VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

MAKADIA RISHIT DILIPBHAI (1BM23CS177)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
September 2025 – January 2026

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "Computer Network (23CS5PCCON)" carried out by **MAKADIA RISHIT DILIPBHAI (1BM23CS177),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements of the above-mentioned subject and the work prescribed for the said degree.

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

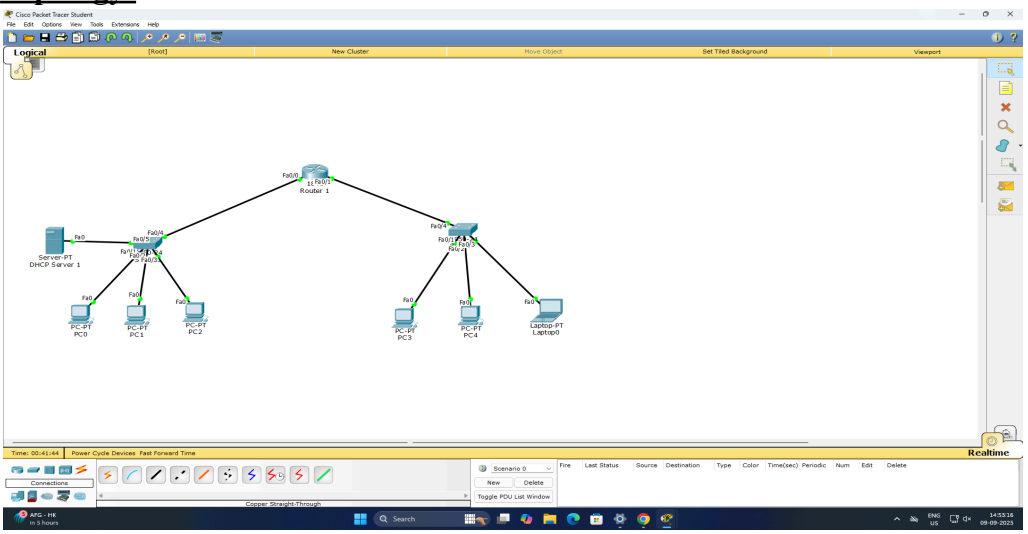
Index

Github Link:
https://github.com/rishitmakadia/1BM23CS177_CON_5

Program – 1:

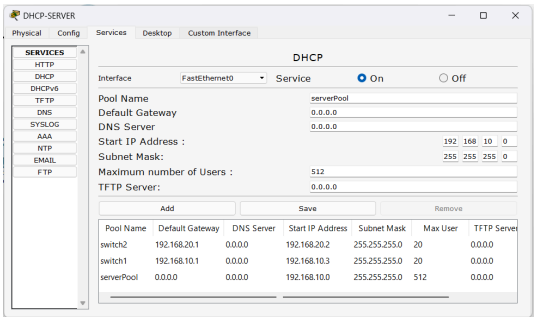
Aim: Configure DHCP within a LAN and outside LAN.

Topology:



Procedure:

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

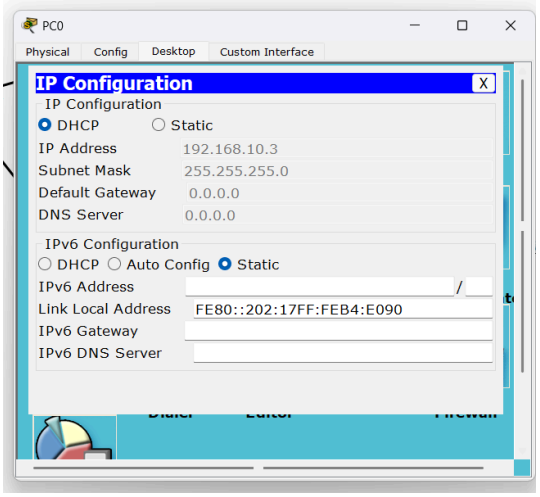


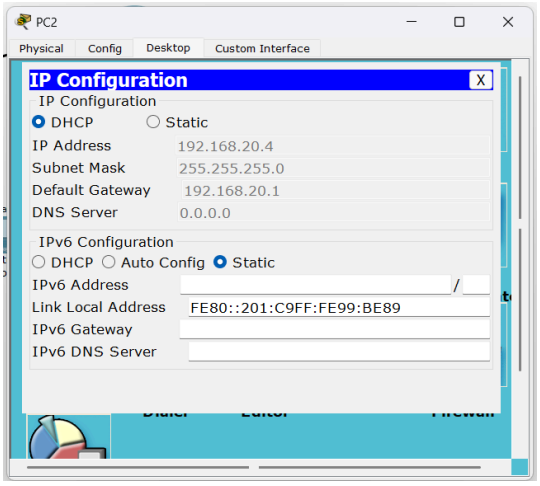
3. Configure Router

```
Router>enable
Router#configure terminal
Router(config)# int fa0/0
Router(config-if)# ip address 192.168.10.1 255.255.255.0
Router(config-if)# ip helper-address 192.168.10.2
Router(config-if)# no shutdown
Router(config-if)# exit
```

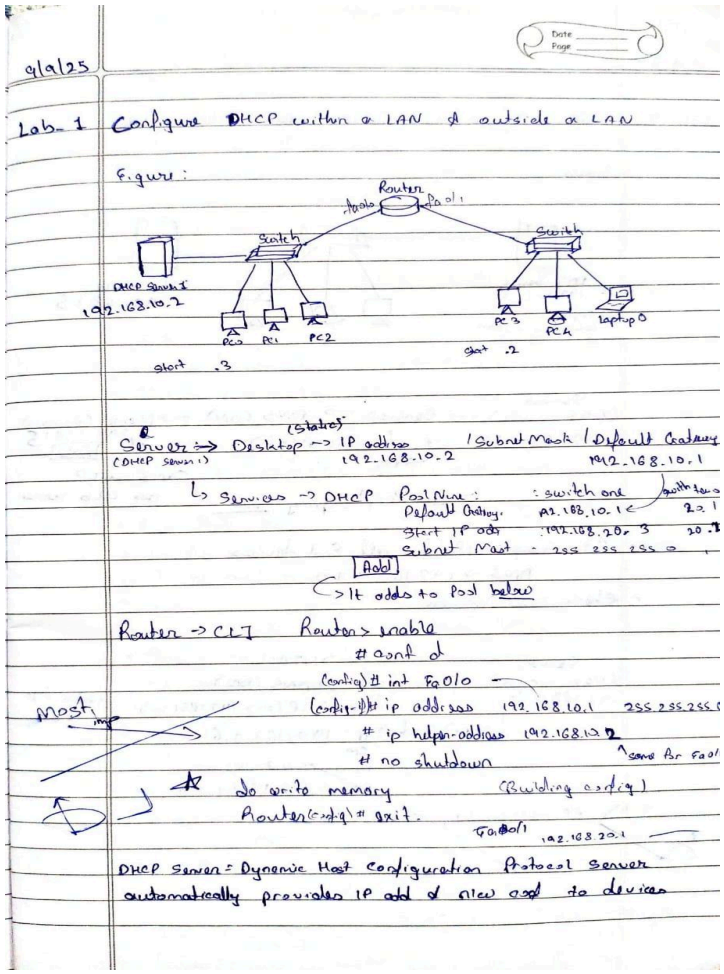
```
Router(config)# int fa0/1
Router(config-if)# ip address 192.168.20.1 255.255.255.0
Router(config-if)# ip helper-address 192.168.10.2
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# exit
Router# write memory
```

Output





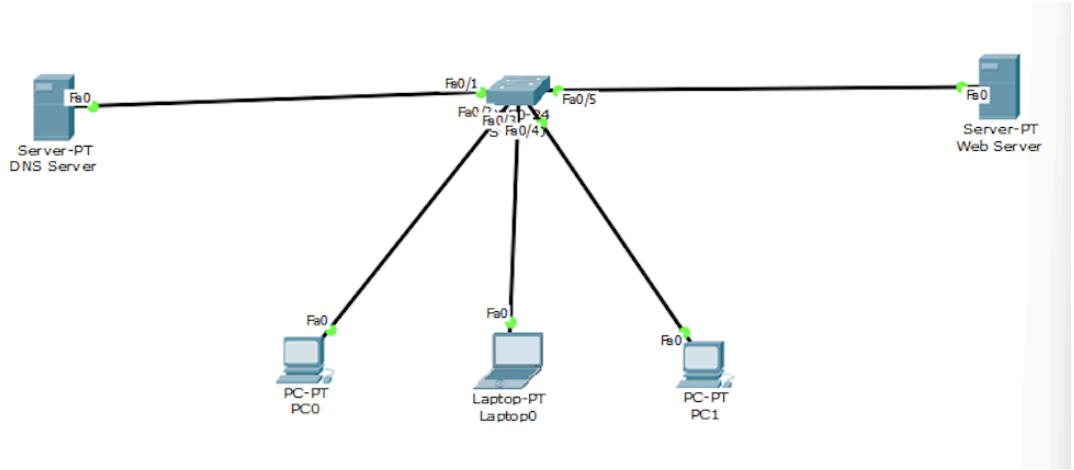
Observation



Program 2

Aim: Configure Web Server, DNS within a LAN.

Topology:



Procedure:

1. Configure Server

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

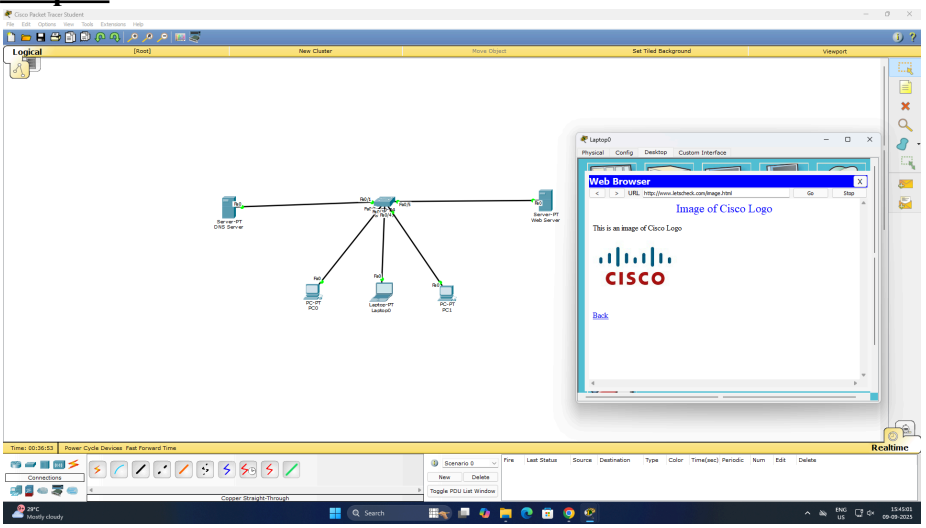
2. Configure Clients

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

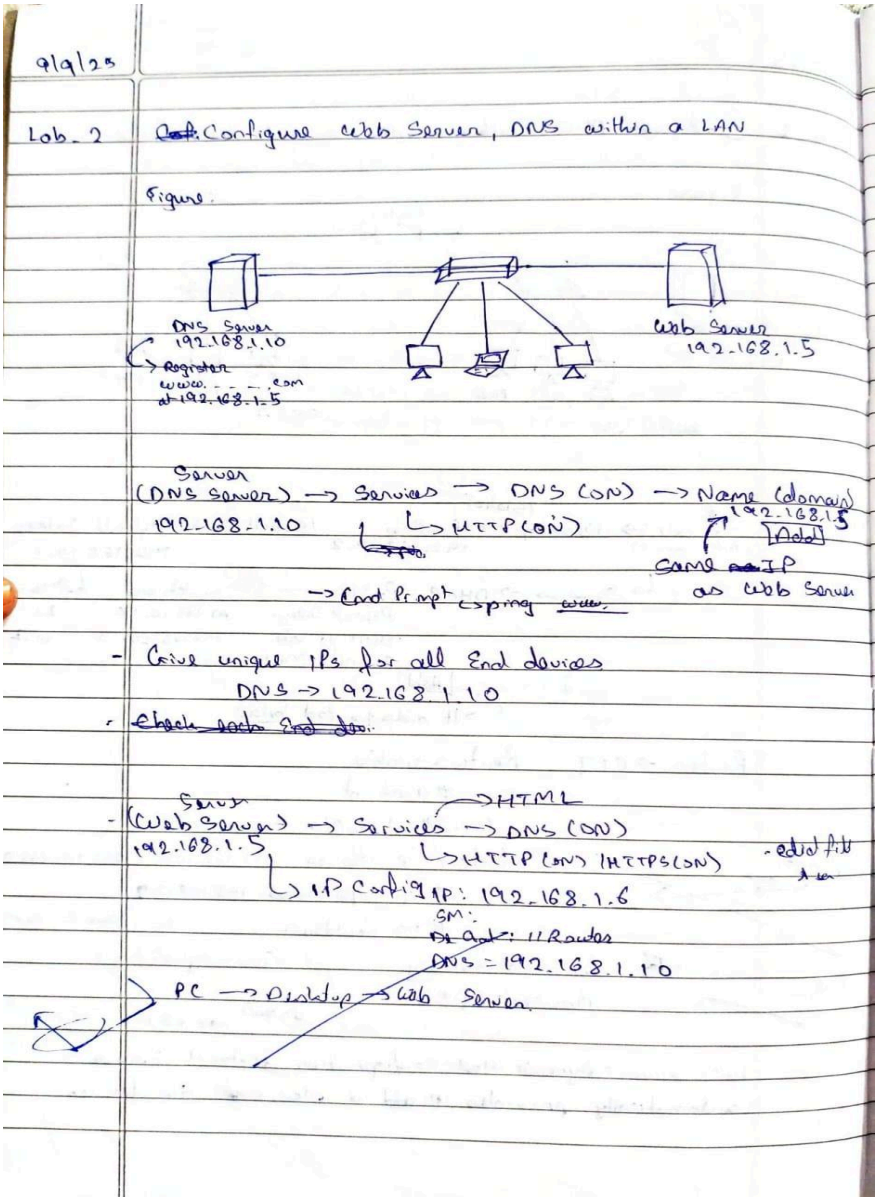
3. Verification

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

Output:



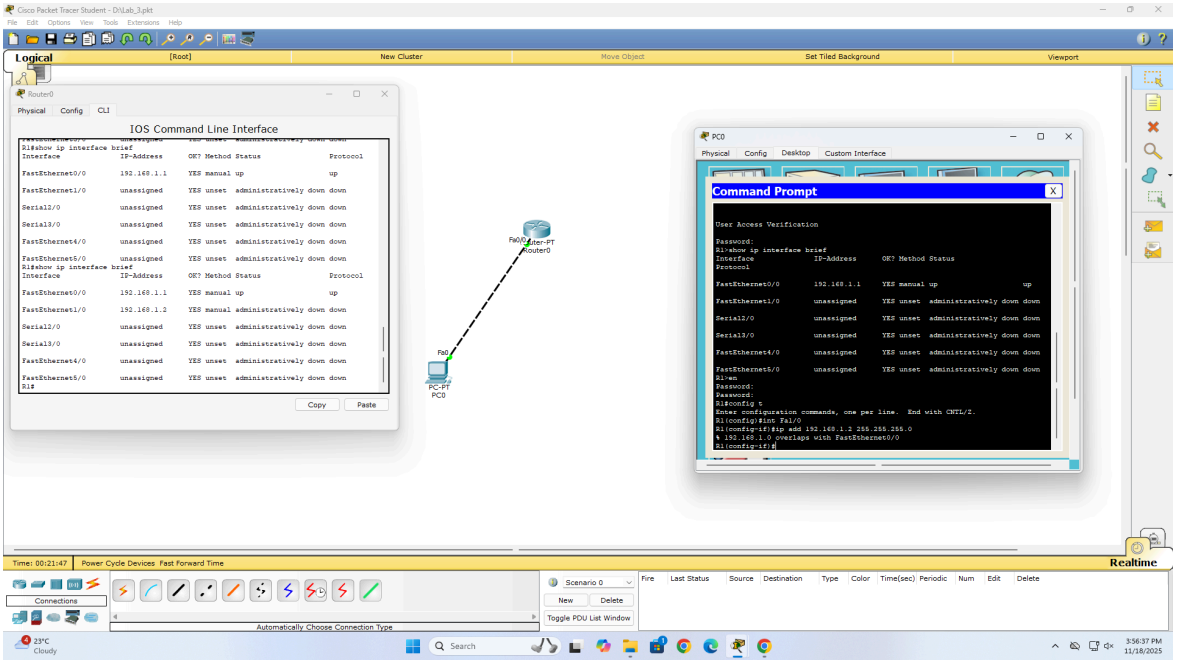
Observation:



Program 3

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

Topology:



Procedure:

1. Configure Router

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

2. Configure PC

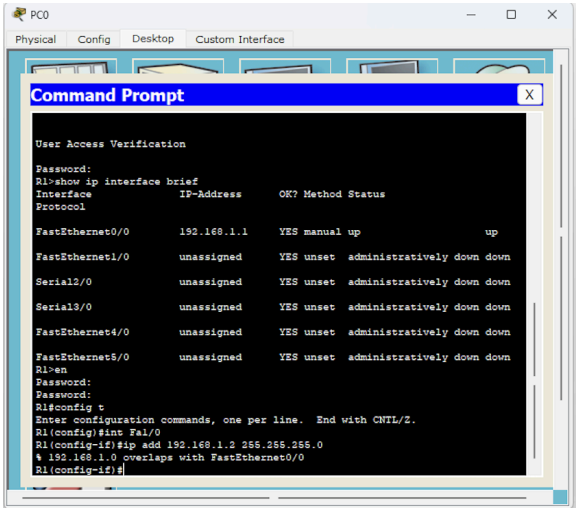
- Set static IP: 192.168.1.2/24
- Default gateway: 192.168.1.1

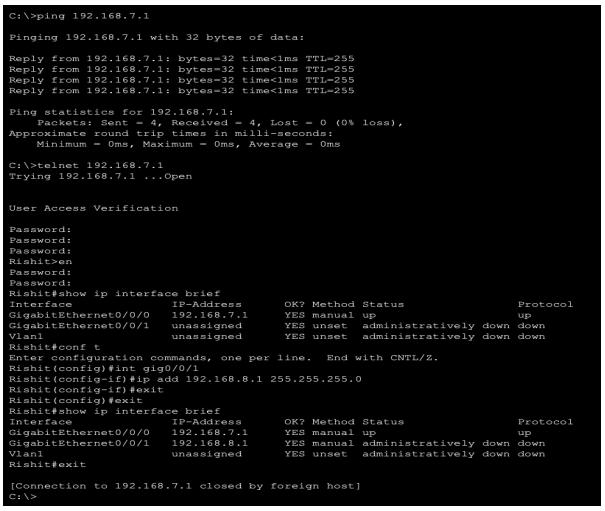
3. Establish TELNET Session

ip: 192.168.1.2
mask: 255.255.255.0

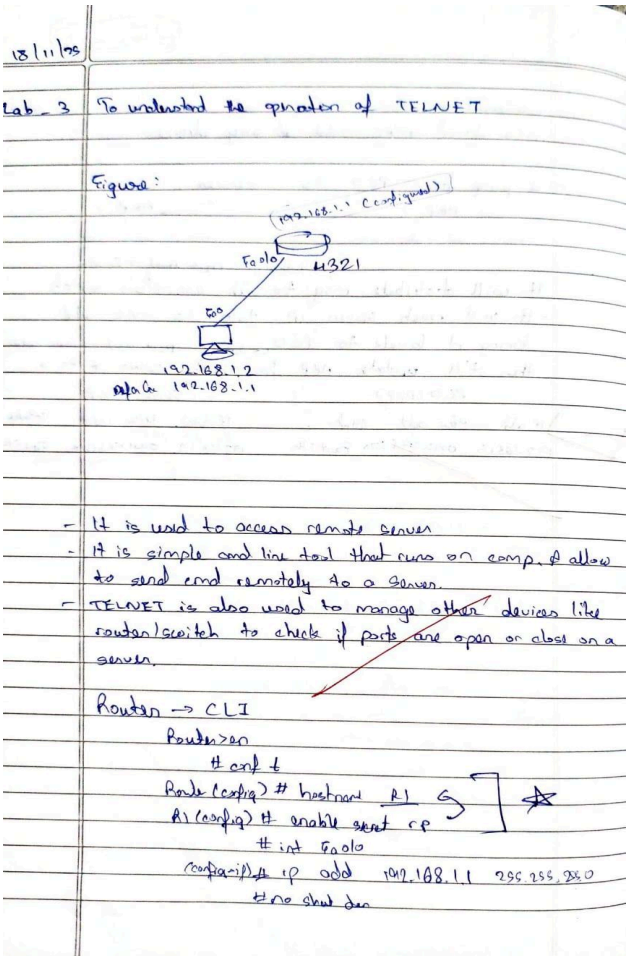
gateway: 192.168.1.1
Test connectivity
ping 192.168.1.1
telnet 192.168.1.1
Password: tp
enable
Secret: class

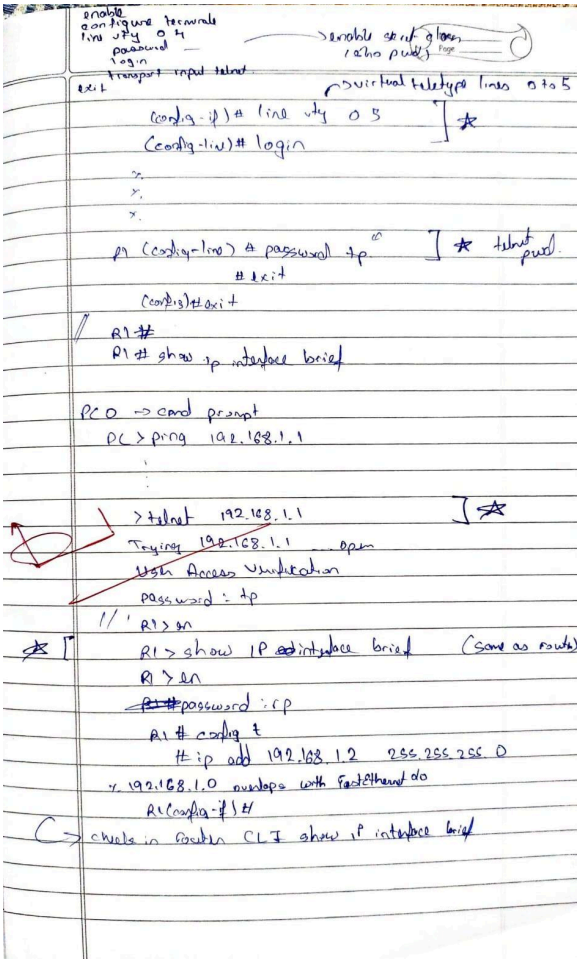
Output:





Observation:

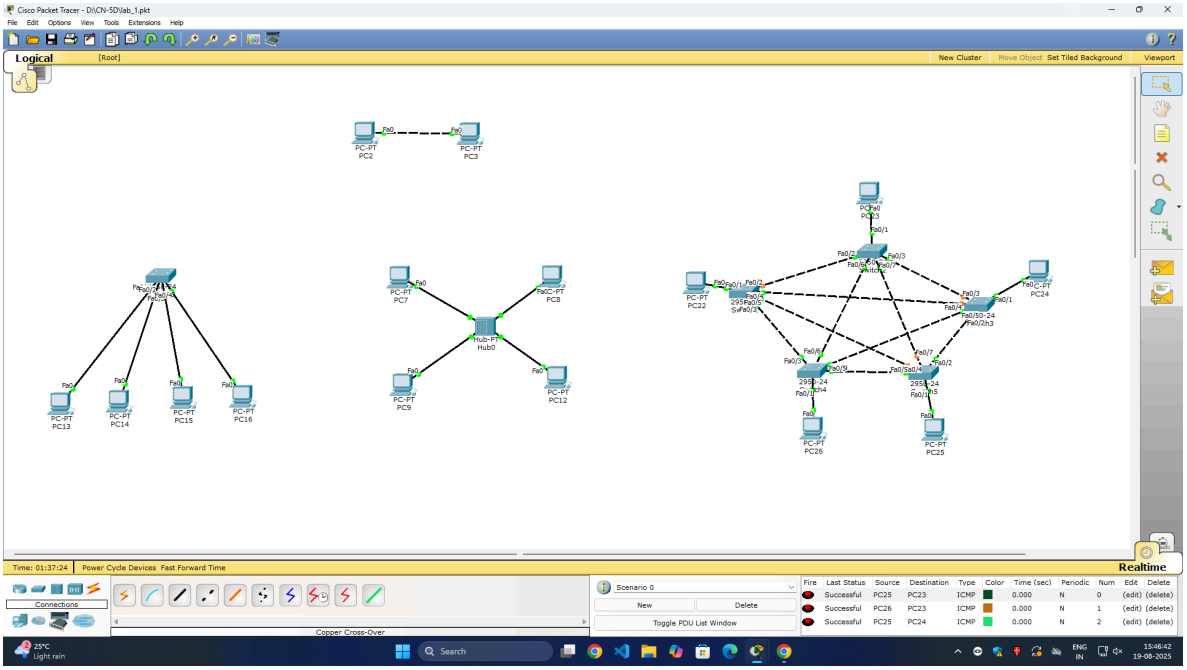




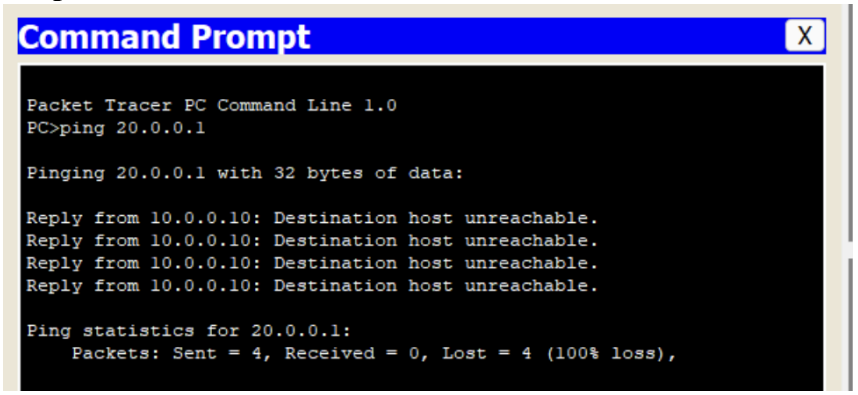
Program 4

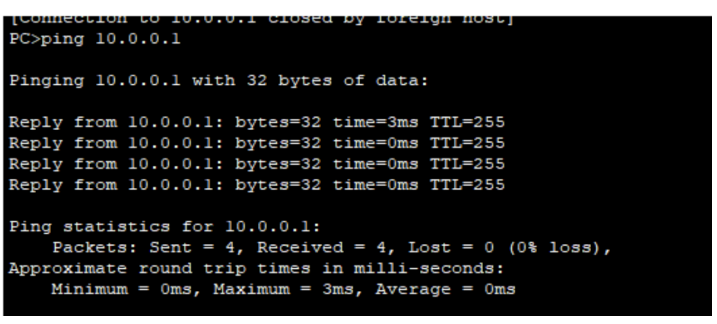
Aim: Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply

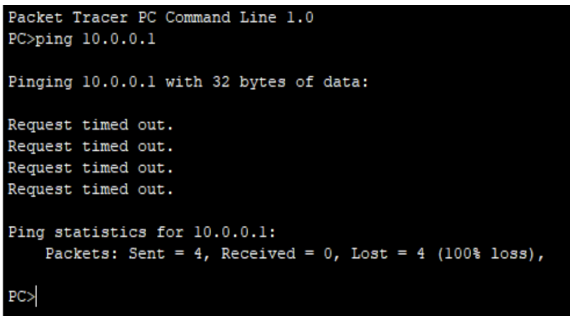
Topology:



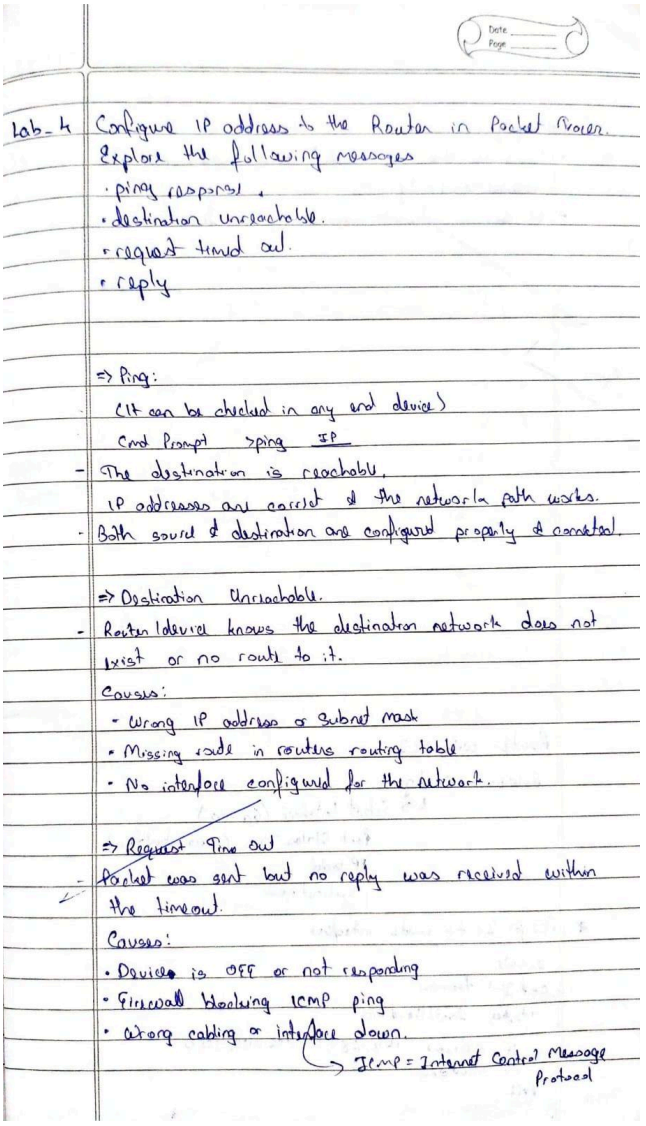
Output:







Observation:



Lab-4 Configure IP address to the Router in Packet Tracer. Explore the following messages
- ping response
- destination unreachable.
- request timed out.
- reply

=> Ping:
(It can be checked in any end device)
Cmd Prompt >ping IP
- The destination is reachable.
IP addresses are correct & the network path works.
- Both source & destination are configured properly & connected.

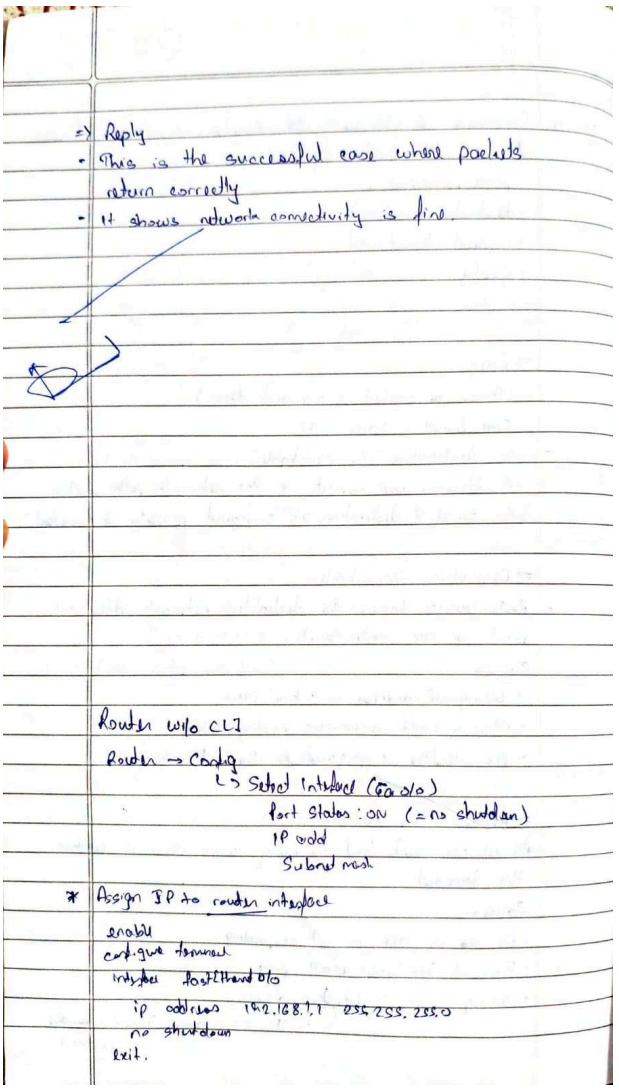
=> Destination Unreachable.
- Router/device knows the destination network does not exist or no route to it.

Causes:
- Wrong IP address or Subnet mask
- Missing route in routers routing table
- No interface configured for the network.

=> Request Time out
- Packet was sent but no reply was received within the timeout.

Causes:
- Device is OFF or not responding
- Firewall blocking ICMP ping
- Wrong cabling or interface down.

ICMP = Internet Control Message Protocol



=> Reply
- This is the successful case where packets return correctly
- It shows network connectivity is fine.

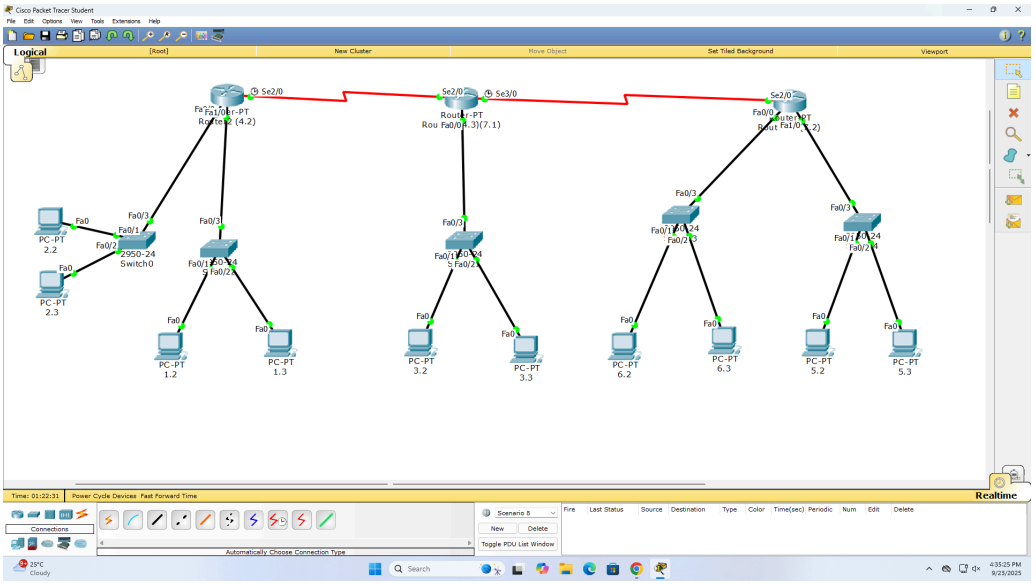
Router w/o CLI
Router → Config
↳ Select interface (Fa 0/0)
Port Status: ON (= no shutdown)
IP add
Subnet mask

* Assign IP to router interface
enable
configure terminal
interface FastEthernet 0/0
ip address 192.168.1.1 255.255.255.0
no shutdown
exit.

Program 5

Aim: Configure default route, static route to the Router

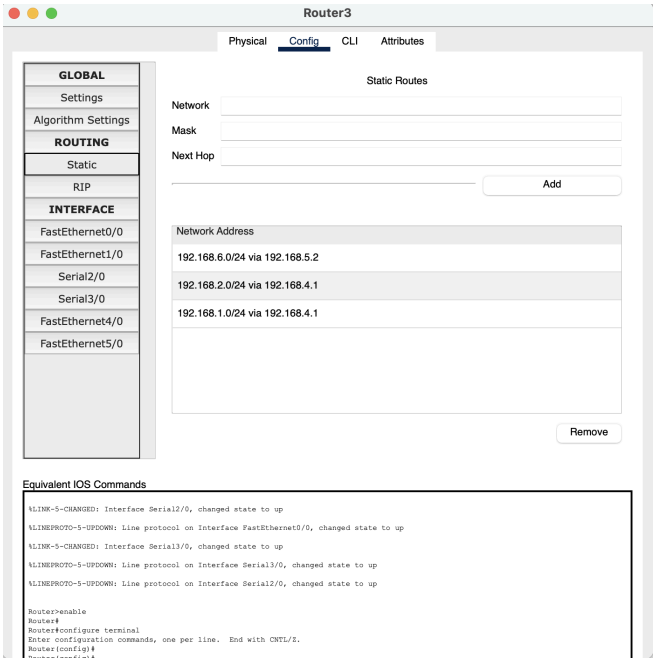
Topology:



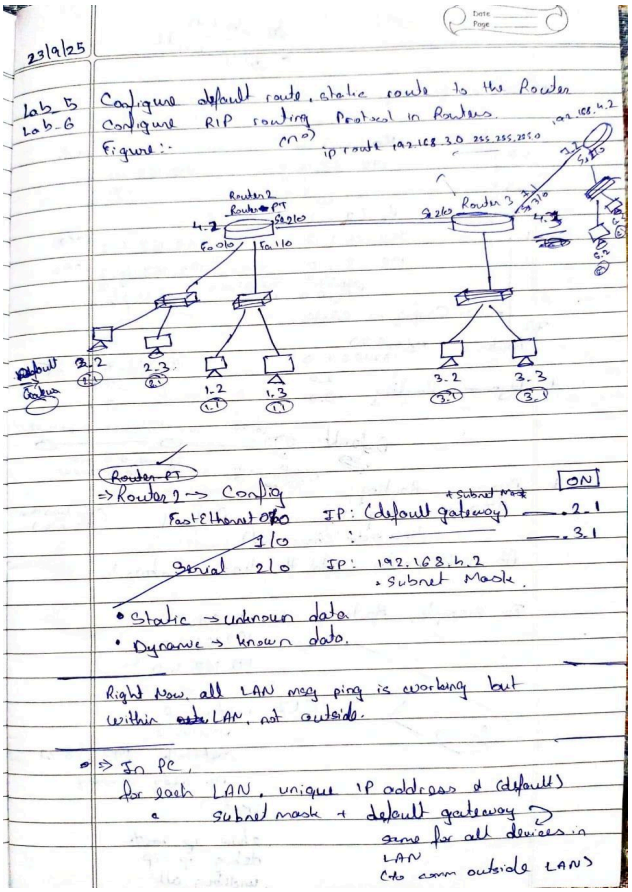
Procedure:

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

Output:



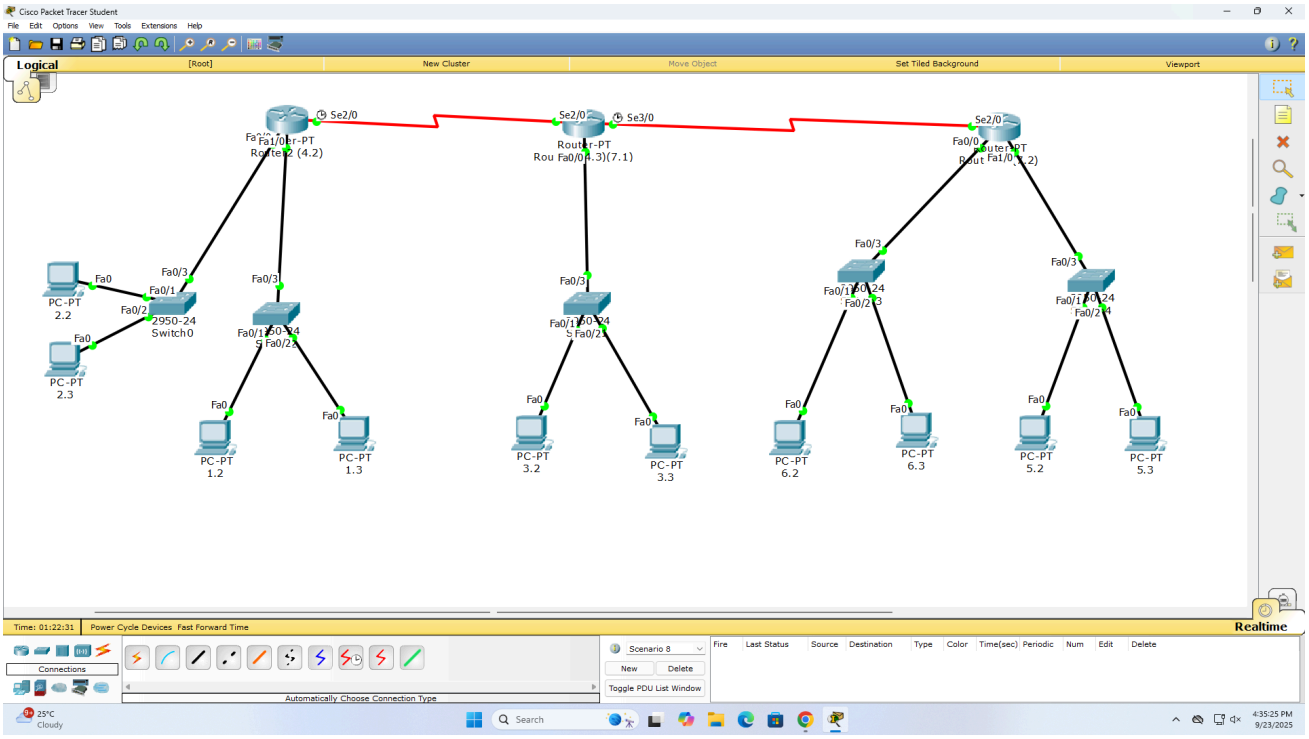
Observation:



Program 6

Aim: Configure RIP routing Protocol in Routers

Topology:



Procedure:

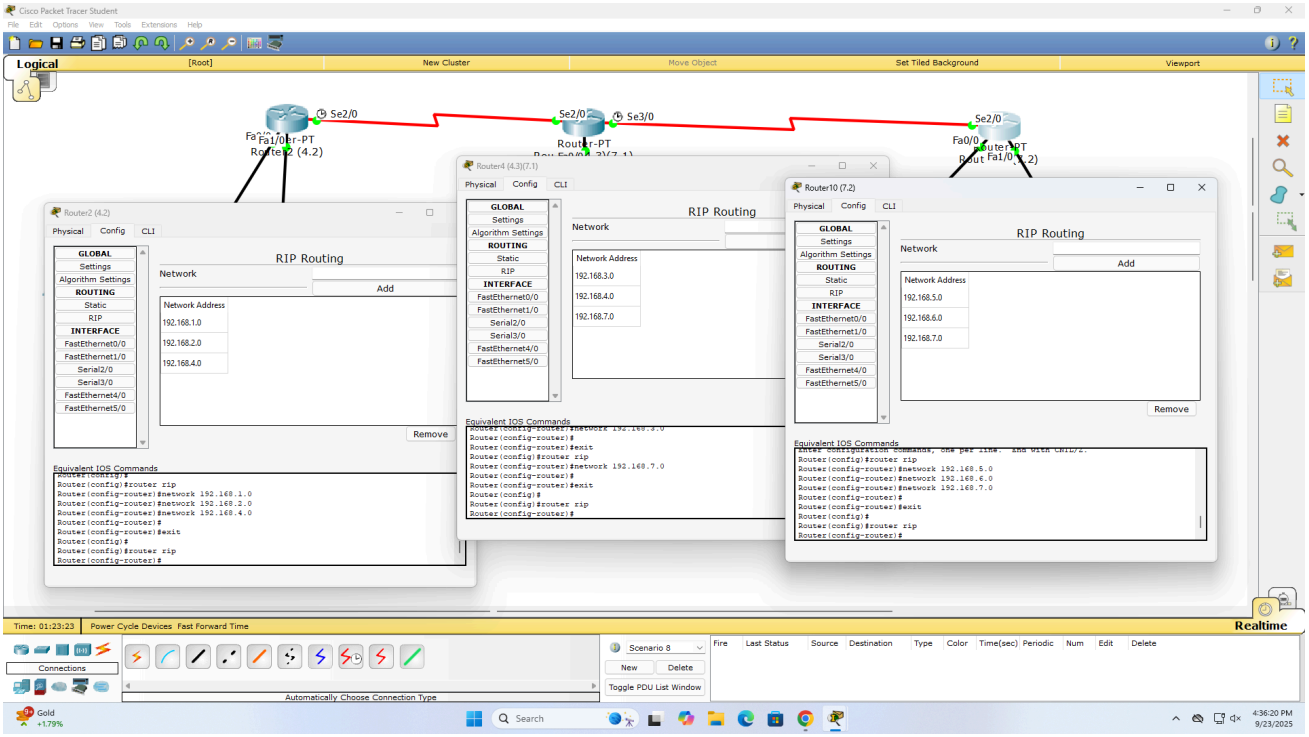
1. Configure Interfaces:

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

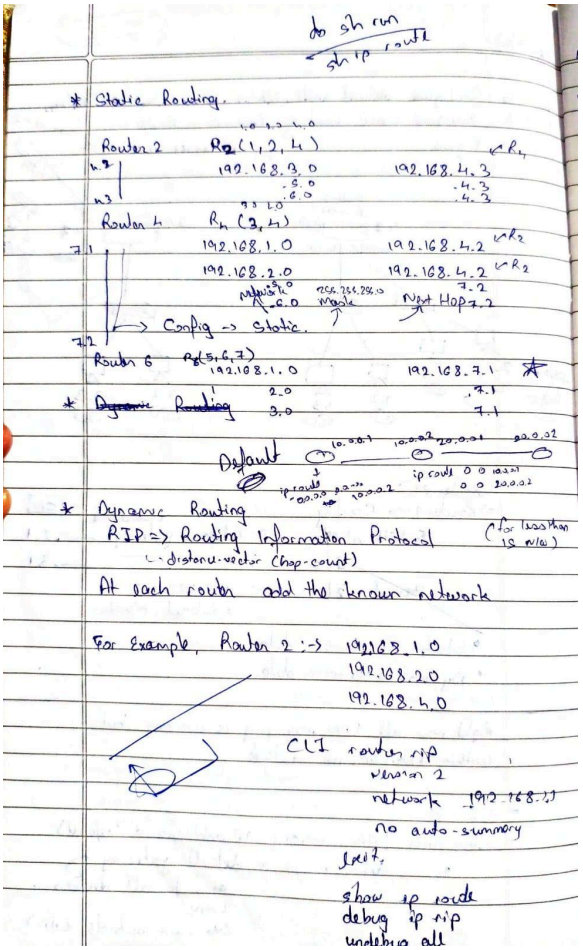
2. Configure RIP Routing:

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

Output:



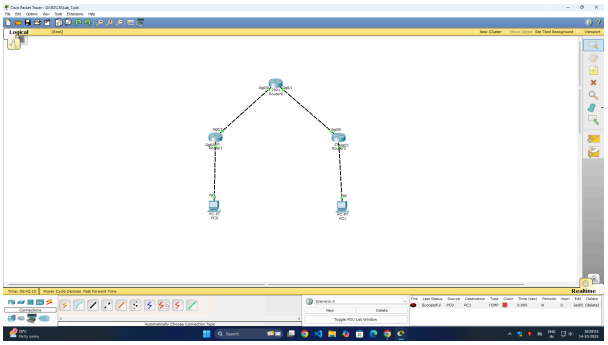
Observation:



Program 7

Aim: Configure OSPF routing protocol

Topology:



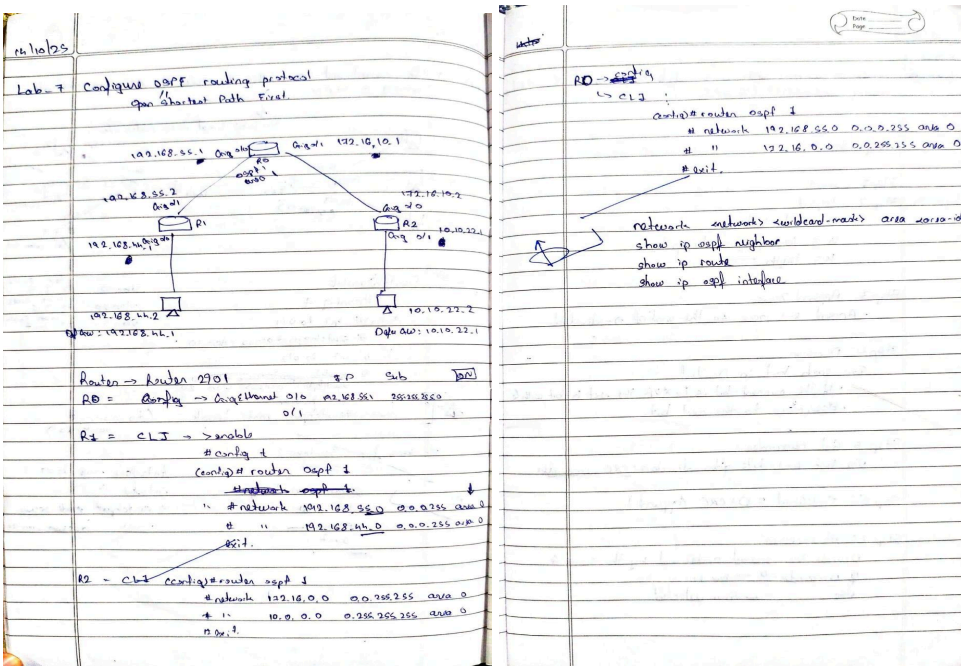
Procedure:

Router> enable
Router# conf t
Router1(config)# router ospf 1
Router1(config-router)# network 192.168.44.0 0.0.0.255 area 0
Router1(config-router)# exit
Router1(config)# exit
Router1# write memory

Output:

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

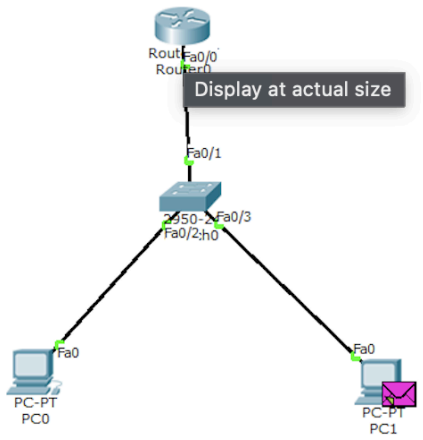
Observation:



Program 8

Aim: Demonstrate the TTL/ Life of a Packet

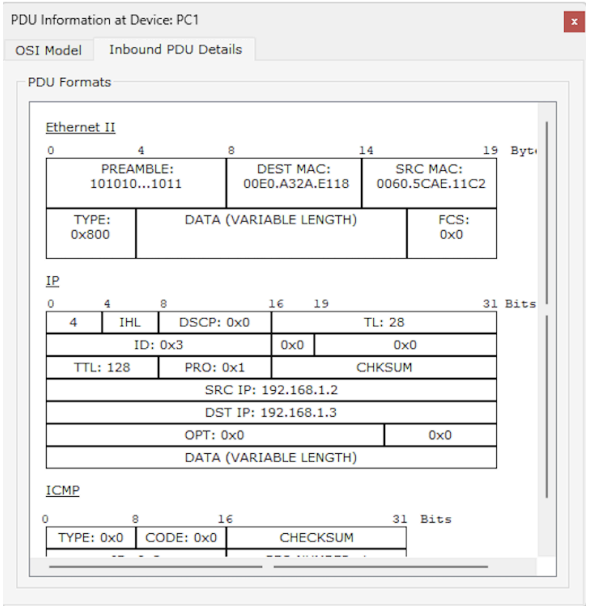
Topology:

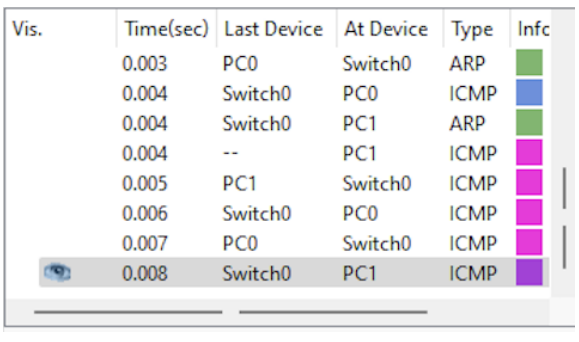


Procedure:

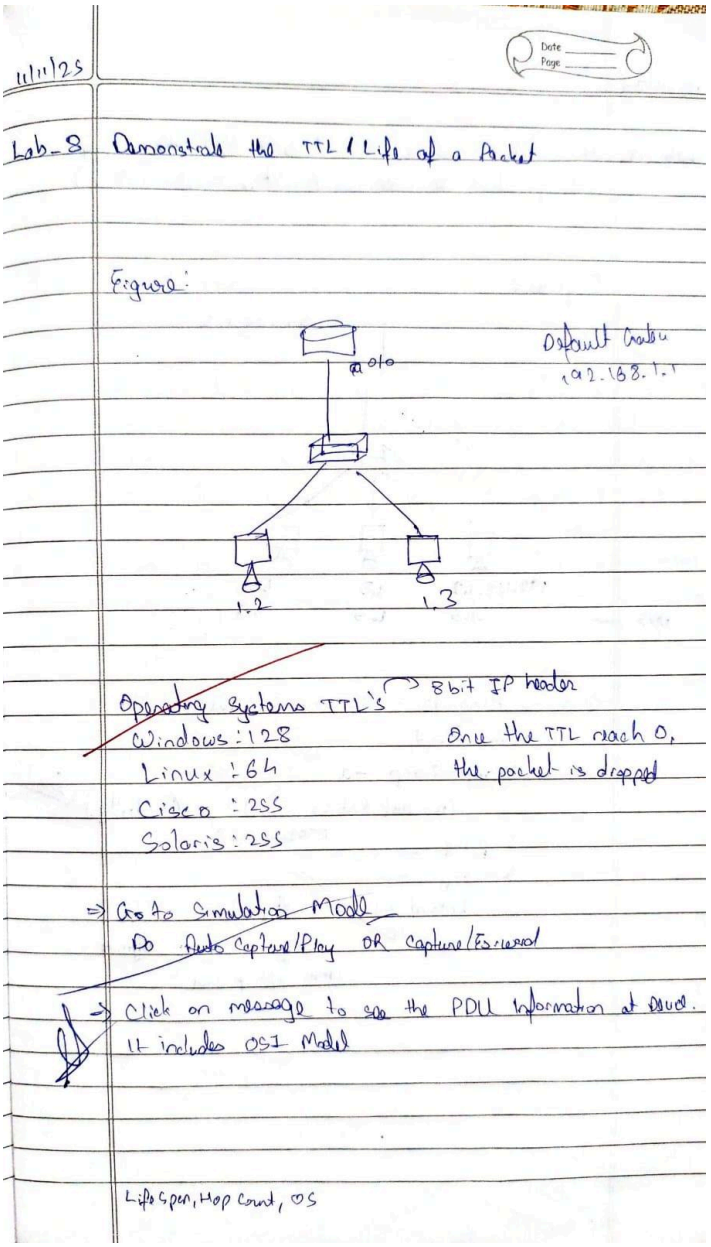
1. Enable Packet Capture:
    - Right-click on a link between devices
    - Select "Capture / Forward"
    - Send ping from PC19 to PC20
2. Analyze TTL in Captured Packets:
    - Open "Event List" in Simulation mode
    - Double-click any ICMP packet
    - Go to "Inbound PDU Details" or "Outbound PDU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

Output:





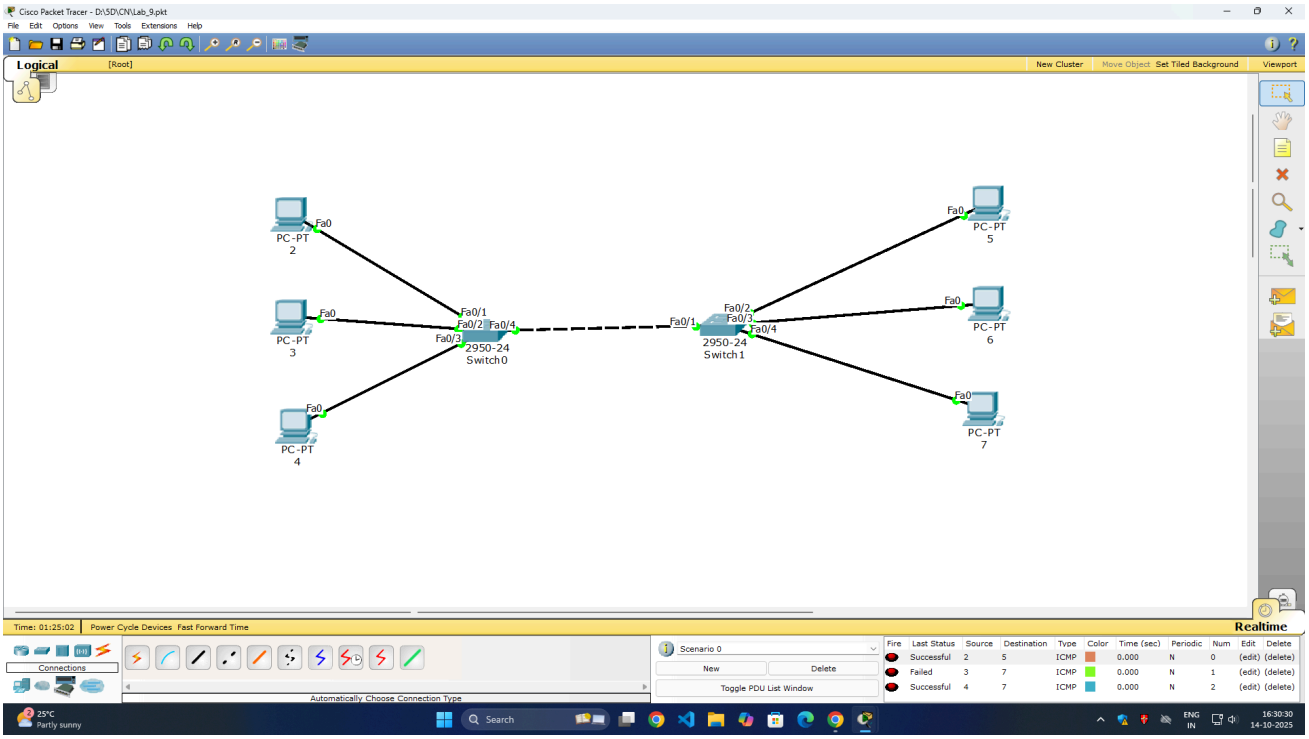
Observation:



Program 9

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

Topology:



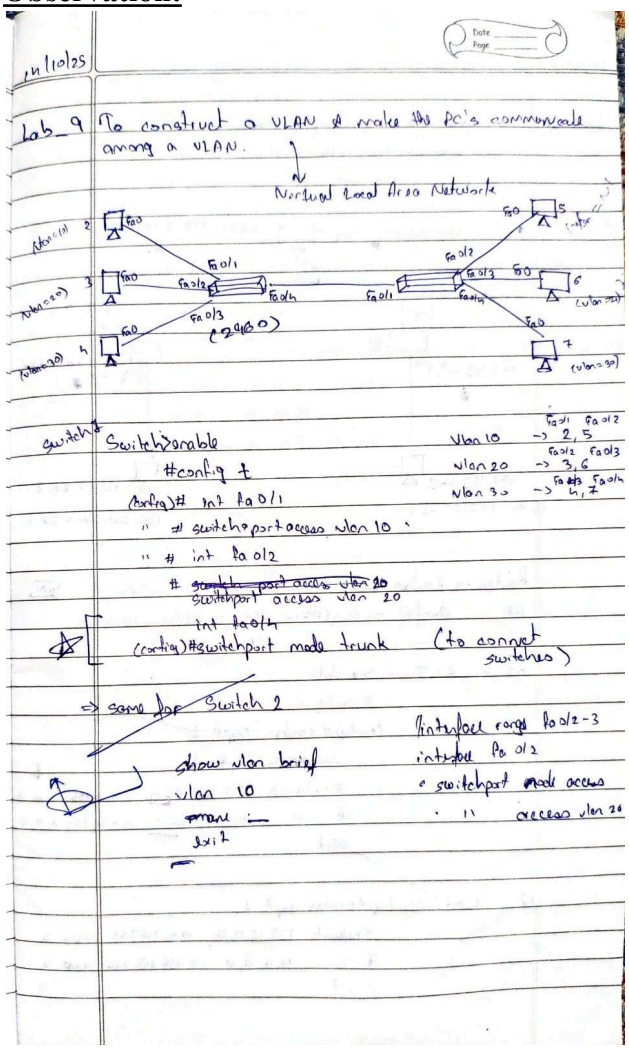
Procedure:

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

Output:

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

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

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/4, changed state to up

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

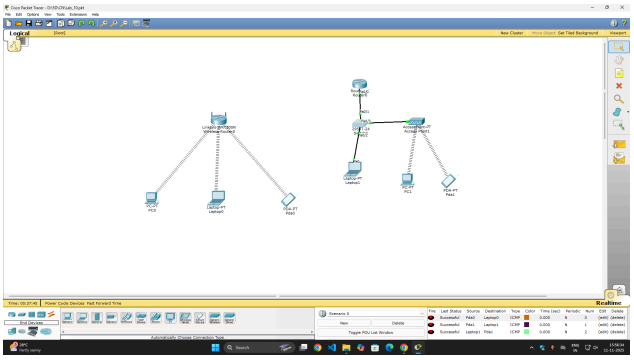
Observation:



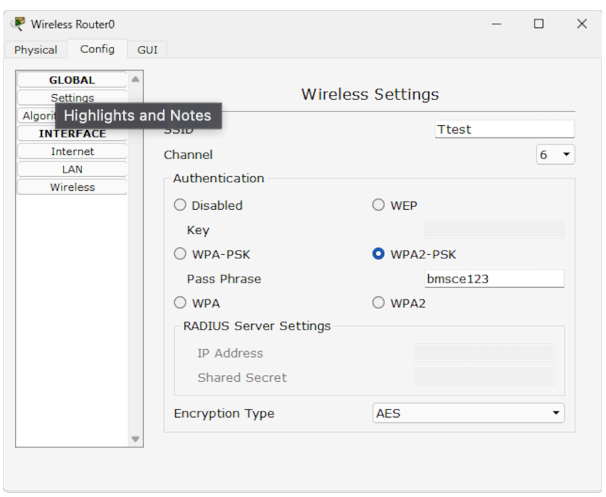
Program 10

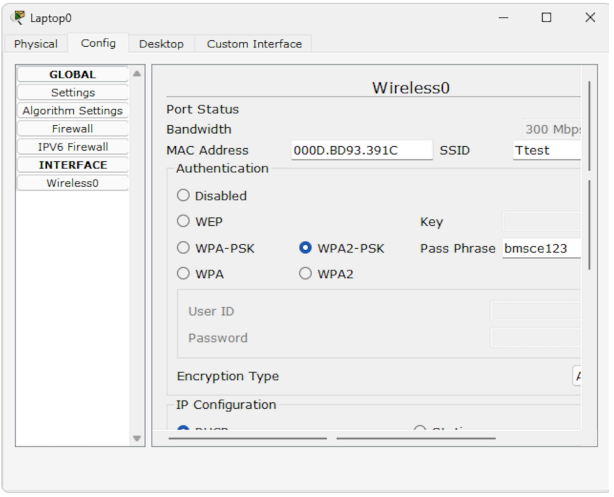
Aim: To construct a WLAN and make the nodes communicate wirelessly

Topology:

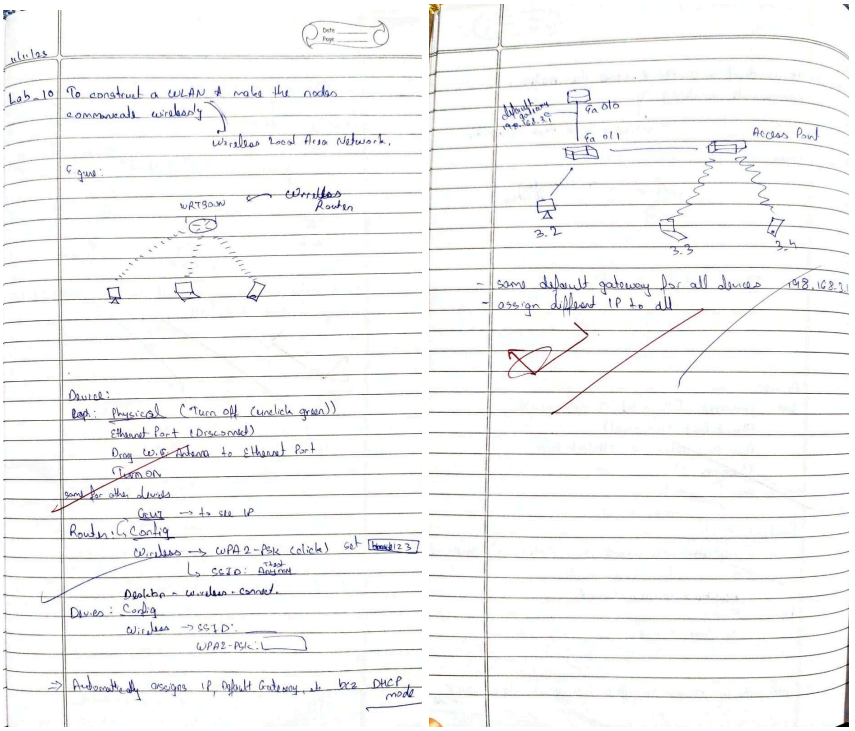


Output:





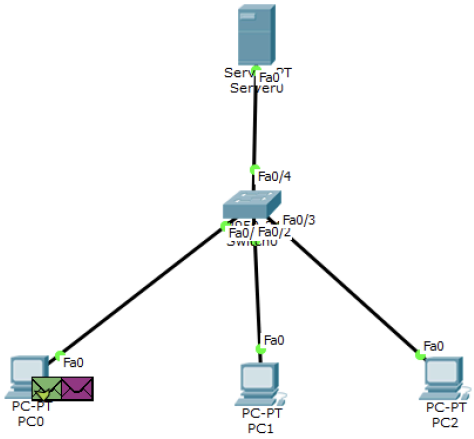
Observation:



Program 11

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

Topology:



Procedure:

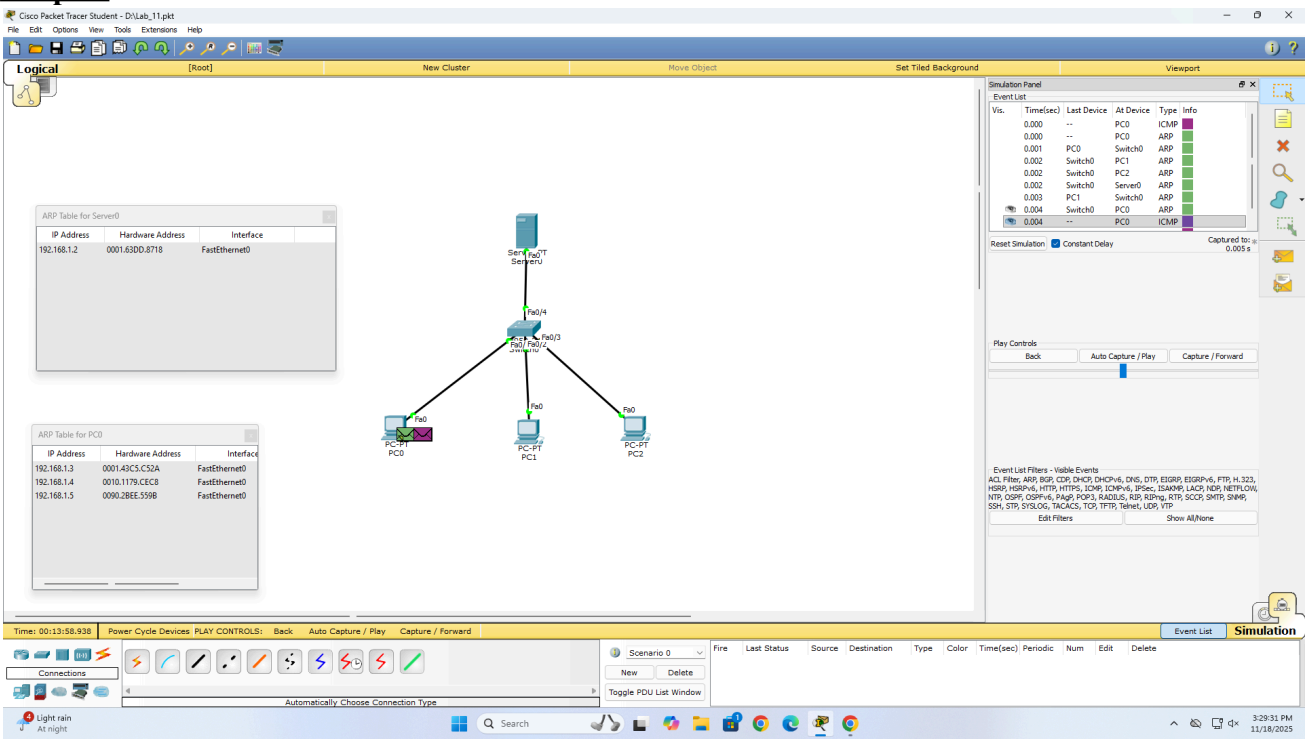
ping 192.168.1.20

arp -a

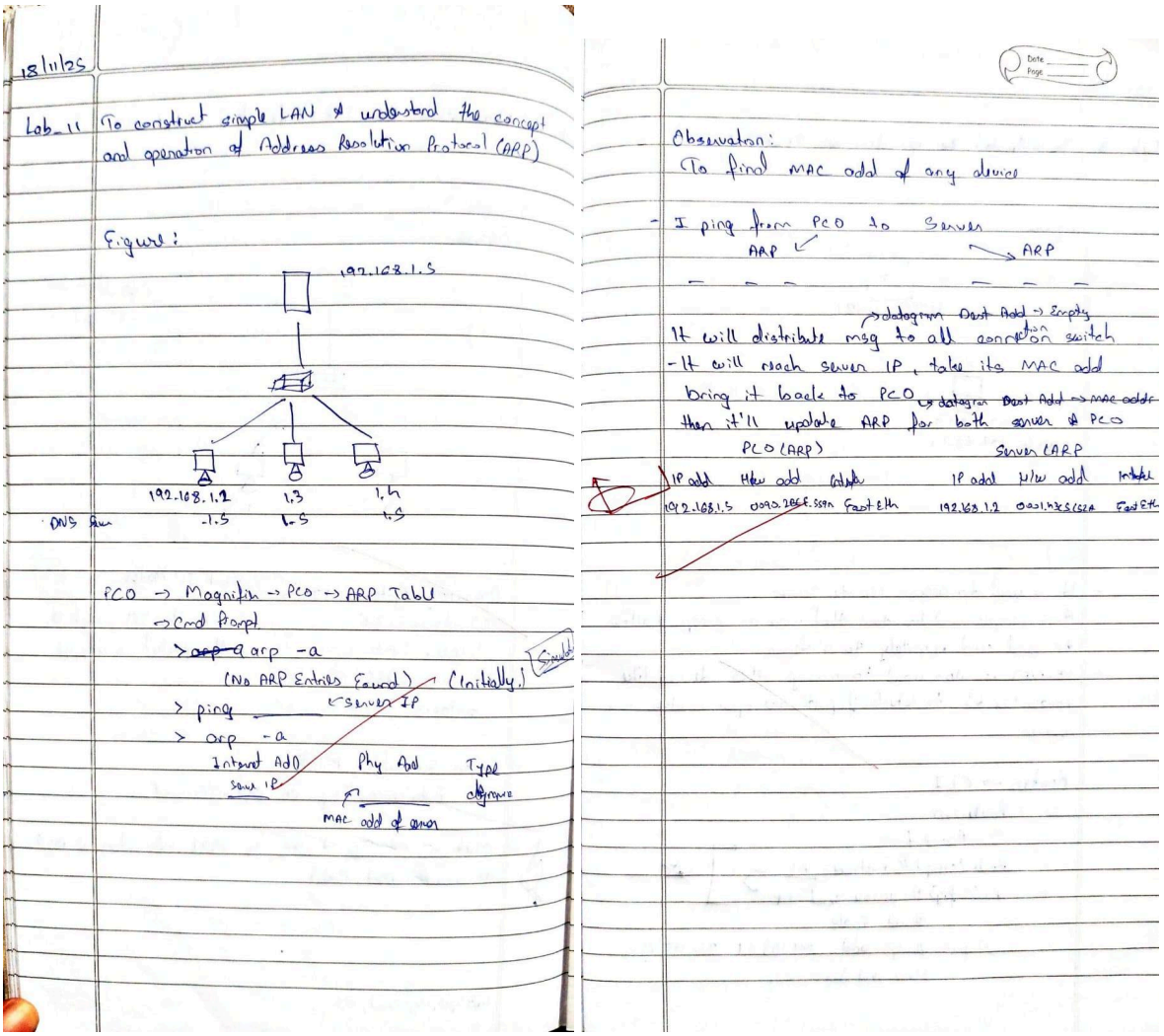
Interface: 192.168.1.10

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

Output:



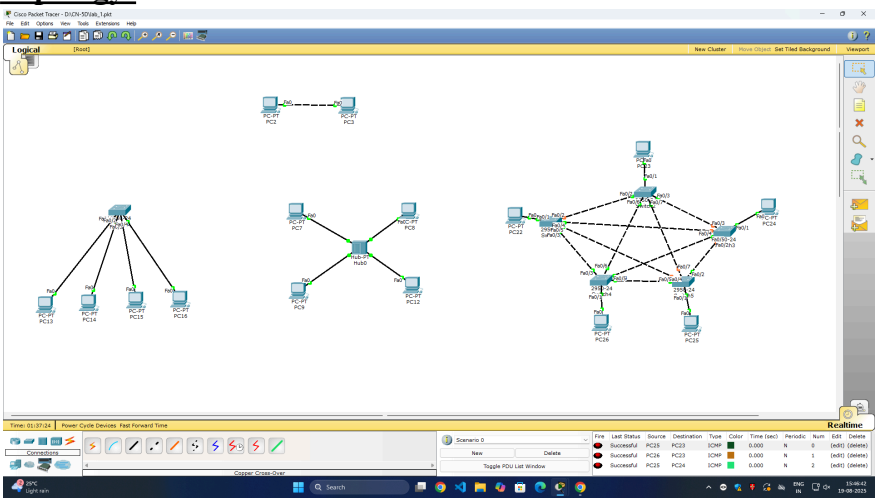
Observation:



Program 12

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

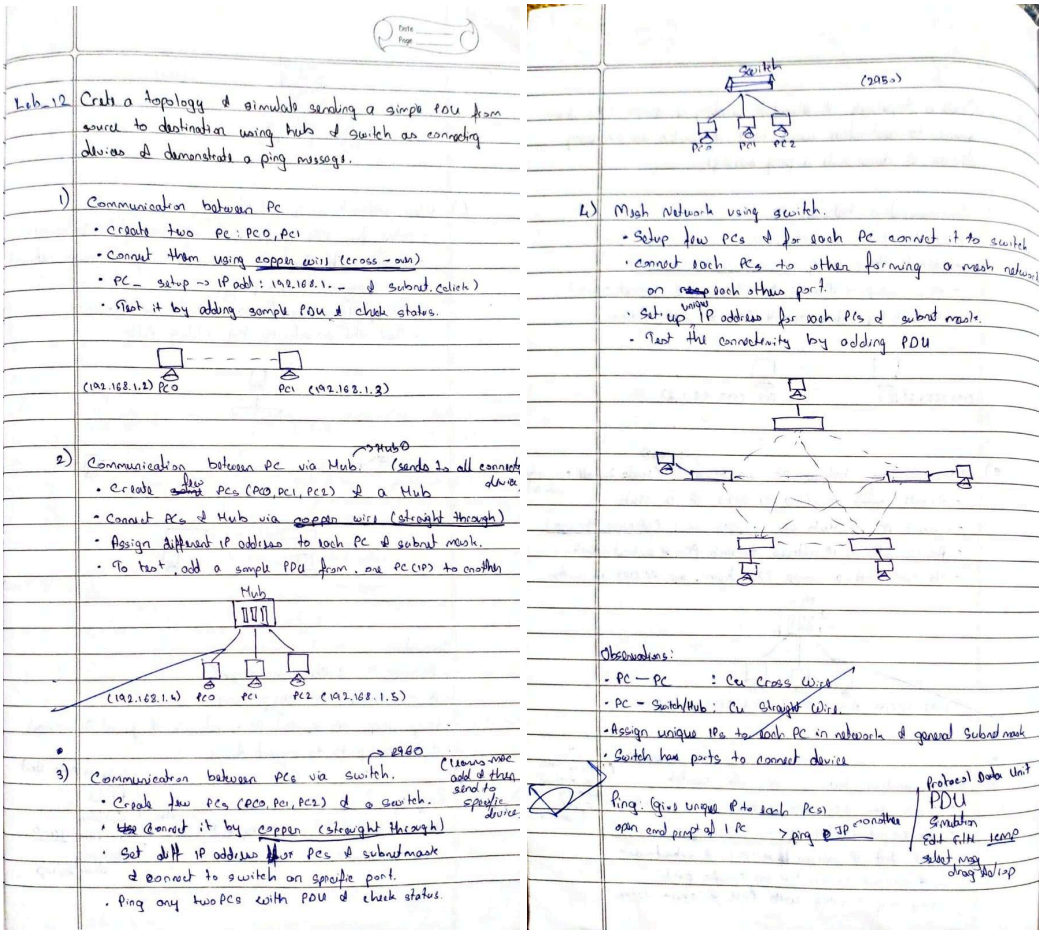
Topology:



Output:

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

Observation:



Program 14

Aim: Write a program for congestion control using Leaky bucket algorithm.

Code:

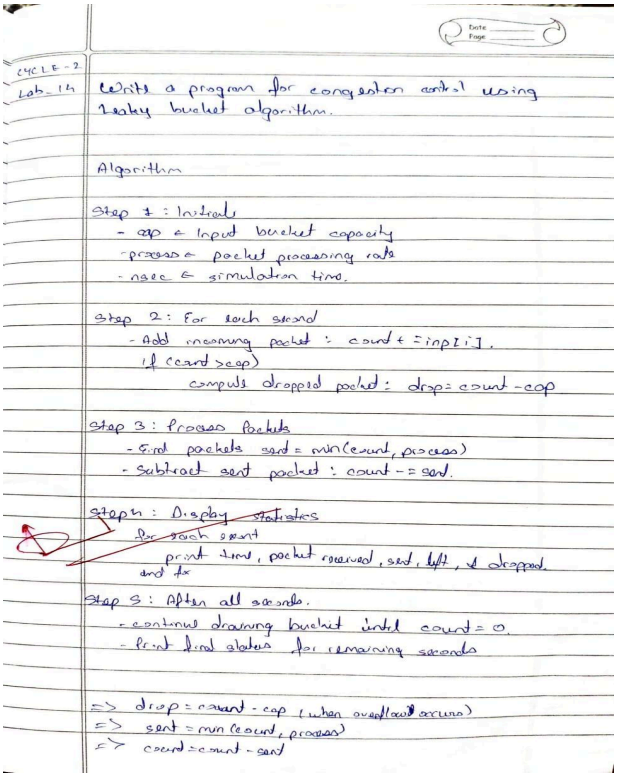
```
#include <stdio.h>
int min(int x, int y) {
   if (x < y)
      return x;
   else
      return y;
}
int main() {
   int drop = 0, mini, nsec, cap, count = 0, i, inp[25], process;
   printf("Enter the bucket size:\n");
   scanf("%d", &cap);
   printf("Enter the processing rate:\n");
   scanf("%d", &process);
   printf("Enter the number of seconds you want to simulate:\n");
   scanf("%d", &nsec);
   for (i = 0; i < nsec; i++) {
      printf("Enter the size of the packet entering at %d sec:\n", i + 1);
      scanf("%d", &inp[i]);
   }
   printf("\nSecond | Packet Received | Packet Sent | Packet Left | Dropped\n");
   printf("-----------------------------------------------------------\n");
   for (i = 0; i < nsec; i++) {
      count += inp[i];
      if (count > cap) {
         drop = count - cap;
         count = cap;
      }
      printf("%d\t %d\t\t", i + 1, inp[i]);
      mini = min(count, process);
      printf("%d\t\t", mini);
      count = count - mini;
      printf("%d\t\t %d\n", count, drop);
      drop = 0;
   }
   for (; count != 0; i++) {
      if (count > cap) {
         drop = count - cap;
         count = cap;
      }
      printf("%d\t 0\t\t", i + 1);
      mini = min(count, process);
      printf("%d\t\t", mini);
      count = count - mini;
      printf("%d\t\t %d\n", count, drop);
      drop = 0;
   }
   return 0;
```

}

Output:

```
Enter the bucket size:
10
Enter the processing rate:
4
Enter the number of seconds you want to simulate:
5
Enter the size of the packet entering at 1 sec:
3
Enter the size of the packet entering at 2 sec:
7
Enter the size of the packet entering at 3 sec:
4
Enter the size of the packet entering at 4 sec:
6
Enter the size of the packet entering at 5 sec:
5

Second | Packet Received | Packet Sent | Packet Left | Dropped
---------------------------------------------------------------
1         3               3             0             0
2         7               4             3             0
3         4               4             3             0
4         6               4             5             0
5         5               4             6             0
```

Observation:



CYCLE - 2

Lab - 14

Write a program for congestion control using leaky bucket algorithm.

Algorithm

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

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

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

Step 4 : Display ~~statistics~~
for each sent
    print time, packet received, sent, left, & dropped.
end for

Step 5 : After all seconds.
- continue draining bucket until count = 0.
- Print final status for remaining seconds

=> drop = count - cap (when overflow occurs)
=> sent = min(count, process)
=> count = count - sent

Program 15

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

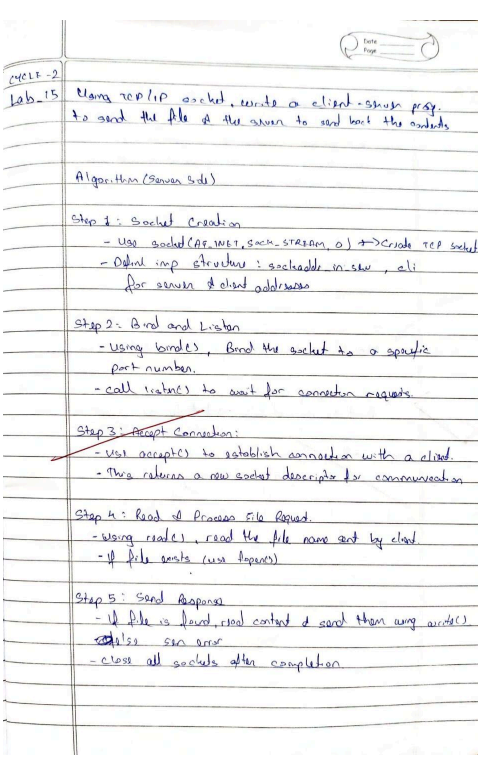
Algorithm:

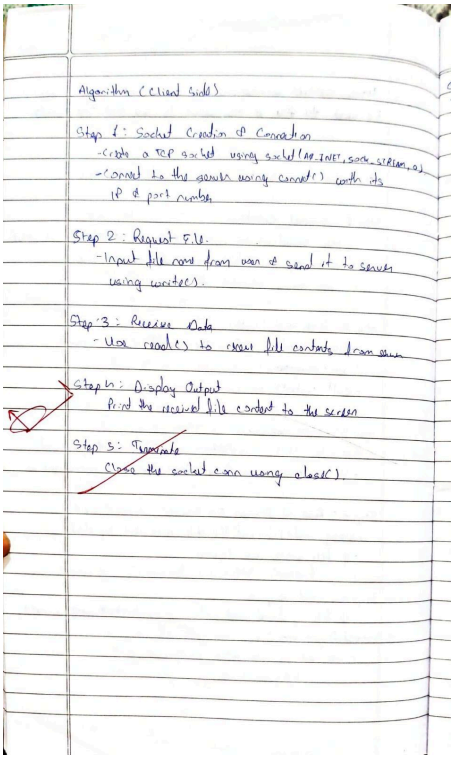
```
# tcp_client.py
import socket
# Step 1: Create TCP socket

client_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
# Step 2: Connect to server

client_socket.connect(('localhost', 8080))
# Step 3: Send filename

filename = input("Enter filename to request: ")
client_socket.send(filename.encode())

# Step 4: Receive file contents

data = client_socket.recv(4096).decode()
print("\n--- File Content ---\n") print(data

# Step 5: Close connection client_socket.close()
```

```
# tcp_server.py import socket

# Step 1: Create a TCP socket server_socket =
socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Step 2: Bind to address and port
server_socket.bind(('localhost', 8080))
# Step 3: Listen for client connections server_socket.listen(1)
print("Server is listening on port 8080...")
# Step 4: Accept connection
conn, addr = server_socket.accept() print("Connected by:",
addr)
# Step 5: Receive file name filename =
conn.recv(1024).decode().strip()
try:
        # Step 6: Open and read file with open(filename, 'r') as
        f:
                data = f.read()
        conn.send(data.encode()) # Send file contents
except FileNotFoundError: conn.send(b"File not found on
server.")
# Step 7: Close connection conn.close()
server_socket.close()
```

Output:

Observation:





Program 16

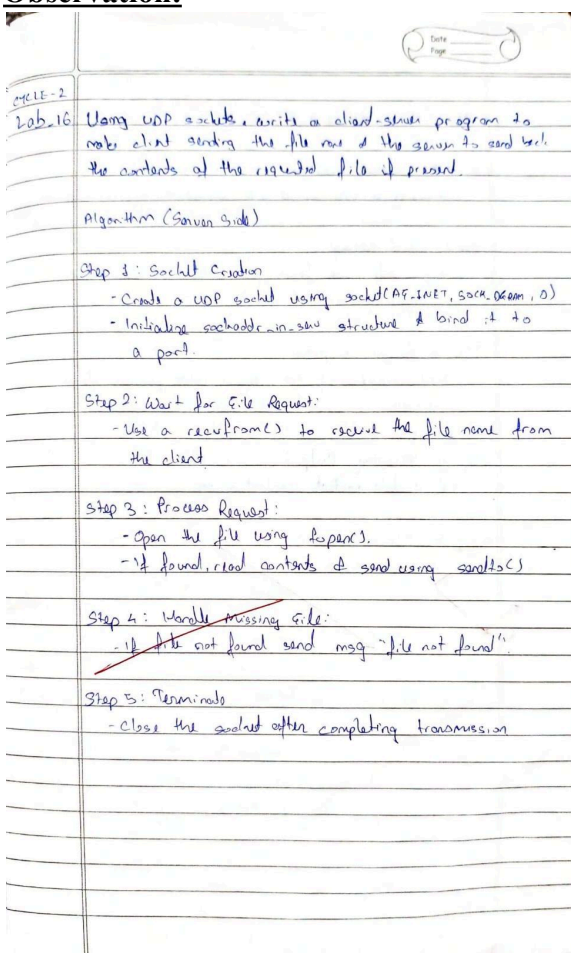
Aim: Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

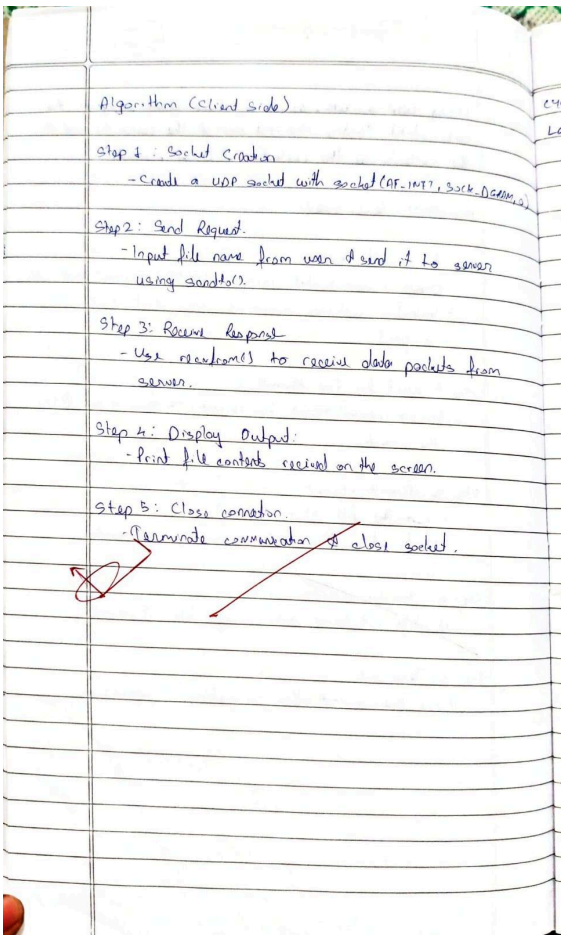
Algorithm:

```
# tcp_client.py
import socket
# Step 1: Create TCP socket

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Step 2: Connect to server

client_socket.connect(('localhost', 8080))
# Step 3: Send filename

filename = input("Enter filename to request: ")
client_socket.send(filename.encode())
# Step 4: Receive file contents

data = client_socket.recv(4096).decode()
print("\n--- File Content ---\n") print(data

# Step 5: Close connection client_socket.close()
```

```
# tcp_server.py import socket

# Step 1: Create a TCP socket server_socket =
socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Step 2: Bind to address and port server_socket.bind(('localhost',
8080))
# Step 3: Listen for client connections server_socket.listen(1)
print("Server is listening on port 8080...")
# Step 4: Accept connection
conn, addr = server_socket.accept() print("Connected by:", addr)
# Step 5: Receive file name filename =
conn.recv(1024).decode().strip()
try:
        # Step 6: Open and read file with open(filename, 'r') as f:
                data = f.read()

        conn.send(data.encode()) # Send file contents

except FileNotFoundError: conn.send(b"File not found on
server.")

# Step 7: Close connection conn.close() server_socket.close()
```

Observation:





Program 17

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

Code:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main() {
   char rem[50], a[50], s[50], c, msj[50], gen[30];
   int i, genlen, t, j, flag = 0, k, n;
   printf("Enter the generation polynomial:\n");
   gets(gen);
   printf("Generator polynomial is CRC-CCITT: %s\n", gen);
   genlen = strlen(gen);
   k = genlen - 1;
   printf("Enter the message:\n");
   n = 0;
   while ((c = getchar()) != '\n') {
      msj[n] = c;
      n++;
   }
   msj[n] = '\0';

   for (i = 0; i < n; i++)
      a[i] = msj[i];

   for (i = 0; i < k; i++)
      a[n + i] = '0';
   a[n + k] = '\0';
   printf("\nMessage polynomial appended with zeros:\n");
   puts(a);
   for (i = 0; i < n; i++) {
      if (a[i] == '1') {
         t = i;
         for (j = 0; j <= k; j++) {
            if (a[t] == gen[j])
               a[t] = '0';
            else
               a[t] = '1';
            t++;
         }
      }
   }

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

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

   printf("\nMessage with checksum appended:\n");
   for (i = 0; i < n; i++)
      a[i] = msj[i];

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

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

   for (i = 0; i < n; i++) {
      if (s[i] == '1') {
         t = i;
         for (j = 0; j <= k; j++, t++) {
            if (s[t] == gen[j])
               s[t] = '0';
            else
```

```
                s[t] = '1';          }
        }
    }

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

    for (i = 0; i < k; i++) {
        if (rem[i] == '1')
            flag = 1;
    }
    if (flag == 0)
        printf("Received polynomial is error-free \n");
    else
        printf("Received polynomial contains error \n");

    return 0;
}
```

Output:

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

Message polynomial appended with zeros:
1101010101010100 00
Checksum (remainder):
11

Message with checksum appended:
1101010101010100 11
Enter the received message:
1101010101010100 11
Received polynomial is error-free

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

Observation: