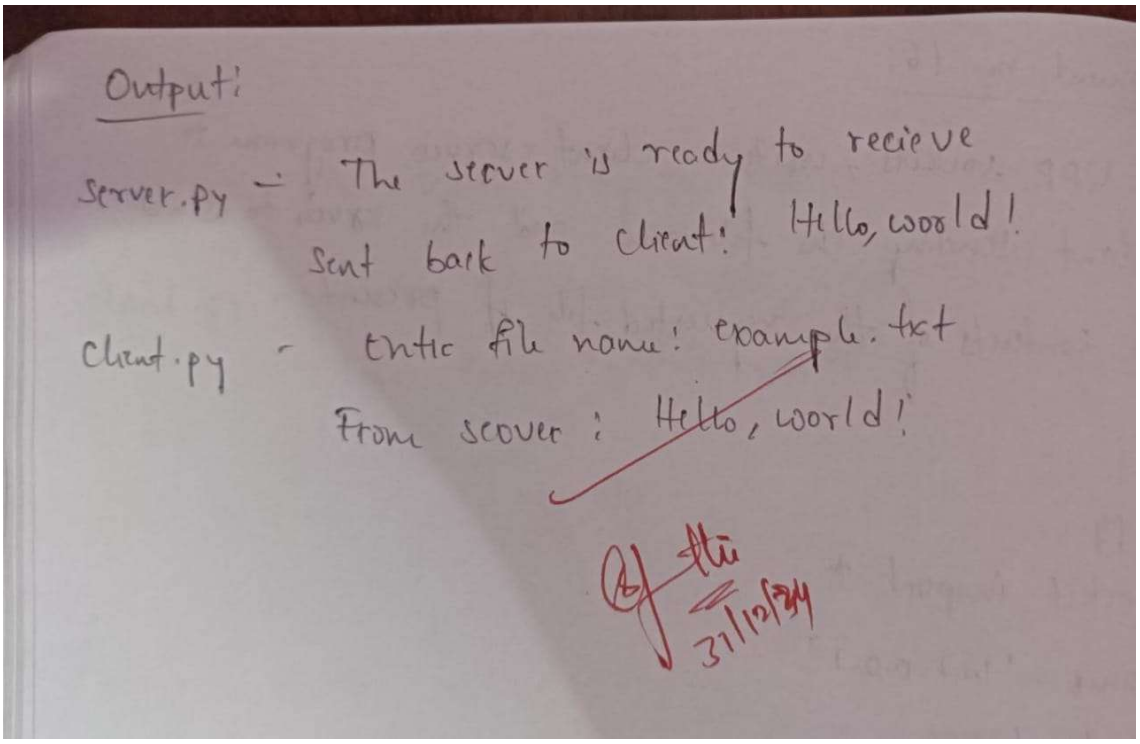


Figure 90: Observation Book