VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

Arnav Dinesh (1BM23CS052)

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 **Arnav Dinesh (1BM23CS052),** 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.

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

Index

GitHub Link:
https://github.com/ArnavRD/CN-LAB

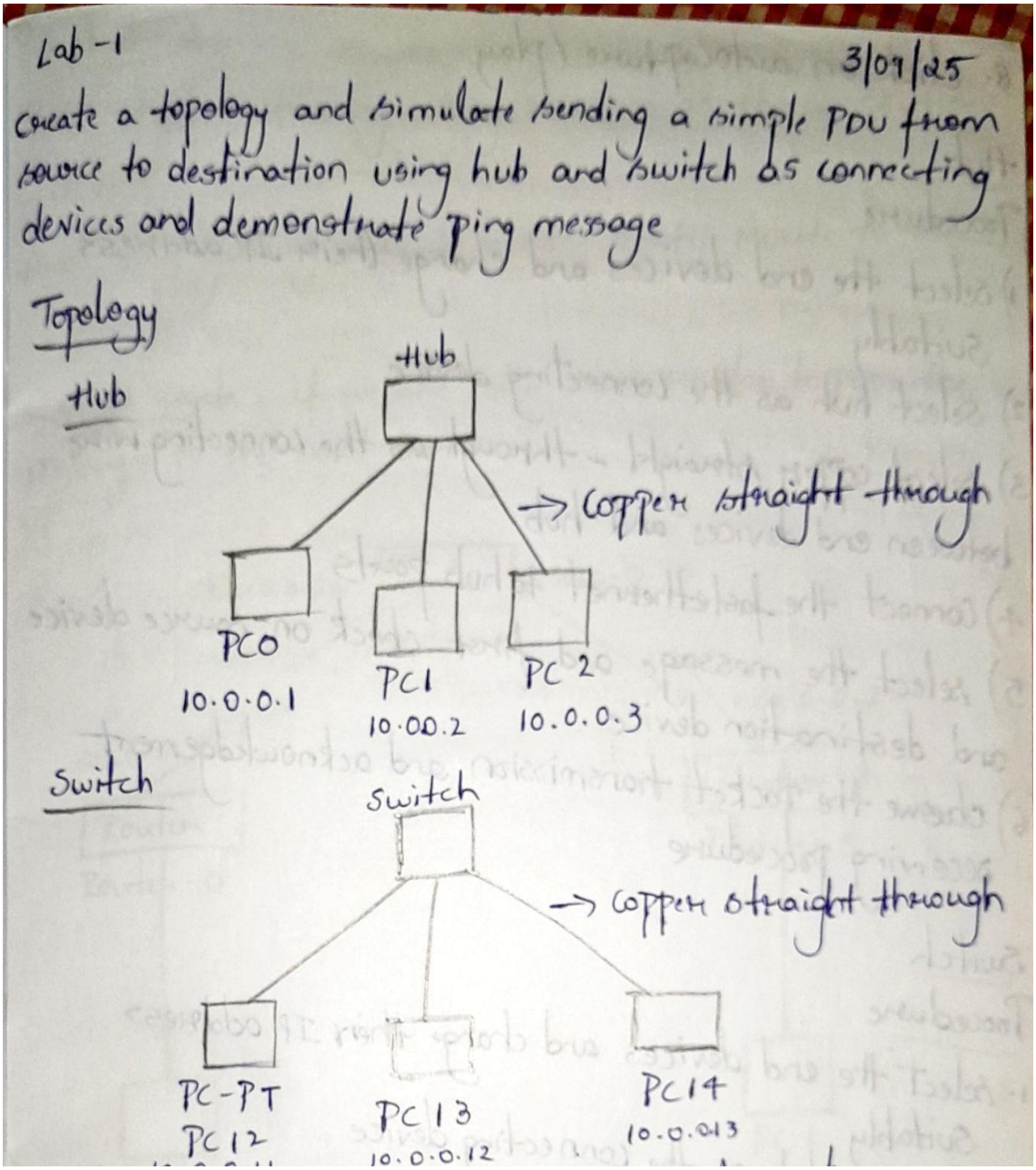
CYCLE 1:

Program 1

Aim of the program:

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

Procedure and topology:

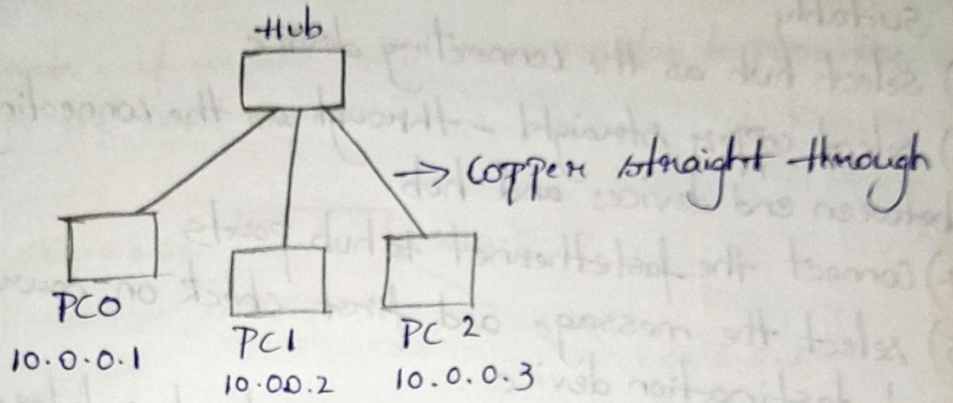


Lab-1 3/09/25

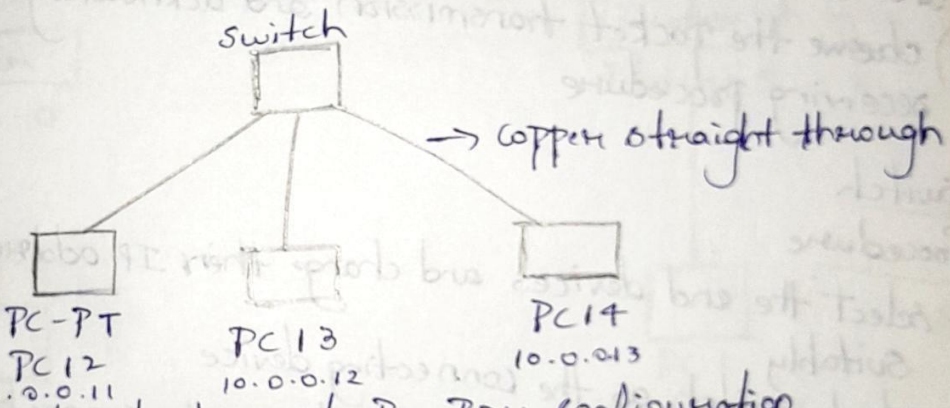
Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message

Topology

Hub



Switch



Aim: To understand simple PC-PDU configuration

Procedure

1. Select PC from end devices select generic PC and drop it in workspace similarly, select generic server
2. Choose copper - Cross - over in the connections and choose fastethernet
3. Click on server and Choose ~~fastethernet~~()
4. Click on PC and ~~go~~ to config tab. set IP address to 10.0.0.1 and click on Subnet Mask
5. Repeat same step and set IP address for server
6. In simulation mode, in edit filters click only ICMP
7. Add a simple PDU from PC to server

8. click on autocapture / play

Hub

Procedure

1) select the end devices and change their IP address suitably
2) select hub as the connecting device
3) select copper straight - through as the connecting wire between end devices and hub
4) connect the fastethernet to hub ports
5) select the message and first check on source device and destination device
6) observe the packet transmission and acknowledgement receiving procedure

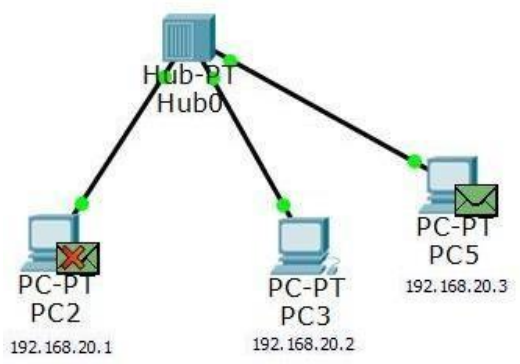
Switch

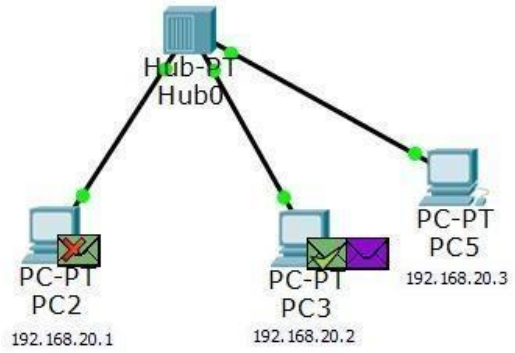
Procedure

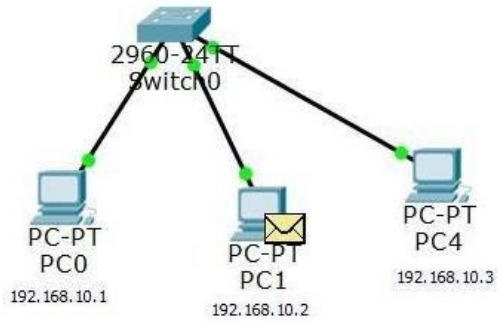
1. select the end devices and change their IP address suitably
2. select switch as the connecting device
3. Select copper straight through as the connecting wire b/w end devices and hub
4. Connect the fastethernet to hub ports
5. select the message and first click on source device and then destination device
6. observe the packet transmission and acknowledgement receiving procedure.

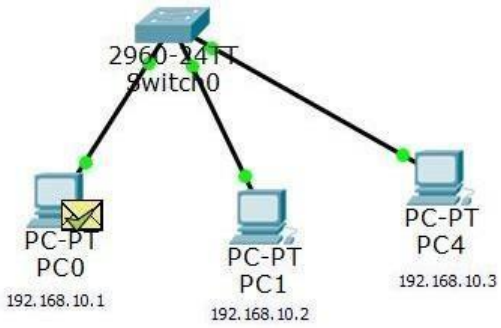
03/09

Screenshots/ Output:

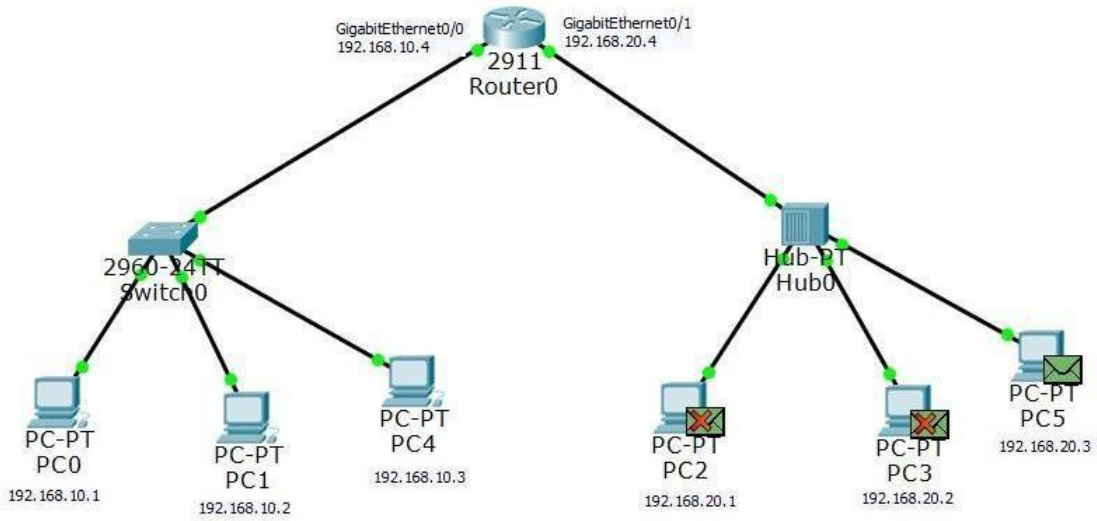


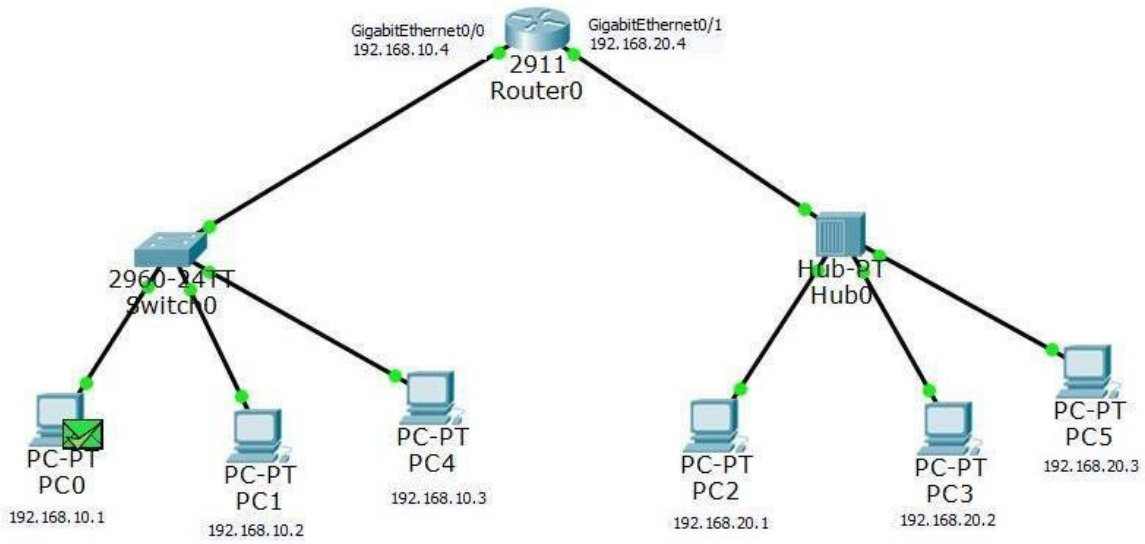






Updated topology





Observation:

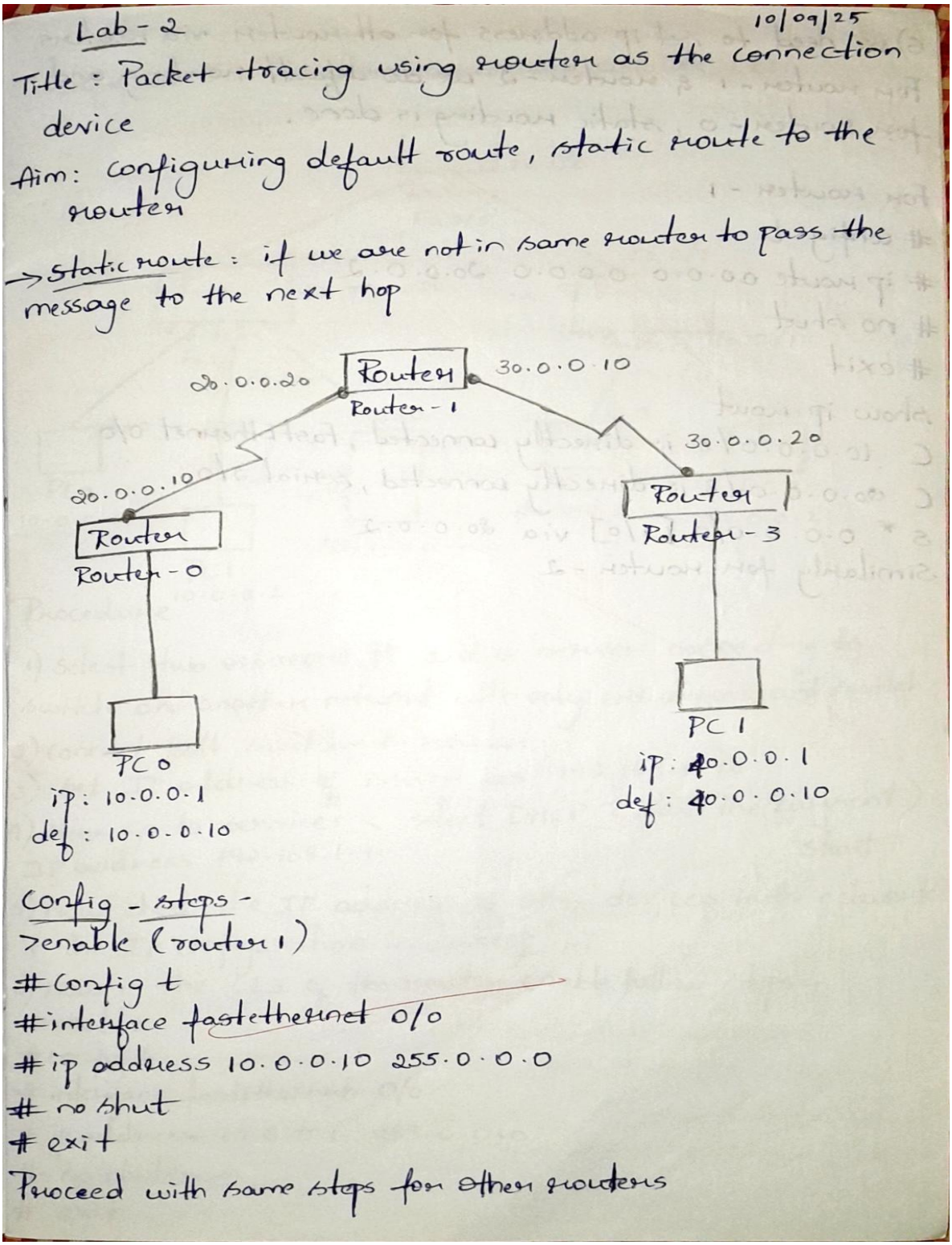
- In the hub-based topology, the PDU was broadcast to all ports, while the switch forwarded the PDU only to the destination MAC after learning addresses from incoming frames.
- Successful ICMP echo and echo-reply messages confirm that both devices enabled connectivity, with the switch demonstrating selective unicast forwarding and reduced unnecessary traffic.

Program 2

Aim of the program:

Configure default route, static route to the Router

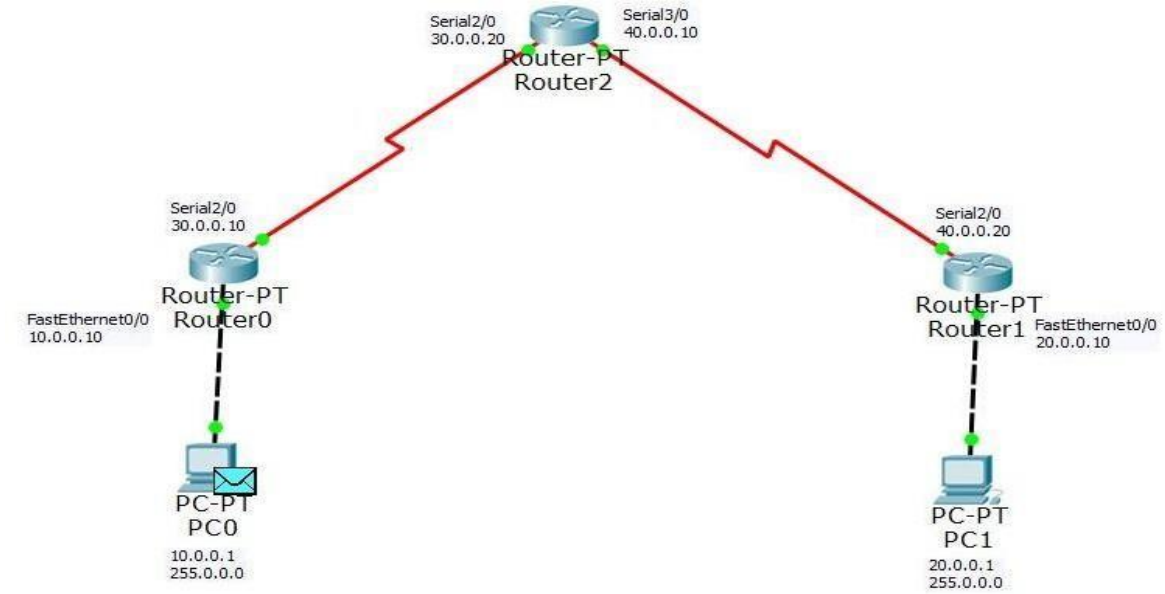
Procedure and topology:

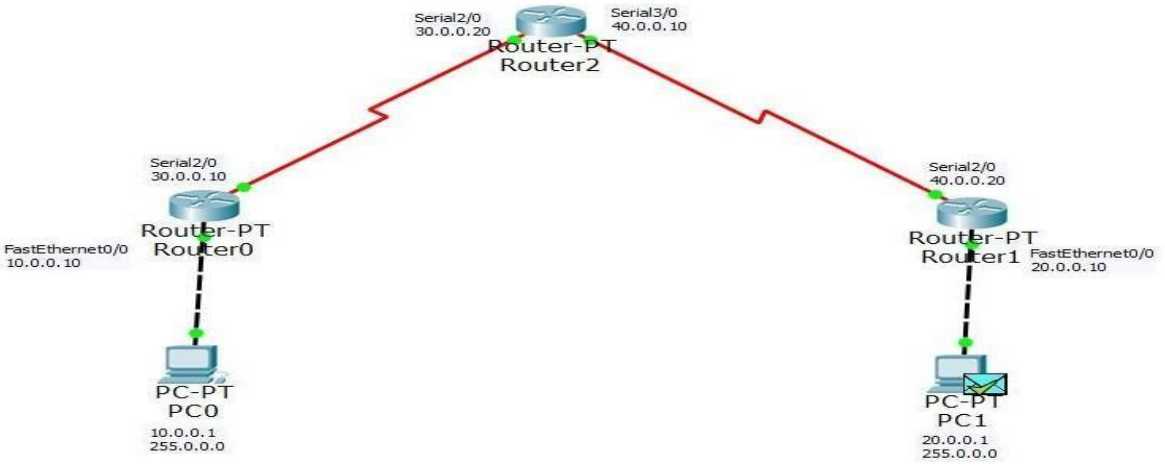


6) We need to set ip address for all routers via routers
For router-1 & router-2 we do default routing and
for router-0, static routing is done.

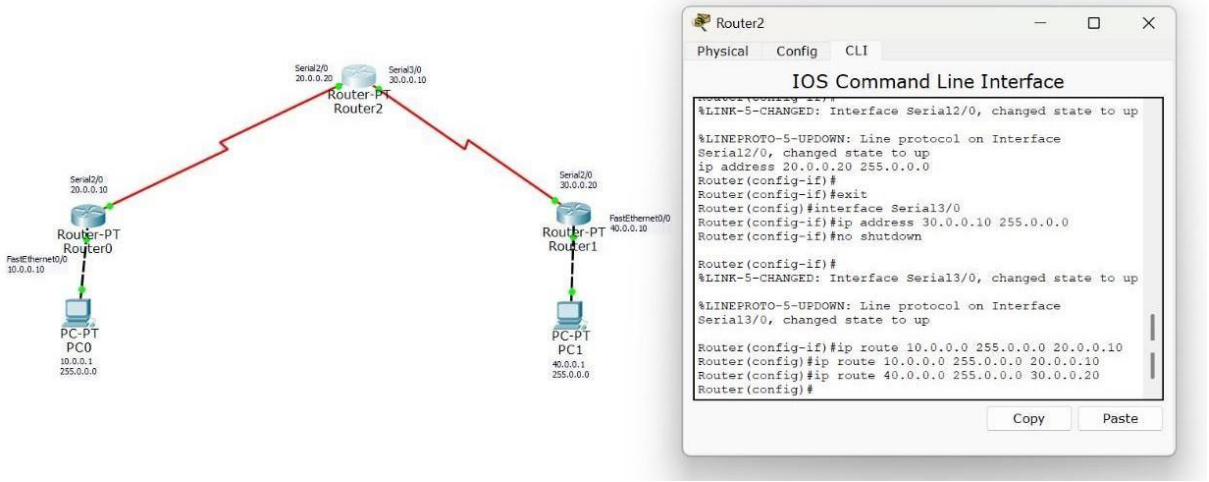
For router-1
#config t
#ip route 0.0.0.0 0.0.0.0 20.0.0.2
#no shut
#exit
show ip rout
C 10.0.0.0/8 is directly connected, FastEthernet 0/0
C 20.0.0.0/8 is directly connected, Serial 2/0
S* 0.0.0.0/0 [1/0] via 20.0.0.2
Similarly for router-2

Screenshots/ Output:

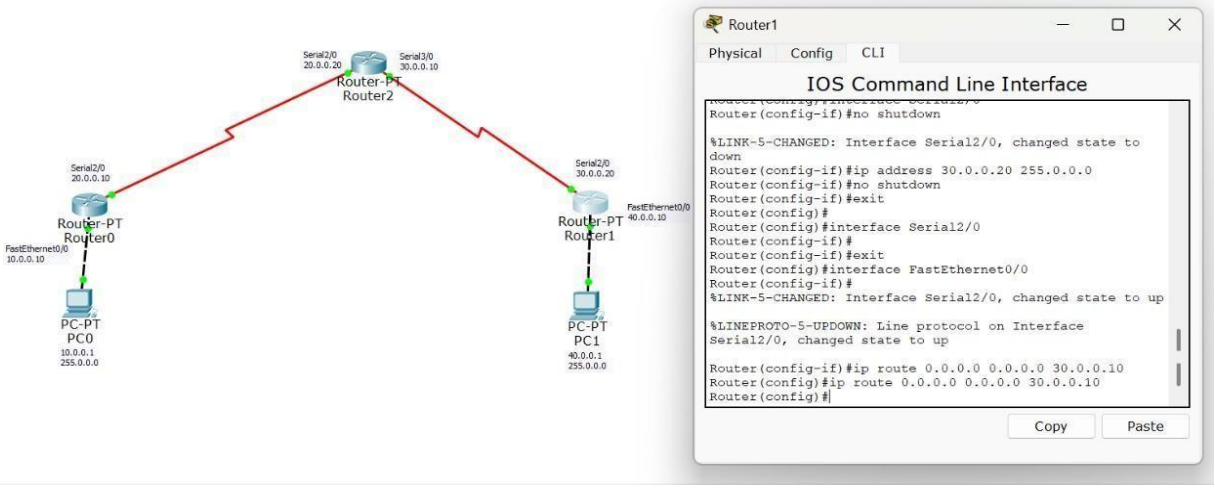




Static routing CLI commands:



Default routing CLI commands:



Observation:

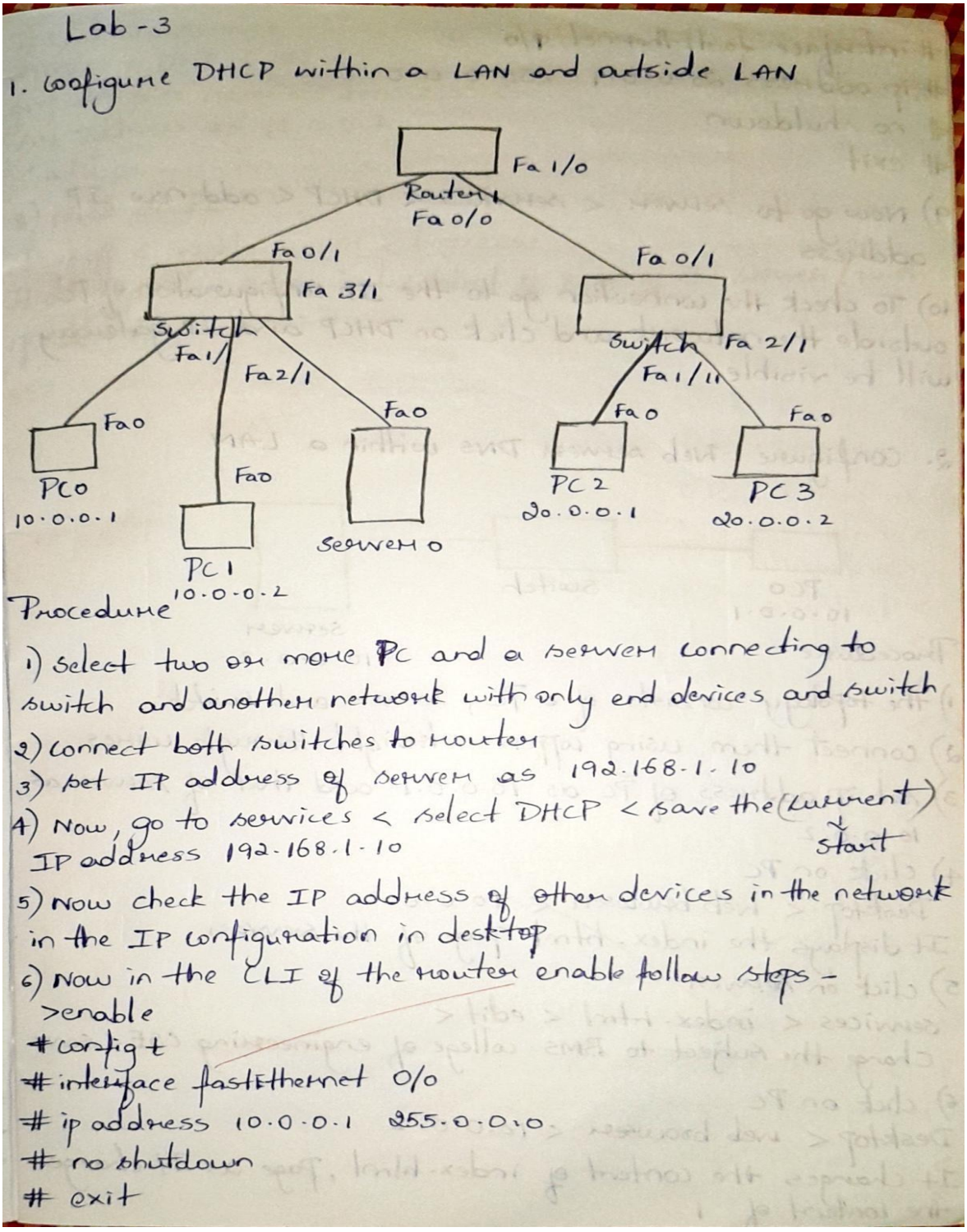
- The configured static and default routes correctly updated the router's routing table, enabling deterministic next-hop selection for remote networks.
- Successful ping tests verified that traffic was forwarded according to the static/default route entries, ensuring end-to-end reachability across different network segments.

Program 3

Aim of the program:

Configure DHCP within a LAN and outside LAN.

Procedure and topology:

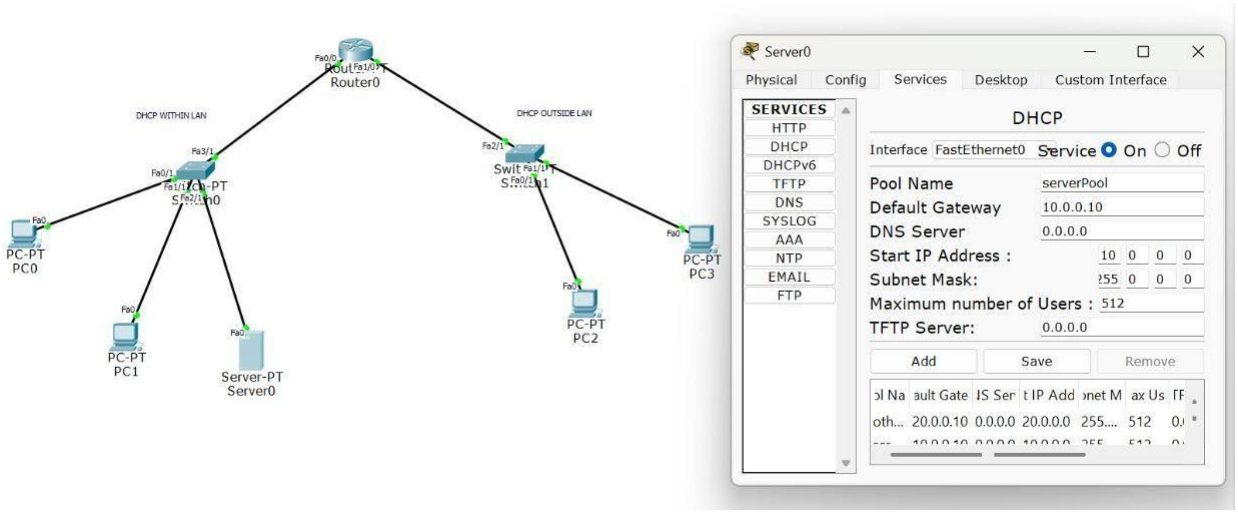


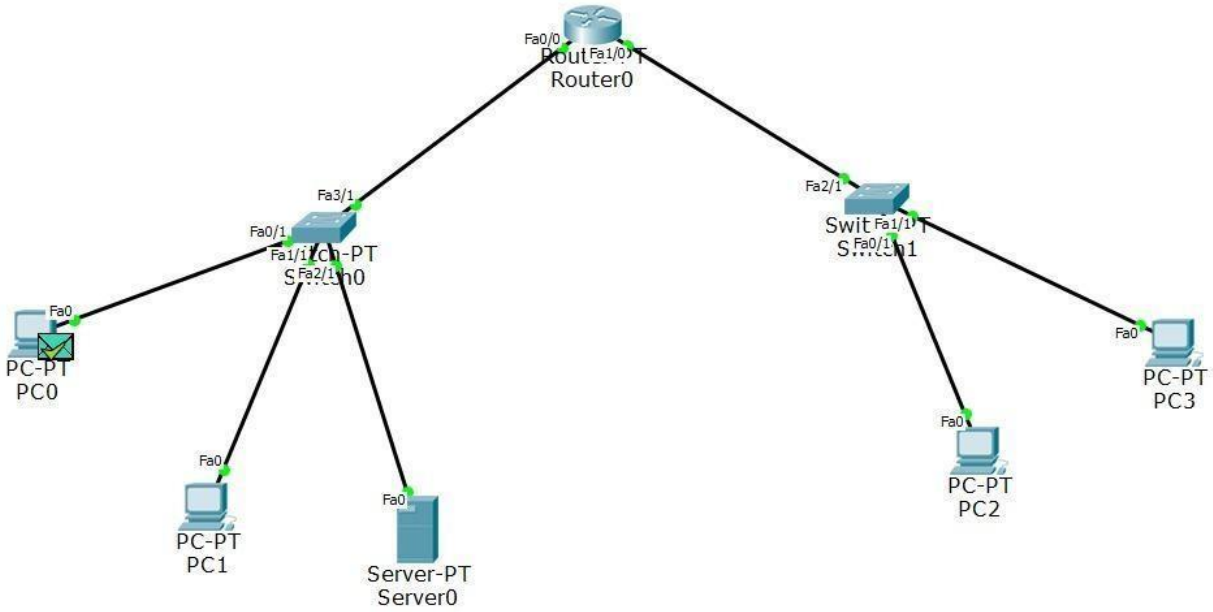
#interface fastEthernet 0/0
#ip address 20.0.0.1 255.0.0.0
no shutdown
exit

9) Now go to network < services < DHCP < add new IP address

10) To check the connection go to the IP configuration of PC outside the network and click on DHCP and IP gateway will be visible

Screenshots/ Output:





Observation:

- The DHCP server successfully allocated IP addresses to clients within the LAN, confirming proper scope configuration and automatic distribution of network parameters.
- DHCP relay (IP Helper) enabled clients outside the LAN to obtain leases from the central DHCP server, demonstrating correct inter-network forwarding of DHCP Discover and Offer messages.

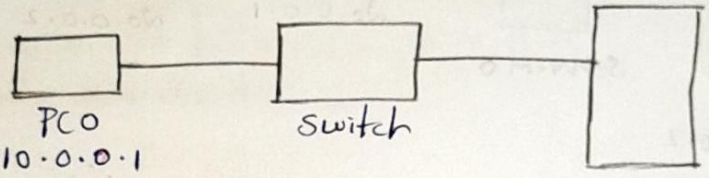
Program 4

Aim of the program:

Configure Web Server, DNS within a LAN.

Procedure and topology:

2. Configure Web server DNS within a LAN



Procedure

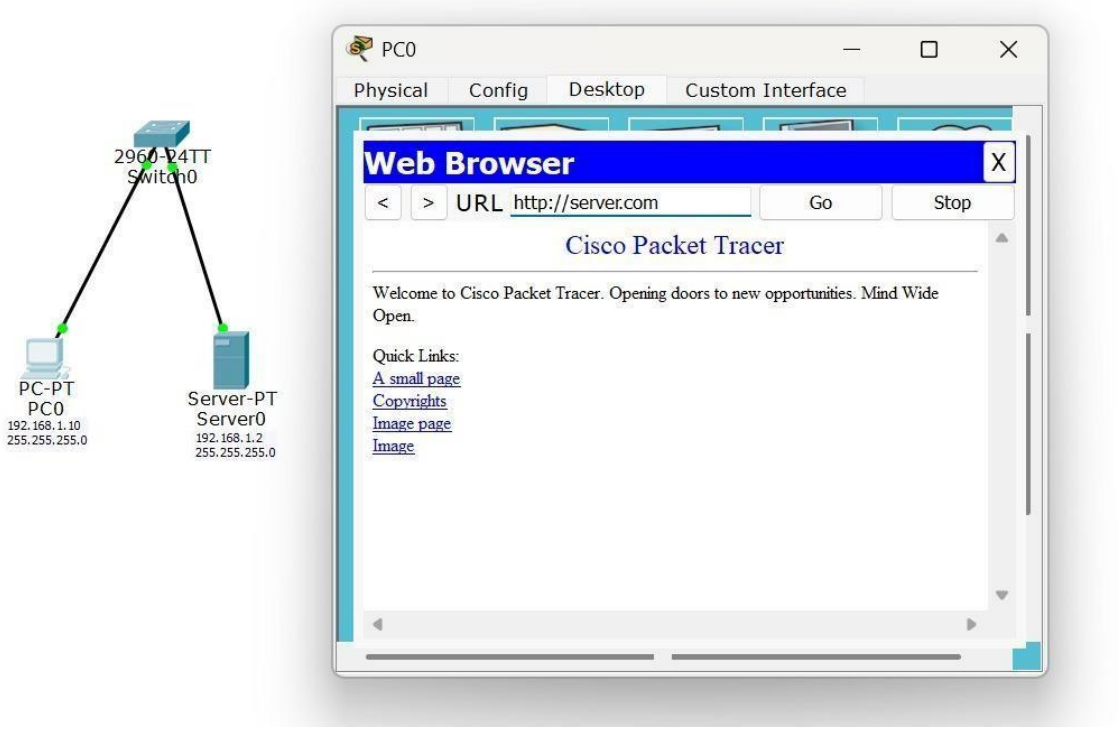
1) the topology consists of a PC, server and switch
2) connect them using copper - straight through wires
3) set ip address of PC as 10.0.0.1 and that of server as 10.0.0.2
4) click on PC
Desktop < web browser < 10.0.0.2.
It displays the index.html Page of the server
5) click on server
services < index.html < edit <
change the subject to BMS college of engineering CSE < save
6) click on PC
Desktop < web browser < 10.0.0.2
It changes the content of index.html, Page as it changes the content of 1

7) click on server
services < DNS < ON <
give domain name as bmscecse <
set address as 10.0.0.2
< add

8) click on PC
Desktop < webbrowser < bmscecse
It displays the same index.html of page of the server (with ip address 10.0.0.2)

17.09

Screenshots/ Output:



Observation:

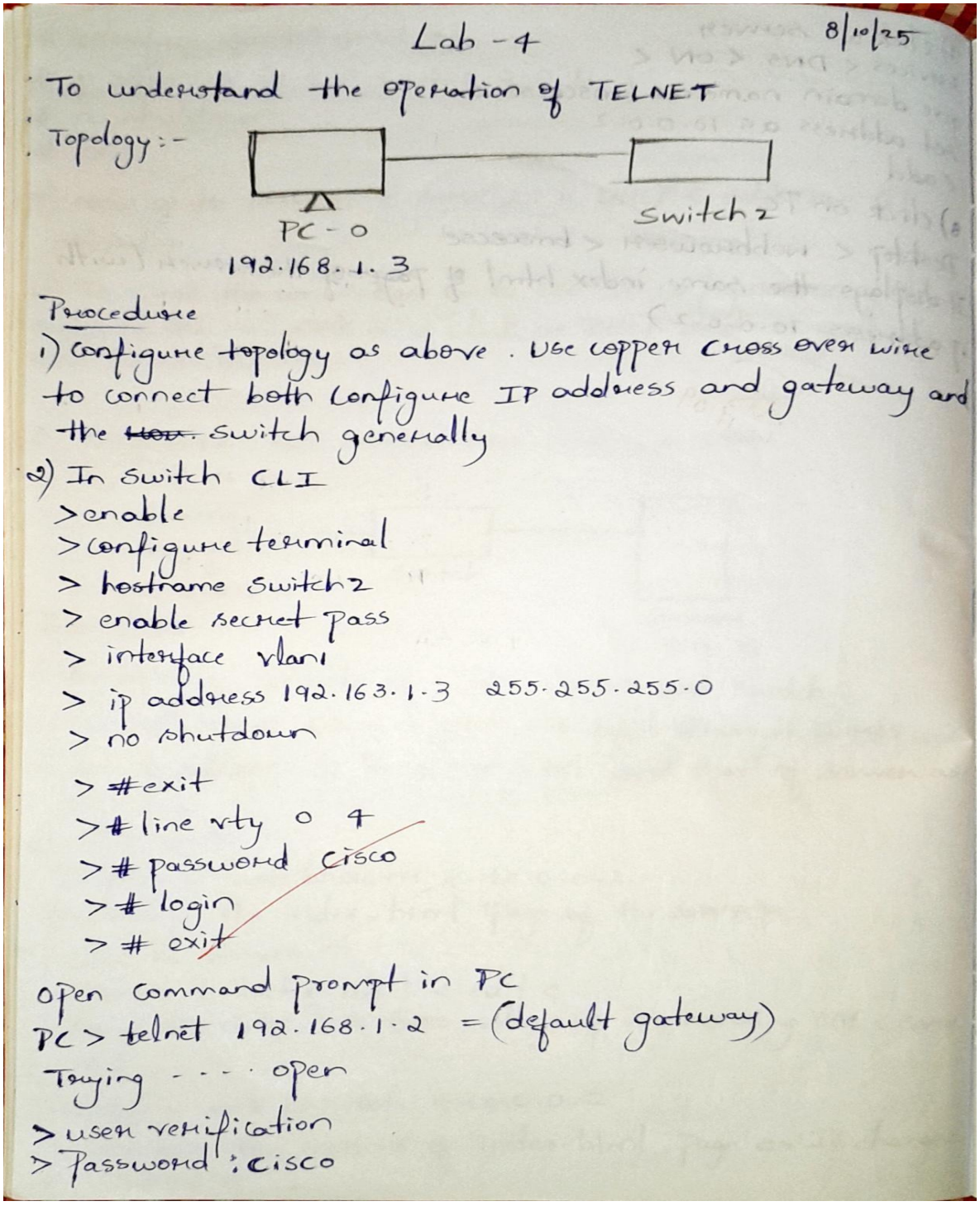
- The DNS server successfully resolved domain names to the corresponding web server's IP address, confirming proper hostname-to-IP mapping within the LAN.
- HTTP requests reached the web server using the DNS-resolved address, validating correct server configuration and internal LAN communication.

Program 5

Aim of the program:

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

Procedure and topology:



observation:

-> Telnet connection was successfully established b/w PC and switch

-> LAN access the switch command line

-> Can manage the switch directly, connected through console

-> Prompt to configure the password (cisco)

8110

router

gateway = 192.168.1.1

Fastethernet 0 = 192.168.1.2

PC0

PC1 = 192.168.1.3

gateway = 192.168.20.1

PC2 = 192.168.20.2

PC3 = 192.168.20.3

VLAN num

VLAN name

Screenshots/ Output:

Observation:

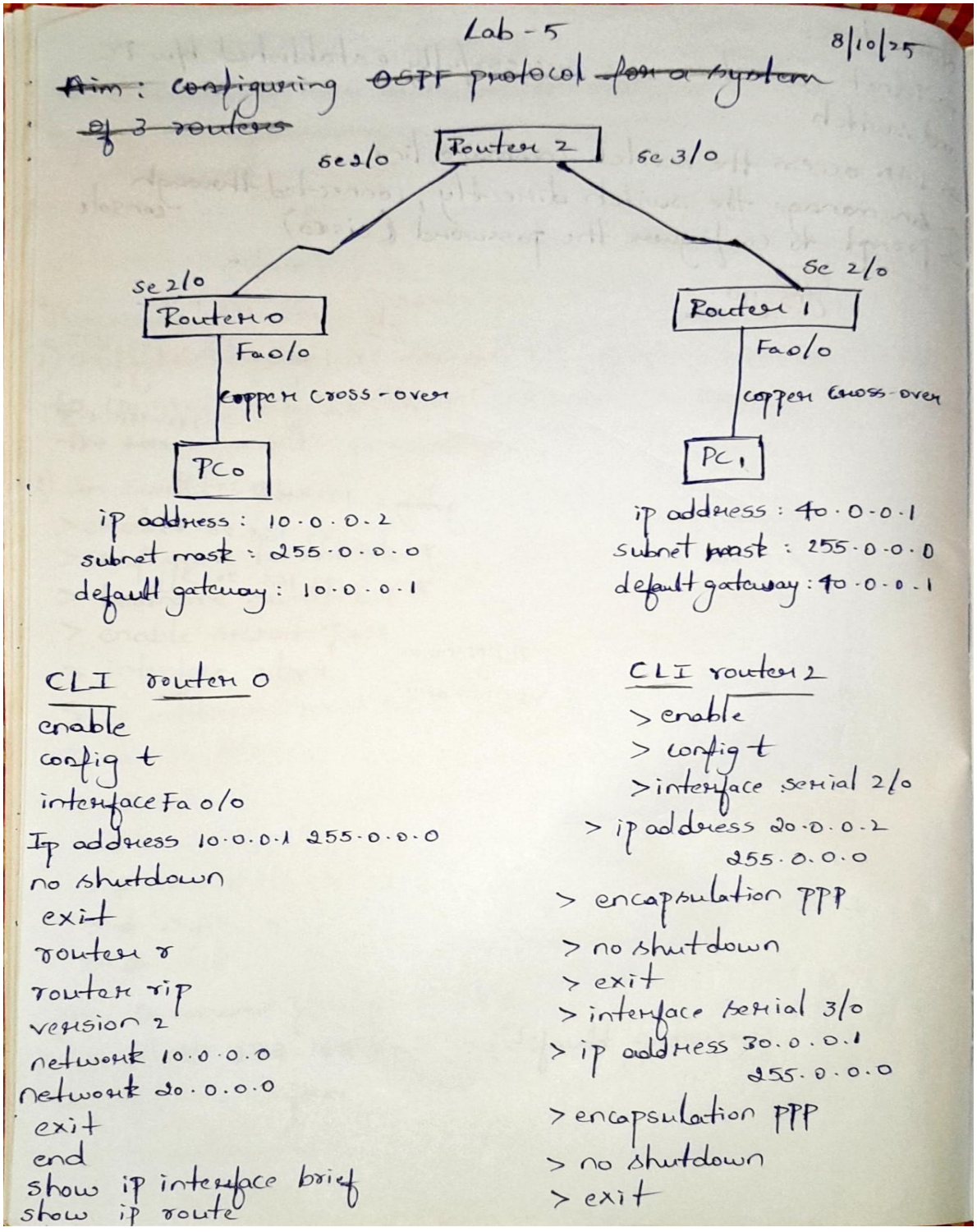
- The Telnet session successfully established a remote CLI connection to the router, confirming proper VTY line configuration and IP reachability between the IT office PC and the server room router.
- Command execution over the Telnet session demonstrated reliable remote device management.

Program 6

Aim of the program:

Configure RIP routing Protocol in Routers

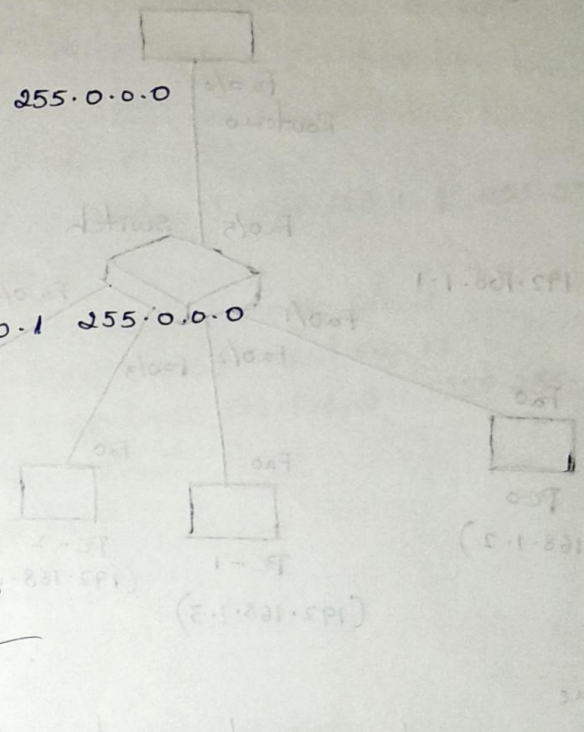
Procedure and topology:



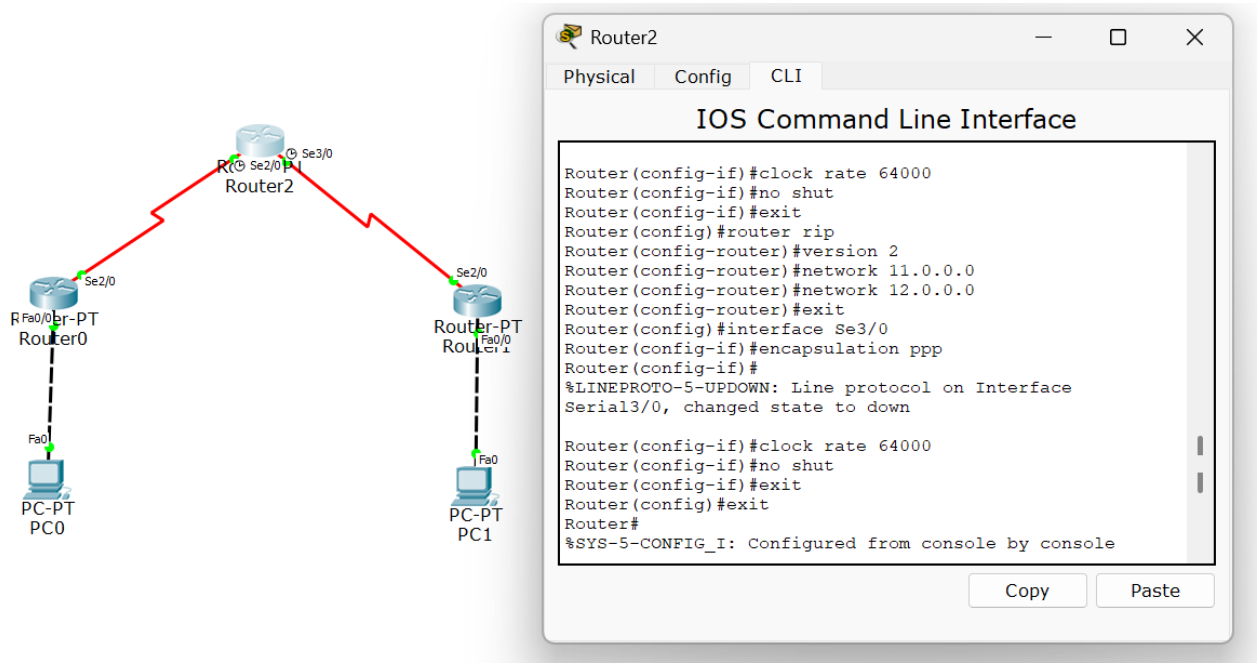
>end
>show ip interface brief
>show ip route

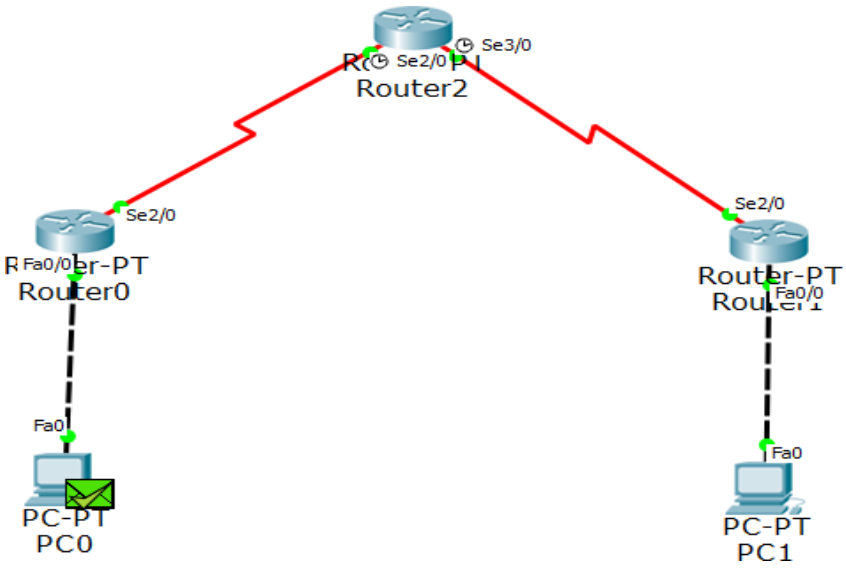
CLI router1

>enable
>config t
>interface se 2/0
>ip address 30.0.0.2 255.0.0.0
>encapsulation PPP
>no shutdown
>exit
>interface fa 0/0
>ip address 40.0.0.1 255.0.0.0
>no shutdown
>exit
>router rip
>version 2
>network 30.0.0.0
>network 40.0.0.0
>exit
>end



Screenshots/ Output:





Observation:

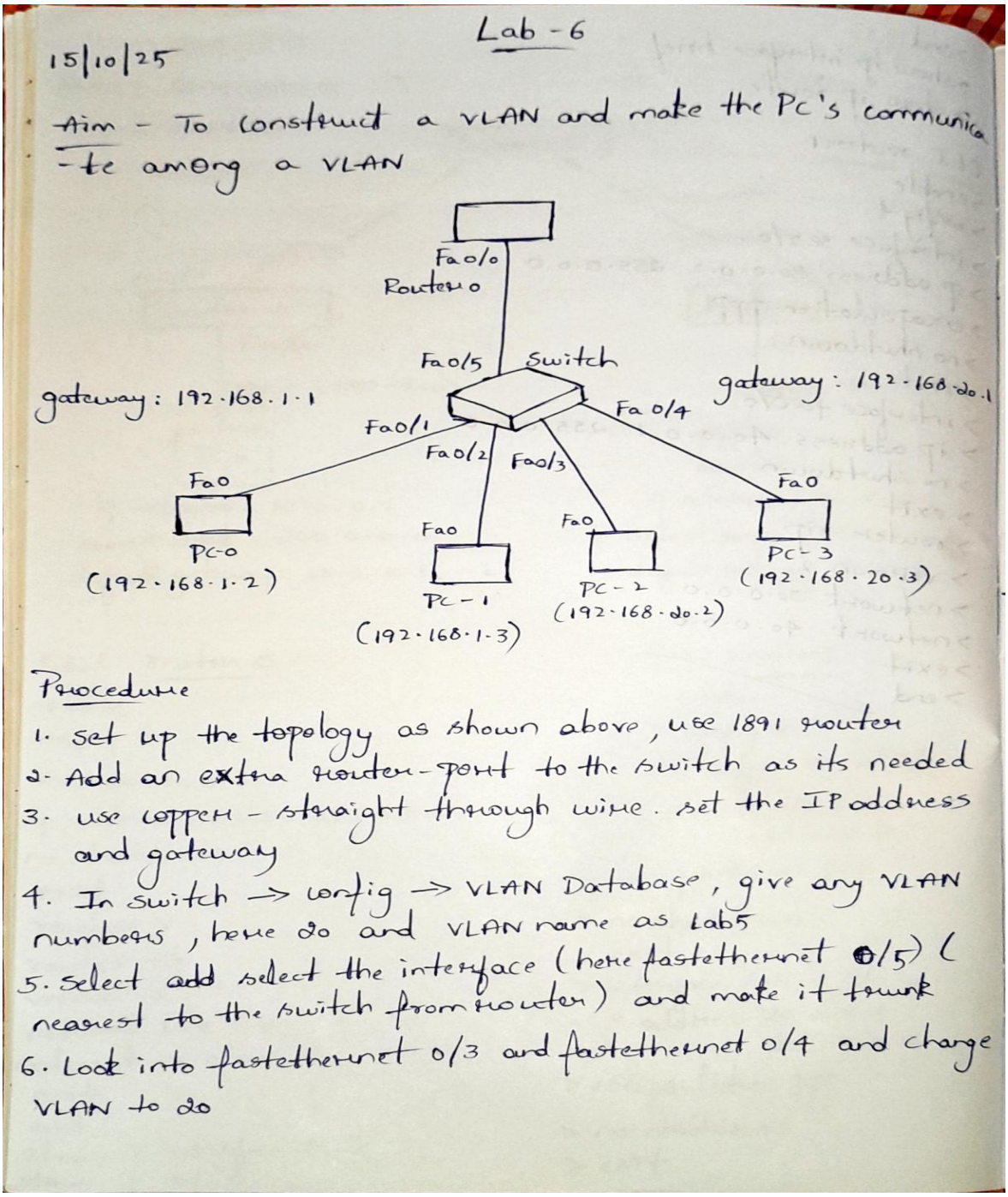
- RIP routing updates were successfully exchanged between routers, allowing each router to dynamically learn remote network routes through hop-count–based distance vector advertisements.
- The routing tables converged correctly, and successful ping tests confirmed end-to-end connectivity maintained by periodic RIP updates and route propagation.

Program 7

Aim of the program:

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

Procedure and topology:



7. In Router, select VLAN DATABASE, enter the number and name of the VLAN created.

In CLI of router

```
Router(VLAN)# exit
APPLY completed.
Exiting...
Router# config t
Router(config)# int f 0/0.1
Router(config-subif)# ip address 192.168.1.1 255.255.255.0
Router(config-if)# no shut
Router(config)# interface fastethernet 0/0.1
Router(config-subif)# encapsulation dot1q 10
Router(config-subif)# ip address 192.168.20.1 255.255.255.0
Router(config-subif)# no shut
Router(config-subif)# exit
```

Result

in PC0

```
PC> ping 192.168.20.3
pinging 192.168.20.3 with 32 bytes of data
Reply from 192.168.20.3 : bytes = 32 time = 1ms TTL = 128
Reply from 192.168.20.3 : bytes = 32 time = 1ms TTL = 128
Reply from 192.168.20.3 : bytes = 32 time = 0ms TTL = 128
Reply from 192.168.20.3 : bytes = 32 time = 0ms TTL = 128

Ping : statistics for 192.168.20.3
    Packets : sent = 4 , Recieved = 4, Lost = 0
Approximate round trip time in milliseconds.
    Minimum = 0 ms, Minimum = 1 ms, Average = 0ms
```

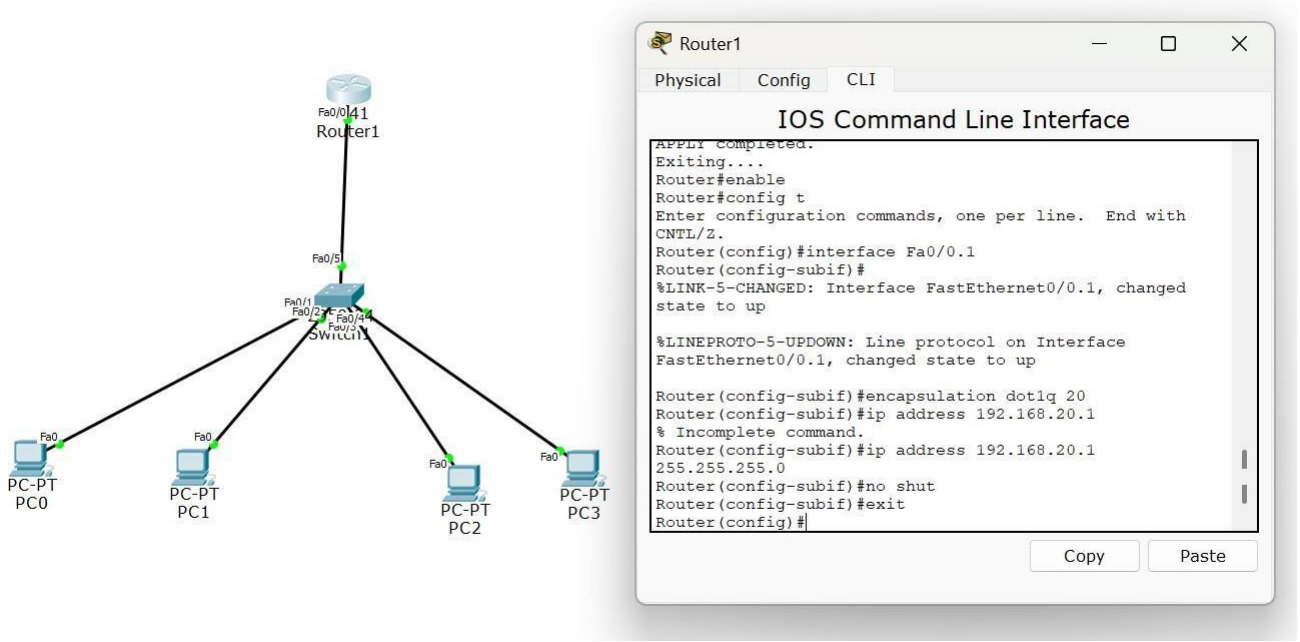
Observation:

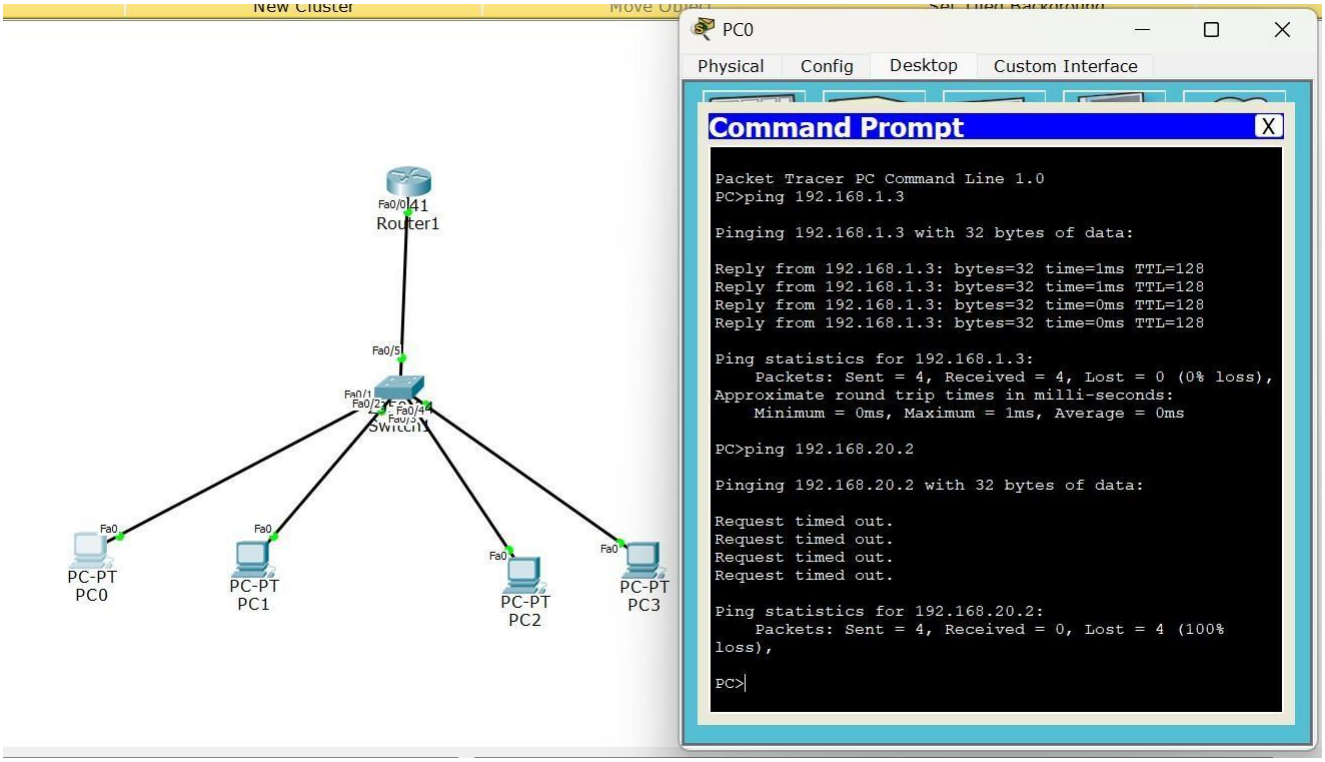
1) VLAN - virtual local area network is any broadcast domain that is partitioned and isolated in a completed network at the data link layer

2) It is a virtualized connection that converts multiple devices and network nodes from different LANs into one locally logical network.

15.10

Screenshots/ Output:





Observation:

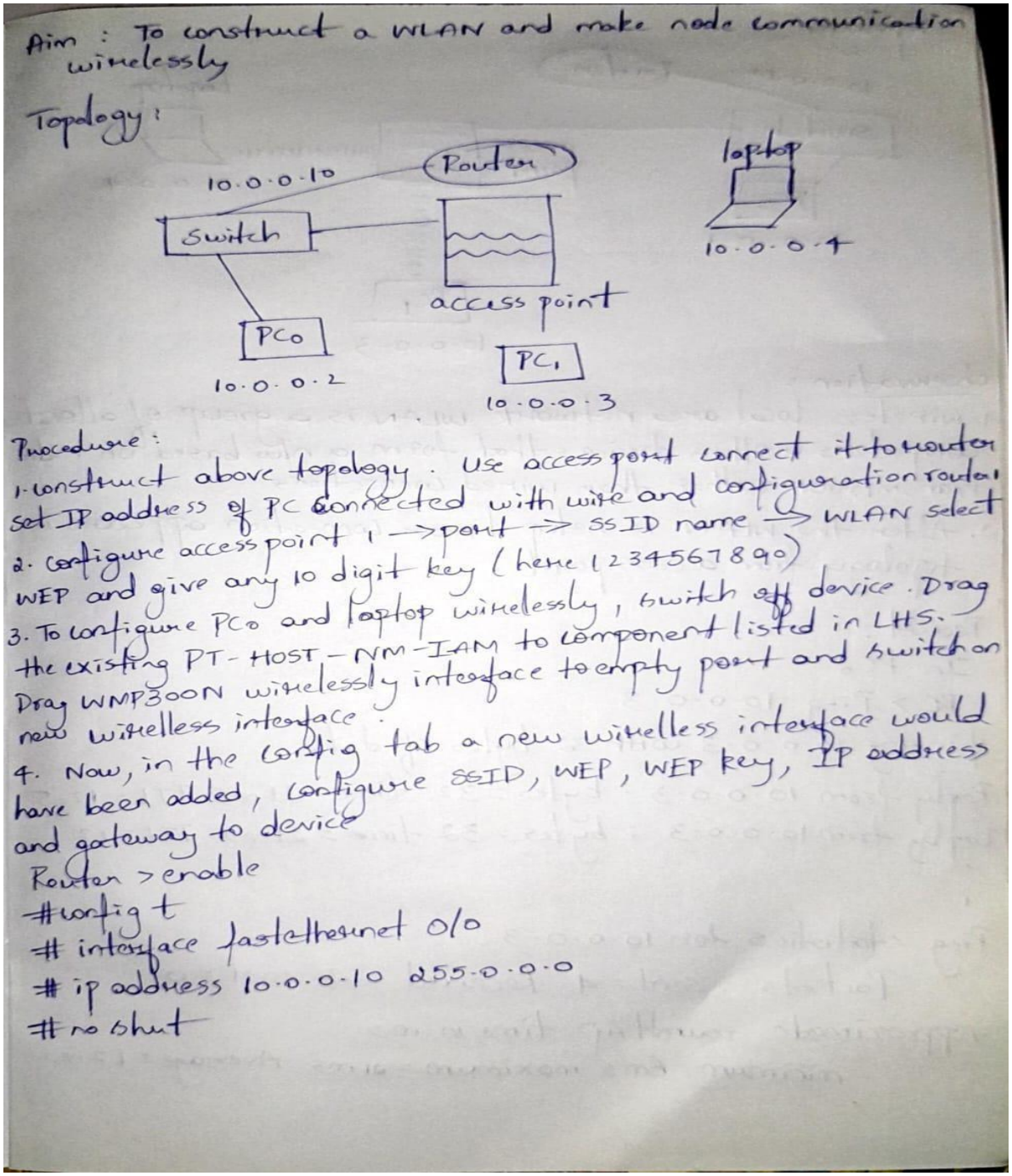
- VLAN segmentation successfully separated broadcast domains, and switch ports were correctly assigned to their respective VLAN IDs using access mode configuration.
- Inter-VLAN communication was achieved through the Layer-3 device, and successful ping tests confirmed proper VLAN membership, tagging, and routing functionality.

Program 8

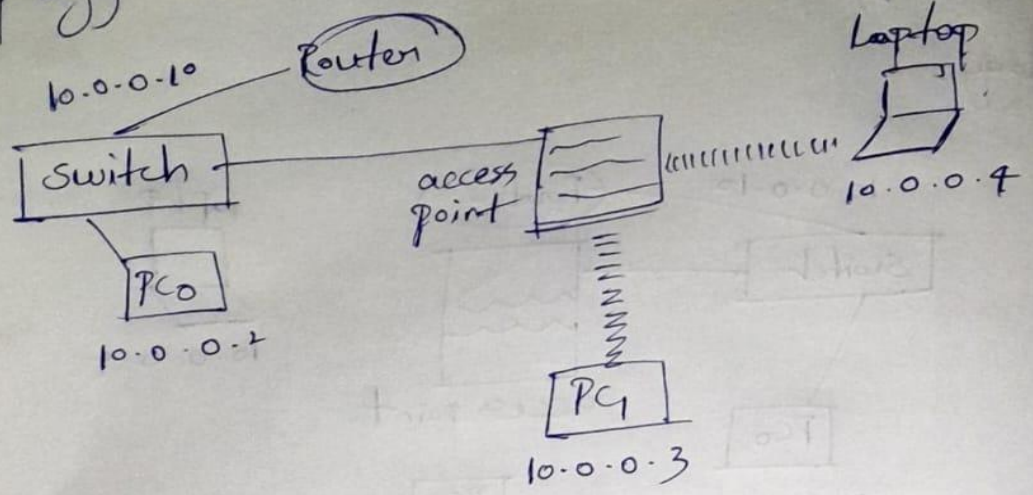
Aim of the program:

To construct a WLAN and make the nodes communicate wirelessly

Procedure and topology:



Topology



observation:

1. wireless local area network WLAN is a group of allocated computers & other devices that form a n/w based on radio transmission rather than wired connections
2. After the WLAN is setup linear connection appears in topology from access point.

Result:

In PC0

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 = 21ms TTL = 12.5

Reply from 10.0.0.3 : bytes = 32 time = 21ms TTL = 12

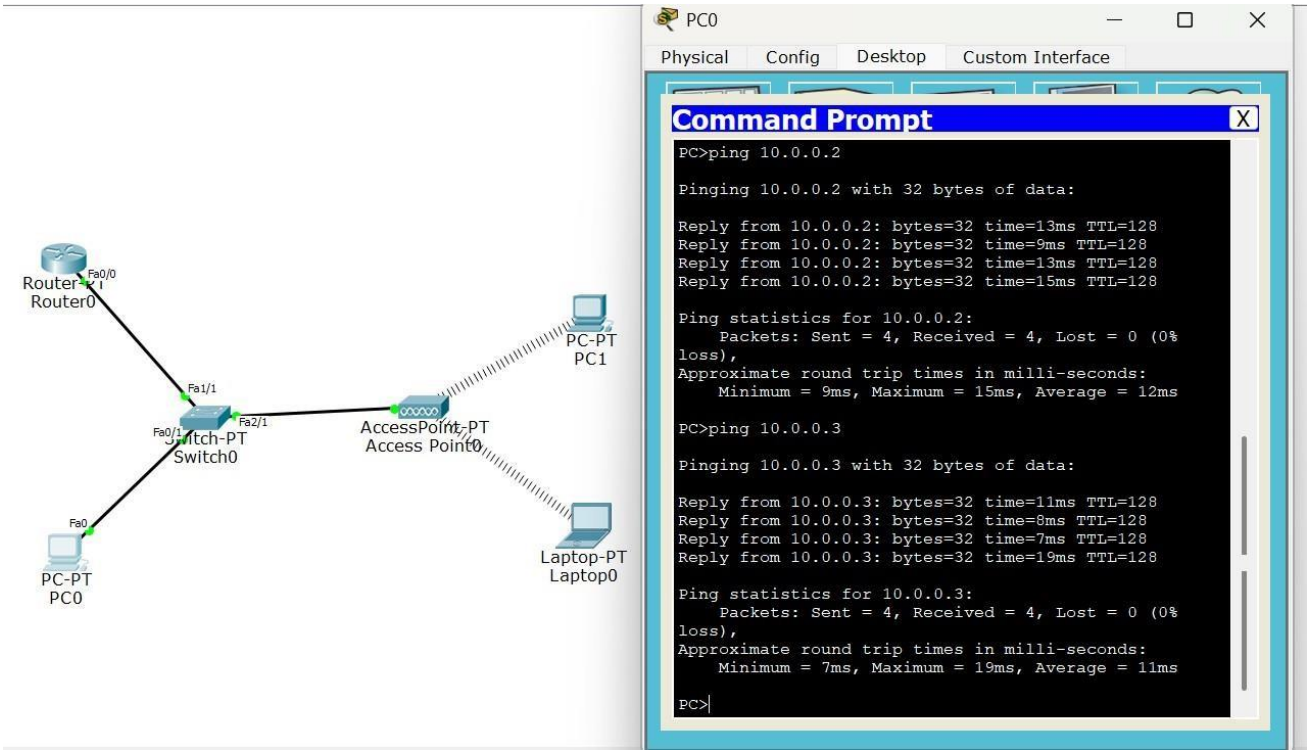
Ping statistics for 10.0.0.3

Packets : sent = 4 Received = 4 lost = 0

Approximate roundtrip time in ms

minimum = 6ms maximum = 21ms Average = 12ms

Screenshots/ Output:



Observation:

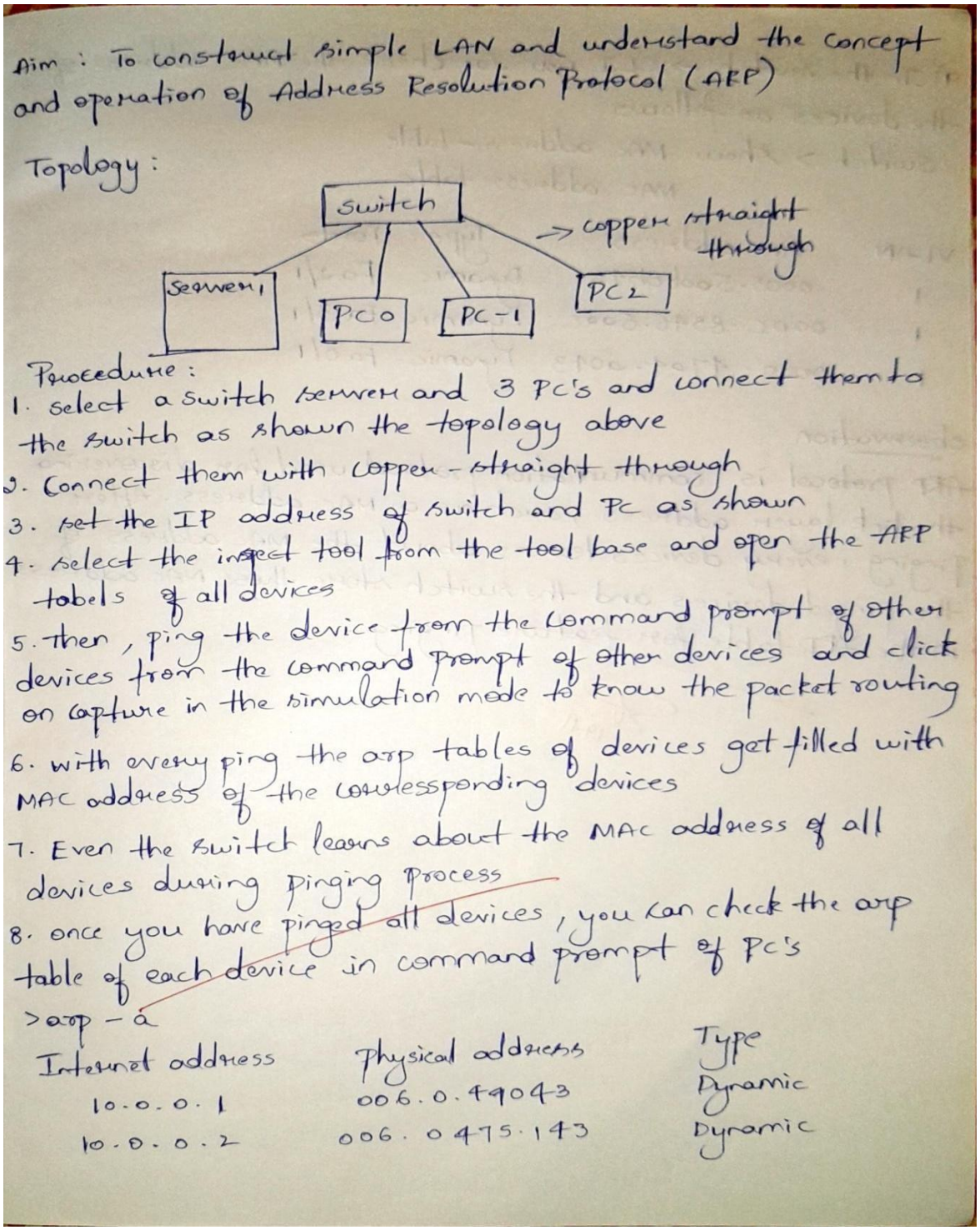
- The WLAN configuration enabled wireless nodes to associate with the access point using the configured SSID and security settings, confirming proper authentication and signal coverage.
- Successful ping communication between wireless devices verified stable wireless Connectivity.

Program 9

Aim of the program:

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

Procedure and topology:



9) In the switch → CLI you can check the MAC address of the devices as follows

Switch > show MAC address-table
MAC address table

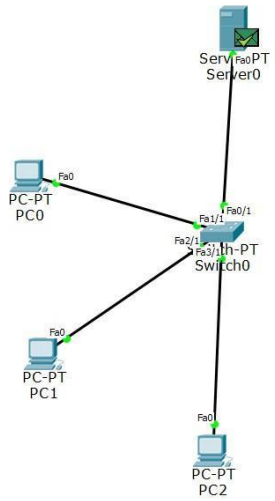
VLAN	MAC address	Type	Ports
1	0005.5a0b.0b46	Dynamic	Fa 2/1
1	000C.8546.6a9c	Dynamic	Fa 3/1
1	0060.4704.a043	Dynamic	Fa 0/1

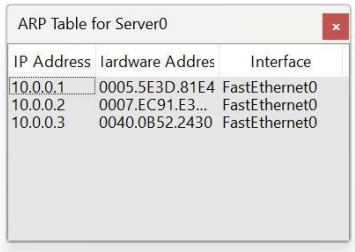
observation

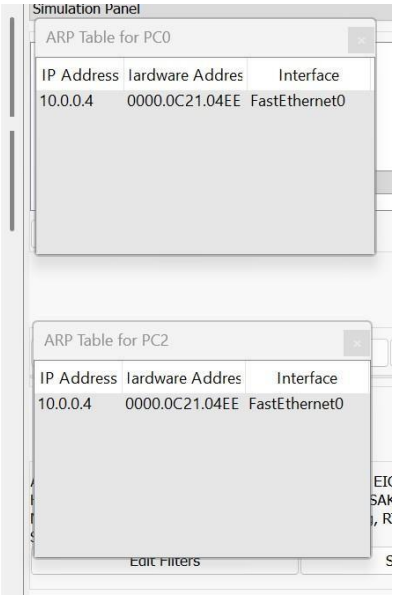
ARP protocol is communication protocol used for discovering the link layer address, such as a MAC address. After Pinging, every devices learns about the MAC address of the pinged devices and the switch store these MAC address in the ARP table for feature pinging.

19.11

Screenshots/ Output:







Observation:

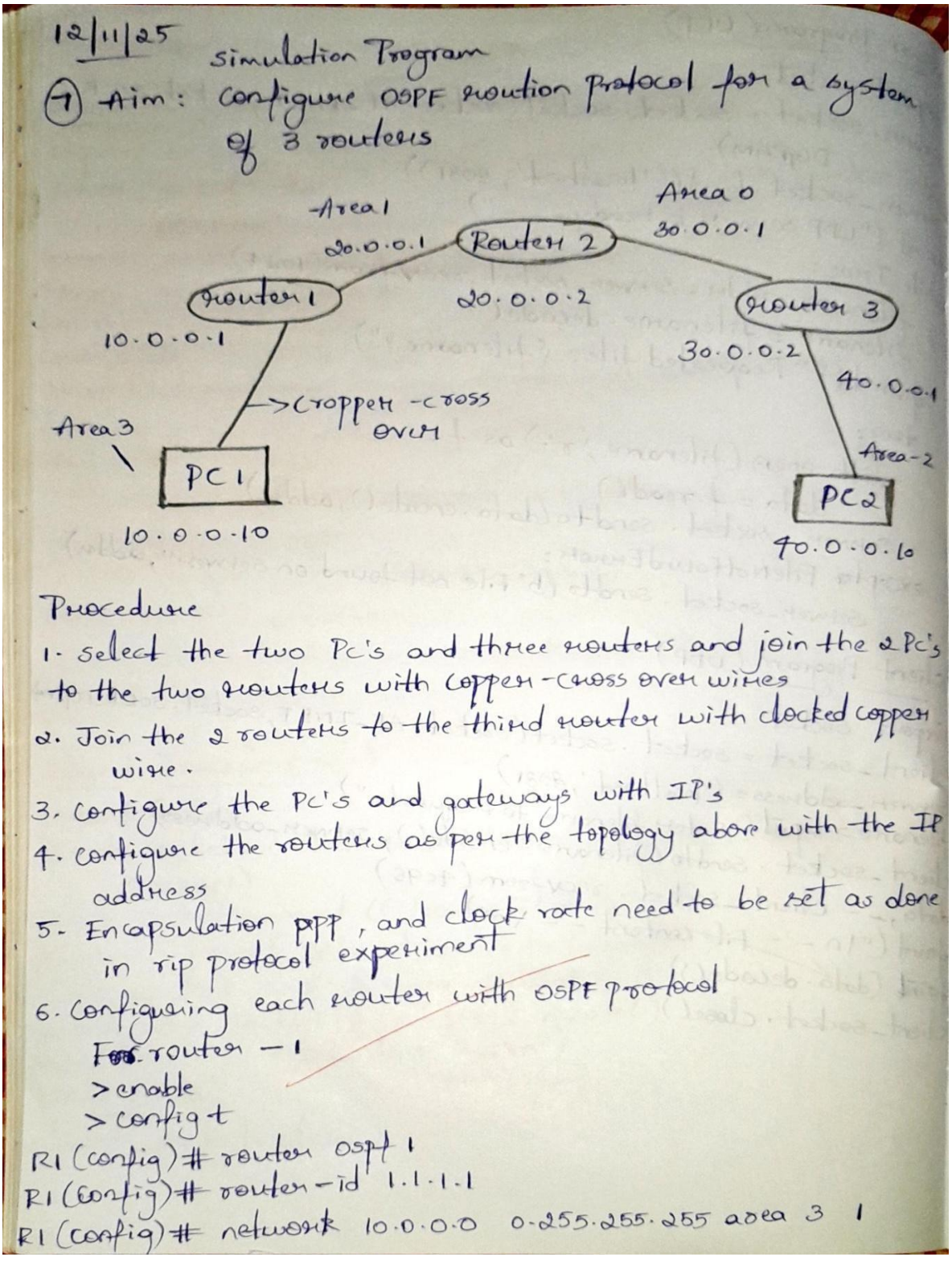
- ARP successfully resolved the destination IP address to its corresponding MAC address, as seen from ARP request and reply exchanges between LAN hosts.
- The populated ARP tables and successful ping communication confirmed correct layer-2 addressing, frame forwarding, and basic LAN operation.

Program 10

Aim of the program:

Configure OSPF routing protocol

Procedure and topology:



R1(config-r)# network 20.0.0.0 0.255.255.255 area 1 3
R1(config-r)# exit

Router 0

>config t
Ro(config)# router ospf 1
Ro(config)# router-id 2.2.2.2
Ro(config-r)# network 20.0.0.0 0.255.255.255 area 1 3
Ro(config-r)# network 30.0.0.0 0.255.255.255 area 0
Ro(config-r)# exit

Router 2

>config t
R2(config)# router ospf 1
R2(config)# router-id 3.3.3.3
R2(config)# network 30.0.0.0 0.255.255.255 area 0
R2(config)# network 40.0.0.0 0.255.255.255 area 2

7. configuring the interface

R1(config-if)# interface loopback 0
R1(config-if)# ip add 172.16.1.252 255.255.0.0
R1(config-if)# no shutdown

R2(config-if)# interface loopback 0
R2(config-if)# ip address 172.16.1.253 255.255.255.0
no shutdown

R3(config-if)# interface loopback 0
ip address 172.16.1.254 255.255.0.0
no shutdown

R3 # show ip route
C to 40.0.00/8 is directly connected

30.0.0.0/8 is directly connected, serial 3/0
30.0.0.1/32 is directly connected serial 3/0
40.0.0.1/32 is directly connected

8) In Router R1

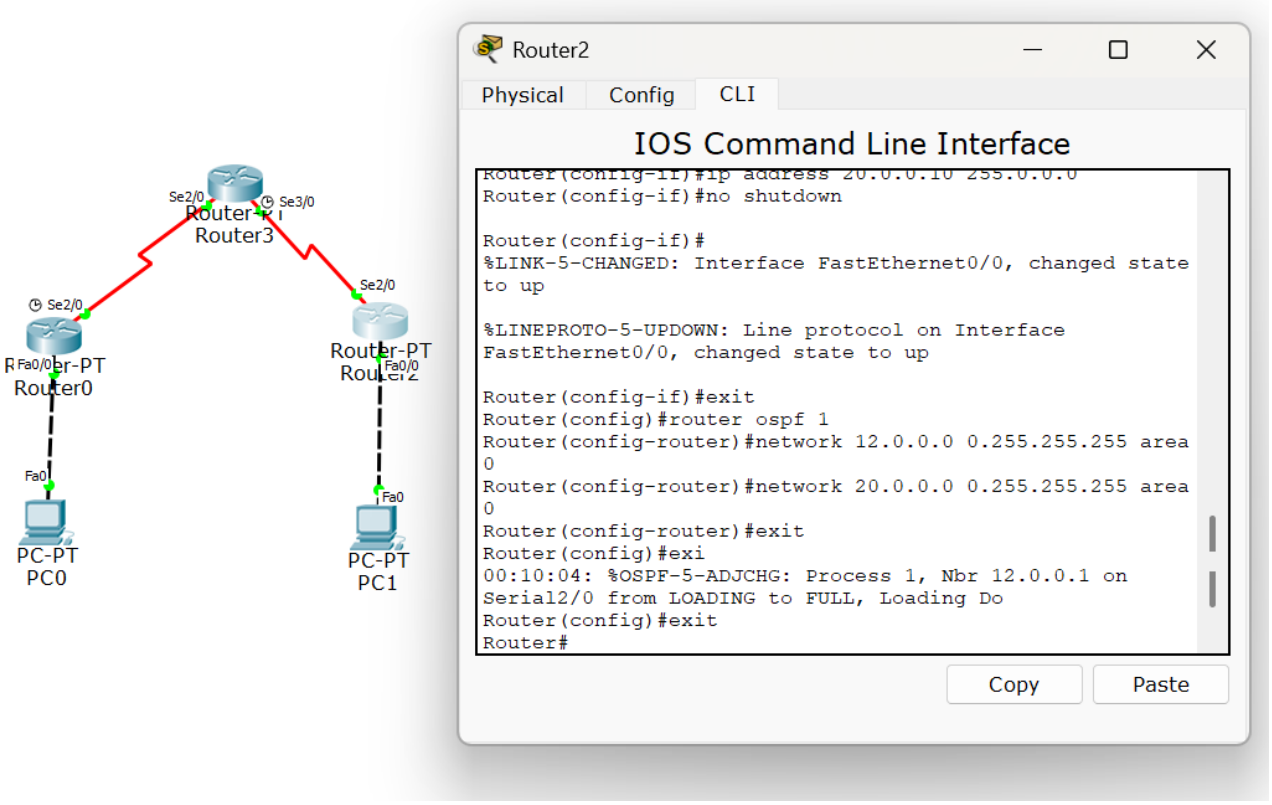
R1 (config) # router ospf 1
area 1 virtual link 2.2.2.2

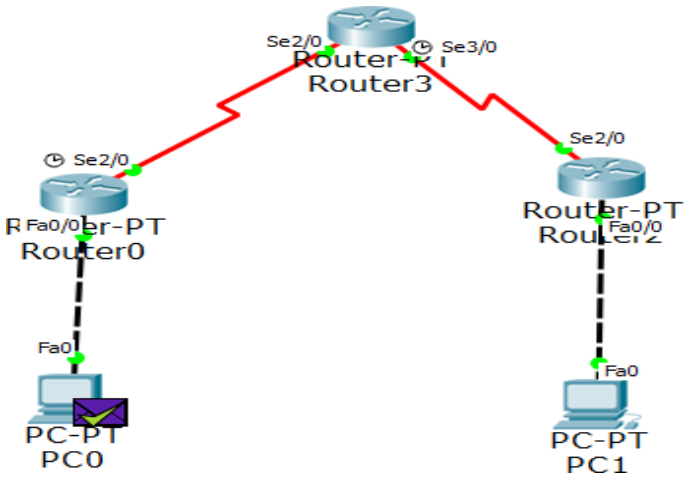
In Router R2

R2 (config) # router ospf 1
area 1 virtual link 2.2.2.2

12.11

Screenshots/ Output:





Observation:

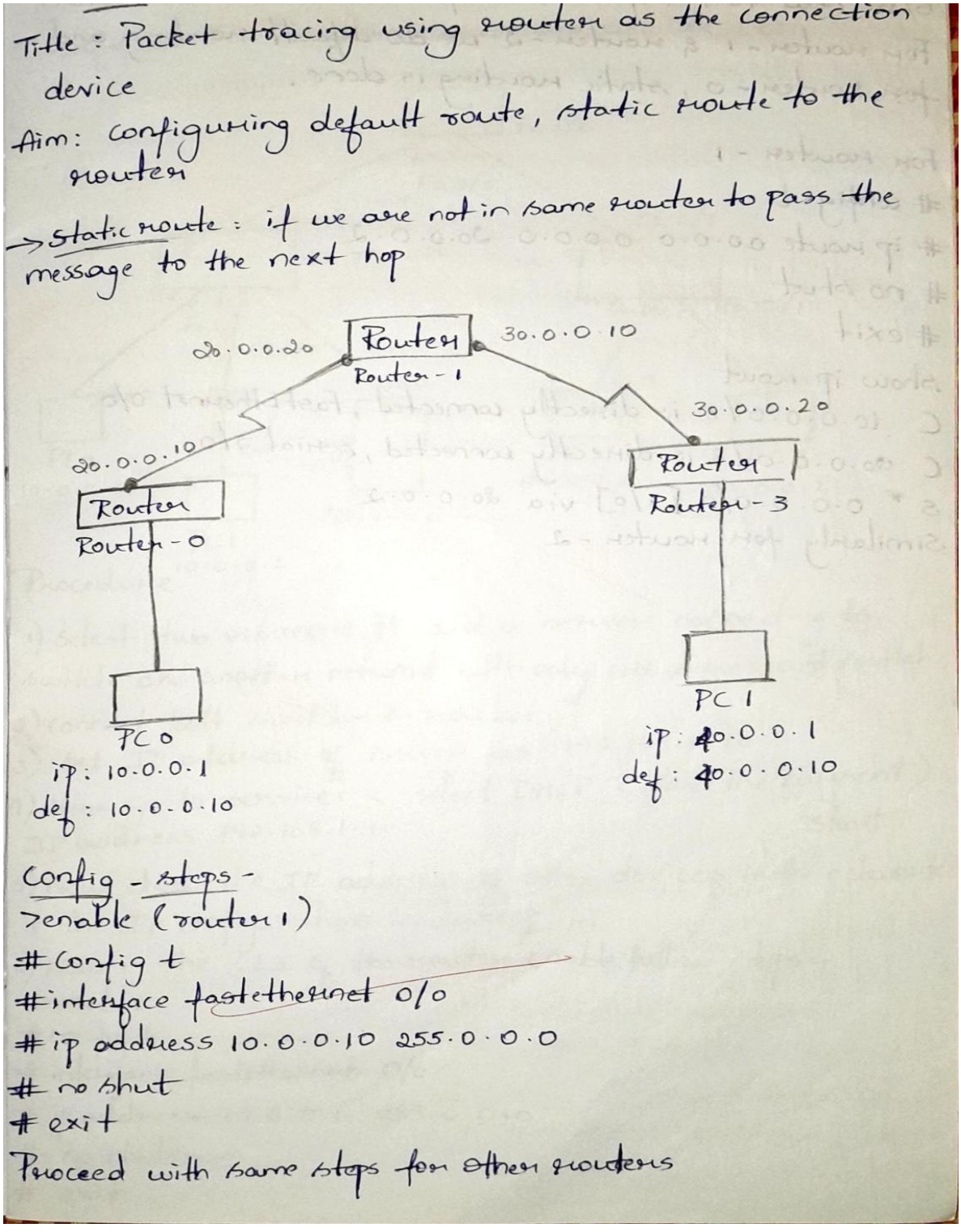
- OSPF successfully established neighbour adjacencies and exchanged LSAs, allowing routers to build a synchronized link-state database across the OSPF area.
- The routing tables converged using SPF calculations, and successful pings confirmed efficient path selection and dynamic route learning through OSPF.

Program 11

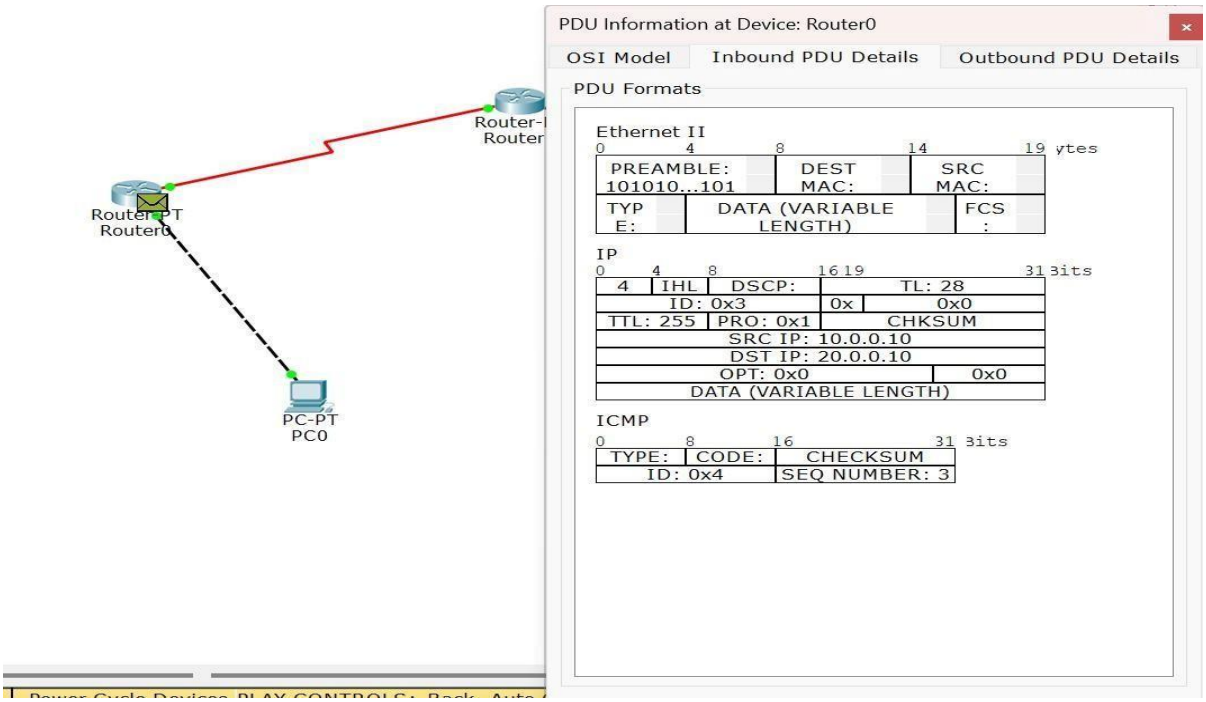
Aim of the program:

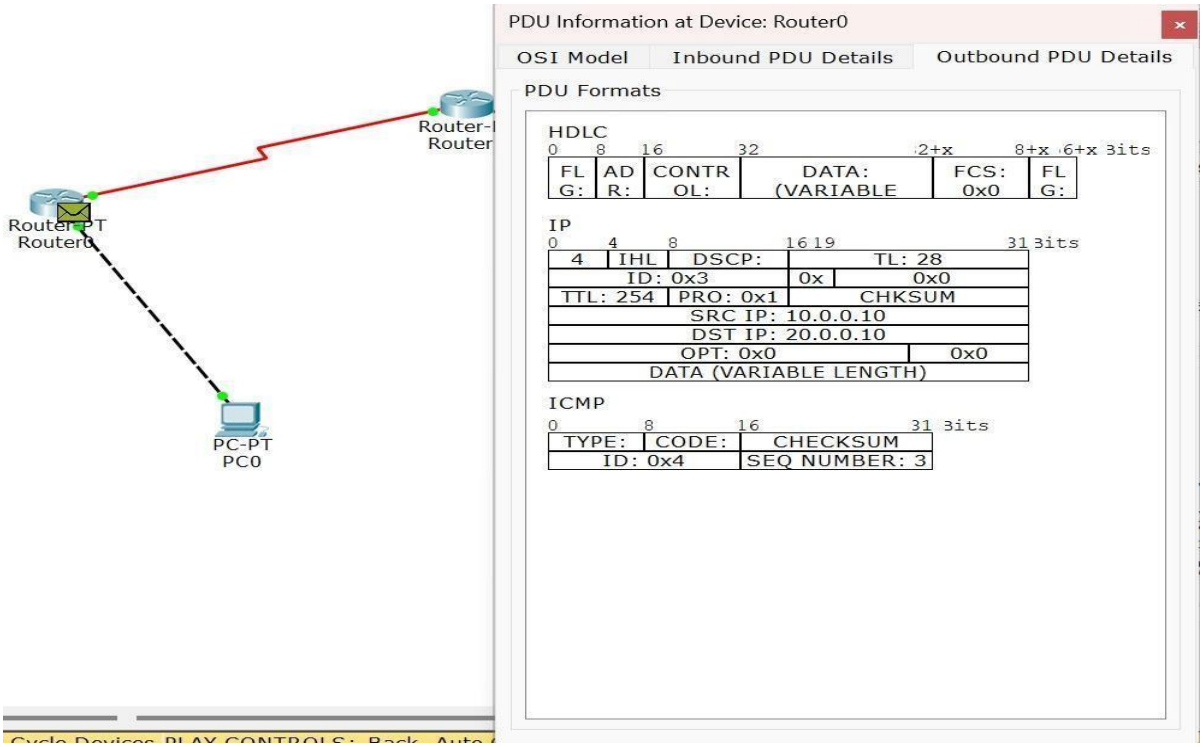
Demonstrate the TTL/ Life of a Packet

Procedure and topology:



Screenshots/ Output:





Observation:

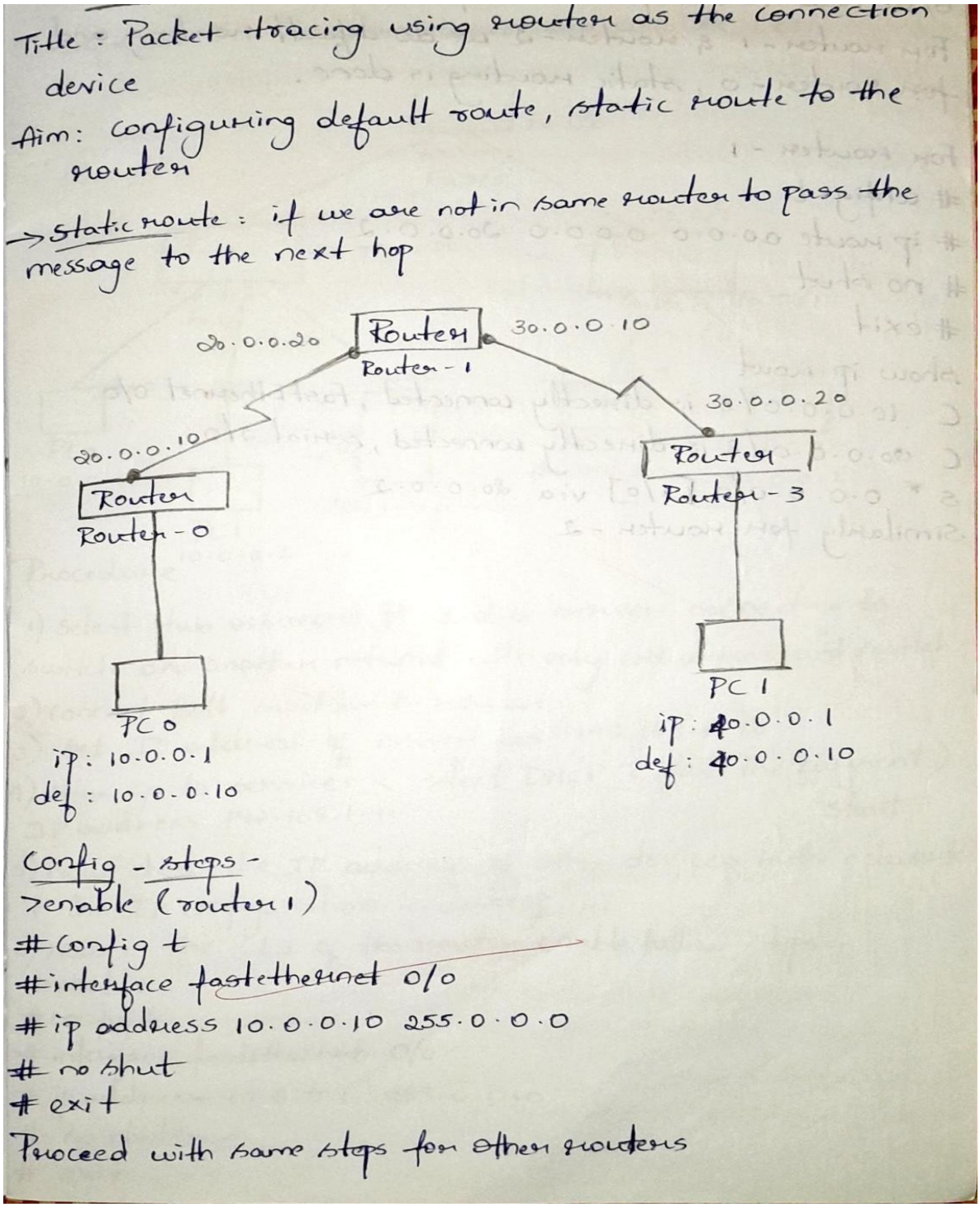
- The TTL field in the IP header decreased by one at each router hop, demonstrating its role in preventing packets from looping indefinitely in the network.

Program 12

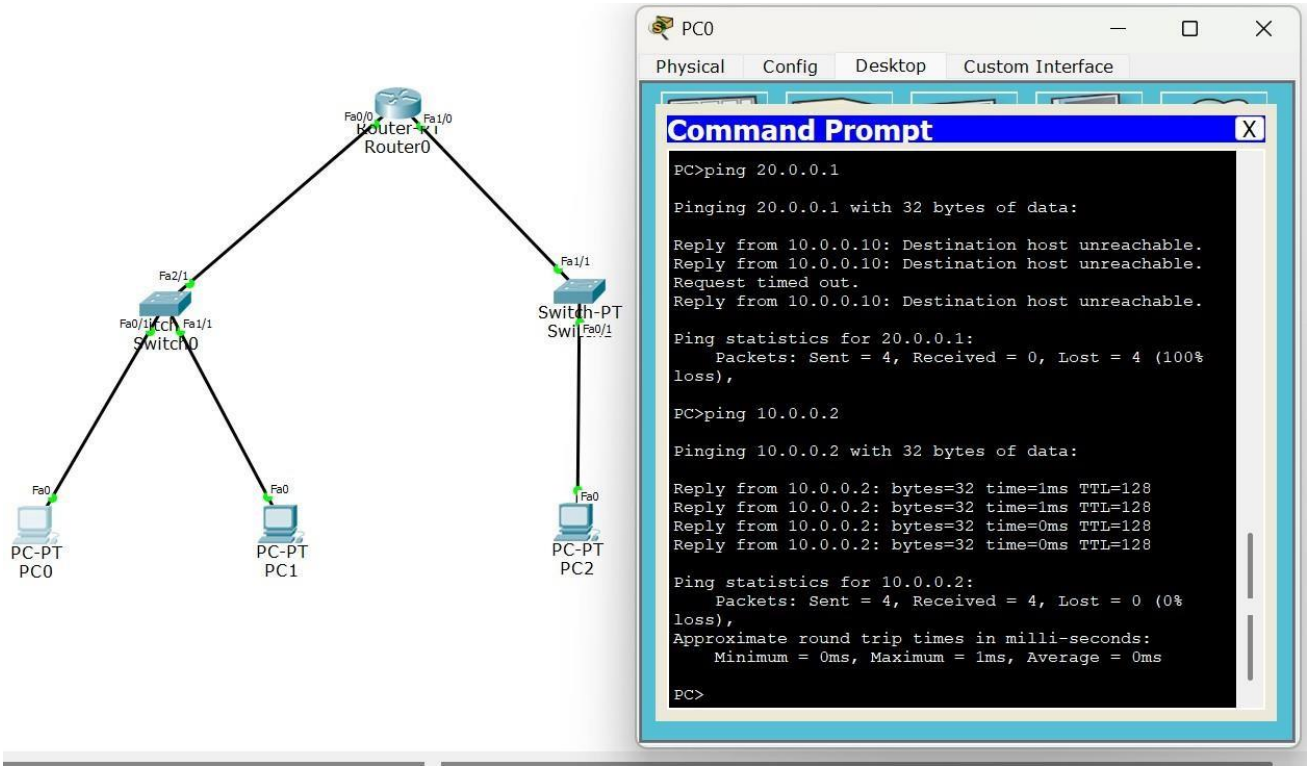
Aim of the program:

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

Procedure and topology:



Screenshots/ Output:



Observation:

- Proper IP addressing on router interfaces enabled successful ICMP echo and echo-reply communication, confirming correct Layer-3 configuration and reachability.
- “Destination Unreachable” and “Request Timed Out” responses occurred when routes or interfaces were misconfigured, demonstrating how routers handle missing paths and non-responsive hosts.

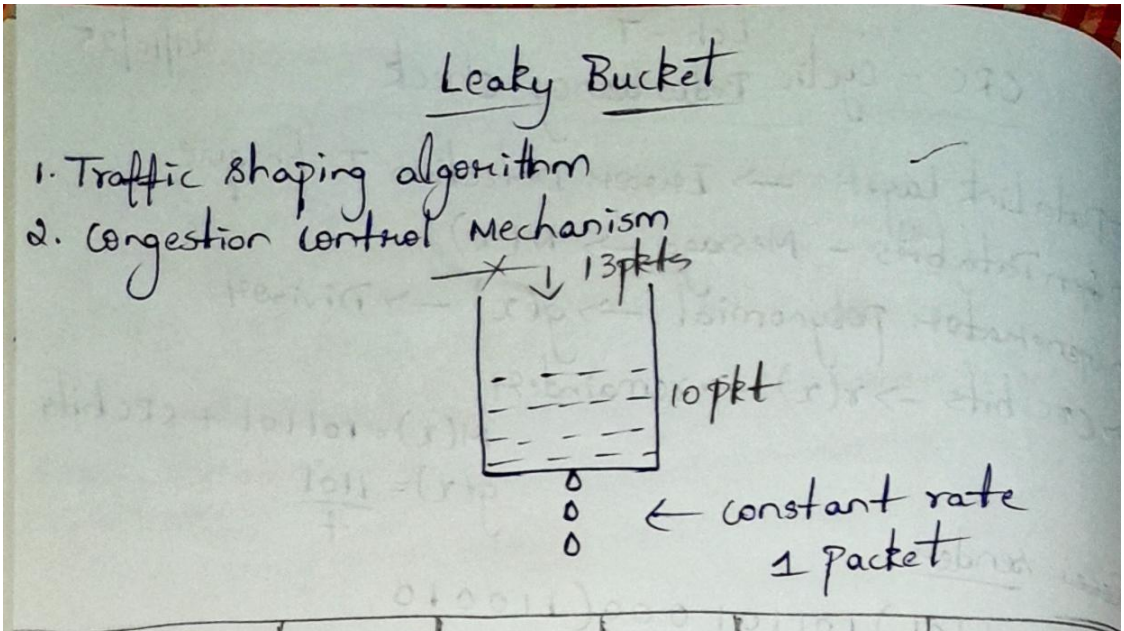
CYCLE 2:

Program 13

Aim of the program:

Write a program for congestion control using Leaky bucket algorithm

Procedure and topology:



Time (t)	Buffer Before	Arrival	Buffer-arival capture	Drops	Depature	Buffer-end
0	0	3	3	0	1	2
1	2	2	4	0	1	3
2	3	1	4	0	1	3
3	3	4	7	3	1	2
4	3	0	3	0	1	2
5	2	0	2	0	1	1
6	1	0	1	0	1	0

LEAKY BUCKET

```
#include <stdio.h>
int main(){
  int n, rate, capacity;
  printf("Enter the number of units: ");
  scanf("%d",&n);
  printf("Enter constant output rate :");
  scanf("%d", &capacity);
  int arrival[n];
  for(int i=0; i<n; i++){
    printf("Enter number of packets arriving at time %d:",
          i+1);
    scanf("%d", &arrival[i]);
  }
  int buffer=0;
  printf("\n ------\n");
  printf("time | buffer_before | arrival | buffer_arrival_captured
        drops | departure | buffer_end\n");
  printf("\n ------\n");
  for(int t=0; t<n; t++){
    int buffer_before = buffer;
    int drops =0;
    int departure=0;
    buffer += arrival[t];
    if(buffer > capacity){
      drops = buffer - capacity;
```

```
            buffer = capacity;
        }
    int buffer_arrival_capture = buffer;
    if (buffer >= rate)
        departure = rate;
    else departure = buffer;
    printf("%2d | %4d | %3d | %4d | %3d | %3d | %4d \n", drops,
t+1, buffer_before, arrival[t], buffer_arrival_capture,
    }
    printf(" ------- \n");
    int t = 0;
    while (buffer > 0){
        int buffer_before = buffer;
        int departure = (buffer >= rate) ? rate : buffer;
        buffer -= departure;
        printf("%2d | %4d | %3d | %4d | %3d | %3d | %4d \n",
            ++t, buffer_before, 0, buffer_before, 0, departure, buffer);
    }
    printf(" ----- \n");
    printf("simulation complete\n");
    return 0;
}
```

OUTPUT

```
Enter number of time units : 7
Enter constant output rate : 1
Enter bucket capacity : 10
Enter number of packets arriving at time 1 : 3
                                       2 : 2
                                       3 : 1
                                       4 : 4
                                       5 : 0
                                       6 : 0
                                       7 : 0
```

time	buffer_before	arrival	buffer_after capture	drops	departure	buffer_end
1	0	3	3	0	1	2
2	2	2	4	0	1	3
3	3	1	4	0	1	3
4	3	4	7	0	1	6
5	6	0	6	0	1	5
6	5	0	5	0	1	4
7	4	0	4	0	1	3
8	3	0	3	0	1	2
9	2	0	2	0	1	1
10	1	0	1	0	1	0

Simulation Complete.

Screenshots/ Output:

```
Output
Enter bucket size: 4
Enter outgoing size: 1
Enter number of inputs: 7
Enter the incoming packet size: 3
Bucket buffer size 3 out of 4
After outgoing 2 packets left out of 4 in buffer
Enter the incoming packet size: 2
Bucket buffer size 4 out of 4
After outgoing 3 packets left out of 4 in buffer
Enter the incoming packet size: 1
Bucket buffer size 4 out of 4
After outgoing 3 packets left out of 4 in buffer
Enter the incoming packet size: 4
Dropped 3 packets
Bucket buffer size 4 out of 4
After outgoing 3 packets left out of 4 in buffer
Enter the incoming packet size: 0
Bucket buffer size 3 out of 4
After outgoing 2 packets left out of 4 in buffer
Enter the incoming packet size: 0
Bucket buffer size 2 out of 4
After outgoing 1 packets left out of 4 in buffer
Enter the incoming packet size: 0
Bucket buffer size 1 out of 4
After outgoing 0 packets left out of 4 in buffer
```

Observation:

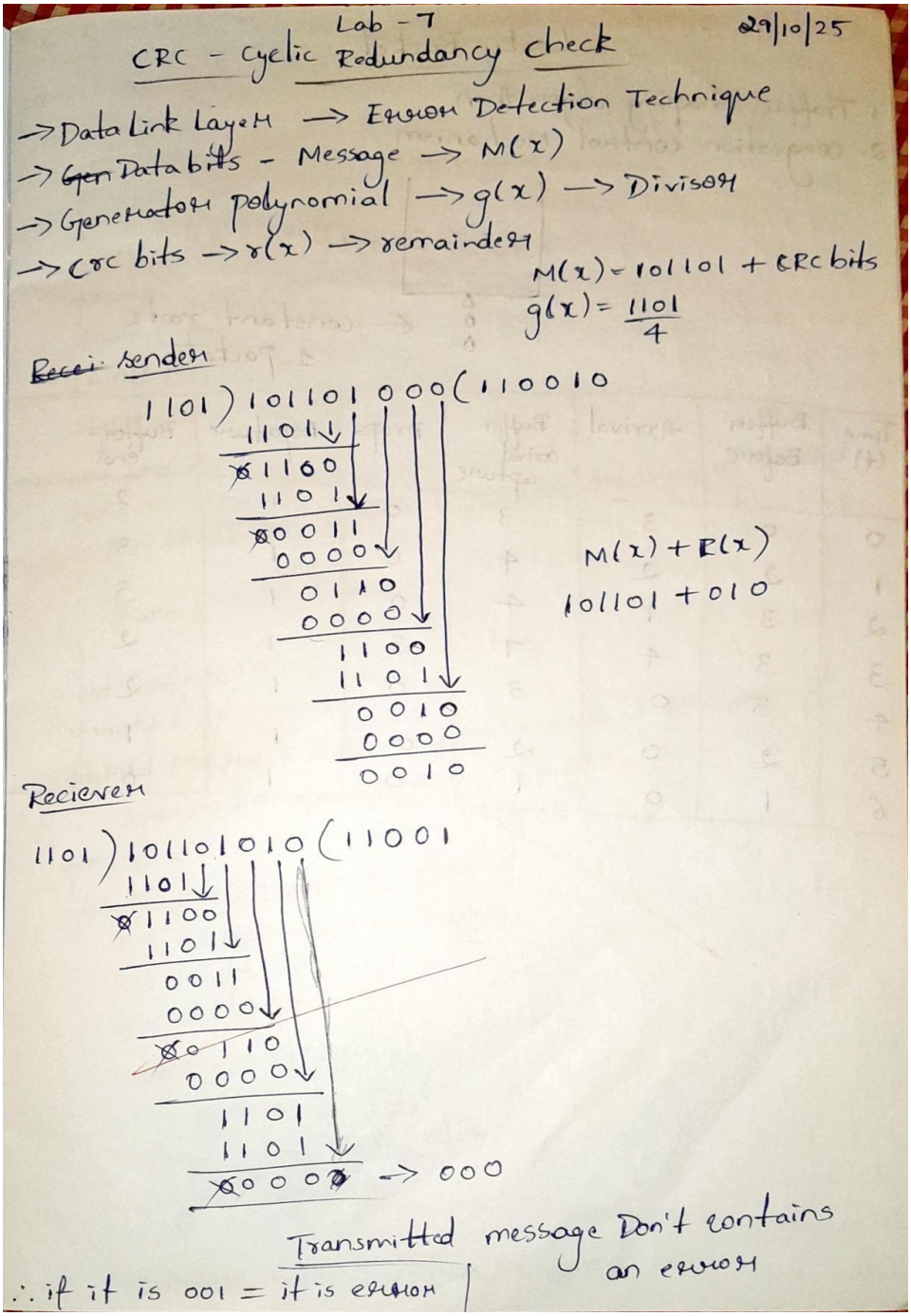
- The leaky bucket mechanism regulated the outgoing packet flow at a constant rate, preventing sudden traffic bursts from overwhelming the network.
- Packets exceeding the bucket capacity were dropped, demonstrating effective congestion control by smoothing traffic and enforcing rate-limiting.

Program 14

Aim of the program:

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

Procedure and topology:



CRC - Cyclic Redundancy and check

```
#include <stdio.h>
int main(){
  int n, rate, capacity;
  printf("Enter number of time units: ");
  scanf("%d", &n);
  printf("Enter constant output rate: ");
  scanf("%d", &rate);
  printf("Enter bucket capacity: ");
  scanf("%d", &capacity);

  int arrival[n];
  for(int i=0; i<n; i++){
    printf("Enter number of packets arriving at time %d: ",
           i+1);
    scanf("%d", &arrival[i]);
  }
  int buffer=0;
  printf("\n ---------- \n");
  printf("time | buffer_before | arrival | buffer_arrival_
    capture | drops | departure | buffer_end \n");
  printf("\n ---------- \n");
  for(int t=0; t<n; t++){
    int buffer_before = buffer;
    int drops = 0;
    int departure = 0;
    buffer += arrival[t];
    if(buffer > capacity){
      drops = buffer - capacity;
      buffer = capacity;
    }
    int buffer_arrival_capture = buffer;
```

```
    if (buffer >= rate)
      departure = rate;
    else
      departure = buffer;
    buffer -= departure;
    int buffer_end = buffer;
    printf("%2d | %4d | %3d | %4d | %3d | %3d | %4d\n",
    t+1, buffer_before, arrival[t], buffer_arrival_capture,
      drops, departure, buffer_end);
  }
  printf("- - - -\n");
  int t=n;
  while (buffer > 0){
    int buffer_before = buffer;
    int departure = (buffer >= rate) ? rate : buffer;
    buffer -= departure;
    printf("%2d | %4d | %3d | %4d | %3d | %3d | %4d\n",
  ++t, buffer_before, 0, departure, buffer);
  }
  printf("- - - -\n");
  printf("Simulation complete.\n");
  return 0;
}
```

OUTPUT

Enter data to be transmitted : 101010

Enter the divisor polynomial : 1011

Data padded with n-1 zeroes : 101010000

CRC value is : 001

Final codeword to be sent : 101010001

Enter the received data : 101010001

Screenshots/ Output:

```
Output

Enter message bits: 101100
Enter polynomial g(x): 1001

Padded data (Message + zeros): 101100000
CRC bits (remainder): 001
Transmitted message: 101100001

Enter received bits: 101100001

No Error detected. Message OK.


=== Code Execution Successful ===
```

Observation:

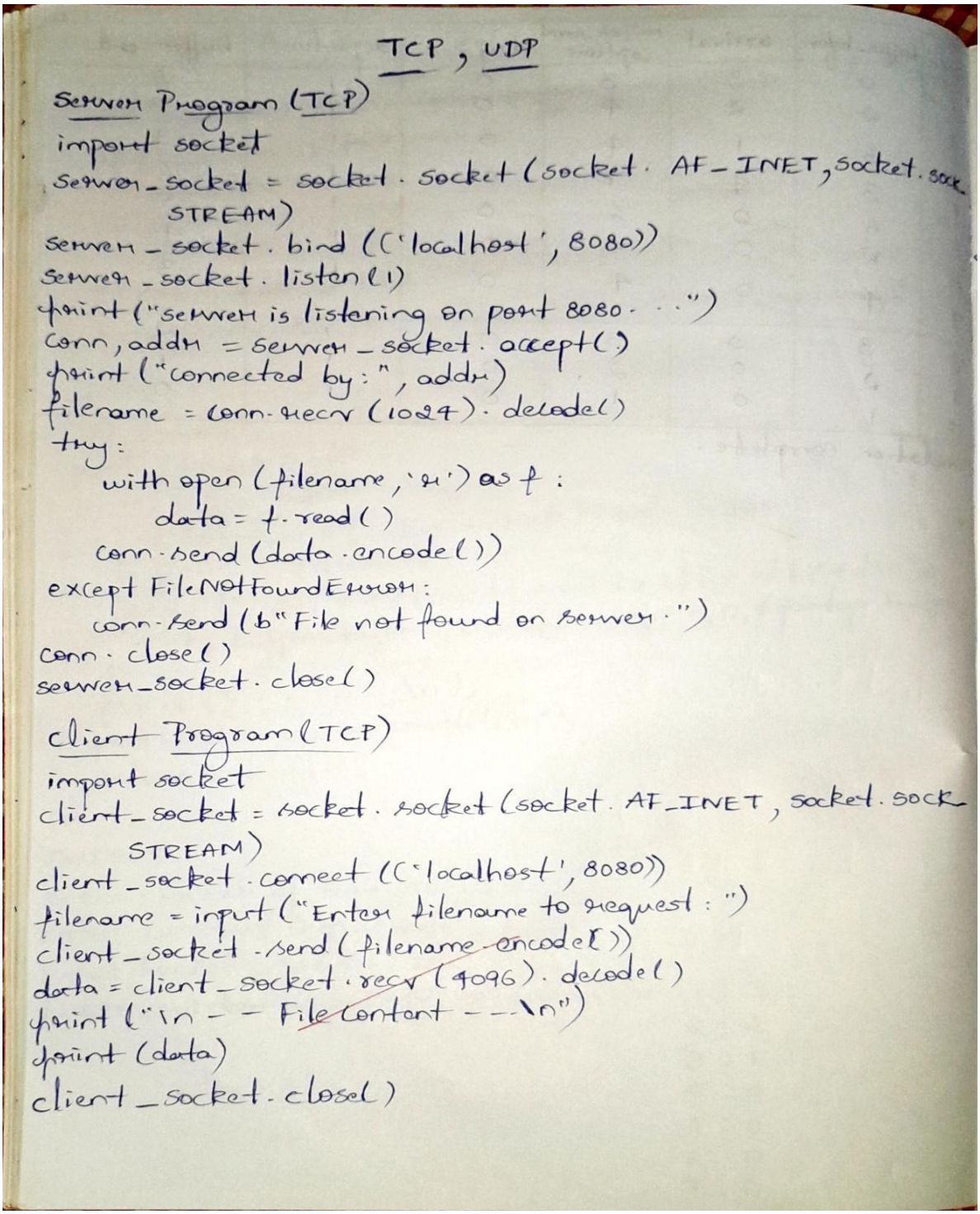
- The CRC-CCITT (16-bit) algorithm correctly generated a checksum for the transmitted data, ensuring reliable detection of single-bit and burst errors.
- Intentional error tests produced mismatched CRC values at the receiver, confirming accurate error detection through polynomial division.

Program 15

Aim of the program:

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.

Procedure and topology:



TCP, UDP

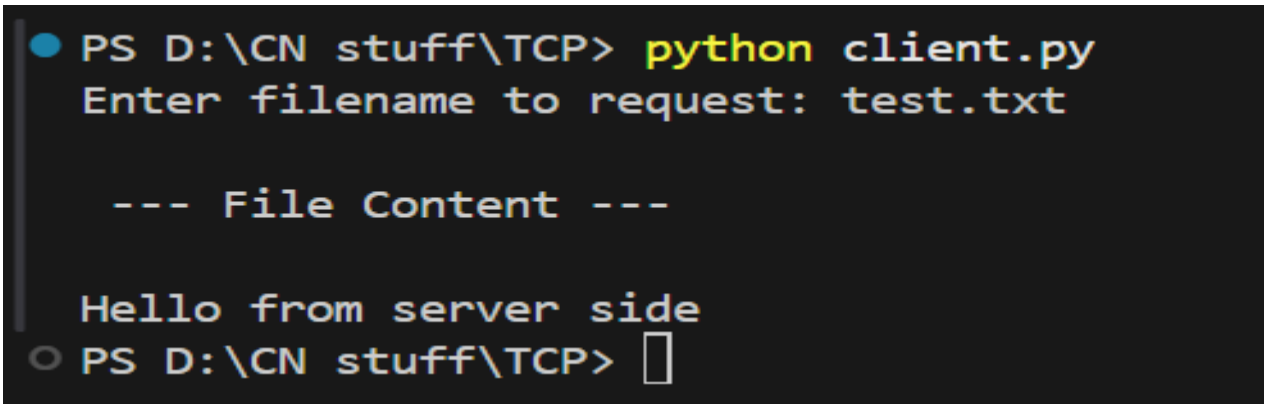
Server Program (TCP)

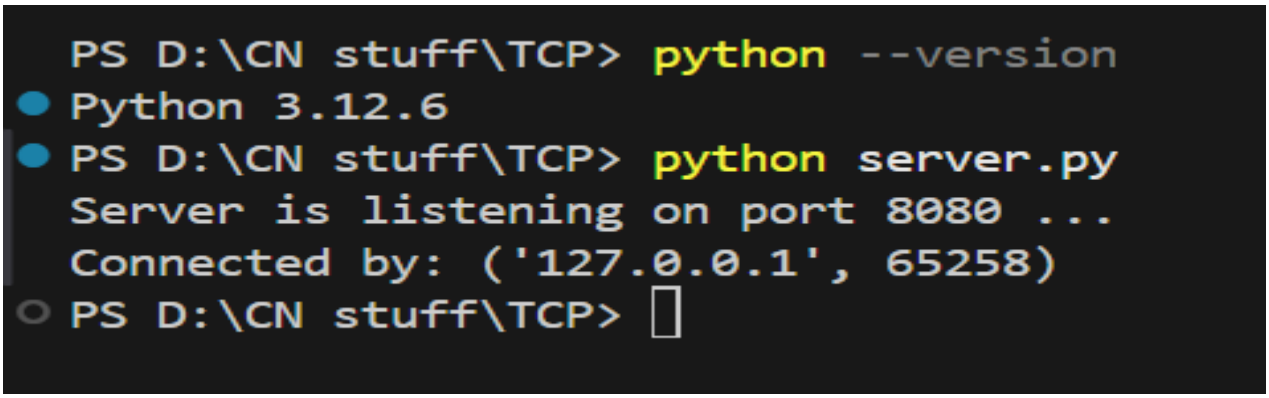
```
import socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_
        STREAM)
server_socket.bind(('localhost', 8080))
server_socket.listen(1)
print("server is listening on port 8080...")
conn, addr = server_socket.accept()
print("connected by:", addr)
filename = conn.recv(1024).decode()
try:
    with open(filename, 'r') as f:
        data = f.read()
    conn.send(data.encode())
except FileNotFoundError:
    conn.send(b"File not found on server.")
conn.close()
server_socket.close()
```

Client Program (TCP)

```
import socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_
        STREAM)
client_socket.connect(('localhost', 8080))
filename = input("Enter filename to request:")
client_socket.send(filename.encode())
data = client_socket.recv(4096).decode()
print("\n--- File Content ---\n")
print(data)
client_socket.close()
```

Screenshots/ Output:





Observation:

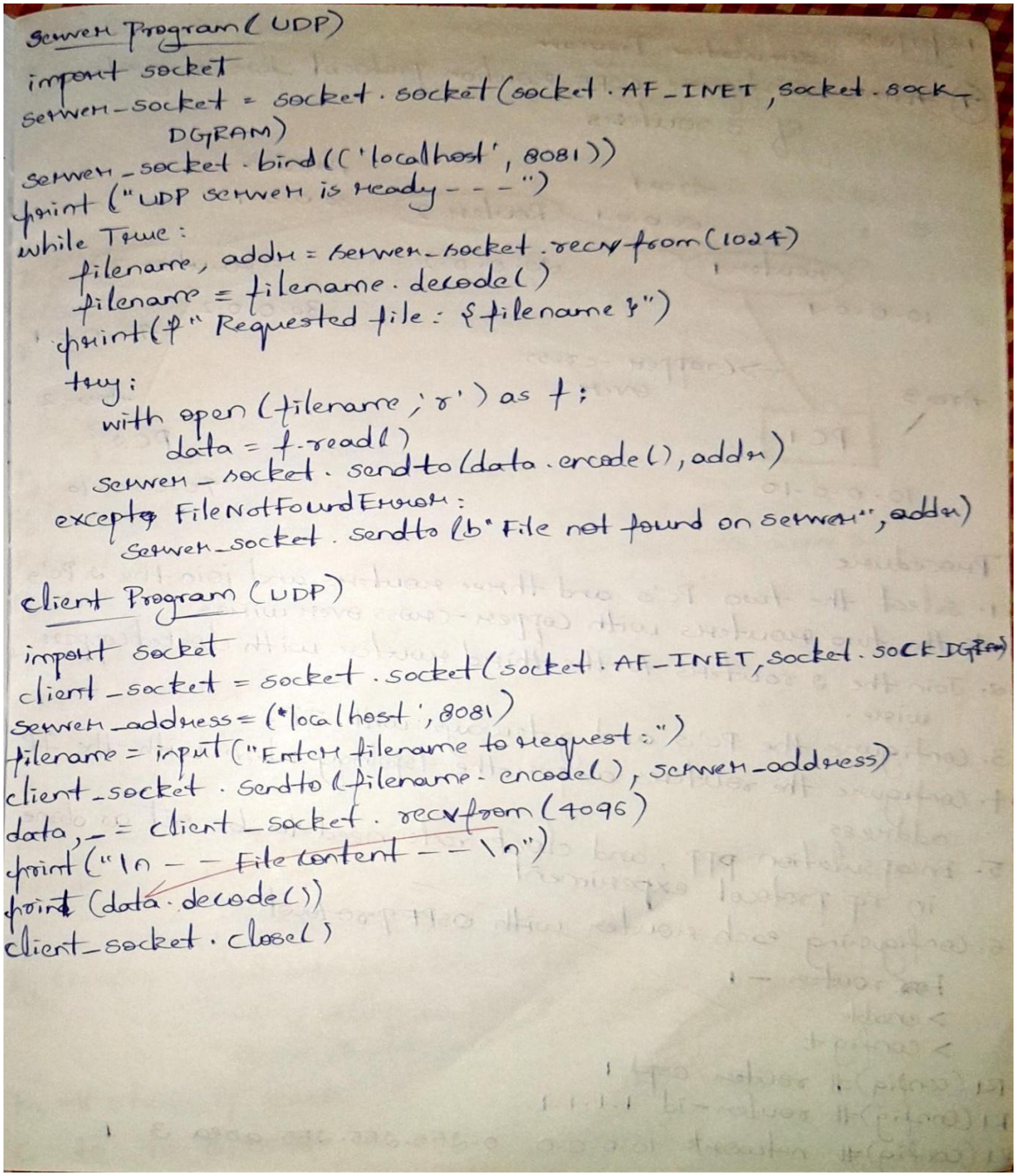
- The TCP client successfully established a reliable connection with the server and transmitted the requested filename using stream-oriented communication.
- The server correctly retrieved and returned the file contents over the same TCP session, demonstrating reliable data transfer, acknowledgment handling, and error-free delivery.

Program 16

Aim of the program:

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.

Procedure and topology:



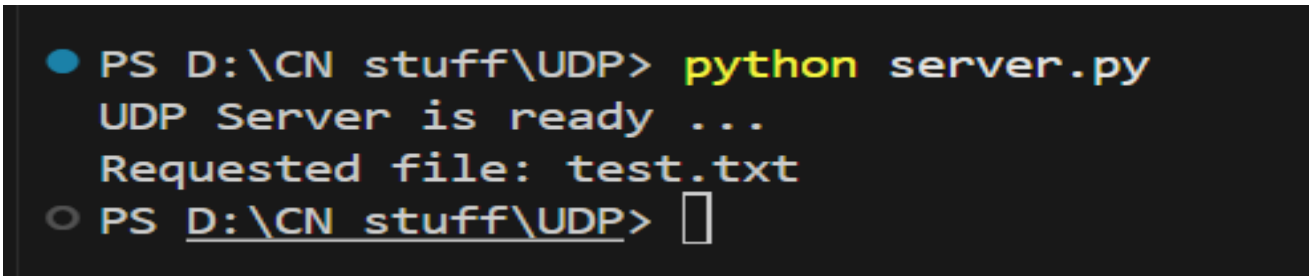
Server Program (UDP)

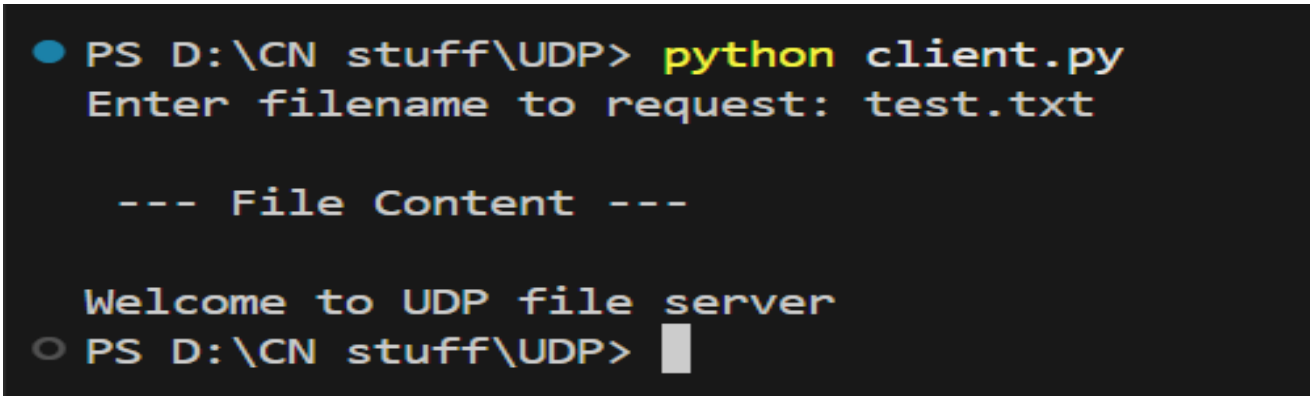
```
import socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_
        DGRAM)
server_socket.bind(('localhost', 8081))
print("UDP server is ready---")
while True:
    filename, addr = server_socket.recvfrom(1024)
    filename = filename.decode()
    print(f" Requested file: {filename}")
    try:
        with open(filename, 'r') as f:
            data = f.read()
        server_socket.sendto(data.encode(), addr)
    except FileNotFoundError:
        server_socket.sendto(b"File not found on server", addr)
```

Client Program (UDP)

```
import socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_address = ('localhost', 8081)
filename = input("Enter filename to request:")
client_socket.sendto(filename.encode(), server_address)
data, _ = client_socket.recvfrom(4096)
print("\n--File content--\n")
print(data.decode())
client_socket.close()
```

Screenshots/ Output:





Observation:

- The UDP client successfully sent the filename as a connectionless datagram, demonstrating non-reliable, low-overhead message transfer without session establishment.
- The server responded with the file contents using UDP packets, and correct reception validated functional data exchange despite the absence of acknowledgment and retransmission mechanisms.