VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

S Mohammed Aiaz (1BM24CS418)

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 **S Mohammed Aiaz (1BM24CS418),** 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.

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

Index

Part - A

Part - B

Github Link:

https://github.com/aiyaz99/Computer_Networks.git

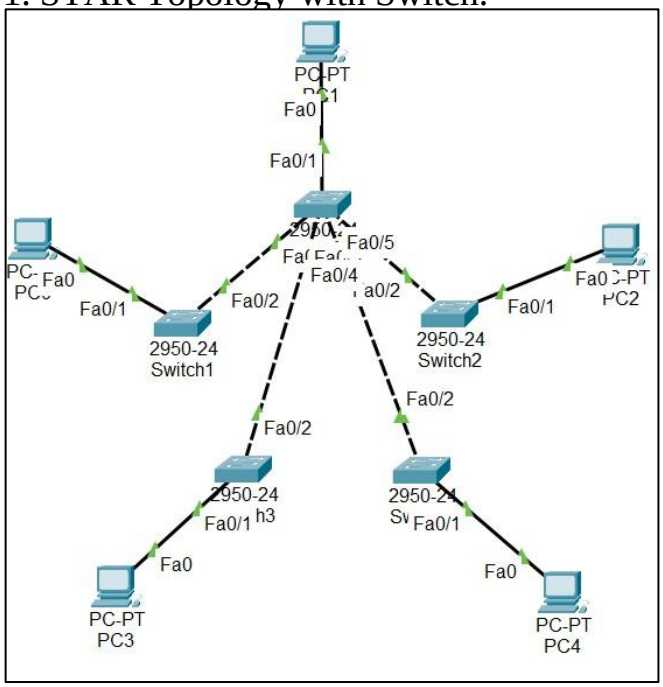
PART – A

Program – 1:

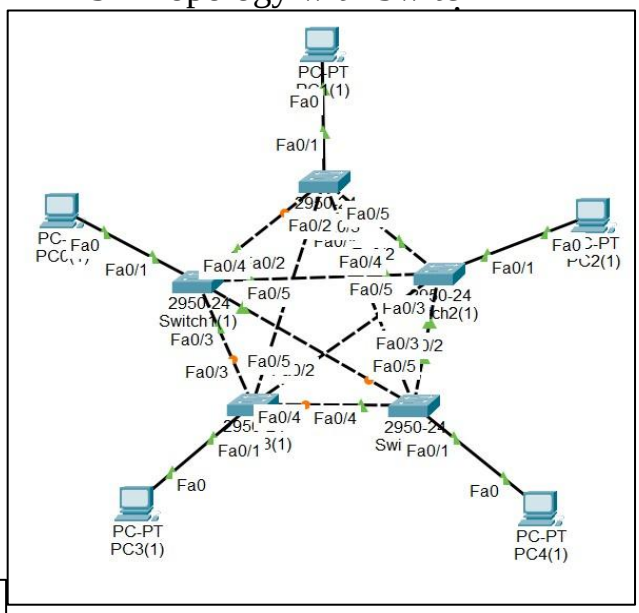
Aim: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message

Topology:

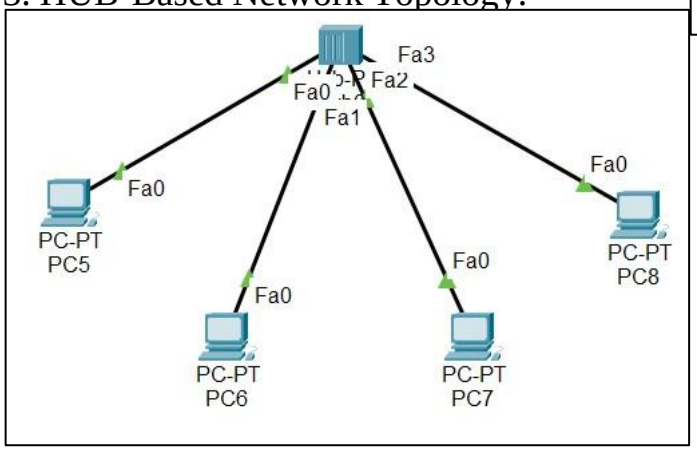
1. STAR Topology with Switch:



MESH Topology with Switch



3. HUB-Based Network Topology:



Configuration:

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

Star topology using switch:

1) Name: Implementation of star topology using Cisco packet tracer.

- Procedure:
  - * open a new project in packet tracer.
  2) Drag and drop: * Control switch (switch 0)
    * 4 edge switches (switch 2, switch 3, switch 4, switch 5)
    * 5 pcs (pc4, pc5, pc6, pc7, pc8)
  3) Connect each PC to its respective switch using Copper straight-through cables.
  4) Connect all edge switches to the Control switch
  5) Assign IP addresses: [click on pc icon/device → desktop → IP Configuration → enter IP address]
    * PC 4 → 192.168.1.2
    * PC 5 → 192.168.1.3
    * PC 6 → 192.168.1.4
    * PC 7 → 192.168.1.5
    * PC 8 → 192.168.1.6

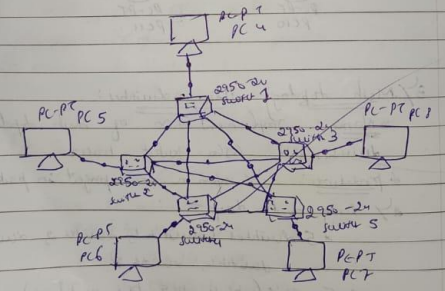
  and subnet mask: 255.255.255.0 is directly entered.

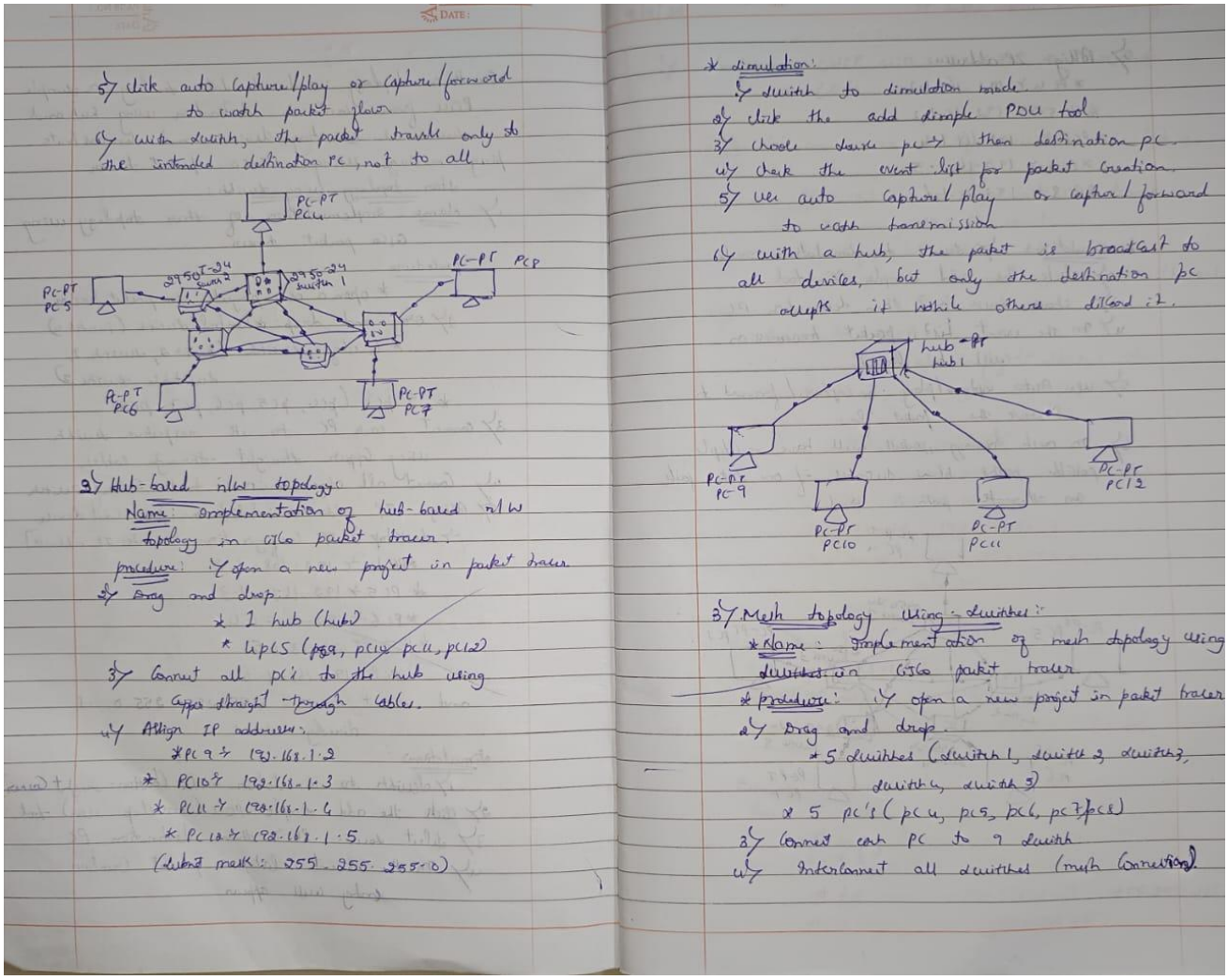
Simulation:

1) Switch to simulation mode (button - right corner)
2) Click the add simple PDU (envelop icon) tool
3) Select source PC → then destination PC.
4) In the Event list, a packet Creation entry will appear.

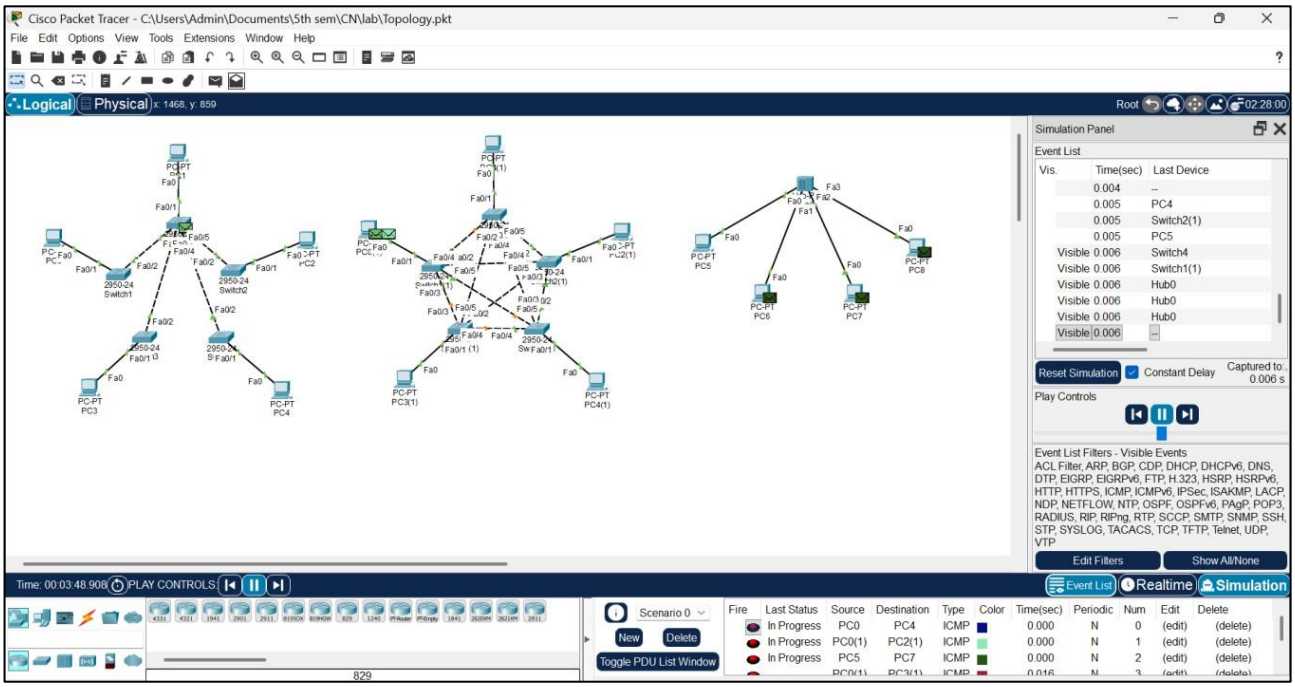
5) Assign IP address:
   * PC 4 → 192.168.1.2
   * PC 5 → 192.168.1.3
   * PC 6 → 192.168.1.4
   * PC 7 → 192.168.1.5
   * PC 8 → 192.168.1.6

* Simulation: 1) Switch to simulation mode
2) Click the add simple PDU tool
3) Choose source PC → then destination PC
4) In the event list, packet transmission will be flagged.
5) Use Auto capture/play or capture/forward to observe the packet flow.
6) In mesh topology, packets will have multiple possible paths like switches, if one link fails, an alternate path is used




5) click auto capture/play or capture/forward to watch packet flow.
6) with switch, the packet travels only to the intended destination PC, not to all.
PC-PT PC4
PC-PT PC5
PC-PT PC6
PC-PT PC7
PC-PT PC8
2950-24 Switch 2
2950-24 Switch 1
* simulation:
1) switch to simulation mode.
2) click the add simple PDU tool
3) choose source pc then destination pc.
4) check the event list for packet creation.
5) use auto capture/play or capture/forward to watch transmission
6) with a hub, the packet is broadcast to all devices, but only the destination pc accepts it while others discard it.
Hub-PT Hub1
PC-PT PC9
PC-PT PC10
PC-PT PC11
PC-PT PC12
3) Hub-based nlw topology:
Name: implementation of hub-based nlw topology in cisco packet tracer.
procedure: 1) open a new project in packet tracer
2) drag and drop
* 1 hub (hub0)
* 4 pcs (pc9, pc10, pc11, pc12)
3) connect all pc's to the hub using copper straight-through cables.
4) Assign IP addresses:
* PC9 → 192.168.1.2
* PC10 → 192.168.1.3
* PC11 → 192.168.1.4
* PC12 → 192.168.1.5
(subnet mask: 255.255.255.0)
3) Mesh topology using switches:
* Name: Implementation of mesh topology using switches in cisco packet tracer
* procedure: 1) open a new project in packet tracer
2) drag and drop.
* 5 switches (switch1, switch2, switch3, switch4, switch5)
* 5 pc's (pc4, pc5, pc6, pc7, pc8)
3) connect each pc to a switch
4) interconnect all switches (mesh connection)

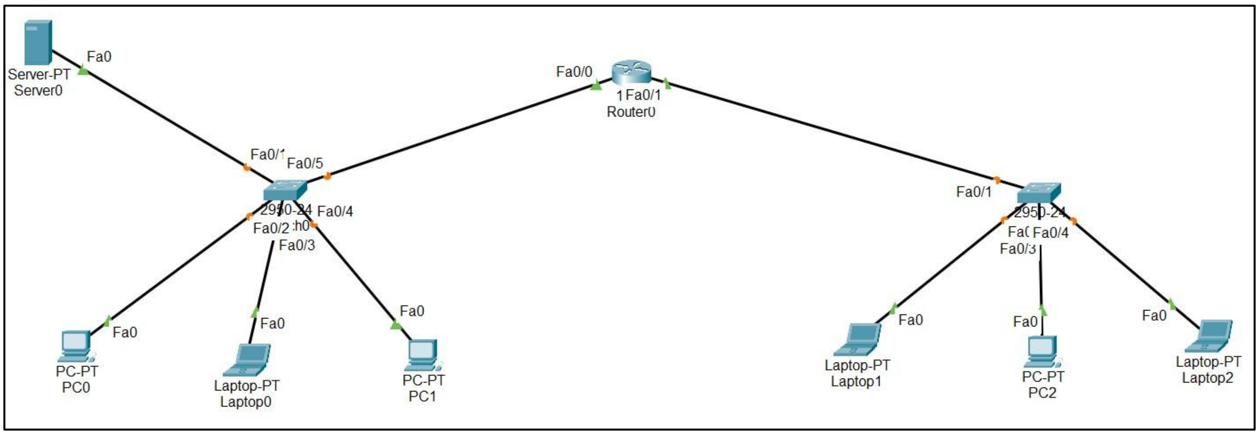
Output:


Cisco Packet Tracer - C:\Users\Admin\Documents\5th sem\CN\lab\Topology.pkt
File Edit Options View Tools Extensions Window Help
Logical Physical
Simulation Panel
Event List
Vis. Time(sec) Last Device
0.004 --
0.005 PC4
0.005 Switch2(1)
0.005 PC5
Visible 0.006 Switch4
Visible 0.006 Switch1(1)
Visible 0.006 Hub0
Visible 0.006 Hub0
Visible 0.006 Hub0
Visible 0.006 --
Reset Simulation
Constant Delay
Captured to: 0.006 s
Play Controls
Event List Filters - Visible Events
ACL Filter, ARP, BGP, CDP, DHCP, DHCPv6, DNS, DTP, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, RADIUS, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters
Show All/None
Time: 00:03:48.908 PLAY CONTROLS:
Event List
Realtime
Simulation
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time(sec) Periodic Num Edit Delete
In Progress PC0 PC4 ICMP 0.000 N 0 (edit) (delete)
In Progress PC0(1) PC2(1) ICMP 0.000 N 1 (edit) (delete)
In Progress PC5 PC7 ICMP 0.000 N 2 (edit) (delete)
829

Program 2:

Aim: Configure DHCP within a LAN and outside LAN.

Network diagram:



Configuration:

9/9/25 IP helper address also relay agent

Q. Configure DHCP within a LAN and outside a LAN

Step 1: click on server and go to desktop and click on IP Config
give IP : 192.168.10.2, subnet is automatic
give gateway : 192.168.10.1

Step 2: Go to config click DHCP select poolname as switch one and gateway as 192.168.10.1

Step 3:
Server → DHCP
→ desktop Config → Static →
192.168.10.2, gateway : 192.168.10.1

Services:
→ DHCP ON ☑
pool name : switch one
gateway : 192.168.10.1
start IP add : 192.168.10.3
subnet mask : 255.255.255.0
add ✓

switch two Test ✓
Gateway : 192.168.20.1
start IP add : 192.168.20.2
subnet mask ; 255.255.255.0 → add

Router → CLI → No
Router # enable
Config t
int Fa0/1
ip address 192.168.10.1 255.255.255.0
ip helper-address 192.168.10.2

Output:

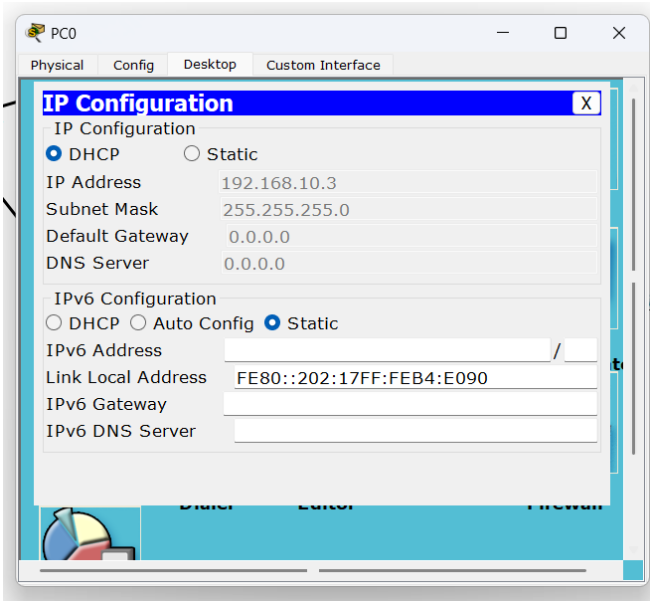


Fig 1. Ip address assigned by DHCP server within Lan (PC0)

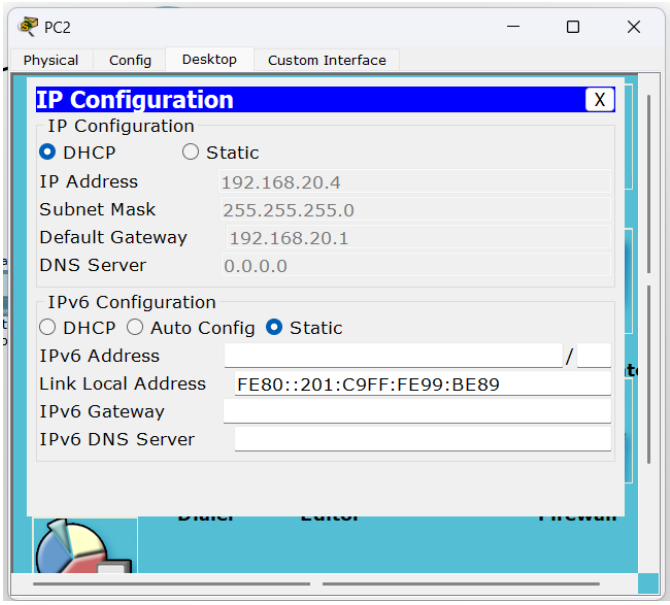
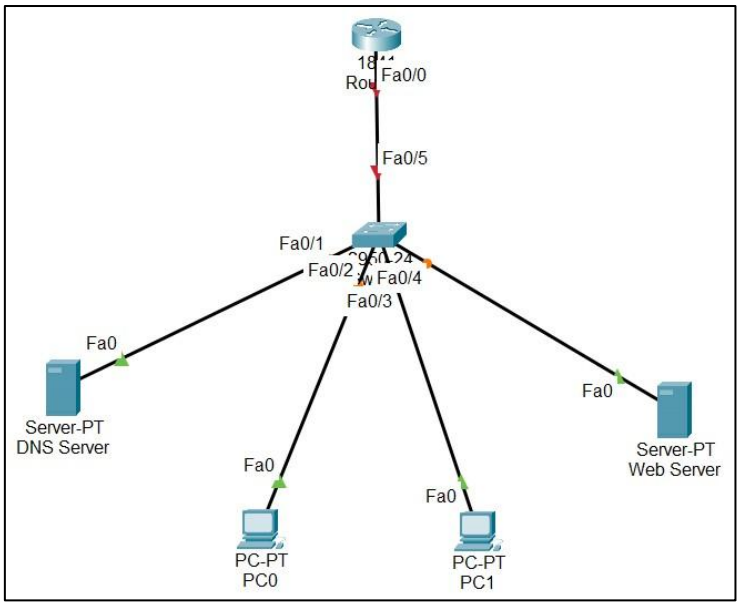


Fig 2. Ip address assigned by DHCP server outside Lan (PC2)

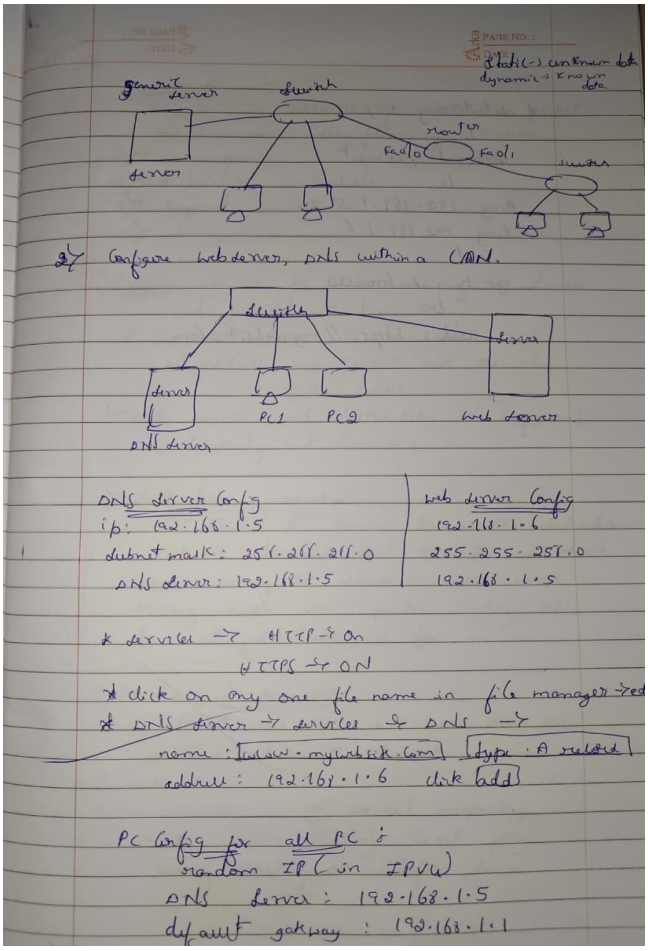
Program 3:

Aim: Configure Web Server, DNS within a LAN.

Network diagram:



Configuration:



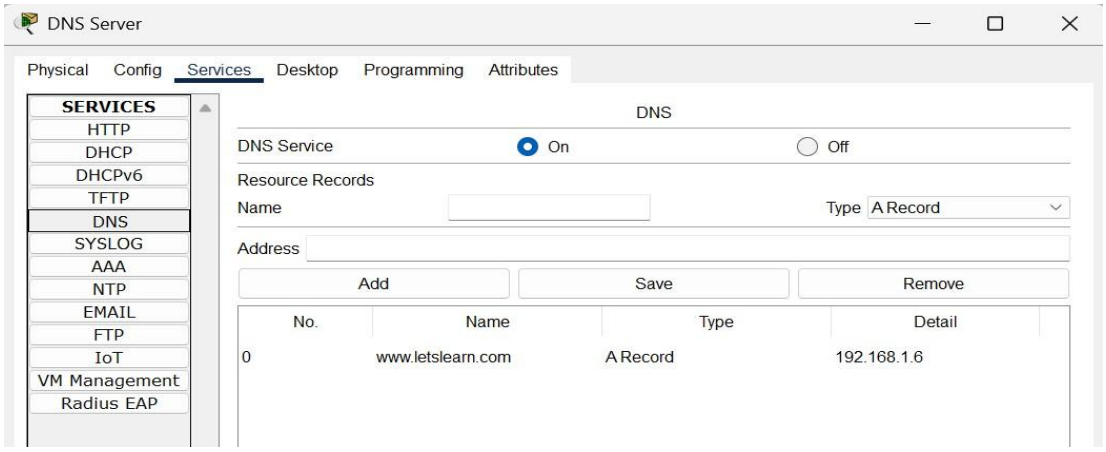


Fig 1. DNS server – DNS Services

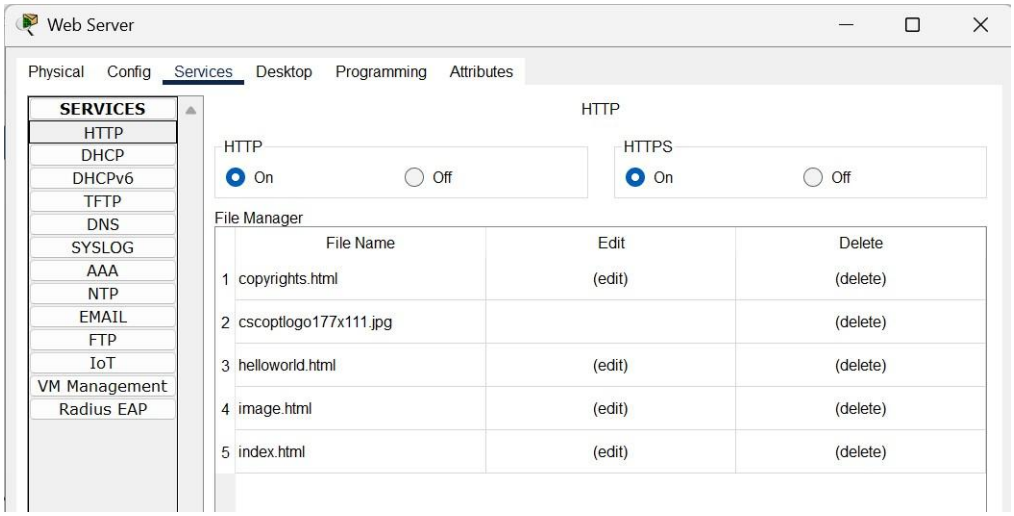


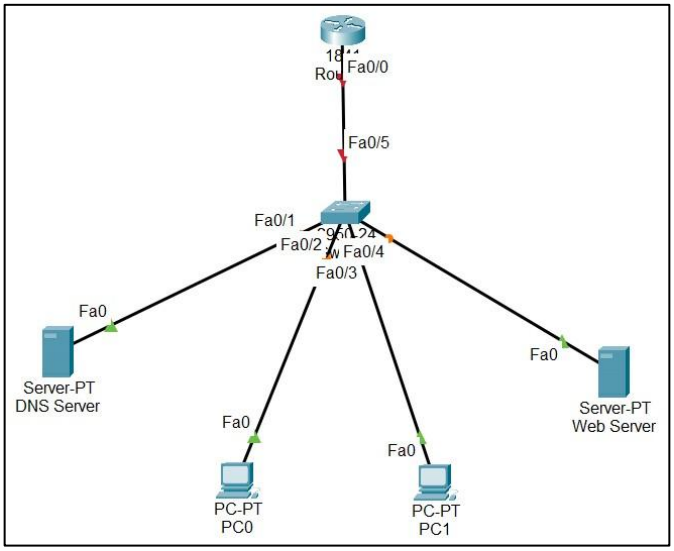
Fig 2. WEB server – HTTP Services

Fig 3. PC0 – accessing data from web browser

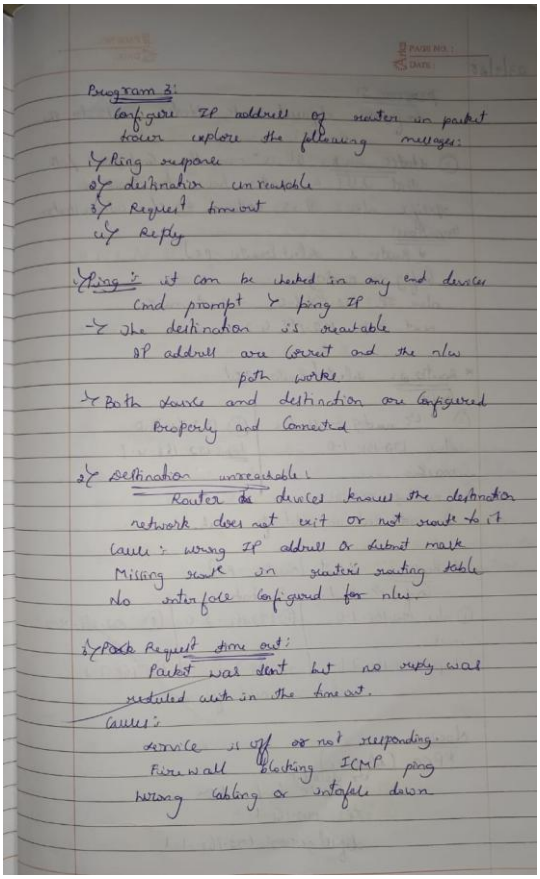
Program 4:

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

Network diagram:



Configuration:



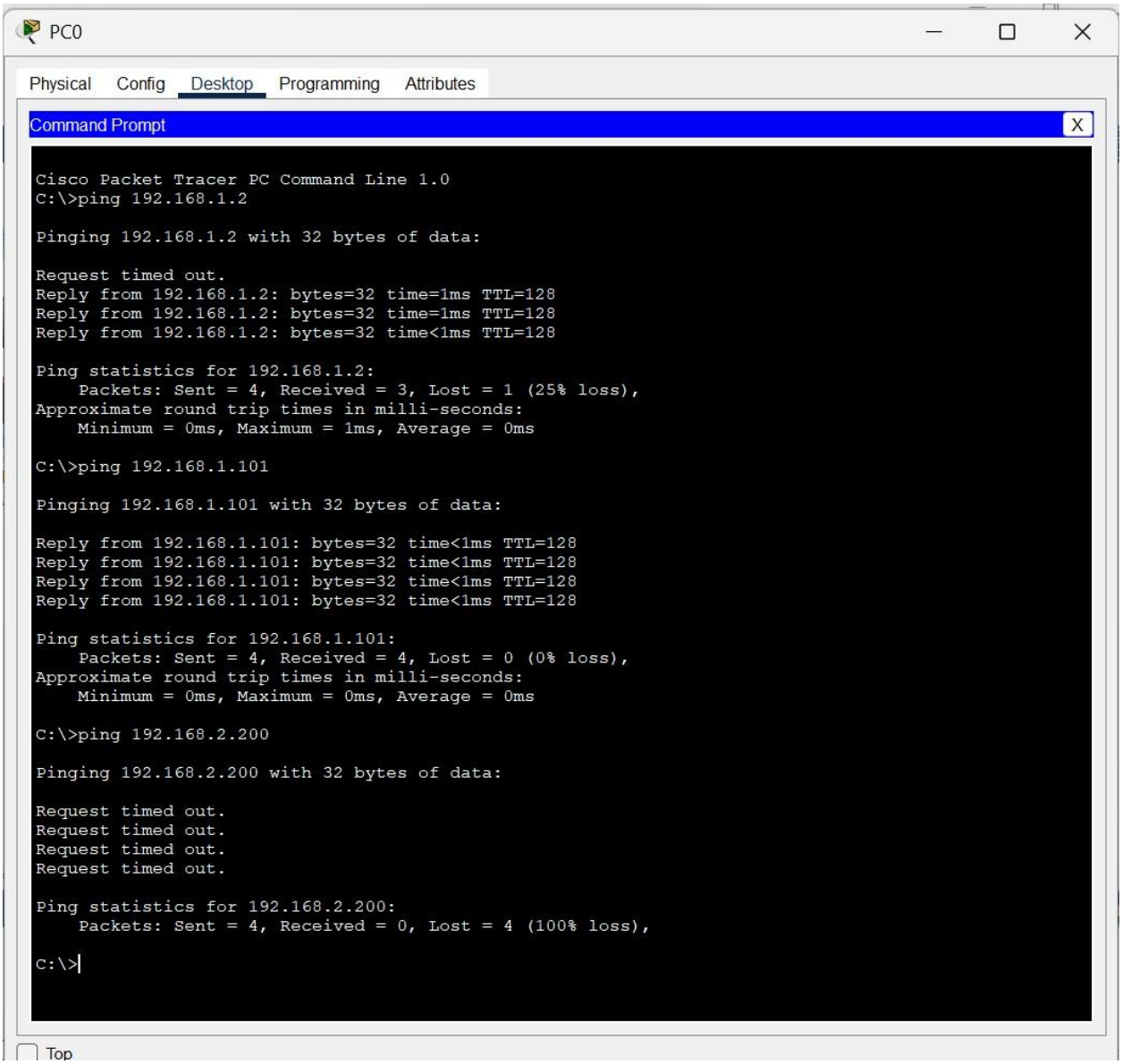
Program 3:
Configure IP address of router in packet tracer explore the following messages:
1) Ping response
2) destination unreachable
3) Request timeout
4) Reply

1) Ping :- it can be checked in any end device
Cmd prompt → ping IP
→ The destination is reachable
IP address are correct and the n/w path works
→ Both source and destination are configured properly and connected

2) Destination unreachable:
Router or device knows the destination network does not exit or not route to it
Cause: wrong IP address or subnet mask
Missing route in router's routing table
No interface configured for n/w

3) Request time out:
Packet was sent but no reply was received with in the time out.
Cause:
device is off or not responding
Firewall blocking ICMP ping
Wrong cabling or interface down

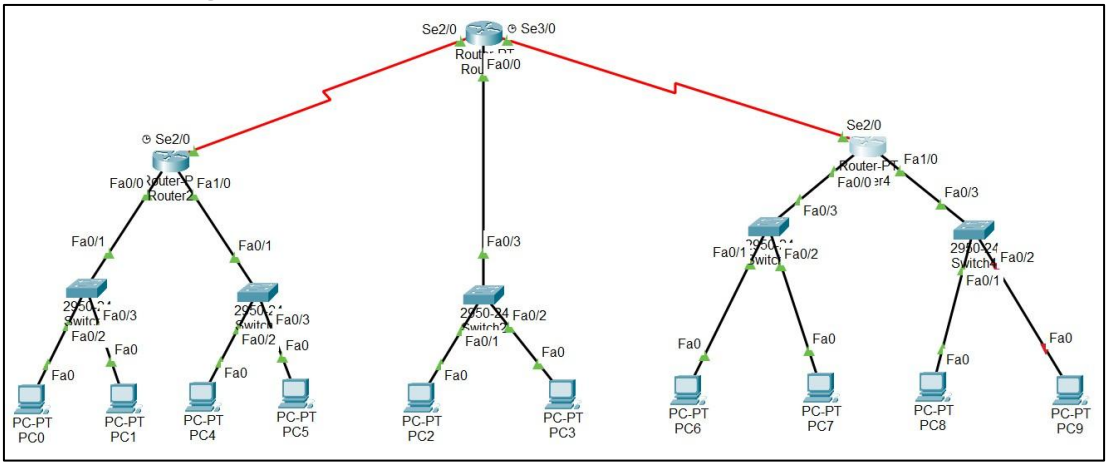
Output:


PC0
Physical
Config
Desktop
Programming
Attributes
Command Prompt
X
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.2
Pinging 192.168.1.2 with 32 bytes of data:
Request timed out.
Reply from 192.168.1.2: bytes=32 time=1ms TTL=128
Reply from 192.168.1.2: bytes=32 time=1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Ping statistics for 192.168.1.2:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 1ms, Average = 0ms
C:\>ping 192.168.1.101
Pinging 192.168.1.101 with 32 bytes of data:
Reply from 192.168.1.101: bytes=32 time<1ms TTL=128
Reply from 192.168.1.101: bytes=32 time<1ms TTL=128
Reply from 192.168.1.101: bytes=32 time<1ms TTL=128
Reply from 192.168.1.101: bytes=32 time<1ms TTL=128
Ping statistics for 192.168.1.101:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
C:\>ping 192.168.2.200
Pinging 192.168.2.200 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 192.168.2.200:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\>
Top

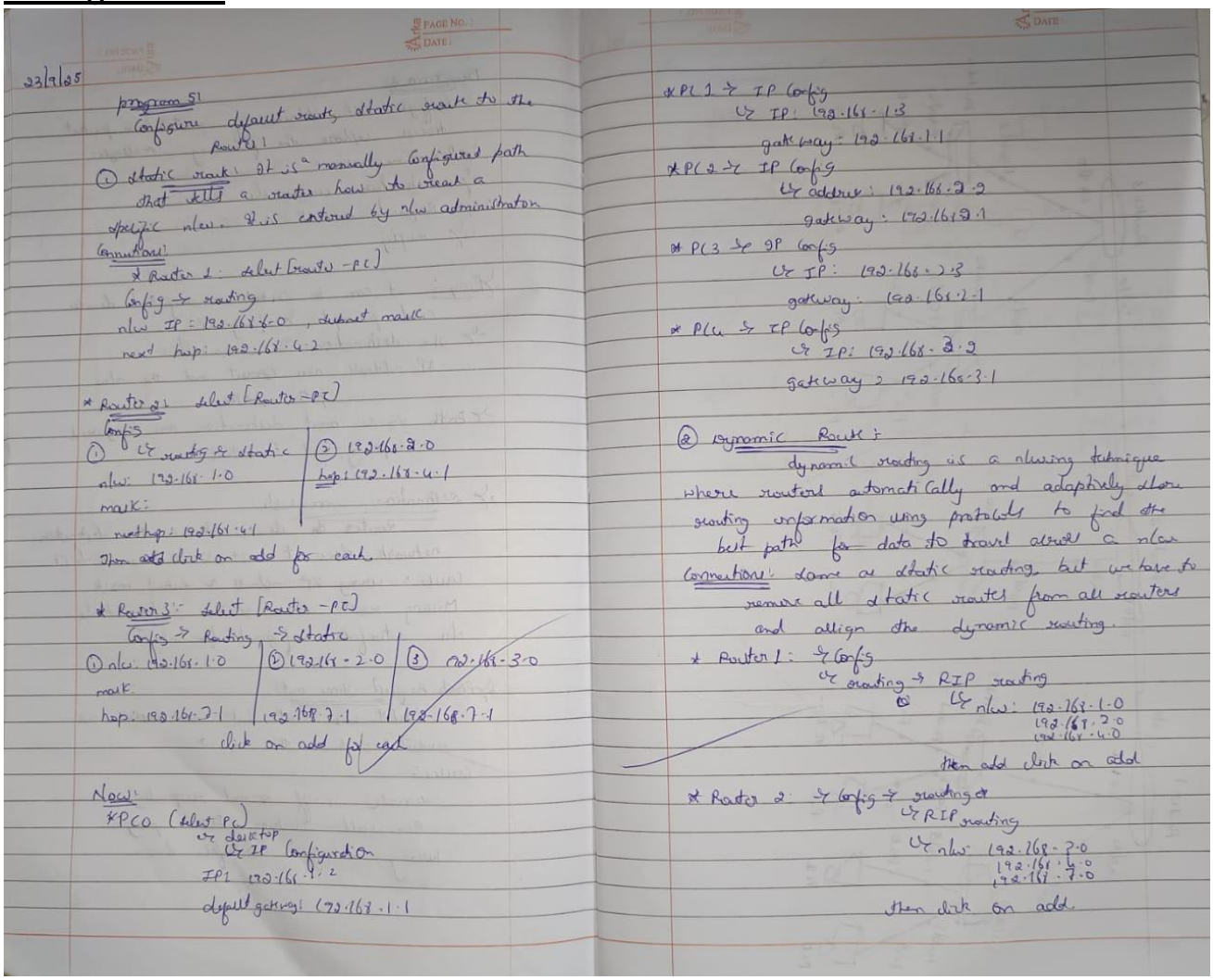
Program 5:

Aim: Configure default route, static route to the Router.

Network diagram:



Configuration:



Output:

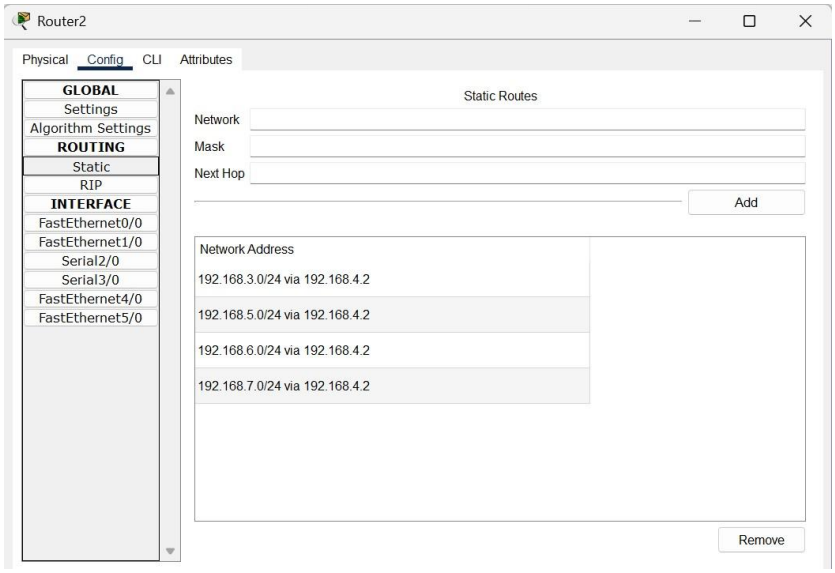


Fig 1. Router 2 – Static routing

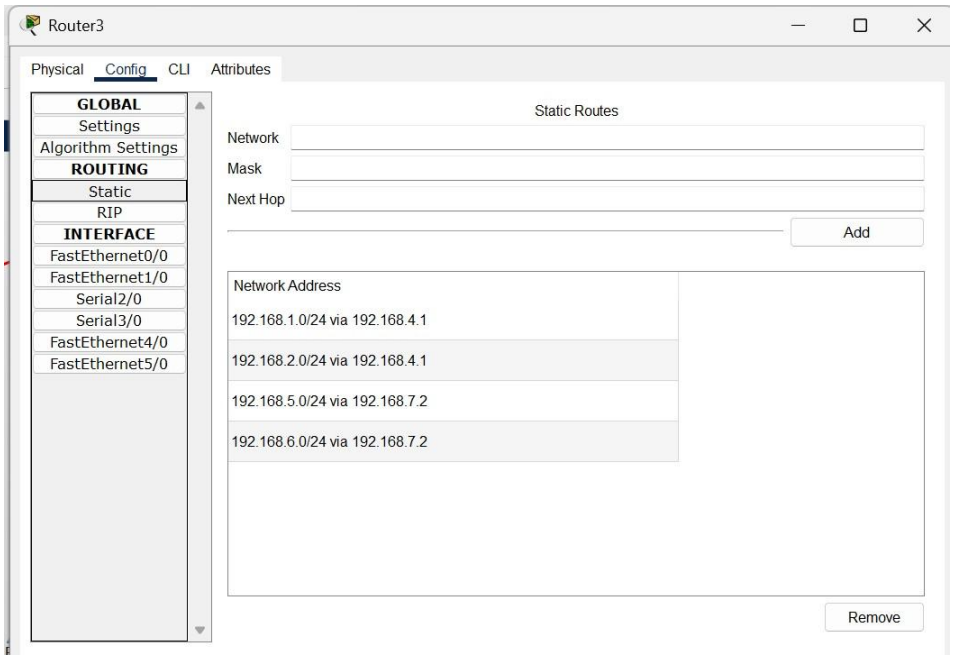


Fig 2. Router 3 – Static routing

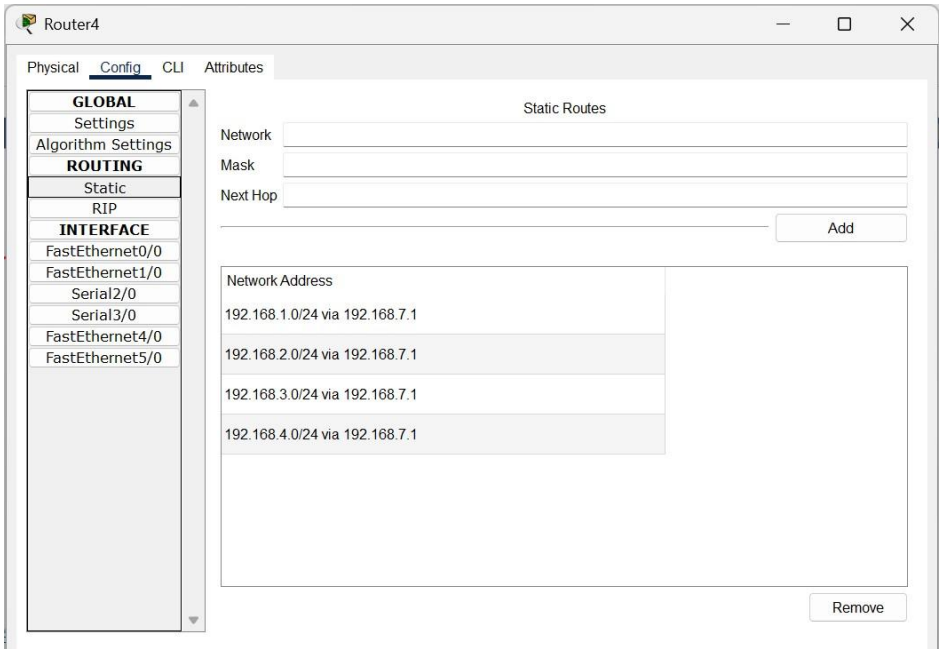
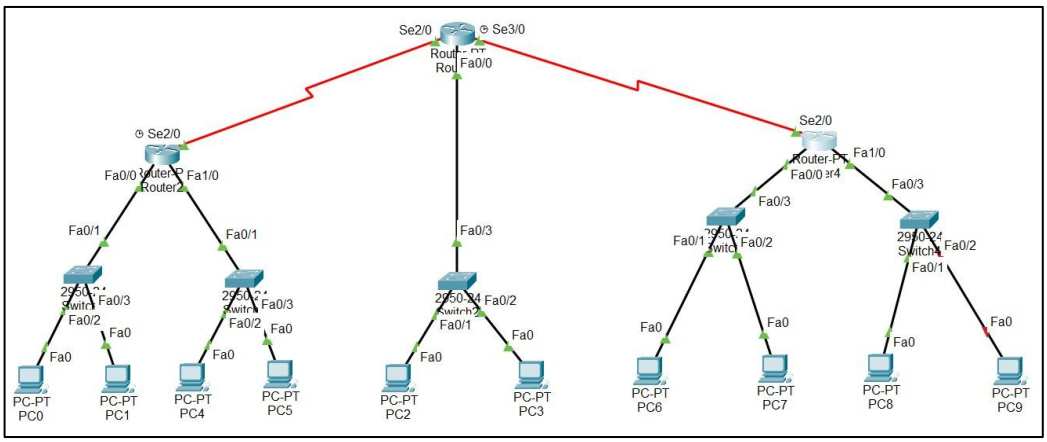


Fig 3. Router 4 – Static routing

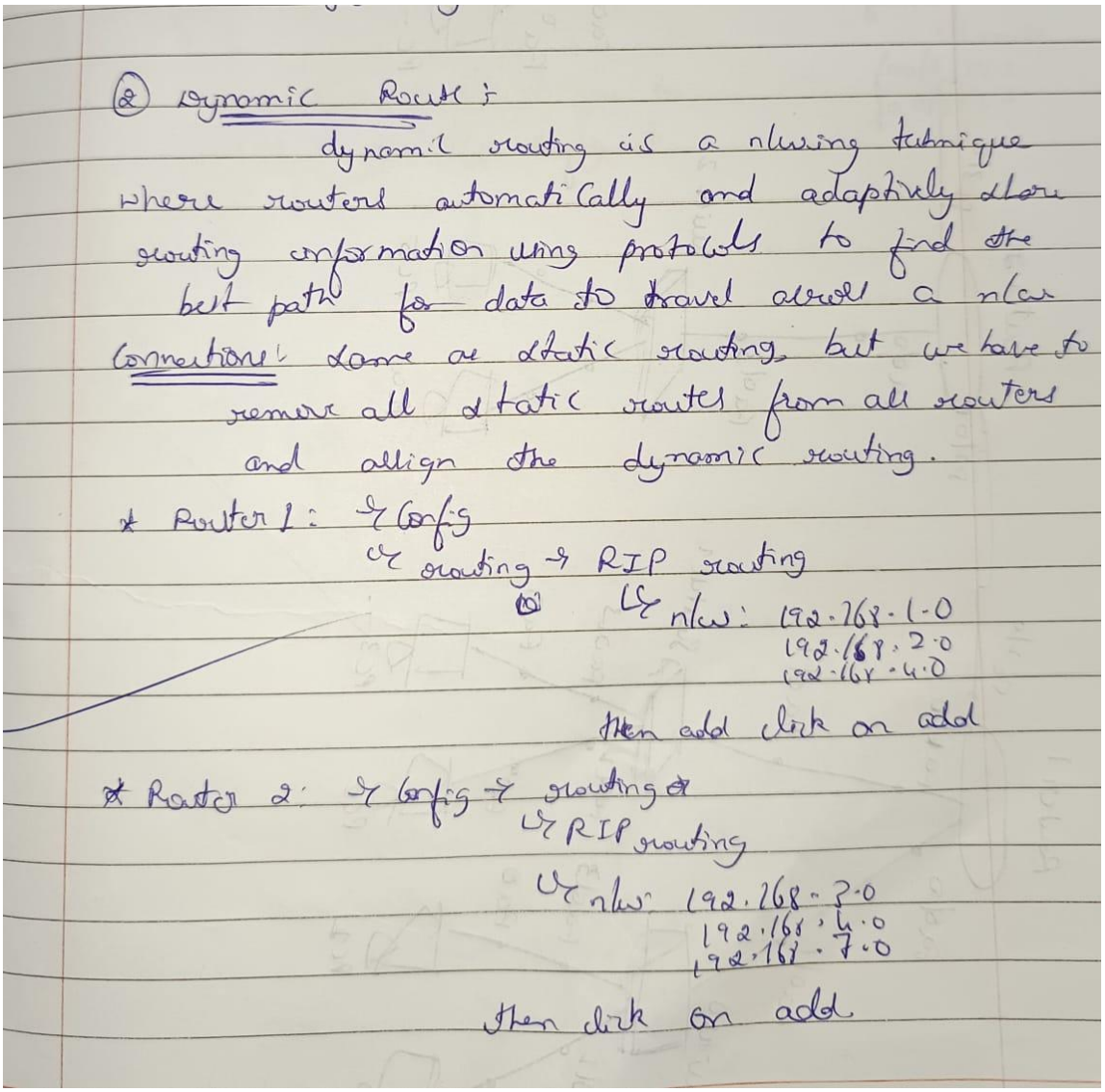
Program 6:

Aim: Configure RIP routing Protocol in Routers.

Network diagram:



Configuration:



Output:

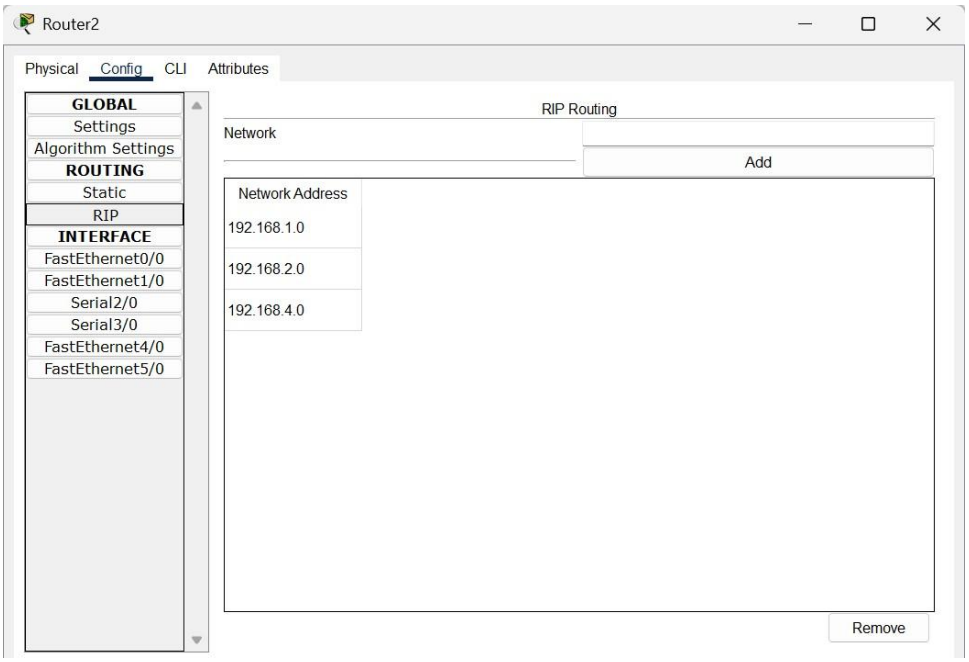


Fig 1. Router 2 – RIP routing

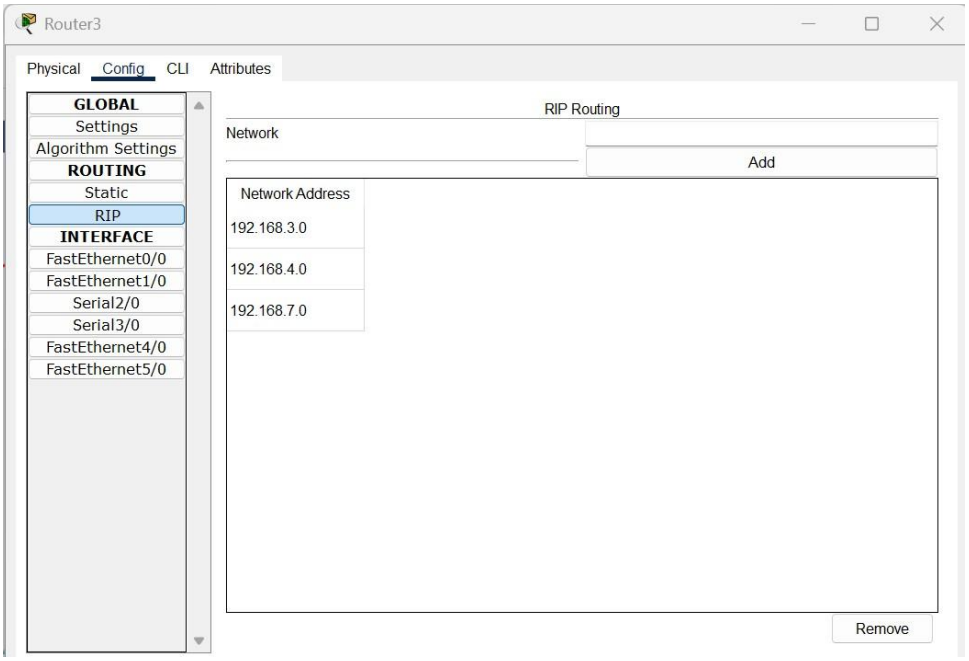


Fig 2. Router 3 – RIP routing

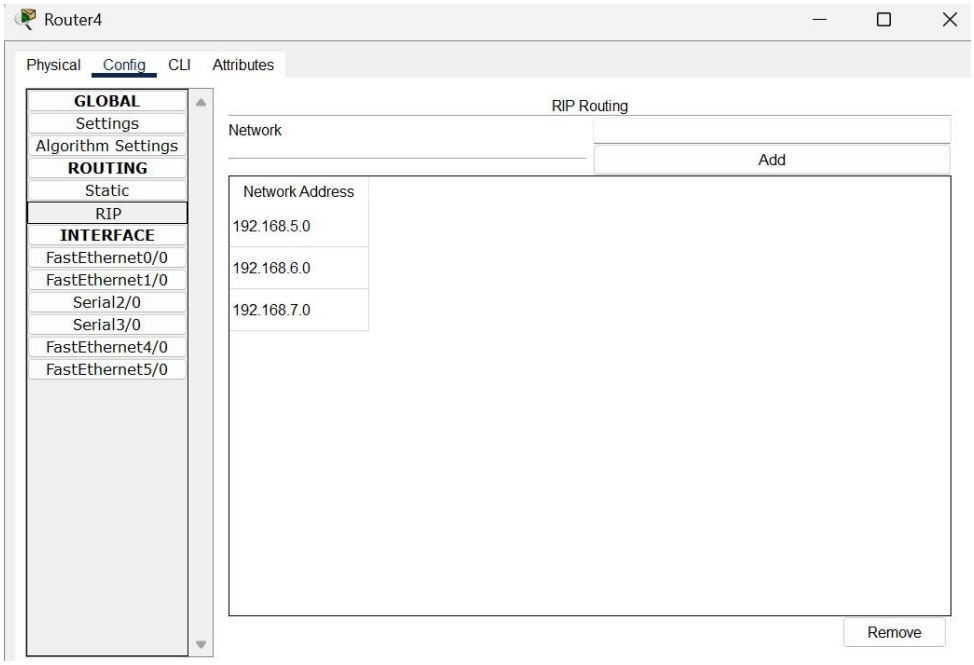
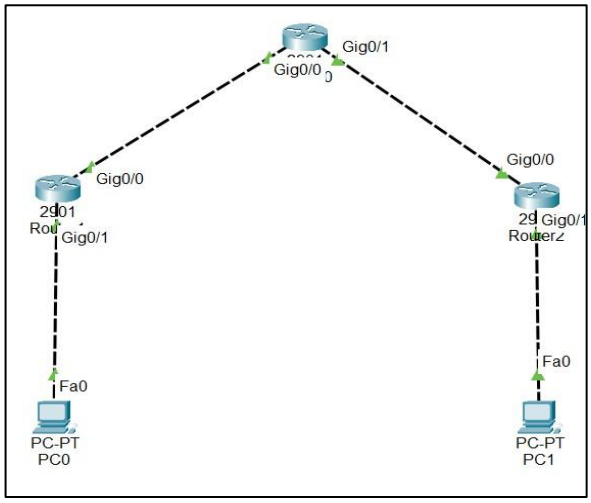


Fig 3. Router 4 – RIP routing

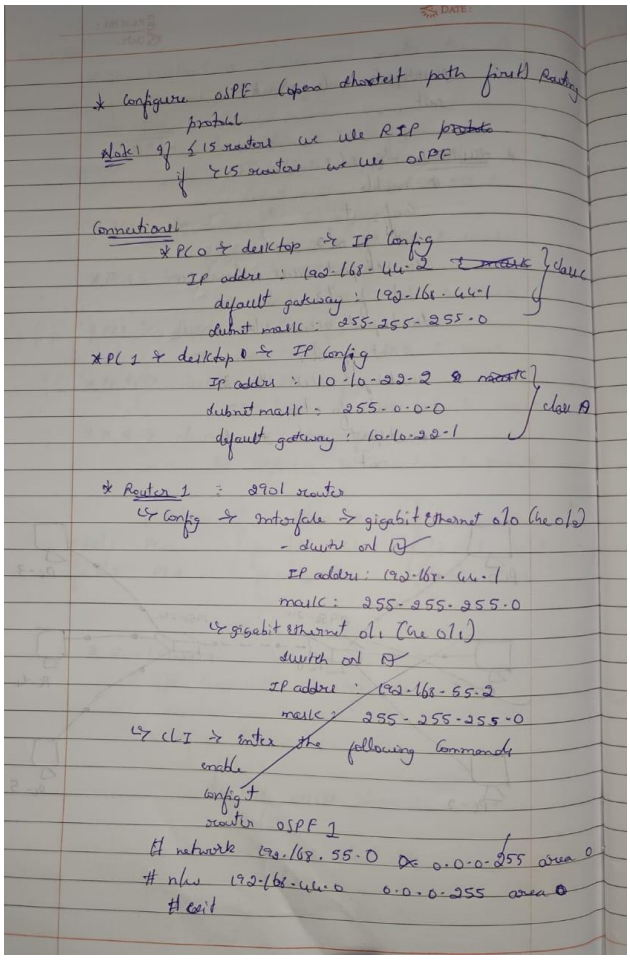
Program 7:

Aim: Configure OSPF routing protocol.

Network diagram:



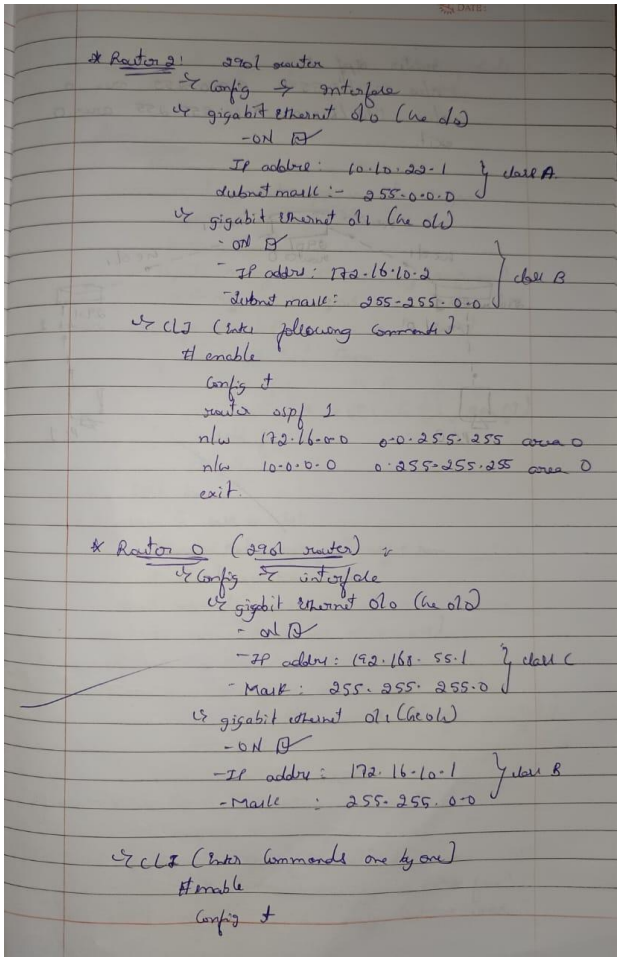
Configuration:



* Configure OSPF (open shortest path first) Routing protocol
Note: If 2 IS routers we use RIP protocol
if >2 IS routers we use OSPF

Connections:
* PC0 → desktop → IP Config
IP addr: 192.168.44.2
default gateway: 192.168.44.1
Subnet mask: 255.255.255.0 } class C
* PC1 → desktop → IP Config
IP addr: 10.10.22.2
Subnet mask: 255.0.0.0
default gateway: 10.10.22.1 } class A

* Router 1 : 2901 router
→ Config → interface → gigabit ethernet 0/0 (he 0/0)
- Switch on
IP addr: 192.168.44.1
mask: 255.255.255.0
→ gigabit ethernet 0/1 (he 0/1)
Switch on
IP addr: 192.168.55.2
mask: 255.255.255.0
→ CLI → enter the following commands
enable
config t
router OSPF 1
network 192.168.55.0 0.0.0.255 area 0
n/w 192.168.44.0 0.0.0.255 area 0
exit



* Router 2: 2901 router
→ Config → interface
→ gigabit ethernet 0/0 (he 0/0)
-ON
IP addr: 10.10.22.1
Subnet mask: 255.0.0.0 } class A
→ gigabit ethernet 0/1 (he 0/1)
- ON
- IP addr: 172.16.10.2
- Subnet mask: 255.255.0.0 } class B
→ CLI (enter following commands)
enable
Config t
router ospf 1
n/w 172.16.0.0 0.0.255.255 area 0
n/w 10.0.0.0 0.255.255.255 area 0
exit

* Router 0 (2901 router)
→ Config → interface
→ gigabit ethernet 0/0 (he 0/0)
- ON
- IP addr: 192.168.55.1
- Mask: 255.255.255.0 } class C
→ gigabit ethernet 0/1 (he 0/1)
- ON
- IP addr: 172.16.10.1
- Mask: 255.255.0.0 } class B

→ CLI (enter commands one by one)
enable
Config t

Output:

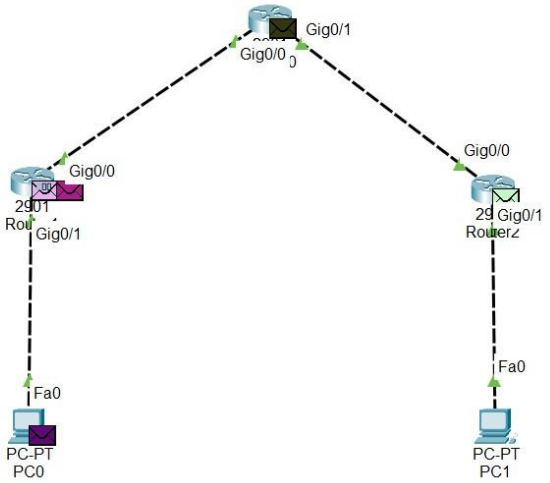


Fig 1. Sending PDU message from PC0 to PC1

PDU List Window

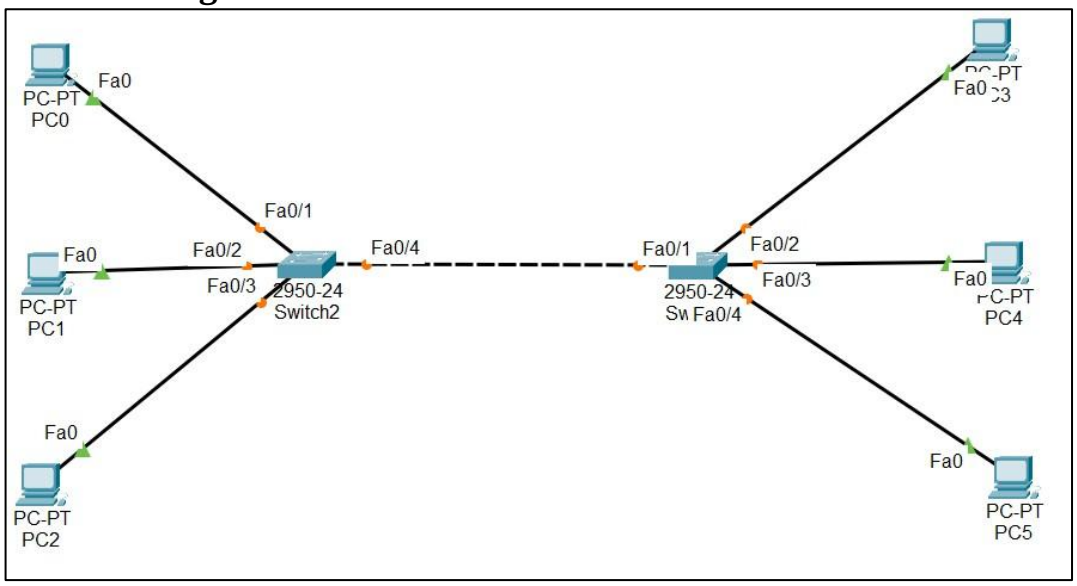
Fire	Last Status	Source	Destination	Type	Color	Time(sec	Periodic	Num	Edit	Delete
	Successful	PC0	PC1	ICMP		0.000	N	0	(edit)	(delete)
	Successful	PC0	Router2	ICMP		0.000	N	1	(edit)	(delete)
	Successful	PC0	Router0	ICMP		0.000	N	2	(edit)	(delete)
	Successful	Router0	PC1	ICMP		0.000	N	3	(edit)	(delete)
	Successful	Router1	PC1	ICMP		0.000	N	4	(edit)	(delete)
	Successful	Router1	Router2	ICMP		0.000	N	5	(edit)	(delete)

Fig 2. Checking PDU messages

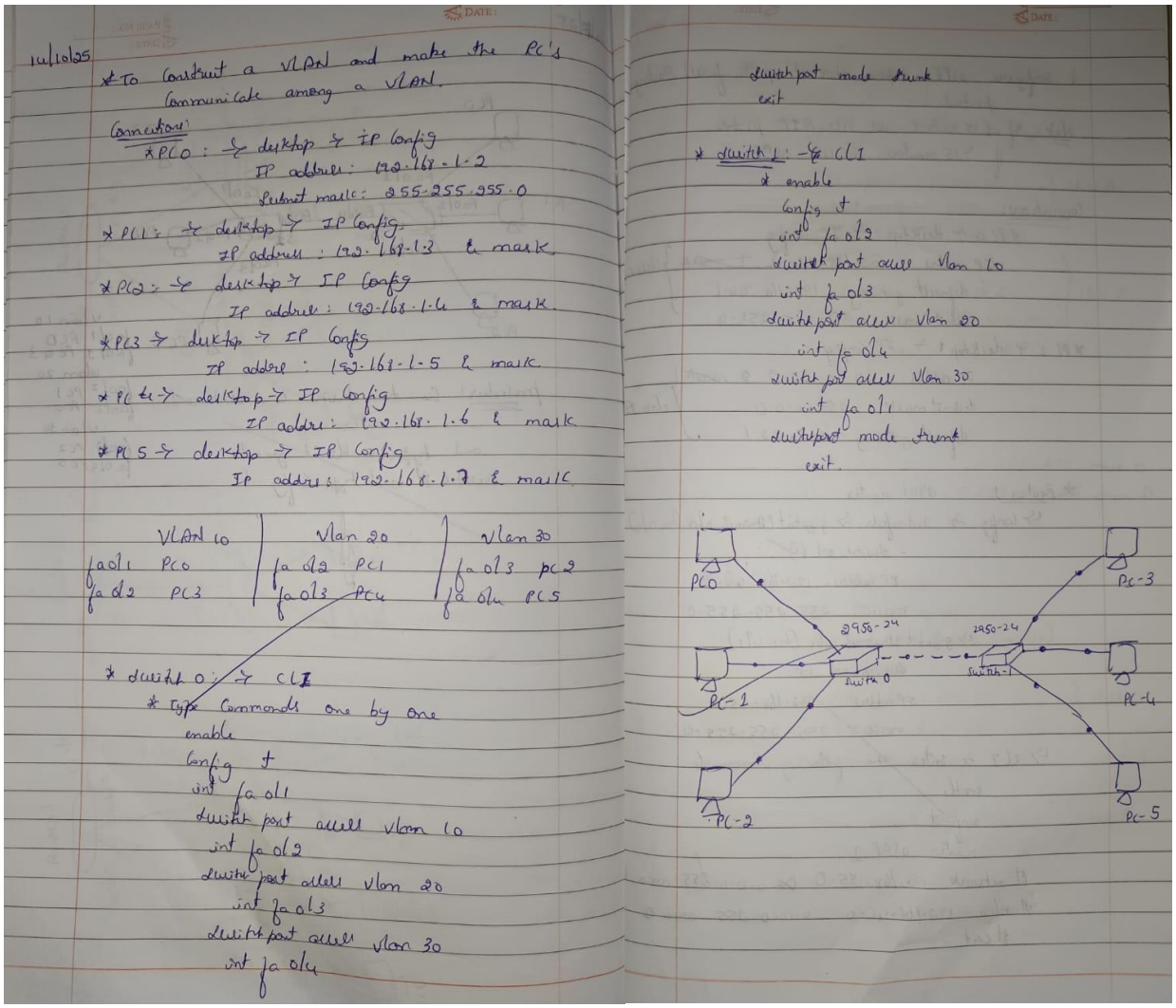
Program 8:

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

Network diagram:



Configuration:



Output:

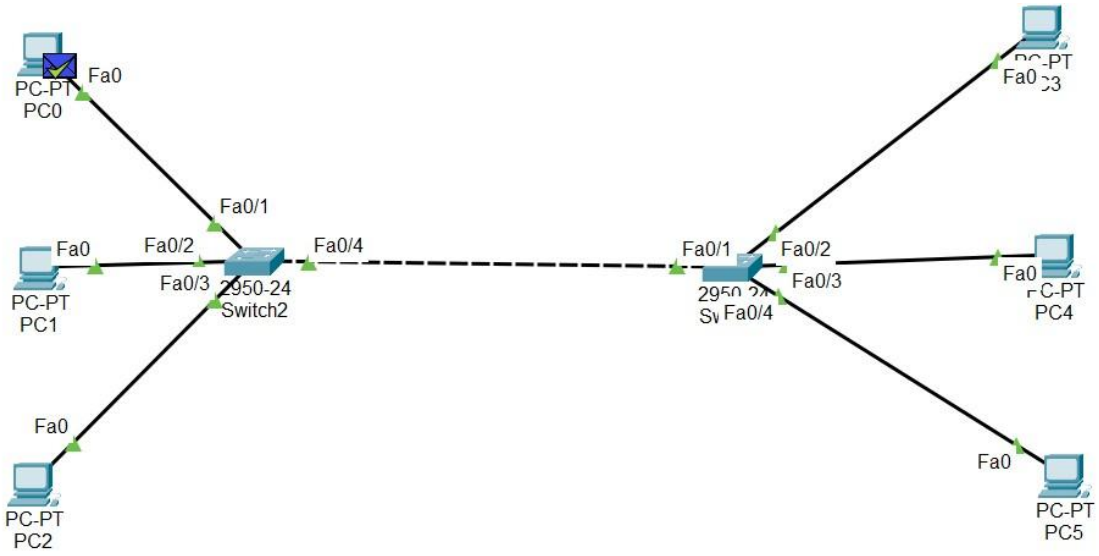
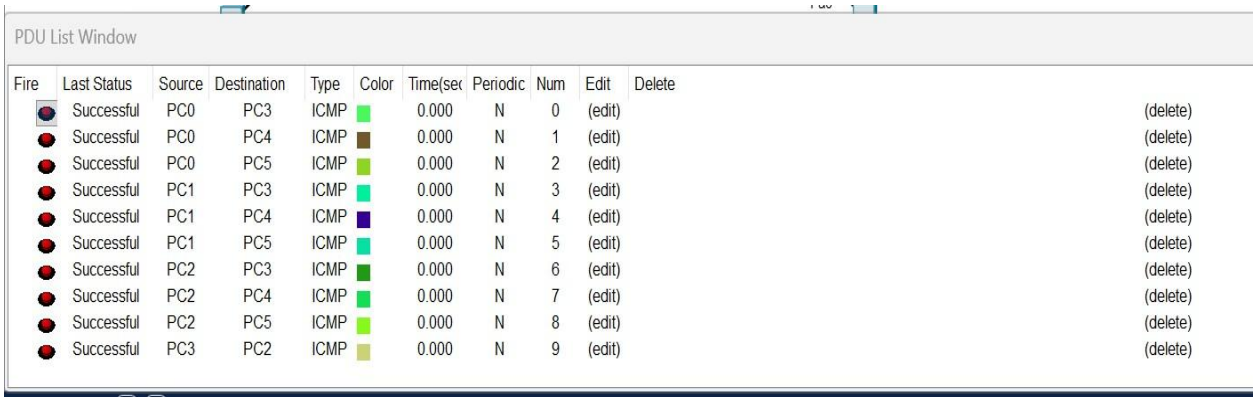


Fig 1. Sending PDU message from PC0 to PC5



PDU List Window

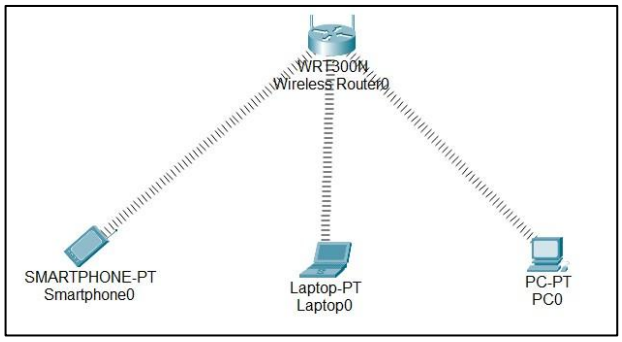
Fire	Last Status	Source	Destination	Type	Color	Time(sec	Periodic	Num	Edit	Delete
	Successful	PC0	PC3	ICMP		0.000	N	0	(edit)	(delete)
	Successful	PC0	PC4	ICMP		0.000	N	1	(edit)	(delete)
	Successful	PC0	PC5	ICMP		0.000	N	2	(edit)	(delete)
	Successful	PC1	PC3	ICMP		0.000	N	3	(edit)	(delete)
	Successful	PC1	PC4	ICMP		0.000	N	4	(edit)	(delete)
	Successful	PC1	PC5	ICMP		0.000	N	5	(edit)	(delete)
	Successful	PC2	PC3	ICMP		0.000	N	6	(edit)	(delete)
	Successful	PC2	PC4	ICMP		0.000	N	7	(edit)	(delete)
	Successful	PC2	PC5	ICMP		0.000	N	8	(edit)	(delete)
	Successful	PC3	PC2	ICMP		0.000	N	9	(edit)	(delete)

Fig 2. Checking PDU messages

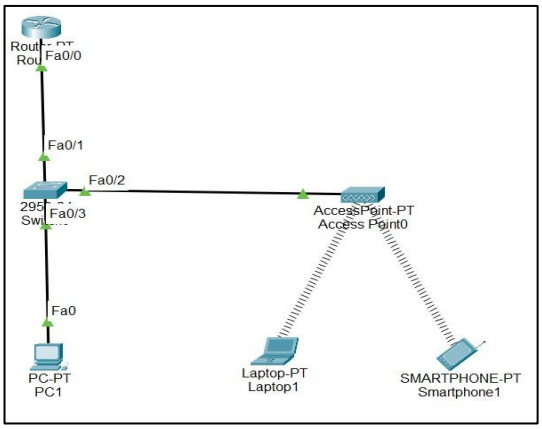
Program 9:

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

Network diagram:

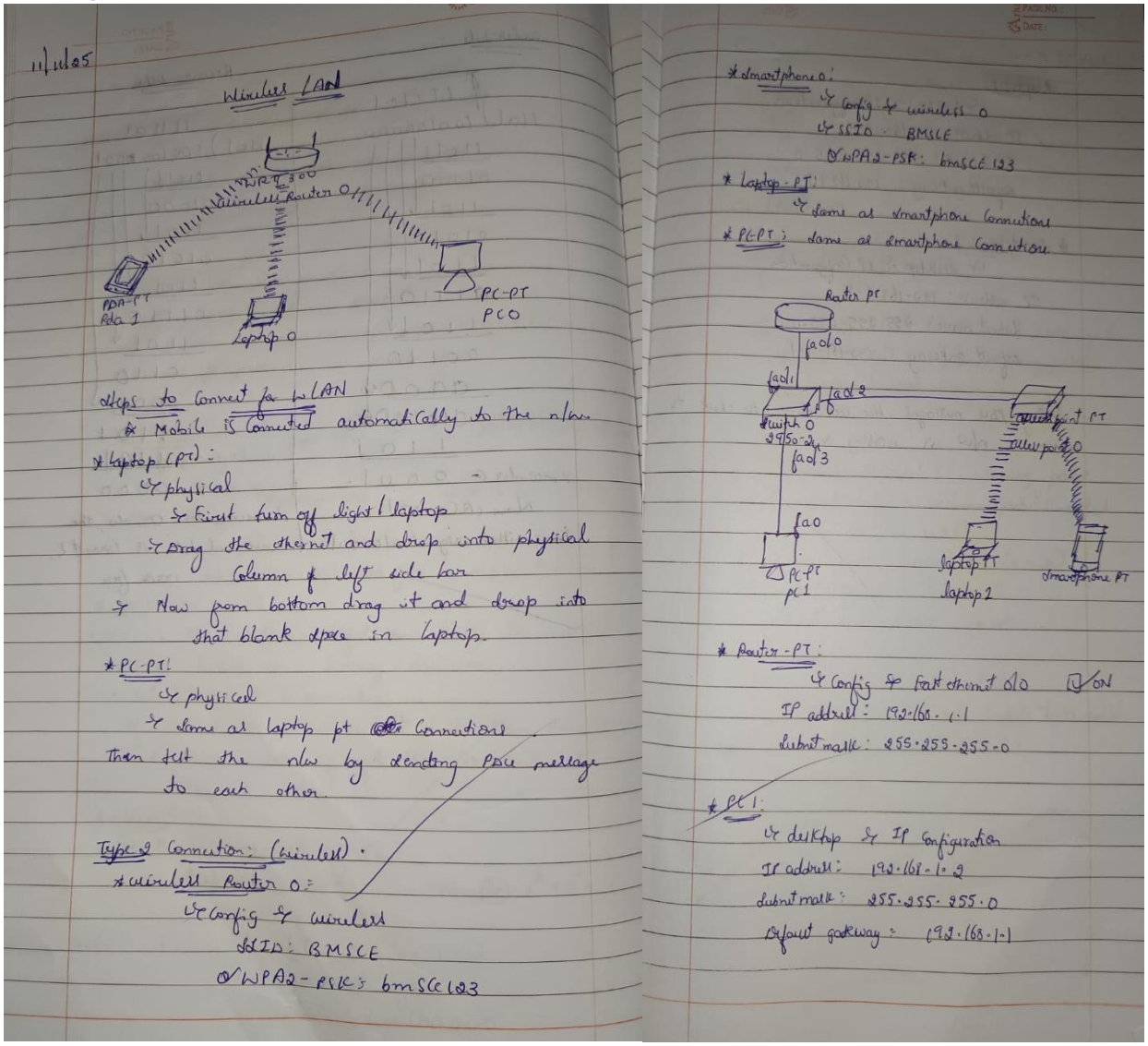


Configuration 1



Configuration 2

Configuration:



Output:

1. Do Physical Connections In:

- Laptop • PC

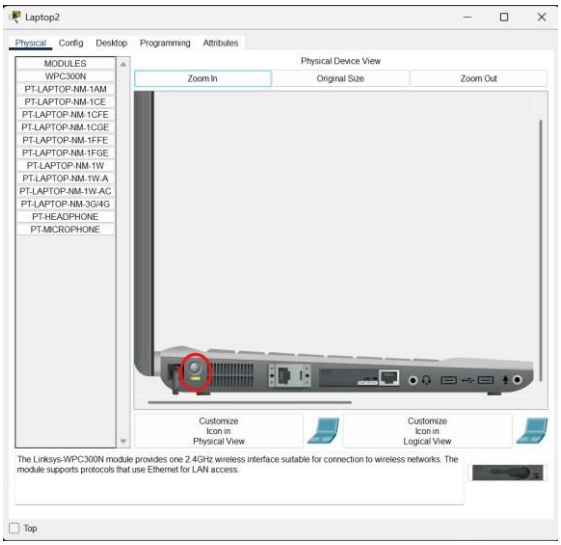


Fig 1.1 Step1: Turn off light / Power off laptop

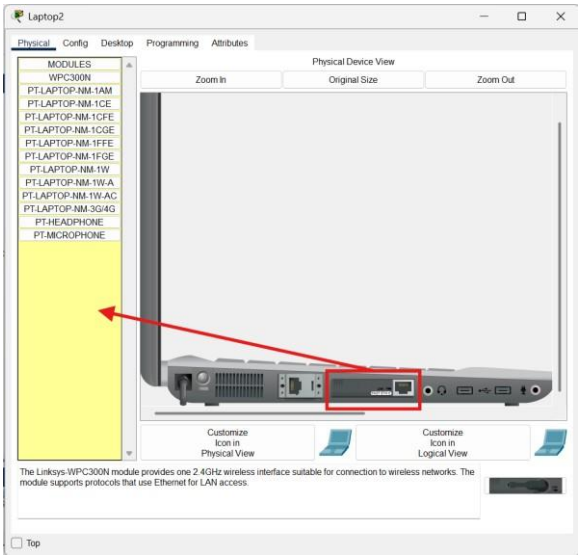


Fig 1.2 Step2: Drag and Drop the Ethernet into pointed location

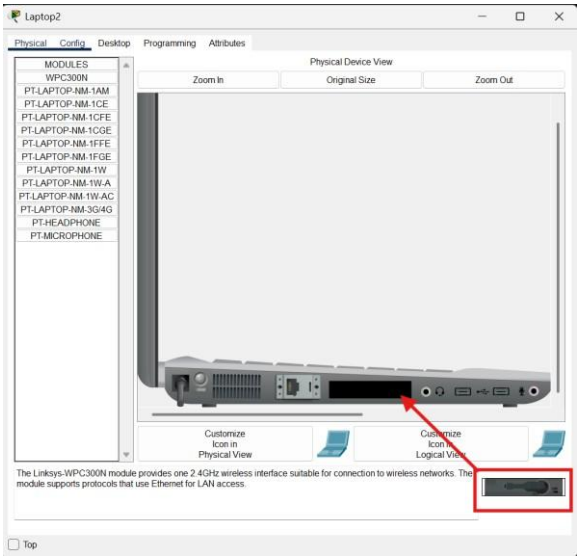


Fig 1.3 Step3: Drag and Drop the device into pointed location and Turn on light/Laptop

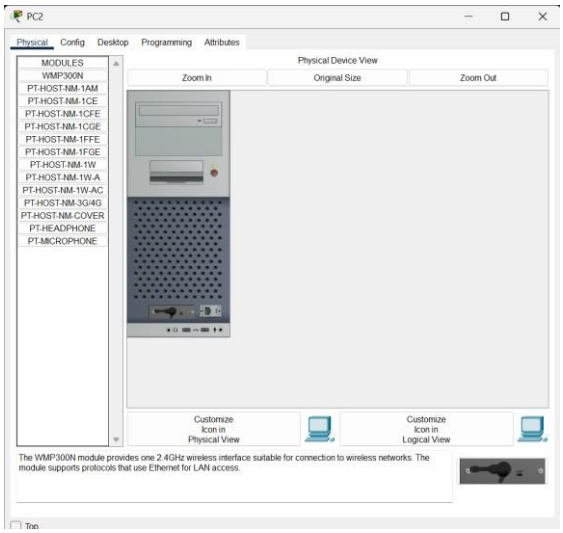


Fig 2. PC physical connection (combined 3 steps)

2. Do Wireless Connection in:

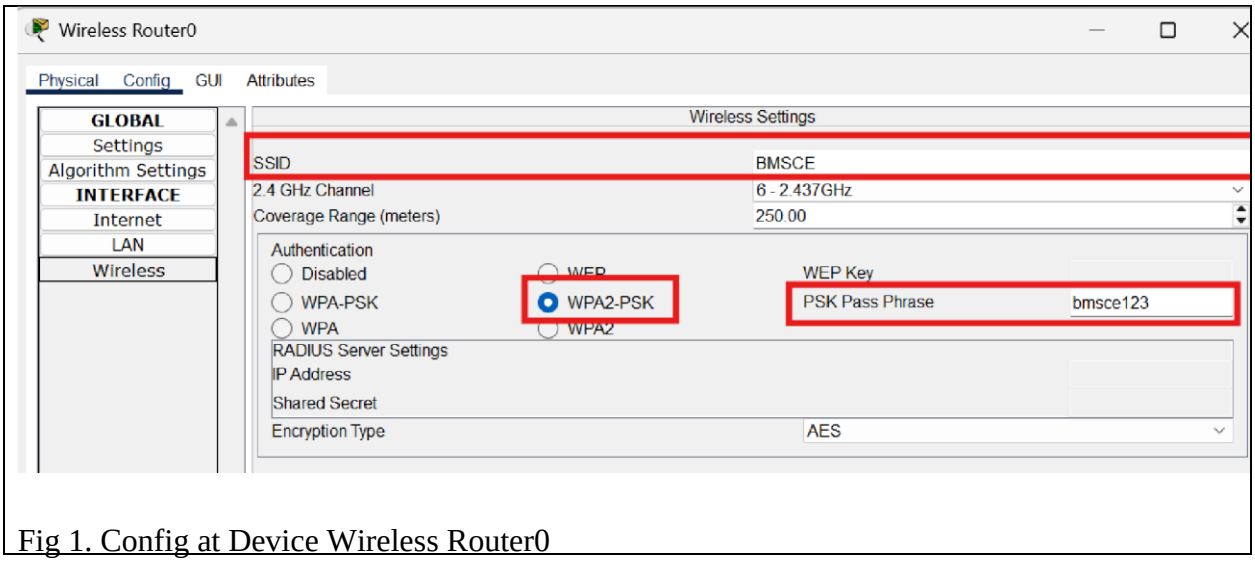


Fig 1. Config at Device Wireless Router0

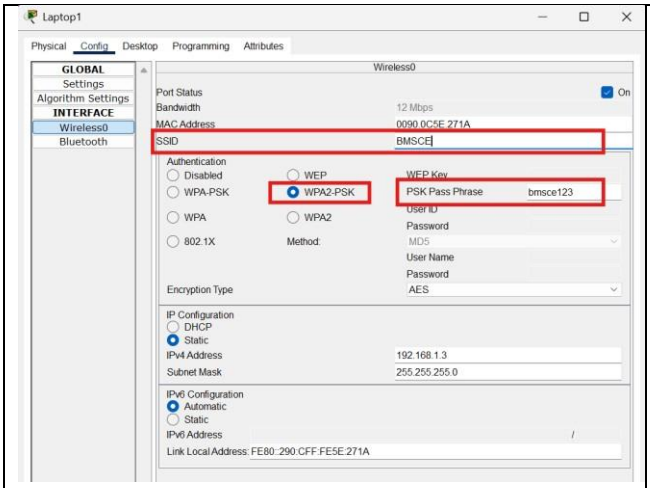


Fig 2. Config at Device Laptop0

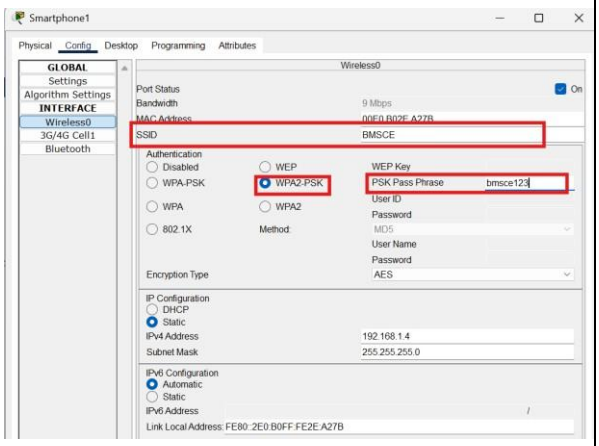


Fig 3. Config at Device Smartphone0

PDU List Window

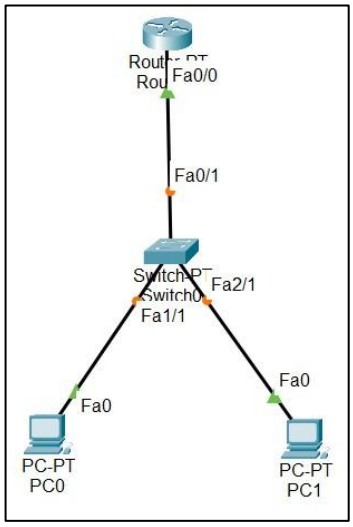
Fire	Last Status	Source	Destination	Type	Color	Time(sec	Periodic	Num	Edit
	Failed	Smar...	Laptop0	ICMP		0.000	N	0	(edit)
	Successful	Lapto...	PC0	ICMP		0.000	N	1	(edit)
	Failed	PC0	Laptop0	ICMP		0.000	N	2	(edit)
	Successful	PC0	Smartphone0	ICMP		0.000	N	3	(edit)
	Failed	PC0	Laptop0	ICMP		0.000	N	4	(edit)
	Successful	Lapto...	Smartphone0	ICMP		0.000	N	5	(edit)
	Successful	Lapto...	PC0	ICMP		0.000	N	6	(edit)
	Successful	PC0	Smartphone0	ICMP		0.000	N	7	(edit)
	Successful	Lapto...	PC1	ICMP		0.000	N	8	(edit)

Fig 3. Checking PDU messages

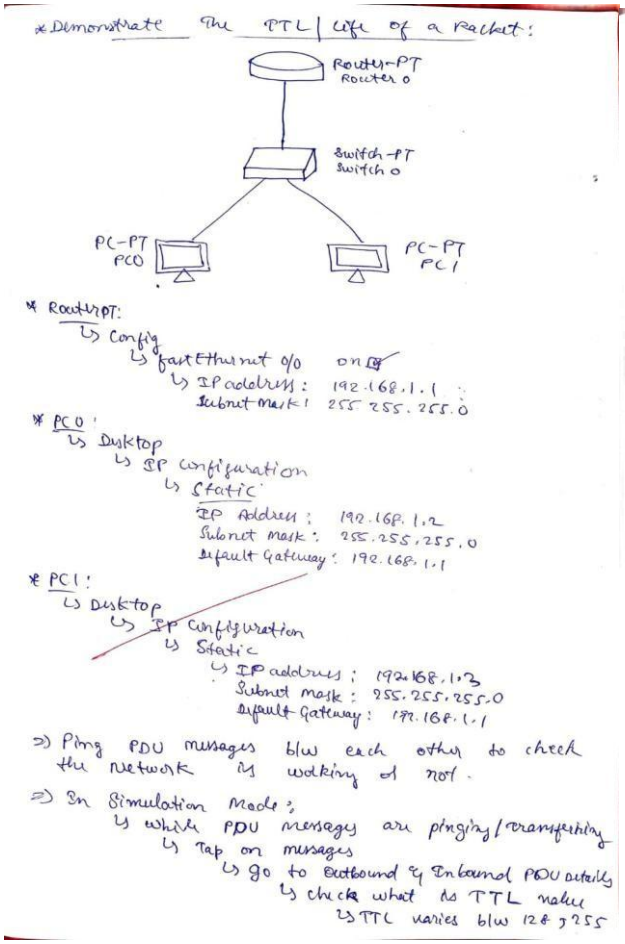
Program 10:

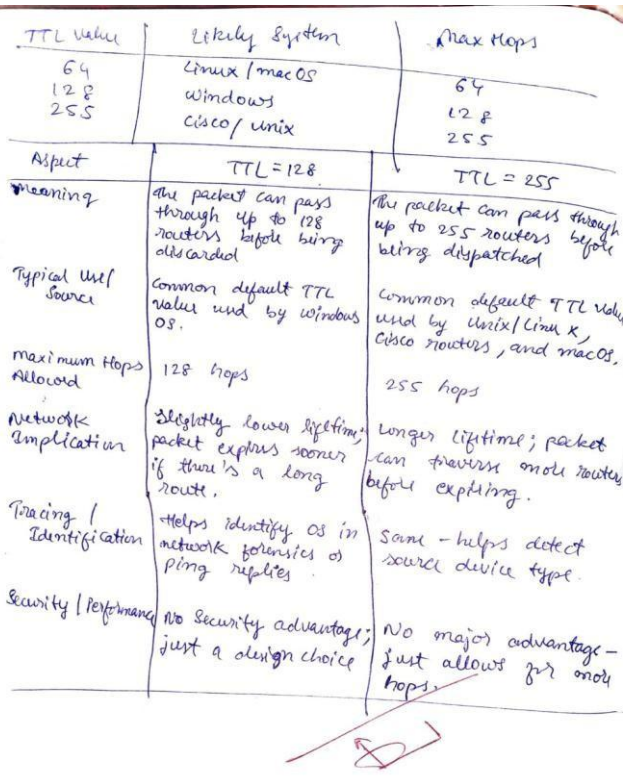
Aim: Demonstrate the TTL/ Life of a Packet.

Network diagram:



Configuration:





Output:

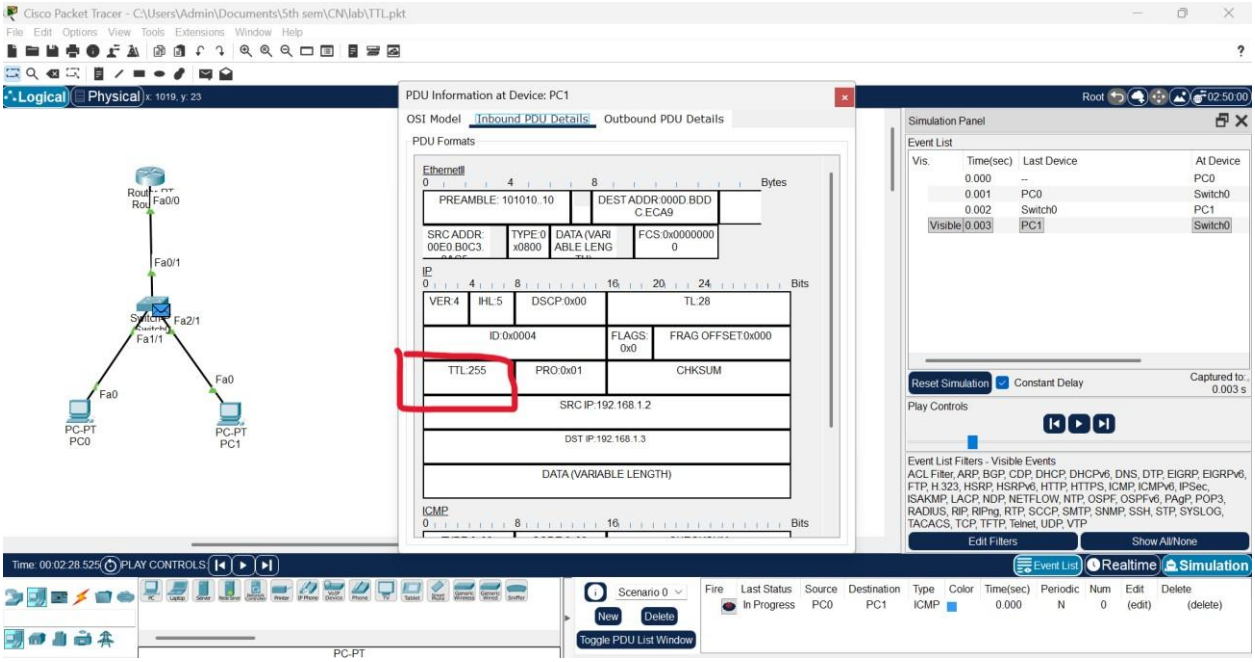


Fig 1. Inbound PDU Details at Device PC1

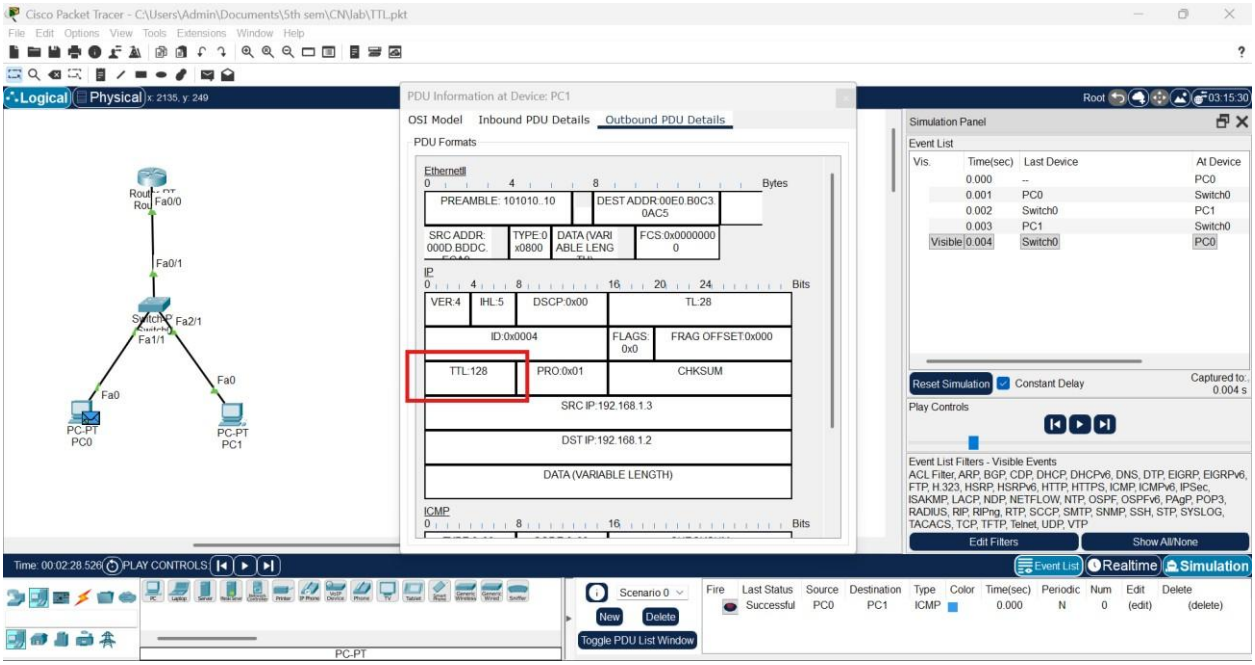


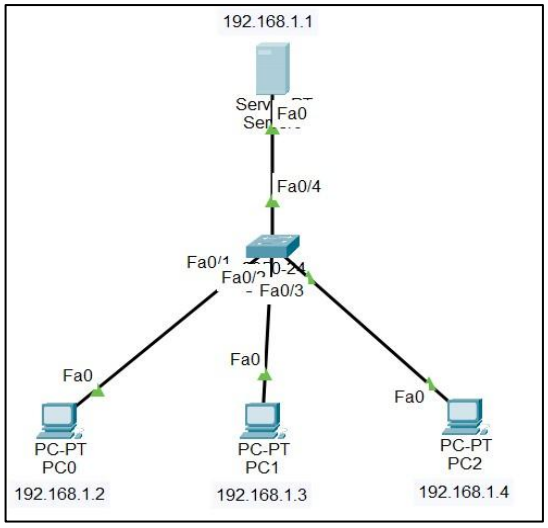
Fig 1. Outbound PDU Details at Device PC1

Program 11:

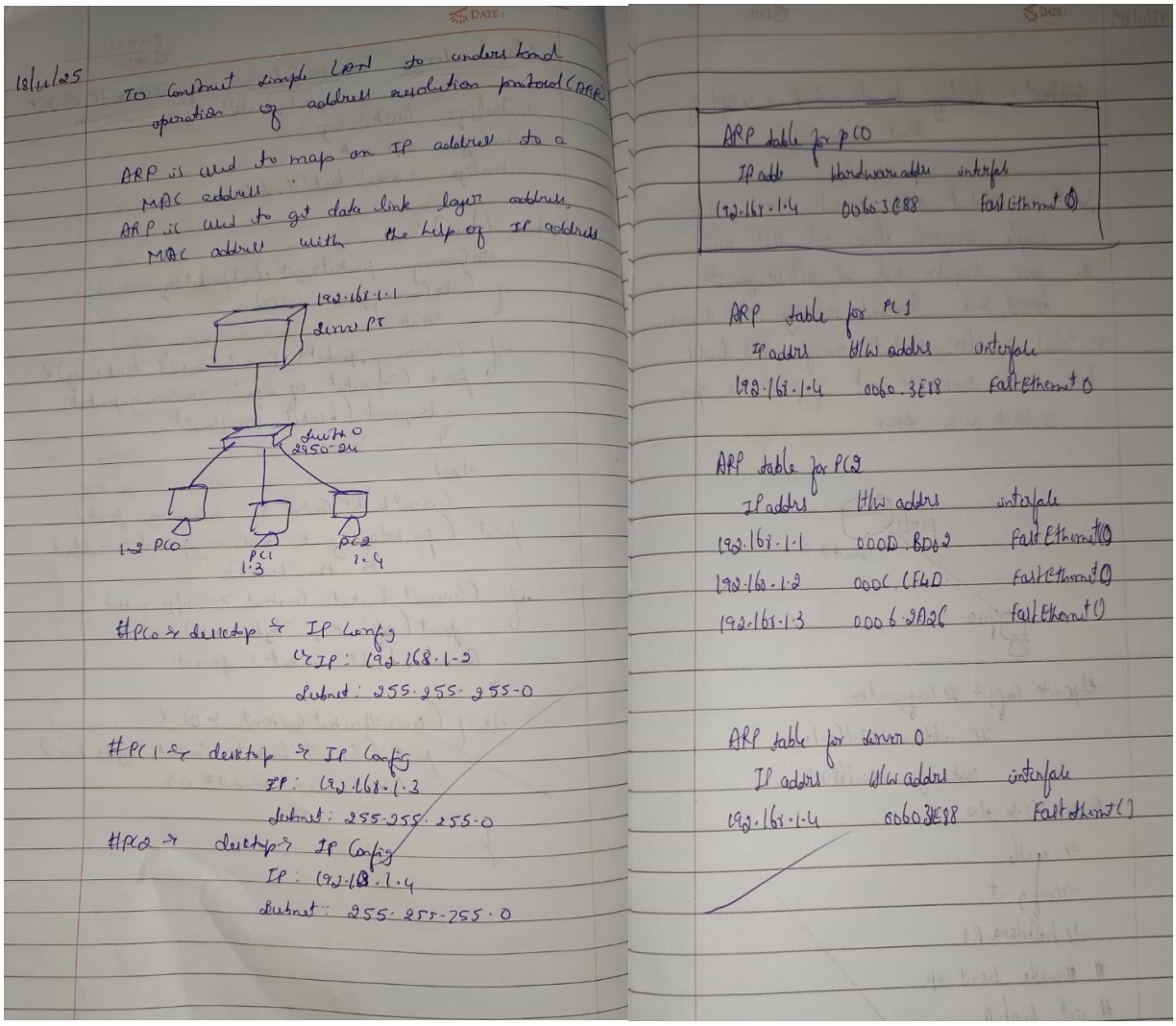
Aim: Aim: To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP).

.

Network diagram:



Configuration:



Output:

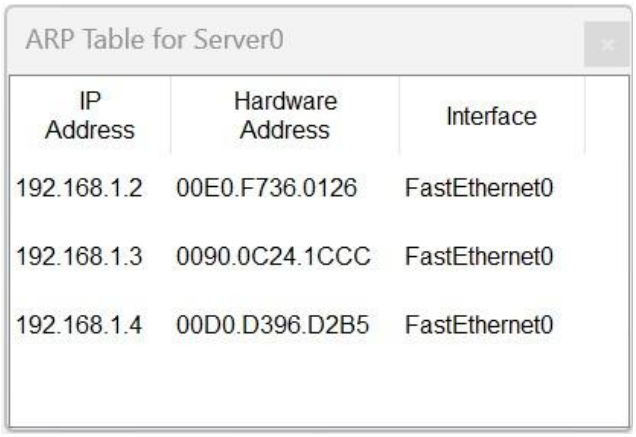


Fig 1.1 ARP table at Server0

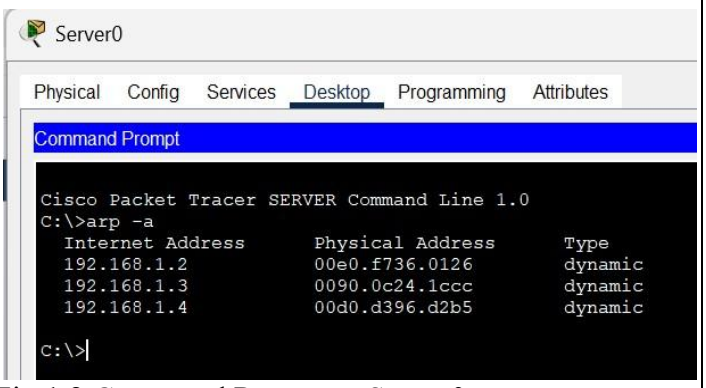


Fig 1.2 Command Prompt at Server0

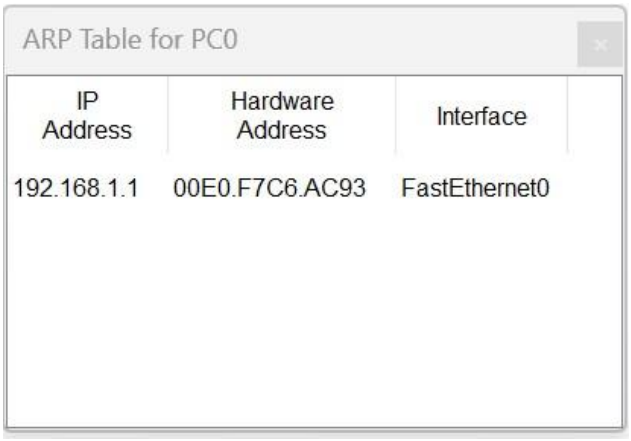


Fig 2.1 ARP table at PC0

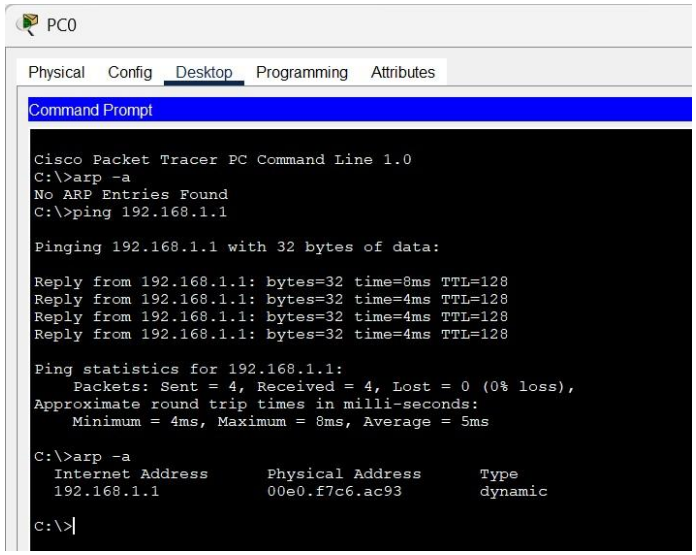


Fig 2.2 Command Prompt at PC0

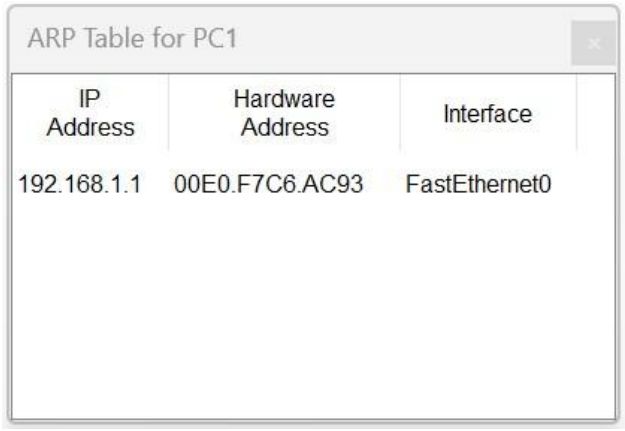


Fig 3.1 ARP table at PC1

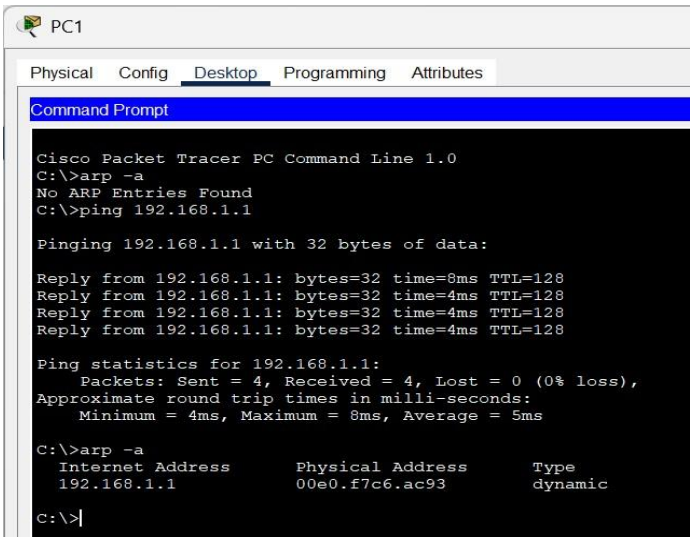
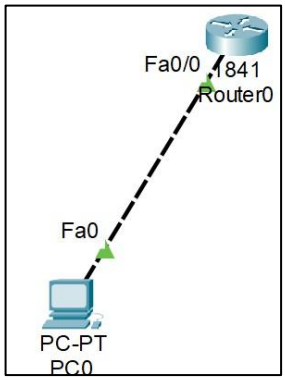


Fig 3.2 Command Prompt at PC1

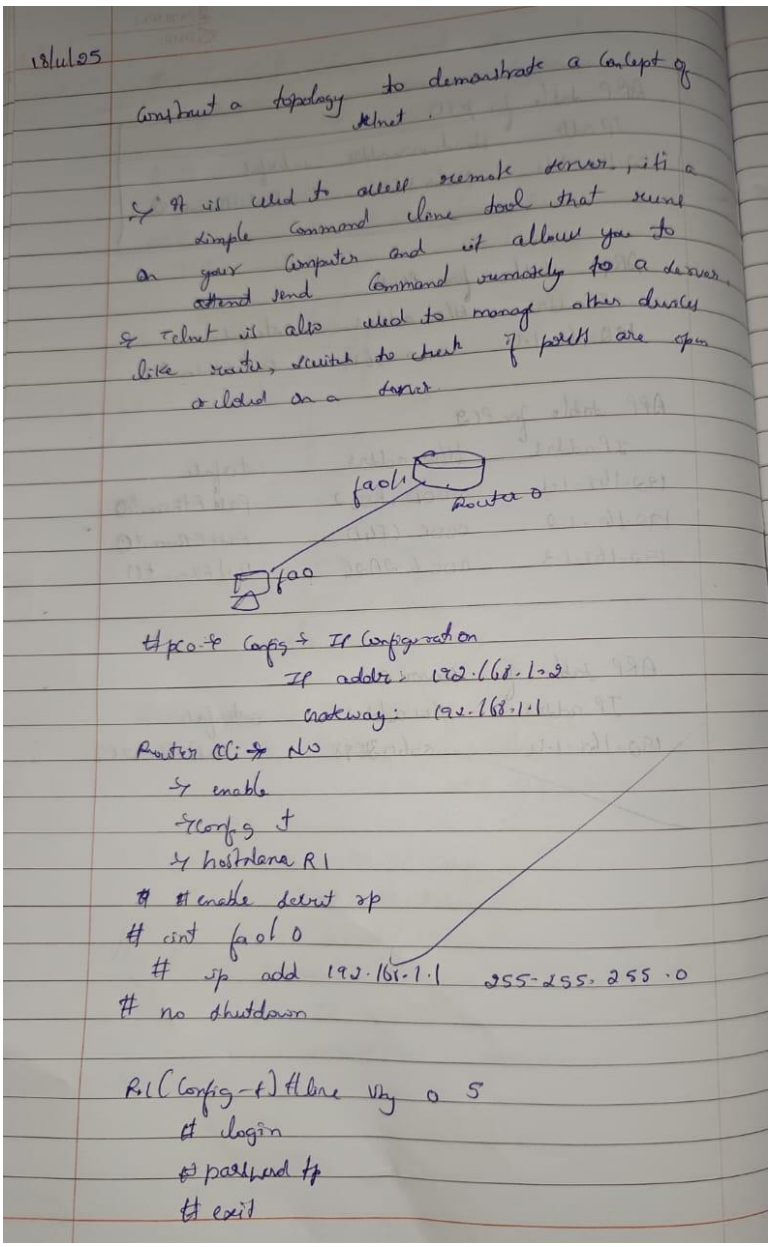
Program 12:

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

Network diagram:



Configuration:



Output:

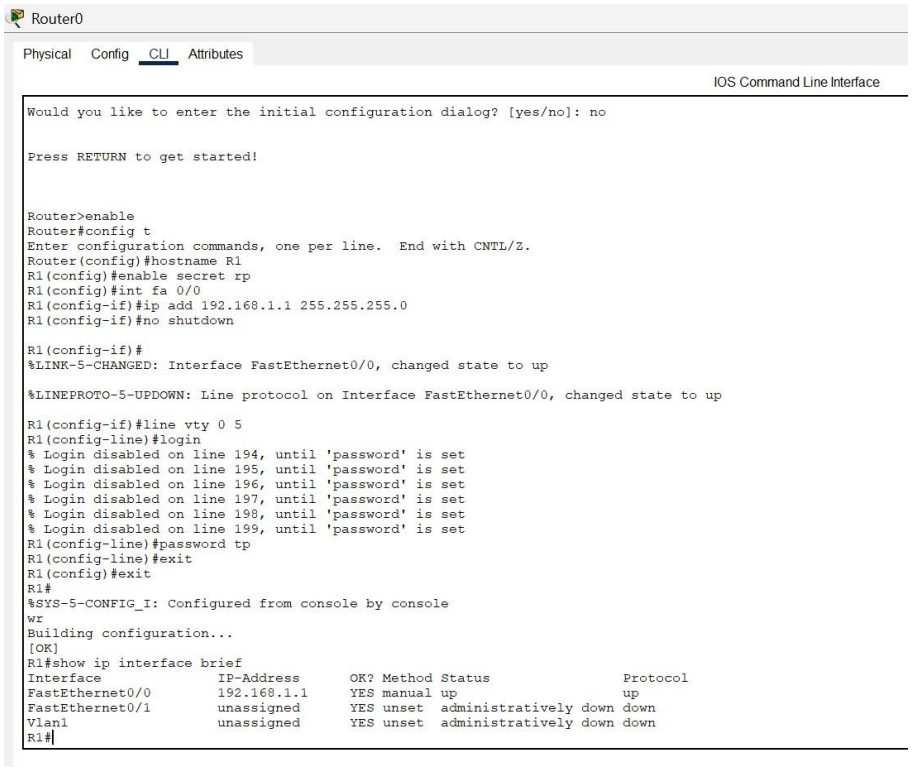


Fig 1. Router0 – CLI commands

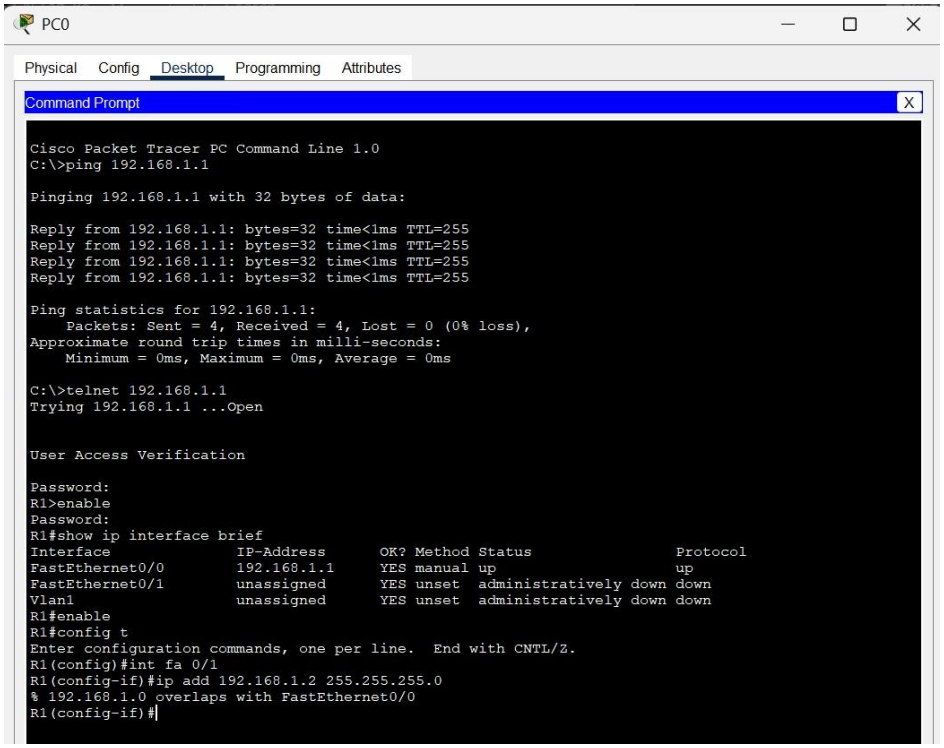


Fig2. PC command line prompt

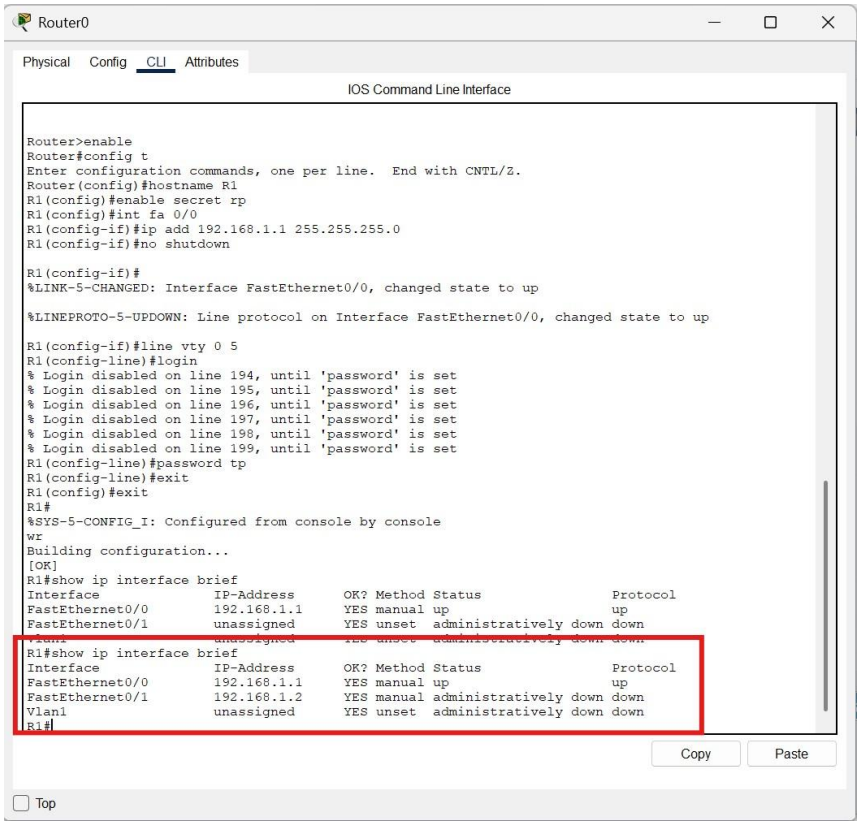


Fig 3. Updated the changes into Router0

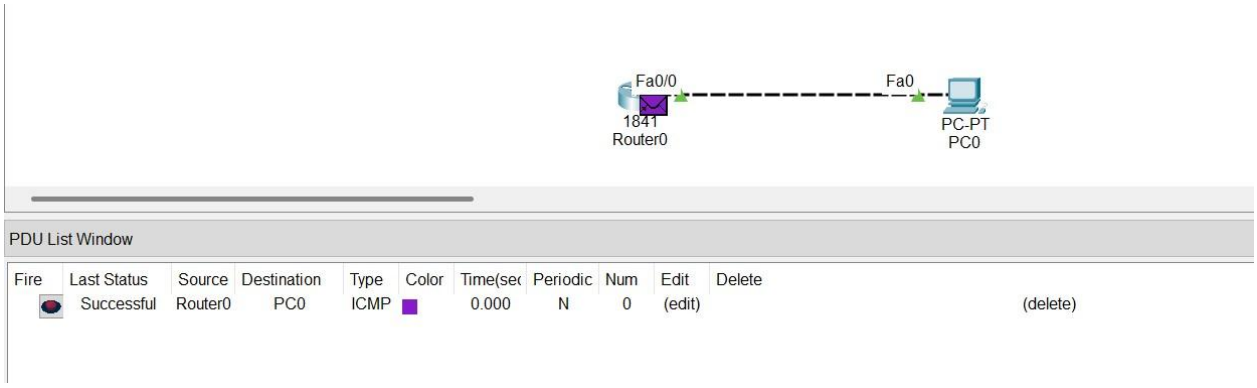


Fig 4. PDU message Successful

PART – B

Program 1:

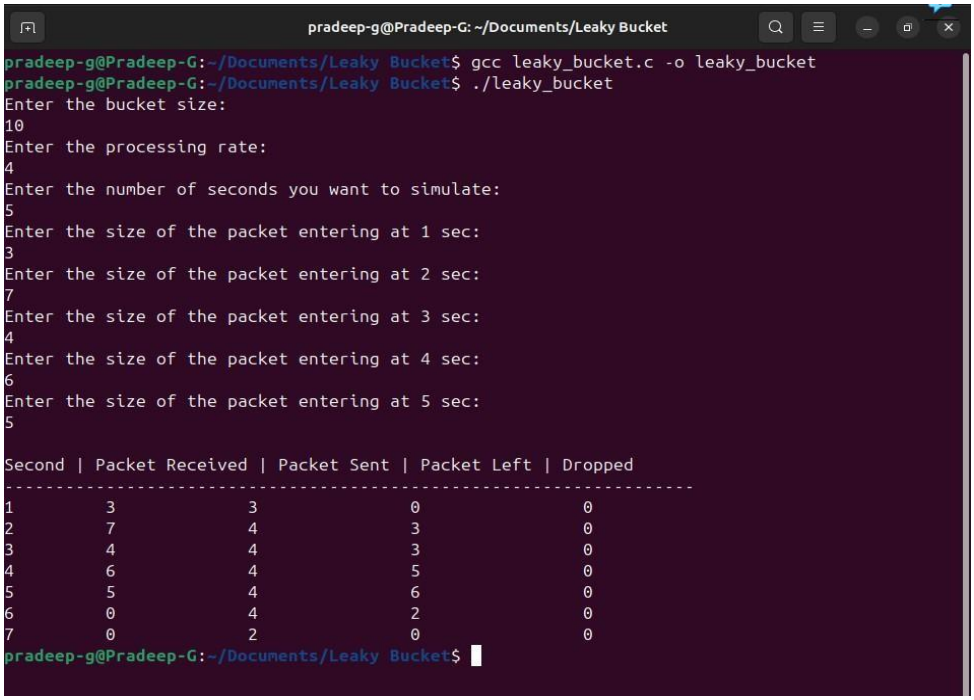
Aim: Write a program for congestion control using Leaky bucket algorithm.

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;
    }

    // Remaining packets after time ends
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;
}
```

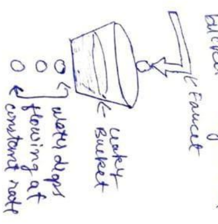
Output:

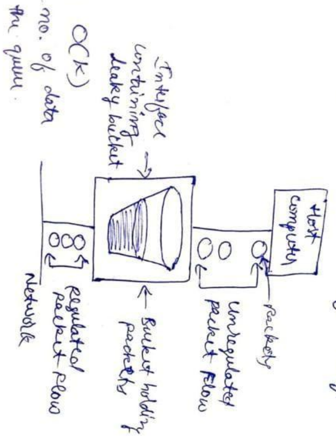

pradeep-g@Pradeep-G: ~/Documents/Leaky Bucket
pradeep-g@Pradeep-G:~/Documents/Leaky Bucket$ gcc leaky_bucket.c -o leaky_bucket
pradeep-g@Pradeep-G:~/Documents/Leaky Bucket$ ./leaky_bucket
Enter the bucket size:
10
Enter the processing rate:
4
Enter the number of seconds you want to simulate:
5
Enter 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
6 0 4 2 0
7 0 2 0 0
pradeep-g@Pradeep-G:~/Documents/Leaky Bucket$

Observation:

12/11/25

* write a program for congestion control using Leaky bucket algorithm.





Time & Space complexity: O(K)
where K - no. of data packets in the queue.

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 no. 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 ("\n Second | Packet Recived | 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;
    }
    // Remaining packets after time ends
    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 no. of seconds you want to simulate:
5
Enter the size of the packet entering at 1sec : 3
Enter the size of the packet entering at 2sec : 7
  "      "    "   "    "      "     "  3sec : 4
  "      "    "   "    "      "     "  4sec : 6
  "      "    "   "    "      "     "  5sec : 5
```

Second	Packet Recived	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
6	0	4	2	0
7	0	2	0	0

Program 2:

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.

Code:

```
# 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:

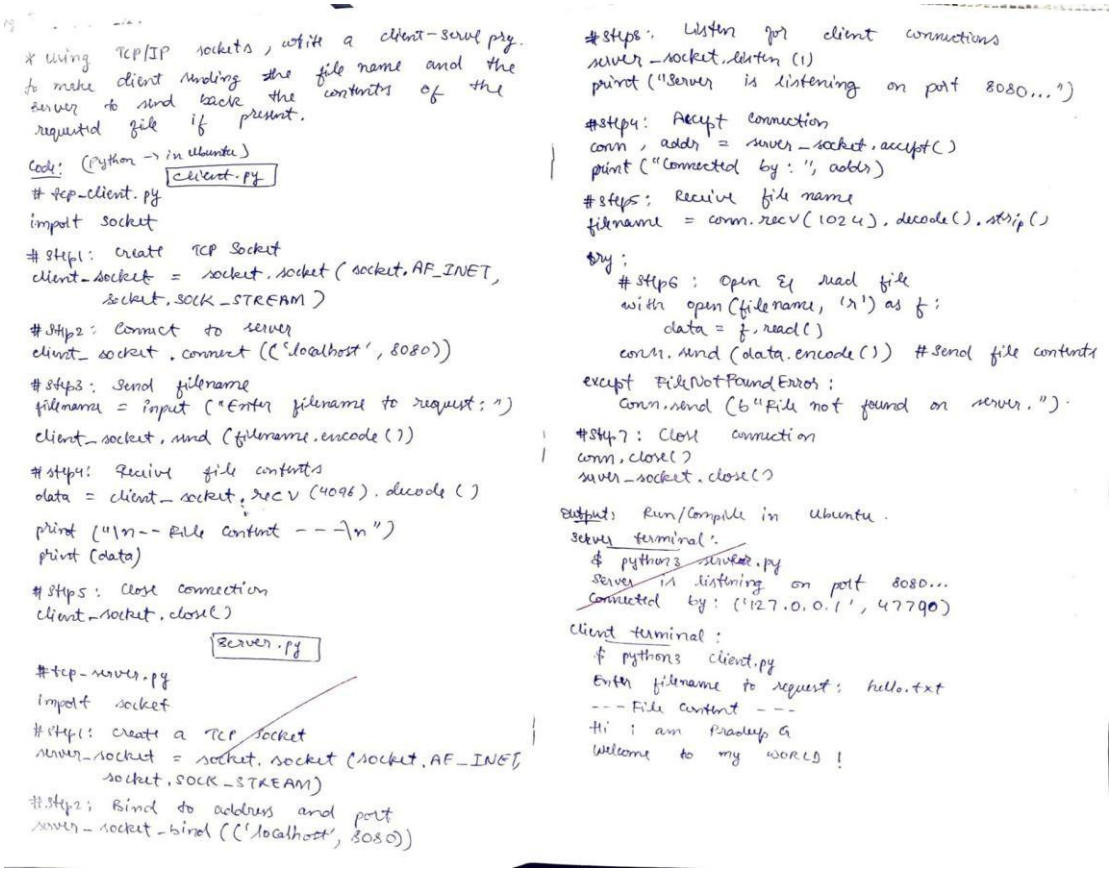
Server side Terminal:

```
pradeep-g@Pradeep-G: ~/Documents/TCP   ×   pradeep-g@Pradeep-G: ~/Documents/TCP   ×
pradeep-g@Pradeep-G:~/Documents/TCP$ python3 server.py
Server is listening on port 8080...
Connected by: ('127.0.0.1', 47790)
pradeep-g@Pradeep-G:~/Documents/TCP$
```

Client side Terminal:

```
pradeep-g@Pradeep-G: ~/Documents/TCP   ×   pradeep-g@Pradeep-G: ~/Documents/TCP   ×
pradeep-g@Pradeep-G:~/Documents/TCP$ python3 client.py
Enter filename to request: hello.txt

--- File Content ---

Hi i am Pradeep G
Welcome to my WORLD!

pradeep-g@Pradeep-G:~/Documents/TCP$
```

Observation:



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

Code: (Python → in ubuntu)

client.py

```
# tcp-client.py
import socket

# Step1: create TCP Socket
client_socket = socket.socket(socket.AF_INET,
    socket.SOCK_STREAM)

# Step2: Connect to server
client_socket.connect(('localhost', 8080))

# Step3: Send filename
filename = input("Enter filename to request: ")
client_socket.send(filename.encode())

# Step4: Receive file contents
data = client_socket.recv(4096).decode()

print("\n--- File Content ---\n")
print(data)

# Step5: Close connection
client_socket.close()
```

Server.py

```
#tcp-server.py
import socket

# Step1: create a TCP socket
server_socket = socket.socket(socket.AF_INET,
    socket.SOCK_STREAM)

# Step2: Bind to address and port
server_socket.bind(('localhost', 8080))

# Step3: Listen for client connections
server_socket.listen(1)
print("Server is listening on port 8080...")

# Step4: Accept connection
conn, addr = server_socket.accept()
print("Connected by:", addr)

# Step5: Receive file name
filename = conn.recv(1024).decode().strip()

try:
    # Step6: Open & 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.")

# Step7: Close connection
conn.close()
server_socket.close()
```

Output: Run/Compile in ubuntu.

Server terminal:

```
$ python3 server.py
Server is listening on port 8080...
Connected by: ('127.0.0.1', 47790)
```

Client terminal:

```
$ python3 client.py
Enter filename to request: hello.txt
--- File Content ---
Hi i am Pradeep G
Welcome to my WORLD!
```

Program 3:

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.

Code:

```
# udp_client.py
 import
socket


# Step 1: Create UDP socket
client_socket =
socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)


server_address = ('localhost',
8081)
filename = input("Enter filename
to request: ")


# Step 2: Send filename to server
client_socket.sendto(filename.en
code(), server_address)


# Step 3: Receive response
data, addr =
client_socket.recvfrom(4096)


print("\n--- File Content ---
\n") print(data.decode())


# Step 4: Close socket
client_socket.close()
```

```
# udp_server.py
 import
socket

# Step 1: Create UDP socket
server_socket =
socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)

# Step 2: Bind to address and port
server_socket.bind(('localhost',
8081))
print("UDP Server is ready...")
 while
True:
    # Step 3: Receive filename
from client     filename, addr
= server_socket.recvfrom(1024)
filename =
filename.decode().strip()
     print(f"Requested
file:
{filename}")

try:
        # Step 4: Open file and
send  content               with
open(filename, 'r') as f:
            data = f.read()

server_socket.sendto(data.
encode(), addr)
     except FileNotFoundError:
server_socket.sendto(b"Fil e not
found on server.", addr)
```

Output:

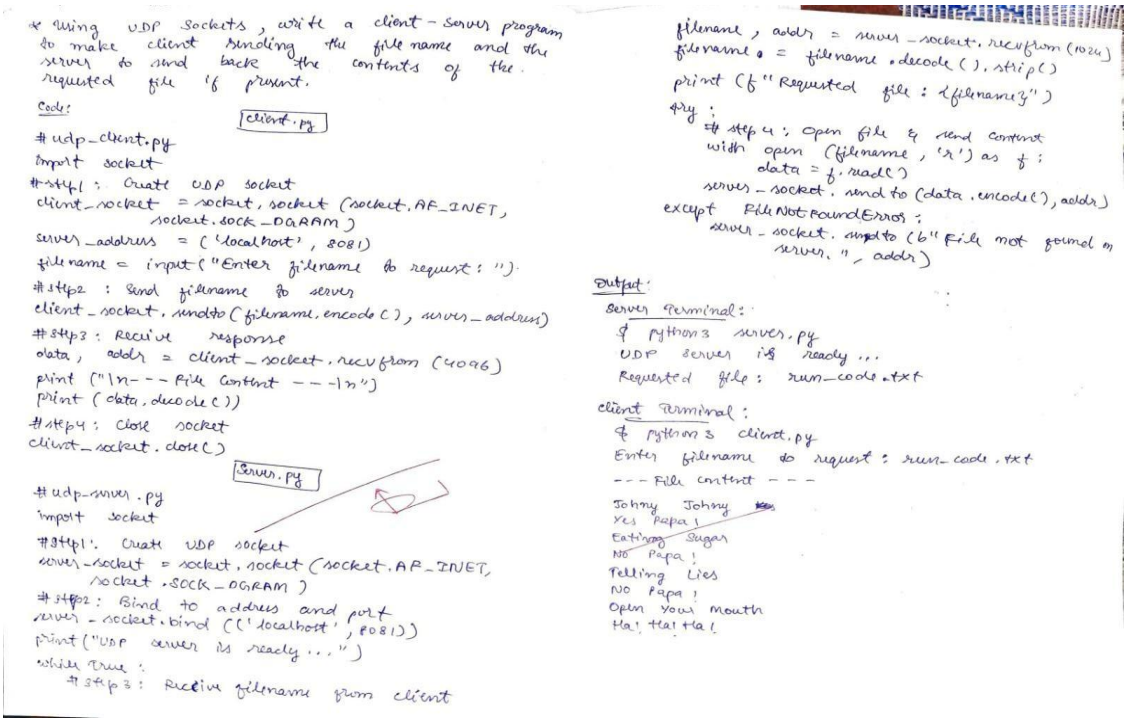
Server side Terminal:

```
pradeep-g@Pradeep-G: ~/Documents/UDP   pradeep-g@Pradeep-G: ~/Documents/UDP
pradeep-g@Pradeep-G:~/Documents/UDP$ python3 server.py
UDP Server is ready...
Requested file: run_code.txt
```

Client side Terminal:

```
pradeep-g@Pradeep-G: ~/Documents/UDP
pradeep-g@Pradeep-G: ~/Documents/UDP   pradeep-g@Pradeep-G: ~/Documents/UDP
pradeep-g@Pradeep-G:~/Documents/UDP$ python3 client.py
Enter filename to request: run_code.txt

--- File Content ---

▶️How to Run in Ubuntu
Terminal 1: Start the server
python3 udp_server.py

Terminal 2: Run the client
python3 udp_client.py

Enter a filename

Example:

sample.txt

pradeep-g@Pradeep-G:~/Documents/UDP$
```

Observation:



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.

Code:

client.py

```
# udp_client.py
import socket
# step1 : Create UDP socket
client_socket = socket.socket(socket.AF_INET,
                socket.SOCK_DGRAM)
server_address = ('localhost', 8081)
filename = input("Enter filename to request: ")
# step2 : Send filename to server
client_socket.sendto(filename.encode(), server_address)
# step3 : Receive response
data, addr = client_socket.recvfrom(4096)
print("\n--- File Content ---\n")
print(data.decode())
# step4 : Close socket
client_socket.close()
```

server.py

```
# udp_server.py
import socket
# step1: Create UDP socket
server_socket = socket.socket(socket.AF_INET,
                socket.SOCK_DGRAM)
# step2: Bind to address and port
server_socket.bind(('localhost', 8081))
print("UDP server is ready...")
while True:
    # step3: Receive filename from client
    filename, addr = server_socket.recvfrom(1024)
    filename = filename.decode().strip()
    print(f"Requested file: {filename}")
    try:
        # step4: Open file & send content
        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)
```

Output:

Server Terminal:

```
$ python3 server.py
UDP server is ready...
Requested file: run_code.txt
```

Client Terminal:

```
$ python3 client.py
Enter filename to request: run_code.txt
--- File Content ---
Johny Johny
Yes Papa!
Eating Sugar
No Papa!
Telling Lies
No Papa!
Open your mouth
Ha! Ha! Ha!
```

Program 4:

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

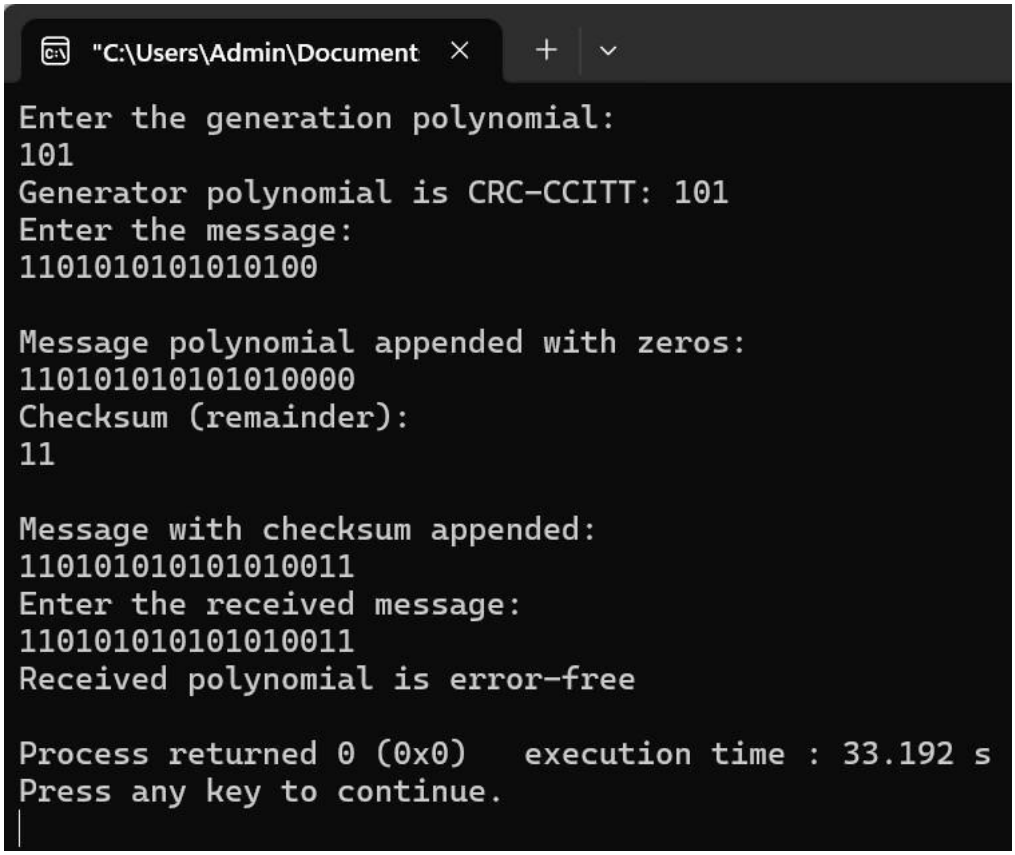
Code:

```
#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])
```

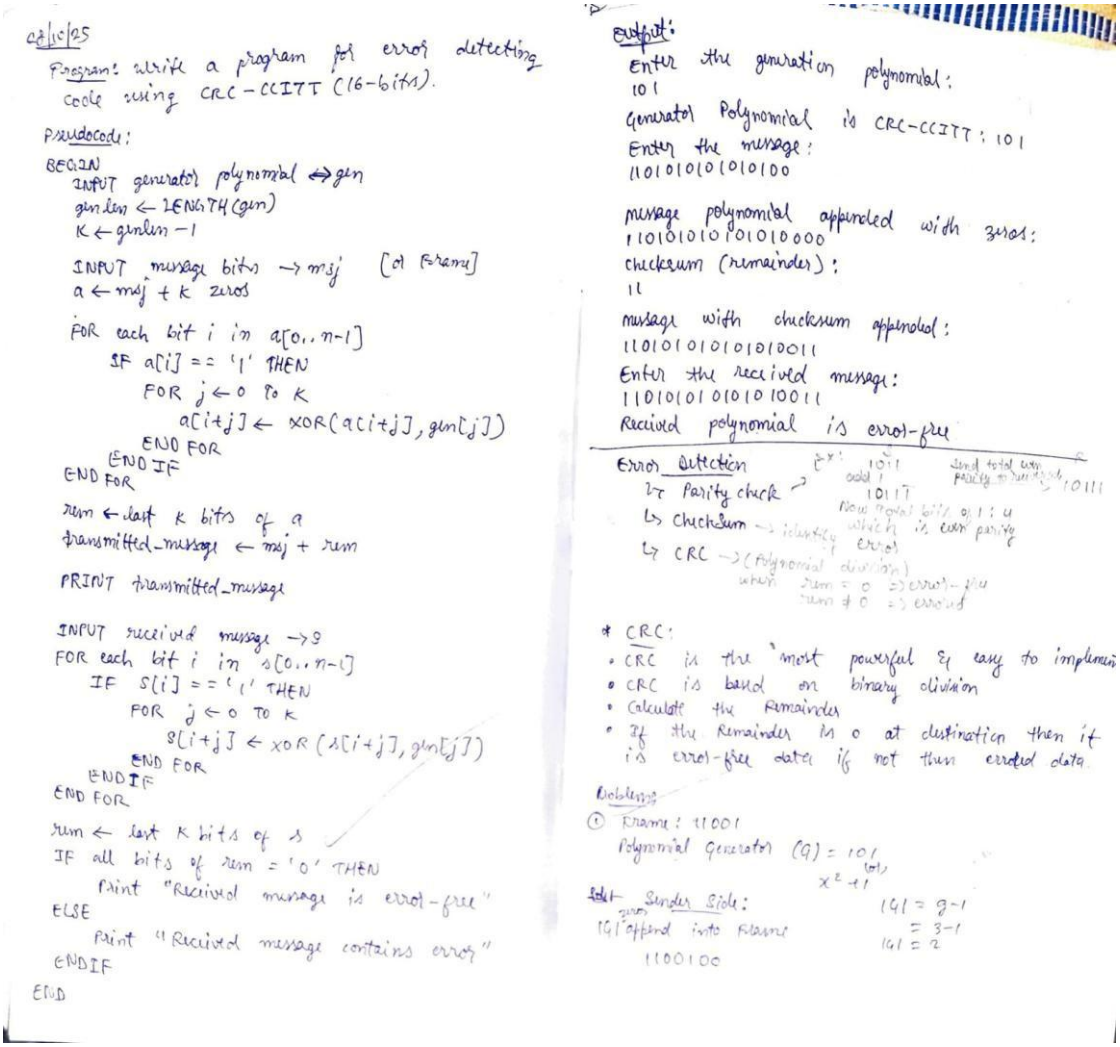
```
a[t] = '0';                 else
a[t] = '1';                 t++;
            }
        }
    }      for (i = 0; i <
k; i++)         rem[i] =
a[n + i];     rem[k] =
'\0';
     printf("Checksum
(remainder):\n");     puts(rem);
     printf("\nMessage with checksum
appended:\n");     for (i = 0; i < 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:



Observation:



a2/10/25

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

Pseudocode:

```
BEGIN
  INPUT generator polynomial <=> gen
  genlen <- LENGTH(gen)
  K <- genlen - 1

  INPUT message bits -> msj   [of frame]
  a <- msj + K zeros

  FOR each bit i in a[0..n-1]
     IF a[i] == '1' THEN
        FOR j <- 0 to K
           a[i+j] <- XOR(a[i+j], gen[j])
        END FOR
     END IF
  END FOR

  rem <- last K bits of a
  transmitted_message <- msj + rem

  PRINT transmitted_message

  INPUT received message -> s
  FOR each bit i in s[0..n-1]
     IF s[i] == '1' THEN
        FOR j <- 0 TO K
           s[i+j] <- XOR(s[i+j], gen[j])
        END FOR
     END IF
  END FOR

  rem <- last K bits of s
  IF all bits of rem = '0' THEN
     Print "Received message is error-free"
  ELSE
     Print "Received message contains error"
  ENDIF
END
```

Output:

```
Enter the generation polynomial:
101
Generator Polynomial is CRC-CCITT: 101
Enter the message:
11010101010100

message polynomial appended with zeros:
11010101010100000
checksum (remainder):
11
message with checksum appended:
11010101010100011
Enter the received message:
11010101010100011
Received polynomial is error-free
```

Error Detection

- Parity check
- Checksum → identify which is error
- CRC → (Polynomial division)
  - when rem = 0 => error-free
  - rem ≠ 0 => errored

* CRC:

- CRC is the most powerful & easy to implement
- CRC is based on binary division
- Calculate the Remainder
- If the Remainder is 0 at destination then it is error-free data if not then errored data

Problem:

① Frame: 11001

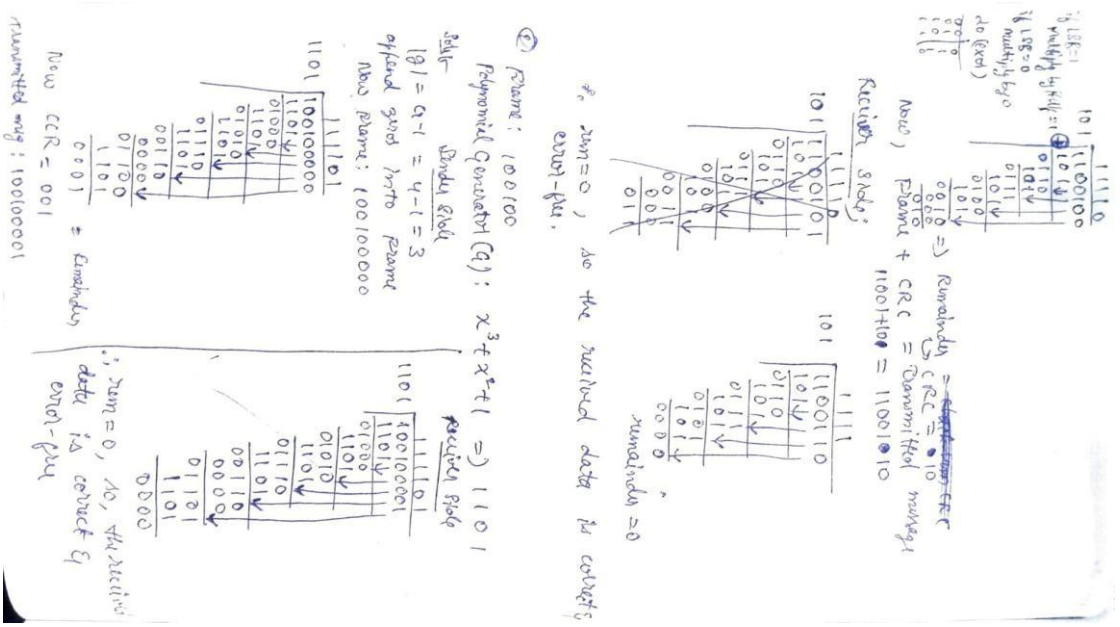
Polynomial Generator (G) = 101 (or) x^2+1

Sender Side:

|G| = g-1 = 3-1 = 2

|G| append into Frame

1100100



② Frame: 100100

Polynomial Generator (G): x^3+x^2+1 => 1101

Sender Side

|G| = 4-1 = 3

append zeros into Frame

New Frame: 100100000

New CRC = 001

Remainder

Transmitted msg: 100100001

rem = 0, so the received data is correct & error-free.

Receiver Side:

New) Frame + CRC = 11001010

1100+100 = 11001010

1) Remainder = CRC = 010

Transmitted message

Remainder = 0