VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

NAVNEET KUMAR (1BM23CS207)

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 **Navneet Kumar (1BM23CS207),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements of the above-mentioned subject and the work prescribed for the said degree.

Sarala D V Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index

Part - A

Part - B

GitHub Link:
https://github.com/navneet207/1BM23CS207_CN

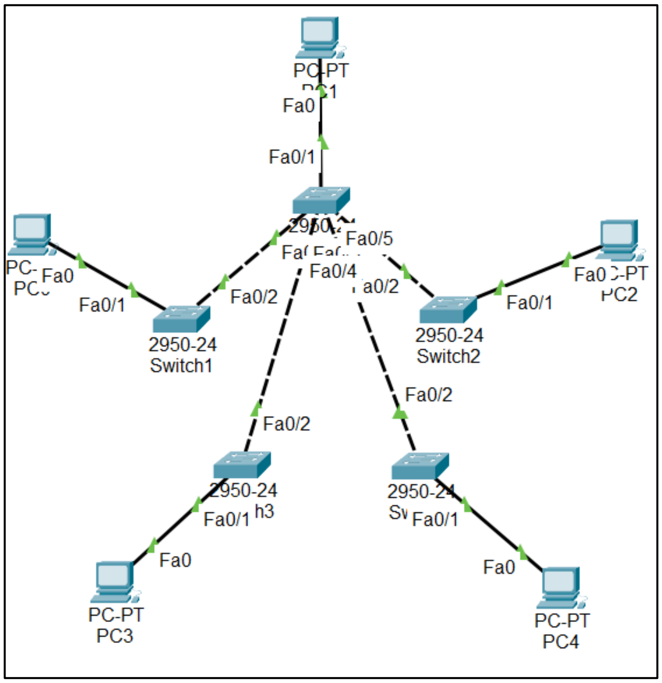
PART - A

Program 1:

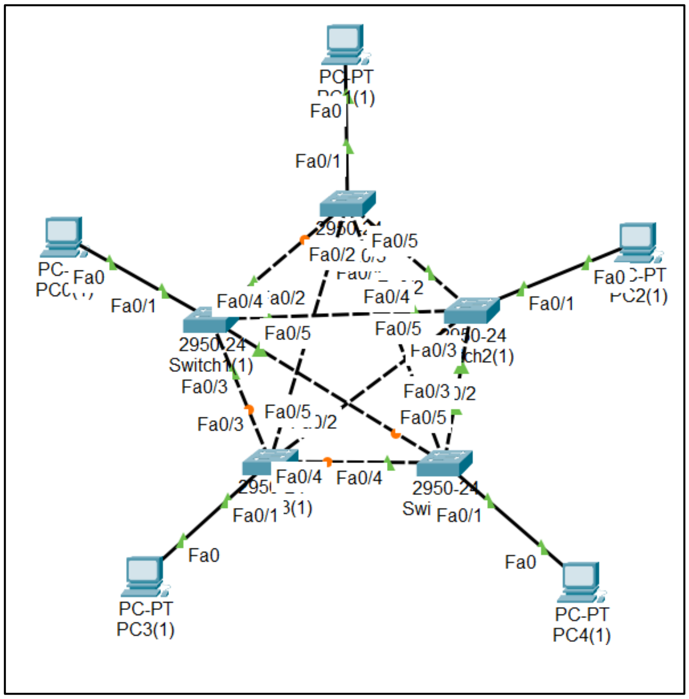
Aim: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

Network diagram:

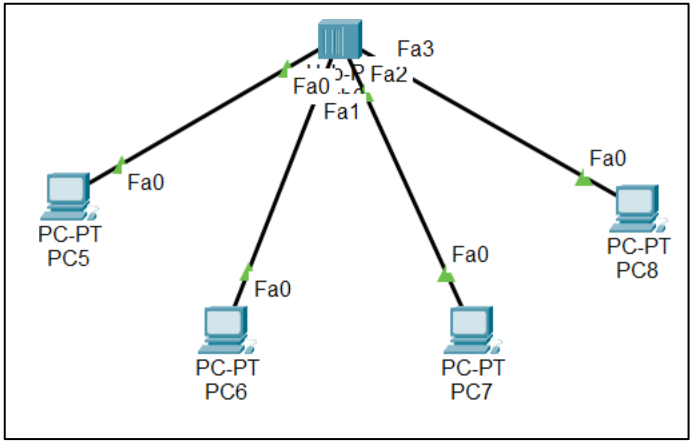
1. STAR Topology with Switch:



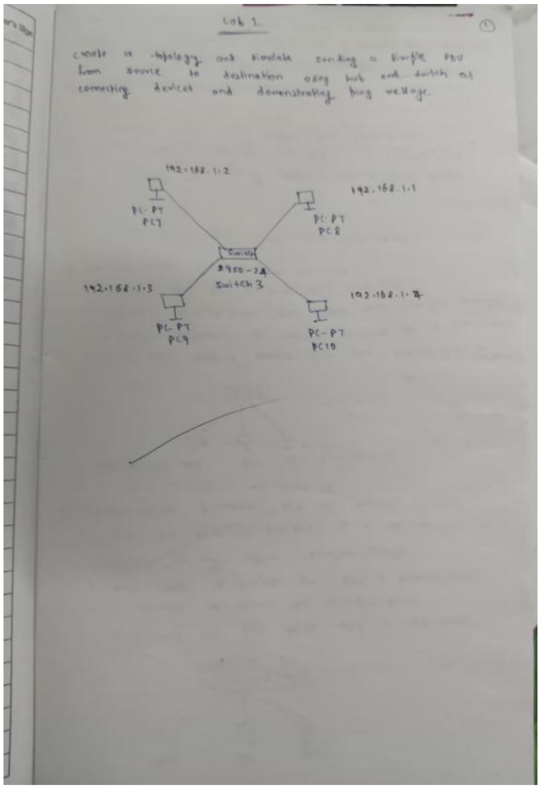
2. MESH Topology with Switch:



3. HUB-Based Network Topology:



Configuration:



Lab 1

①

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

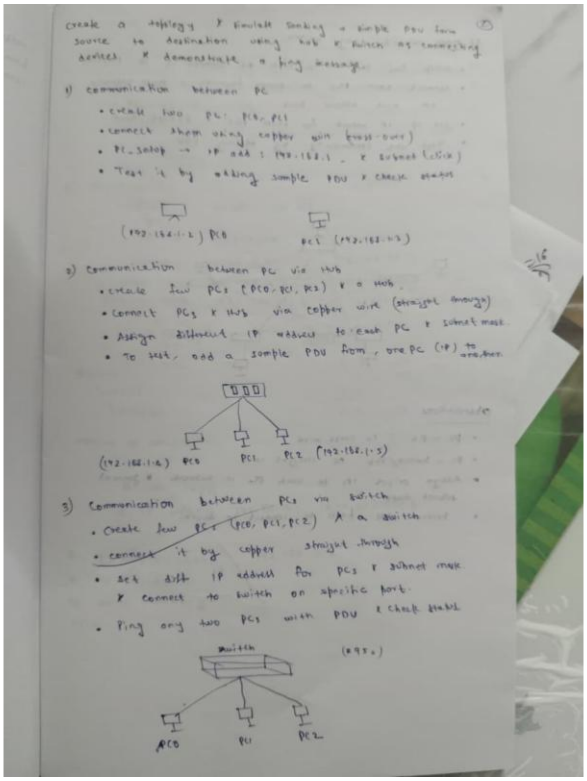
192.168.1.2
PC-PT
PC7

192.168.1.1
PC-PT
PC8

Switch
2950-24
Switch 3

192.168.1.3
PC-PT
PC9

192.168.1.4
PC-PT
PC10



create a topology & simulate sending a simple PDU form source to destination using hub & switch as connecting devices & demonstrate a ping message. ②

1) communication between PC

- create two PC: PC0, PC1
- connect them using copper wire (cross-over)
- PC_setup → IP add : 192.168.1._ & subnet (click)
- Test it by adding sample PDU & check status

(192.168.1.2) PC0

PC1 (192.168.1.3)

2) Communication between PC via Hub

- create few PCs (PC0, PC1, PC2) & a Hub.
- connect PCs & Hub via copper wire (straight through)
- Assign different IP address to each PC & subnet mask.
- To test, add a sample PDU from, one PC (IP) to another.

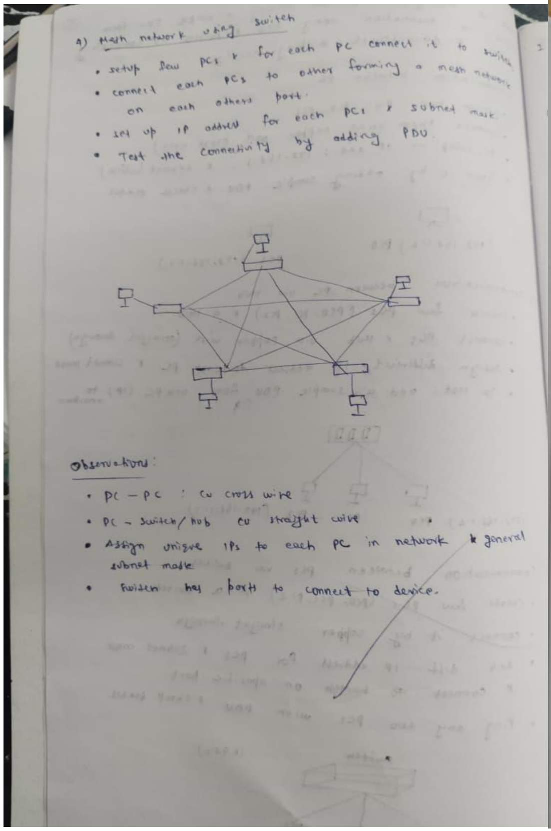
(192.168.1.4) PC0 PC1 PC2 (192.168.1.5)

3) Communication between PCs via switch

- Create few PCs (PC0, PC1, PC2) & a switch
- connect it by copper straight through
- set diff IP address for PCs & subnet mask & connect to switch on specific port.
- Ping any two PCs with PDU & check status

switch (2950)

PC0 PC1 PC2

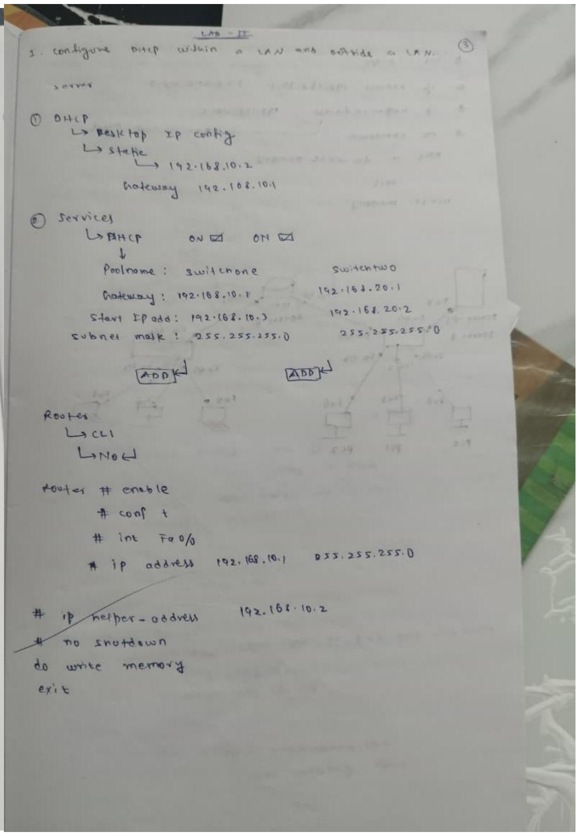


4) Mesh network using switch

- setup few PCs & for each PC connect it to switch
- connect each PCs to other forming a mesh network on each others port.
- set up IP address for each PCs & subnet mask
- Test the connectivity by adding PDU

Observations:

- PC – PC : cu cross wire
- PC – Switch/hub cu straight wire
- Assign unique IPs to each PC in network & general subnet mask
- Switch has ports to connect to device.



LAB – II

1. configure DHCP within a LAN and outside a LAN. ③

server

① DHCP
  ↳ Desktop IP config
   ↳ static
     ↳ 192.168.10.2
   Gateway 192.168.10.1

② Services
  ↳ DHCP ON ☑ ON ☑

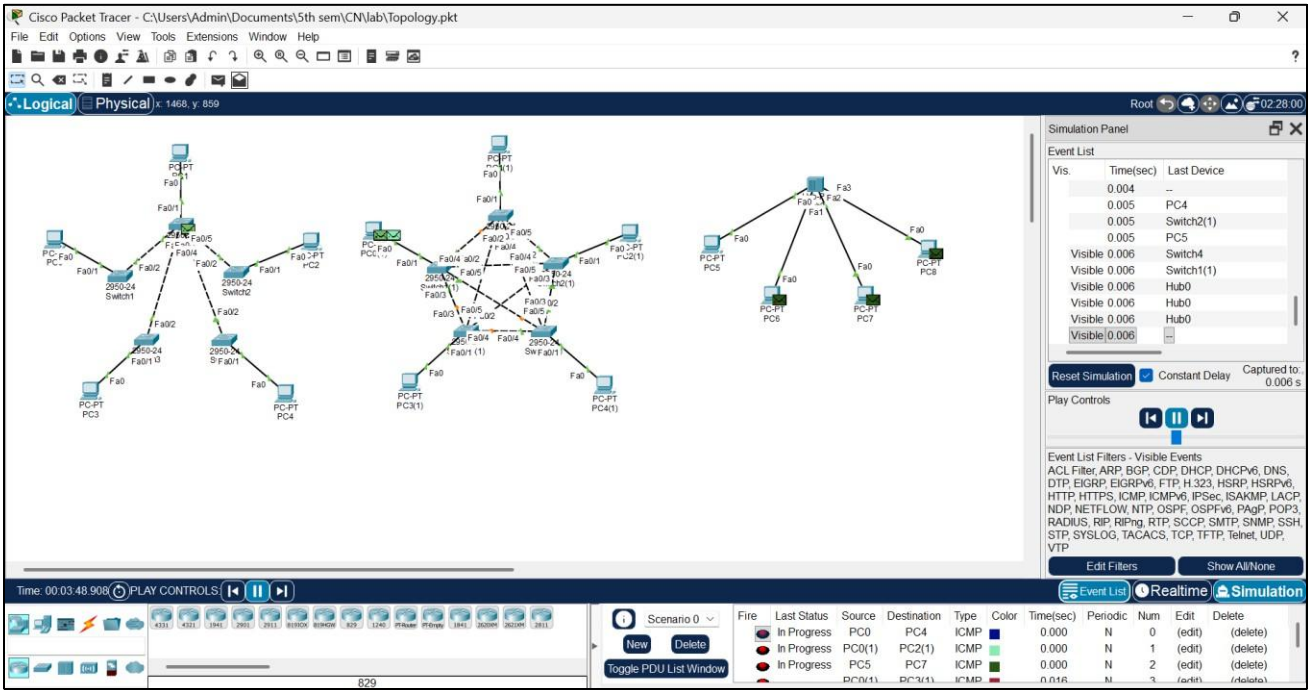
Poolname :	switchone	switchtwo
Gateway :	192.168.10.1	192.168.20.1
Start IP add :	192.168.10.3	192.168.20.2
subnet mask :	255.255.255.0	255.255.255.0

ADD ↵ ADD ↵

Router
 ↳ CLI
  ↳ No ↵

```
Router # enable
       # conf t
       # int Fa0/0
       # ip address 192.168.10.1 255.255.255.0

# ip helper-address 192.168.10.2
# no shutdown
do write memory
exit
```

Output:



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

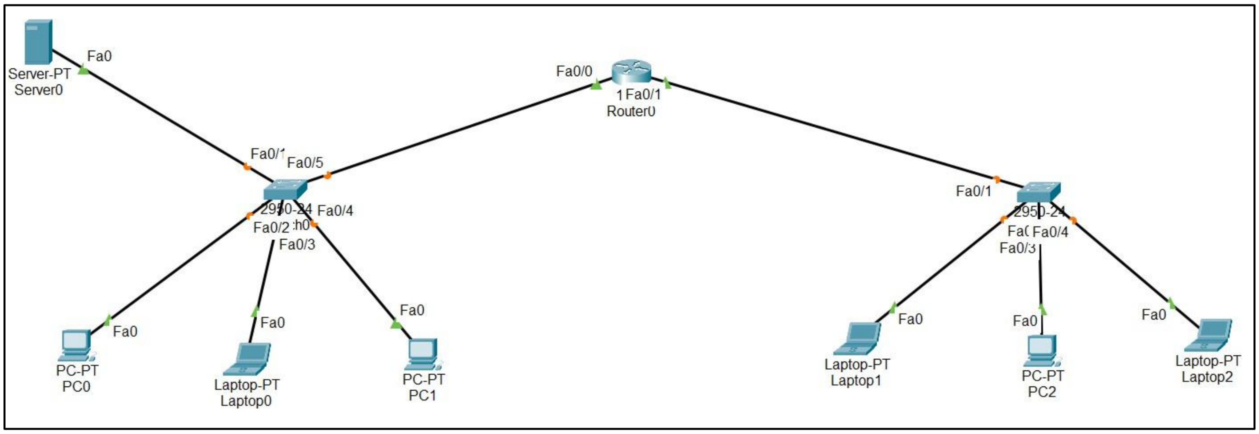
Event List Filters - Visible Events
ACL Filter, ARP, BGP, CDP, DHCP, DHCPv6, DNS, DTP, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, RADIUS, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP

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)
●		PC0(1)	PC3(1)	ICMP	🟥	0.016	N	3	(edit)	(delete)

Program 2:

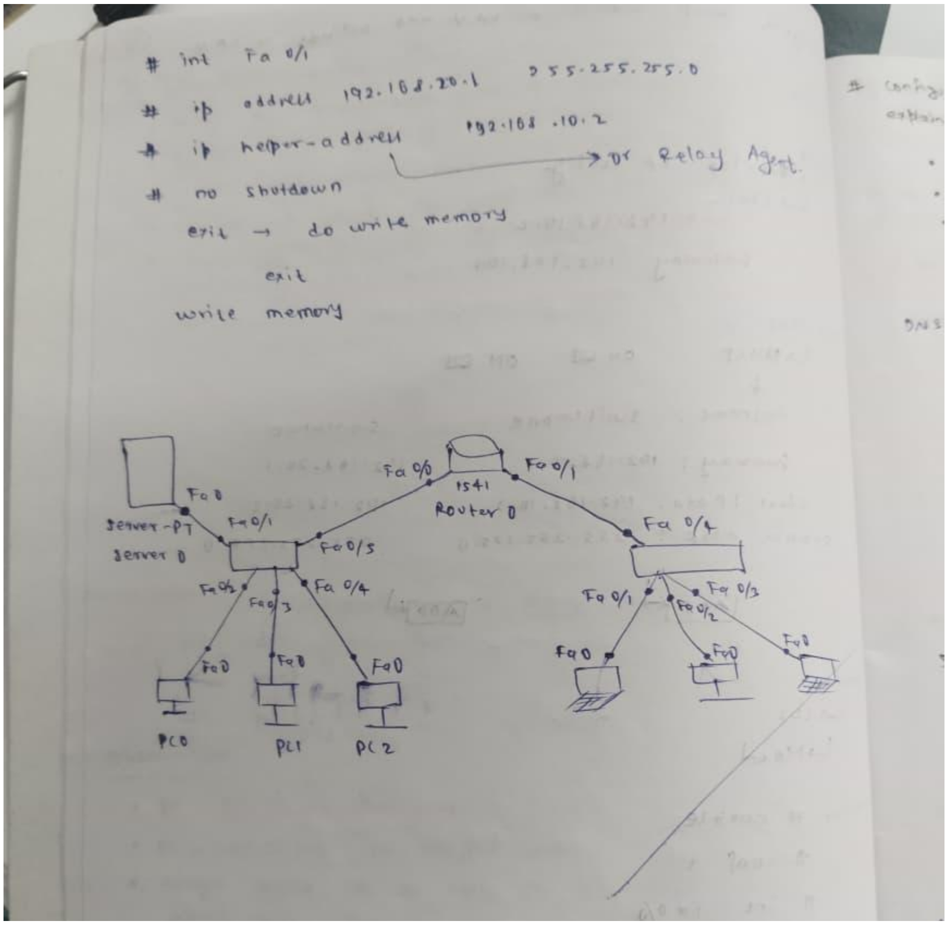
Aim: Configure DHCP within a LAN and outside LAN.

Network diagram:



Configuration:

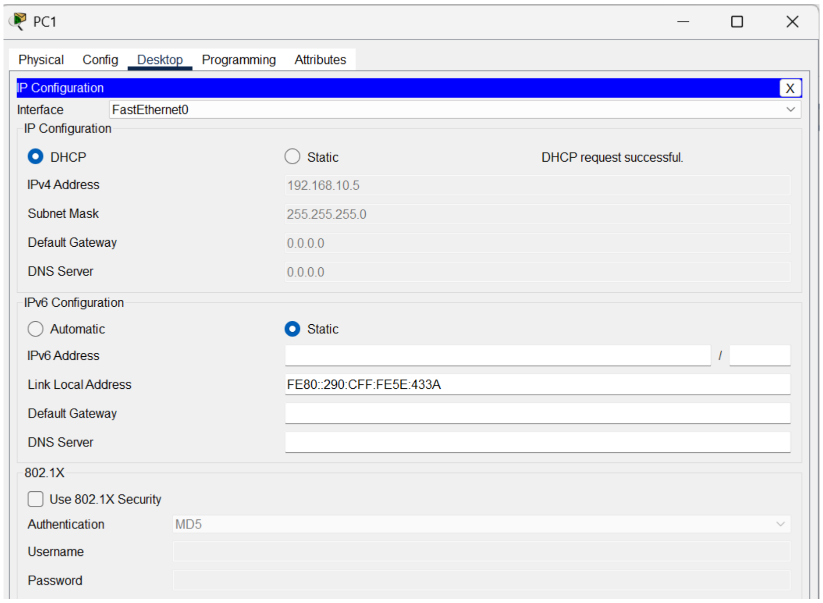
z



```
# int   Fa 0/1
# ip  address  192.168.20.1   255.255.255.0
# ip  helper-address   192.168 .10.2
                    → or Relay Agent.
# no  shutdown
  exit → do write memory
         exit
  write memory
```

Fa 0
Server-PT
Server 0
Fa 0/1
Fa 0/5
Fa 0/2
Fa 0/3
Fa 0/4
Fa 0
Fa 0
Fa 0
PC0
PC1
PC 2
Fa 0/0
1541
Router 0
Fa 0/1
Fa 0/4
Fa 0/1
Fa 0/2
Fa 0/3
Fa 0
Fa 0
Fa 0

Output:



PC1

Physical | Config | Desktop | Programming | Attributes

IP Configuration

Interface: FastEthernet0

IP Configuration

● DHCP ○ Static DHCP request successful.

IPv4 Address: 192.168.10.5

Subnet Mask: 255.255.255.0

Default Gateway: 0.0.0.0

DNS Server: 0.0.0.0

IPv6 Configuration

○ Automatic ● Static

IPv6 Address: /

Link Local Address: FE80::290:CFF:FE5E:433A

Default Gateway:

DNS Server:

802.1X

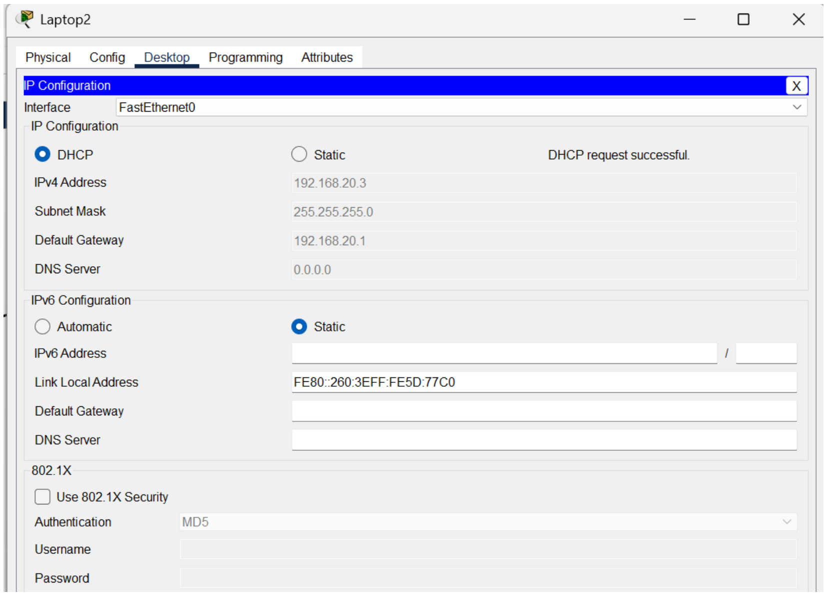
□ Use 802.1X Security

Authentication: MD5

Username:

Password:

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



Laptop2

Physical | Config | Desktop | Programming | Attributes

IP Configuration

Interface: FastEthernet0

IP Configuration

● DHCP ○ Static DHCP request successful.

IPv4 Address: 192.168.20.3

Subnet Mask: 255.255.255.0

Default Gateway: 192.168.20.1

DNS Server: 0.0.0.0

IPv6 Configuration

○ Automatic ● Static

IPv6 Address: /

Link Local Address: FE80::260:3EFF:FE5D:77C0

Default Gateway:

DNS Server:

802.1X

□ Use 802.1X Security

Authentication: MD5

Username:

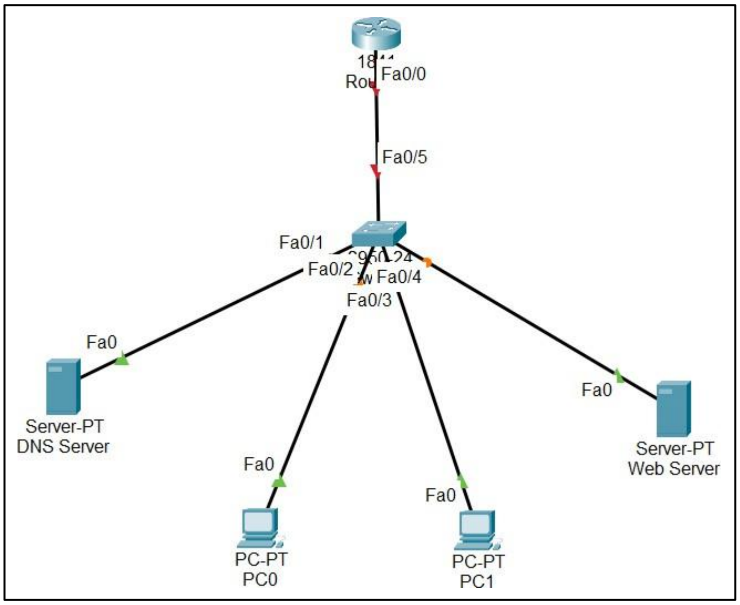
Password:

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

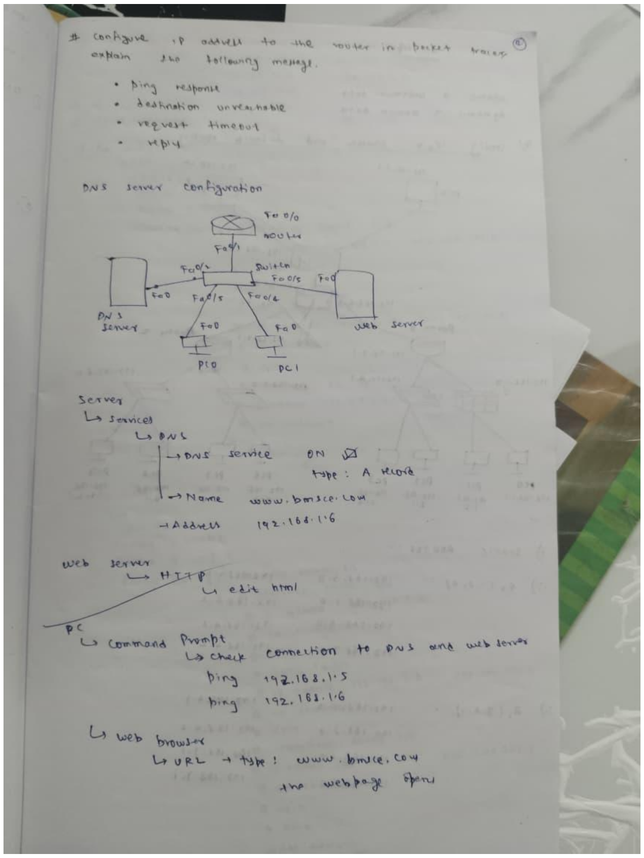
Program 3:

Aim: Configure Web Server, DNS within a LAN.

Network diagram:



Configuration:



configure IP address to the router in packet tracer
explain the following message.

- ping response
- destination unreachable
- request timeout
- reply

DNS server configuration

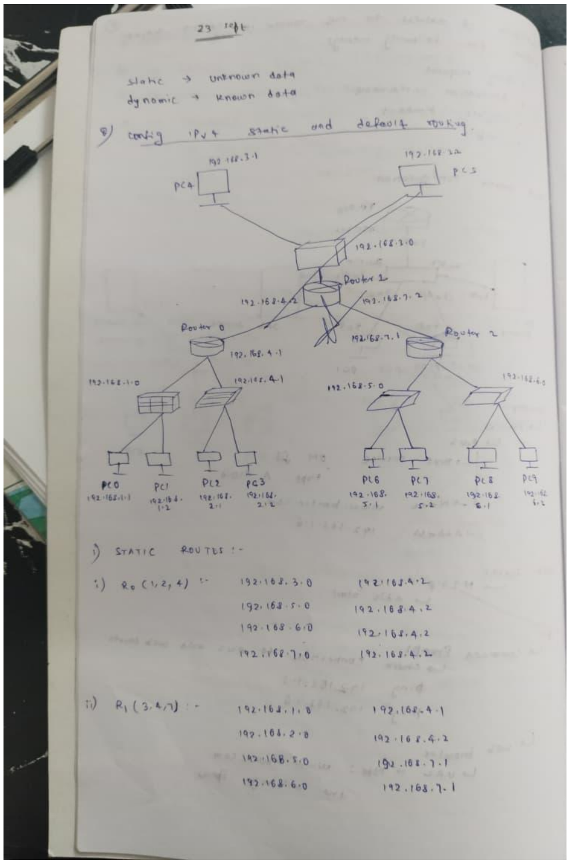
Fa 0/0
router
Fa0/1
Switch
Fa0/2
Fa0/5
Fa0
Fa0/3
Fa0/4
Fa0
DNS
Server
Fa0
Fa0
web server
PC0
PC1

Server
↳ services
  ↳ DNS
    → DNS service ON ☑
                 type : A record
    → Name www.bmsce.com
    → Address 192.168.1.6

web server
  ↳ HTTP
    ↳ edit html

PC
↳ Command Prompt
  ↳ check connection to DNS and web server
    ping 192.168.1.5
    ping 192.168.1.6

↳ web browser
  ↳ URL → type : www.bmsce.com
    the webpage opens



23 sept

static → unknown data
dynamic → known data

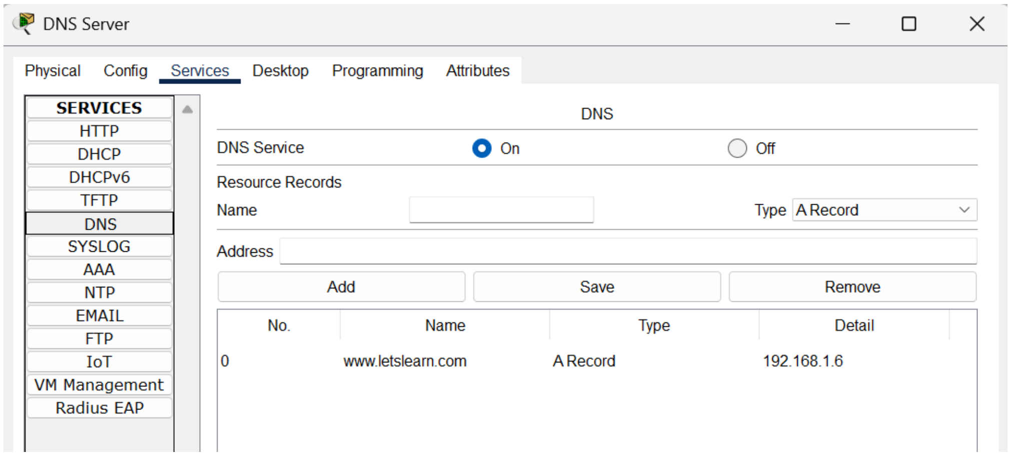
8) config IPv4 static and default routing.

192.168.3.1
PC4
192.168.3.2
PC5
192.168.3.0
Router 1
192.168.4.2
192.168.7.2
Router 0
192.168.4.1
192.168.7.1
Router 2
192.168.1.0
192.168.4.1
192.168.5.0
192.168.6.0
PC0
192.168.1.1
PC1
192.168.1.2
PC2
192.168.2.1
PC3
192.168.2.2
PC6
192.168.5.1
PC7
192.168.5.2
PC8
192.168.6.1
PC9
192.168.6.2

1) STATIC ROUTES :-

i) R_0 (1, 2, 4) :-	192.168.3.0	192.168.4.2
	192.168.5.0	192.168.4.2
	192.168.6.0	192.168.4.2
	192.168.7.0	192.168.4.2
ii) R_1 (3, 4, 7) :-	192.168.1.0	192.168.4.1
	192.168.2.0	192.168.4.2
	192.168.5.0	192.168.7.1
	192.168.6.0	192.168.7.1

Output:



DNS

DNS Service ● On ○ Off

Resource Records

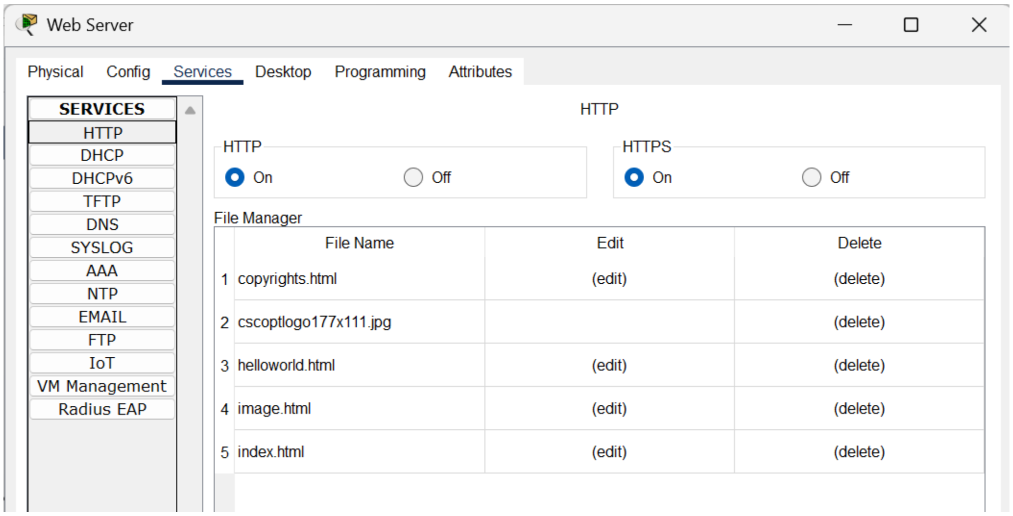
Name

Type A Record

Address

No.	Name	Type	Detail
0	www.letslearn.com	A Record	192.168.1.6

Fig 1. DNS server – DNS Services



HTTP

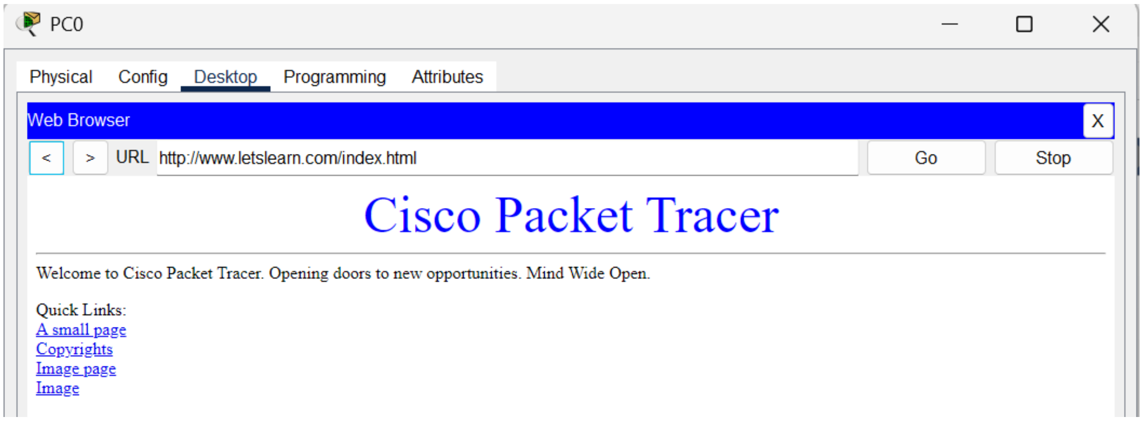
HTTP ● On ○ Off

HTTPS ● On ○ Off

File Manager

	File Name	Edit	Delete
1	copyrights.html	(edit)	(delete)
2	cscoptlogo177x111.jpg		(delete)
3	helloworld.html	(edit)	(delete)
4	image.html	(edit)	(delete)
5	index.html	(edit)	(delete)

Fig 2. WEB server – HTTP Services



URL http://www.letslearn.com/index.html

Cisco Packet Tracer

Welcome to Cisco Packet Tracer. Opening doors to new opportunities. Mind Wide Open.

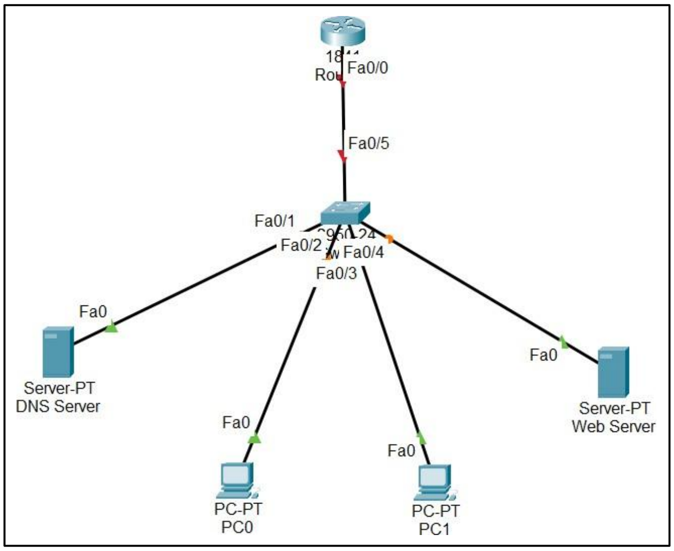
Quick Links:
A small page
Copyrights
Image page
Image

Fig 3. PC0 – accessing data from web browser

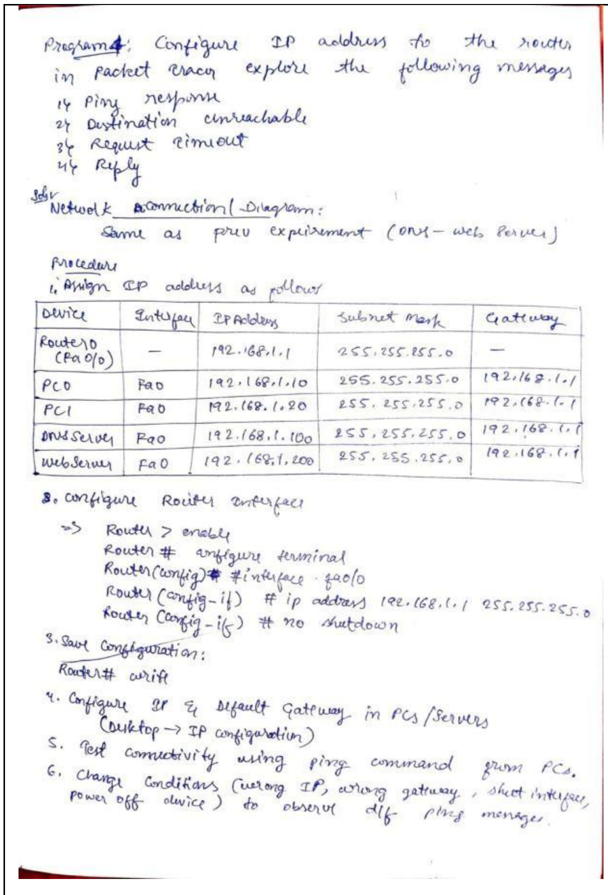
Program 4:

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 4: Configure IP address to the router in packet tracer explore the following messages

1) Ping response
2) Destination unreachable
3) Request timeout
4) Reply

Soln
Network connection/ Diagram:
Same as prev expeirement (DNS - web Server)

Procedure
1. Assign IP address as follows

Device	Interface	IP Address	Subnet mask	Gateway
Router0 (Fa 0/0)	—	192.168.1.1	255.255.255.0	—
PC0	Fa0	192.168.1.10	255.255.255.0	192.168.1.1
PC1	Fa0	192.168.1.20	255.255.255.0	192.168.1.1
DNS Server	Fa0	192.168.1.100	255.255.255.0	192.168.1.1
WebServer	Fa0	192.168.1.200	255.255.255.0	192.168.1.1

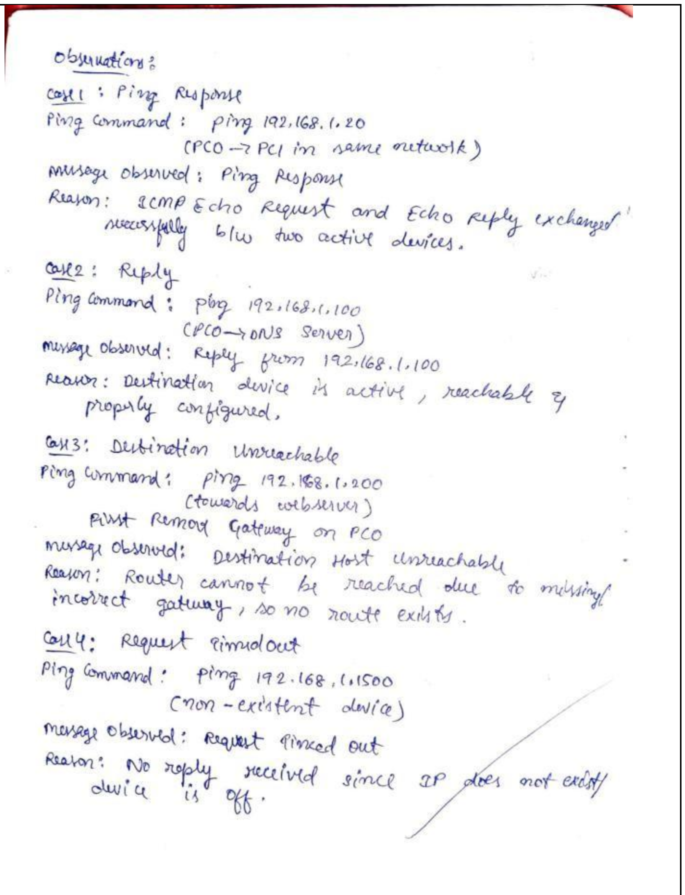
2. configure Router interface

```
=>  Router > enable
    Router # configure terminal
    Router(config)# #interface fa0/0
    Router (config-if) # ip address 192.168.1.1 255.255.255.0
    Router (Config-if) # no shutdown
```

3. Save Configuration:

```
Router# write
```

4. Configure IP & Default Gateway in PCs/Servers (Desktop → IP configuration)
5. Test connectivity using ping command from PCs.
6. Change Conditions (wrong IP, wrong gateway, shut interface, power off device) to observe dif ping messages.



Observations:

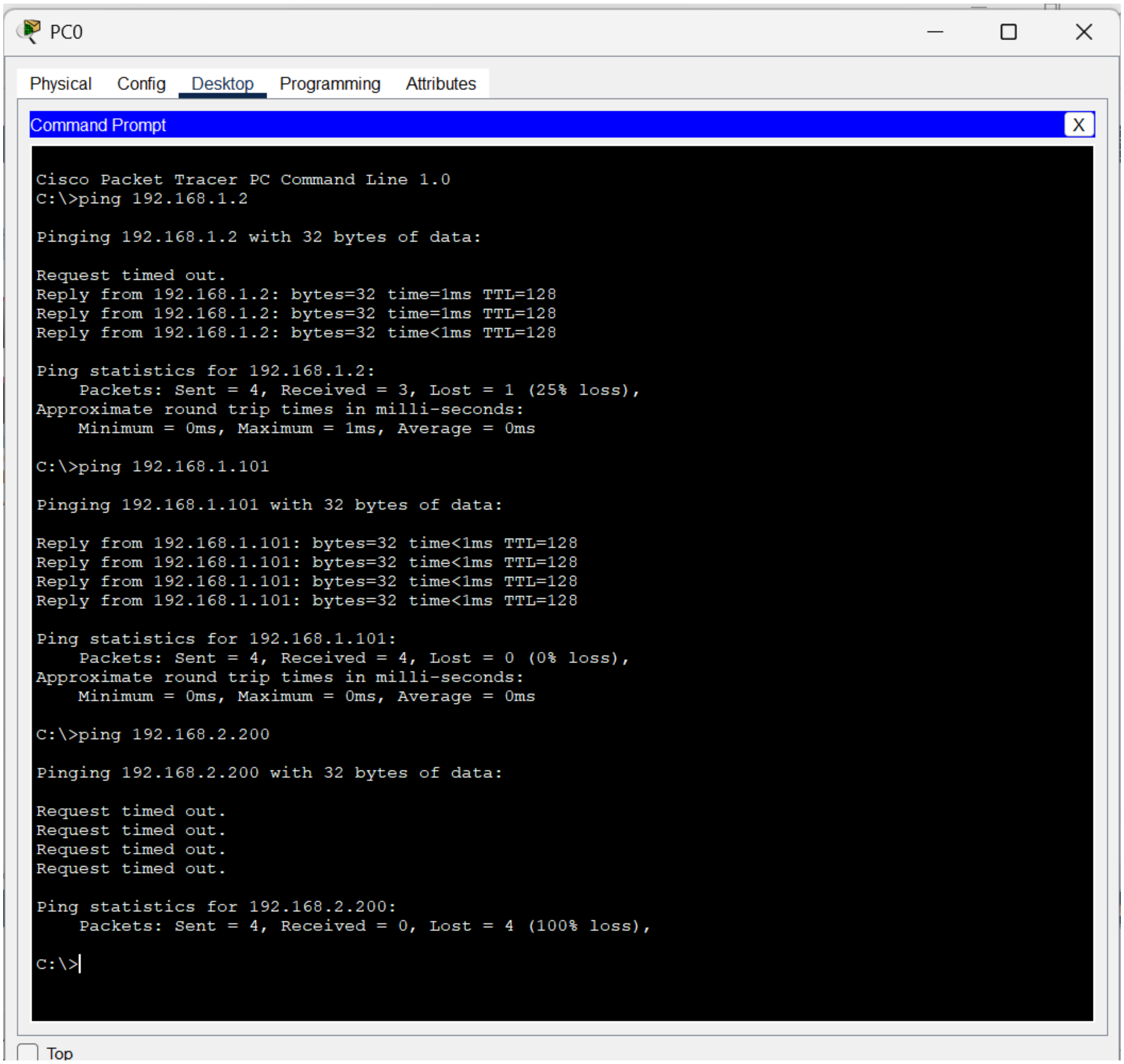
Case 1 : Ping Response
Ping Command : ping 192.168.1.20
(PC0 → PC1 in same network)
Message observed : Ping Response
Reason: ICMP Echo Request and Echo reply exchanged successfully b/w two active devices.

Case 2 : Reply
Ping Command : ping 192.168.1.100
(PC0 → DNS Server)
Message observed: Reply from 192.168.1.100
Reason: Destination device is active, reachable & properly configured.

Case 3: Destination Unreachable
Ping Command: ping 192.168.1.200
(towards webserver)
First Remove Gateway on PC0
Message observed: Destination Host unreachable
Reason: Router cannot be reached due to missing/ incorrect gateway, so no route exists.

Case 4: Request timed out
Ping Command: ping 192.168.1.1500
(non-existent device)
Message observed: Request timed out
Reason: No reply received since IP does not exist/ device is off.

Output:

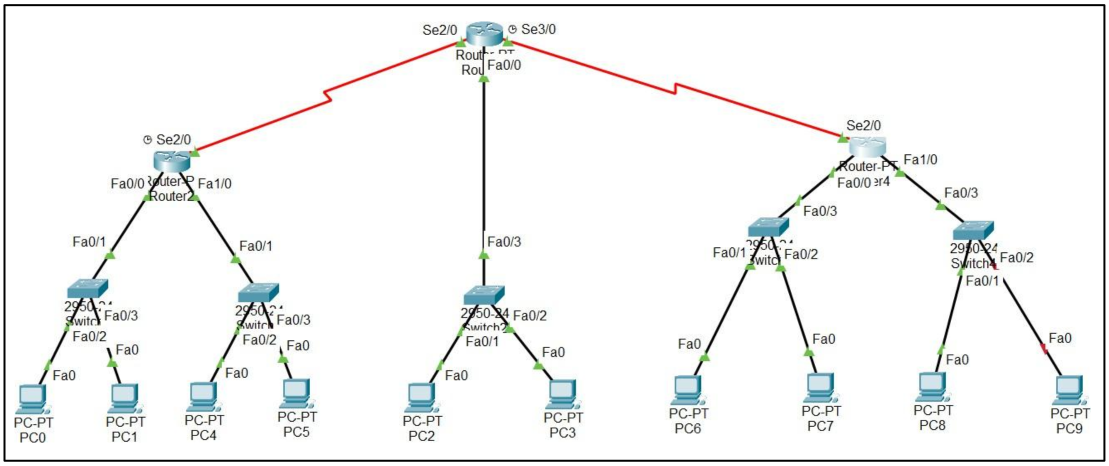


```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.2

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:\>
```

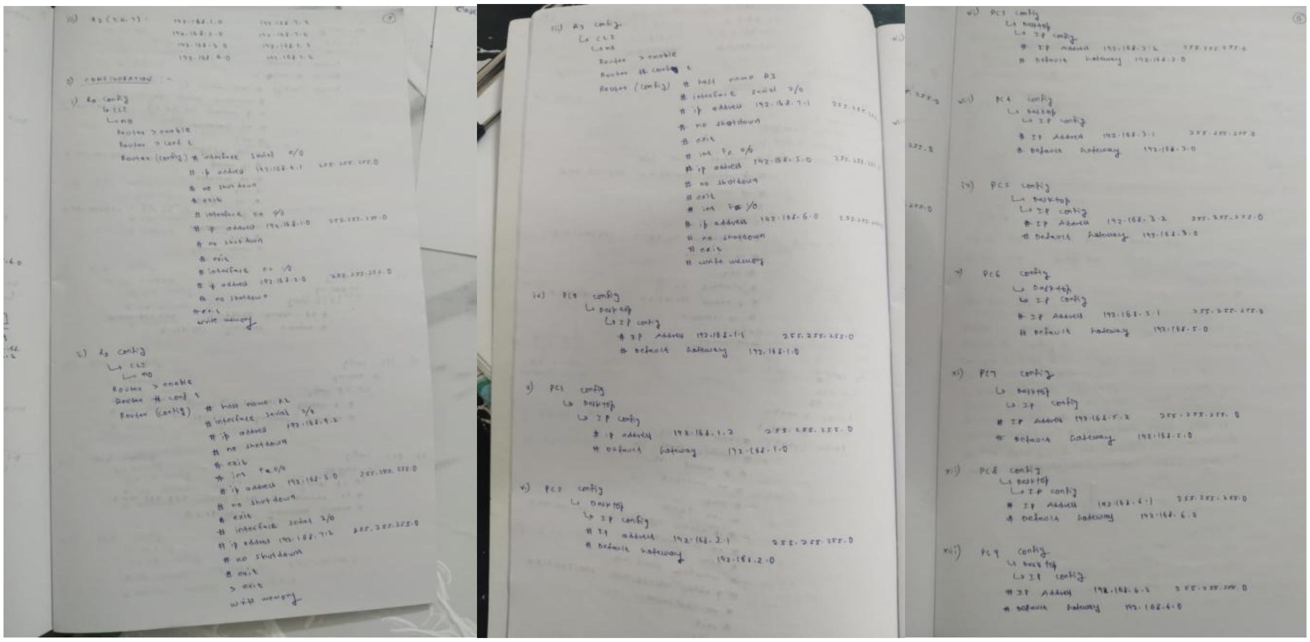
Program 5:

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

Network diagram:



Configuration:

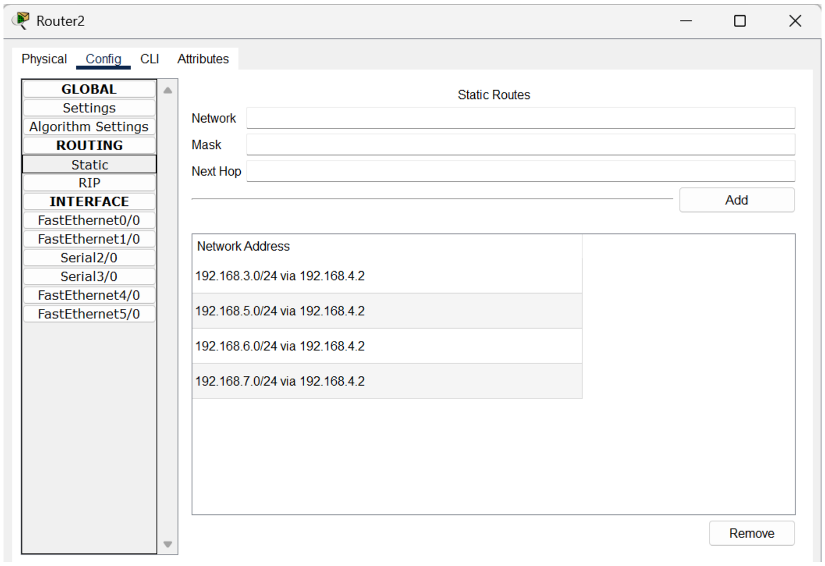


```
iii) R2 (S.R.T) :    192.168.1.0      192.168.7.1
                     192.168.2.0      192.168.7.2
                     192.168.3.0      192.168.7.1
                     192.168.4.0      192.168.7.2

2) CONFIGURATION :-

i) R0 config
     ↳ CLI
       ↳ no
        Router > enable
        Router > conf t
        Router (config) # interface serial 2/0
                        # ip address 192.168.4.1     255.255.255.0
                        # no shutdown
                        # exit
                        # interface fa 0/0
                        # ip address 192.168.1.0     255.255.255.0
                        # no shutdown
                        # exit
                        # interface fa 1/0
                        # ip address 192.168.2.0     255.255.255.0
                        # no shutdown
                        # exit
                        write memory

ii) R2 config
     ↳ CLI
       ↳ no
    Router > enable
    Router # conf t
    Router (config)  # host name R2
                     # interface serial 2/0
                     # ip address     192.168.4.2
                     # no shutdown
                     # exit
                     # int fa0/0
                     # ip address 192.168.3.0     255.255.255.0
                     # no shutdown
                     # exit
                     # interface serial 3/0
                     # ip address 192.168.7.2     255.255.255.0
                     # no shutdown
                     # exit
                     > exit
                     write memory
```

```
iii) R3 config
      ↳ CLI
        ↳ no
       Router > enable
       Router # config t
       Router (config)  # host name R3
                        # interface serial 2/0
                        # ip address 192.168.7.1     255.255.255.0
                        # no shutdown
                        # exit
                        # int fa 0/0
                        # ip address 192.168.5.0     255.255.255.0
                        # no shutdown
                        # exit
                        # int Fa 1/0
                        # ip address 192.168.6.0     255.255.255.0
                        # no shutdown
                        # exit
                        # write memory

iv) PC0 config
      ↳ Desktop
        ↳ IP config
          # IP Address 192.168.1.1         255.255.255.0
          # Default Gateway 192.168.1.0

v) PC1 config
     ↳ Desktop
       ↳ IP config
         # ip address 192.168.1.2       255.255.255.0
         # Default Gateway 192.168.1.0

vi) PC2 config
      ↳ Desktop
        ↳ IP config
         # IP address 192.168.2.1       255.255.255.0
         # Default Gateway      192.168.2.0
```

```
vii) PC3 config
      ↳ Desktop
        ↳ IP config
         # IP Address 192.168.2.2     255.255.255.0
         # Default Gateway 192.168.2.0

viii) PC4 config
       ↳ Desktop
         ↳ IP config
         # IP Address    192.168.3.1       255.255.255.0
         # Default Gateway    192.168.3.0

ix) PC5 config
      ↳ Desktop
        ↳ IP config
         # IP Address  192.168.3.2     255.255.255.0
         # Default Gateway  192.168.3.0

x) PC6 config
      ↳ Desktop
       ↳ IP config
       # IP Address    192.168.5.1       255.255.255.0
       # Default Gateway    192.168.5.0

xi) PC7 config
      ↳ Desktop
       ↳ IP config
     # IP Address 192.168.5.2      255.255.255.0
     # Default Gateway   192.168.5.0

xii) PC8 config
      ↳ Desktop
       ↳ IP config
       # IP Address     192.168.6.1     255.255.255.0
       # Default Gateway    192.168.6.0

xiii) PC9 config
       ↳ Desktop
        ↳ IP config
       # IP Address     192.168.6.2      255.255.255.0
       # Default Gateway    192.168.6.0
```

Output:

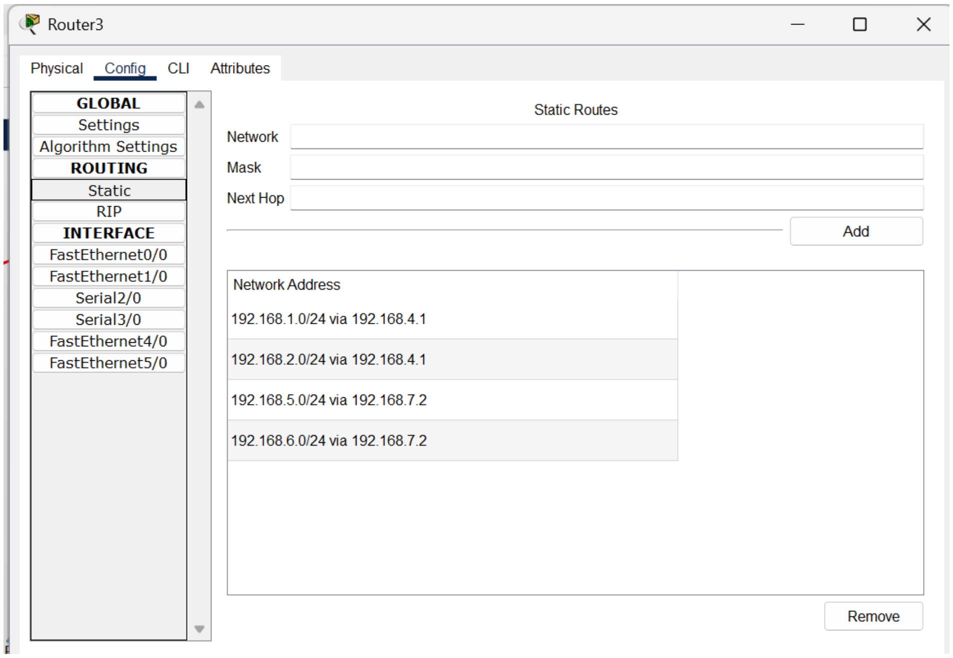


Static Routes

Network
Mask
Next Hop

Network Address
192.168.3.0/24 via 192.168.4.2
192.168.5.0/24 via 192.168.4.2
192.168.6.0/24 via 192.168.4.2
192.168.7.0/24 via 192.168.4.2

Fig 1. Router 2 – Static routing



Network Address

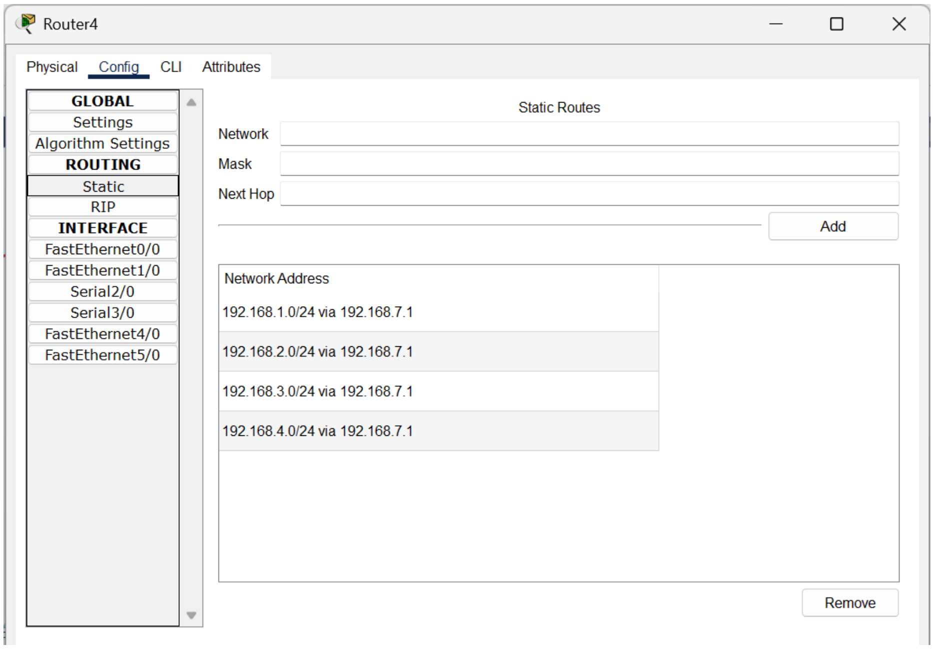
192.168.1.0/24 via 192.168.4.1

192.168.2.0/24 via 192.168.4.1

192.168.5.0/24 via 192.168.7.2

192.168.6.0/24 via 192.168.7.2

Fig 2. Router 3 – Static routing



Network Address

192.168.1.0/24 via 192.168.7.1

192.168.2.0/24 via 192.168.7.1

192.168.3.0/24 via 192.168.7.1

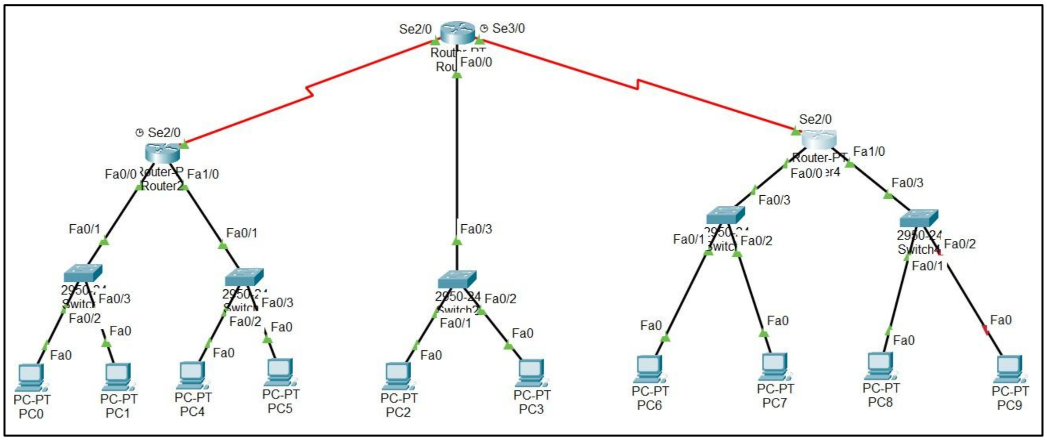
192.168.4.0/24 via 192.168.7.1

Fig 3. Router 4 – Static routing

Program 6:

Aim: Configure RIP routing Protocol in Routers.

Network diagram:



Configuration:

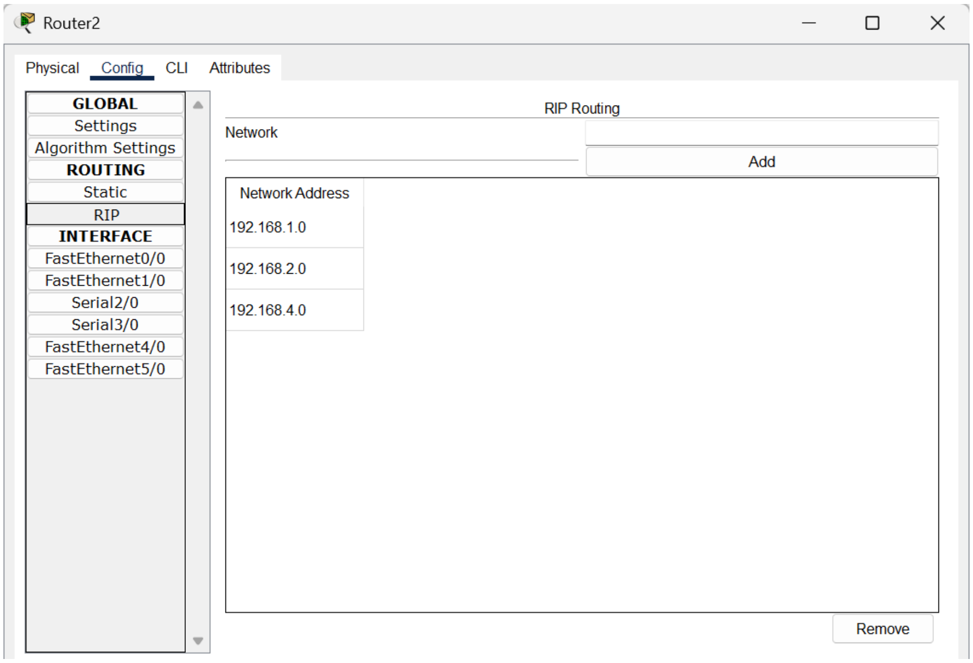
② Dynamic Route :-

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.,

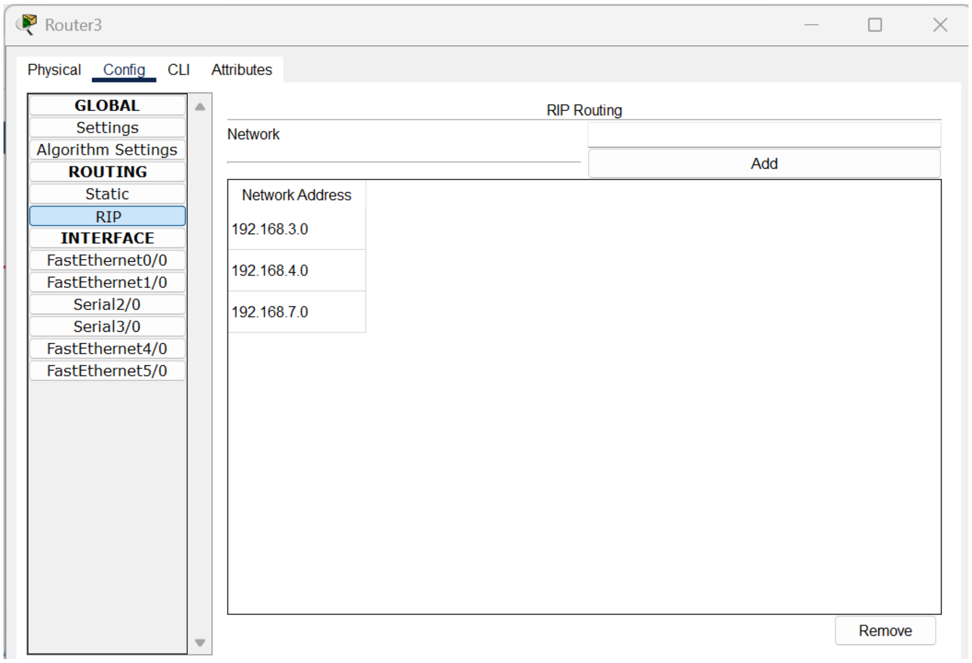
```
* Router 1: (select Router-PT)
    ↳ config
        ↳ Routing
            ↳ RIP Routing
                ↳ Networks: 192.168.1.0
                            192.168.2.0
                            192.168.4.0
                    then click on add [For each]
* Router 2:
    ↳ Config
        ↳ Routing
            ↳ 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:



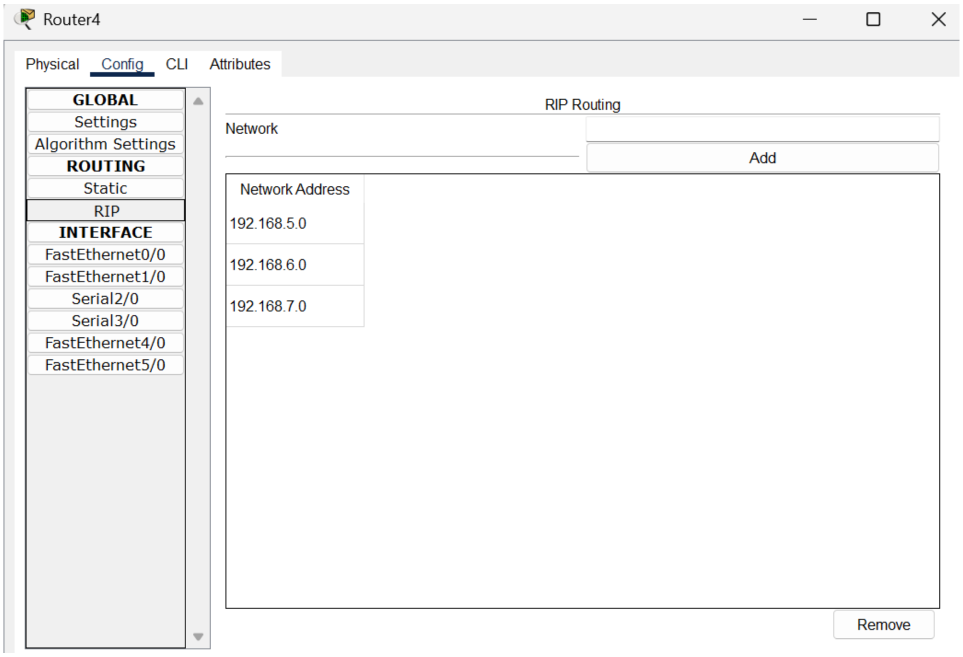
Network Address
192.168.1.0
192.168.2.0
192.168.4.0

Fig 1. Router 2 – RIP routing



Network Address
192.168.3.0
192.168.4.0
192.168.7.0

Fig 2. Router 3 – RIP routing



RIP Routing

Network

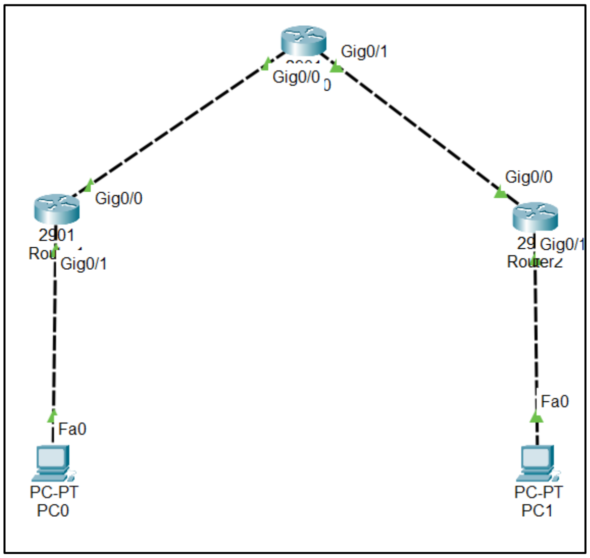
Network Address
192.168.5.0
192.168.6.0
192.168.7.0

Fig 3. Router 4 – RIP routing

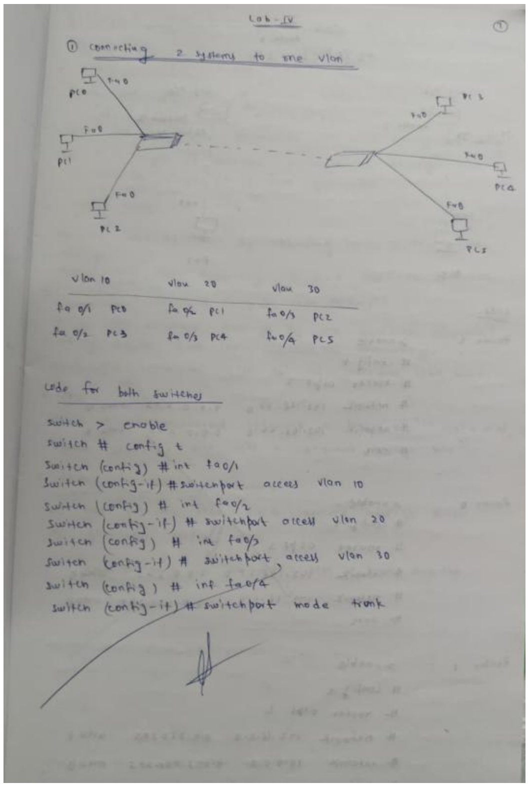
Program 7:

Aim: Configure OSPF routing protocol.

Network diagram:



Configuration:



Lab - IV

①

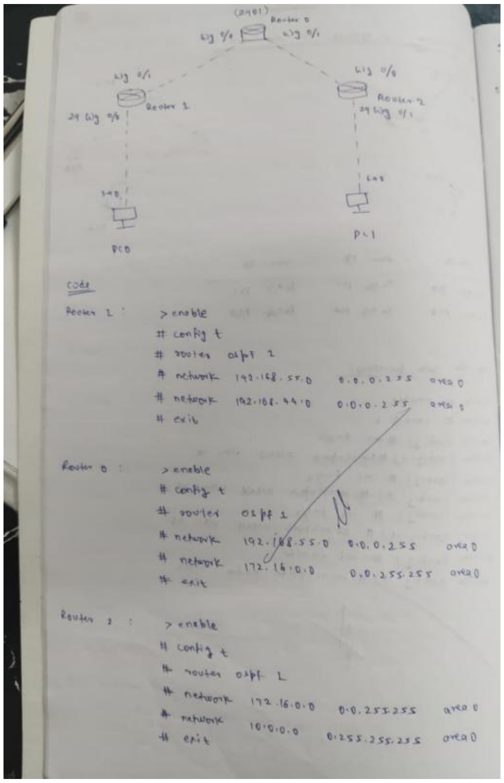
① connecting 2 systems to one vlan

PC0
Fa0
Fa0
PC1
Fa0
PC2
PC3
Fa0
Fa0
PC4
Fa0
PC5

vlan 10		vlan 20		vlan 30	
fa 0/1	PC0	fa 0/2	PC1	fa 0/3	PC2
fa 0/2	PC3	fa 0/3	PC4	fa 0/4	PC5

code for both switches

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



(2901)
Router 0
Gig 0/0
Gig 0/1
Gig 0/1
Router 1
29 Gig 0/0
Gig 0/0
Router 2
29 Gig 0/1
Fa0
Fa0
PC0
PC1

code

```
Router 1 :    > enable
              # config t
              # router ospf 1
              # network  192.168.55.0   0.0.0.255   area 0
              # network  192.168.44.0   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
```

Output:

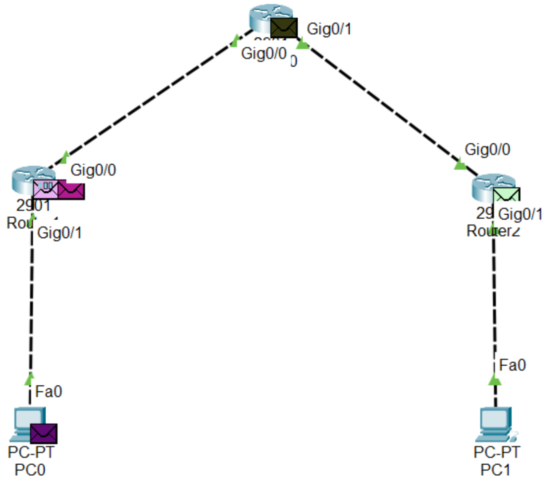
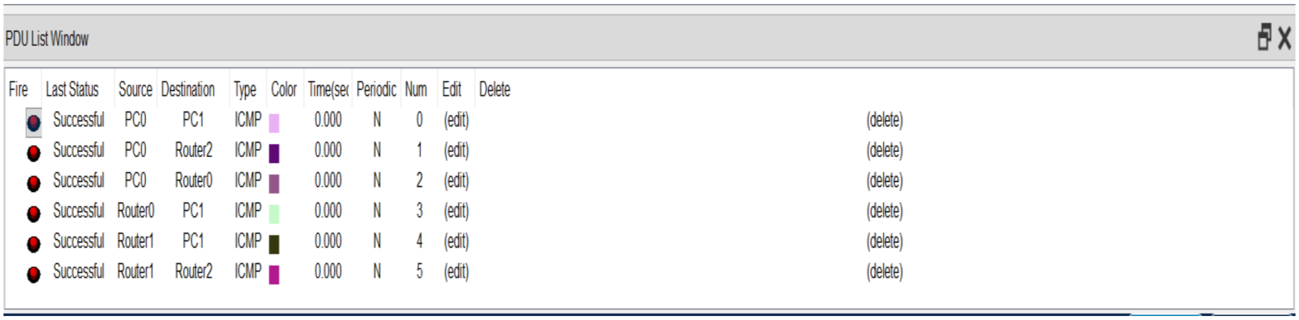


Fig 1. Sending PDU message from PC0 to PC1



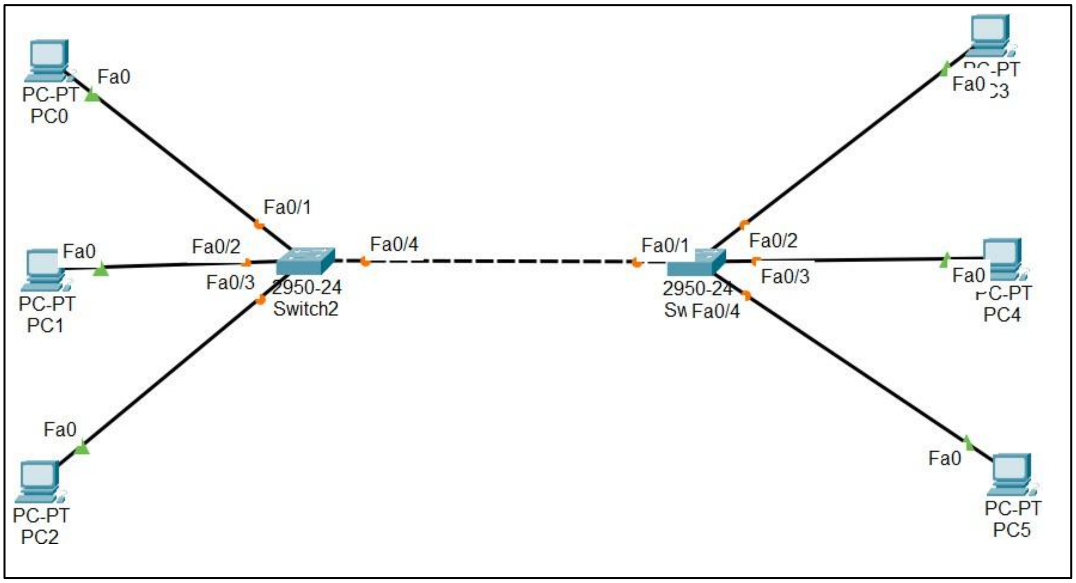
Fire	Last Status	Source	Destination	Type	Color	Time(sec	Periodic	Num	Edit	Delete
●	Successful	PC0	PC1	ICMP	■	0.000	N	0	(edit)	(delete)
●	Successful	PC0	Router2	ICMP	■	0.000	N	1	(edit)	(delete)
●	Successful	PC0	Router0	ICMP	■	0.000	N	2	(edit)	(delete)
●	Successful	Router0	PC1	ICMP	■	0.000	N	3	(edit)	(delete)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

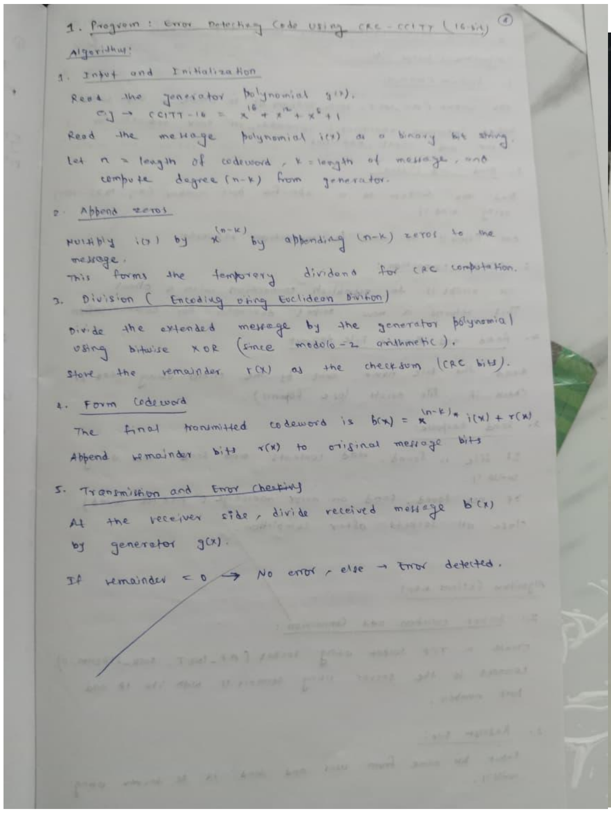
Program 8:

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

Network diagram:



Configuration:



1. Program : Error Detecting Code using CRC - CCITT (16-bit) ④

Algorithm:

1. Input and Initialization

Read the generator polynomial g(x).
eg → CCITT-16 = $x^{16} + x^{12} + x^{5} + 1$

Read the message polynomial i(x) as a binary bit string.

Let n = length of codeword, k = length of message, and compute degree (n-k) from generator.

2. Append zeros

Multiply i(x) by $x^{(n-k)}$ by appending (n-k) zeros to the message.
This forms the temporery dividend for CRC computation.

3. Division (Encoding using Euclidean Division)

Divide the extended message by the generator polynomial using bitwise XOR (since modolo-2 arithmetic).
Store the remainder r(x) as the checksum (CRC bits).

4. Form Codeword

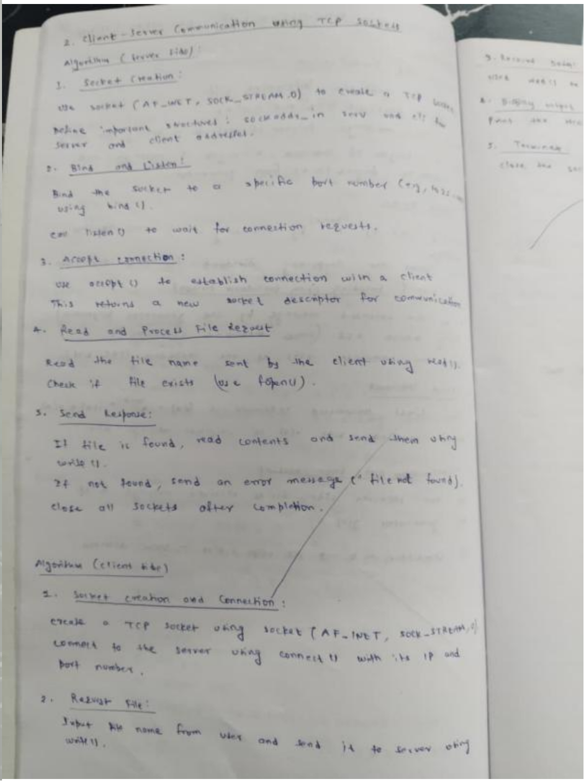
The final transmitted codeword is $b(x) = x^{(n-k)} * i(x) + r(x)$

Append remainder bits r(x) to original message bits

5. Transmission and Error Checking

At the receiver side, divide received message b'(x) by generator g(x).

If remainder = 0 → No error, else → Error detected.



2. Client - Server Communication using TCP Sockets

Algorithm (Server Side):

1. Socket Creation:

Use socket (AF_INET, SOCK_STREAM, 0) to create a TCP socket.
Define important structures : sockaddr_in serv and cli for server and client addresses.

2. Bind and Listen:

Bind the socket to a specific port number (e.g, 5432) using bind().
call listen() to wait for connection requests.

3. Accept connection :

Use accept () to establish connection with a client
This returns a new socket descriptor for communication

4. Read and Process File Request

Read the file name sent by the client using read().
Check if file exists (use fopen()).

5. Send Response:

If file is found, read contents and send them using write().
If not found, send an error message (" file not found).
close all sockets after completion.

Algorithm (client side)

1. Socket creation and Connection :

create a TCP socket using socket (AF_INET, SOCK_STREAM, 0)
connect to the server using connect() with its IP and port number.

2. Request file:

Input file name from user and send it to server using write().

3. Receive Data:
use read() to
4. Display output
Print the
5. Terminate
close the so

Output:

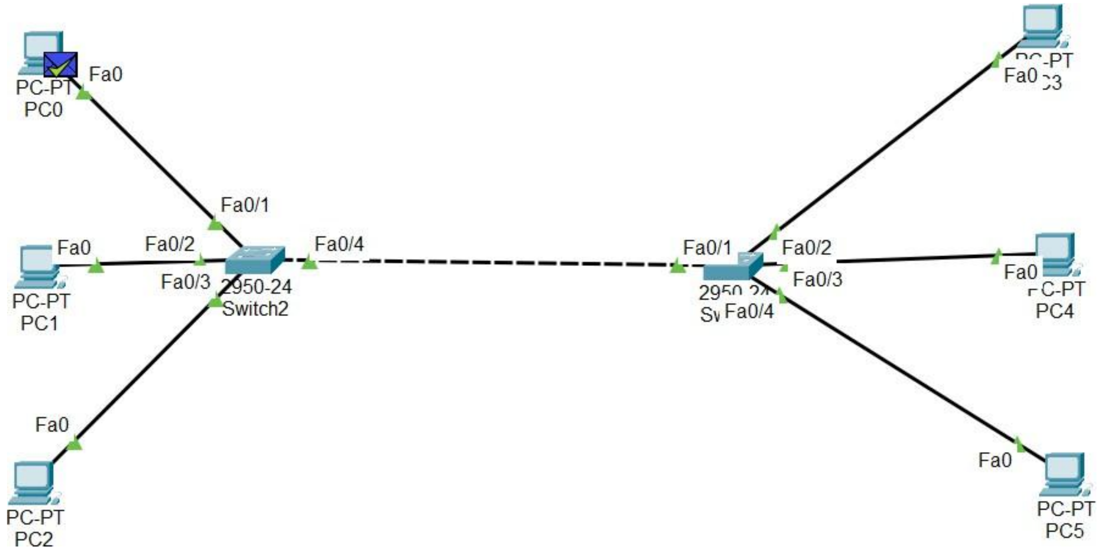
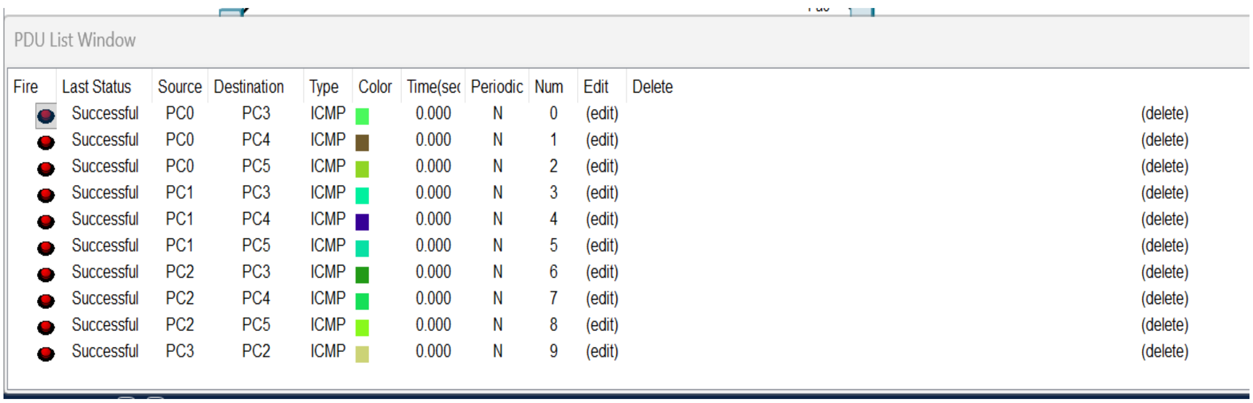


Fig 1. Sending PDU message from PC0 to PC5


PDU List Window

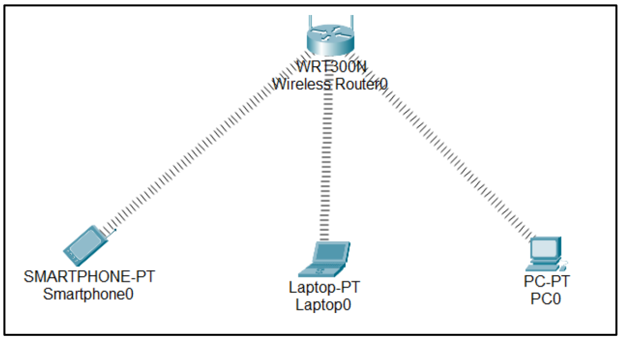
Fire	Last Status	Source	Destination	Type	Color	Time(sec	Periodic	Num	Edit	Delete
●	Successful	PC0	PC3	ICMP	🟩	0.000	N	0	(edit)	(delete)
●	Successful	PC0	PC4	ICMP	🟫	0.000	N	1	(edit)	(delete)
●	Successful	PC0	PC5	ICMP	🟩	0.000	N	2	(edit)	(delete)
●	Successful	PC1	PC3	ICMP	🟦	0.000	N	3	(edit)	(delete)
●	Successful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

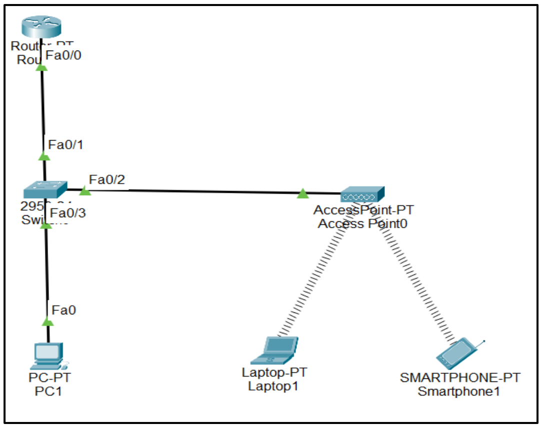
Program 9:

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

Network diagram:

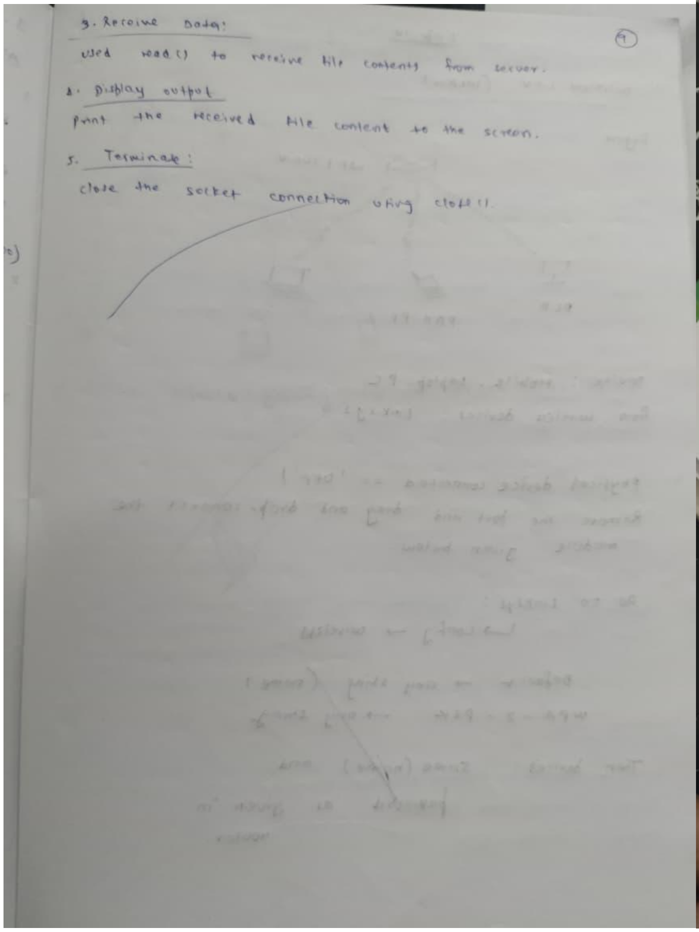


Configuration 1



Configuration 2

Configuration:



3. Receive Data:

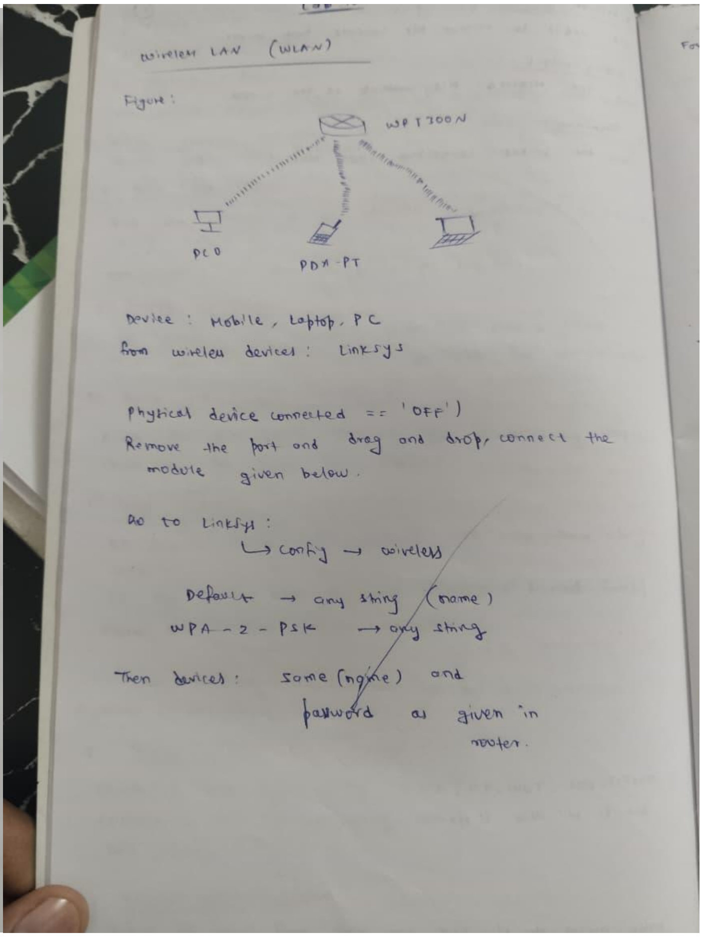
Used read() to receive file contents from server.

4. Display output

Print the received file content to the screen.

5. Terminate:

Close the socket connection using close().



Wireless LAN (WLAN)

Figure:

WRT300N
PC0
PDA-PT

Device : Mobile, Laptop, PC

from wireless devices : Linksys

Physical device connected == 'OFF')

Remove the port and drag and drop, connect the module given below.

Go to Linksys :

↳ config → wireless

Default → any string (name)

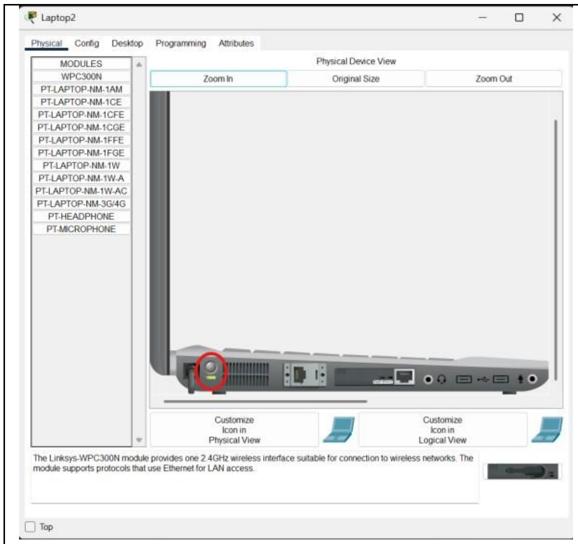
WPA-2-PSK → any string

Then devices : same (name) and password as given in router.

Output:

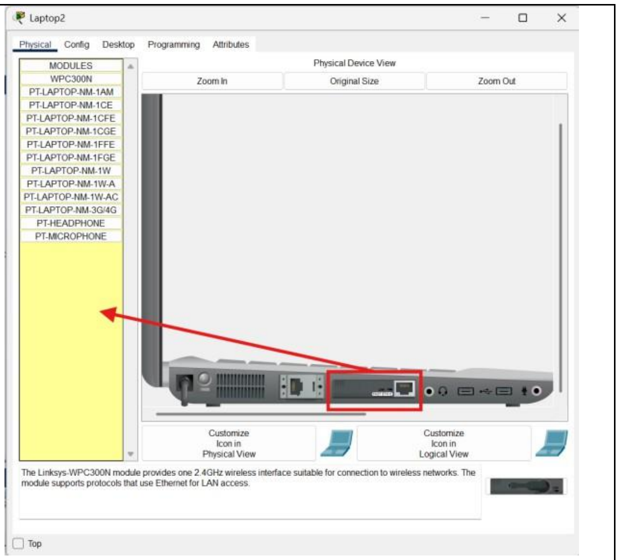
1. Do Physical Connections In:

- Laptop
- PC



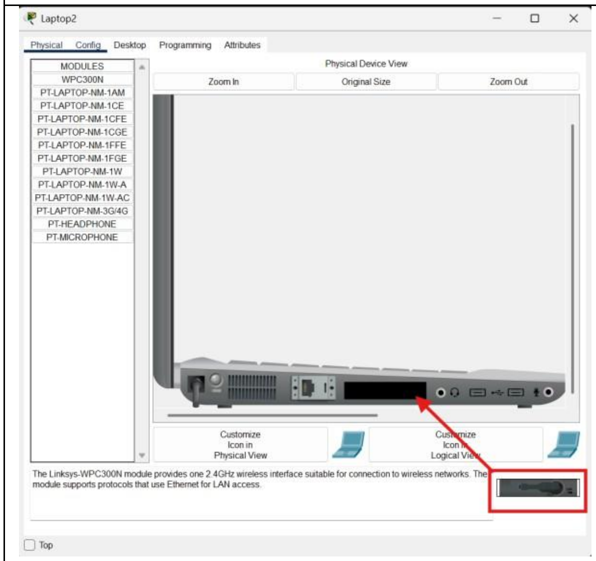
The Linksys-WPC300N module provides one 2.4GHz wireless interface suitable for connection to wireless networks. The module supports protocols that use Ethernet for LAN access.

Fig 1.1 Step1: Turn off light / Power off laptop



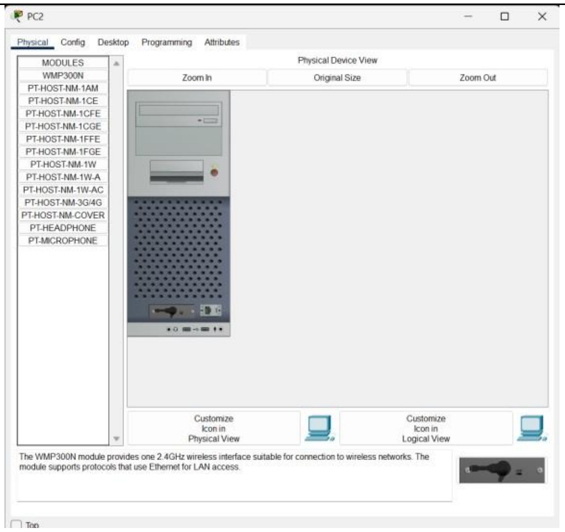
The Linksys-WPC300N module provides one 2.4GHz wireless interface suitable for connection to wireless networks. The module supports protocols that use Ethernet for LAN access.

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



The Linksys-WPC300N module provides one 2.4GHz wireless interface suitable for connection to wireless networks. The module supports protocols that use Ethernet for LAN access.

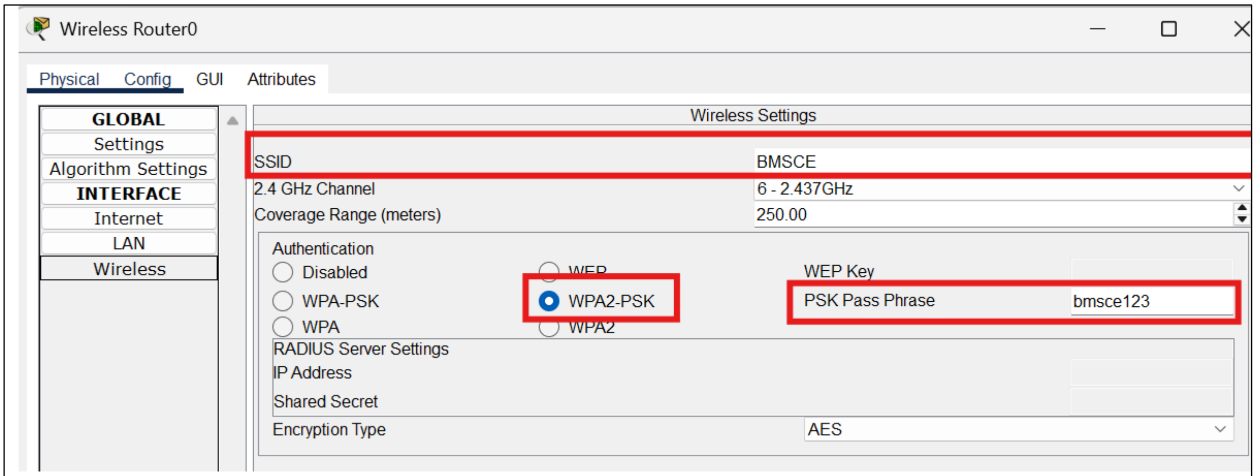
Fig 1.3 Step3: Drag and Drop the device into pointed location and Turn on light/Laptop



The WMP300N module provides one 2.4GHz wireless interface suitable for connection to wireless networks. The module supports protocols that use Ethernet for LAN access.

Fig 2. PC physical connection (combined 3 steps)

2. Do Wireless Connection in:


Wireless Router0

Wireless Settings

SSID: BMSCE
2.4 GHz Channel: 6 - 2.437GHz
Coverage Range (meters): 250.00

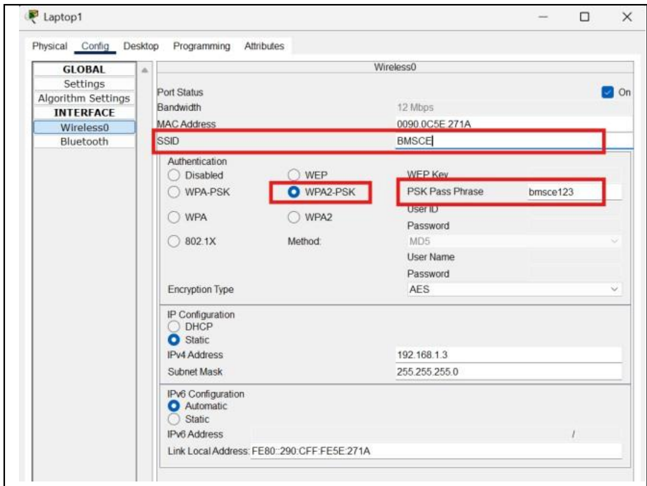
Authentication
○ Disabled ○ WEP
○ WPA-PSK ● WPA2-PSK
○ WPA ○ WPA2

WEP Key
PSK Pass Phrase: bmsce123

RADIUS Server Settings
IP Address
Shared Secret

Encryption Type: AES

Fig 1. Config at Device Wireless Router0


Laptop1

Wireless0

Port Status: ☑ On
Bandwidth: 12 Mbps
MAC Address: 0090.0C5E.271A
SSID: BMSCE

Authentication
○ Disabled ○ WEP
○ WPA-PSK ● WPA2-PSK
○ WPA ○ WPA2
○ 802.1X Method:

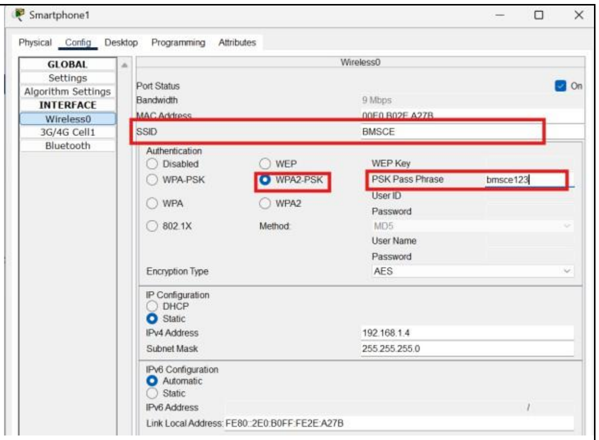
WEP Key
PSK Pass Phrase: bmsce123
User ID
Password
MD5
User Name
Password

Encryption Type: AES

IP Configuration
○ DHCP
● Static
IPv4 Address: 192.168.1.3
Subnet Mask: 255.255.255.0

IPv6 Configuration
● Automatic
○ Static
IPv6 Address: /
Link Local Address: FE80::290:CFF:FE5E:271A

Fig 2. Config at Device Laptop0


Smartphone1

Wireless0

Port Status: ☑ On
Bandwidth: 9 Mbps
MAC Address: 00E0.B02E.A27B
SSID: BMSCE

Authentication
○ Disabled ○ WEP
○ WPA-PSK ● WPA2-PSK
○ WPA ○ WPA2
○ 802.1X Method:

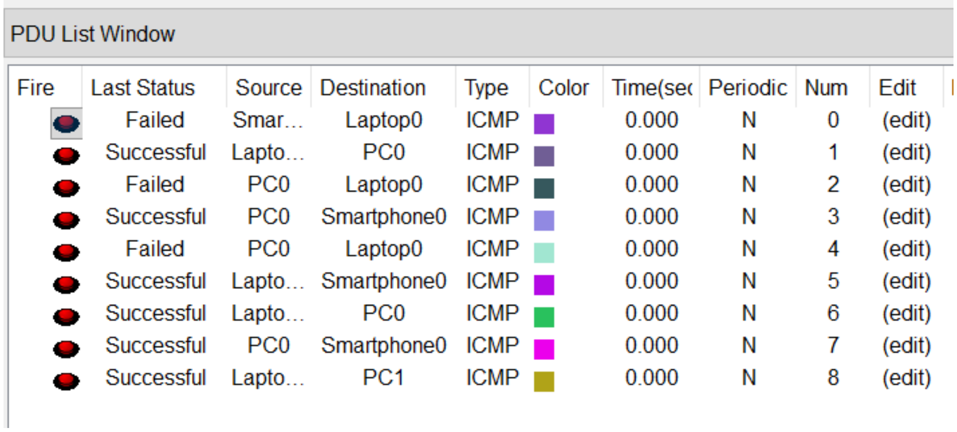
WEP Key
PSK Pass Phrase: bmsce123
User ID
Password
MD5
User Name
Password

Encryption Type: AES

IP Configuration
○ DHCP
● Static
IPv4 Address: 192.168.1.4
Subnet Mask: 255.255.255.0

IPv6 Configuration
● Automatic
○ Static
IPv6 Address: /
Link Local Address: FE80::2E0:B0FF:FE2E:A27B

Fig 3. Config at Device Smartphone0


PDU List Window

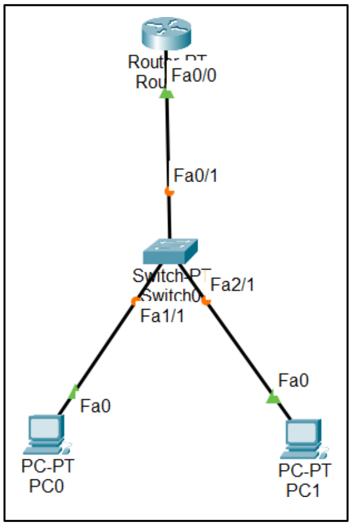
Fire	Last Status	Source	Destination	Type	Color	Time(sec	Periodic	Num	Edit
●	Failed	Smar...	Laptop0	ICMP	■	0.000	N	0	(edit)
●	Successful	Lapto...	PC0	ICMP	■	0.000	N	1	(edit)
●	Failed	PC0	Laptop0	ICMP	■	0.000	N	2	(edit)
●	Successful	PC0	Smartphone0	ICMP	■	0.000	N	3	(edit)
●	Failed	PC0	Laptop0	ICMP	■	0.000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

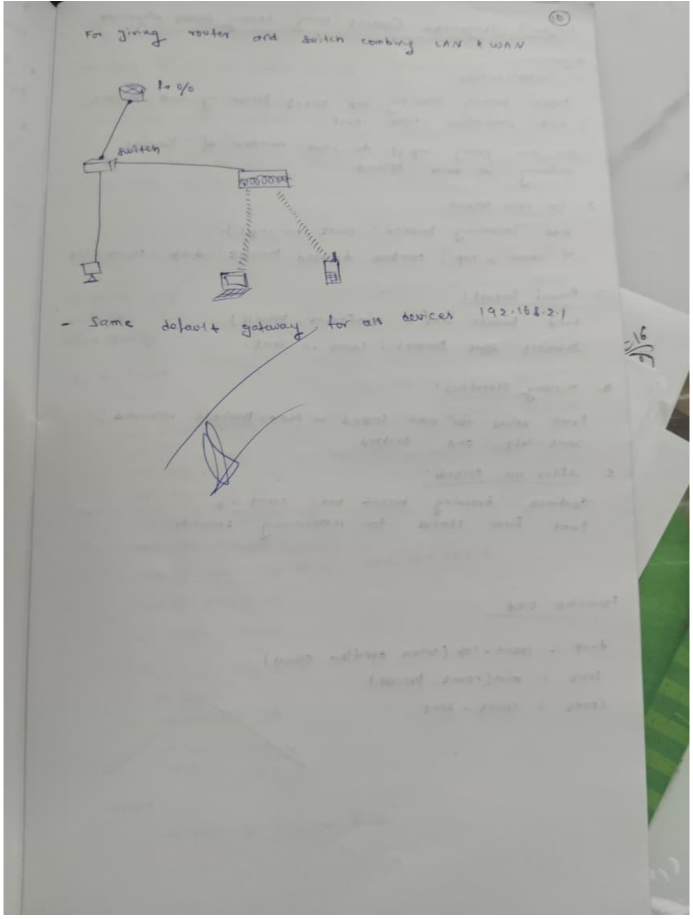
Program 10:

Aim: Demonstrate the TTL/ Life of a Packet.

Network diagram:



Configuration:



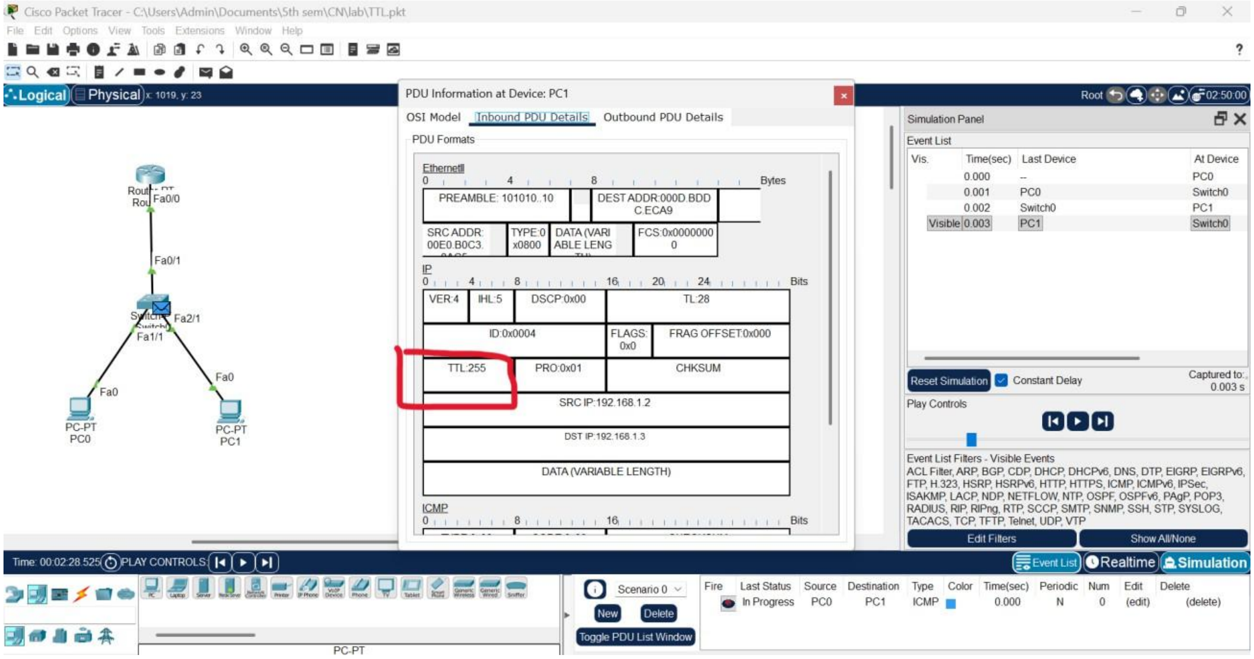
10

For giving router and switch combing LAN & WAN

1o 0/0
switch

- Same default gateway for all devices 192.168.2.1

Output:



Cisco Packet Tracer - C:\Users\Admin\Documents\5th sem\CN\lab\TTL.pkt

PDU Information at Device: PC1

OSI Model | Inbound PDU Details | Outbound PDU Details

PDU Formats

EthernetII

PREAMBLE: 101010..10 | DEST ADDR:000D.BDDC.ECA9

SRC ADDR: 00E0.B0C3. | TYPE:0x0800 | DATA (VARIABLE LENGTH) | FCS:0x00000000

IP

VER:4 | IHL:5 | DSCP:0x00 | TL:28

ID:0x0004 | FLAGS: 0x0 | FRAG OFFSET:0x000

TTL:255 | PRO:0x01 | CHKSUM

SRC IP:192.168.1.2

DST IP:192.168.1.3

DATA (VARIABLE LENGTH)

ICMP

Simulation Panel

Event List

Vis.	Time(sec)	Last Device	At Device
	0.000	--	PC0
	0.001	PC0	Switch0
	0.002	Switch0	PC1
Visible	0.003	PC1	Switch0

Reset Simulation ☑ Constant Delay

Captured to: 0.003 s

Play Controls

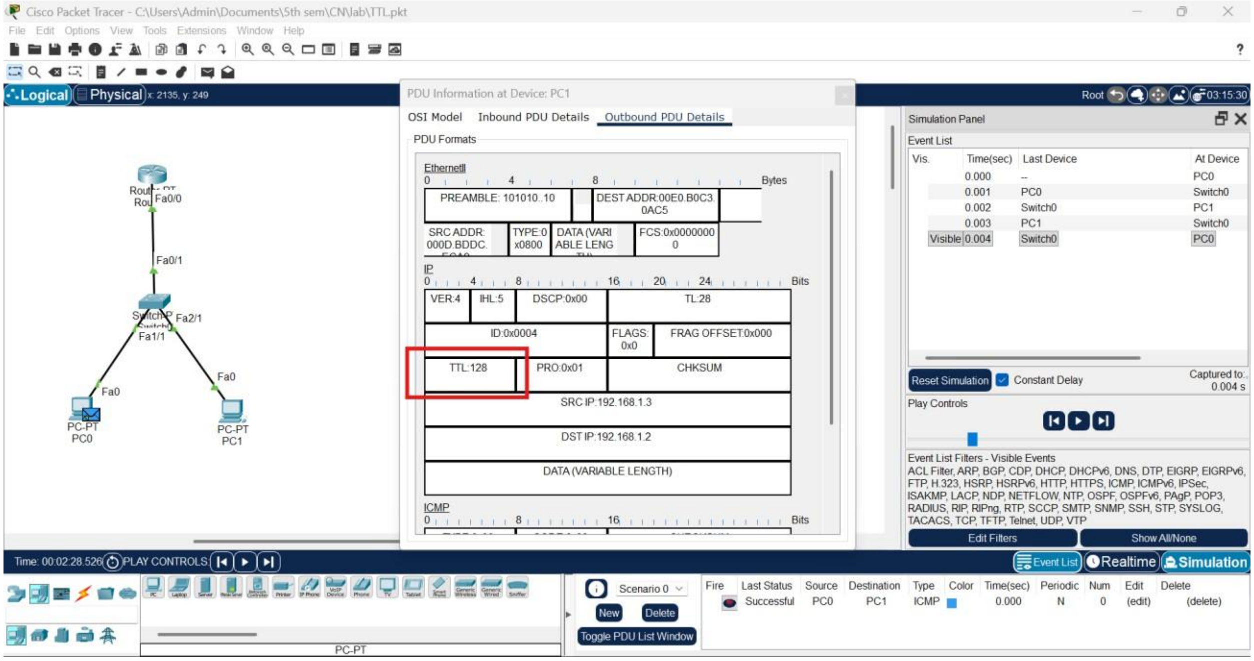
Event List Filters - Visible Events

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

Time: 00:02:28.525

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	In Progress	PC0	PC1	ICMP	🟦	0.000	N	0	(edit)	(delete)

Fig 1. Inbound PDU Details at Device PC1



Cisco Packet Tracer - C:\Users\Admin\Documents\5th sem\CN\lab\TTL.pkt

PDU Information at Device: PC1

OSI Model | Inbound PDU Details | Outbound PDU Details

PDU Formats

EthernetII

PREAMBLE: 101010..10 | DEST ADDR:00E0.B0C3.0AC5

SRC ADDR: 000D.BDDC. | TYPE:0x0800 | DATA (VARIABLE LENGTH) | FCS:0x00000000

IP

VER:4 | IHL:5 | DSCP:0x00 | TL:28

ID:0x0004 | FLAGS: 0x0 | FRAG OFFSET:0x000

TTL:128 | PRO:0x01 | CHKSUM

SRC IP:192.168.1.3

DST IP:192.168.1.2

DATA (VARIABLE LENGTH)

ICMP

Simulation Panel

Event List

Vis.	Time(sec)	Last Device	At Device
	0.000	--	PC0
	0.001	PC0	Switch0
	0.002	Switch0	PC1
	0.003	PC1	Switch0
Visible	0.004	Switch0	PC0

Reset Simulation ☑ Constant Delay

Captured to: 0.004 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

Time: 00:02:28.526

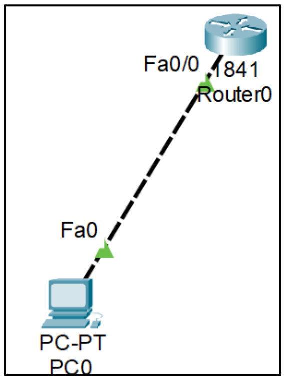
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC0	PC1	ICMP	🟦	0.000	N	0	(edit)	(delete)

Fig 1. Outbound PDU Details at Device PC1

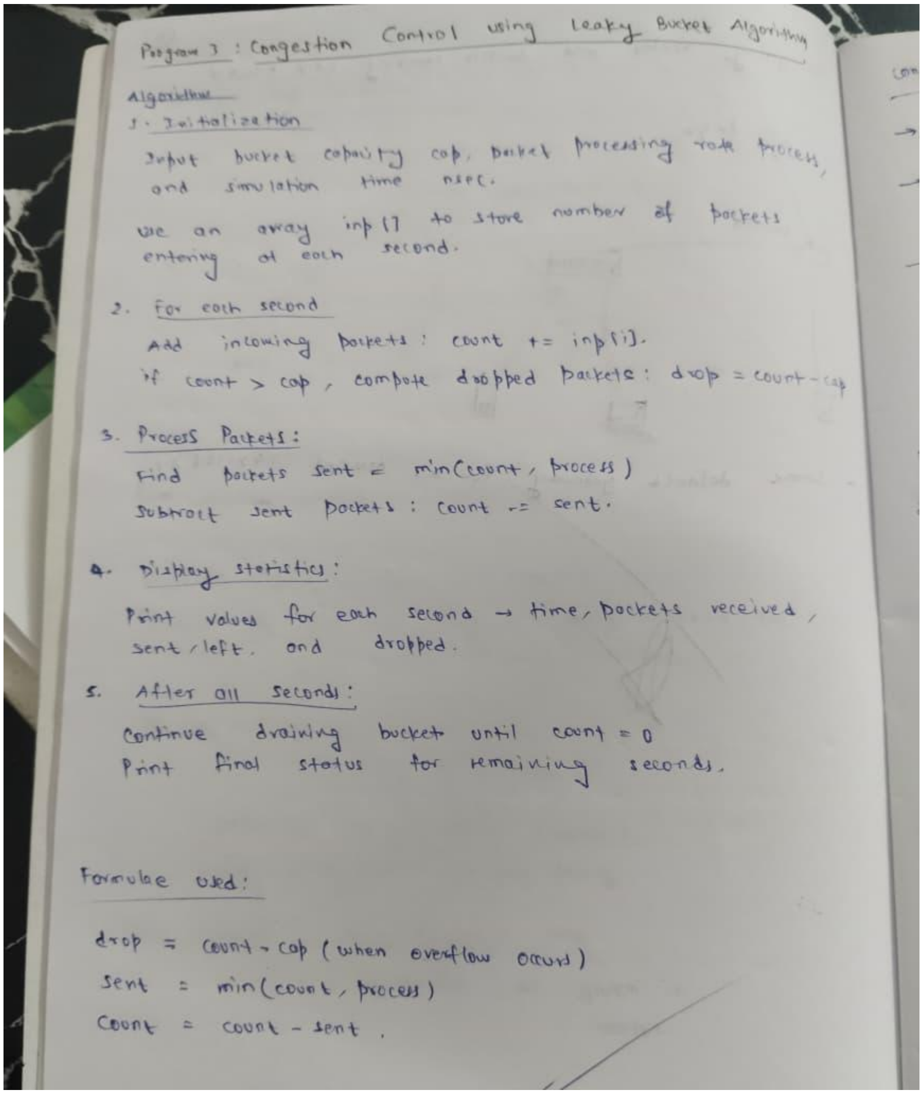
Program 11:

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

Network diagram:



Configuration:



Program 3 : Congestion Control using Leaky Bucket Algorithm

Algorithm

1. Initialization

Input bucket capacity cap, packet processing rate process, and simulation time nsec.

We an array inp [] to store number of packets entering at each second.

2. For each second

Add incoming packets : count += inp[i].
if count > cap, compute dropped packets : drop = count - cap

3. Process Packets :

Find packets sent = min(count, process)
Subtract sent packets : count -= sent.

4. Display statistics :

Print values for each second → time, packets received, sent / left, and dropped.

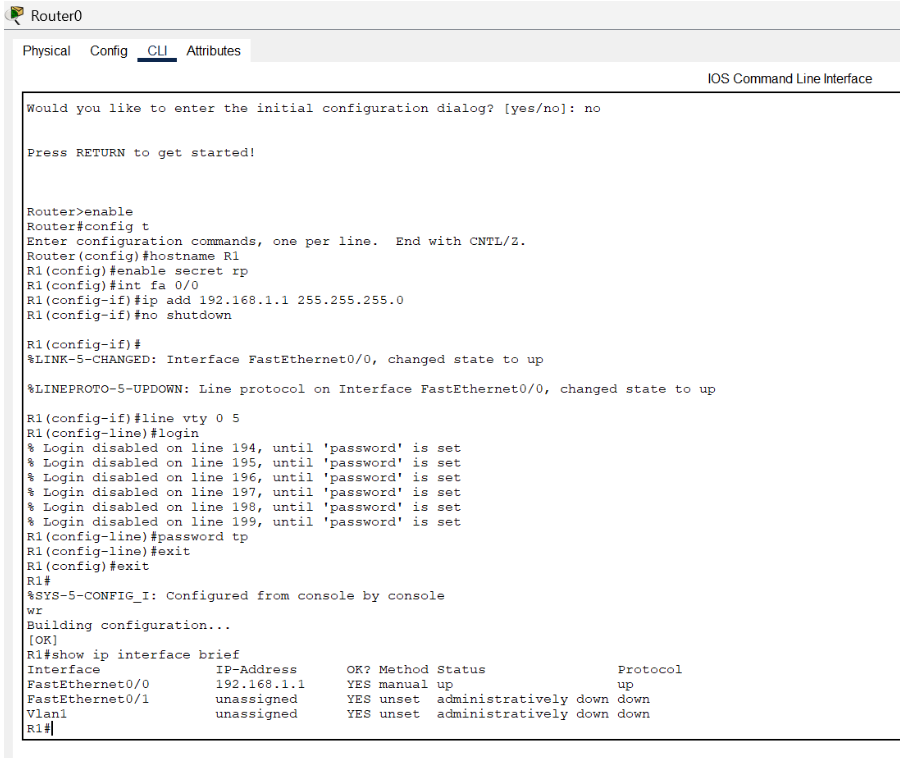
5. After all seconds :

Continue draining bucket until count = 0
Print final status for remaining seconds.

Formulae used :

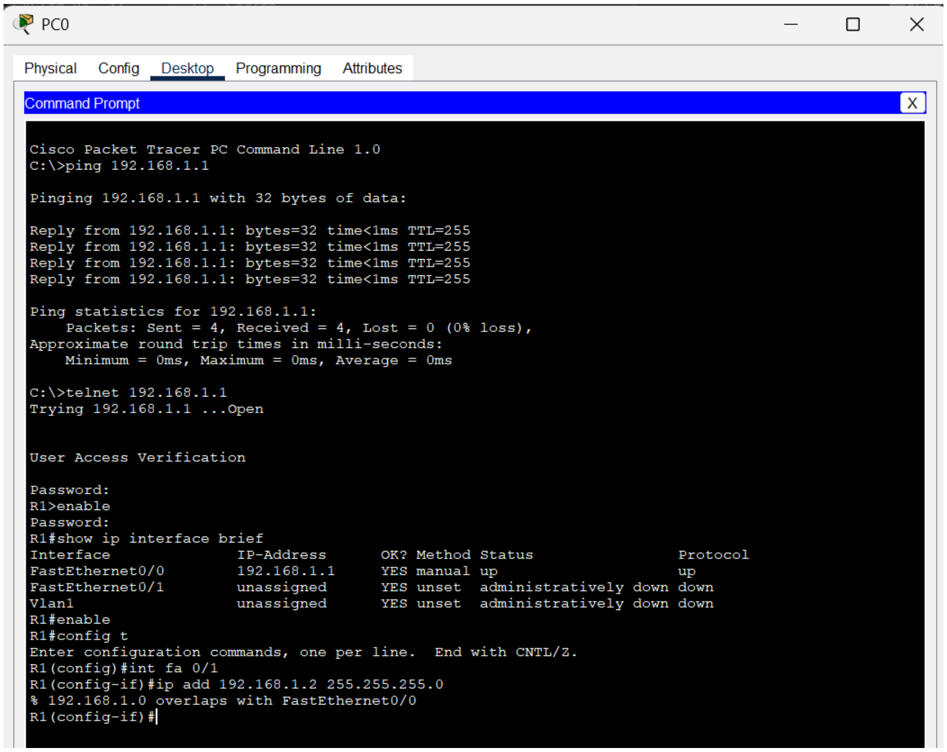
drop = count - cap (when overflow occurs)
sent = min(count, process)
count = count - sent .

Output:



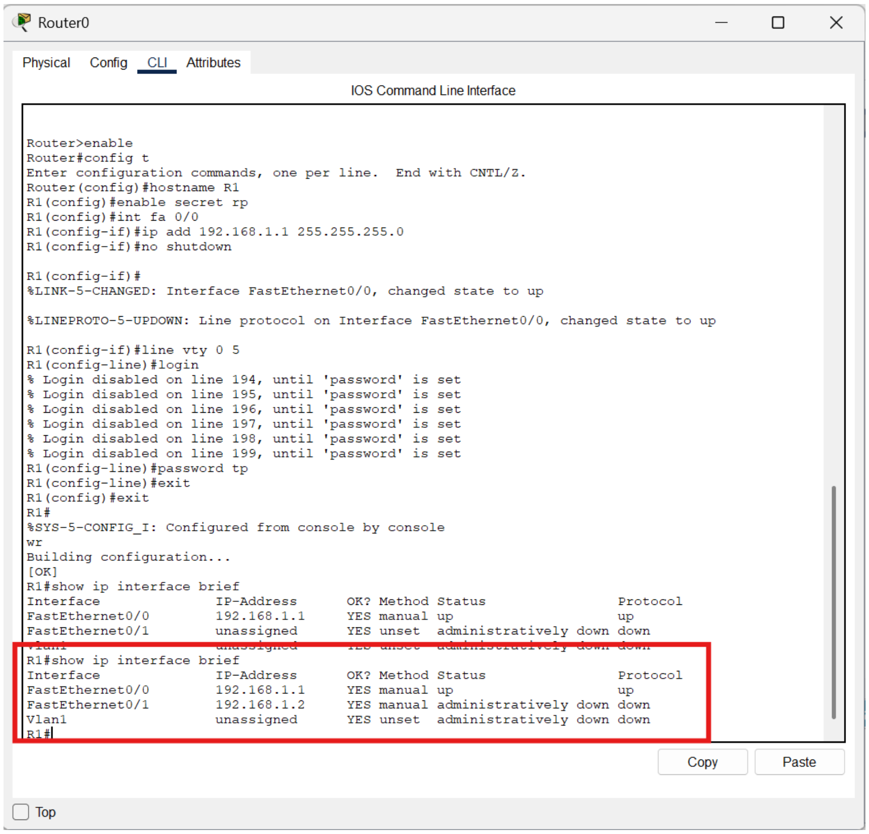
```
Would you like to enter the initial configuration dialog? [yes/no]: no


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



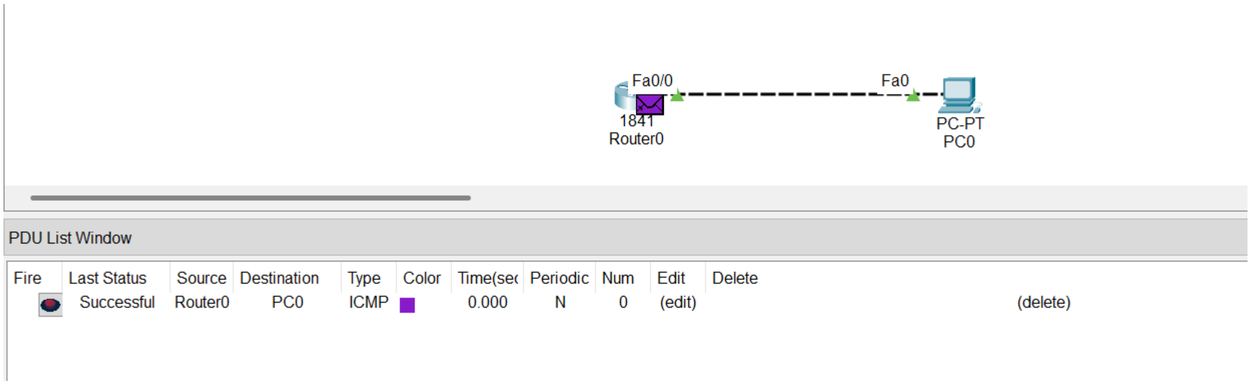
```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.1

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



```
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#show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        192.168.1.1     YES manual up                    up
FastEthernet0/1        192.168.1.2     YES manual administratively down down
Vlan1                  unassigned      YES unset  administratively down down
R1#
```

Fig 3. Updated the changes into Router0



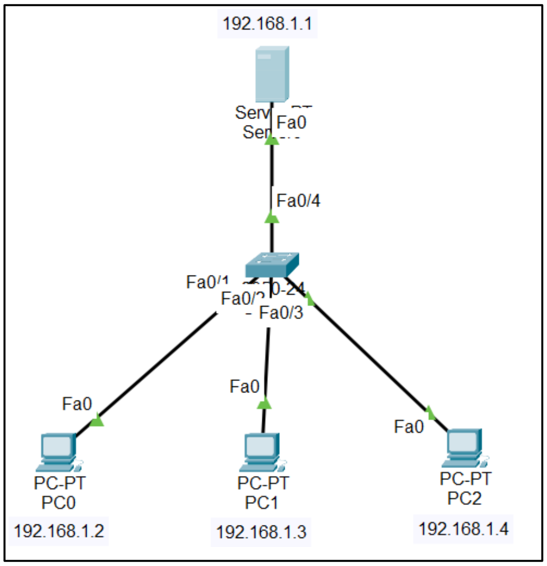
Fire	Last Status	Source	Destination	Type	Color	Time(sec	Periodic	Num	Edit	Delete
●	Successful	Router0	PC0	ICMP	🟪	0.000	N	0	(edit)	(delete)

Fig 4. PDU message Successful

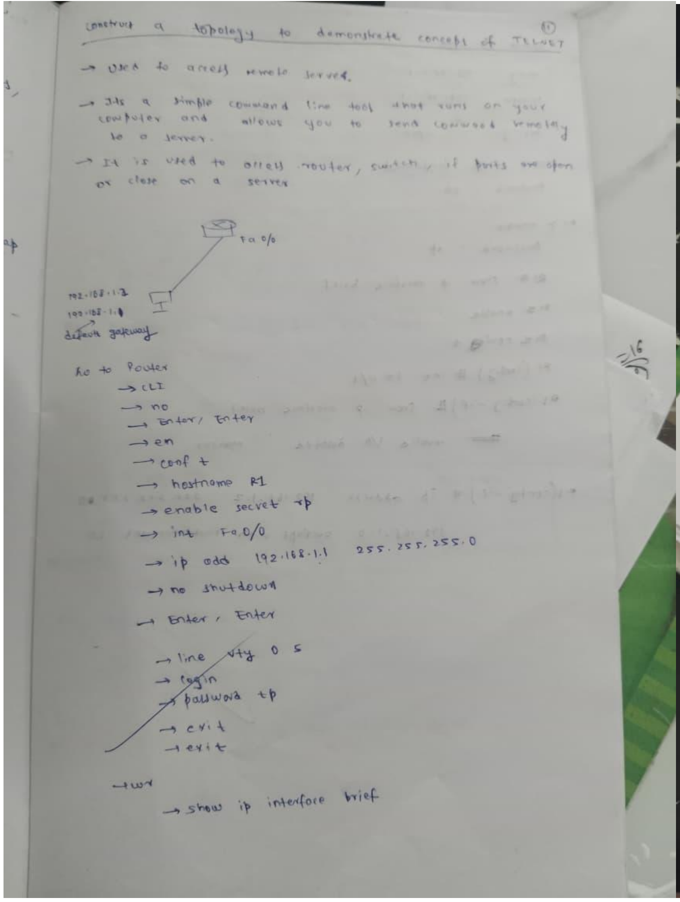
Program 12:

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

Network diagram:



Configuration:



Construct a topology to demonstrate concept of TELNET ①

→ Used to access remote server.

→ Its a simple command line tool that runs on your computer and allows you to send command remotely to a server.

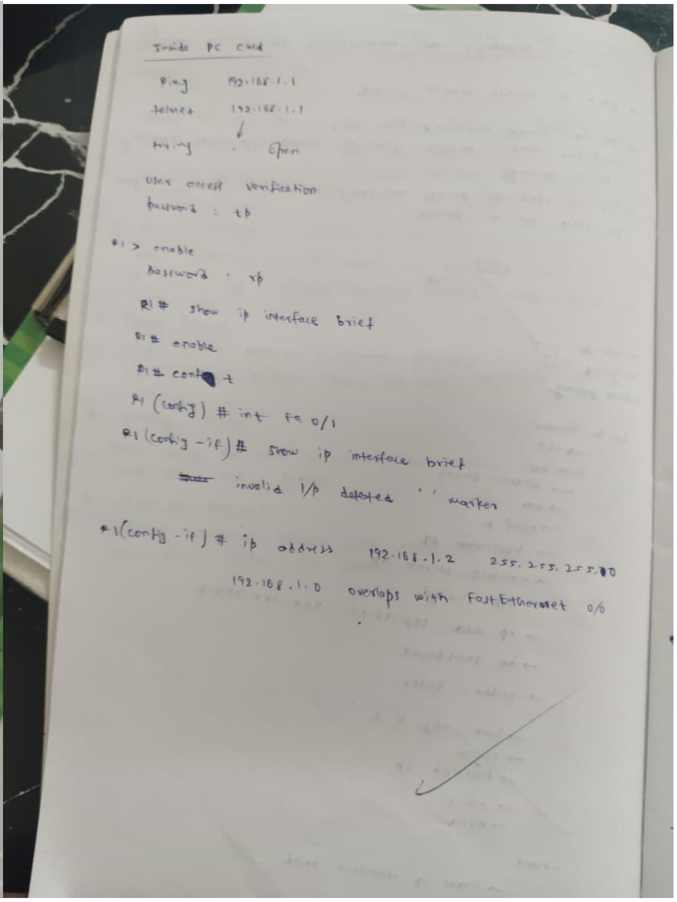
→ It is used to access router, switch, if ports are open or close on a server

Fa 0/0

192.168.1.2
192.168.1.1
default gateway

Go to Router

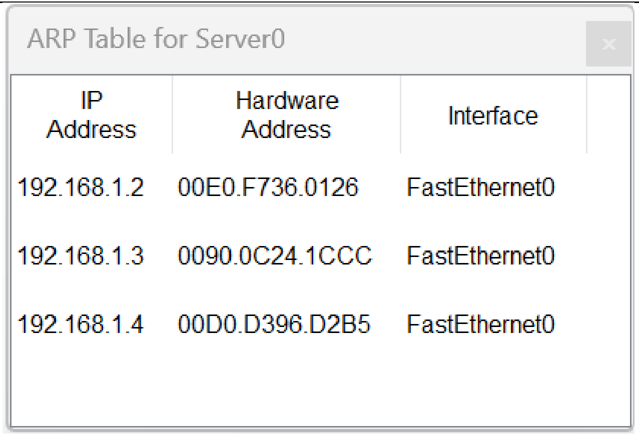
```
→ CLI
→ no
→ Enter / Enter
→ en
→ conf t
→ hostname R1
→ enable secret tp
→ int Fa0/0
→ ip add 192.168.1.1   255.255.255.0
→ no shutdown
→ Enter , Enter
  → line vty 0 5
  → login
  → password tp
  → exit
  → exit
→ wr
  → show ip interface brief
```



Inside PC cmd

```
ping     192.168.1.1
telnet   192.168.1.1
              ↓
trying  . . . Open

user access verification
password : tp

R1 > enable
password : tp

R1 # show ip interface brief
R1 # enable
R1 # conf t
R1 (config) # int Fa 0/1
R1 (config-if) # show ip interface brief
      invalid i/p detected ' ' marker

R1(config-if) # ip address   192.168.1.2    255.255.255.0
          192.168.1.0  overlaps with  FastEthernet 0/0
```

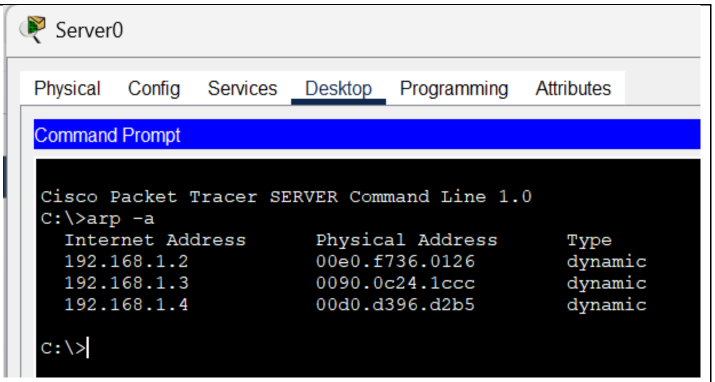
Output:



ARP Table for Server0

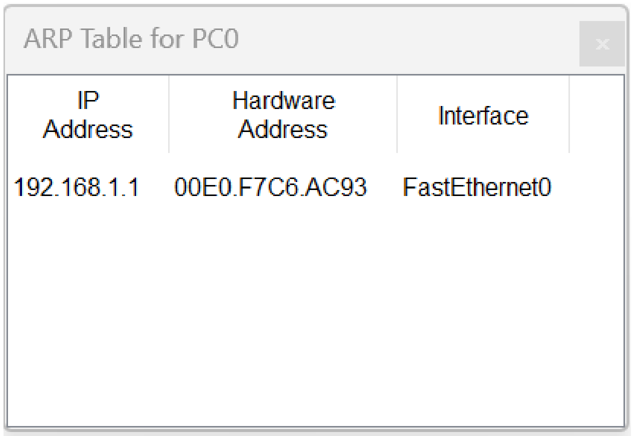
IP Address	Hardware Address	Interface
192.168.1.2	00E0.F736.0126	FastEthernet0
192.168.1.3	0090.0C24.1CCC	FastEthernet0
192.168.1.4	00D0.D396.D2B5	FastEthernet0

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:\>
```

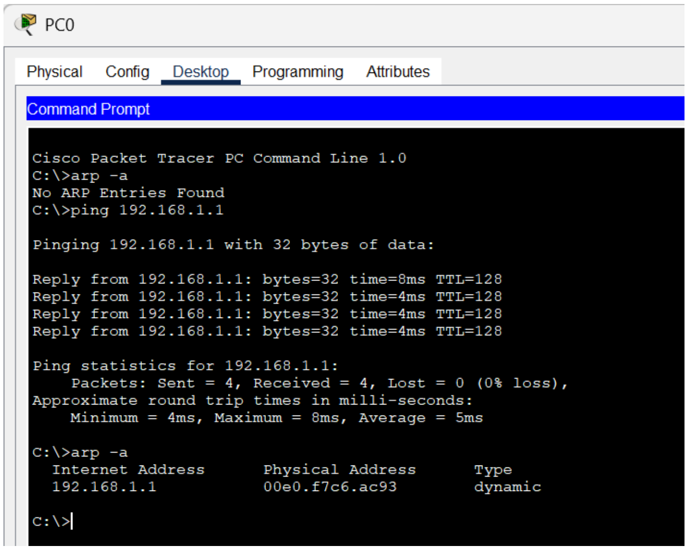
Fig 1.2 Command Prompt at Server0



ARP Table for PC0

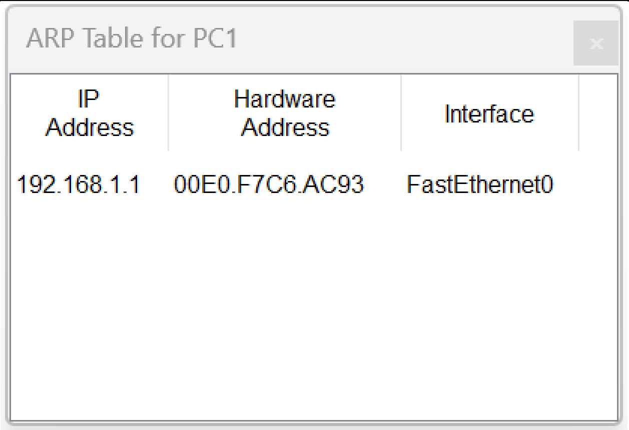
IP Address	Hardware Address	Interface
192.168.1.1	00E0.F7C6.AC93	FastEthernet0

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:\>
```

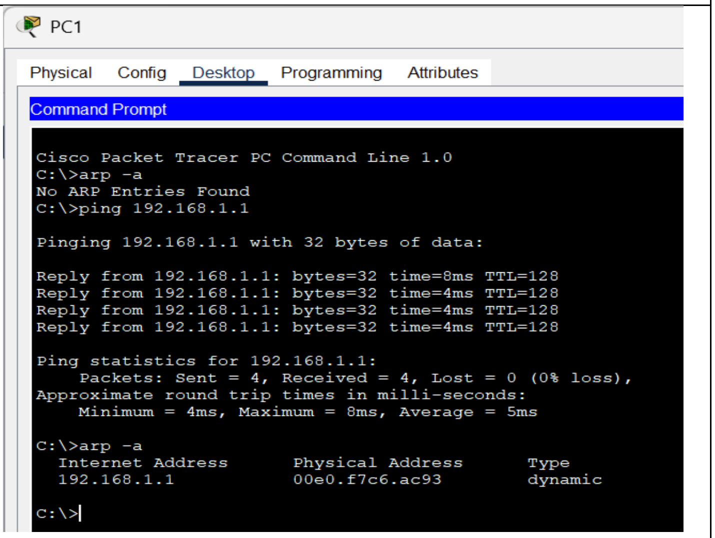
Fig 2.2 Command Prompt at PC0



ARP Table for PC1

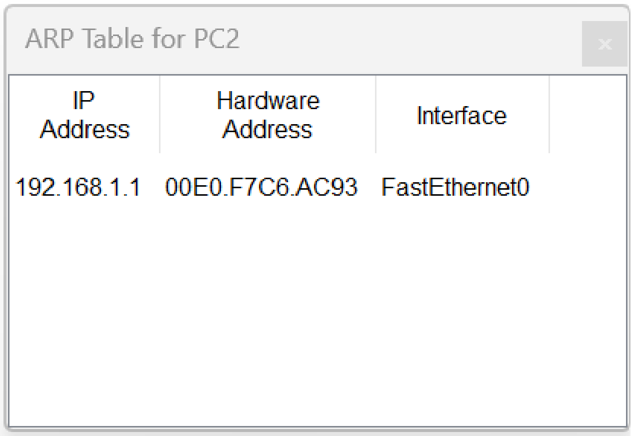
IP Address	Hardware Address	Interface
192.168.1.1	00E0.F7C6.AC93	FastEthernet0

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:\>
```

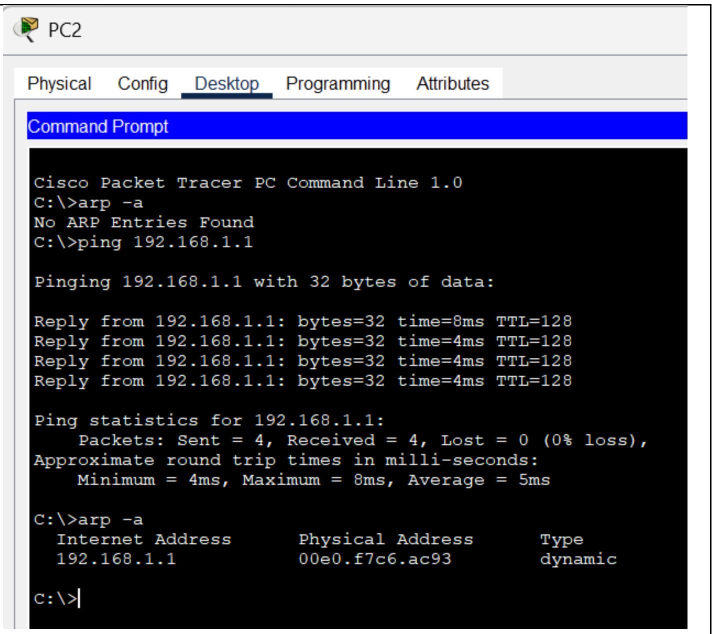
Fig 3.2 Command Prompt at PC1



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

<u>Program 1:</u>

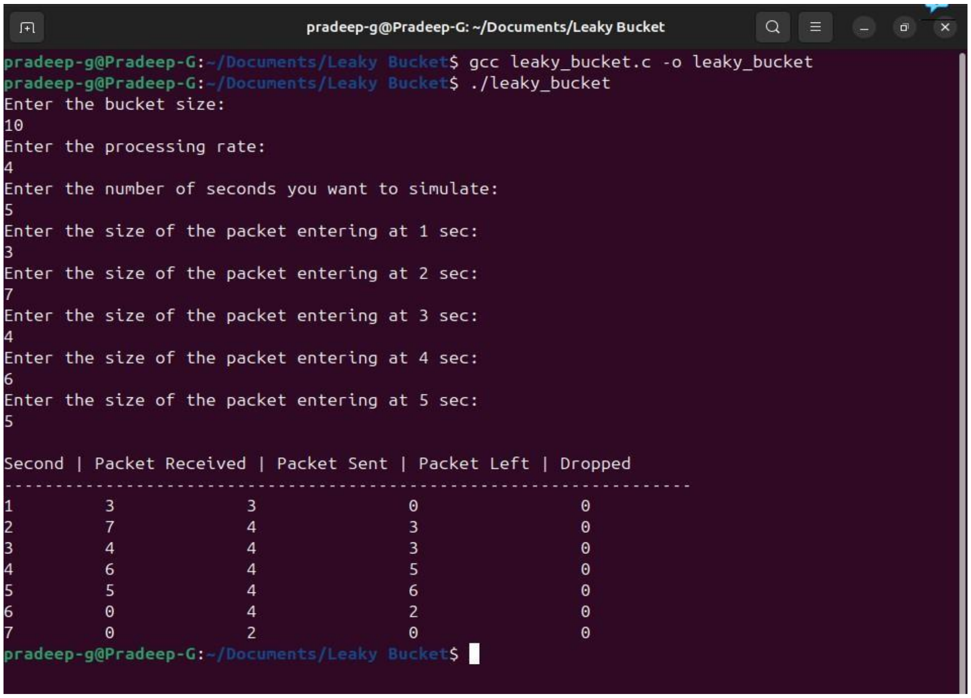
Aim: Write a program for congestion control using Leaky bucket algorithm.

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);
```

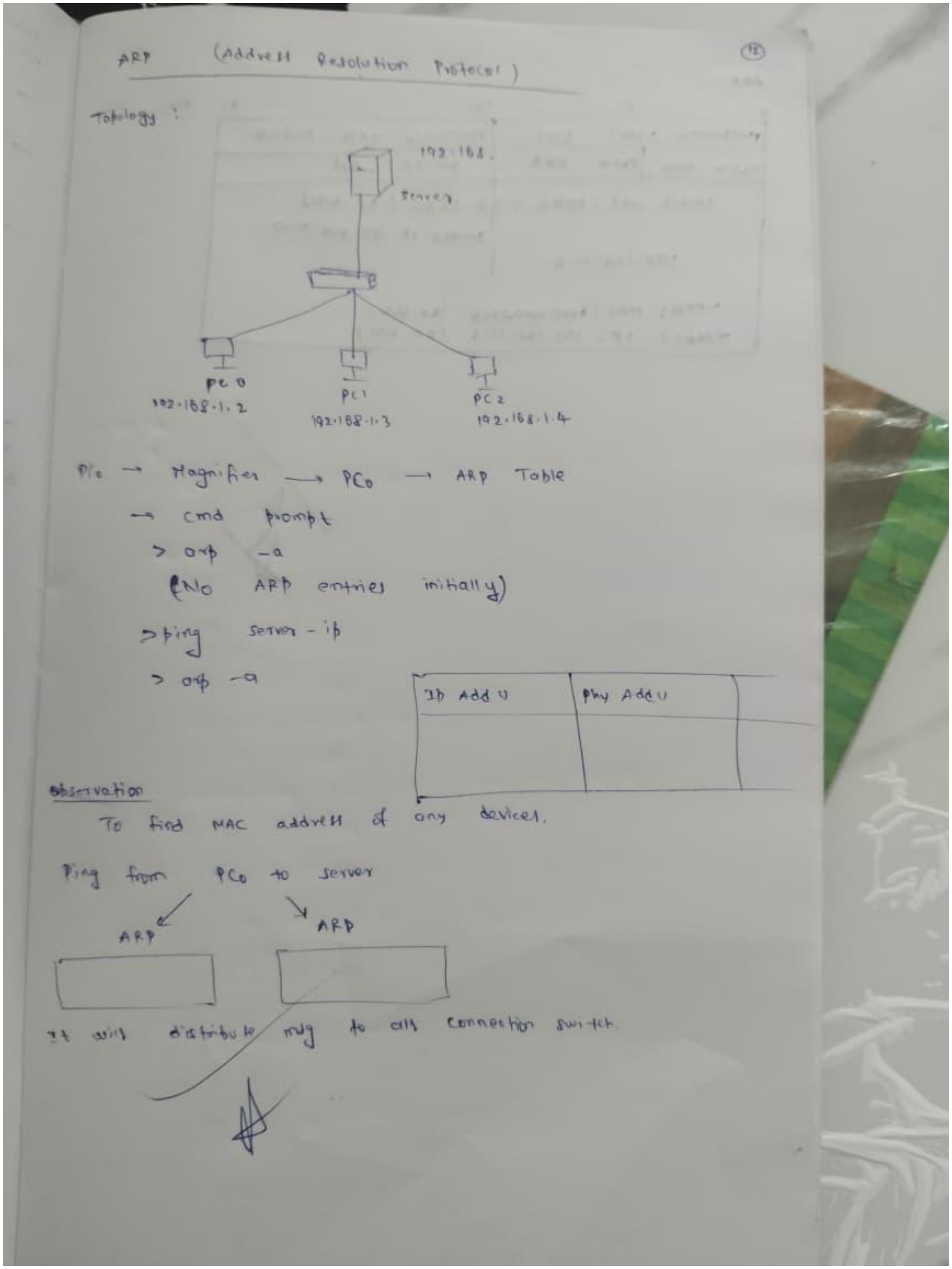
```
        scanf("%d", &inp[i]);
    }

    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$ 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:



ARP (Address Resolution Protocol) ⑫

Topology :

192.168.
Server
PC 0
192.168.1.2
PC1
192.168.1.3
PC 2
192.168.1.4

P/c → Magnifier → PC0 → ARP Table

→ cmd prompt

```
> arp -a
```

(No ARP entries initially)

```
> ping server - ip
> arp -a
```

Ip Addr	phy Addr	

Observation

To find MAC address of any devices.

Ping from PC0 to server

ARP ARP

It will distribute msg to all connection switch.

Program 2:

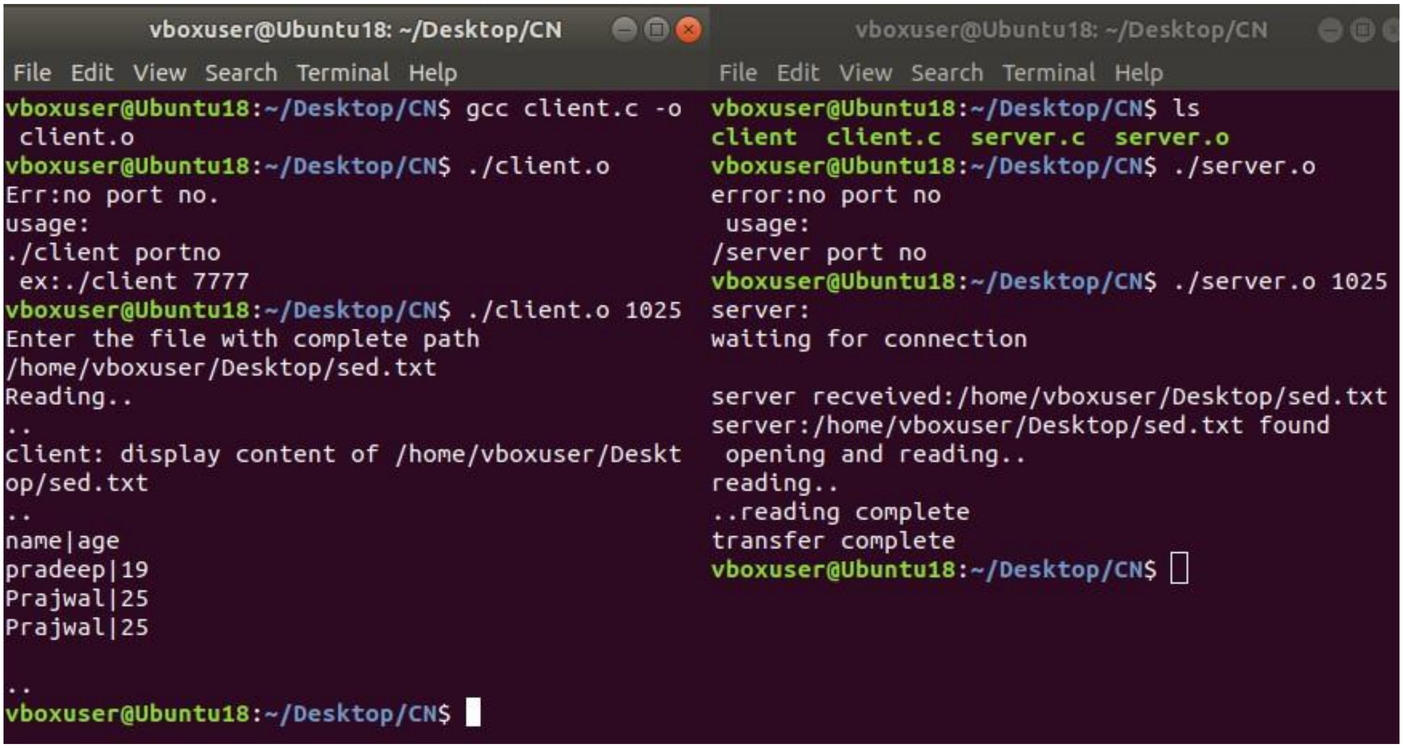
Aim: Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

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:

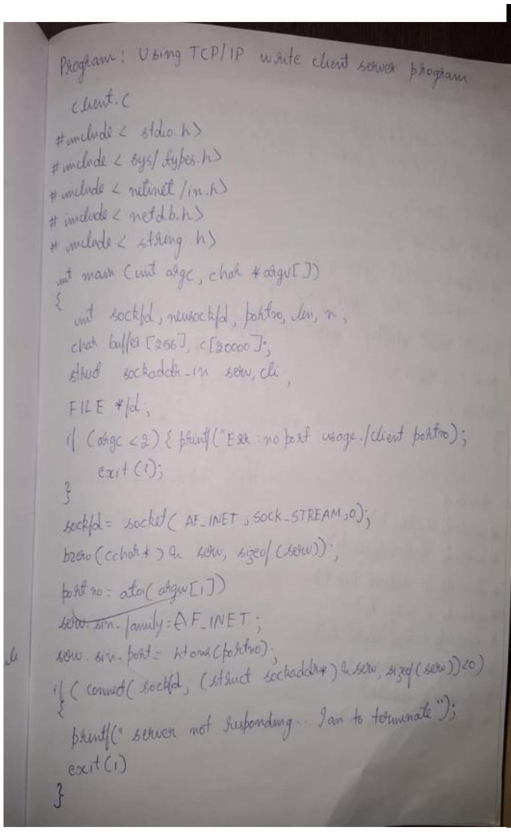


```
vboxuser@Ubuntu18:~/Desktop/CN$ gcc client.c -o
 client.o
vboxuser@Ubuntu18:~/Desktop/CN$ ./client.o
Err:no port no.
usage:
./client portno
 ex:./client 7777
vboxuser@Ubuntu18:~/Desktop/CN$ ./client.o 1025
Enter the file with complete path
/home/vboxuser/Desktop/sed.txt
Reading..
..
client: display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

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

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

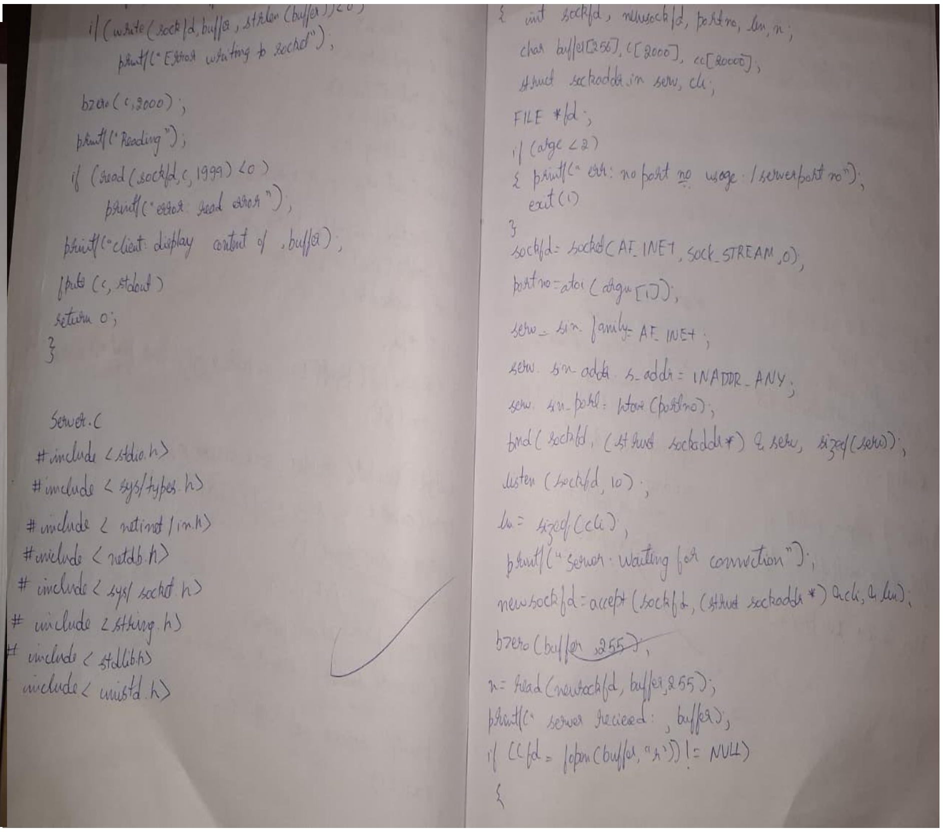
vboxuser@Ubuntu18: ~/Desktop/CN
File Edit View Search Terminal Help

Observation:



Program: Using TCP/IP write client server program

```
client.c
#include < stdio.h>
#include < sys/ types.h>
#include < netinet /in.h>
# include < netdb.h>
# include < string .h>
int main (int argc, char *argv[])
{
 int sockfd, newsockfd, portno, len, n ;
 char buffer [256], c[20000];
 struct sockaddr_in serv, cli ,
 FILE *fd ,
 if (argc <2) { printf("Err : no port usage./client portno);
     exit (1);
 }
 sockfd = socket( AF_INET , SOCK_STREAM ,0);
 bzero (cchar* ) & serv, sizeof (serv)) ,
 port no = atoi( argv[1])
 serv. sin_family = AF_INET ;
 serv . sin_port = htons (portno) ;
 if ( connect( sockfd, (struct sockaddr* ) & serv, sizeof (serv))<0)
 {
  printf(" server not responding . . I am to terminate");
  exit (1)
 }
```



```
 if (write ( sockfd, buffer , strlen (buffer))<0)
     printf(" Error writing to socket"),

  bzero ( c,2000) ;
  printf(" Reading");
  if (read ( sockfd,c, 1999) <0 )
      printf(" error read error"),
  printf("client display content of , buffer);
  fputs (c, stdout )
  return 0;
}


Server.c
#include <stdio.h>
#include < sys/types. h>
#include < netinet / in.h>
#include < netdb.h>
# include < sys/ socket h>
# include < string. h>
# include < stdlib.h>
include < unistd. h>
```

```
{ int sockfd, newsockfd, portno, len, n ;
  char buffer[256], c[2000], cc[20000],
  struct sockaddr.in serv, cli ;
  FILE *fd ;
  if (argc <2)
  { printf(" err: no port no usage . /serverport no");
    exit (1)
  }
  sockfd= socket(AF_INET , SOCK_STREAM ,0),
  portno=atoi (argv [1]),
  serv_ sin. family= AF_INET ;
  serv. sin_addr. s_addr = INADDR_ANY ;
  serv. sin_port : htons (portno);
  bind ( sockfd, (struct sockaddr*) & serv, sizeof (serv)),
  listen (sockfd, 10) ;
  len = sizeof (cli) ,
  printf(" server · waiting for connection");
  newsockfd=accept (sockfd, (struct sockaddr *) &cli, & len);
  bzero (buffer ,255),
  n= read (newsockfd, buffer,255);
  printf(" server recieved: , buffer),
  if ((fd = fopen (buffer, "r")) != NULL)
  {
```

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.

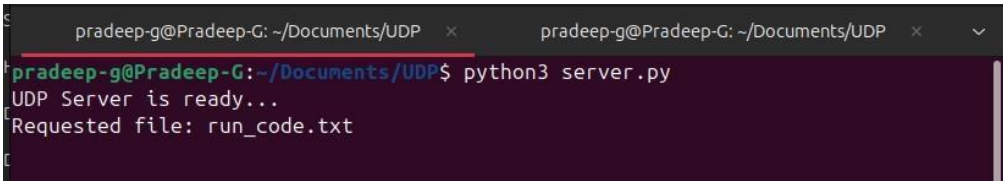
Code:

```
# 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)
```

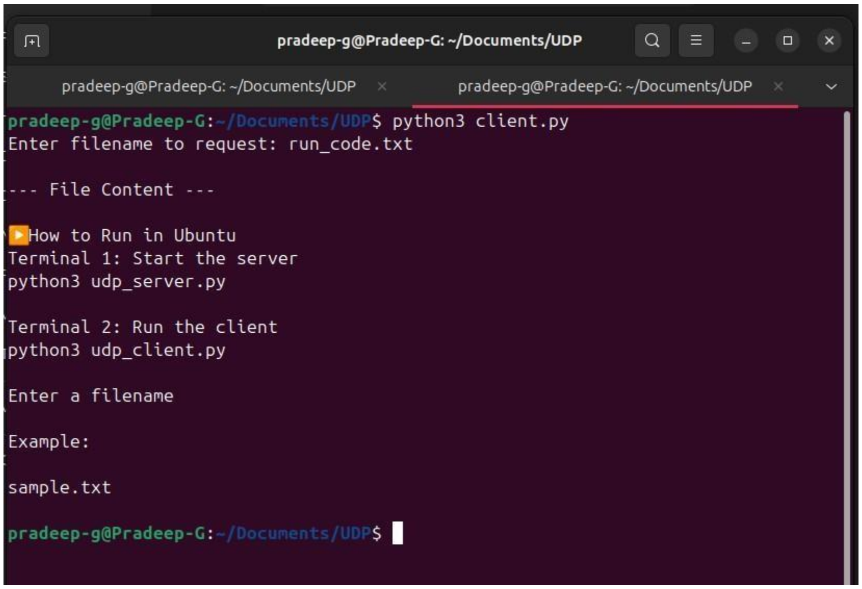
Output:

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:

Program 4:

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

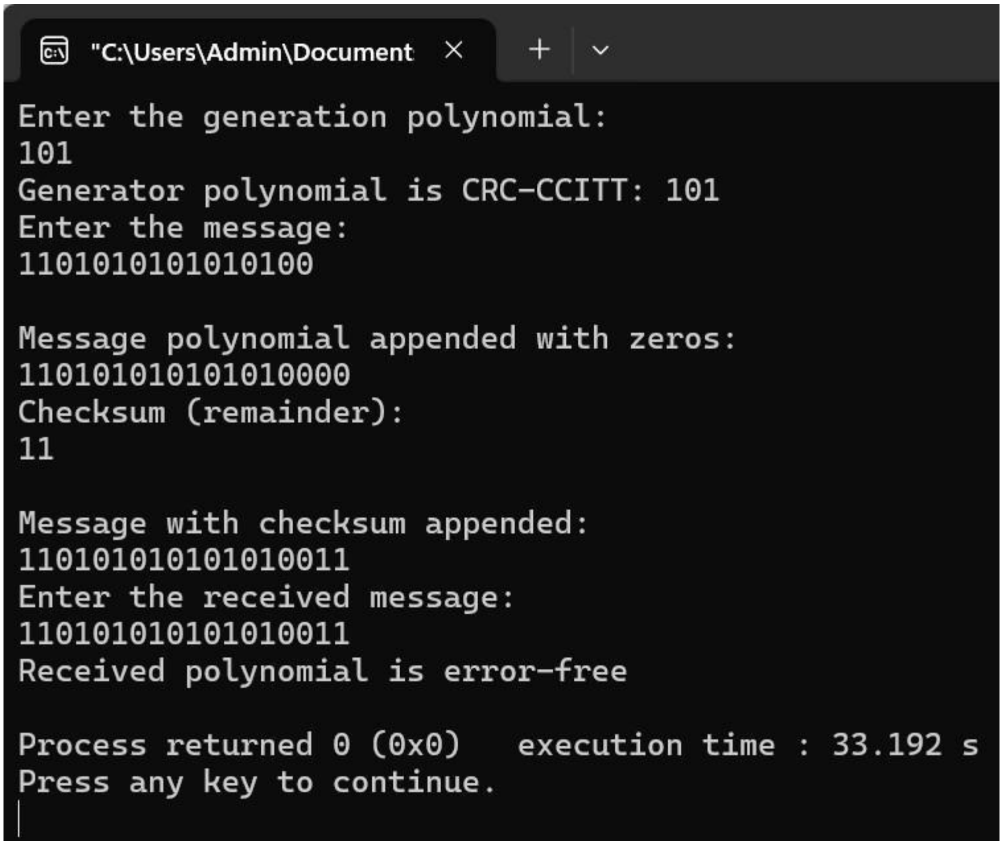
Code:

```
#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];
```

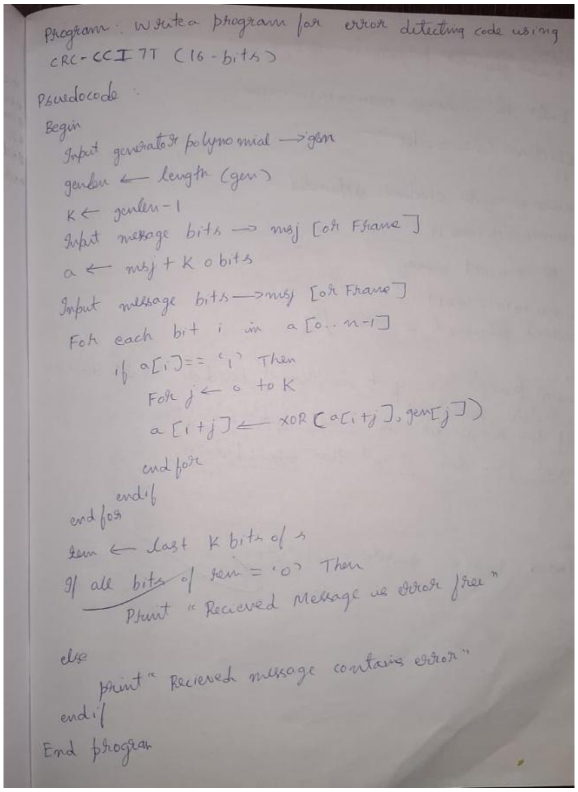
```
for (i = 0; i < k; i++)
    a[n + i] = rem[i];
a[n + k] = '\0';
puts(a);

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:



```
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.
```

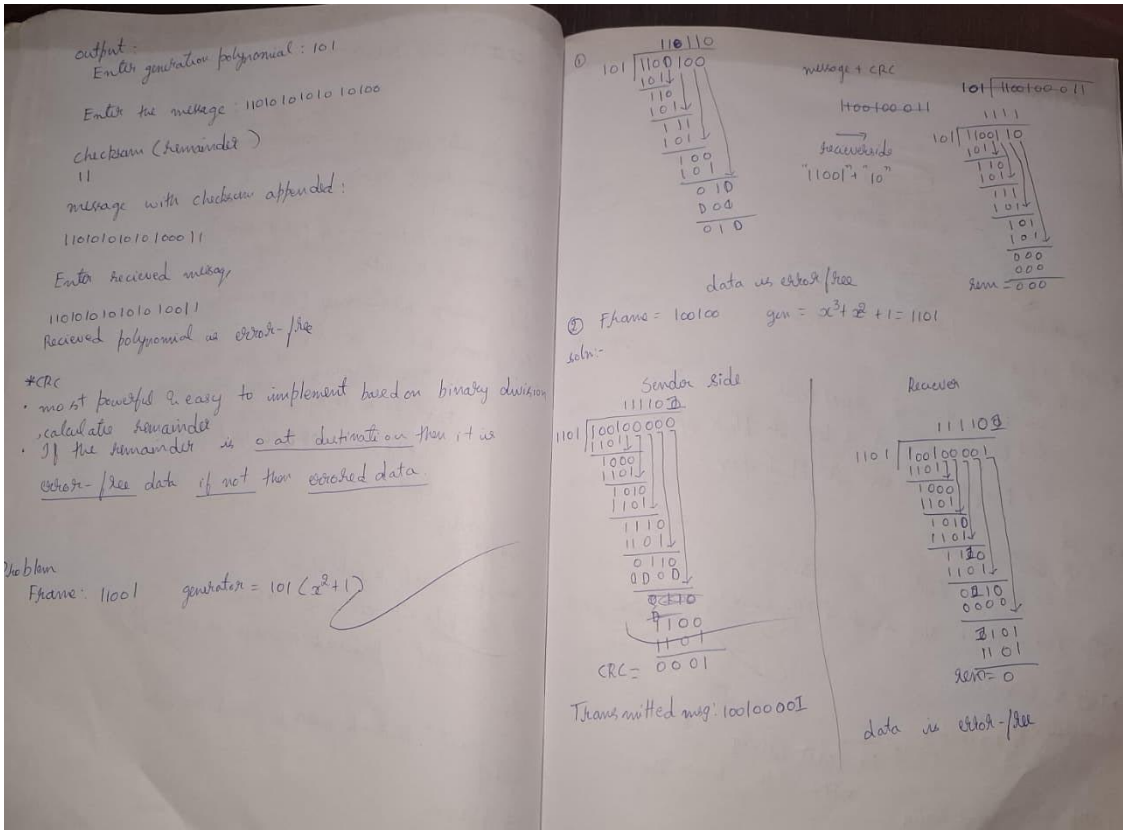
Observation:



Program : Write a program for error detecting code using CRC-CCI7T (16-bits)

Pseudocode :

```
Begin
  Input generator polynomial —> gen
  genlen ← length (gen)
  K ← genlen-1
  Input message bits —> msj [or Frame]
  a ← msj + K o bits
  Input message bits —> msj [or Frame]
  For each bit i in a [0.. n-1]
        if a[i]== '1' Then
            For j ← o to K
            a [i+j] ← XOR (a[i+j], gen[j])
            end for
        endif
  end for
  rem ← last K bits of s
  If all bits of rem = '0' Then
        Print "Recieved Message is error free"
  else
        print "Recieved message contains error"
  endif
End program
```



output :

```
Enter generation polynomial : 101

Enter the message 11010101010 10100

checksum (remainder )
11
message with checksum appended :

1101010101010 0011

Enter recieved message

11010101010 10 10011
Recieved polynomial is error-free
```

*CRC

- most powerful & easy to implement based on binary division, calculate remainder
- If the remainder is o at destination then it is error-free data if not then errored data

Problem
Frame: 11001   generator = 101 (x^2+1)

①

```
         110110
101 ⟌ 1100100
      101↓
       110
       101↓
        111
        101↓
         100
         101
          010
          000
          010
```

message + CRC

~~1100100 011~~

$\longrightarrow$ reciever side

"11001" + "10"

```
101 ⟌ ~~1100100 011~~

          1111
101 ⟌ 1100110
      101↓
       110
       101↓
        111
        101↓
         101
         101↓
          000
          000
rem =     000
```

data is error free

② Frame = 100100   gen = x^3+x^2+1 = 1101

soln:-

Sender side

```
          111101
1101 ⟌ 100100000
       1101↓
        1000
        1101↓
         1010
         1101↓
          1110
          1101↓
           0110
           0000↓
            ~~0110~~
            1100
            1101
CRC =       0001
```

Transmitted msg: 100100001

Reciever

```
          111101
1101 ⟌ 100100001
       1101↓
        1000
        1101↓
         1010
         1101↓
          1110
          1101↓
           0110
           0000
            1101
            1101
          rem = 0
```

data is error-free