VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

Dyuthi (1BM23CS097)

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 2025-26 (even)

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 **Dyuthi (1BM23CS097),** 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.

Sarala D V Assistant Professor Department of CSE, BMSCE	Dr. Selva Kumar Professor & HOD Department of CSE, BMSCE

Index

Sl. No.	Date	Experiment Title	Page No.
1	13-08-2025	Simple PDU from source to destination using hub and switch as connecting devices.	4-8
2	17-09-2025	Default route and static route to the Router	9-13
3	03-09-2025	DHCP within a LAN and outside LAN	14-15
4	10-09-2025	Web Server, DNS within a LAN	16-18
5	08-10-2025	Operation of TELNET to access the router in server room from a PC in IT office	19-21
6	19-11-2025	RIP routing Protocol in Routers	22-23
7	15-10-2025	VLAN to make the PCs communicate among a VLAN	24-25
8	15-10-2025	WLAN to make the nodes communicate wirelessly	26-27
9	19-11-2025	Simple LAN to understand the concept and operation of ARP	28-29
10	08-10-2025	OSPF routing protocol	30-31
11	15-10-2025	TTL/ Life of a Packet	32-33
12	15-10-2026	Ping responses, destination unreachable, request timed out, reply	34-36
13	29-10-2025	Congestion control using Leaky bucket algorithm	37-38
14	29-10-2025	Error detecting code using CRC-CCITT	39-40
15	12-11-2025	TCP File Request–Response Using Client–Server Socket Program	41-42
16	12-11-2025	UDP File Request–Response Using Client–Server Socket Program	43-44

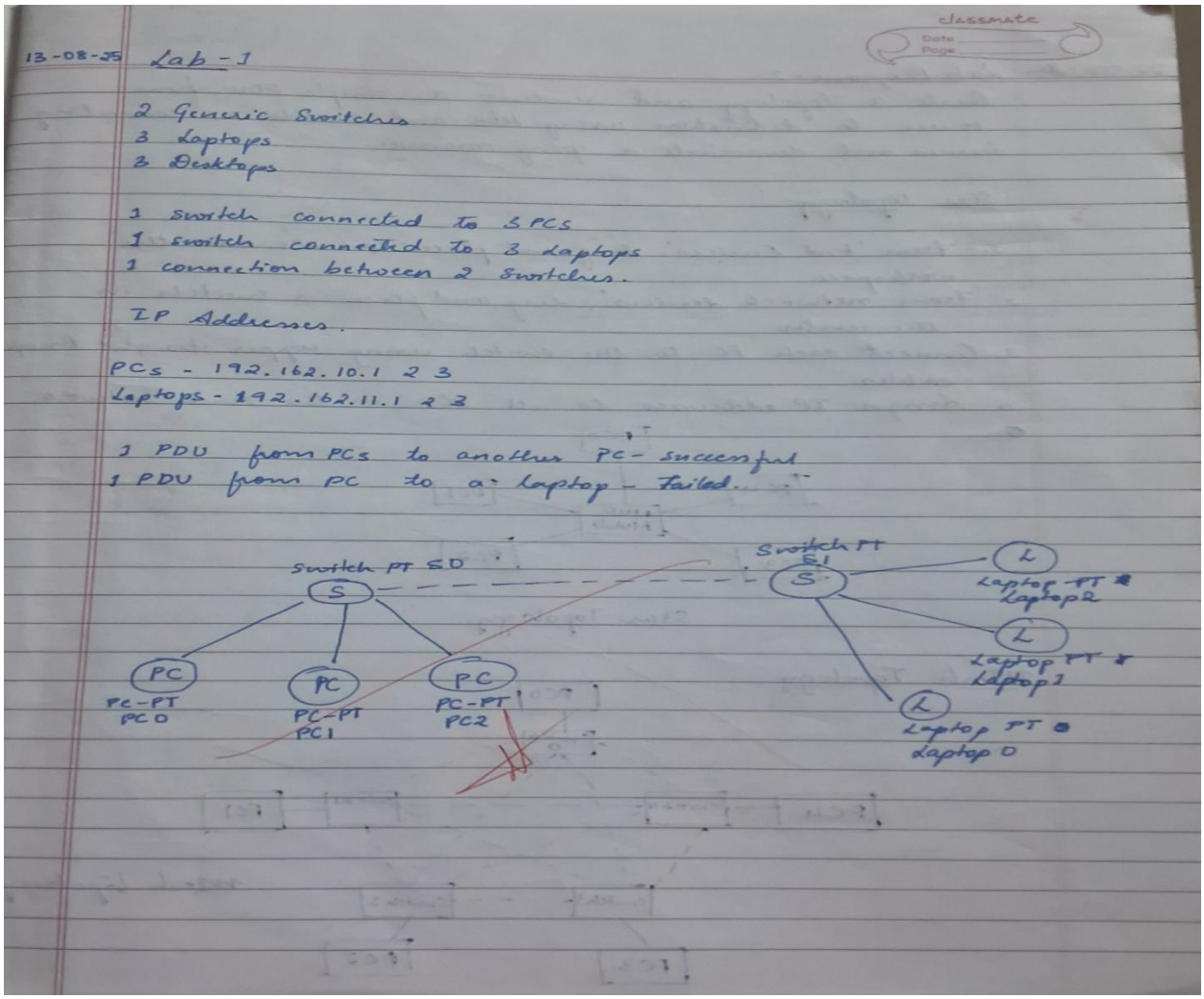
CYCLE 1:

Program 1

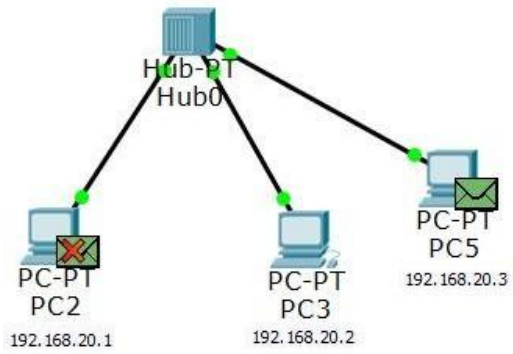
Aim of the program:

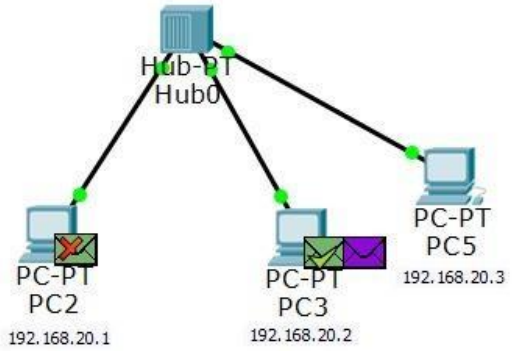
Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

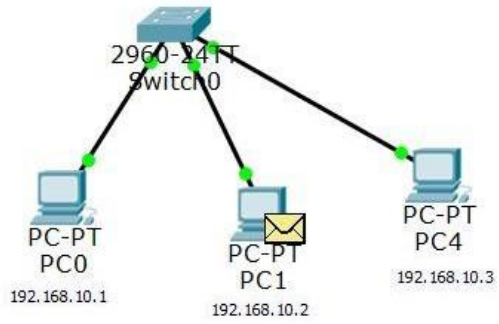
Procedure and topology:

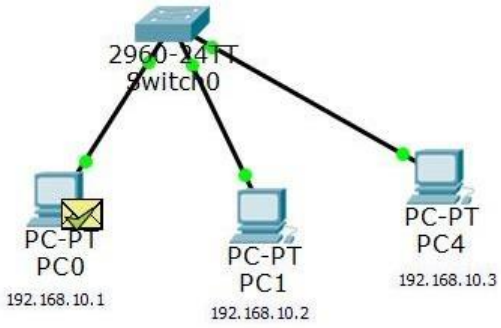


Screenshots/ Output:

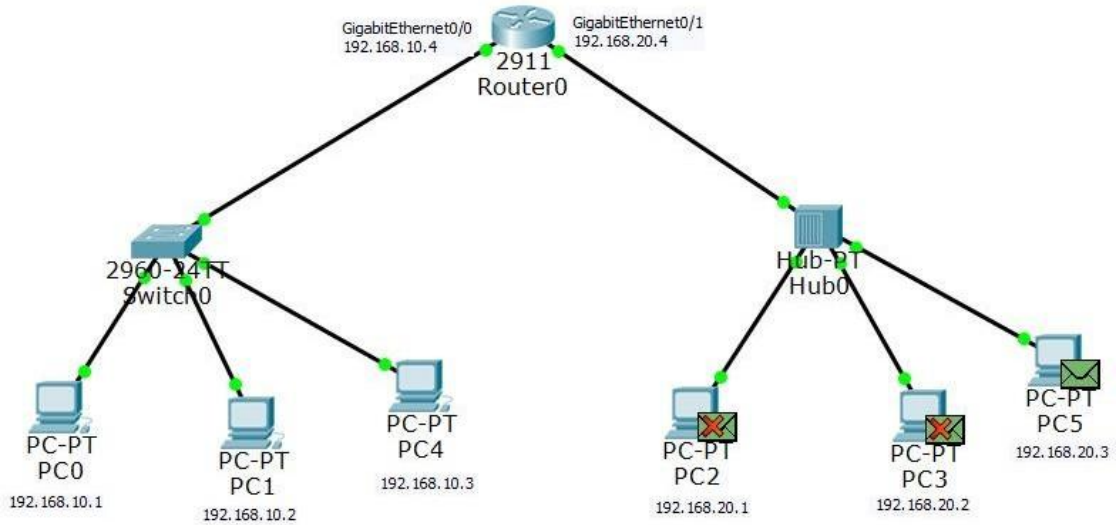


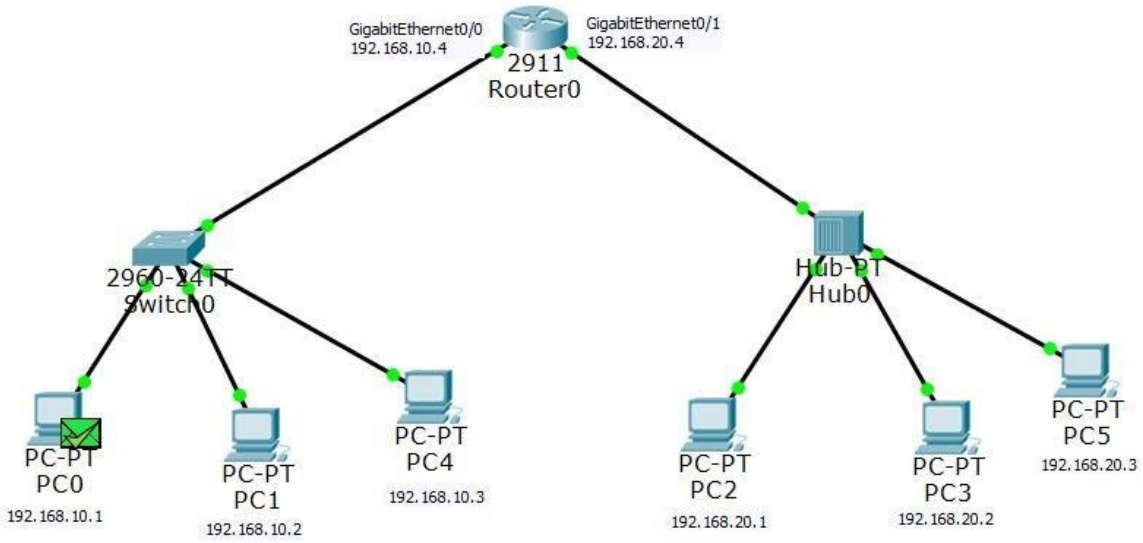






Updated topology





Observation:

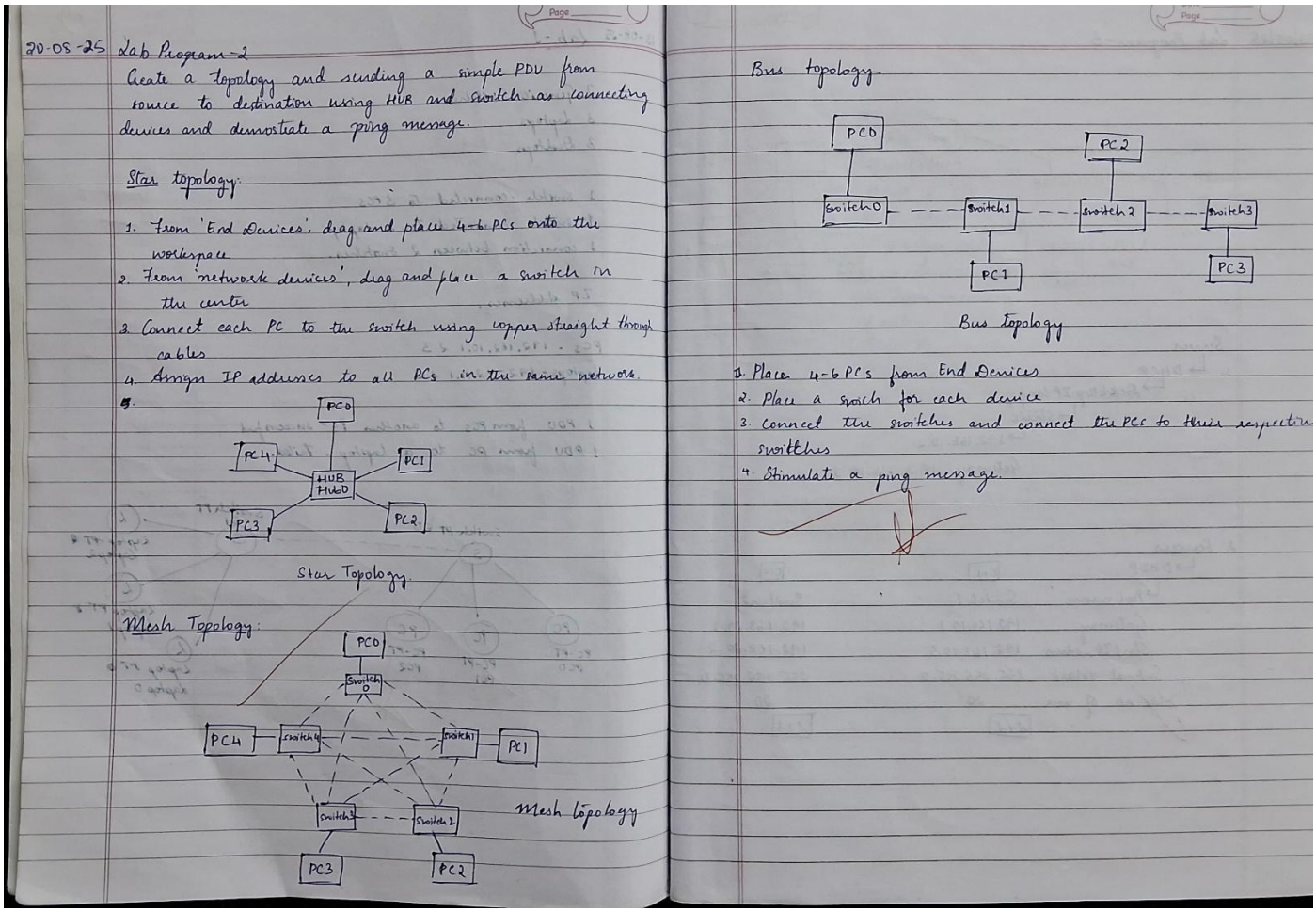
- In the hub-based topology, the PDU was broadcast to all ports, while the switch forwarded the PDU only to the destination MAC after learning addresses from incoming frames.
- Successful ICMP echo and echo-reply messages confirm that both devices enabled connectivity, with the switch demonstrating selective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

Program 2

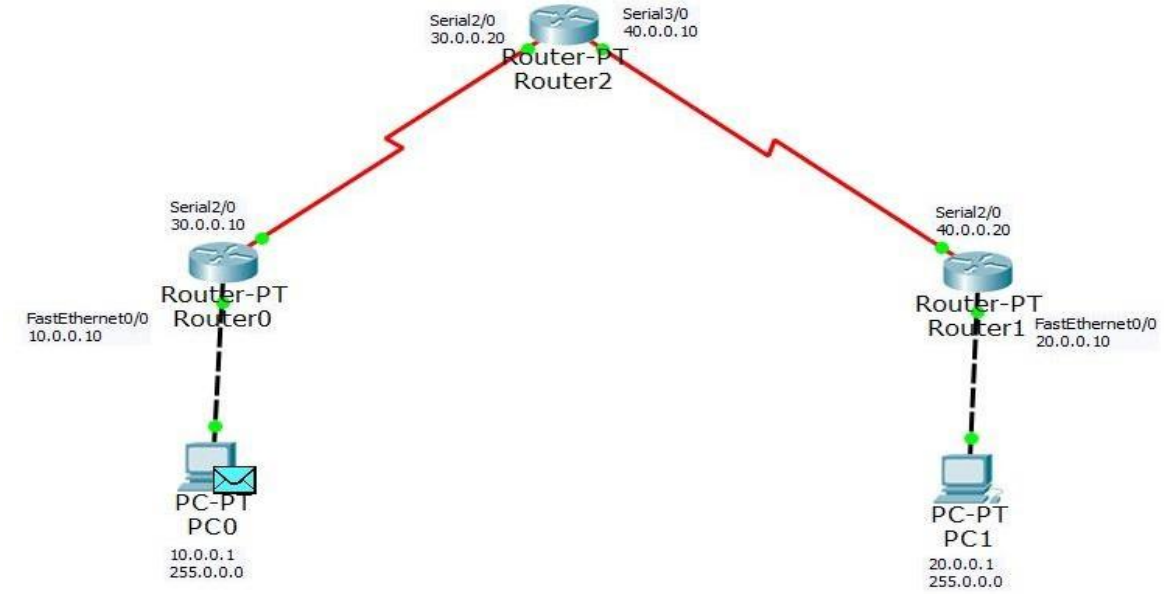
Aim of the program:

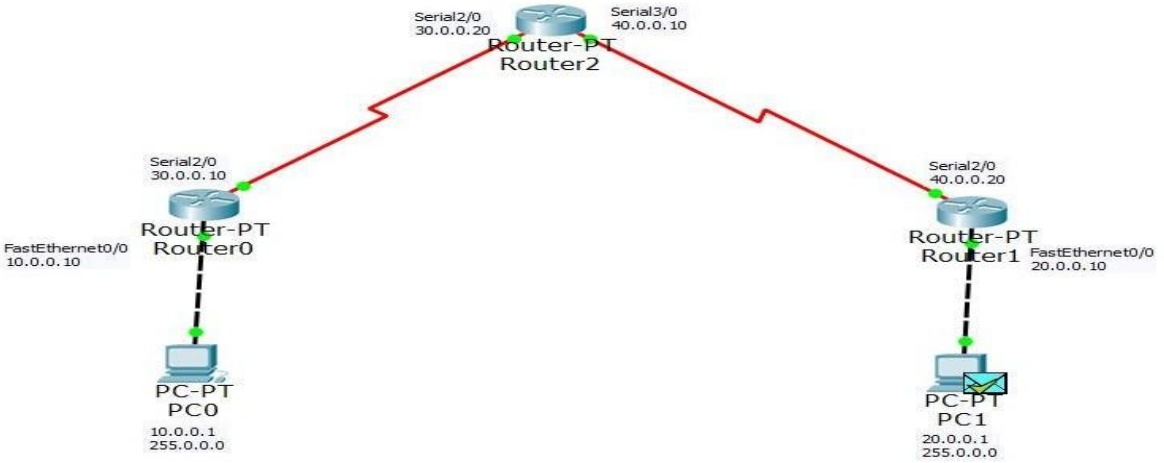
Configure default route, static route to the Router

Procedure and topology:

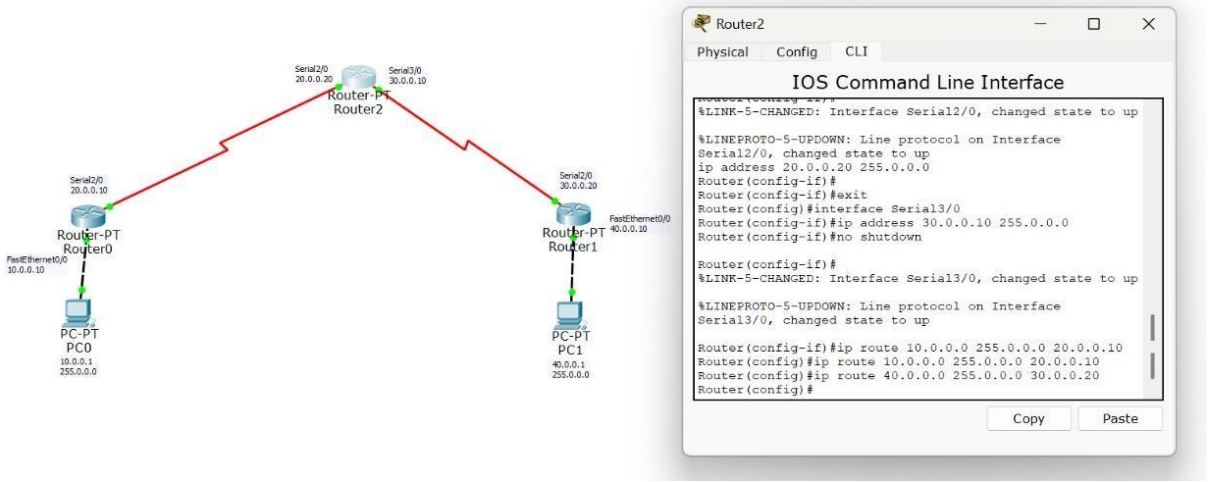


Screenshots/ Output:

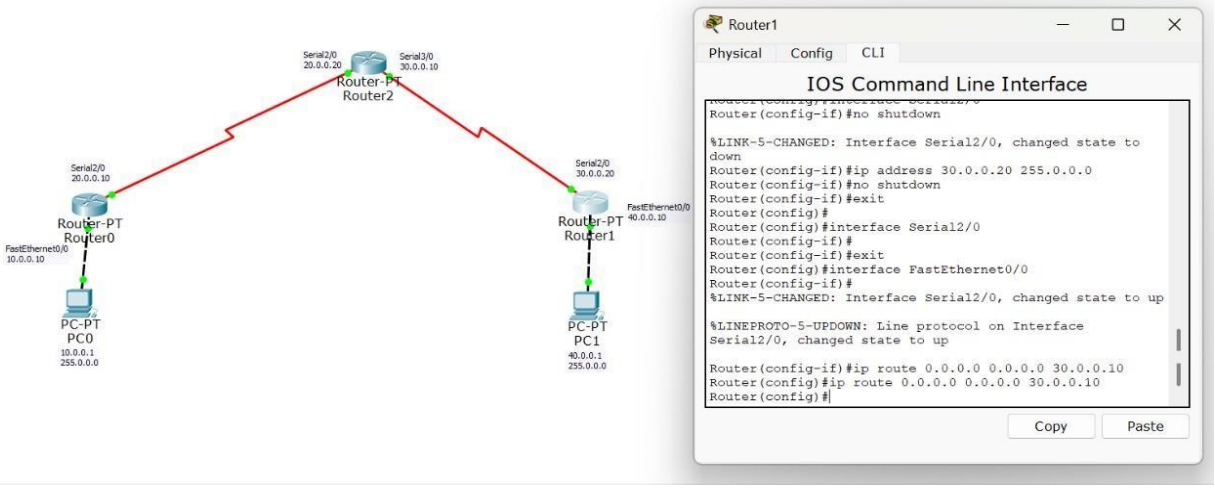




Static routing CLI commands:



Default routing CLI commands:



Observation:

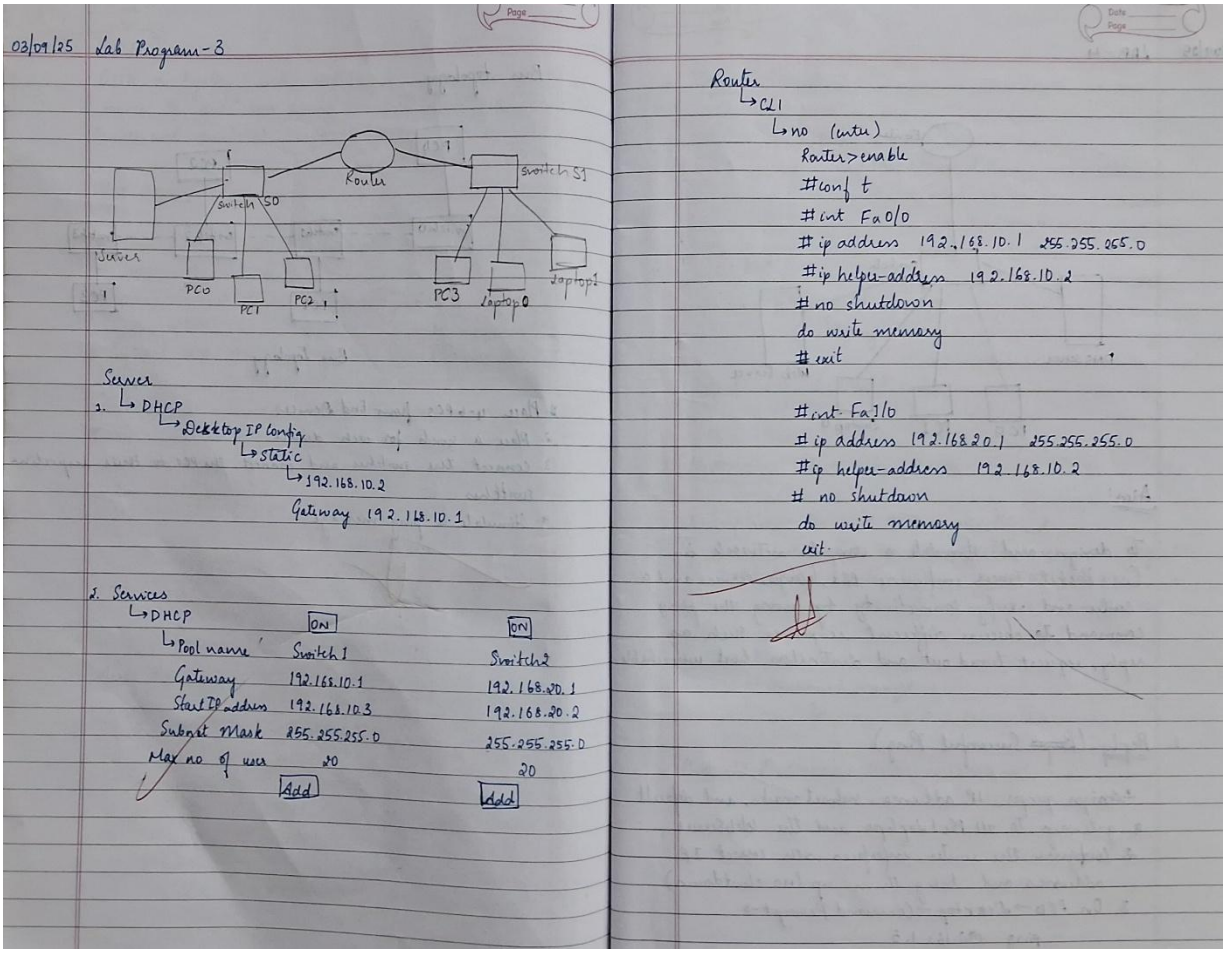
- The configured static and default routes correctly updated the router's routing table, enabling deterministic next-hop selection for remote networks.
- Successful ping tests verified that traffic was forwarded according to the static/default route entries, ensuring end-to-end reachability across different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

Program 3

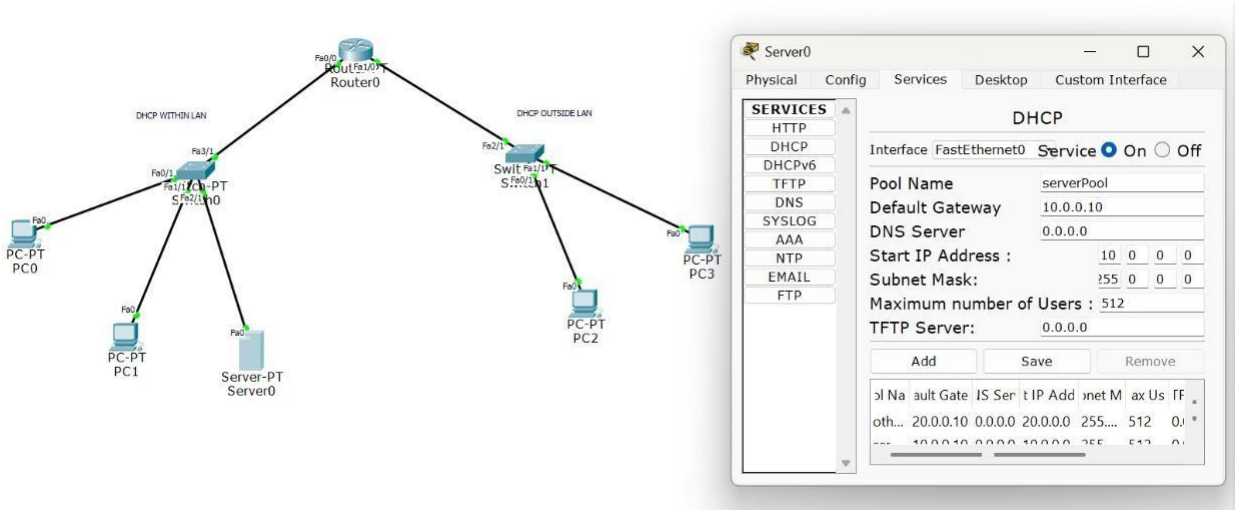
Aim of the program:

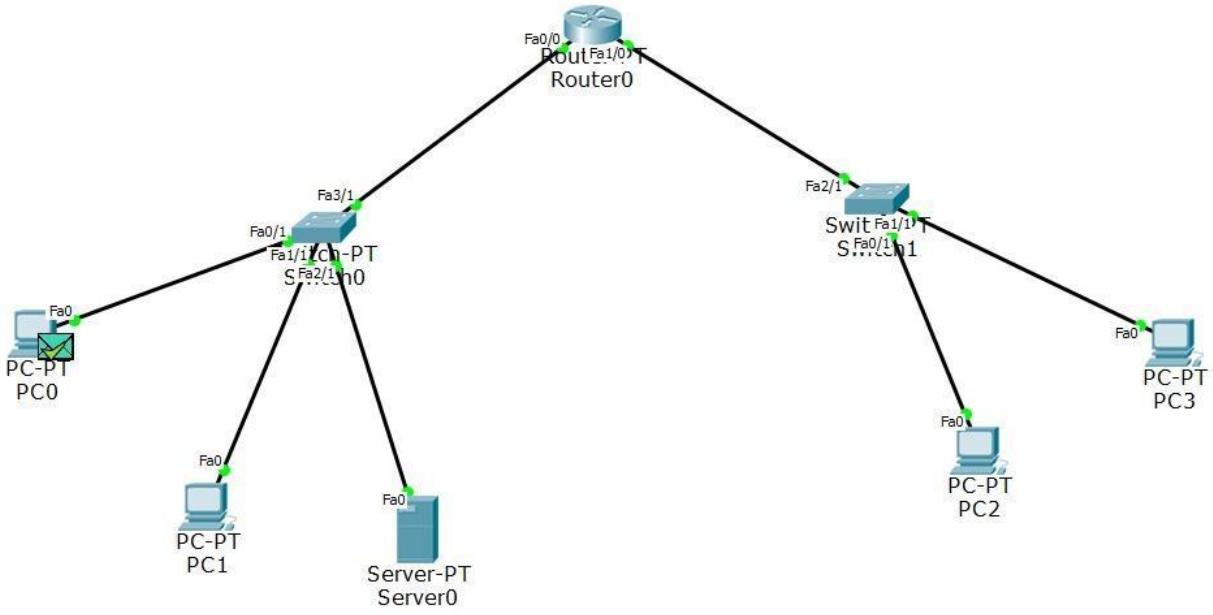
Configure DHCP within a LAN and outside LAN.

Procedure and topology:



Screenshots/ Output:





Observation:

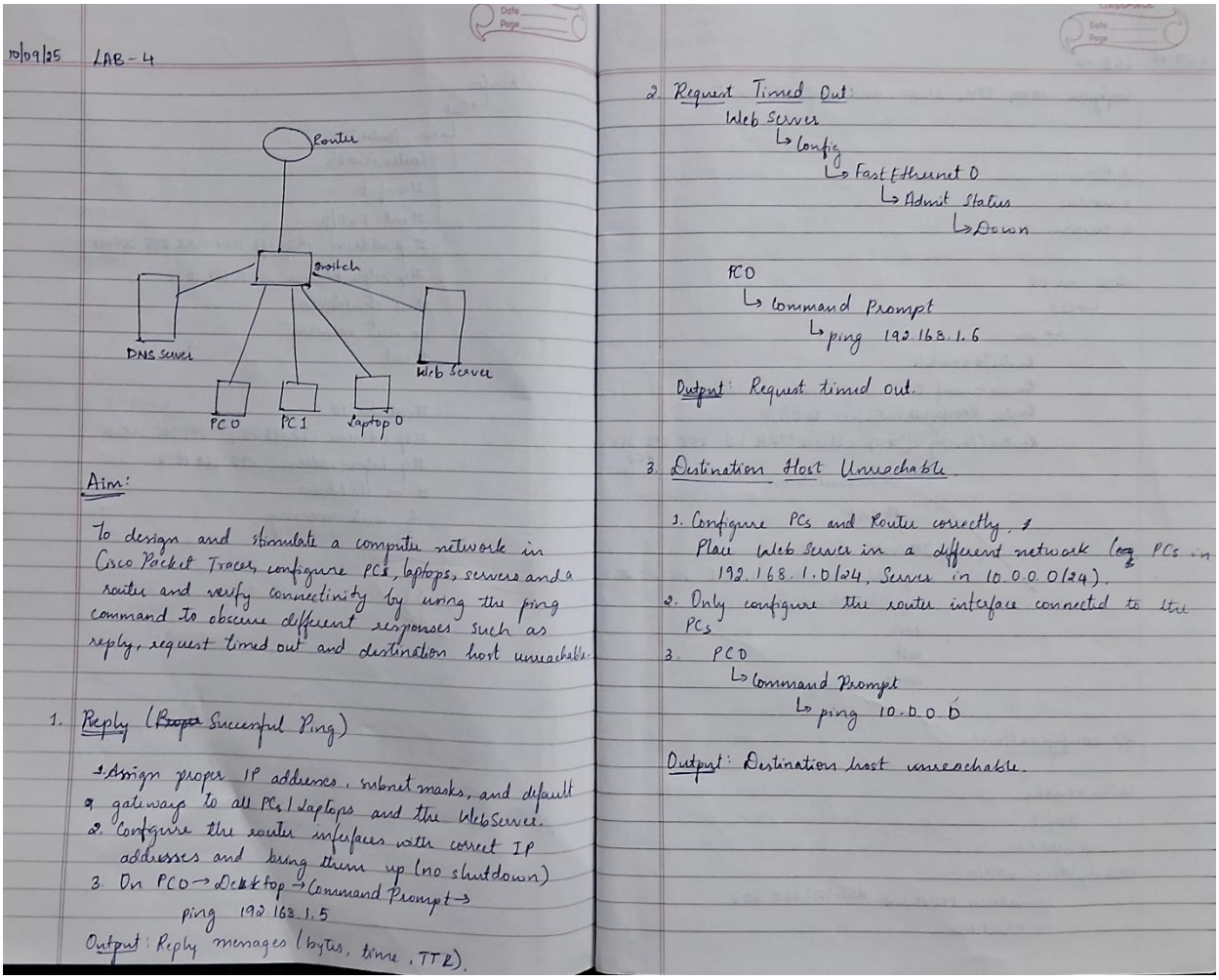
- The DHCP server successfully allocated IP addresses to clients within the LAN, confirming proper scope configuration and automatic distribution of network parameters.
- DHCP relay (IP Helper) enabled clients outside the LAN to obtain leases from the central DHCP server, demonstrating correct inter-network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

Program 4

Aim of the program:

Configure Web Server, DNS within a LAN.

Procedure and topology:



10/09/25 LAB - 4

Aim:

To design and simulate a computer network in Cisco Packet Tracer, configure PCs, laptops, servers and a router and verify connectivity by using the ping command to observe different responses such as reply, request timed out and destination host unreachable.

1. Reply (~~Reqst~~ Successful Ping)

1. Assign proper IP addresses, subnet masks, and default
2. gateways to all PCs / laptops and the Web Server.
2. Configure the router interfaces with correct IP addresses and bring them up (no shutdown)
3. On PC0 → Desktop → Command Prompt →
   ping 192.168.1.5

Output: Reply messages (bytes, time, TTL).

2. Request Timed Out

Web Server
↳ Config
↳ FastEthernet 0
↳ Admin Status
↳ Down

PC0
↳ Command Prompt
↳ ping 192.168.1.6

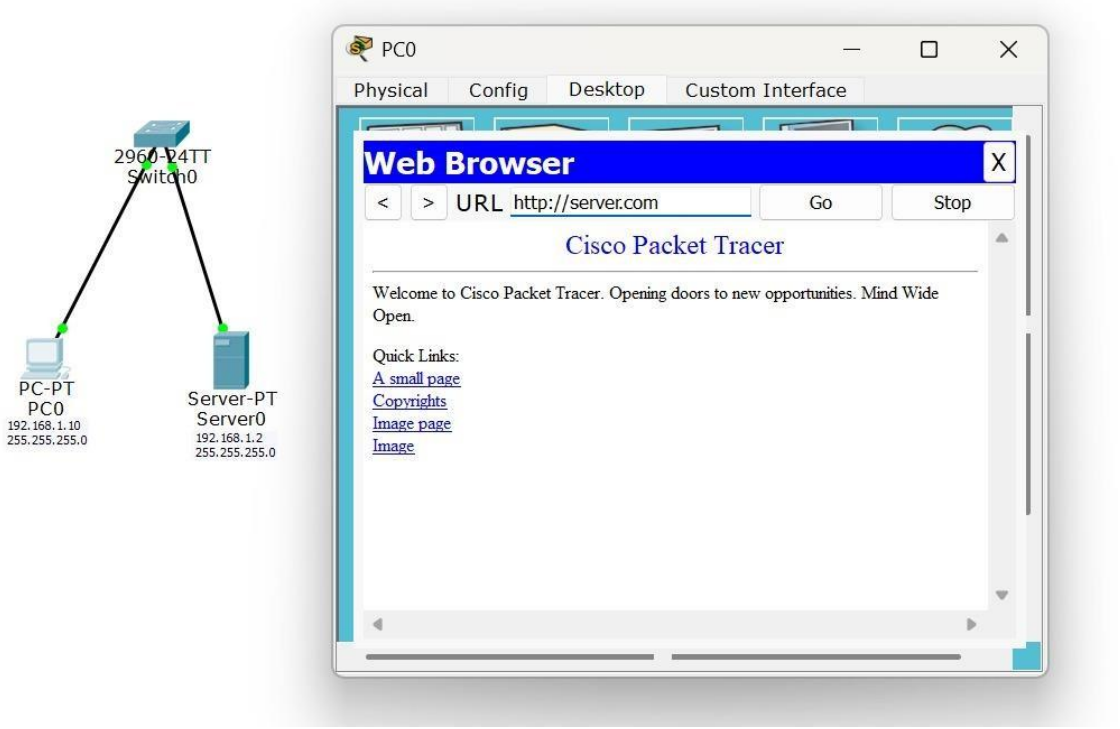
Output: Request timed out.

3. Destination Host Unreachable

1. Configure PCs and Router correctly.
   Place Web Server in a different network (eg PCs in 192.168.1.0/24, Server in 10.0.0.0/24).
2. Only configure the router interface connected to the PCs
3. PC0
   ↳ Command Prompt
   ↳ ping 10.0.0.0

Output: Destination host unreachable.

Screenshots/ Output:



Observation:

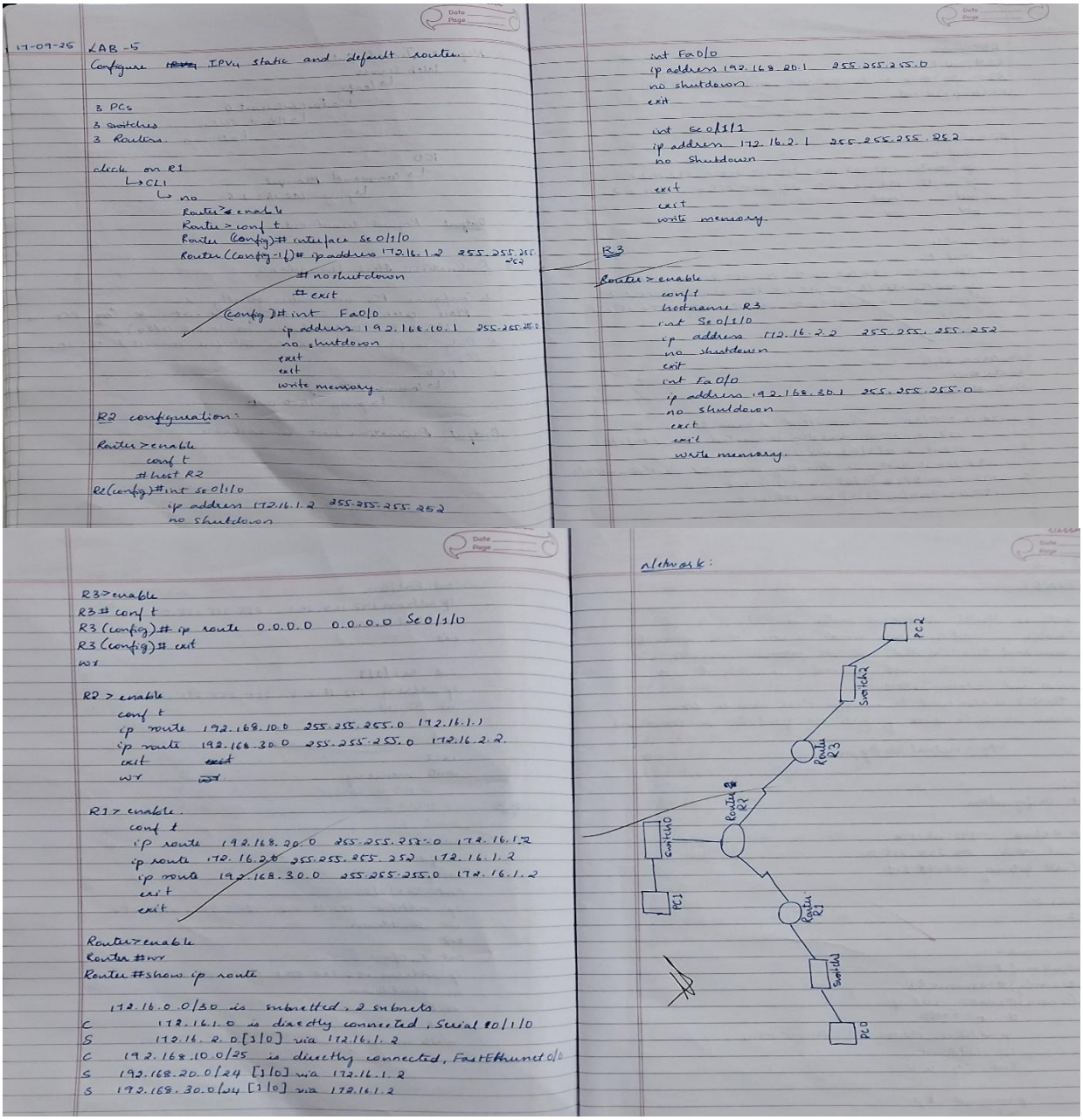
- The DNS server successfully resolved domain names to the corresponding web server's IP address, confirming proper hostname-to-IP mapping within the LAN.
- HTTP requests reached the web server using the DNS-resolved address, validating correct server configuration and internal LAN communication.

Program 5

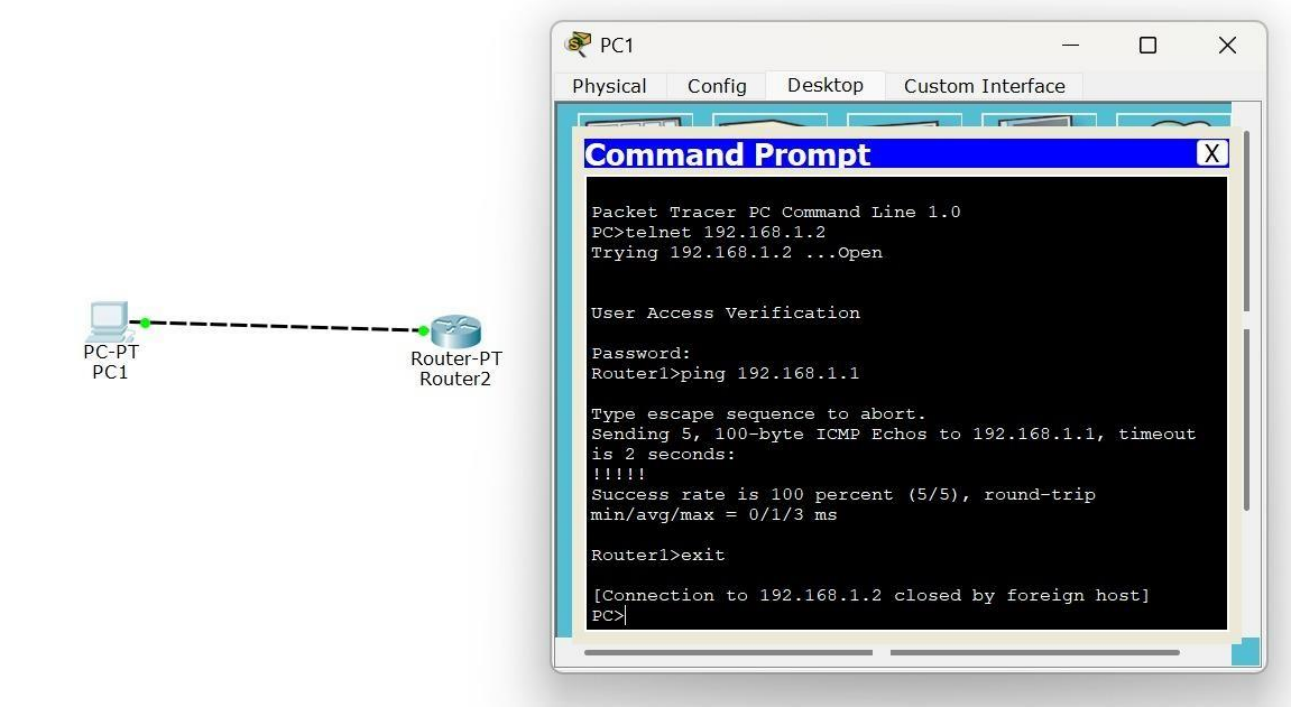
Aim of the program:

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

Procedure and topology:



Screenshots/ Output:



Observation:

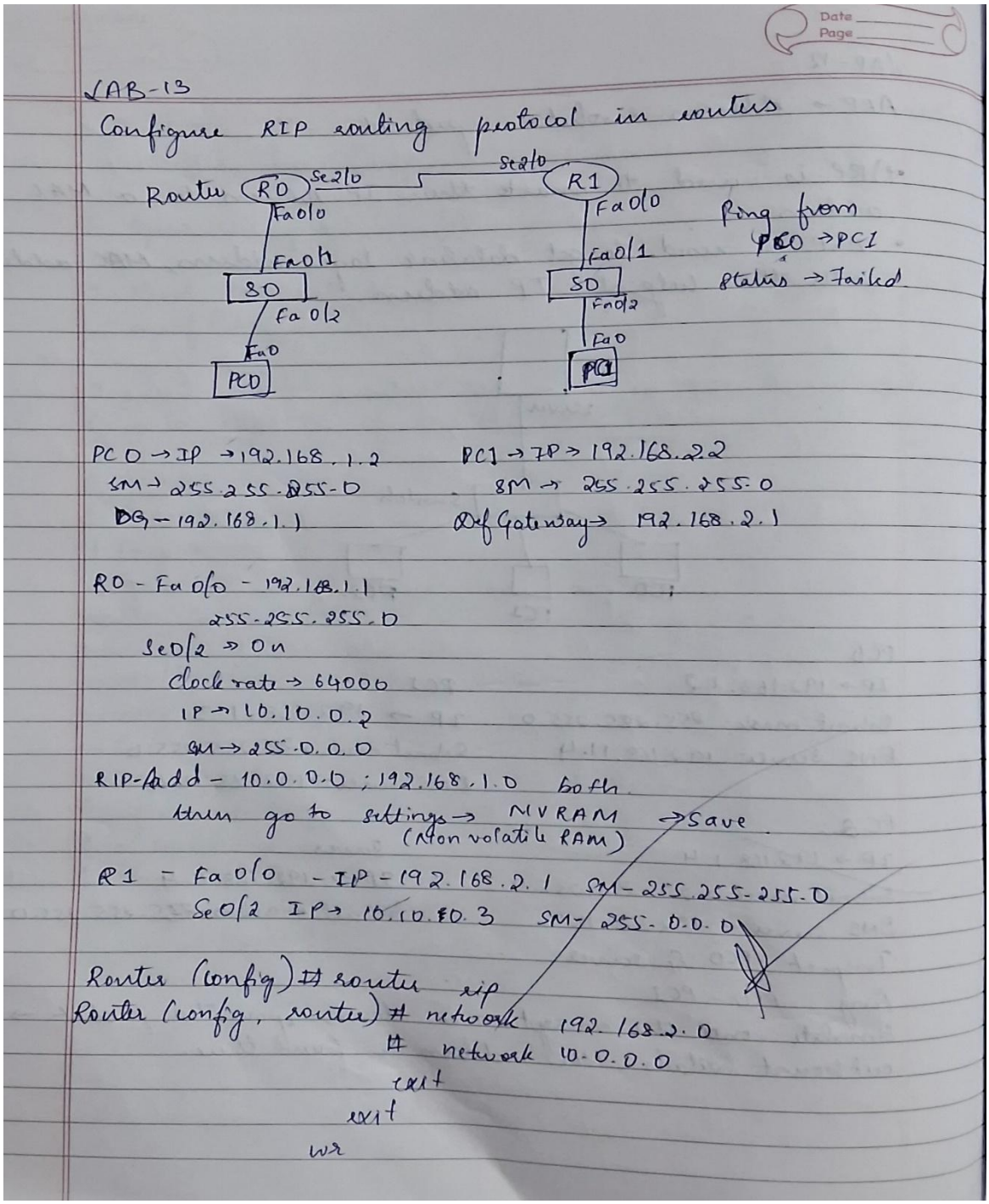
- The Telnet session successfully established a remote CLI connection to the router, confirming proper VTY line configuration and IP reachability between the IT office PC and the server room router.
- Command execution over the Telnet session demonstrated reliable remote device management.

Program 6

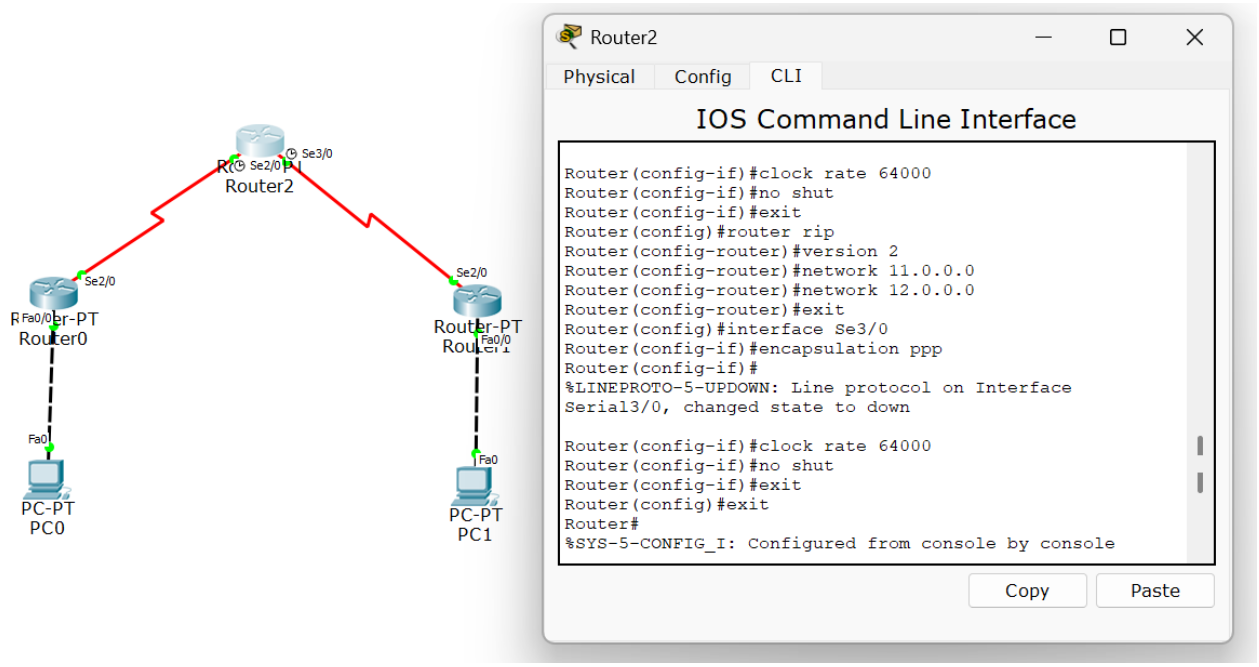
Aim of the program:

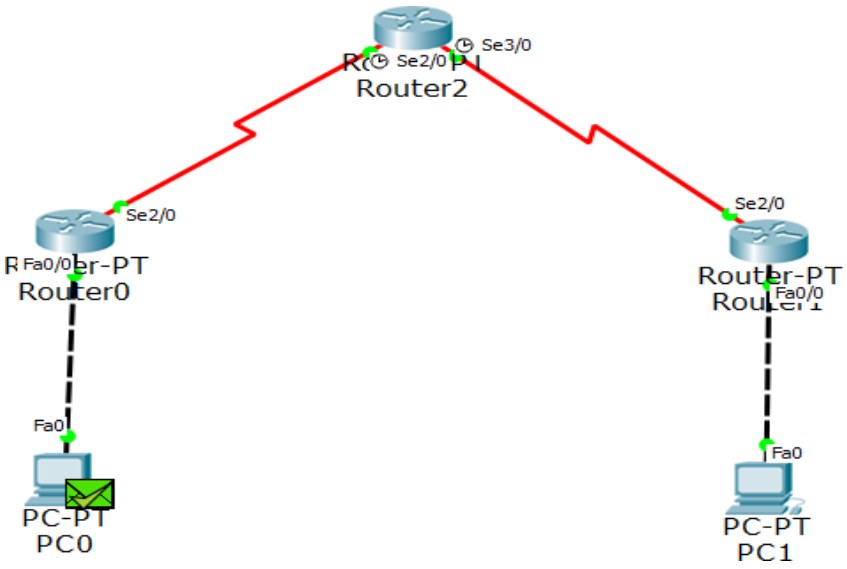
Configure RIP routing Protocol in Routers

Procedure and topology:



Screenshots/ Output:





Observation:

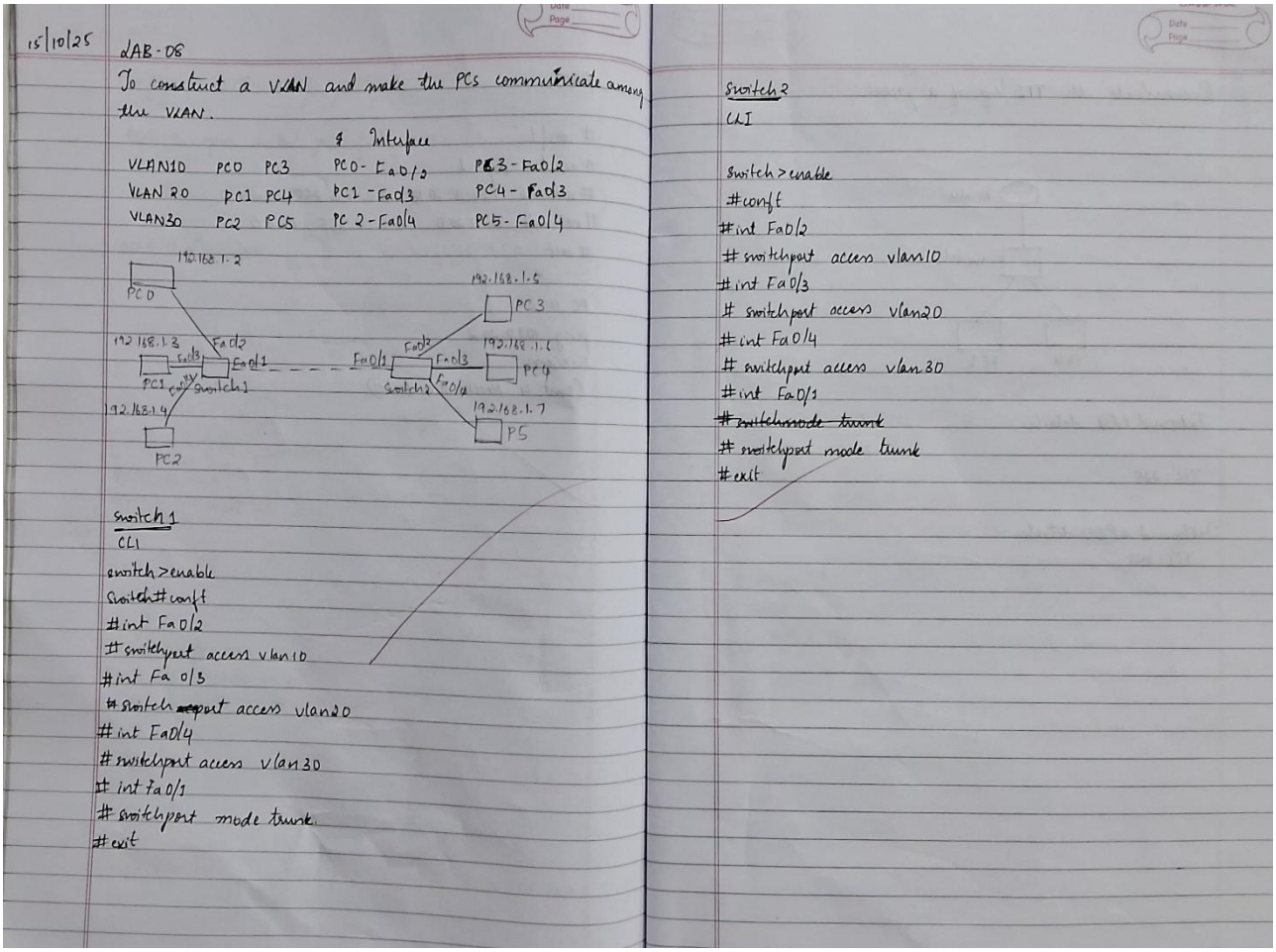
- RIP routing updates were successfully exchanged between routers, allowing each router to dynamically learn remote network routes through hop-count–based distance vector advertisements.
- The routing tables converged correctly, and successful ping tests confirmed end-to-end connectivity maintained by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

Program 7

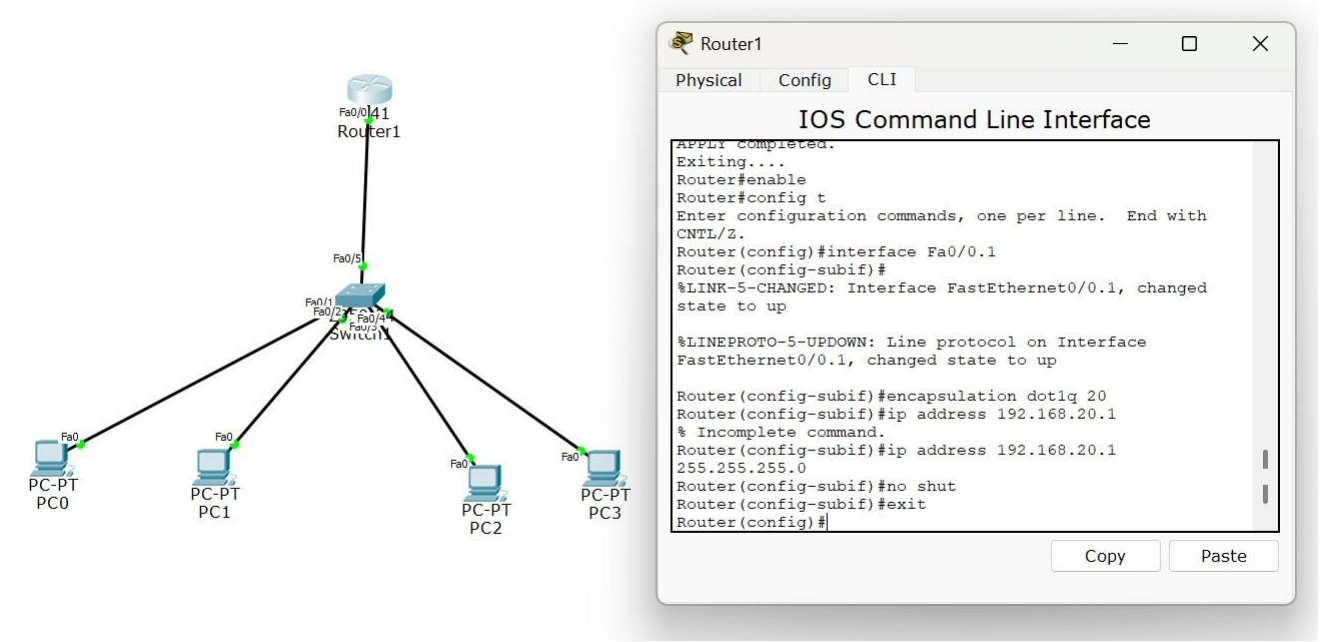
Aim of the program:

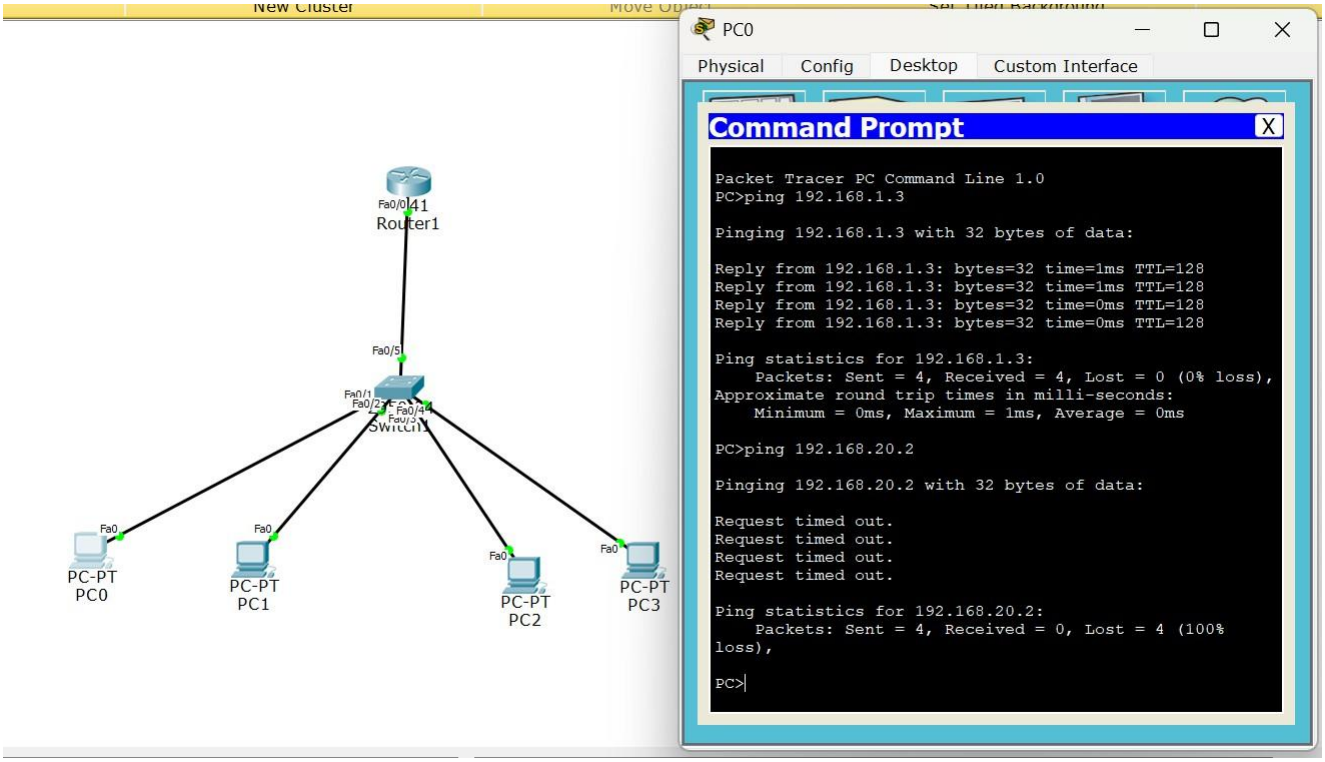
To construct a VLAN and make the PCs communicate among a VLAN

Procedure and topology:



Screenshots/ Output:





Observation:

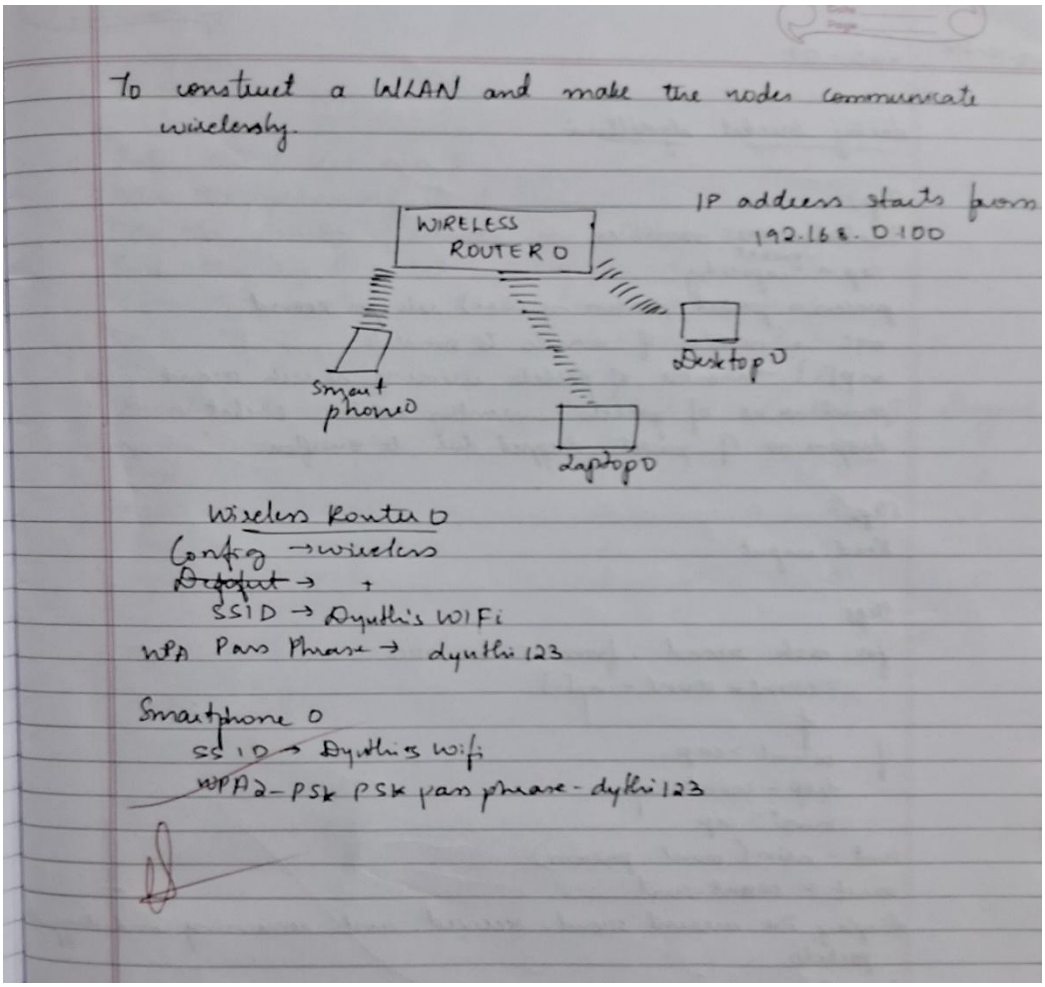
- VLAN segmentation successfully separated broadcast domains, and switch ports were correctly assigned to their respective VLAN IDs using access mode configuration.
- Inter-VLAN communication was achieved through the Layer-3 device, and successful ping tests confirmed proper VLAN membership, tagging,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

Program 8

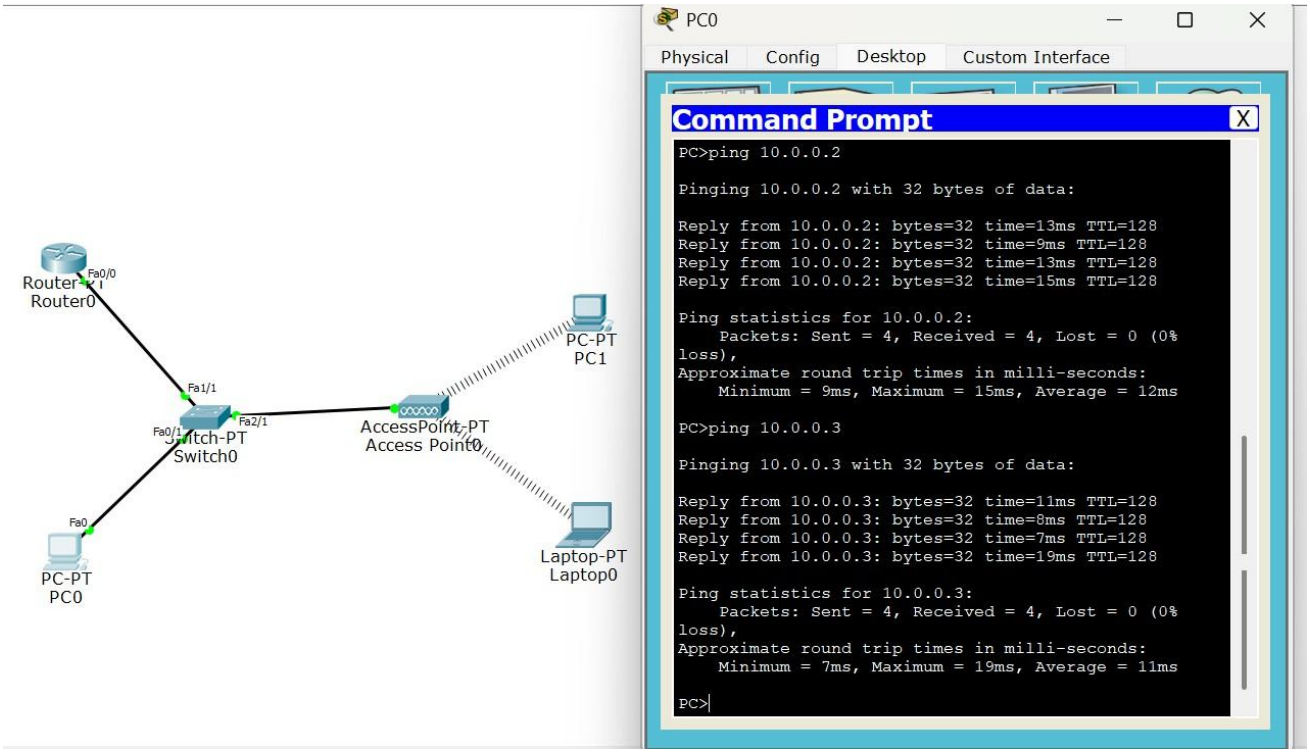
Aim of the program:

To construct a WLAN and make the nodes communicate wirelessly

Procedure and topology:



Screenshots/ Output:



Observation:

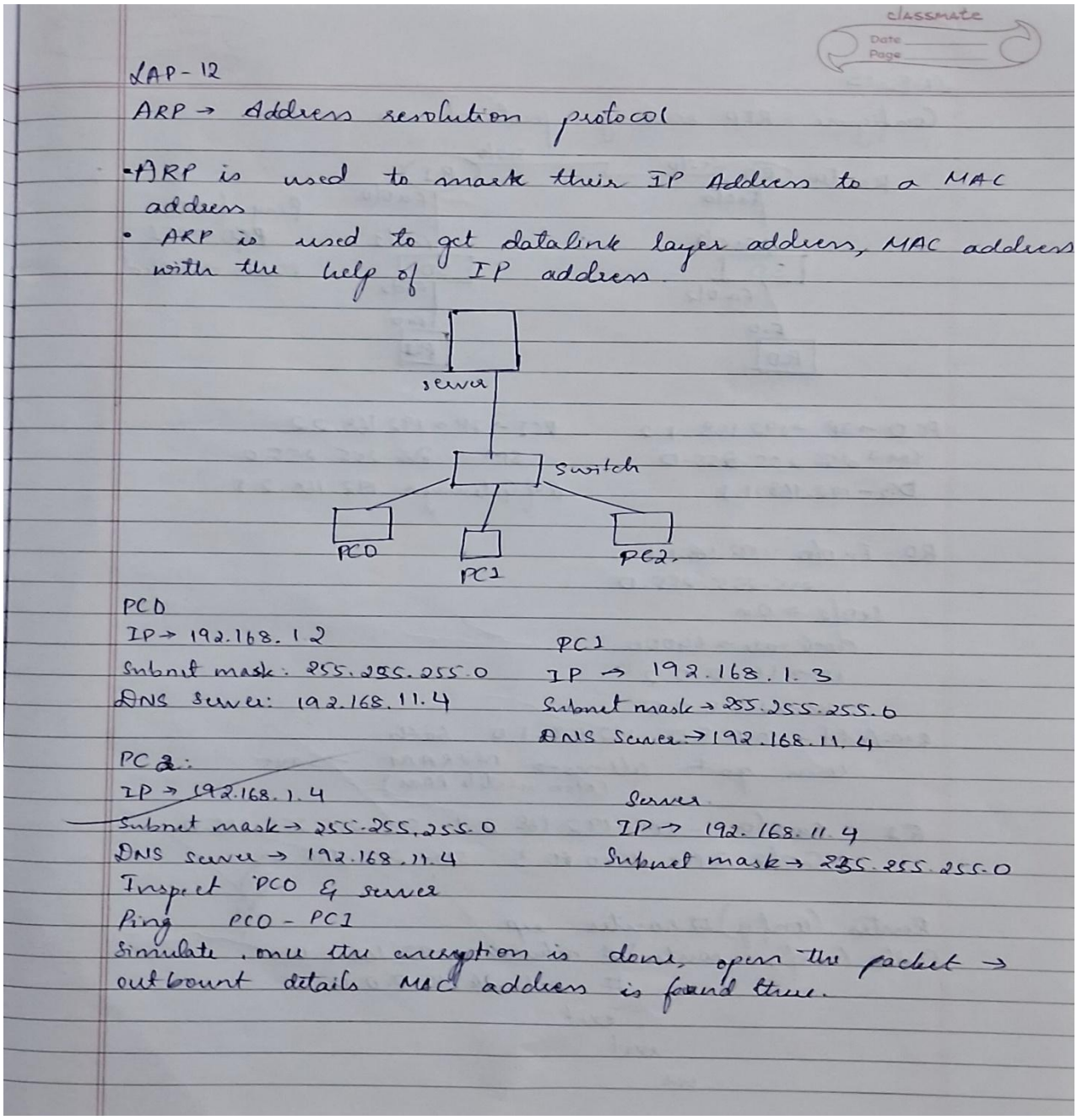
- The WLAN configuration enabled wireless nodes to associate with the access point using the configured SSID and security settings, confirming proper authentication and signal coverage.
- Successful ping communication between wireless devices verified stable wireless Connectivity.

Program 9

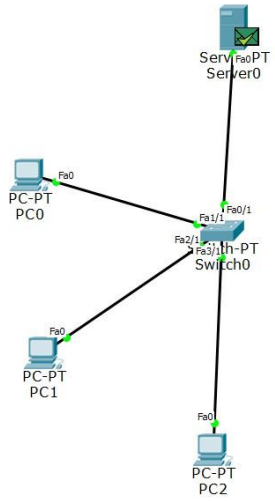
Aim of the program:

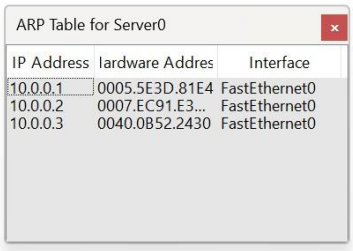
To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)

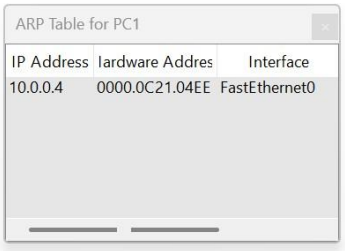
Procedure and topology:

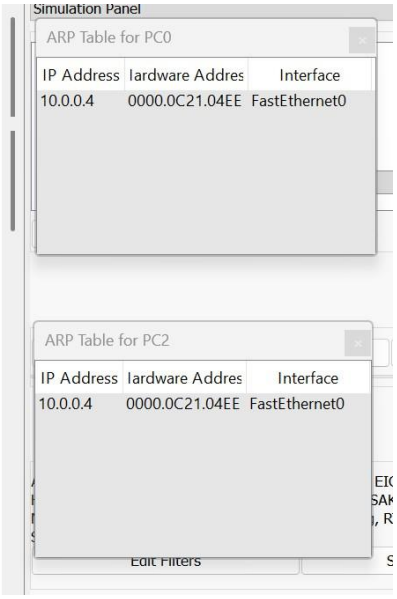


Screenshots/ Output:









Observation:

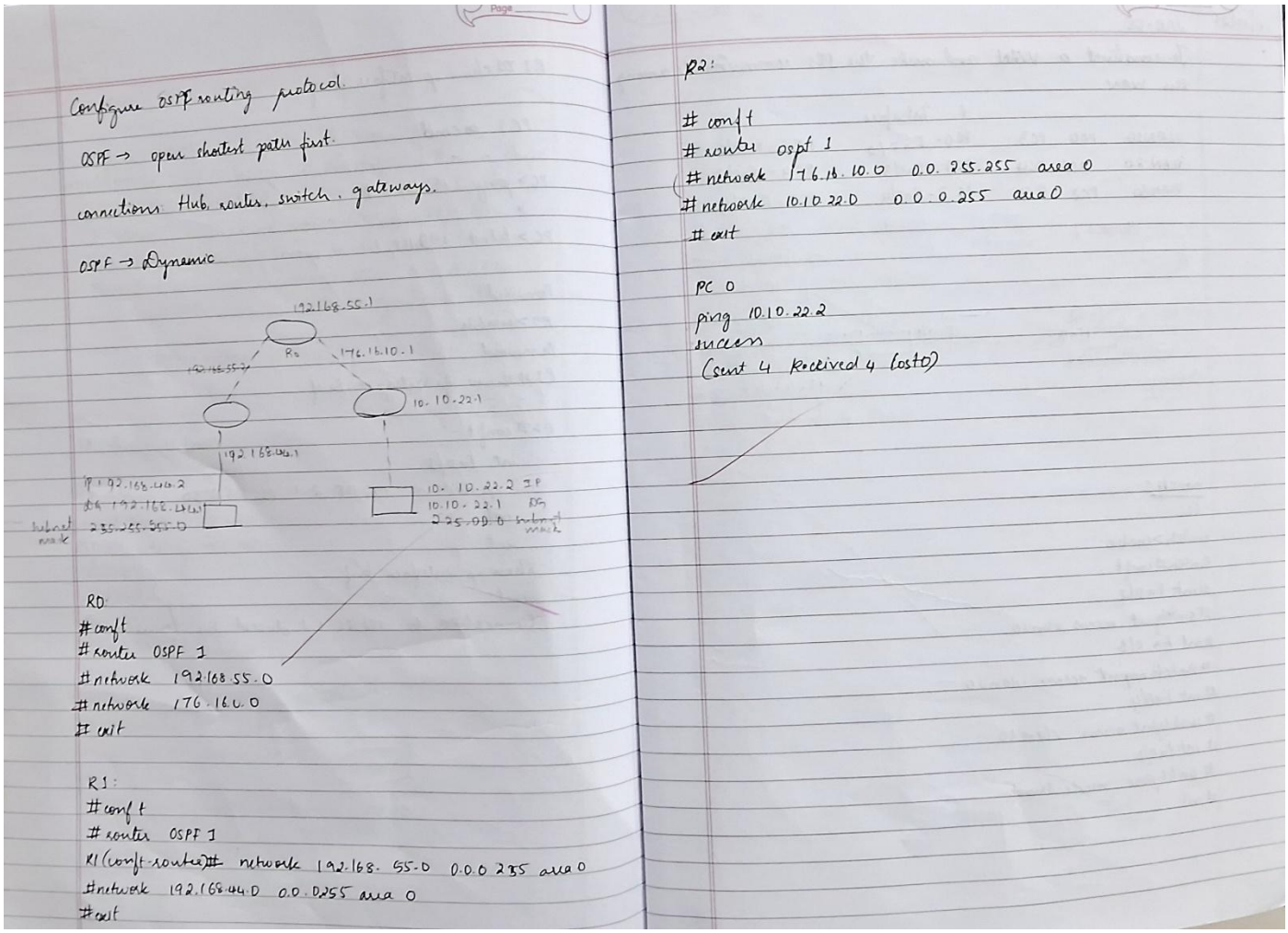
- ARP successfully resolved the destination IP address to its corresponding MAC address, as seen from ARP request and reply exchanges between LAN hosts.
- The populated ARP tables and successful ping communication confirmed correct layer-2 addressing, frame forwarding, and basic LAN operation.

Program 10

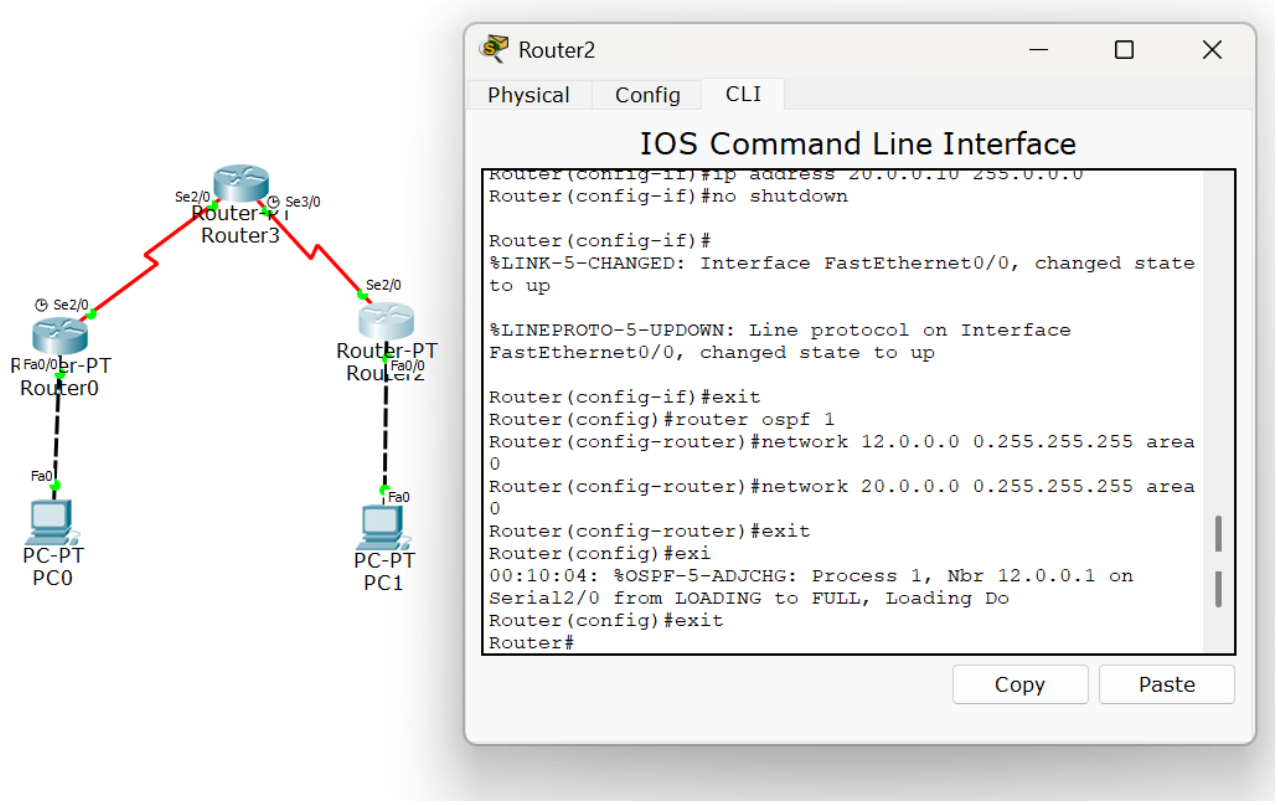
Aim of the program:

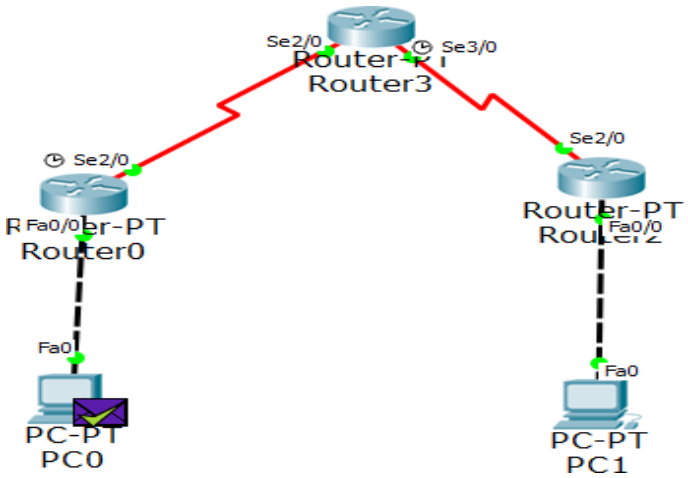
Configure OSPF routing protocol

Procedure and topology:



Screenshots/ Output:





Observation:

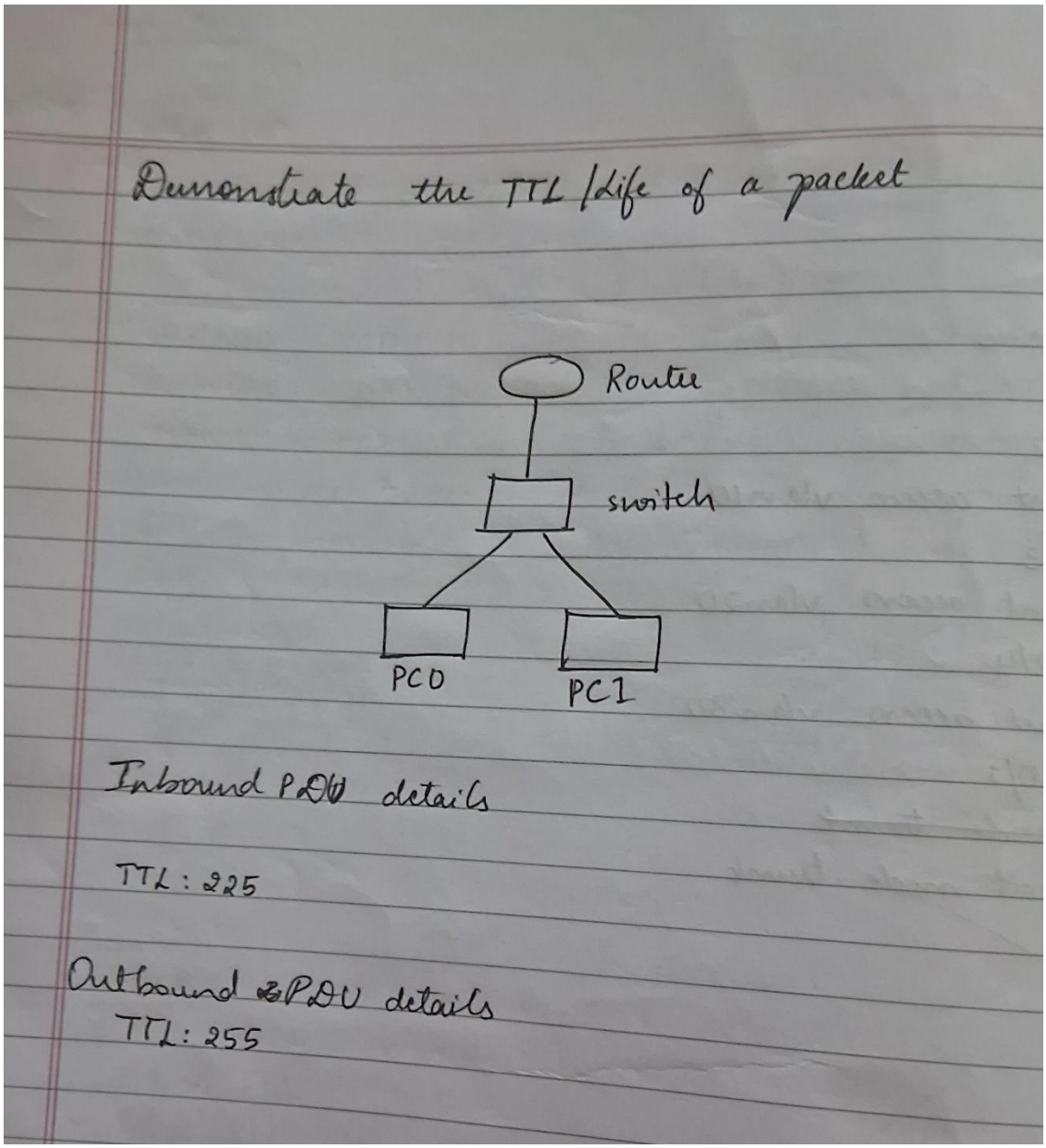
- OSPF successfully established neighbour adjacencies and exchanged LSAs, allowing routers to build a synchronized link-state database across the OSPF area.
- The routing tables converged using SPF calculations, and successful pings confirmed efficient path selection and dynamic route learning through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

Program 11

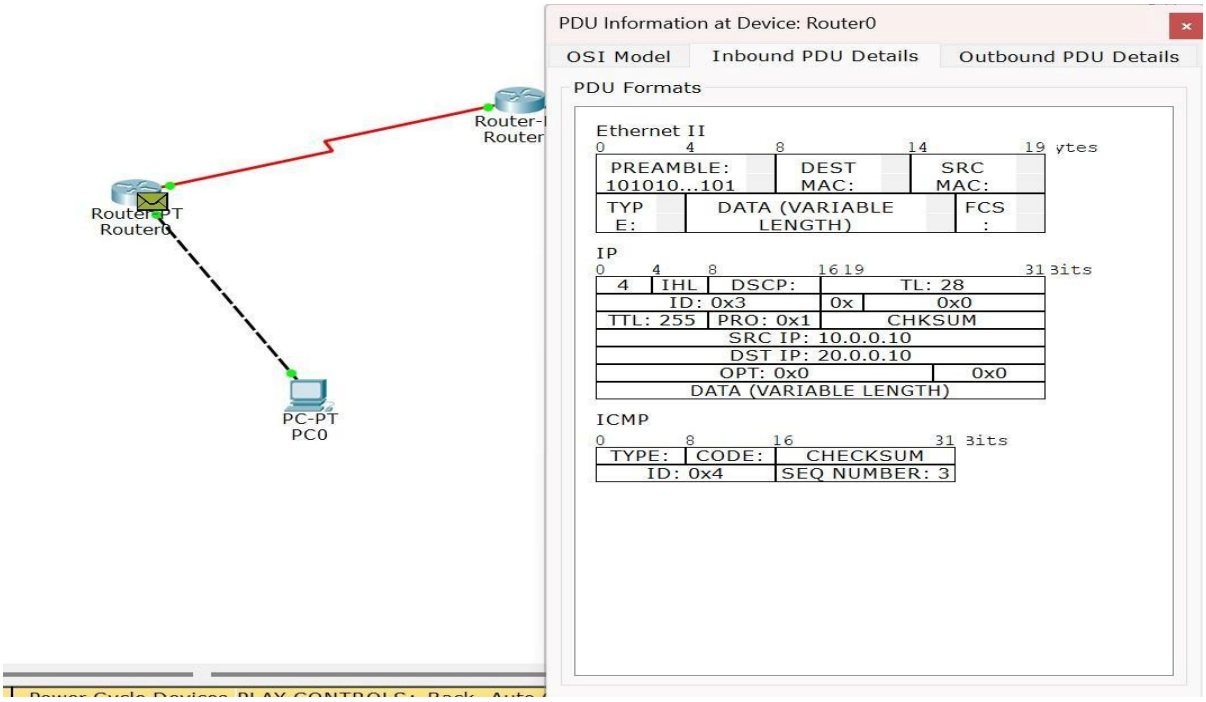
Aim of the program:

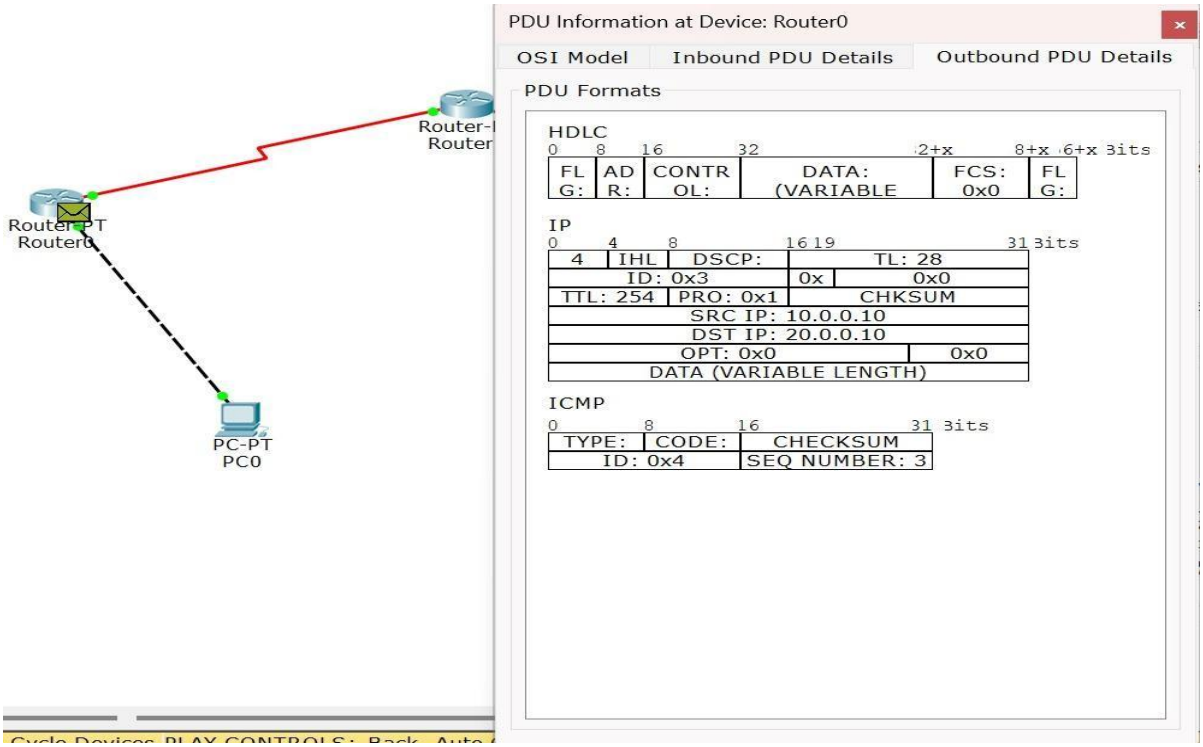
Demonstrate the TTL/ Life of a Packet

Procedure and topology:



Screenshots/ Output:





Observation:

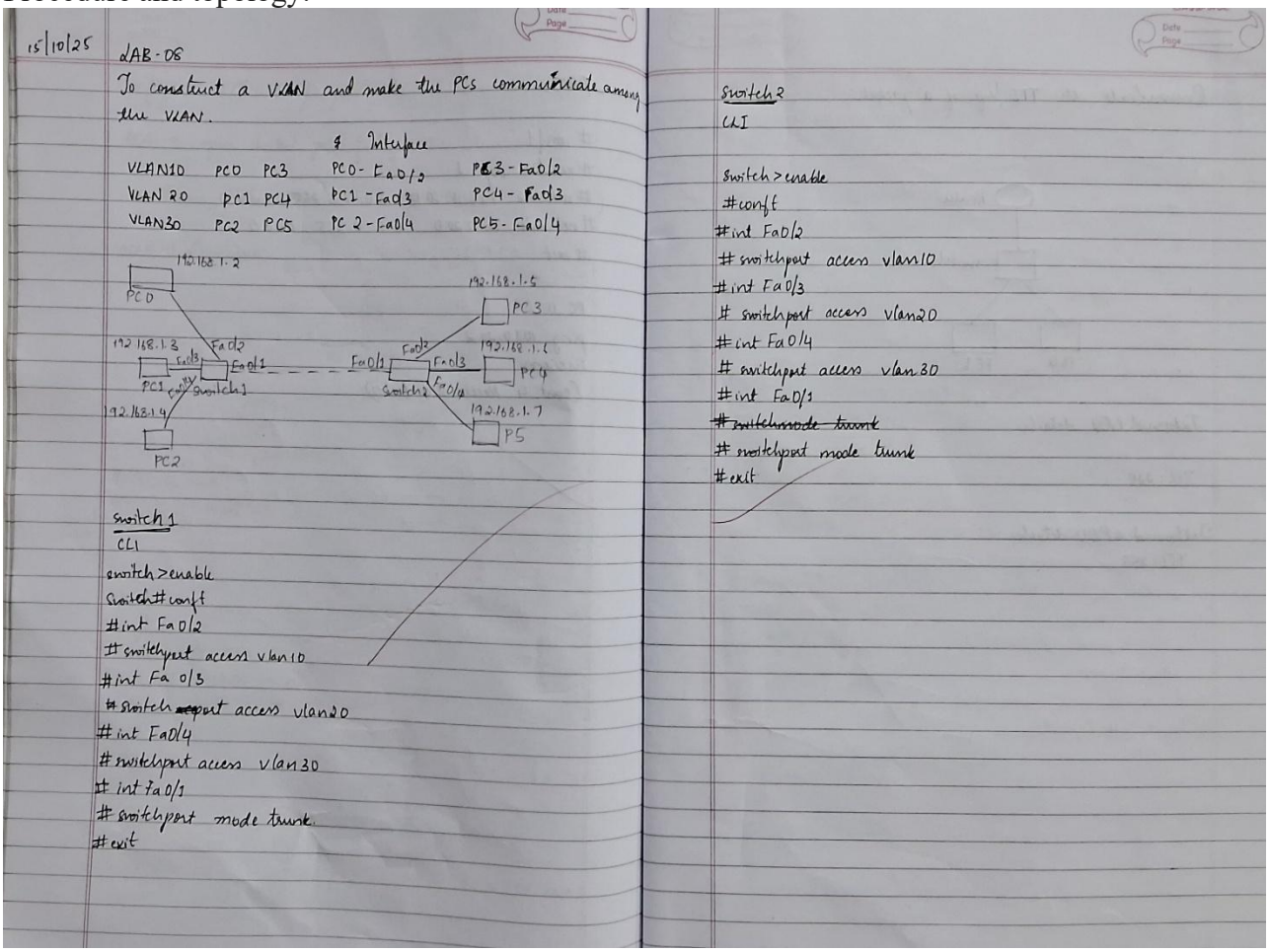
- The TTL field in the IP header decreased by one at each router hop, demonstrating its role in preventing packets from looping indefinitely in the network.

Program 12

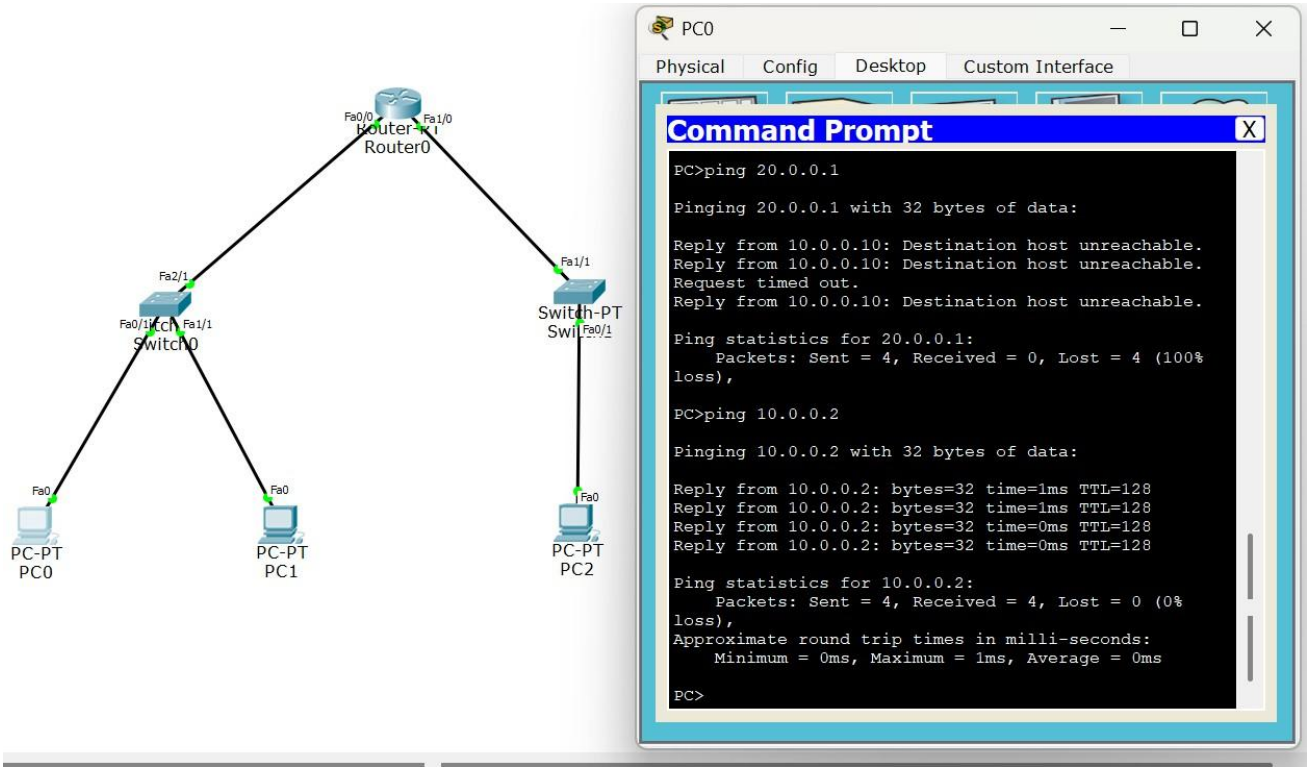
Aim of the program:

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.

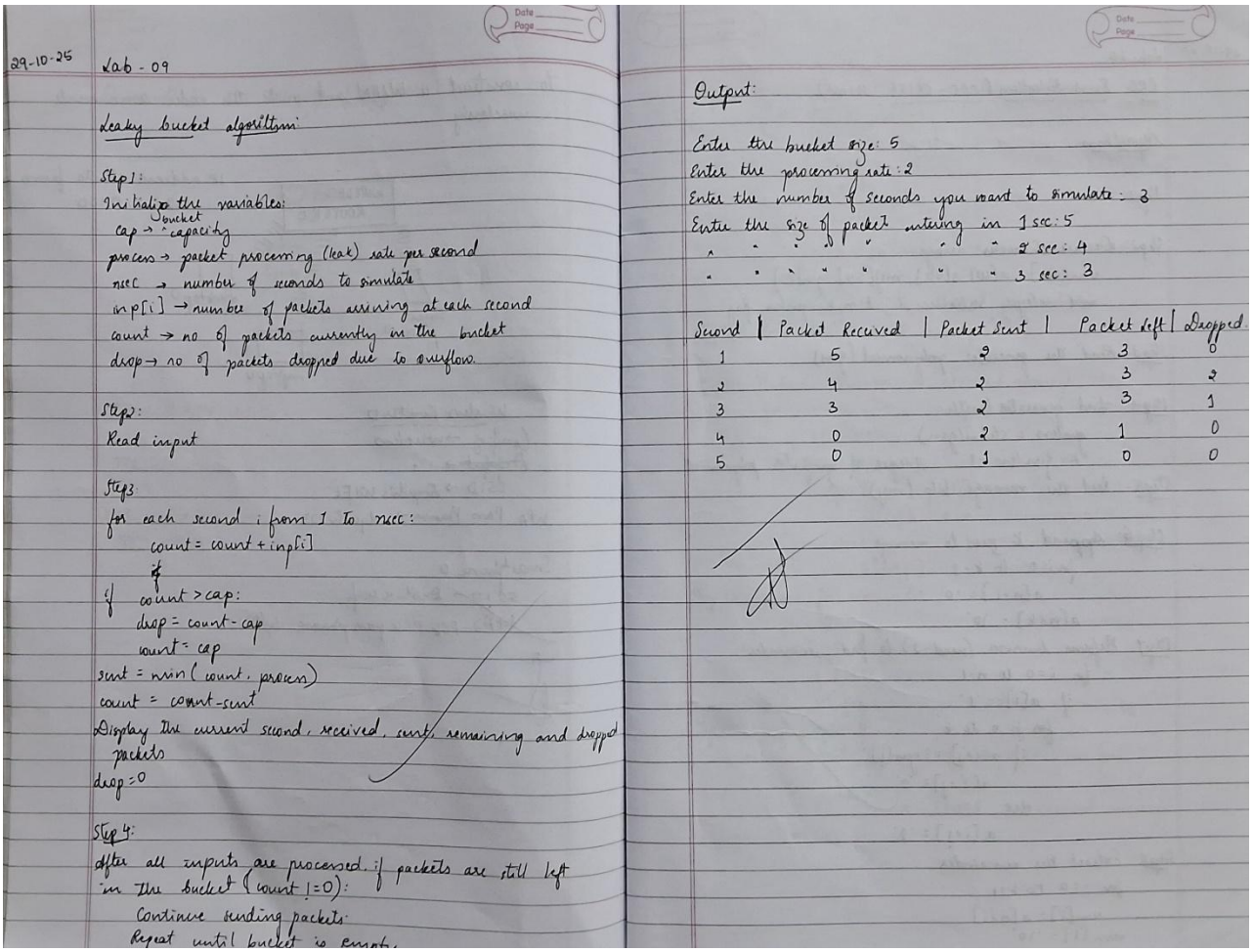
CYCLE 2:

Program 13

Aim of the program:

Write a program for congestion control using Leaky bucket algorithm

Procedure and topology:



29-10-25 Lab - 09

Leaky bucket algorithm:

Step 1:
Initialize the variables:
cap → bucket capacity
process → packet processing (leak) rate per second
nsec → number of seconds to simulate
inp[i] → number of packets arriving at each second
count → no of packets currently in the bucket
drop → no of packets dropped due to overflow

Step 2:
Read input

Step 3:
for each second i from 1 To nsec:
count = count + inp[i]
if count > cap:
drop = count - cap
count = cap
sent = min(count, process)
count = count - sent
Display the current second, received, sent, remaining and dropped packets
drop = 0

Step 4:
After all inputs are processed, if packets are still left in the bucket (count != 0):
Continue sending packets:
Repeat until bucket is empty

Output:

Enter the bucket size: 5
Enter the processing rate: 2
Enter the number of seconds you want to simulate: 3
Enter the size of packet entering in 1 sec: 5
" " " " " " " 2 sec: 4
" " " " " " " 3 sec: 3

Second	Packet Recived	Packet Sent	Packet left	Dropped
1	5	2	3	0
2	4	2	3	2
3	3	2	3	1
4	0	2	1	0
5	0	1	0	0

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:

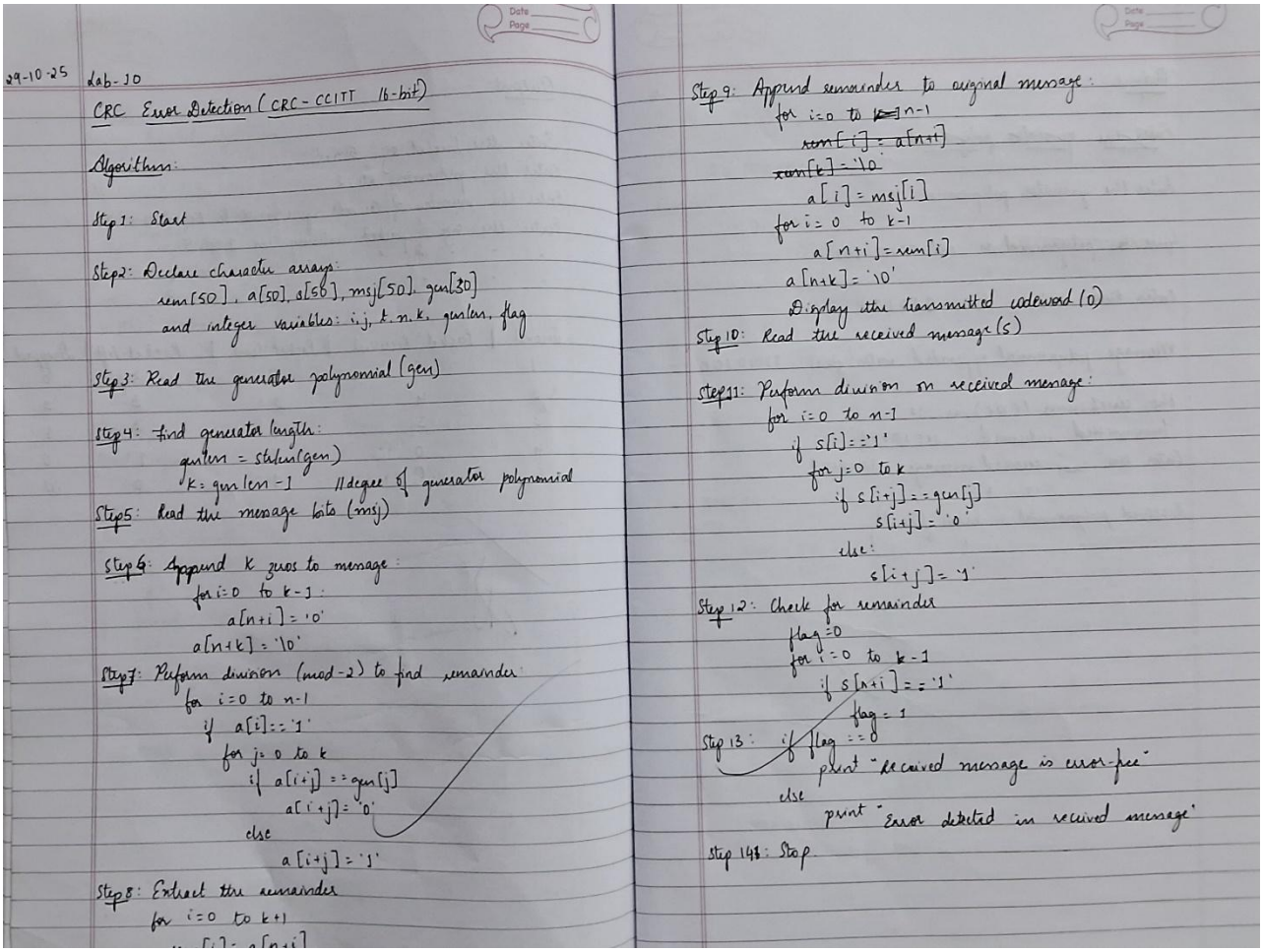
- The leaky bucket mechanism regulated the outgoing packet flow at a constant rate, preventing sudden traffic bursts from overwhelming the network.
- Packets exceeding the bucket capacity were dropped, demonstrating effective congestion control by smoothing traffic and enforcing rate-limiting.

Program 14

Aim of the program:

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

Procedure and topology:



29-10-25 Lab-10

CRC Error Detection (CRC-CCITT 16-bit)

Algorithm:

Step 1: Start

Step 2: Declare character arrays
rem[50], a[50], s[50], msj[50], gen[30]
and integer variables: i, j, t, n, k, genlen, flag

Step 3: Read the generator polynomial (gen)

Step 4: find generator length:
genlen = strlen(gen)
k = genlen - 1 // degree of generator polynomial

Step 5: Read the message bits (msj)

Step 6: Append k zeros to message:
for i=0 to k-1:
a[n+i] = '0'
a[n+k] = '\0'

Step 7: Perform division (mod-2) to find remainder
for i=0 to n-1
if a[i]=='1'
for j=0 to k
if a[i+j] == gen[j]
a[i+j] = '0'
else
a[i+j] = '1'

Step 8: Extract the remainder
for i=0 to k+1

Step 9: Append remainder to original message:
for i=0 to n-1
rem[i] = a[n+i]
rem[k] = '\0'
a[i] = msj[i]
for i=0 to k-1
a[n+i] = rem[i]
a[n+k] = '\0'
Display the transmitted codeword (a)

Step 10: Read the received message (s)

Step 11: Perform division on received message:
for i=0 to n-1
if s[i]=='1'
for j=0 to k
if s[i+j] == gen[j]
s[i+j] = '0'
else
s[i+j] = '1'

Step 12: Check for remainder
flag=0
for i=0 to k-1
if s[n+i] == '1'
flag = 1

Step 13: if flag == 0
print "Received message is error-free"
else
print "Error detected in received message"

Step 14: Stop

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:

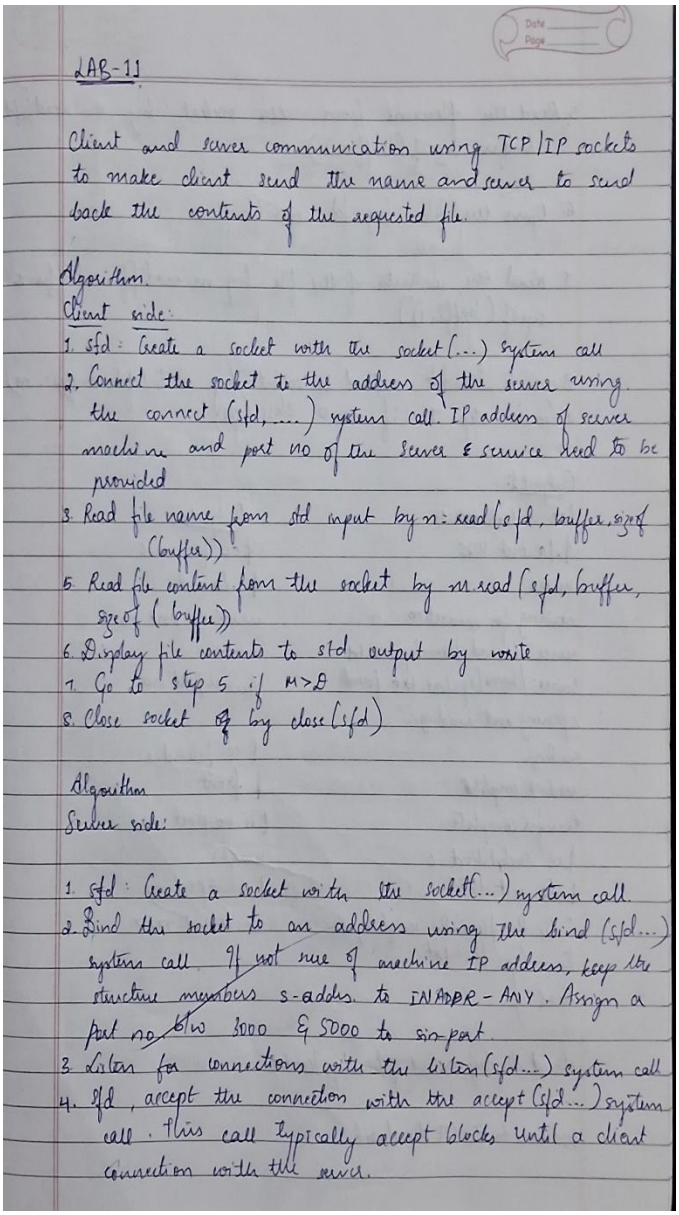
- The CRC-CCITT (16-bit) algorithm correctly generated a checksum for the transmitted data, ensuring reliable detection of single-bit and burst errors.
- Intentional error tests produced mismatched CRC values at the receiver, confirming accurate error detection through polynomial division.

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:



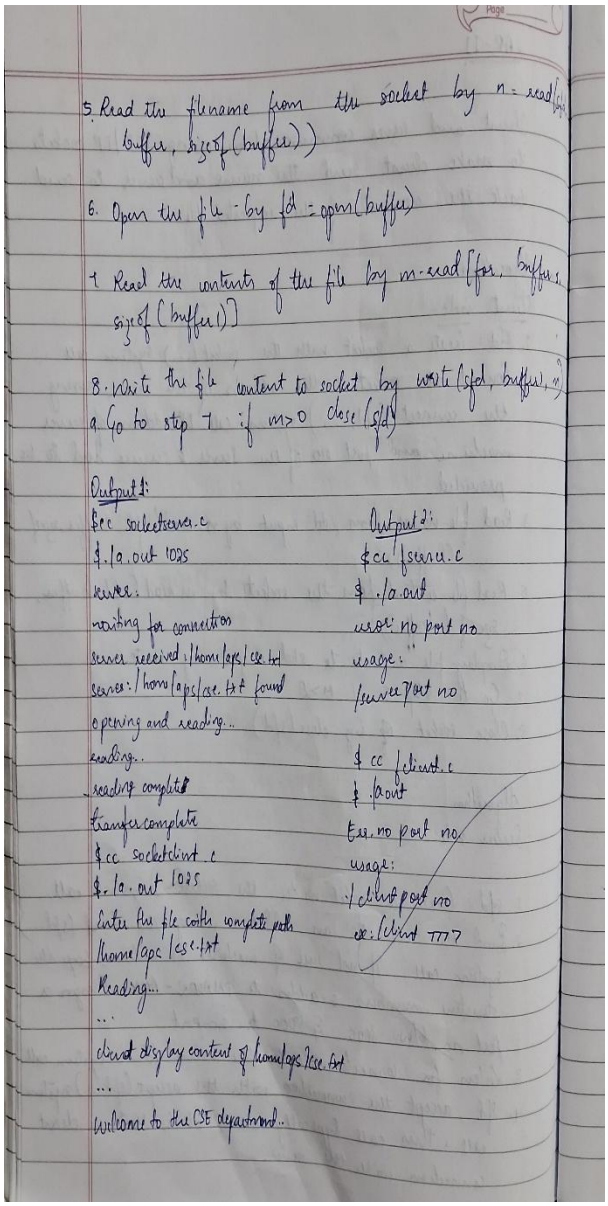
LAB-11

Client and server communication using TCP/IP sockets to make client send the name and server to send back the contents of the requested file.

Algorithm
Client side:
1. sfd: Create a socket with the socket(...) system call
2. Connect the socket to the address of the server using the connect(sfd,....) system call. IP address of server machine and port no of the server & service need to be provided
3. Read file name from std input by n: read(sfd, buffer, sizeof(buffer))
5. Read file content from the socket by n: read(sfd, buffer, sizeof(buffer))
6. Display file contents to std output by write
7. Go to step 5 if n>0
8. Close socket sfd by close(sfd)

Algorithm
Server side:

1. sfd: Create a socket with the socket(...) system call
2. Bind the socket to an address using the bind(sfd...) system call. If not sure of machine IP address, keep the structure members s-addr to INADDR-ANY. Assign a port no b/w 3000 & 5000 to sin-port
3. Listen for connections with the listen(sfd,...) system call
4. sfd, accept the connection with the accept(sfd...) system call. This call typically accept blocks until a client connection with the server.



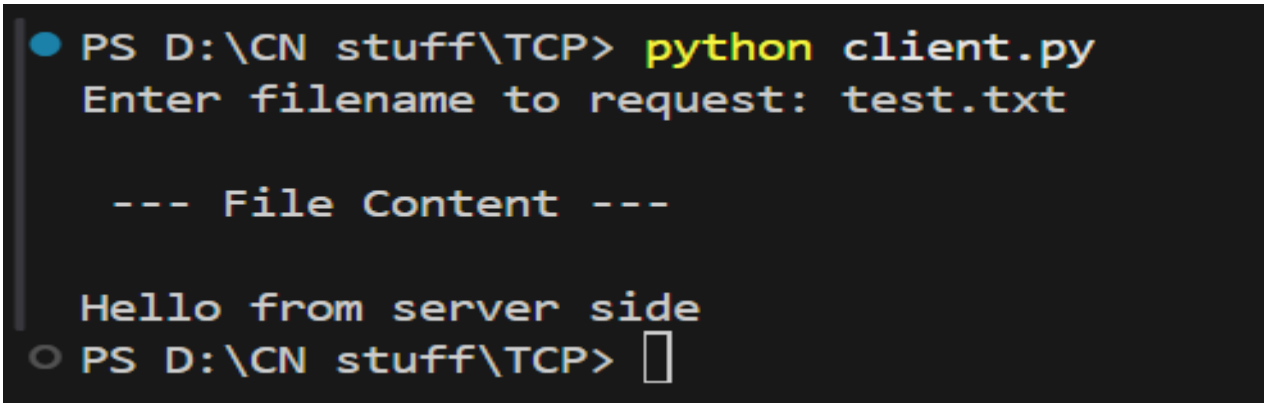
5. Read the filename from the socket by n = read(sfd, buffer, sizeof(buffer))
6. Open the file by fd = open(buffer)
7. Read the contents of the file by n=read(fd, buffer, sizeof(buffer))
8. Write the file content to socket by write(sfd, buffer, n)
9. Go to step 7 if n>0 close(sfd)

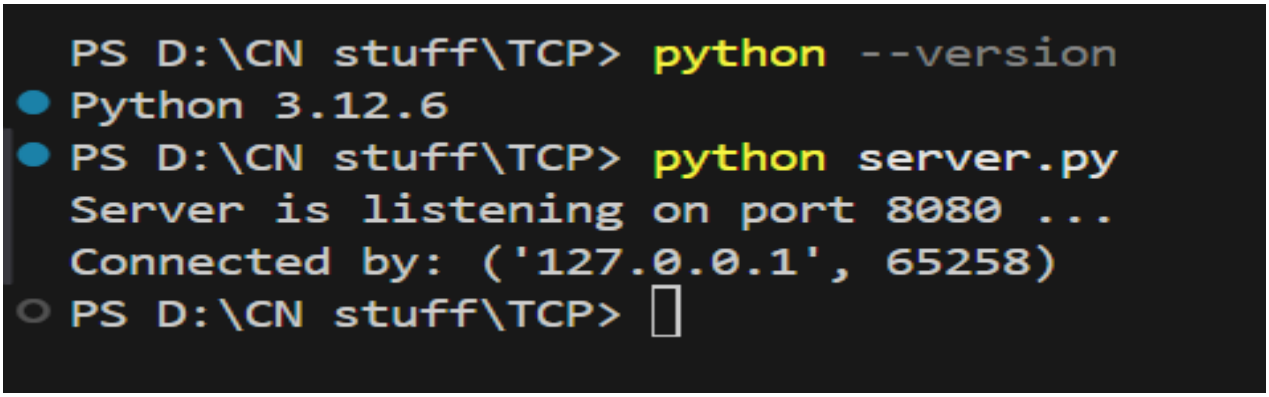
Output1:
$cc socketserv.c
$./a.out 1025
server:
waiting for connection
server received: /home/ops/cse.txt
server: /home/ops/cse.txt found
opening and reading...
reading...
reading complete
transfer complete
$cc socketclient.c
$./a.out 1025
Enter the file with complete path
/home/ops/cse.txt
Reading...
...
client display content of /home/ops/cse.txt
...
Welcome to the CSE department...

Output2:
$cc fserv.c
$./a.out
usage: no port no
usage:
fserver port no

$cc fclient.c
$./a.out
use no port no
usage:
./client port no
ex: ./client 7777

Screenshots/ Output:





Observation:

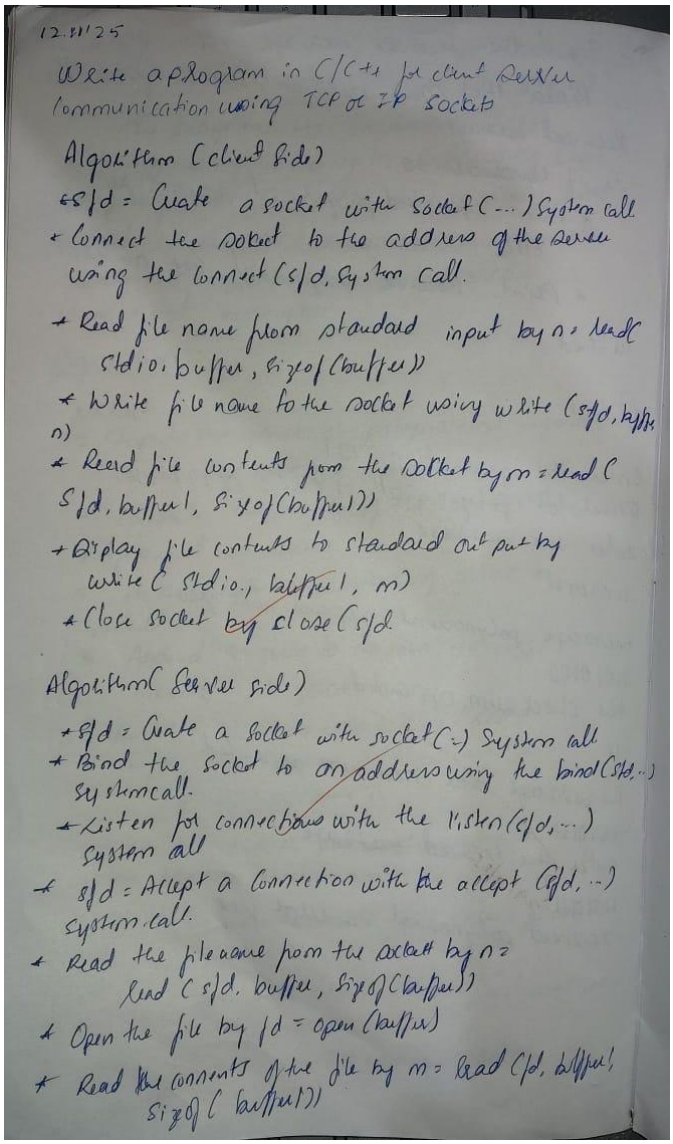
- The TCP client successfully established a reliable connection with the server and transmitted the requested filename using stream-oriented communication.
- The server correctly retrieved and returned the file contents over the same TCP session, demonstrating reliable data transfer, acknowledgment handling,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



12.11.25

Write a program in C/C++ for client Server communication using TCP or IP sockets

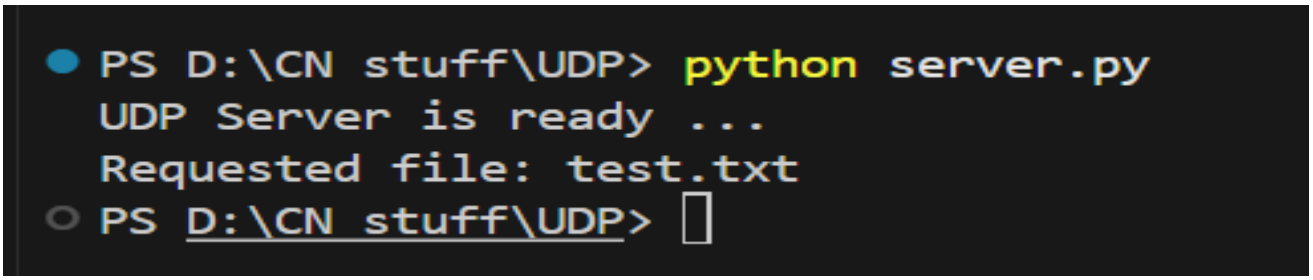
Algorithm (client side)

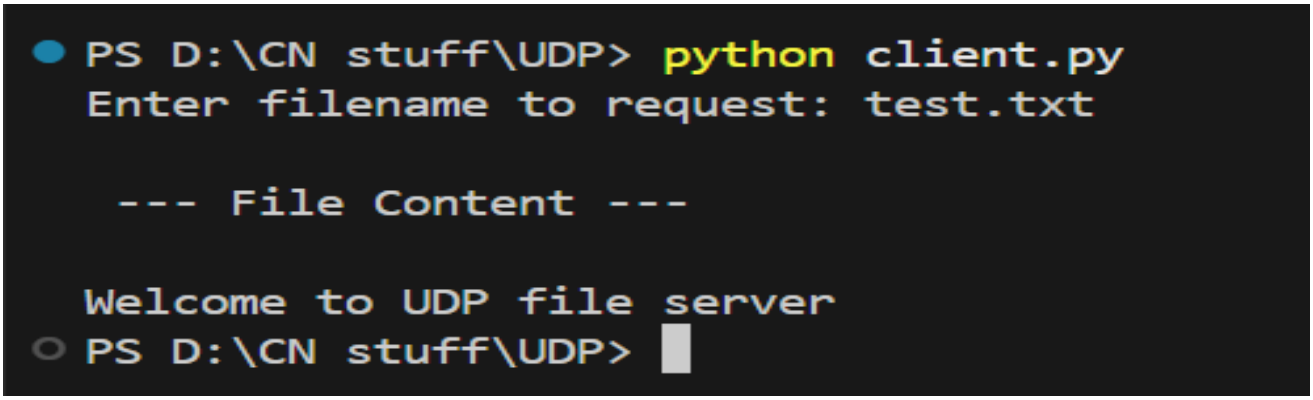
- s/d = Create a socket with Socket(...) System call
- Connect the socket to the address of the Server using the connect(s/d, System call.
- Read file name from standard input by n = read(stdio, buffer, sizeof(buffer))
- Write file name to the socket using write(s/d, buffer, n)
- Read file contents from the socket by m = read(s/d, buffer1, sizeof(buffer1))
- Display file contents to standard output by write(stdio, buffer1, m)
- Close socket by close(s/d)

Algorithm(Server side)

- s/d = Create a socket with socket(..) System call
- Bind the socket to an address using the bind(std...) systemcall.
- Listen for connections with the listen(s/d, ...) System call
- s/d = Accept a connection with the accept(s/d, ...) System call.
- Read the filename from the socket by n = read(s/d, buffer, sizeof(buffer))
- Open the file by fd = open(buffer)
- Read the contents of the file by m = read(fd, buffer1, sizeof(buffer1))

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.