VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

Mithun.M(1BM23CS192)

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 **Mithun.M (1BM23CS192),** 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

Part - A

Sl. No.	Date	Experiment Title	Page No.
1	19/08/25	Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.	1 – 3
2	09/09/25	Configure DHCP within a LAN and outside LAN.	4 – 5
3	09/09/25	Configure Web Server, DNS within a LAN.	6 – 7
4	09/09/25	Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.	8 – 9
5	23/09/25	Configure default route, static route to the Router.	10 – 12
6	23/09/25	Configure RIP routing Protocol in Routers.	13 – 15
7	14/10/25	Configure OSPF routing protocol.	16 – 17
8	14/10/25	To construct a VLAN and make the PC's communicate among a VLAN.	18 – 19
9	11/11/25	To construct a WLAN and make the nodes communicate wirelessly.	20 – 22
10	11/11/25	Demonstrate the TTL/ Life of a Packet.	23 – 24
11	18/11/25	To understand the operation of TELNET by accessing the router in server room from a PC in IT office.	25 – 27
12	18/11/25	To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP).	28 – 30

Part - B

Sl. No.	Date	Experiment Title	Page No.
1	28/10/25	Write a program for congestion control using Leaky bucket algorithm.	31 – 34
2	17/11/25	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.	35 – 36
3	17/11/25	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.	37 – 38
4	28/10/25	Write a program for error detecting code using CRC-CCITT (16-bits).	39 - 43

GitHub Link:
https://github.com/Mithun-M2004/CN_1BM23CS192

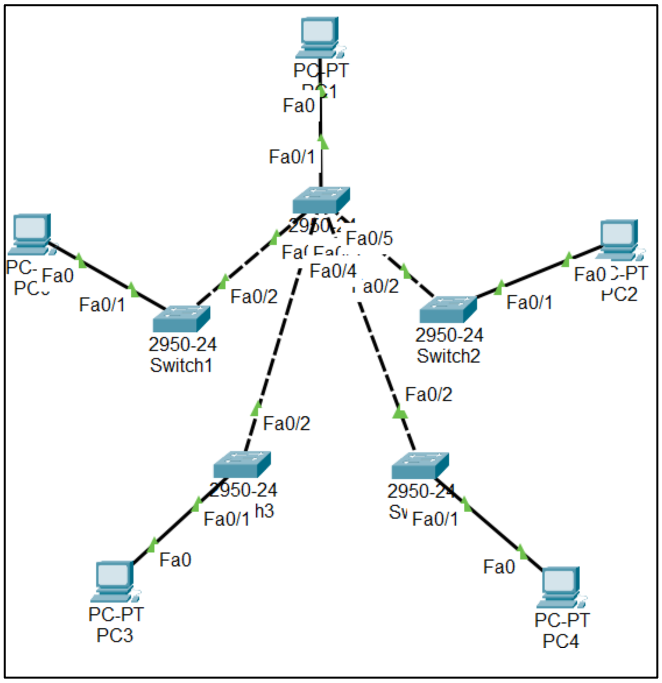
PART - A

Program 1:

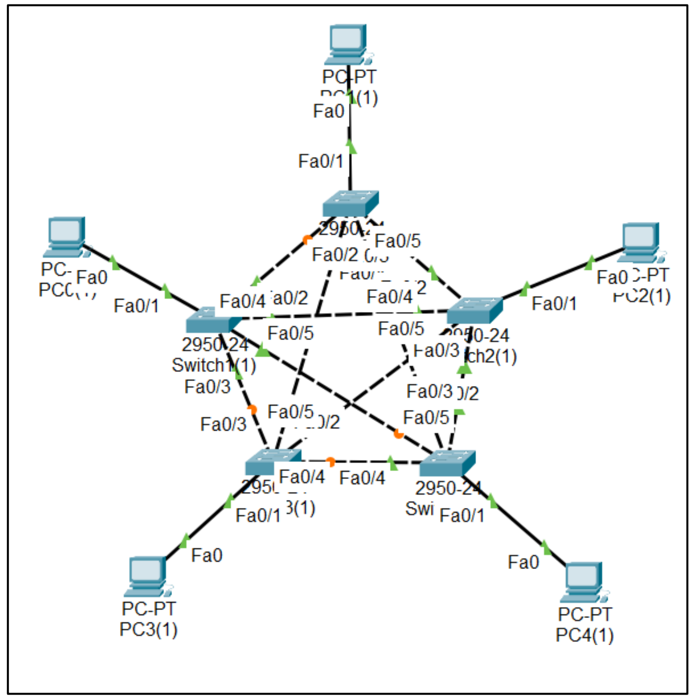
Aim: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

Network diagram:

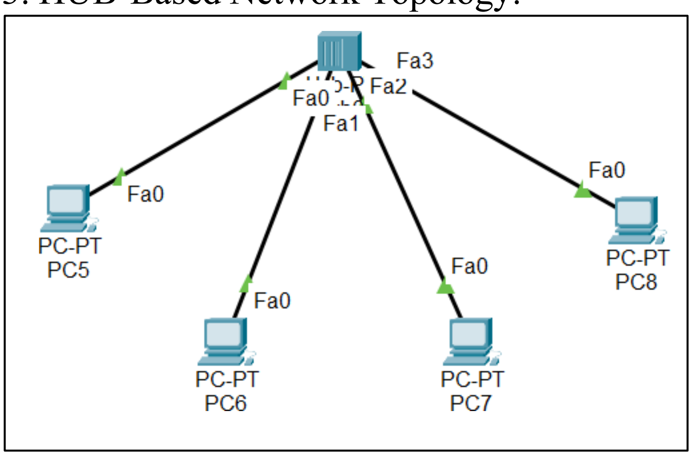
1. STAR Topology with Switch:



2. MESH Topology with Switch:



3. HUB-Based Network Topology:



Configuration:

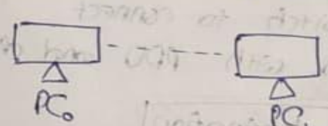
Lab- I

1) Scenario: Communication between two PC

- Create two PC, PC_0, PC_1
- Connect them using copper wire (crossover)
- Click the PC_0 setup its IP address to 192.168.1.2
- Click on the PC_1 setup it IP address to 192.168.1.3
- To test both PC are pinging add sample PDU & click the status.

Output

Status	Source	Destination
Successfull	PC_0	PC_1

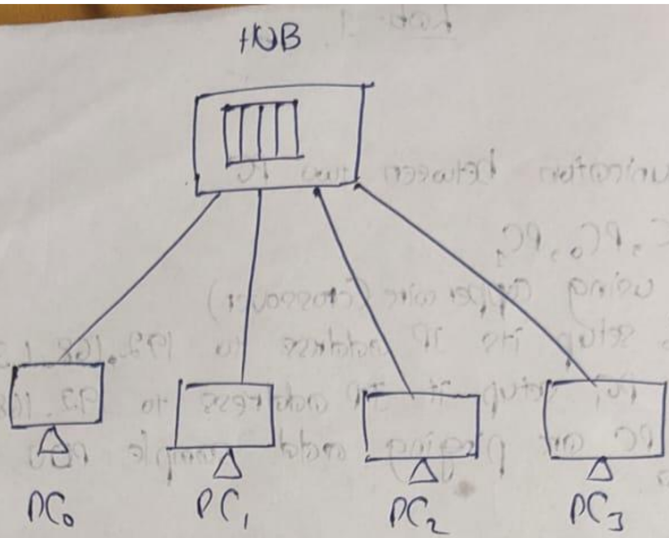


2) Scenario: Communication between PC's via a hub

- Create some PC's (PC_0, PC_1, PC_2, PC_3)
- Setup a hub
- Connect each PC to the hub using copper straight through wire
- Click on each PC, assign IP address to each PC and subnet mask 255.255.255.0
- To test sample ping add a sample PDU from PC to another PC (PC_2) and check the status.

Output

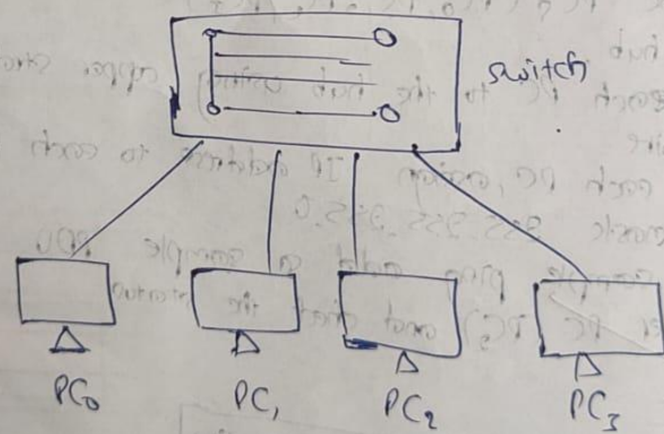
Status	Source	Destination
Successful	PC_0	PC_2



3) Scenario: Communication between PC's via switch

i) follow the same steps as above
ii) instead of hub use switch to connect
iii) then ping any Two PCs with PDU and check status

Status	source	destination
Successful	PC_0	PC_2



4) Scenario : Mesh network using switches

Setup PCs in a mesh pattern, for each PC setup a unique switch

Connect each PC to a unique switch

Connect each switch to other switch.

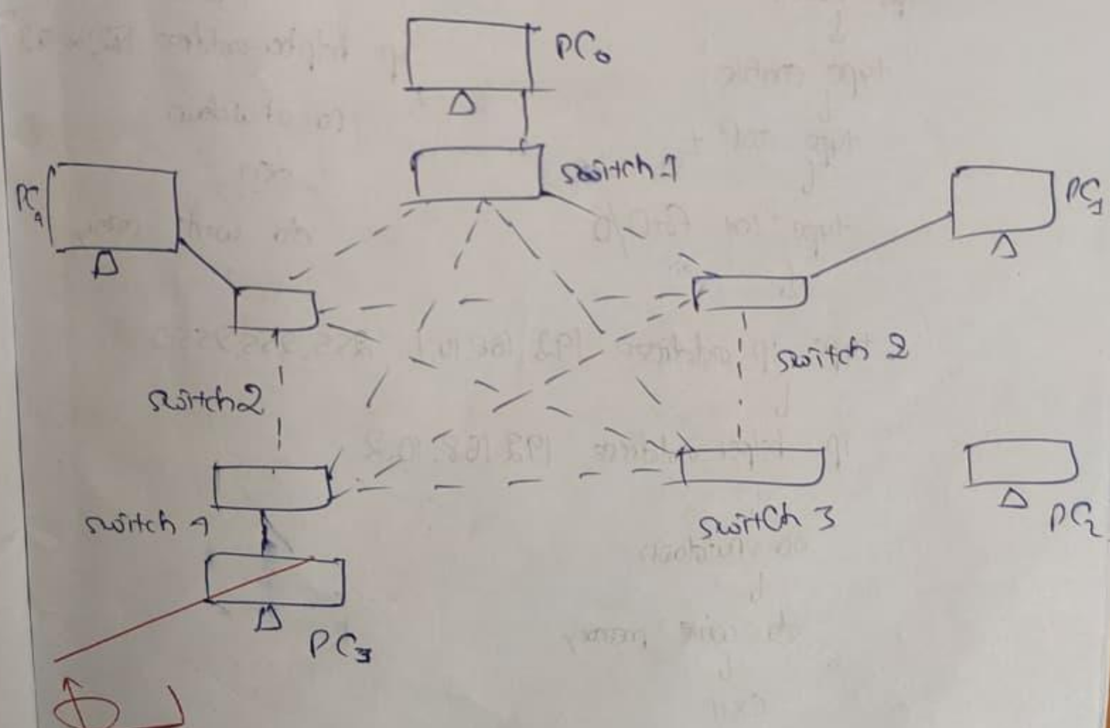
Setup the IP address of each PC where from 192.168.1.5

Setup the subnet mask to 255.255.255.0

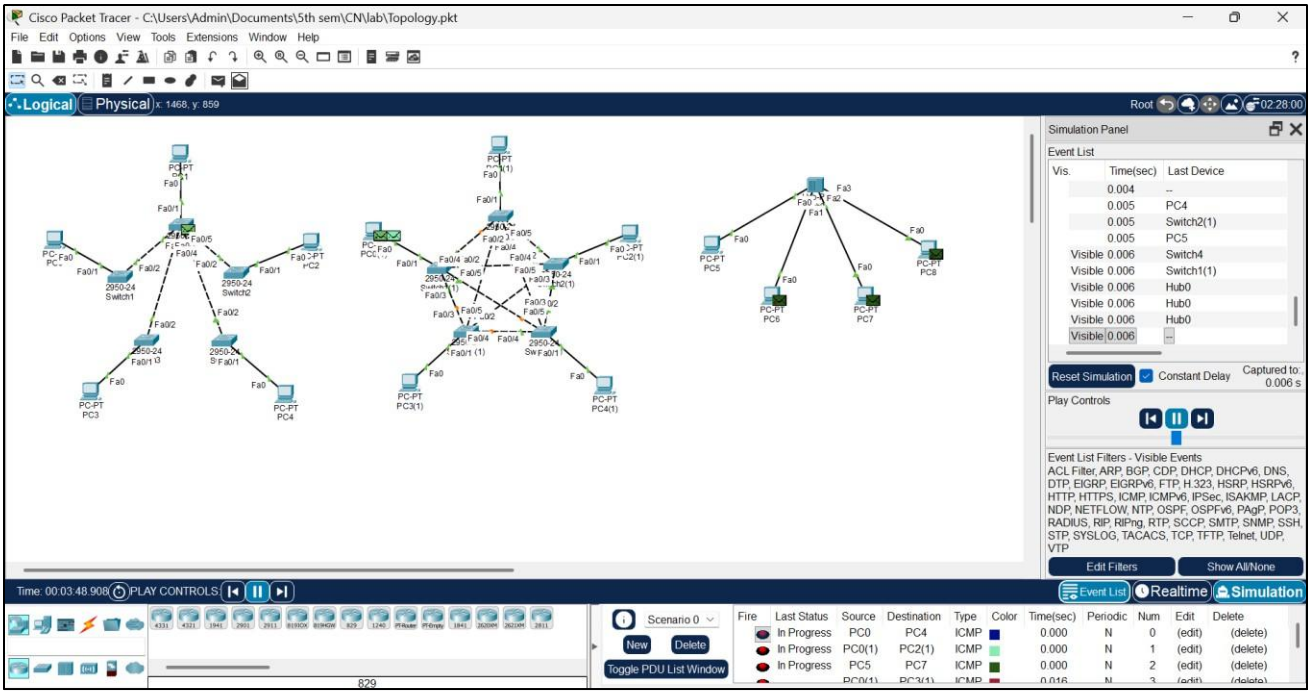
Test the connectivity by adding simple PDU

Output

Status	Source	Destination
successful	PC_0	PC_4
Successful	PC_1	PC_2



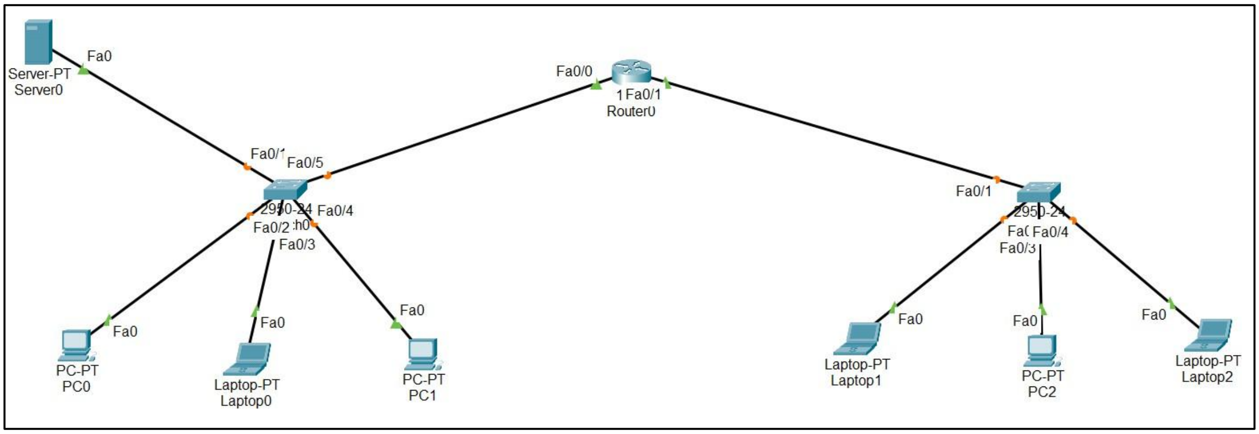
Output:



Program 2:

Aim: Configure DHCP within a LAN and outside LAN.

Network diagram:



Configuration:

z

* Configure ~~DSE~~ DHCP within a lan and outside a lan.

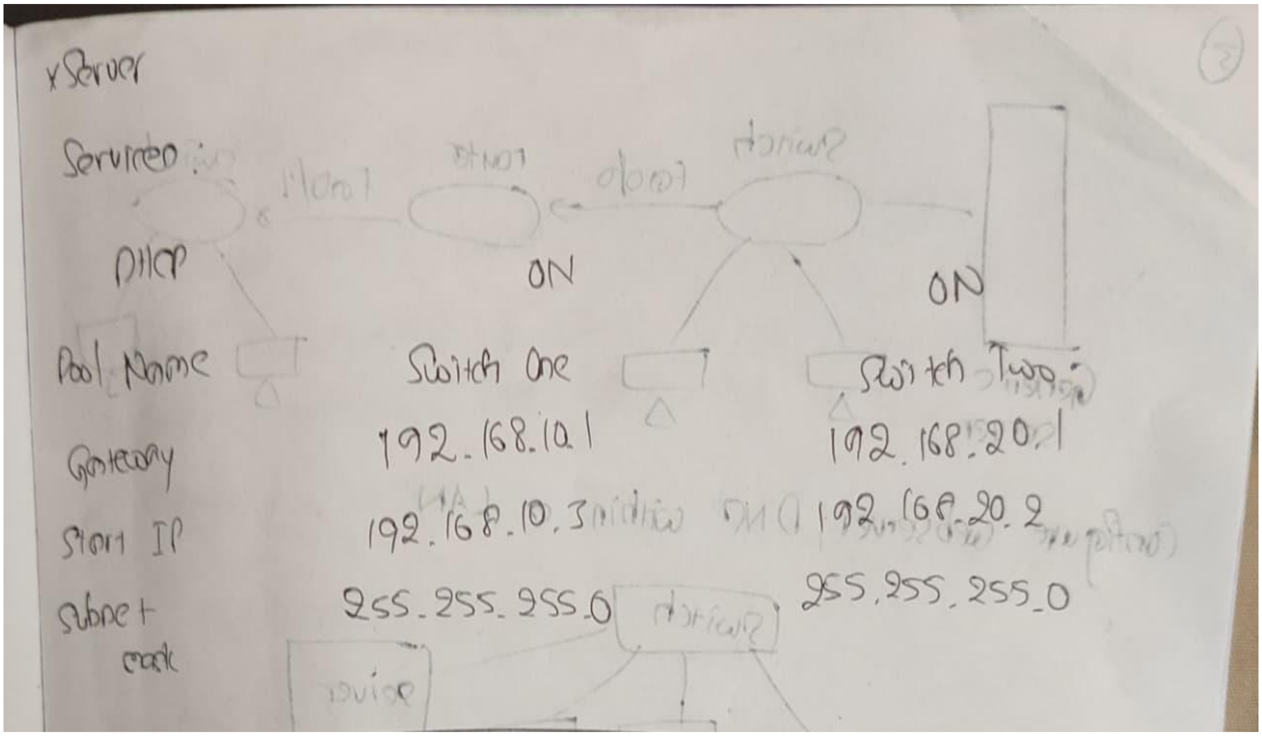
→ Click on DHCP server
→ Go for desktop
→ Go for IP
→ Select static & Give IP.

Then go to router
↓
CLI
↓
type No
↓
type router
↓
type enable
↓
type conf t
↓
type int fa0/0
↓
type ip address 192.168.10.1 255.255.255.0
↓
ip helper-address 192.168.10.2
↓
no shutdown
↓
do write memory
↓
exit

~~int fa0/~~
int Fa0/1
↓
ip address 192.168.20.1
255.255.255.0
ip helper-address 192.168.10.2
no shutdown
exit
do write memory

DHCP
↳ Desktop IP config
↳ static
↳ 192.168.10.2
↳ 192.168.10.1

Gateway



Output:

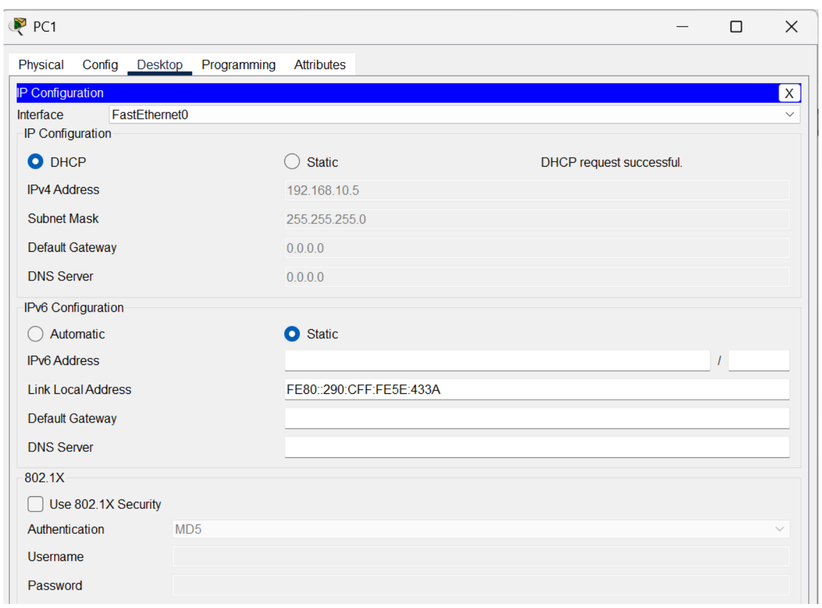


Fig 1. Ip address assigned by DHCP server within Lan (PC1)

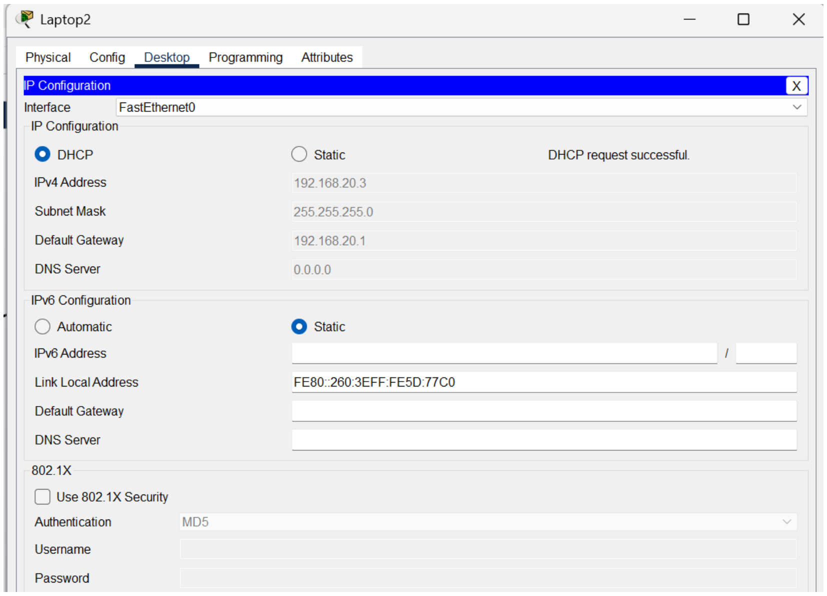
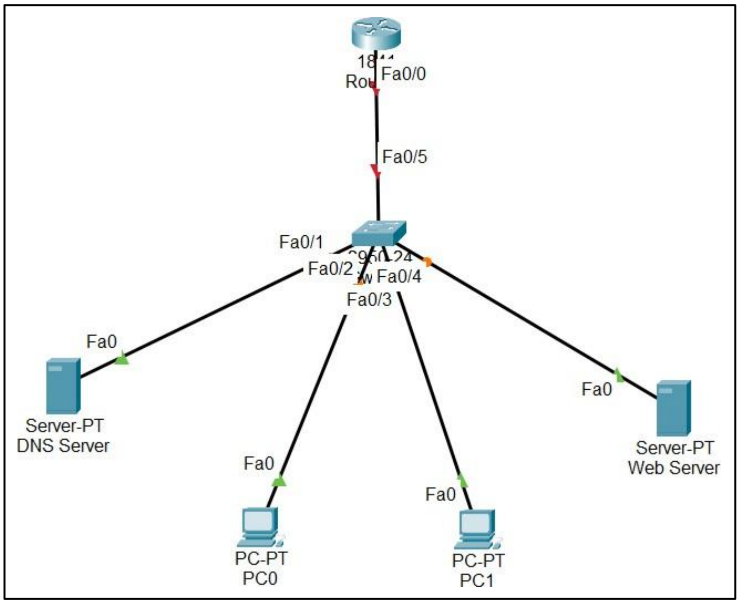


Fig 2. Ip address assigned by DHCP server outside Lan (laptop2)

Program 3:

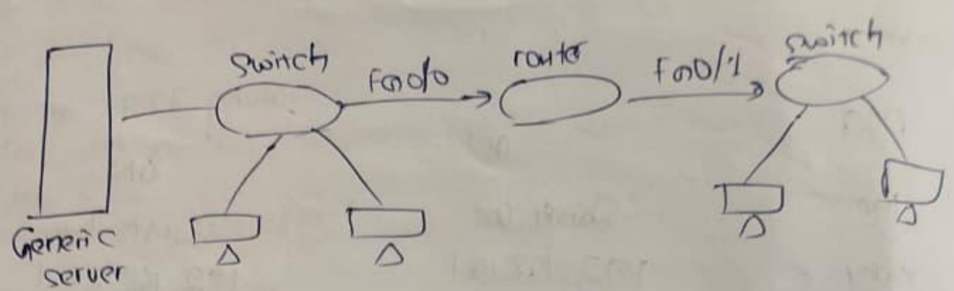
Aim: Configure Web Server, DNS within a LAN.

Network diagram:

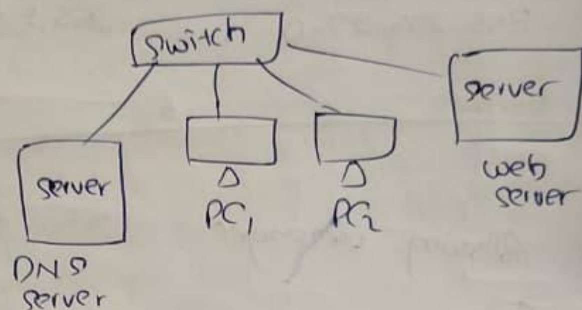


Configuration:

(5)



configure webserver, DNS within a LAN



DNS server config:

ip : 192.168.1.5
subnet : 255.255.255.0
DNS server : 192.168.1.5

webserver config
192.168.1.6
255.255.255.0
192.168.1.5

services:

HTTP → on
HTTPS → on

click on any one file name in file manager → edit

DNS server → services → DNS →

name : www.mywebsite.com
address : 192.168.1.6
click add

Output:

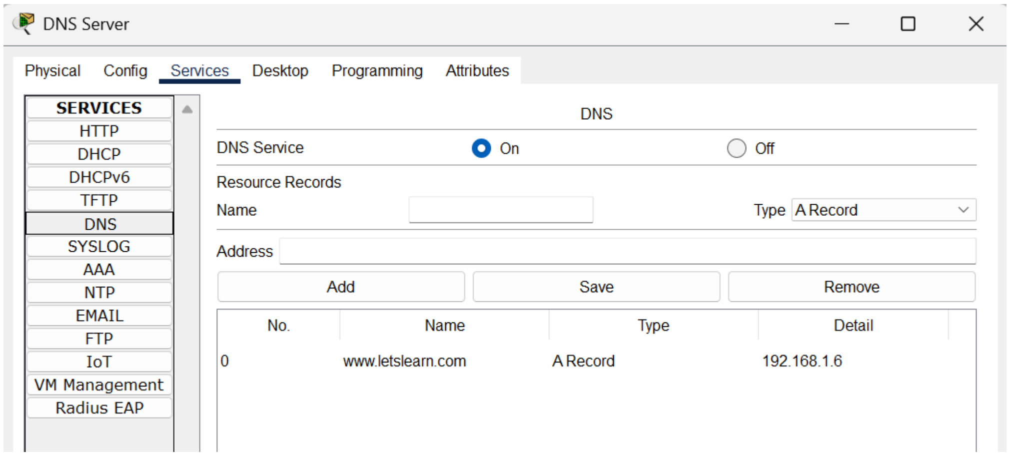


Fig 1. DNS server – DNS Services

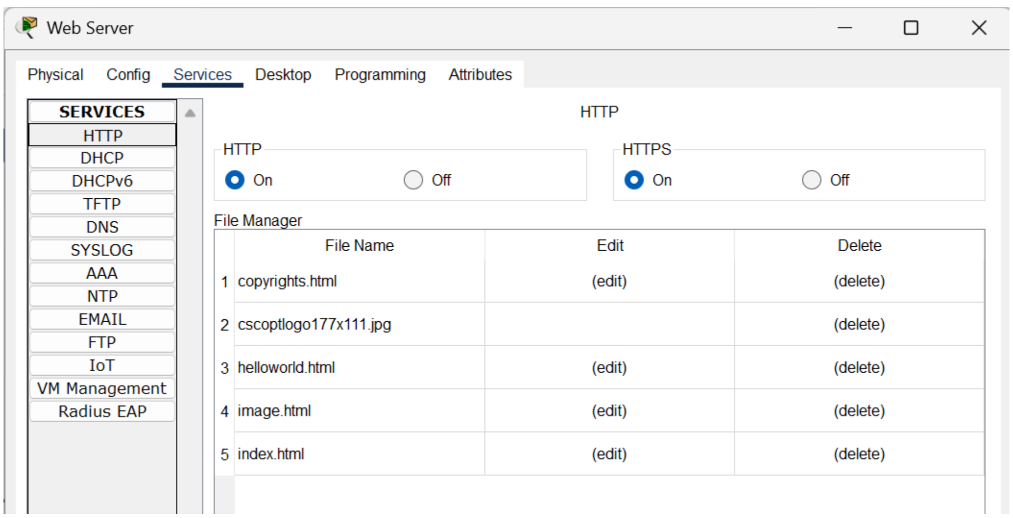


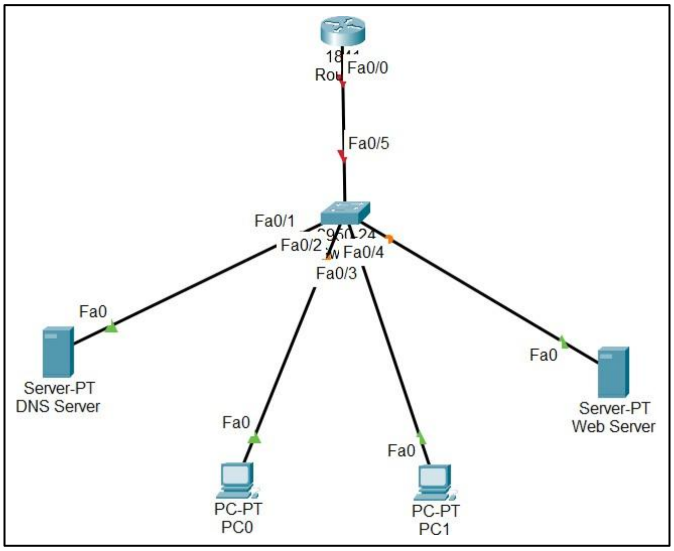
Fig 2. WEB server – HTTP Services

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:

URL: http://mywebsite.com

(Q) Configure ip address of router in packet tracer
Explore the following message
i) ping response
ii) destination unreachable
iii) request timeout
iv) reply

Ping:
It can be checked in any end devices
and prompt> ping ip
* The destination is reachable
IP addresses are correct & network path works
* Both source & destination are configured properly & connected

* destination unreachable:
router devices know the destination network doesn't exist (or) not route to it

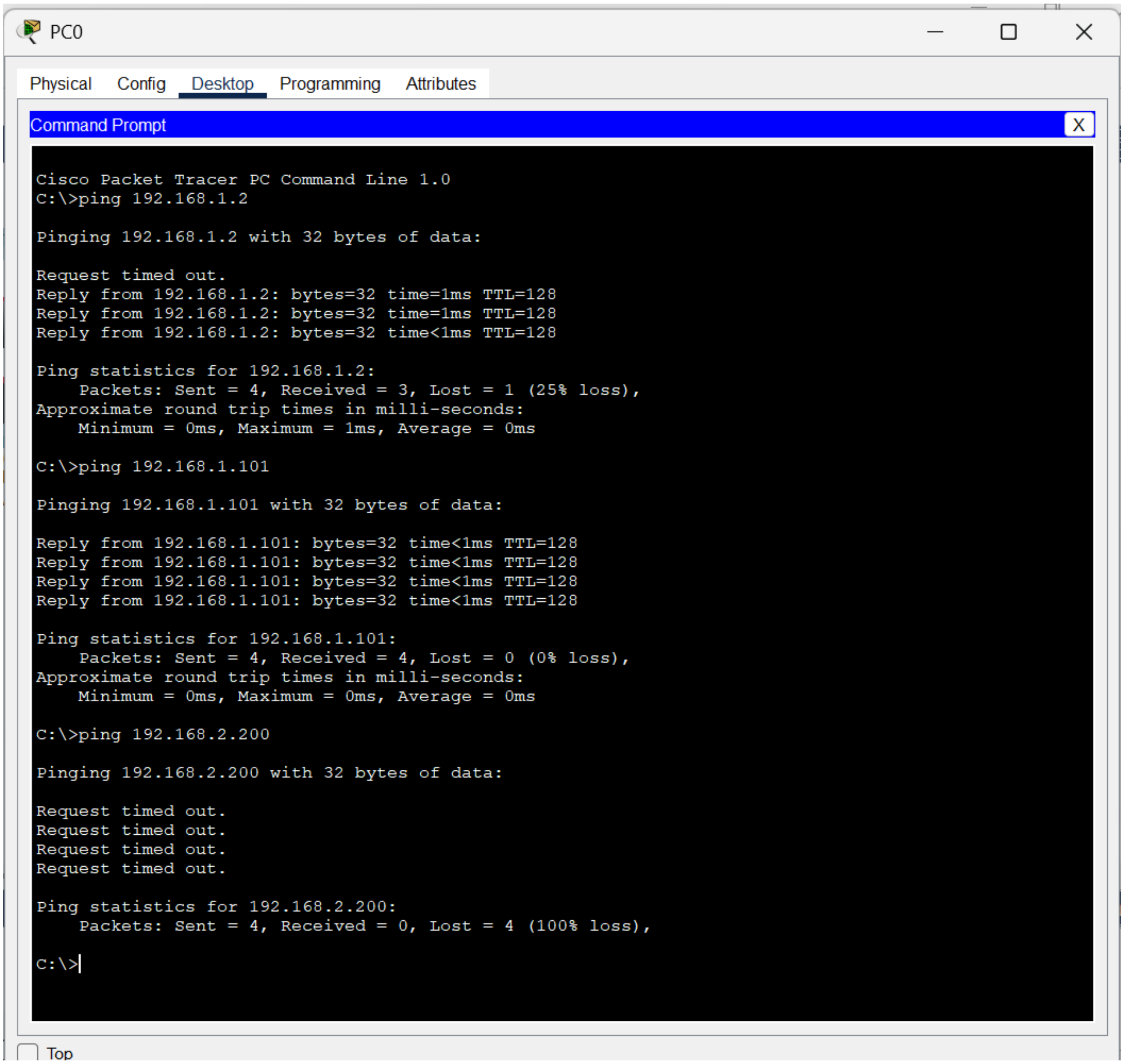
causes:
wrong IP addresses (or) subnet mask
missing route in routers routing table
No interface configured for network

* Request timeout:
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

reply:
* This is the successful case where packet reaches correctly
* It shows network connectivity is fine.

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.

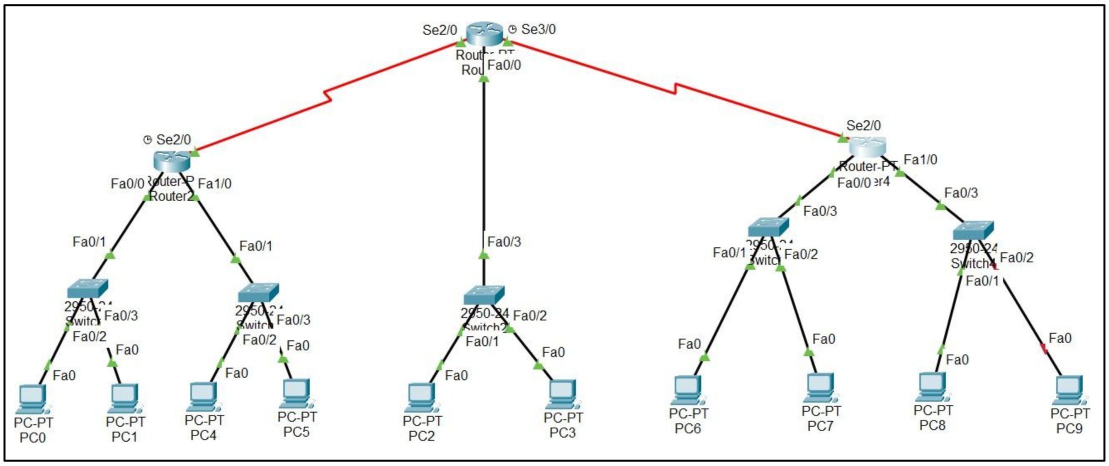
Ping statistics for 192.168.2.200:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>
Top

Program 5:

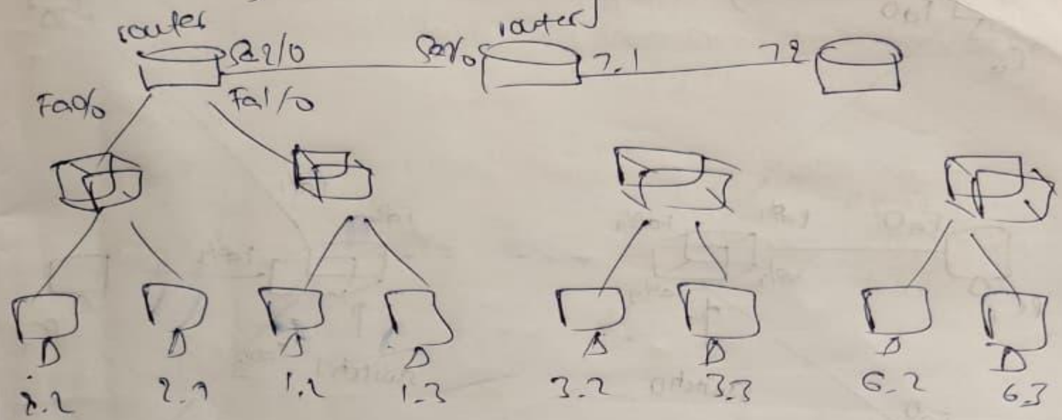
Aim: Configure default route, static route to the Router.

Network diagram:



Configuration:

LAB 5: Configure default route, static route to router
LAB 6: configure RIP routing protocol in router



Router PT:

router 2 → config
FastEthernet 0/0 IP = (default Gateway) → 2.1
1/0
serial 2/0 IP = 192.168.1.1

In PC
For each LAN unique ip address
subnet mask + default gateway same for all devices within the LAN

~~Static Routes~~:
Router 2
2.2
2.3
Router 7
7.1
7.2
Router 6 config static

dynamic routing:
RIP - Routing information protocol
at each router add known networks.
Ex: Router 2 192.168.1.1
192.168.2.0
192.168.7.0

Output:

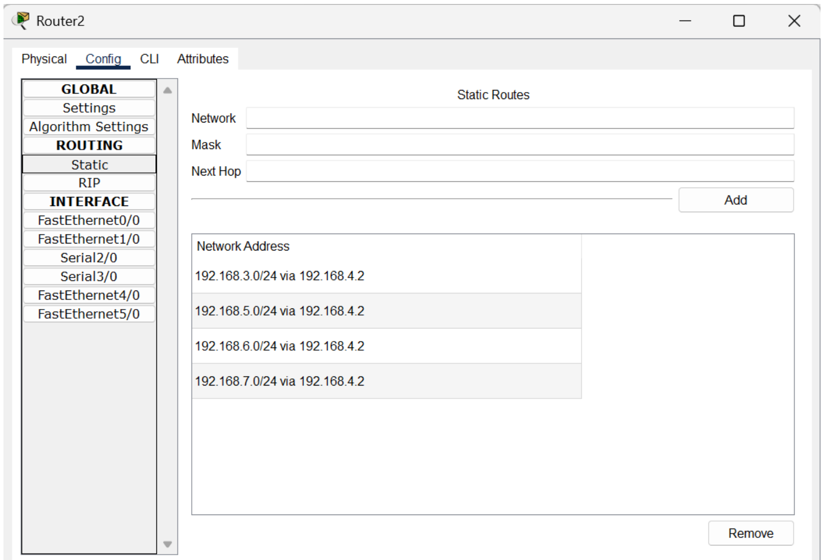


Fig 1. Router 2 – Static routing

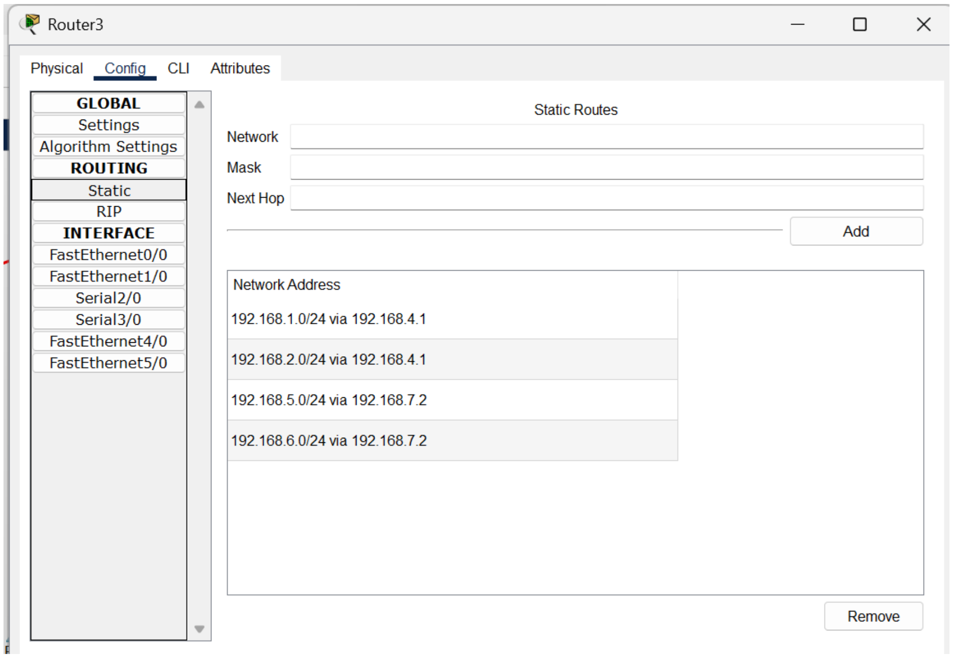


Fig 2. Router 3 – Static routing

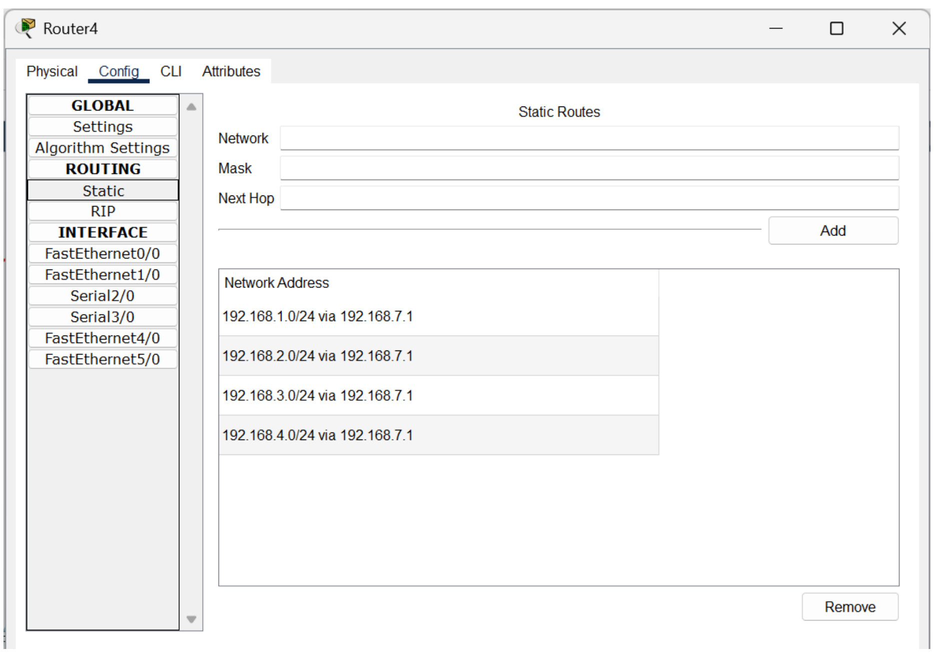
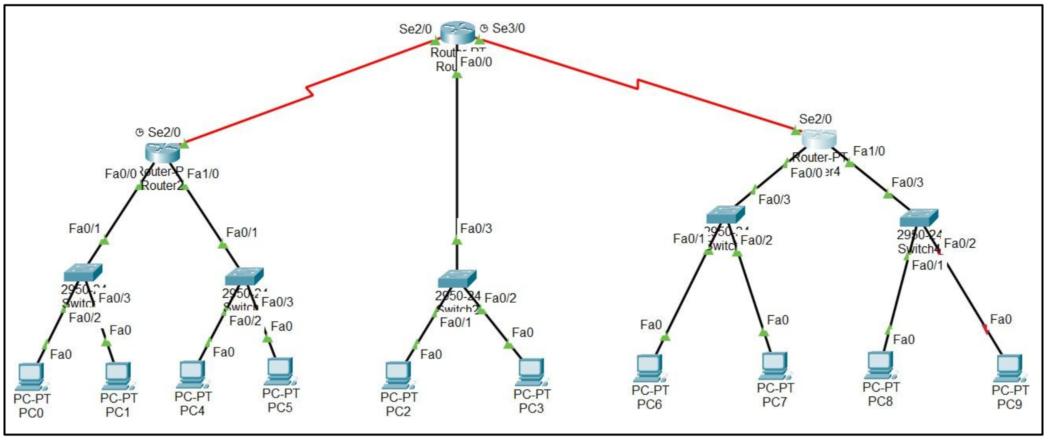


Fig 3. Router 4 – Static routing

Program 6:

Aim: Configure RIP routing Protocol in Routers.

Network diagram:



Configuration:

② Dynamic Route :-

Dynamic Routing is a networking technique where routers automatically and adaptively share routing information using protocols to find the best path for data to travel across a network.

Connections:

Same as static Routing, but we have to remove all static Routes [under Routing] from all routers & assign the Dynamic Routing, i.e.,

* Router 1: (select Router-PT)
↳ config
↳ Routing
↳ RIP Routing
↳ Network: 192.168.1.0
192.168.2.0
192.168.4.0
then click on add [For each]

* Router 2:
↳ Config
↳ Routing
↳ RIP Routing
↳ Network: 192.168.3.0
192.168.4.0
192.168.7.0
then click on add [For each]

* Router 3:
↳ config
↳ Routing
↳ RIP Routing
↳ Network: 192.168.5.0
192.168.6.0
192.168.7.0
then click on add [For each]

Output:

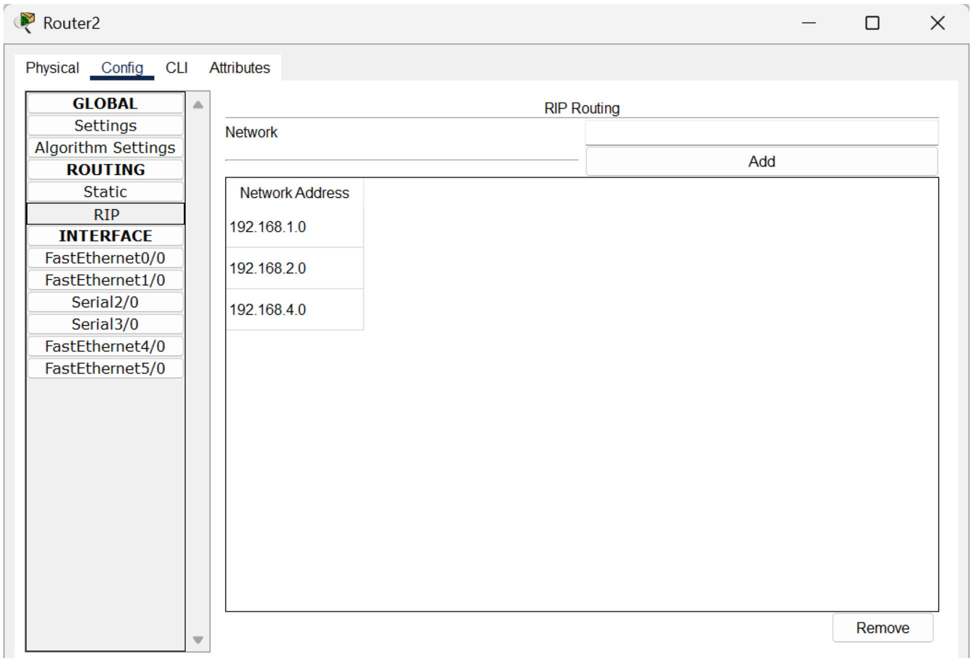


Fig 1. Router 2 – RIP routing

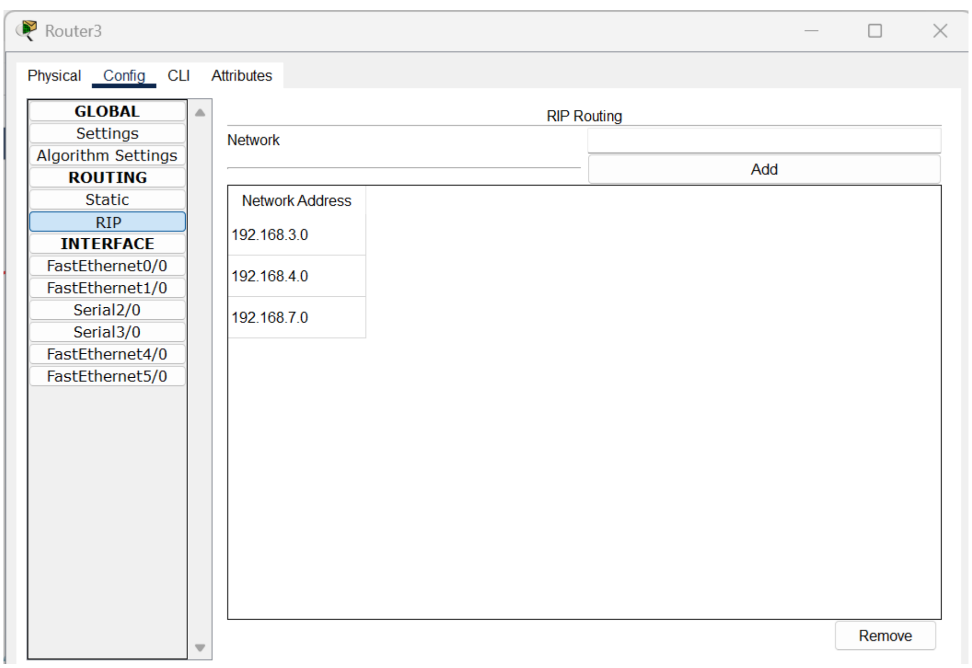


Fig 2. Router 3 – RIP routing

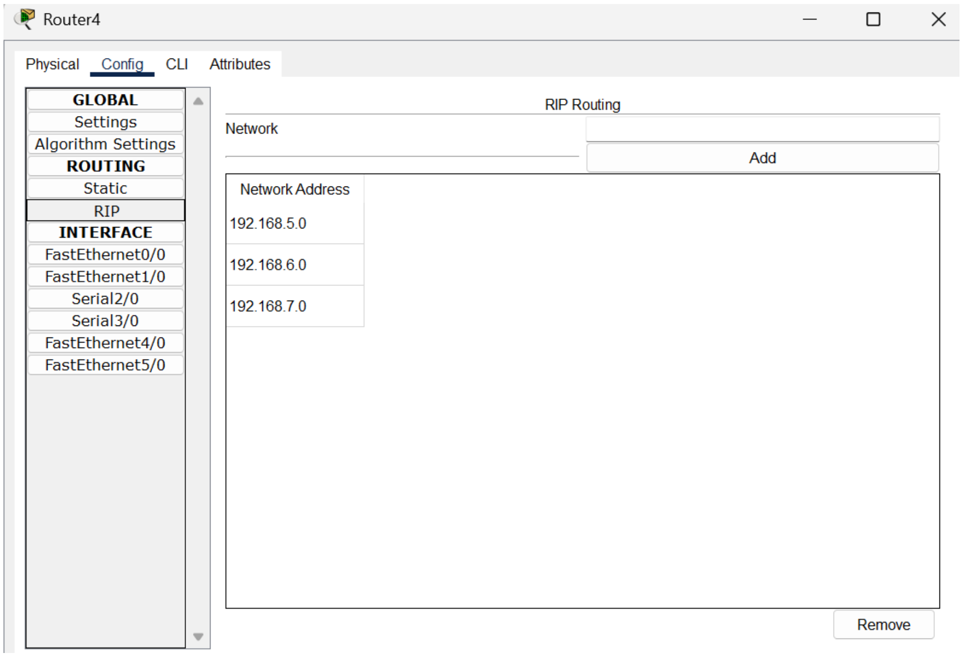
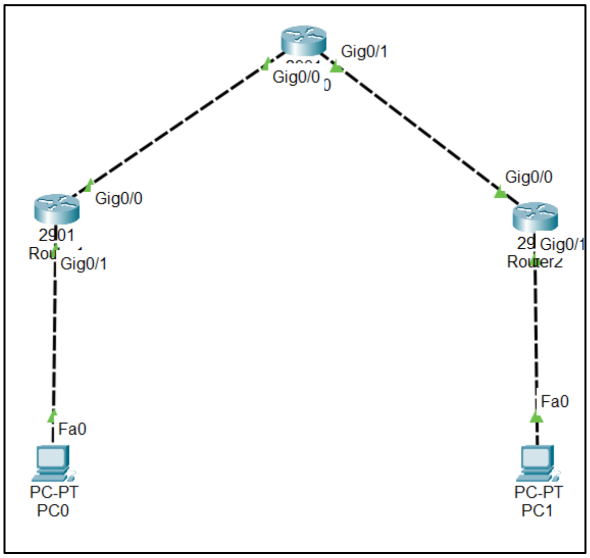


Fig 3. Router 4 – RIP routing

Program 7:

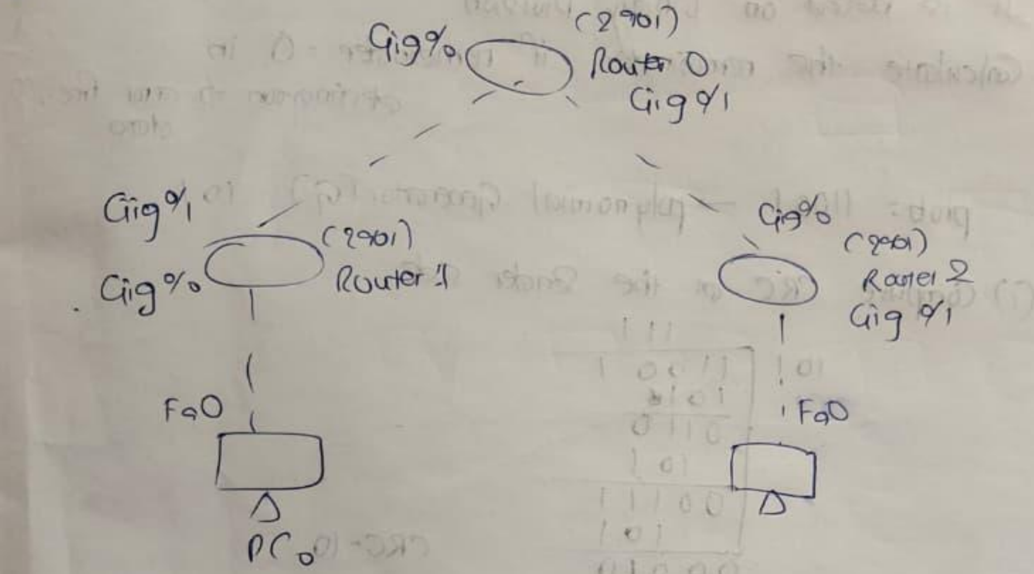
Aim: Configure OSPF routing protocol.

Network diagram:



Configuration:

② OSPF :



→ Code :-

```
Router 1 :  >enable
     # config t
     # router ospf 1
     # network  192.168.55.0    0.0.0.255  area 0
     # network  192.168.99.0    0.0.0.255  area 0
     # exit
Router 0 :  >enable
     # config t
     # router ospf 1
     # network  192.168.55.0    0.0.0.255  area 0
     # network  172.16.0.0      0.0.255.255  area 0
     # exit

Router 2 :  >enable
     # config t
     # router ospf 1
     # network  172.16.0.0    0.0.255.255  area 0
     # network  10.0.0.0      0.255.255.255  area 0
     # exit
```

* CRC is more powerfull & easy to implement
* It is based on Binary Division
* Calculate the remainder if remainder = 0 in destination => error free data

prob: 11001 → polynomial Generator (G) 101

(i) Compute CRC at the sender side

```
      111
101 | 11001
      101
      0110
       101
      00111
        101
      00010
```

CRC = 10

```
      1111
101 | 1100110
      101
      0110
       101
      0011 1
        101
      0001 0 1
          1 0 1
      0 00000
```

Output:

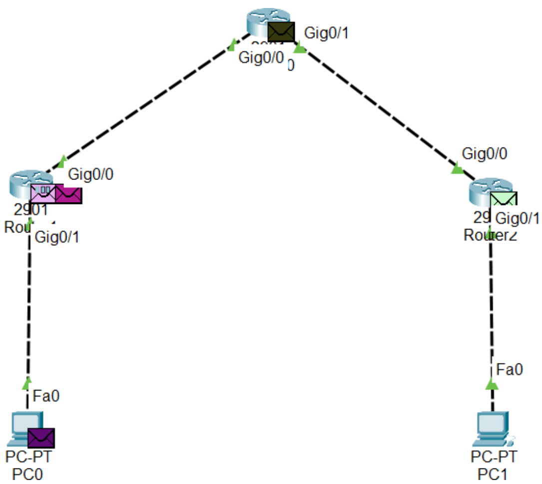
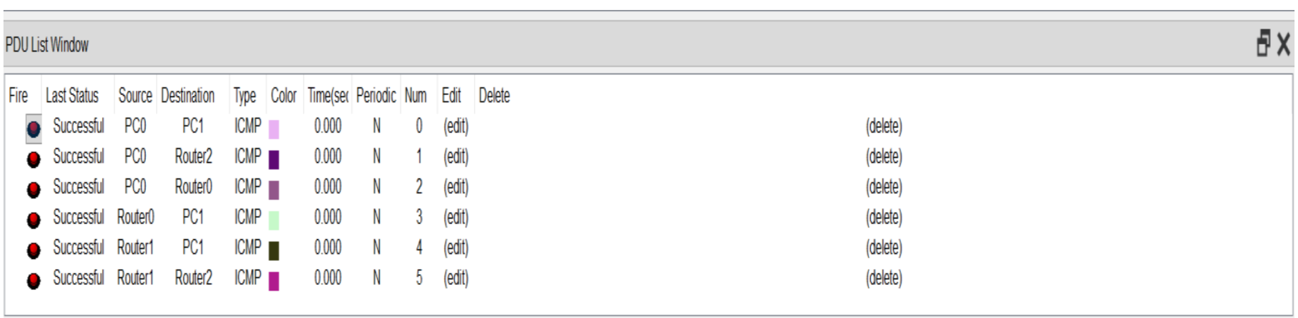


Fig 1. Sending PDU message from PC0 to PC1



PDU List Window

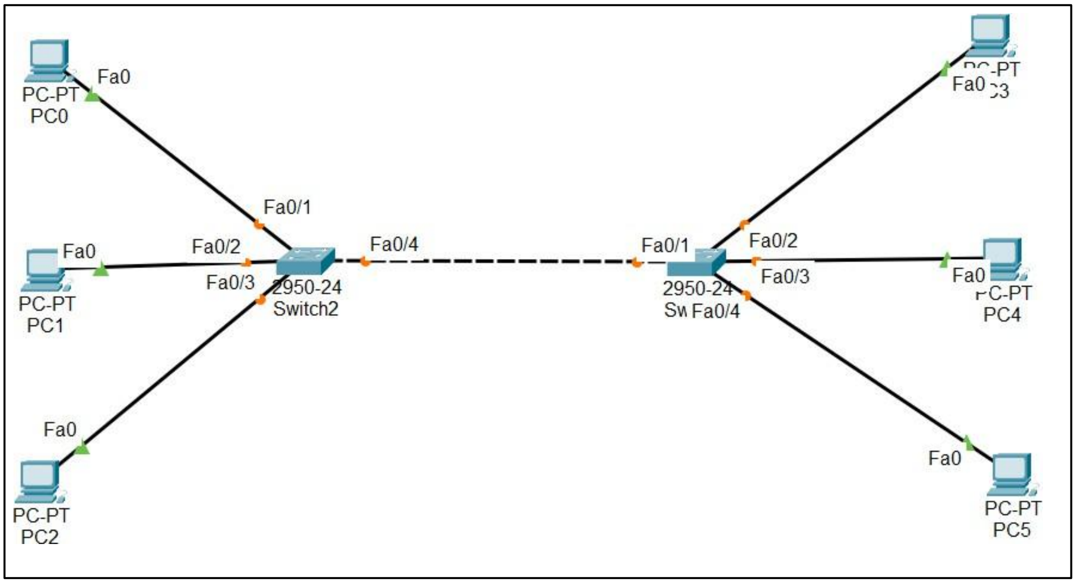
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	PC1	ICMP		0.000	N	0	(edit)	(delete)
	Successful	PC0	Router2	ICMP		0.000	N	1	(edit)	(delete)
	Successful	PC0	Router0	ICMP		0.000	N	2	(edit)	(delete)
	Successful	Router0	PC1	ICMP		0.000	N	3	(edit)	(delete)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

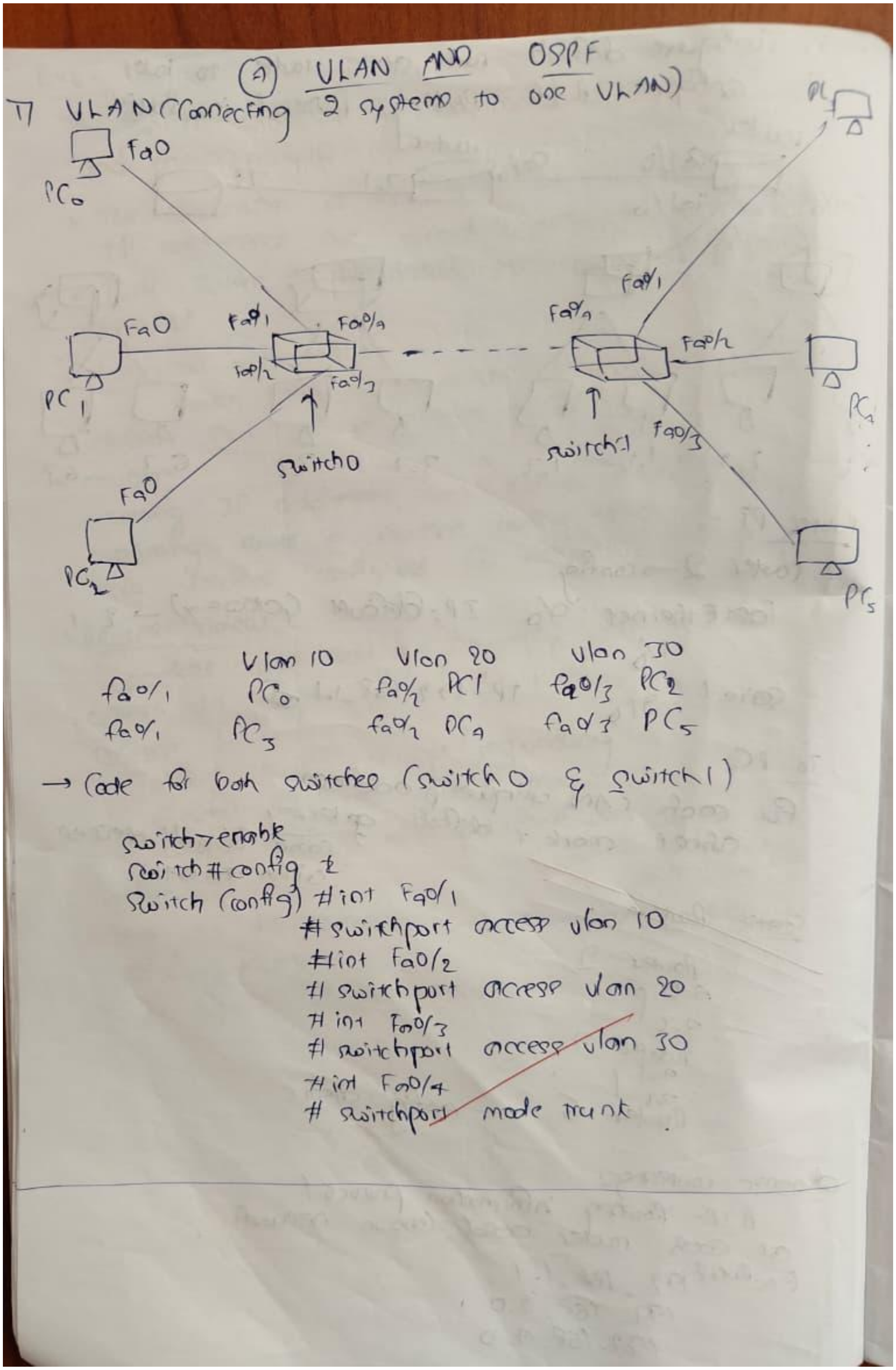
Program 8:

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

Network diagram:



Configuration:



(A) VLAN AND OSPF

1] VLAN (Connecting 2 systems to one VLAN)

	vlan 10	vlan 20	vlan 30
fa0/1	PC0	fa0/2 PC1	fa0/3 PC2
fa0/1	PC3	fa0/2 PC4	fa0/3 PC5

→ Code for both switches (switch0 & switch1)

```
switch>enable
switch#config t
Switch(config)#int Fa0/1
               #switchport access vlan 10
               #int fa0/2
               #switchport access vlan 20
               #int fa0/3
               #switchport access vlan 30
               #int fa0/4
               #switchport mode trunk
```

Output:

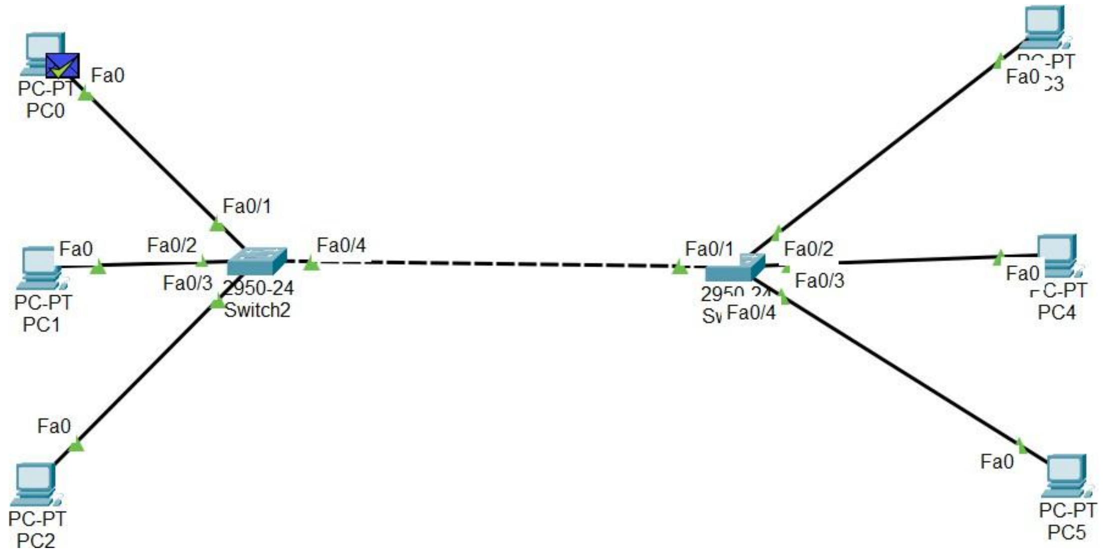
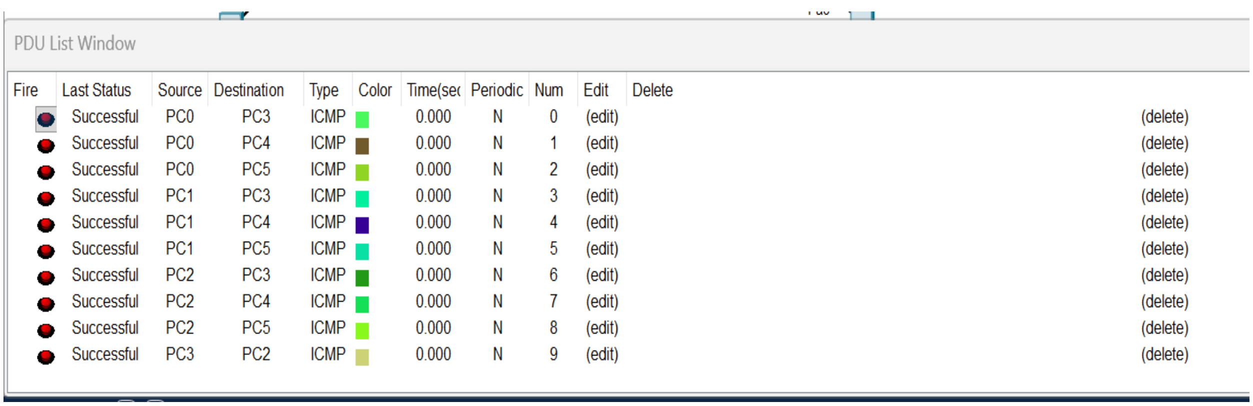


Fig 1. Sending PDU message from PC0 to PC5



PDU List Window

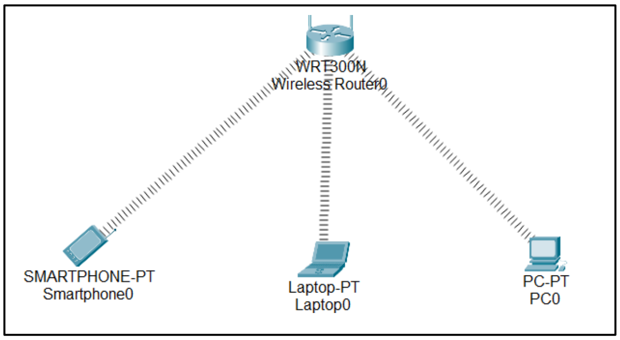
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	PC3	ICMP		0.000	N	0	(edit)	(delete)
	Successful	PC0	PC4	ICMP		0.000	N	1	(edit)	(delete)
	Successful	PC0	PC5	ICMP		0.000	N	2	(edit)	(delete)
	Successful	PC1	PC3	ICMP		0.000	N	3	(edit)	(delete)
	Successful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

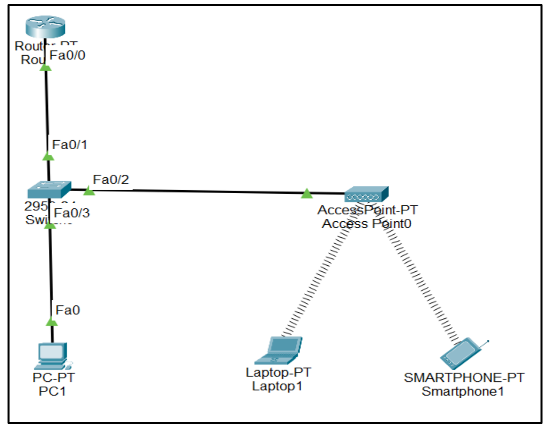
Program 9:

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

Network diagram:

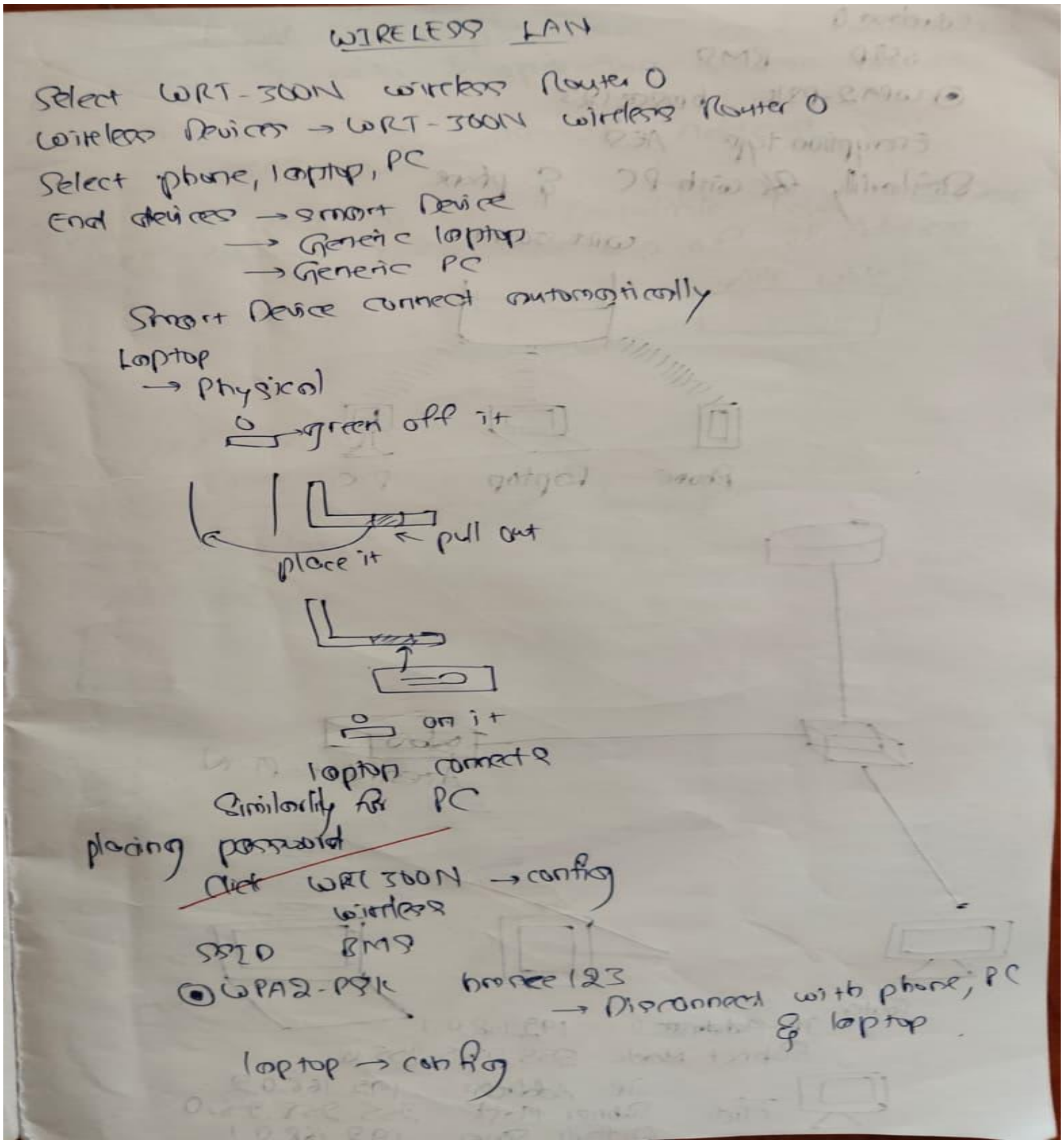


Configuration 1



Configuration 2

Configuration:



WIRELESS LAN

Select WRT-300N wireless Router 0

Wireless Devices → WRT-300N wireless Router 0

Select phone, laptop, PC

End devices → smart Device

→ Generic laptop

→ Generic PC

Smart Device connect automatically

Laptop

→ Physical

green off it

pull out

place it

on it

laptop connects

Similarly for PC

placing password

~~Click~~ WRT 300N → config

wireless

SSID BMS

WPA2-PSK bmsce123

→ Disconnect with phone, PC & laptop

laptop → config

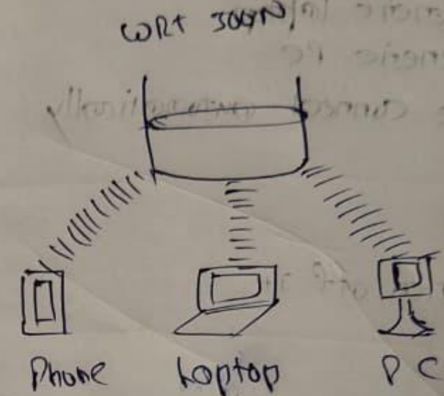
Wireless 0

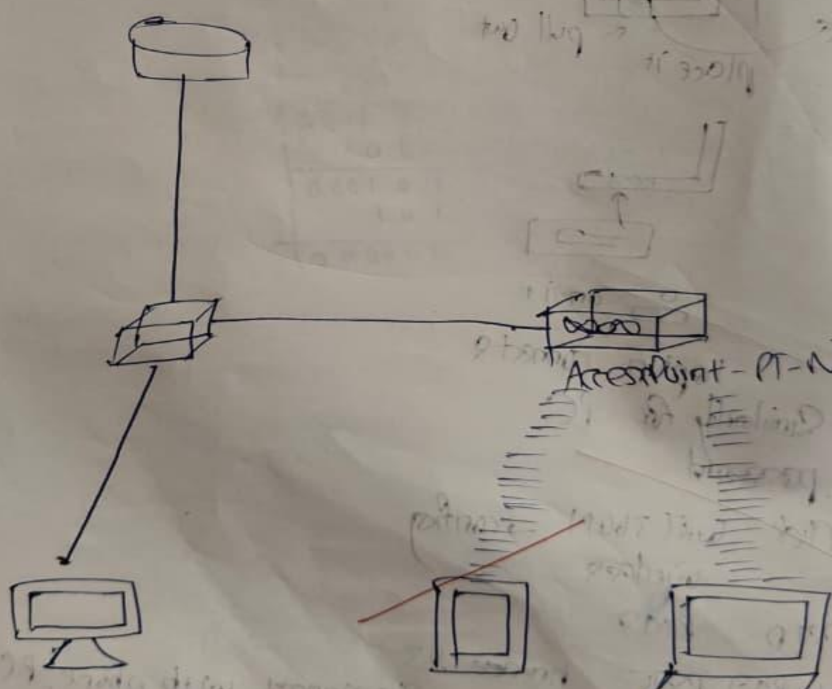
SSID BMS

(•) WPA2-PSK bmsce123

Encryption type AES

Similarlly ~~do~~ with PC & phone





Router IP Address 192.168.0.1

Subnet Mask 255.255.255.0

 Click

IP Address 192.168.0.2

Subnet Mask 255.255.255.0

Default Server 192.168.0.1

Output:

1. Do Physical Connections In:

- Laptop
- PC

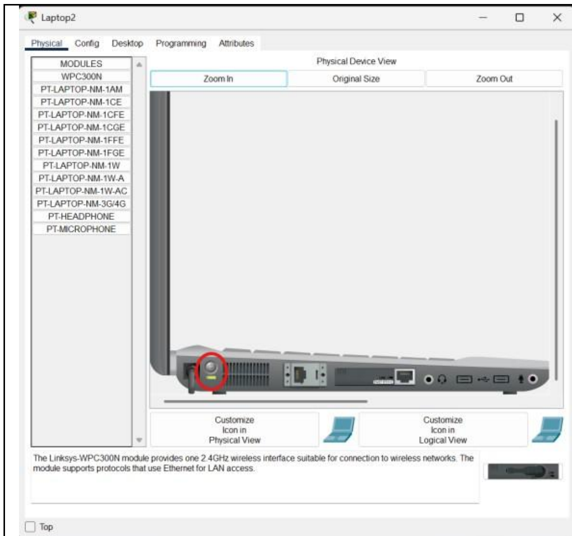


Fig 1.1 Step1: Turn off light / Power off laptop

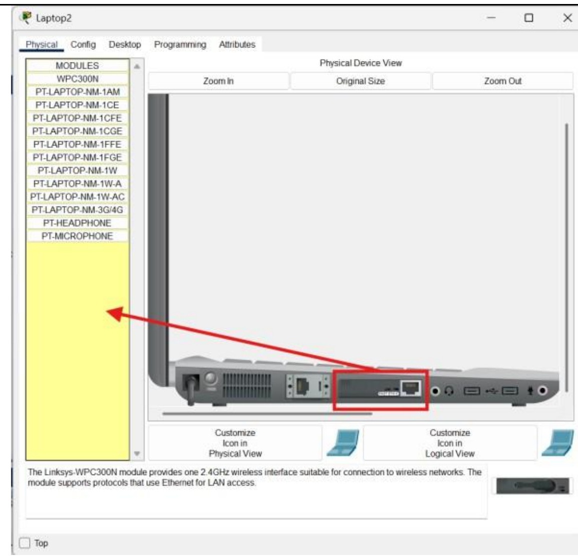


Fig 1.2 Step2: Drag and Drop the Ethernet into pointed location

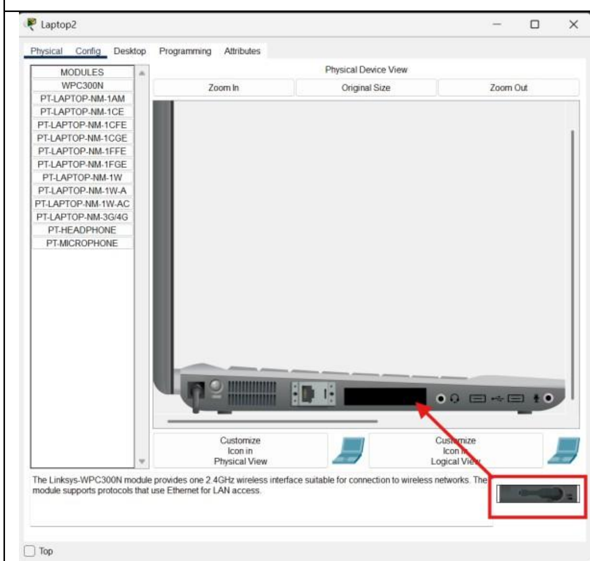


Fig 1.3 Step3: Drag and Drop the device into pointed location and Turn on light/Laptop

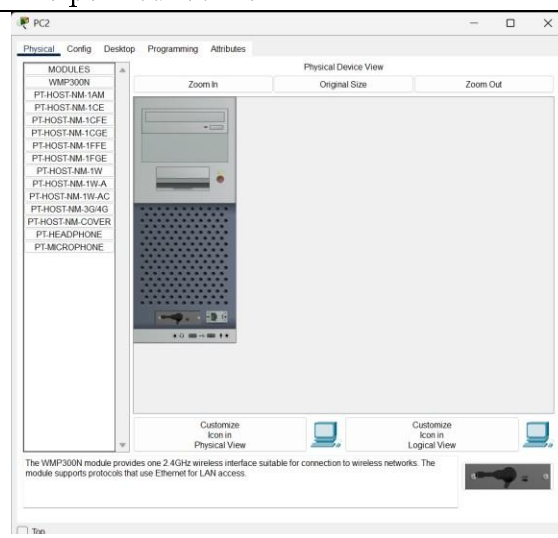


Fig 2. PC physical connection (combined 3 steps)

2. Do Wireless Connection in:

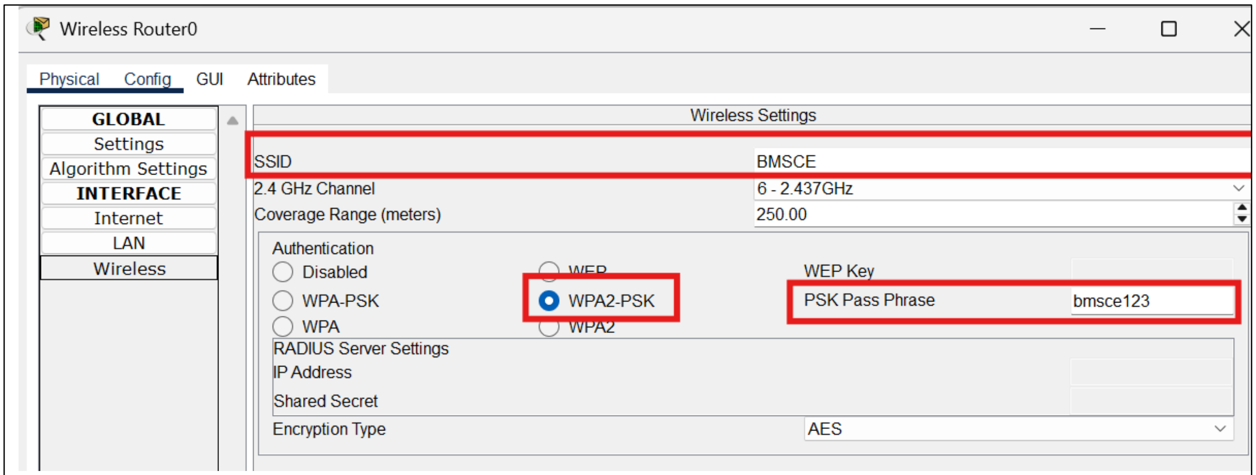


Fig 1. Config at Device Wireless Router0

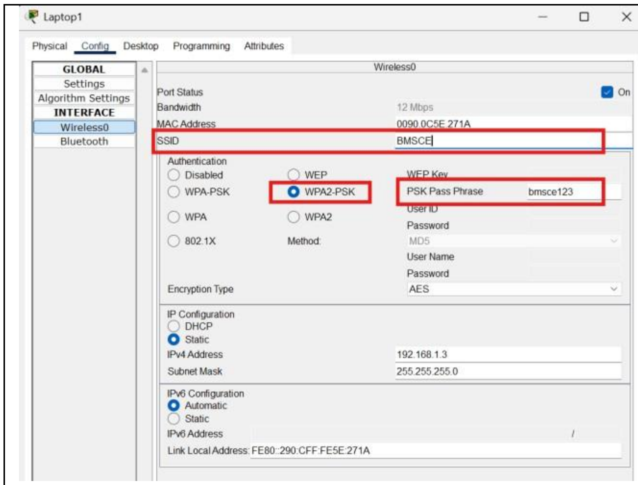


Fig 2. Config at Device Laptop0

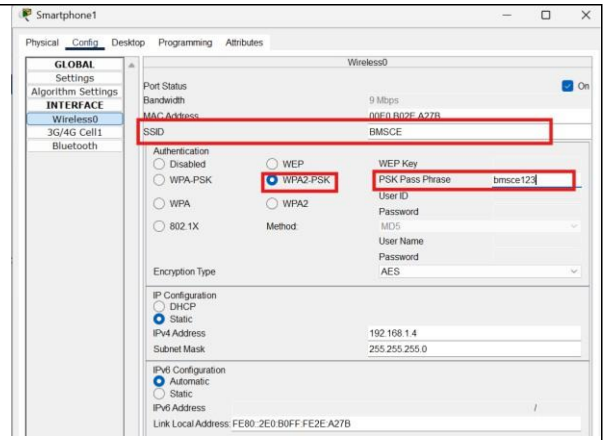
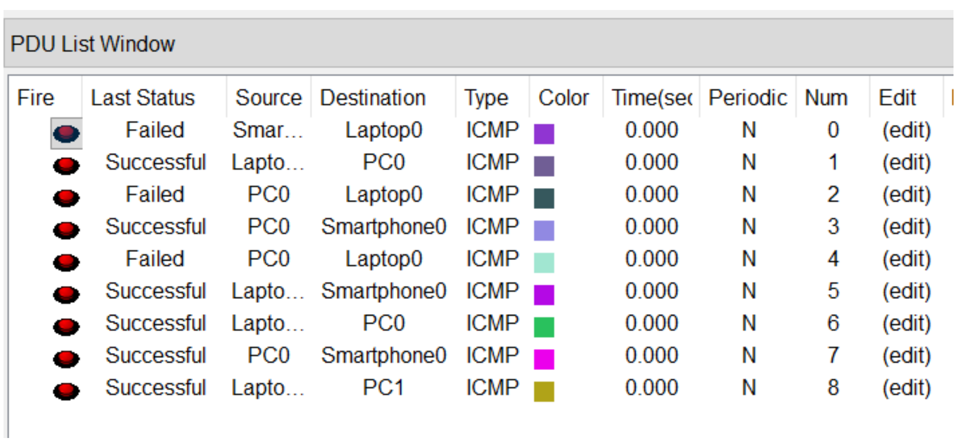


Fig 3. Config at Device Smartphone0



PDU List Window

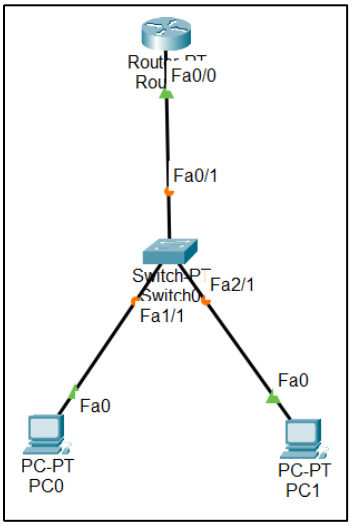
Fire	Last Status	Source	Destination	Type	Color	Time(sec	Periodic	Num	Edit
	Failed	Smar...	Laptop0	ICMP		0.000	N	0	(edit)
	Successful	Lapto...	PC0	ICMP		0.000	N	1	(edit)
	Failed	PC0	Laptop0	ICMP		0.000	N	2	(edit)
	Successful	PC0	Smartphone0	ICMP		0.000	N	3	(edit)
	Failed	PC0	Laptop0	ICMP		0.000	N	4	(edit)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

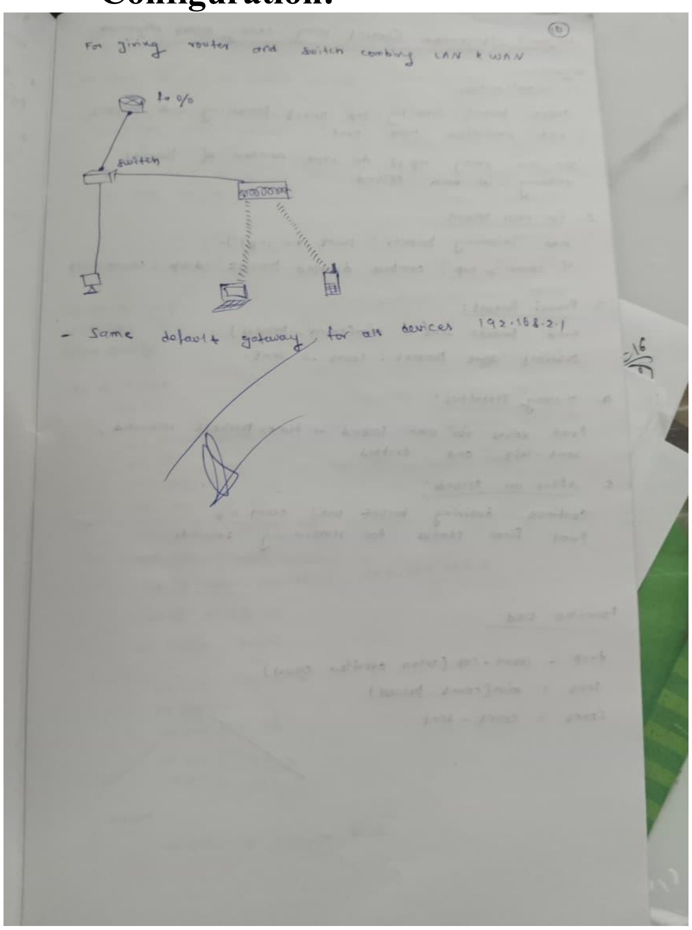
Program 10:

Aim: Demonstrate the TTL/ Life of a Packet.

Network diagram:



Configuration:



Output:

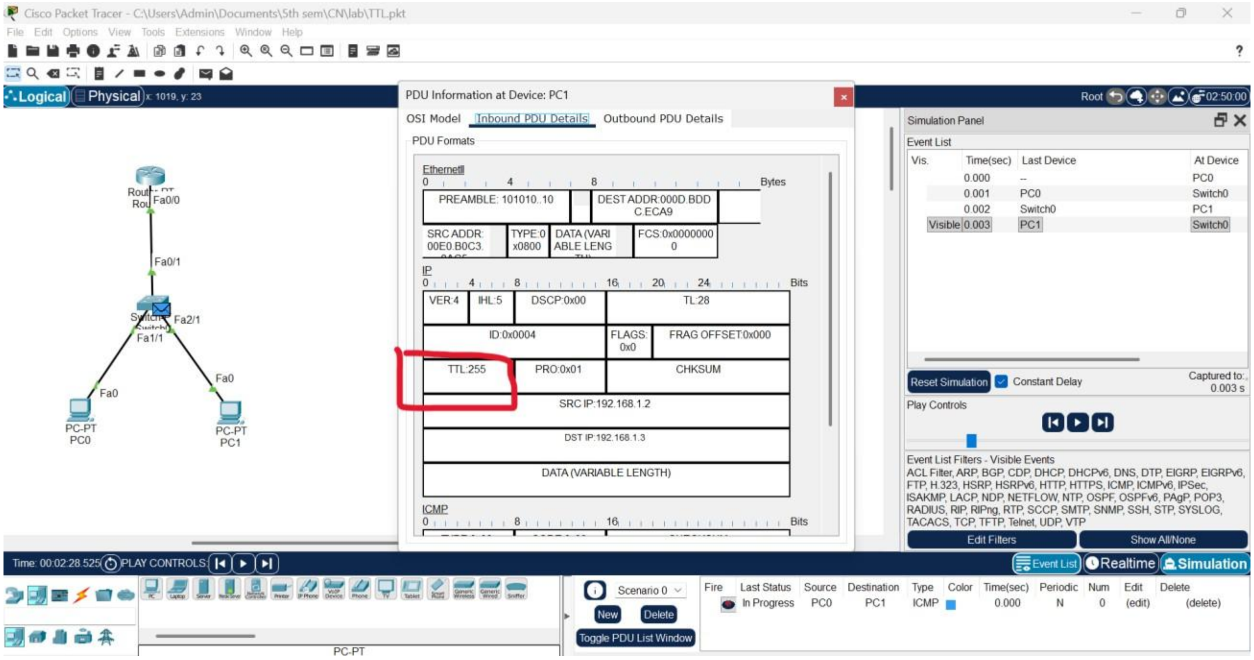


Fig 1. Inbound PDU Details at Device PC1

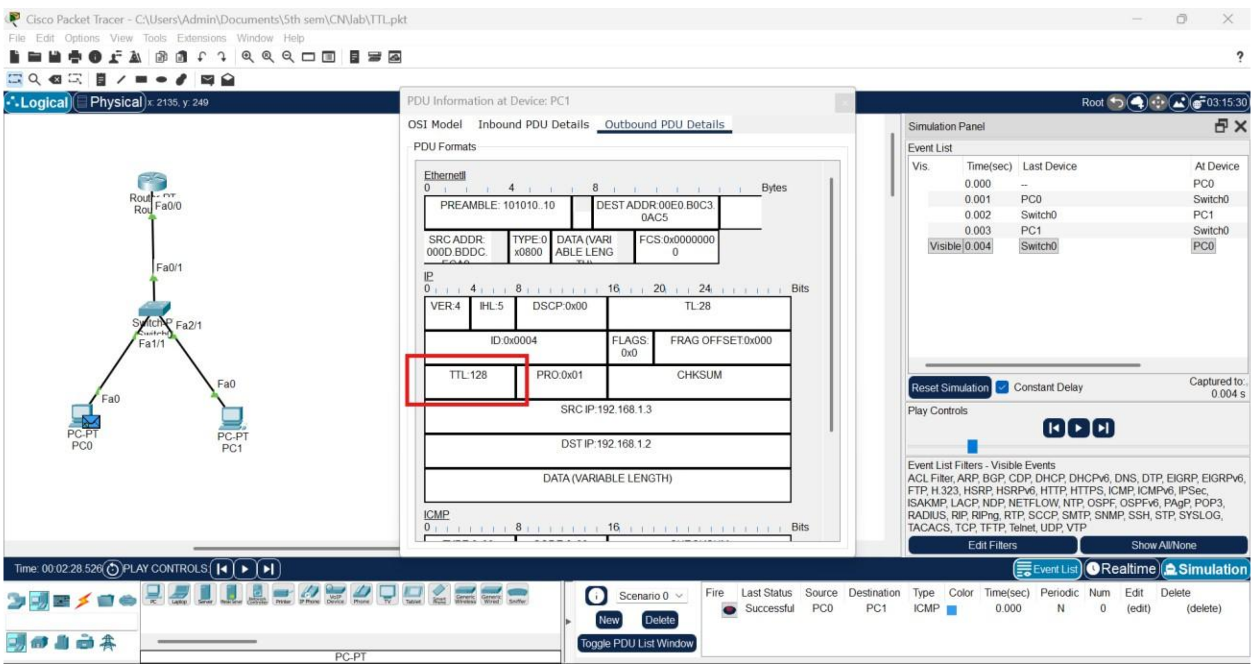
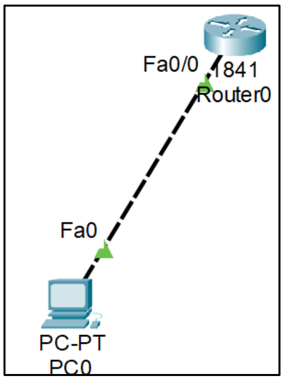


Fig 1. Outbound PDU Details at Device PC1

Program 11:

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

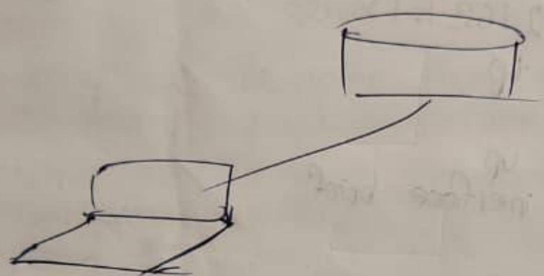
Network diagram:



Configuration:

* Construct a topology to demonstrate concept of Tellnet.

→ It is used to access remote server
→ It is simple command line tool that runs on computer & allow u to send commands remotely to a server.
→ Tellnet is also used to manage other devices like router, switch to check if ports are open or close on a server.



IP. 192.168.1.2
GATE 192.168.1.1

Router
no
enable
conf t
hostname R1
enable secret rp
int fa0/0
ip add 192.168.1.1 255.255.255.0
no shutdown
line vty 0 5
login
password rp
exit
exit

(press enter)

or
show ip interface brief

Go to PC0
go to command prompt (PC)
PC> check ping 192.168.1.1
execute the following
PC> telnet 192.168.1.1
PC> password ip
R1> en
password up
show ip interface brief

en
conf t
int Fa0/1 → unassigned
ip add 192.168.1.1 255.255.255.0

show ip interface brief
(check assigned).

Output:

Router0

Physical Config CLI Attributes

IOS Command Line Interface

```
Would you like to enter the initial configuration dialog? [yes/no]: no


Press RETURN to get started!



Router>enable
Router#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#hostname R1
R1(config)#enable secret rp
R1(config)#int fa 0/0
R1(config-if)#ip add 192.168.1.1 255.255.255.0
R1(config-if)#no shutdown

R1(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up

R1(config-if)#line vty 0 5
R1(config-line)#login
% Login disabled on line 194, until 'password' is set
% Login disabled on line 195, until 'password' is set
% Login disabled on line 196, until 'password' is set
% Login disabled on line 197, until 'password' is set
% Login disabled on line 198, until 'password' is set
% Login disabled on line 199, until 'password' is set
R1(config-line)#password tp
R1(config-line)#exit
R1(config)#exit
R1#
%SYS-5-CONFIG_I: Configured from console by console
wr
Building configuration...
[OK]
R1#show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        192.168.1.1     YES manual up                    up
FastEthernet0/1        unassigned      YES unset  administratively down down
Vlan1                  unassigned      YES unset  administratively down down
R1#
```

Fig 1. Router0 – CLI commands

PC0

Physical Config Desktop Programming Attributes

Command Prompt

```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.1

Pinging 192.168.1.1 with 32 bytes of data:

Reply from 192.168.1.1: bytes=32 time<1ms TTL=255
Reply from 192.168.1.1: bytes=32 time<1ms TTL=255
Reply from 192.168.1.1: bytes=32 time<1ms TTL=255
Reply from 192.168.1.1: bytes=32 time<1ms TTL=255

Ping statistics for 192.168.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>telnet 192.168.1.1
Trying 192.168.1.1 ...Open


User Access Verification

Password:
R1>enable
Password:
R1#show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        192.168.1.1     YES manual up                    up
FastEthernet0/1        unassigned      YES unset  administratively down down
Vlan1                  unassigned      YES unset  administratively down down
R1#enable
R1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#int fa 0/1
R1(config-if)#ip add 192.168.1.2 255.255.255.0
% 192.168.1.0 overlaps with FastEthernet0/0
R1(config-if)#
```

Fig2. PC command line prompt

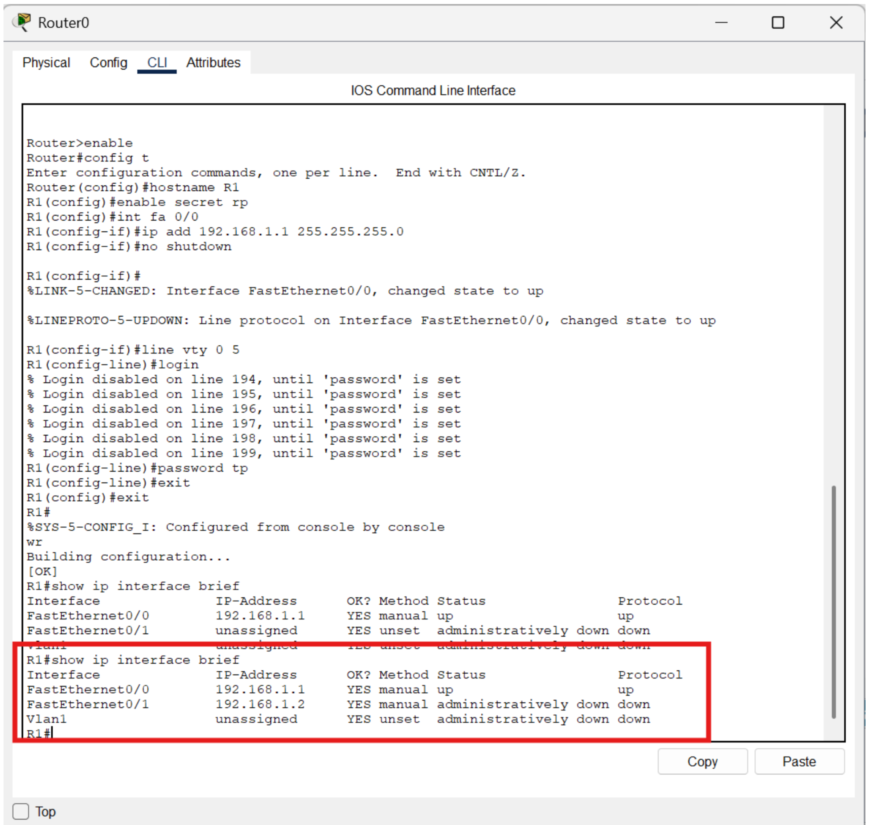


Fig 3. Updated the changes into Router0

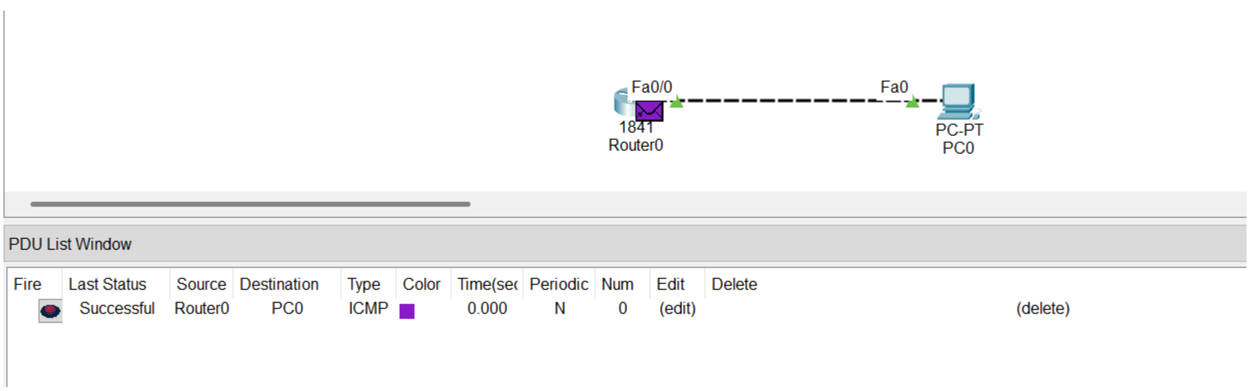
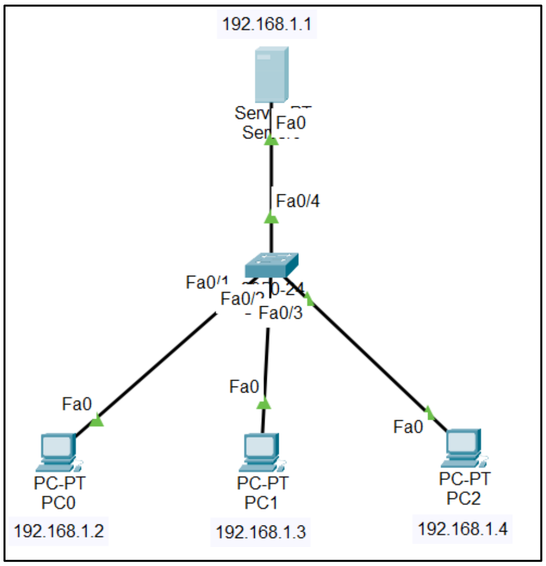


Fig 4. PDU message Successful

<u>Program 12:</u>

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

Network diagram:

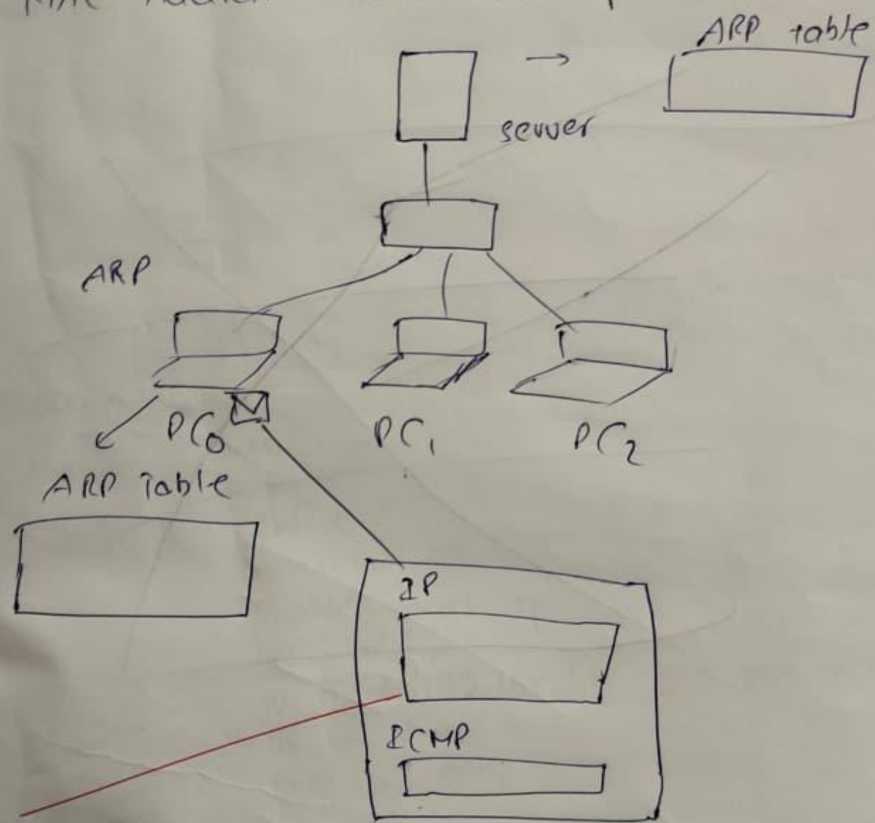


Configuration:

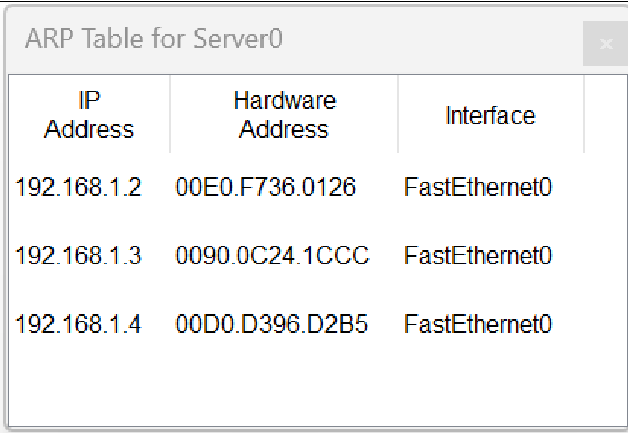
* To construct .. a simple LAN and understand the concept & also 2 operation address resolution protocol.

- ARP is used to map an IP Address to a MAC address.

- ARP is used to get Datalink layer Address, MAC Address with the help of IP Address.



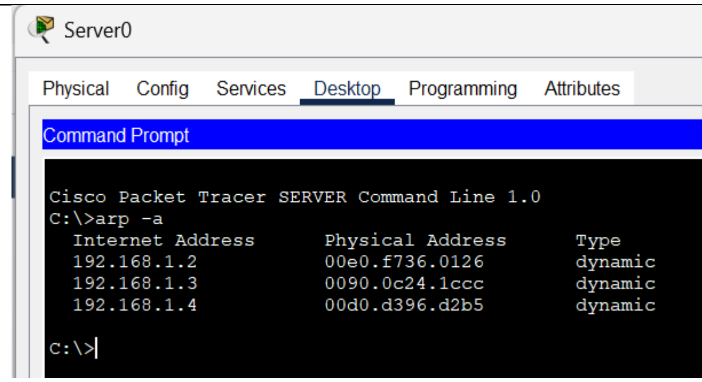
Output:



ARP Table for Server0

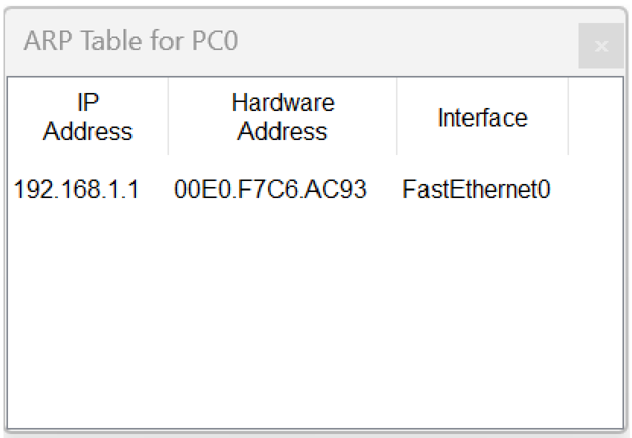
IP Address	Hardware Address	Interface
192.168.1.2	00E0.F736.0126	FastEthernet0
192.168.1.3	0090.0C24.1CCC	FastEthernet0
192.168.1.4	00D0.D396.D2B5	FastEthernet0

Fig 1.1 ARP table at Server0




```
Cisco Packet Tracer SERVER Command Line 1.0
C:\>arp -a
  Internet Address      Physical Address      Type
  192.168.1.2           00e0.f736.0126        dynamic
  192.168.1.3           0090.0c24.1ccc        dynamic
  192.168.1.4           00d0.d396.d2b5        dynamic

C:\>
```

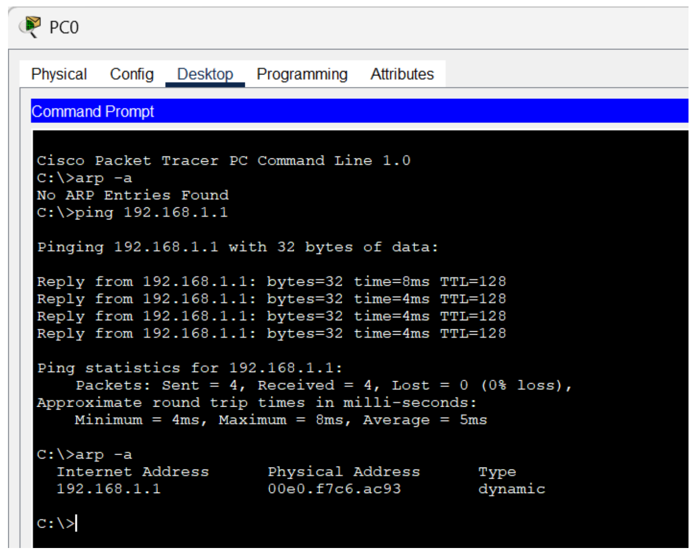
Fig 1.2 Command Prompt at Server0



ARP Table for PC0

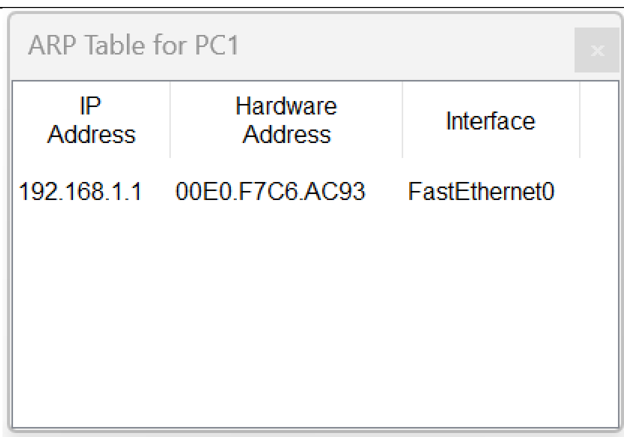
IP Address	Hardware Address	Interface
192.168.1.1	00E0.F7C6.AC93	FastEthernet0

Fig 2.1 ARP table at PC0




```
Cisco Packet Tracer PC Command Line 1.0
C:\>arp -a
No ARP Entries Found
C:\>ping 192.168.1.1

Pinging 192.168.1.1 with 32 bytes of data:

Reply from 192.168.1.1: bytes=32 time=8ms TTL=128
Reply from 192.168.1.1: bytes=32 time=4ms TTL=128
Reply from 192.168.1.1: bytes=32 time=4ms TTL=128
Reply from 192.168.1.1: bytes=32 time=4ms TTL=128

Ping statistics for 192.168.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 8ms, Average = 5ms

C:\>arp -a
  Internet Address      Physical Address      Type
  192.168.1.1           00e0.f7c6.ac93        dynamic

C:\>
```

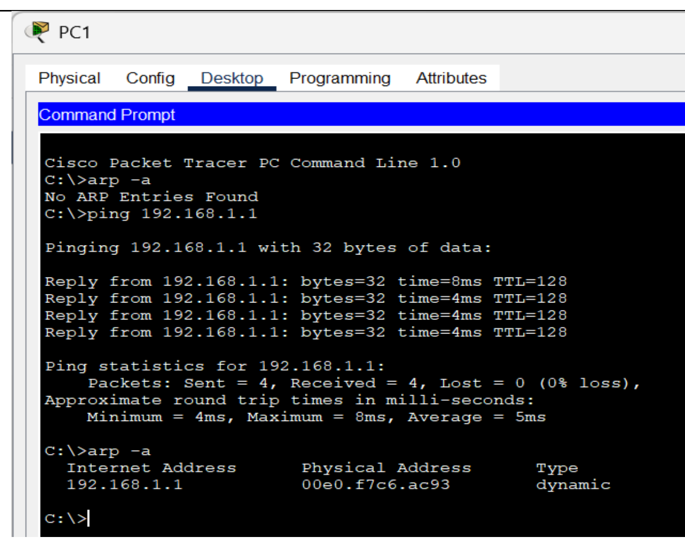
Fig 2.2 Command Prompt at PC0



ARP Table for PC1

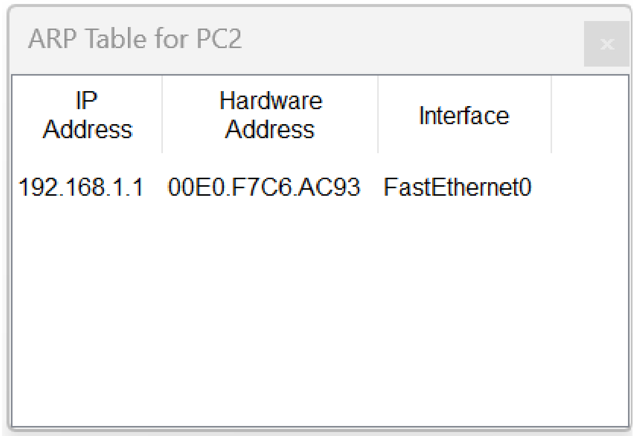
IP Address	Hardware Address	Interface
192.168.1.1	00E0.F7C6.AC93	FastEthernet0

Fig 3.1 ARP table at PC1




```
Cisco Packet Tracer PC Command Line 1.0
C:\>arp -a
No ARP Entries Found
C:\>ping 192.168.1.1

Pinging 192.168.1.1 with 32 bytes of data:

Reply from 192.168.1.1: bytes=32 time=8ms TTL=128
Reply from 192.168.1.1: bytes=32 time=4ms TTL=128
Reply from 192.168.1.1: bytes=32 time=4ms TTL=128
Reply from 192.168.1.1: bytes=32 time=4ms TTL=128

Ping statistics for 192.168.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 8ms, Average = 5ms

C:\>arp -a
  Internet Address      Physical Address      Type
  192.168.1.1           00e0.f7c6.ac93        dynamic

C:\>
```

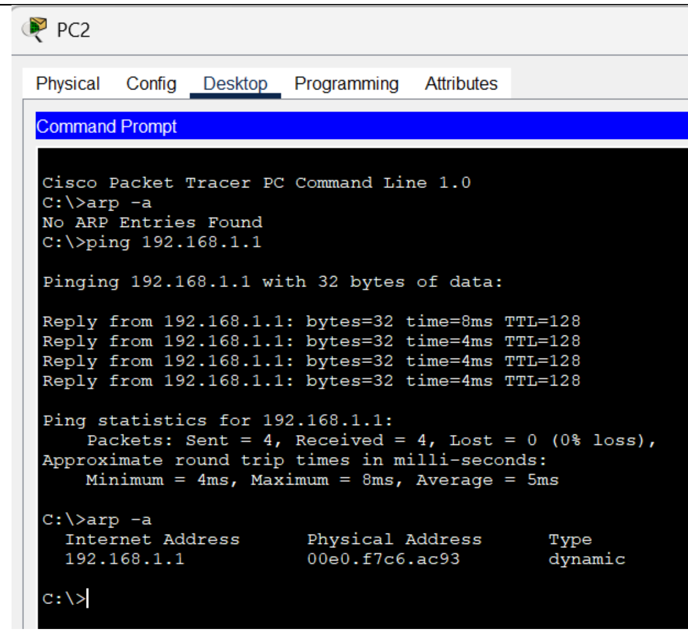
Fig 3.2 Command Prompt at PC1



ARP Table for PC2

IP Address	Hardware Address	Interface
192.168.1.1	00E0.F7C6.AC93	FastEthernet0

Fig 4.1 ARP table at PC2




```
Cisco Packet Tracer PC Command Line 1.0
C:\>arp -a
No ARP Entries Found
C:\>ping 192.168.1.1

Pinging 192.168.1.1 with 32 bytes of data:

Reply from 192.168.1.1: bytes=32 time=8ms TTL=128
Reply from 192.168.1.1: bytes=32 time=4ms TTL=128
Reply from 192.168.1.1: bytes=32 time=4ms TTL=128
Reply from 192.168.1.1: bytes=32 time=4ms TTL=128

Ping statistics for 192.168.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 8ms, Average = 5ms

C:\>arp -a
  Internet Address      Physical Address      Type
  192.168.1.1           00e0.f7c6.ac93        dynamic

C:\>
```

Fig 4.2 Command Prompt at PC2

PART - B

Program 1:

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

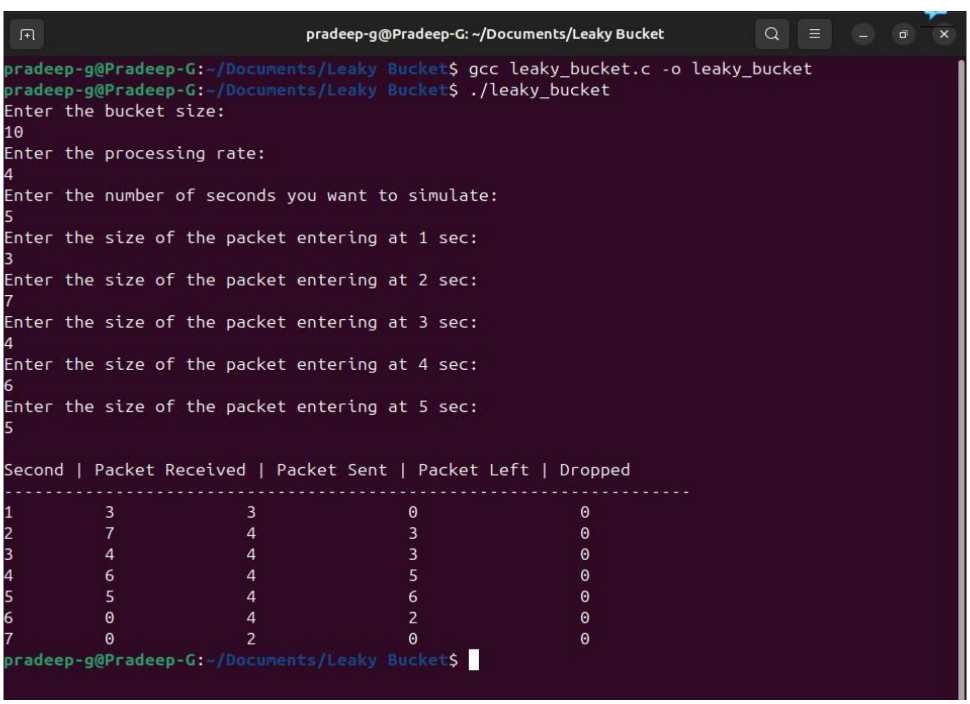
Code:

```
#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:

Leaky Bucket Algorithm

i) Packet arrives at varying rates

ii) The bucket has

Capacity(c) → maximum of packets it can hold

leak rate (r) → no of packet sent per unit time

iii) At each time steps:

1) Add incoming packets to bucket

2) If bucket > → drop packets (overflow)

3) leak (send) packets at constant rate ✓.

iv) Repeat .

Program 2:

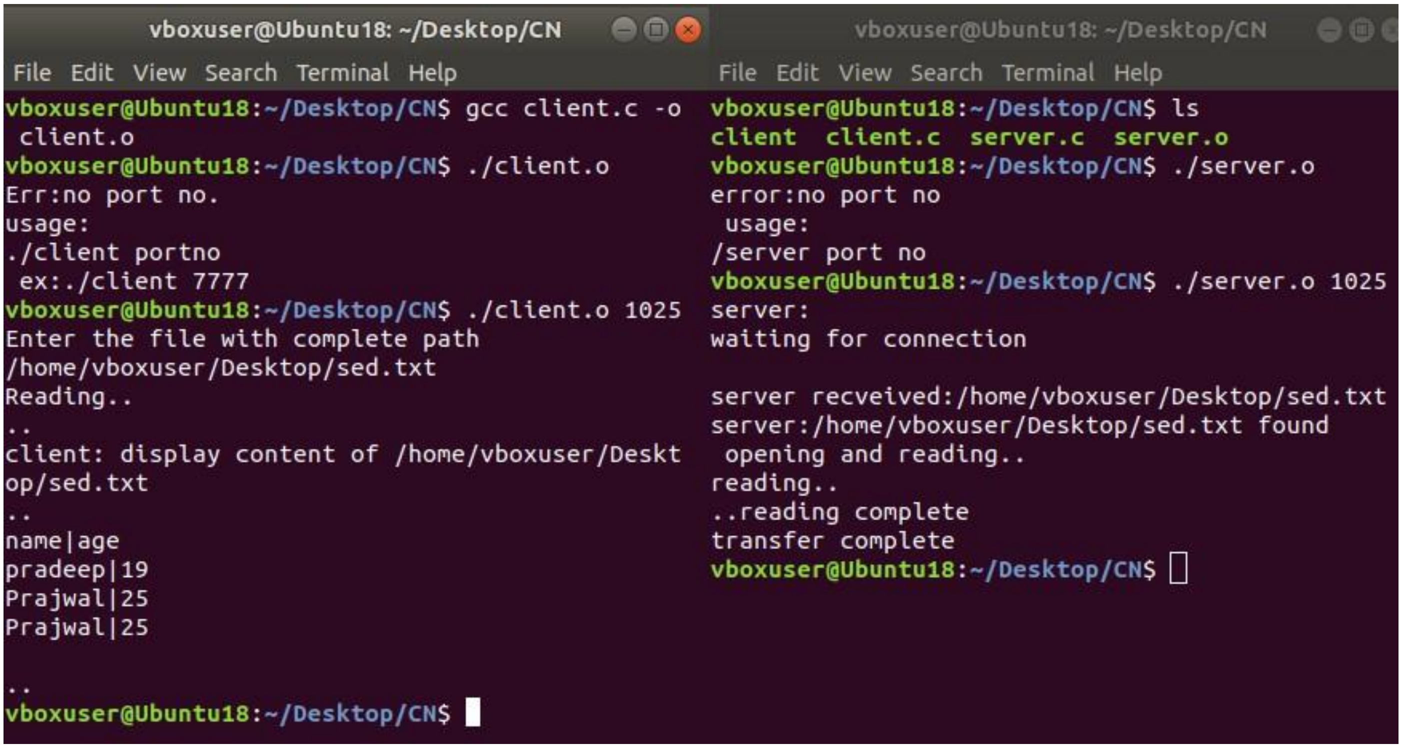
Aim: Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

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:



```
vboxuser@Ubuntu18: ~/Desktop/CN
File Edit View Search Terminal Help
vboxuser@Ubuntu18:~/Desktop/CN$ gcc client.c -o
 client.o
vboxuser@Ubuntu18:~/Desktop/CN$ ./client.o
Err:no port no.
usage:
./client portno
 ex:./client 7777
vboxuser@Ubuntu18:~/Desktop/CN$ ./client.o 1025
Enter the file with complete path
/home/vboxuser/Desktop/sed.txt
Reading..
..
client: display content of /home/vboxuser/Deskt
op/sed.txt
..
name|age
pradeep|19
Prajwal|25
Prajwal|25

..
vboxuser@Ubuntu18:~/Desktop/CN$
```

```
vboxuser@Ubuntu18: ~/Desktop/CN
File Edit View Search Terminal Help
vboxuser@Ubuntu18:~/Desktop/CN$ ls
client  client.c  server.c  server.o
vboxuser@Ubuntu18:~/Desktop/CN$ ./server.o
error:no port no
 usage:
/server port no
vboxuser@Ubuntu18:~/Desktop/CN$ ./server.o 1025
server:
waiting for connection

server recveived:/home/vboxuser/Desktop/sed.txt
server:/home/vboxuser/Desktop/sed.txt found
 opening and reading..
reading..
..reading complete
transfer complete
vboxuser@Ubuntu18:~/Desktop/CN$
```

Observation:

Client Server Communication using TCP/IP protocol

TCP

Algorithm (Client side)

(i) sfd = create a socket with socket (- -) system call

(ii) connect the socket to the address of the server using the connect (sfd, -) system call
The IP address of the server machine and port number of the server service need to be provided

iii) Read the file name from standard input by
n = read (stdio, buffer, sizeof (buffer))

iv) Write the filename to socket using
write (sfd, buffer, n)

v) Read file contents from socket by
m = read (sfd, buffer, sizeof (buffer)

vi) Display file contents to standard output by
write (stdio, buffer, m)

vii) Go to step 5 if m>0

viii) Close socket by close (sfd)

Algorithm (Server side)

* sfd = Create a socket with the socket (- -) system call

* Blind the socket to an address using bind (sfd - -) system call. Assign a port number below 5000 to 5000 to sfn-ports

* Listen for connection with the listen(sfd - -) system call

* sfd = Accept a connection with the accept system call
This call typically blocks until a client connects with server

* Read the file name from socket by n = read (sfd, buffer, sizeof (buffer))

* Open the file by fd = open (buffer)

* Read the connects of file by m = read (fd, buffer, sizeof (buffer))

* Go to step 3

* Close (sfd)

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.

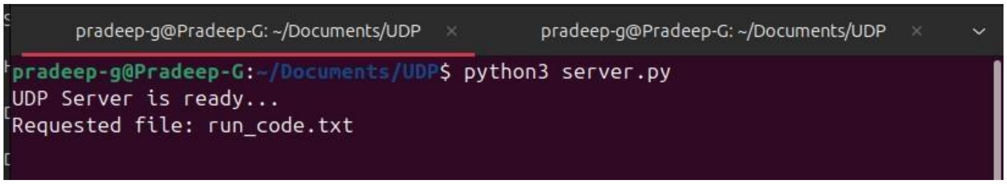
Code:

```python
# 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.encode(), 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()
```

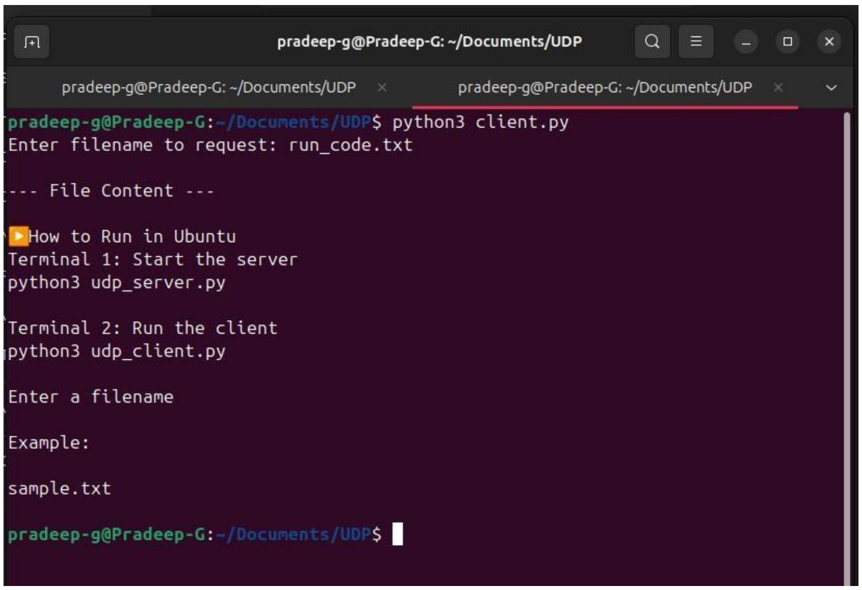
```python
# udp_server.py

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"File not found on server.", addr)
```

Output:

Server side Terminal:



Client side Terminal:



Observation:

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++)
```

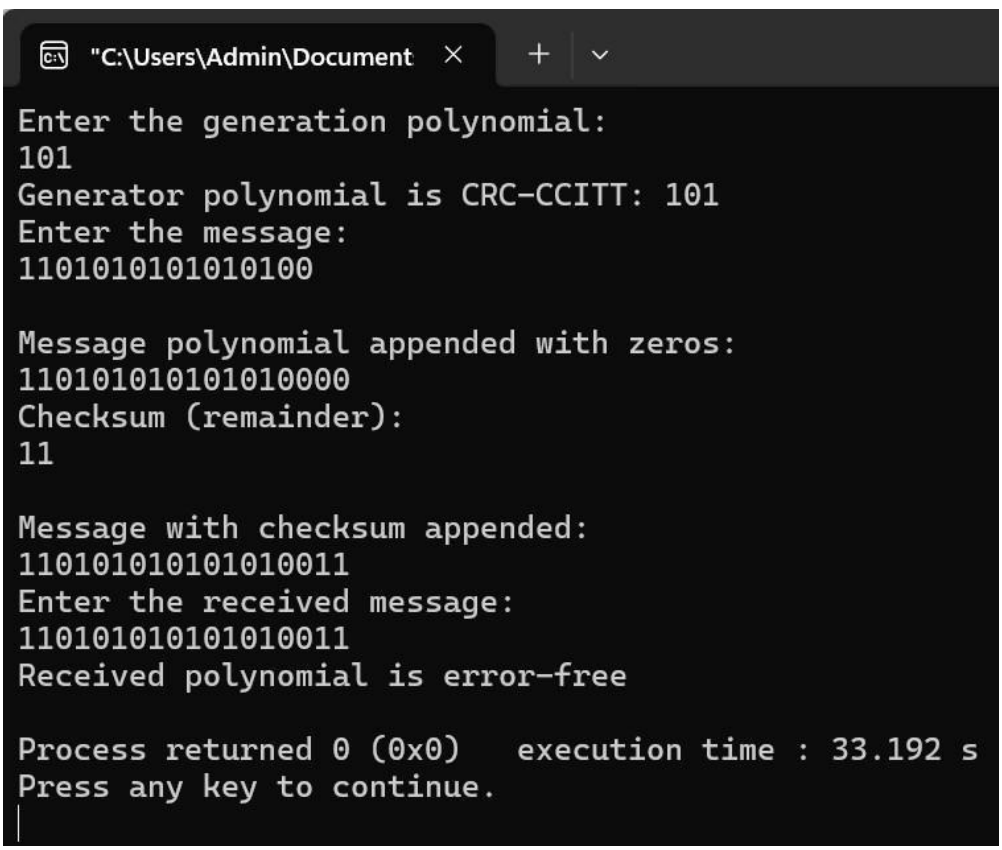
```
a[i] = msj[i];
```

```
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:



```
"C:\Users\Admin\Document
Enter the generation polynomial:
101
Generator polynomial is CRC-CCITT: 101
Enter the message:
1101010101010100

Message polynomial appended with zeros:
110101010101010000
Checksum (remainder):
11

Message with checksum appended:
110101010101010011
Enter the received message:
110101010101010011
Received polynomial is error-free

Process returned 0 (0x0)   execution time : 33.192 s
Press any key to continue.
```

Observation:

Algorithm for CRC

* Encoding Procedure At sender

Steps

Multiply $f(x)$ by $x^{(n-k)}$ by putting zeros in n-k low order position.

Divide $x^{(n-k)} f(x)$ by $g(x)$ to get $r(x)$. Use Euclidian Algorithm with a feedback shift register as shown in above fig.

$x^{(n-k)} f(x) = g(x) q(x) + r(x)$ where $q(x)$ is quotient

$r(x)$ is remainder.

Add remainder $r(x)$ to $x^{(n-k)} f(x)$ by putting check bits in n-k lower order position

Based on randomness, the message can be transmitted with (or) without error

For transmission with error, introduce an error at random position to the message $x^{(n-k)} f(x)$ and display position of error

Transmitted codeword is $b(x) = x^{(n-k)} f(x) + r(x)$

* Decoding Procedure at the Receiver.

Steps

The received message $b(x)$ is divided by $g(x)$ using Euclidian division algo

If remainder is other there is no error in transmission else error in transmission.