# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

**"JnanaSangama", Belgaum -590014, Karnataka.**

**LAB RECORD**

# Computer Network Lab (23CS5PCCON)

***Submitted by***

**SOHAN T SANJEEV (1BM23CS421)**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**

**B.M.S. COLLEGE OF ENGINEERING**

**(Autonomous Institution under VTU)**

**BENGALURU-560019**

**Academic Year 2024-25 (odd)**

# 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 **SOHAN T SANJEEV (1BM23CS421),** 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.

| | |
|---|---|
| Sneha P<br>Assistant Professor<br>Department of CSE, BMSCE | Dr. Kavitha Sooda<br>Professor & HOD<br>Department of CSE, BMSCE |

# Index



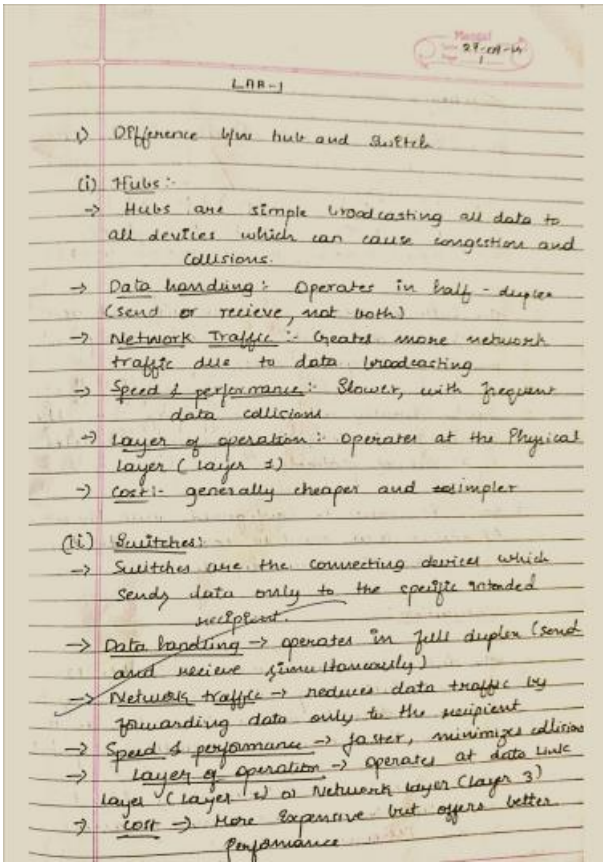
Github Link:

https://github.com/SohanTbmsce/5E----CN

**Program 1**

i. Aim of the program: To create a network topology using a hub and a switch, simulate PDU transmission, and demonstrate a ping message from source to destination.

ii. Procedure along with the topology:



LAB-1

1) Difference b/w hub and switch

(i) Hubs:-

-> Hubs are simple broadcasting all data to all devices which can cause congestion and collisions.

-> Data handling: Operates in half-duplex (send or recieve, not both)

-> Network Traffic :- Creates more network traffic due to data broadcasting

-> Speed & performance: Slower, with frequent data collisions

-> Layer of operation: operates at the Physical layer (layer 1)

-> cost:- generally cheaper and simpler

(ii) Switches:

-> Switches are the connecting devices which sends data only to the specific intended recipient.

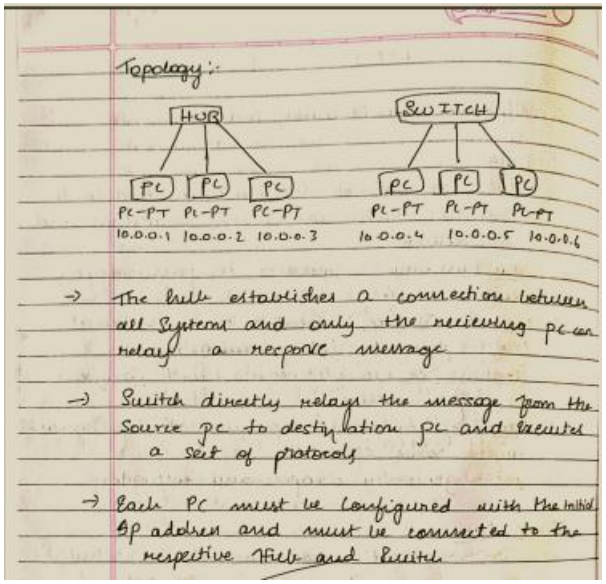
-> Data handling -> operates in full duplex (send and recieve simultaneously)

-> Network traffic -> reduces data traffic by forwarding data only to the recipient

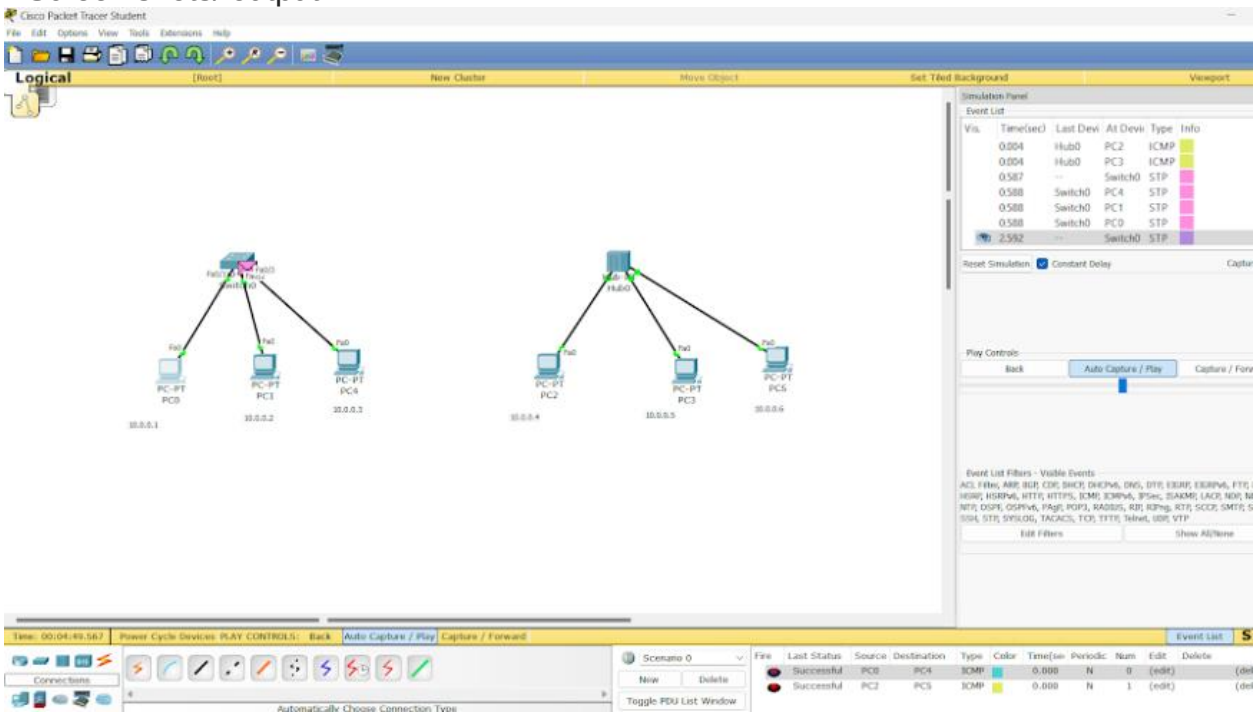
-> Speed & performance -> faster, minimizes collision

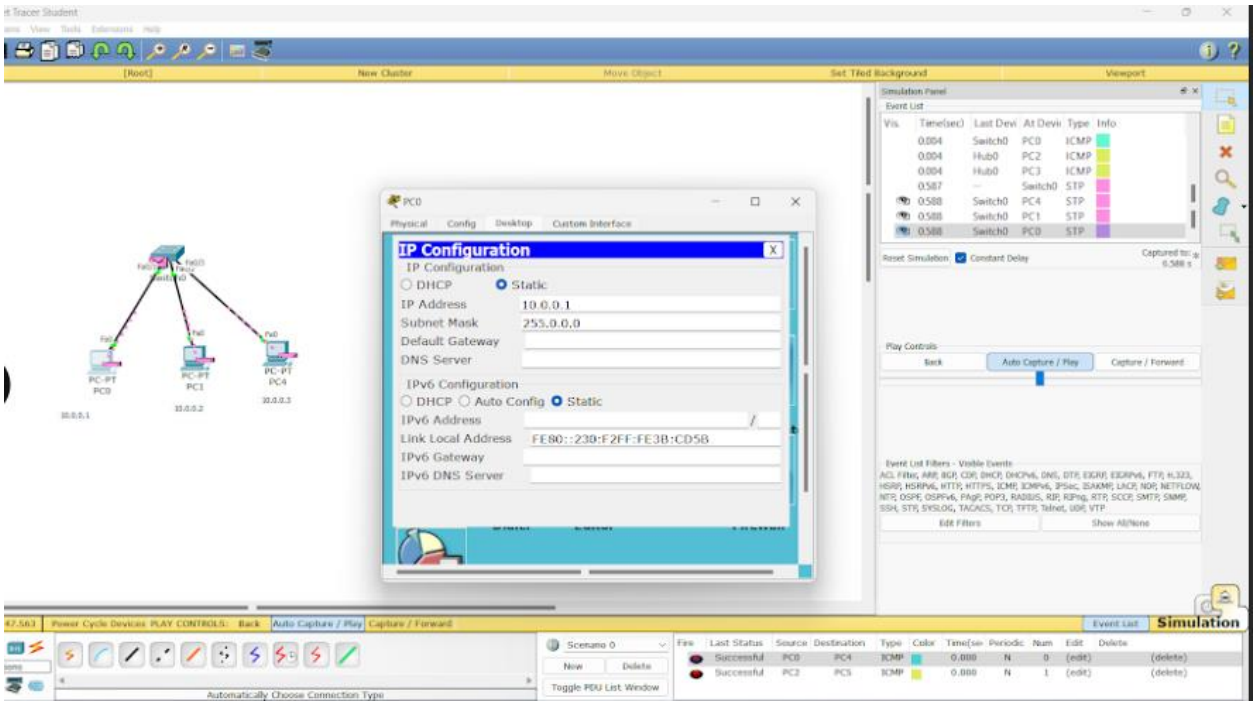
-> Layer of operation -> operates at data link layer (layer 2) or Network layer (layer 3)

-> cost -> More expensive but offers better performance

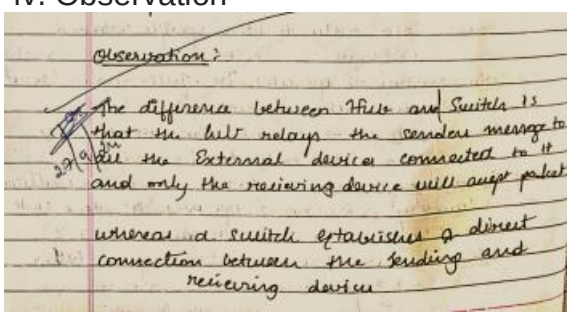



iii. Screen shots/ output:






iv. Observation



Observation:

The difference between Hub and Switch is that the hub relays the sender message to all the External devices connected to it and only the recieving device will accept packet

whereas a switch establishes a direct connection between the sending and recieving device

**Program 2**

i. Aim of the program: To create a topology involving multiple hubs and a switch

ii. Procedure along with the topology:

Lab-2.

Create a topology involving multiple hubs and a switch connecting them to a simulator to simulate PDU [packet Data Unit].

Procedure for creating the topology :-

Aim:- topology involving multiple hubs and a switch.

Step 1:- Select the hubs based on the requirement along with the required End devices [Here pc's] to be connected with the hub.

Step 2:- Select the type of connection [Here, automatically choosen connection type] that is used to connect between Hub and the End devices.

Step 3:- Configure every End device by navigating through Config -> FastEthernet0, after this, provide the ip address which is unique ~~the~~ for each End device.

Step 4:- After Step 3, select a Switch [based on the requirement] and select the type of link [Here, Copper Crossover] that needs to be used to connect between Multiple hubs and the Switch.

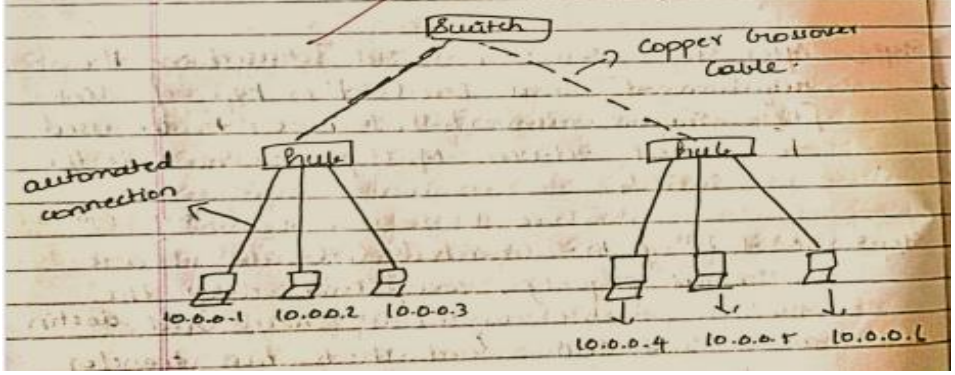
Step 5:- To check if the connected End devices (to a single hub) are configured properly, drag and drop the message on the required Source and destination End device and check the transfer through Simulation mode

Step 6:- Similar to step 5, now drag and drop the message b/w the source End device [connected to one hub] and the destination [connected to another hub] and check in simulation mode if packet transfer happens between the devices connected to two or more different hubs.

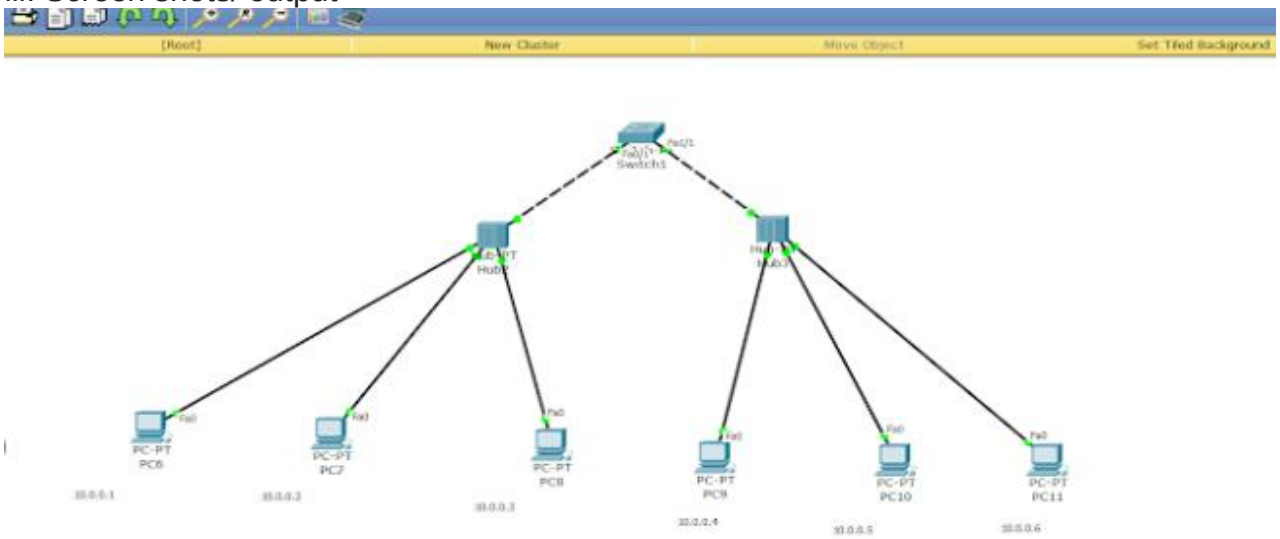
Step 7:- Now, select the desired End device, Right click on it, go to desktop and select command prompt there, we need to use "Ping" command to send single datagrams per second to respective destination device.

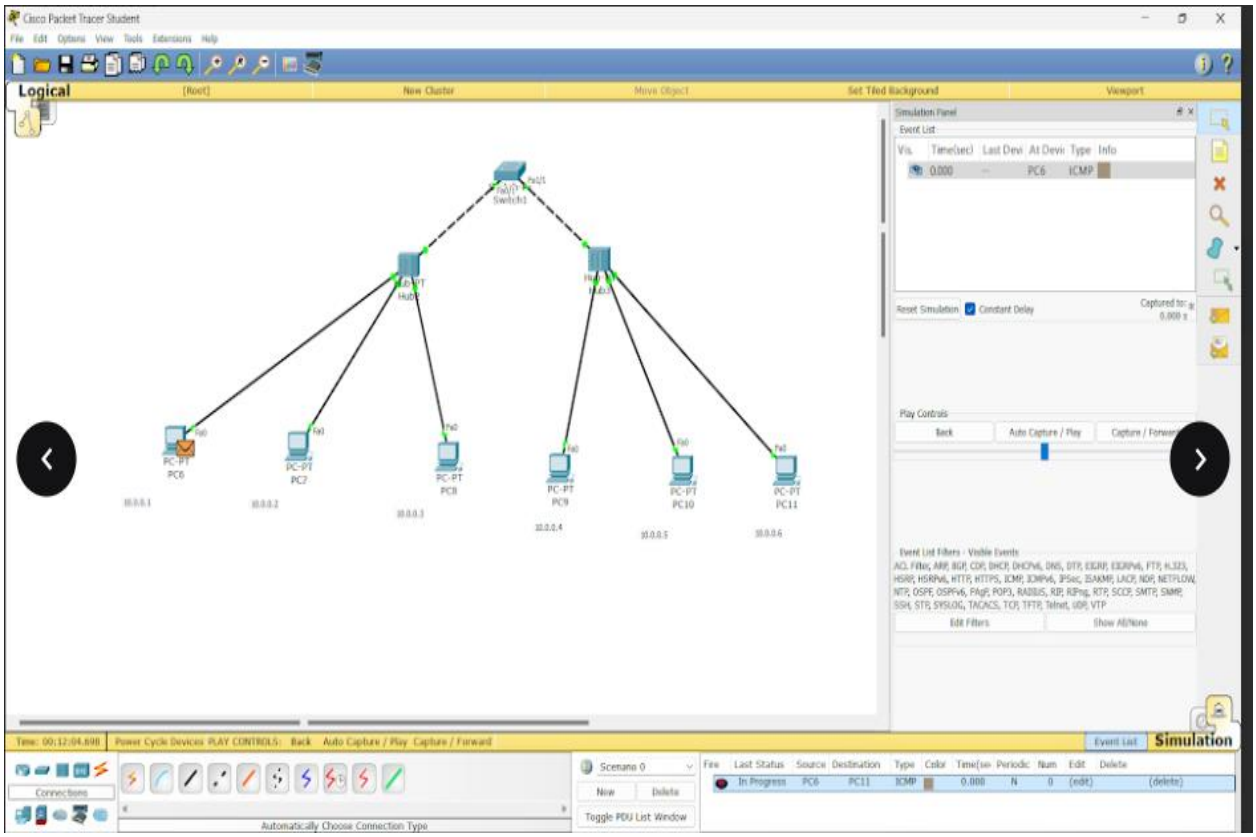
Step 8:- For example, select a configured device go to command prompt & type "ping 10.0.0.3" where 10.0.0.3 is the ip address configured for another end device and check the packet transfer process.

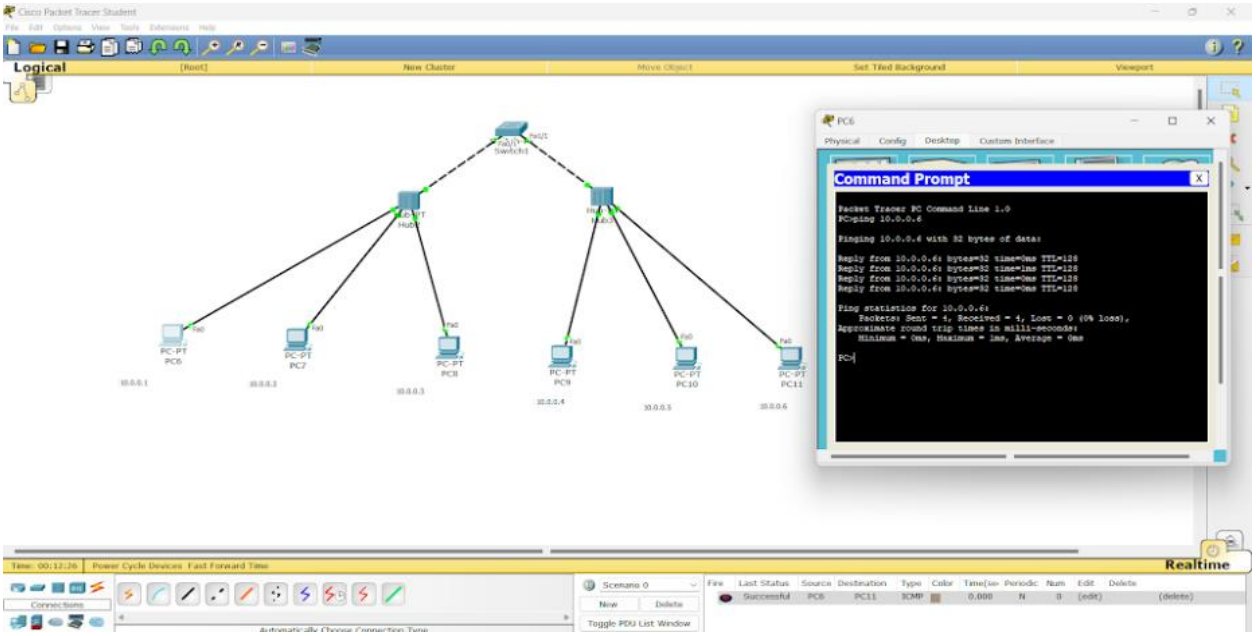
Topology:-



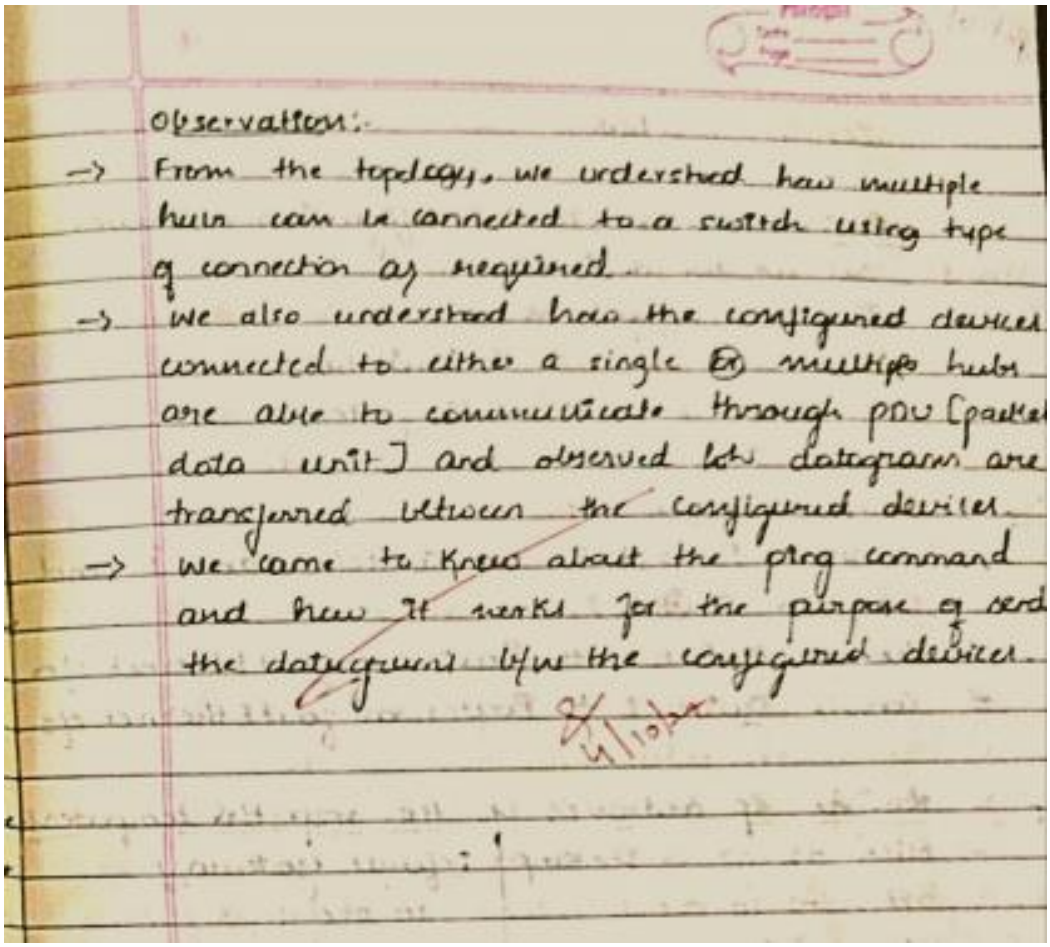
iii. Screen shots/ output




Cisco Packet Tracer Student
Logical
Realtime
PC6
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.6
Pinging 10.0.0.6 with 32 bytes of data:
Reply from 10.0.0.6: bytes=32 time=0ms TTL=128
Reply from 10.0.0.6: bytes=32 time=1ms TTL=128
Reply from 10.0.0.6: bytes=32 time=0ms TTL=128
Reply from 10.0.0.6: bytes=32 time=0ms TTL=128
Ping statistics for 10.0.0.6:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 1ms, Average = 0ms
PC>
PC-PT
PC6
PC7
PC8
PC9
PC10
PC11

iv. Observation



Observation:-

-> From the topology, we understood how multiple hubs can be connected to a switch using type of connection as required.

-> We also understood how the configured devices connected to either a single (Ex) multiple hubs are able to communicate through PDU [packet data unit] and observed how datagrams are transferred between the configured devices.

-> We came to know about the ping command and how it works for the purpose of send the datagrams b/w the configured devices.

**Program 3**

i. Aim of the program: Connecting and Configuring a Router

ii. Procedure along with the topology

Lab-3

Aim:- Connecting and Configuring a Router

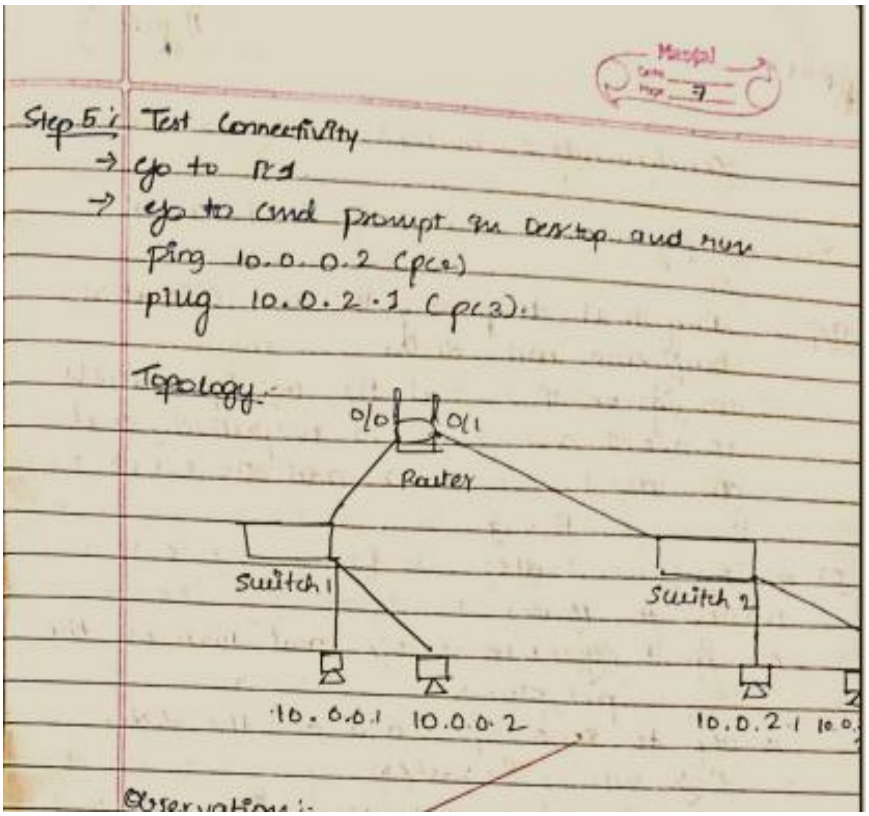
Step 1:- Set up the devices:
Drag and drop :- -> 1 Router
-> 2 Switches
-> 4 Computers.

Step 2:- Connect Devices :-
-> Connect $P_1$ and $P_2$ to Switch 1 and $P_3$ and $P_4$ to Switch 2.
-> Connect Switch 1 to Router on fastEthernet 0/0
-> Connect Switch 2 to Router on fastEthernet 0/1

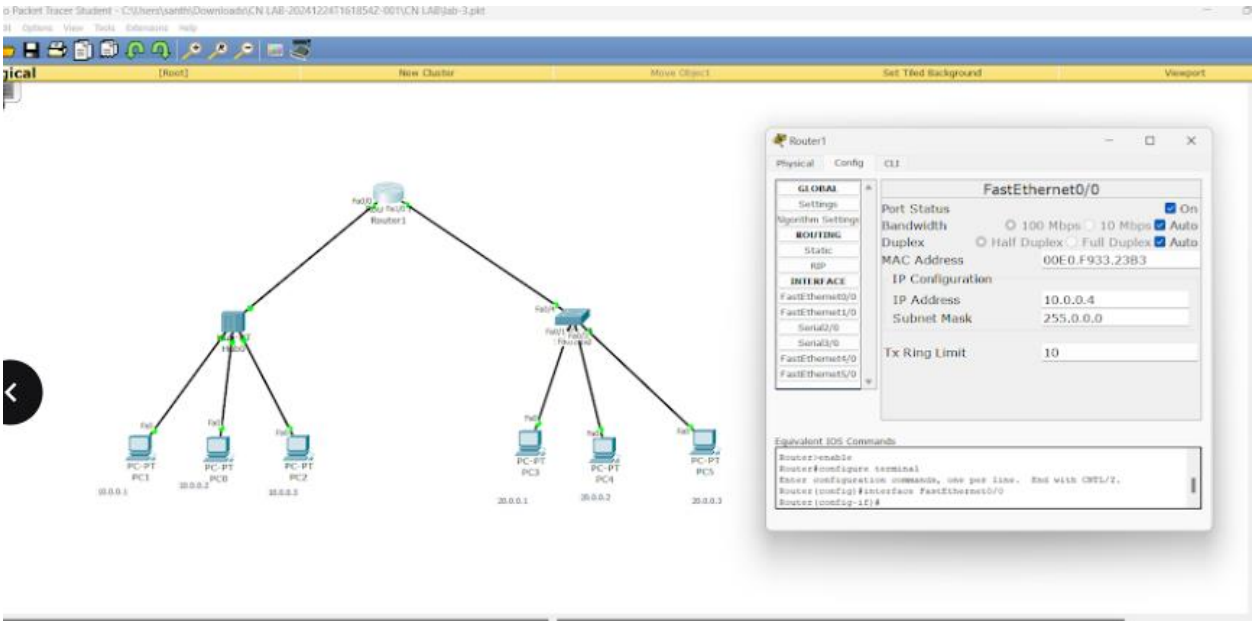
Step 3:- Assign ip addresses to the respective computers
-> click on pc -> Desktop

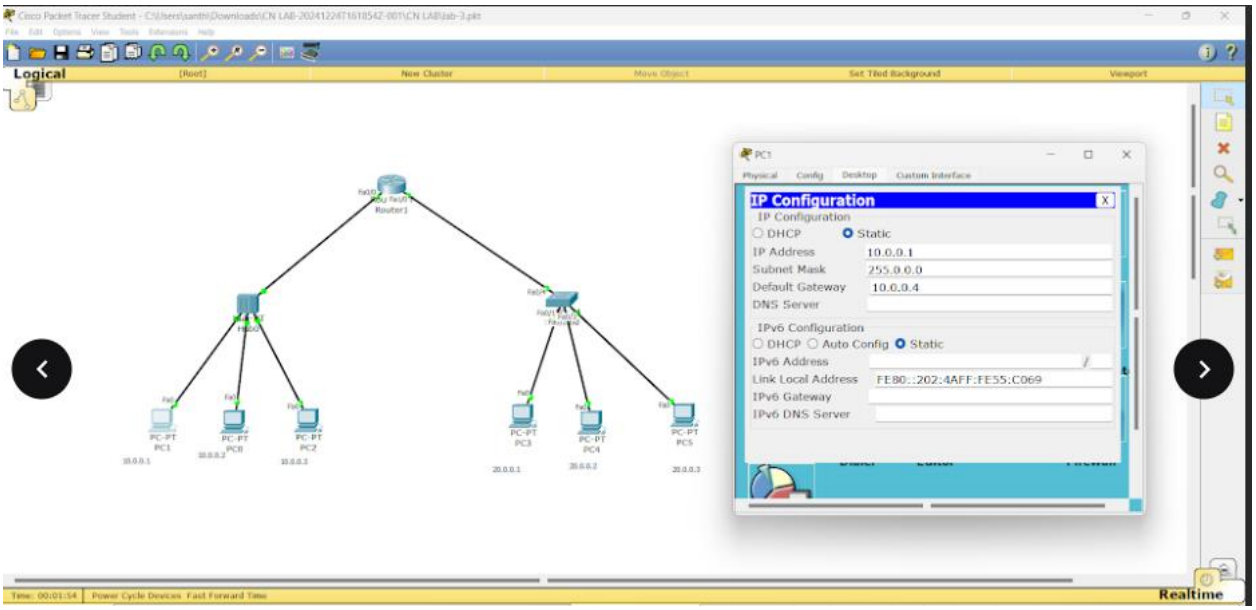
| | Defaulter Gateway |
|---|---|
| PC1 -> 10.0.0.1 | 10.0.0.1 |
| PC2 -> 10.0.0.2 | 10.0.0.1 |
| PC3 -> 10.0.0.3 | 10.0.0.2 |
| PC4 -> 10.0.0.3 | 10.0.0.2 |

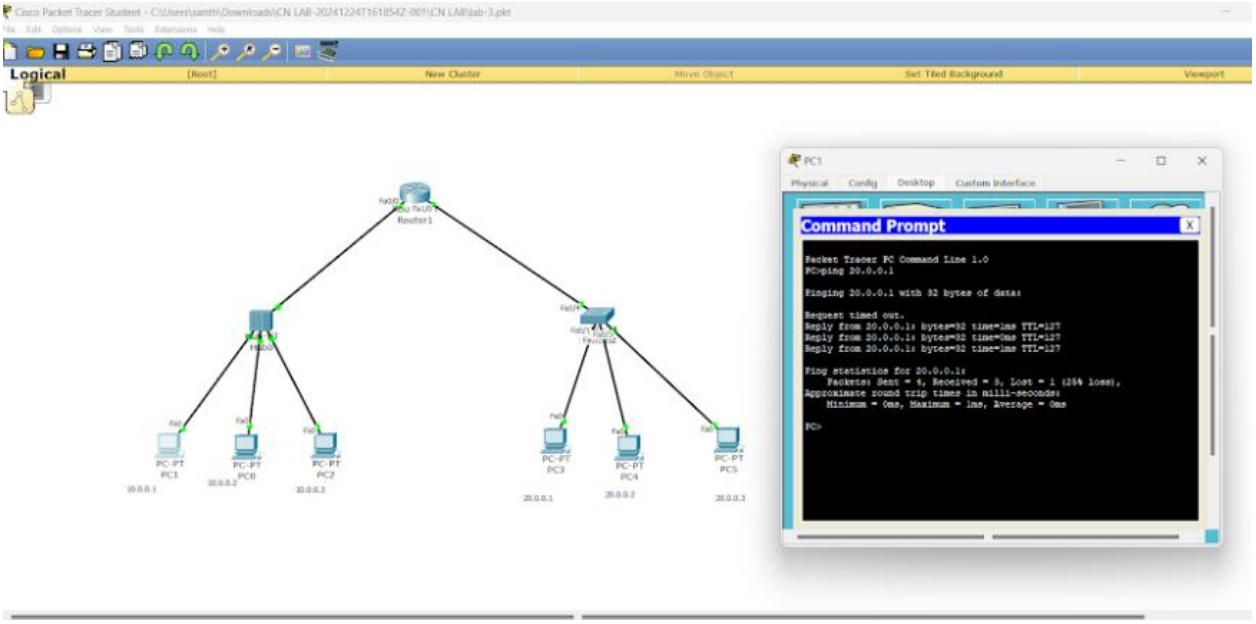
Step 4:- Configure Router :-
click on Router, go to command line interface and enter the following commands :-
(i) Enable
(ii) Configure terminal
(iii) Interface fastEthernet 0/0
ip address 10.0.0.1 255.255.255.0
no shutdown
(iv) Interface fastEthernet 0/0
ip address 10.0.0.2 255.255.255.0
no shutdown



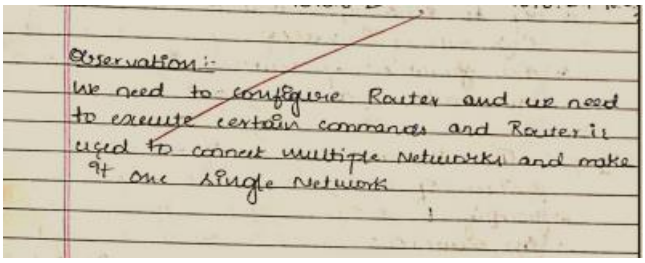

iii. Screen shots/ output:






iv. Observation



Observation :-

We need to configure Router and we need to execute certain commands and Router is used to connect multiple Networks and make it one single Network

**Program 3(1)**

i. Aim of the program: Configuring Two routers

ii. Procedure along with the topology

FRIDAY 18/10/24

Configuring 2 Routers

AIM: configuring 2 Routers in a Network

Step 01: Drag and drop 2 pc's and 2 Routers configure each device

-> configure PC1 and PC2 by ip address 10.0.0.1 and 20.0.0.1 respectively and gateway has 10.0.0.2 and 20.0.0.2 respectively

02 -> configure Router

(1) go to Router

(2) go to fast Ethernet 0/0 and turn on the port status

(3) go to Serial 2/0 and do the same for the 2nd router.

(4) Set ip address of router 0 as 30.0.0.1 and router 1 as 30.0.0.2

Step 03:- Run the commands in Router CLI

-> Show ip route

-> config t

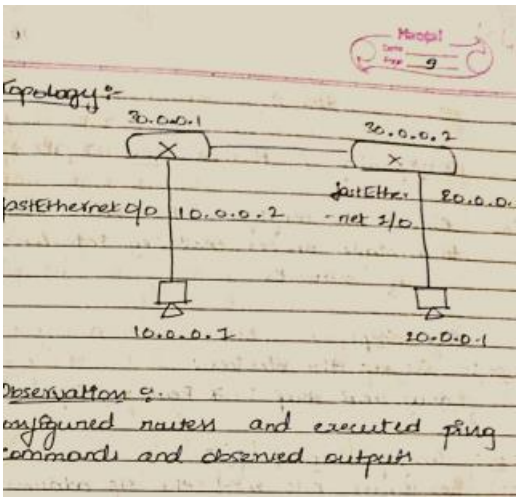
-> in router 0 :-

ip route 10.0.0.0 255.0.0.0 30.0.0.2

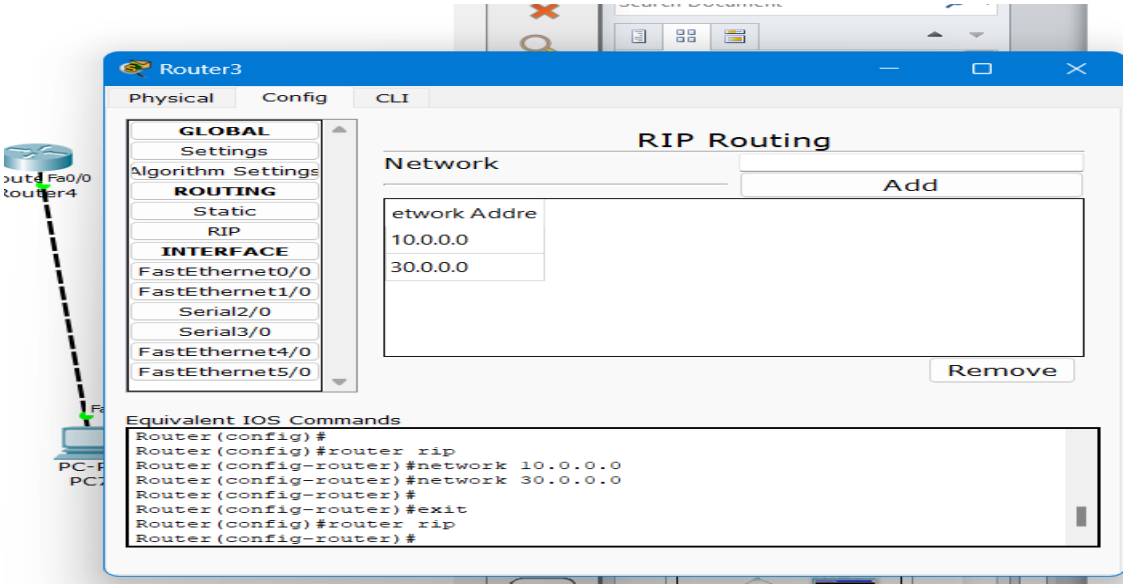
in router 1 :-

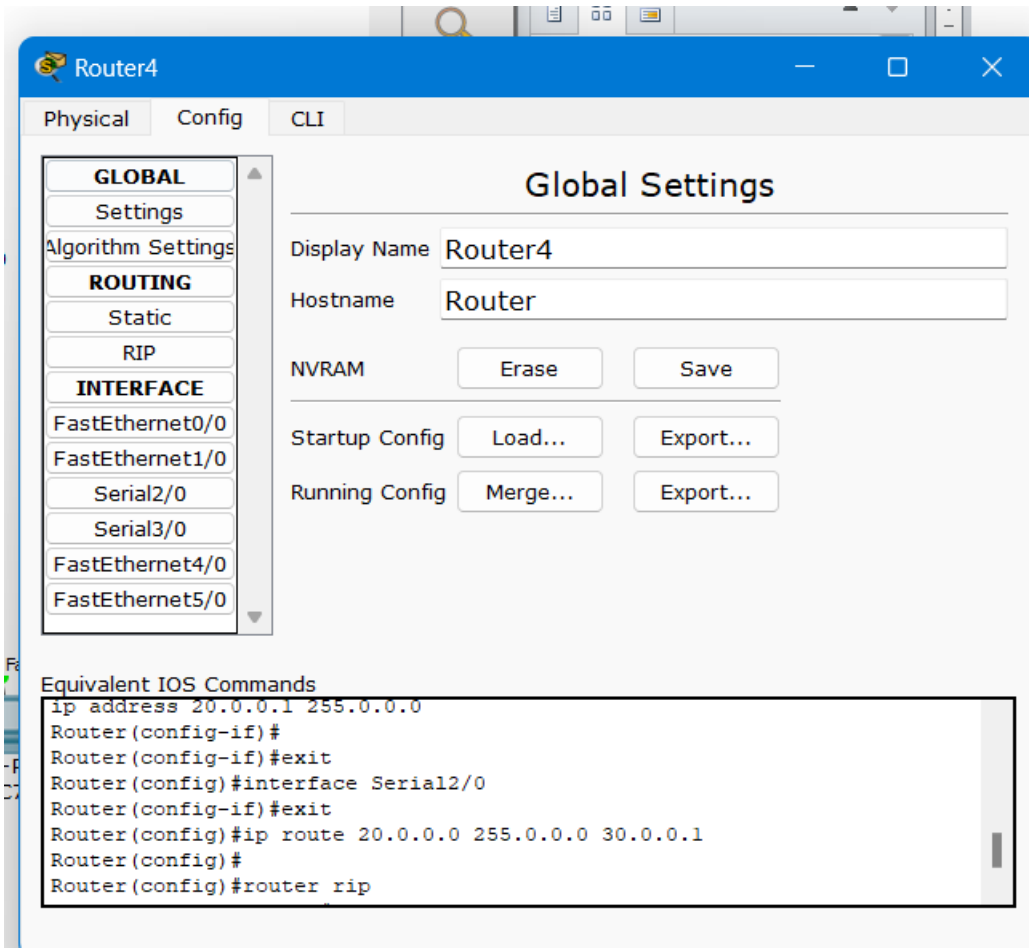
ip route 20.0.0.0 255.0.0.0 30.0.0.1

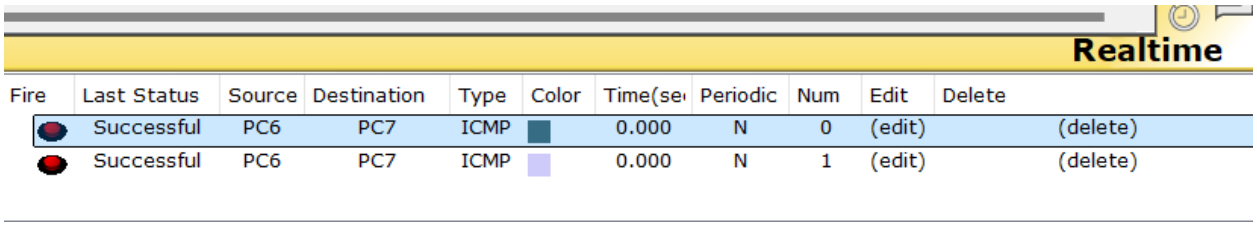
-> go to command prompt in pc and run ping 20.0.0.1




iii. Screen shots/ output:




| Fire | Last Status | Source | Destination | Type | Color | Time(se | Periodic | Num | Edit | Delete |
|---|---|---|---|---|---|---|---|---|---|---|
| | Successful | PC6 | PC7 | ICMP | | 0.000 | N | 0 | (edit) | (delete) |
| | Successful | PC6 | PC7 | ICMP | | 0.000 | N | 1 | (edit) | (delete) |

iv. Observation:

Observation :-

Configured routers and executed ping commands and observed output

Configure IP address of routers in Cisco packet tracer explore the following cmds, ping responses, request time, destination unreachable.

**Program 4**

i. Aim of the program: Configuring 3 routers and 2 Pc's

ii. Procedure along with the topology

25/10/

10

Lab-9

Configuring 3 Routers and 2 pc's

Aim:- Aim is to configure the router network to a single router enabling interconnection of networks

Algo Steps:-

Step 01:- Set up the devices:-
drag and drop :- 3 Routers, 2 pc's.

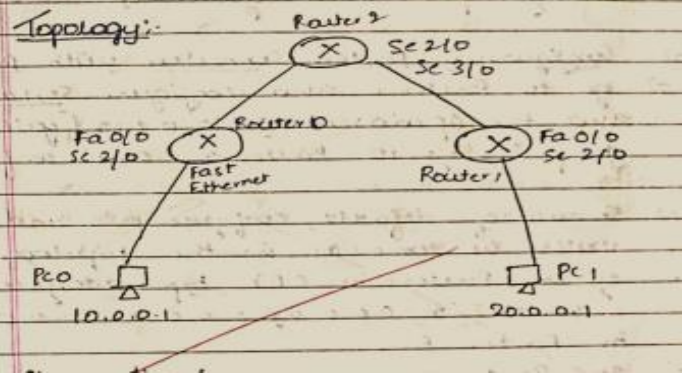
Step 02:- Configuring pc's :-
-> Configure pc1 with the ip address 10.0.0.1 and for pc2, ip address -> 20.0.0.1
-> Enable the gateway for pc1 as 10.0.0.2 and for pc2, with gateway 20.0.0.2

Step 03:- Configuring Routers connected with pc's
-> go to Router 1 and configure Fast Ethernet 0/0 with ip address 10.0.0.2 [gateway for R1], go to port serial 2/0 and the ip address needs to be set as 30.0.0.1
-> go to RIP Routing and add the respective ip addresses which was set for the ports
-> IIIly for Router 2, configure Fast Ethernet 0/0 with ip address 20.0.0.2 [gateway for pc2], go to port serial 2/0 and set ip address as 40.0.0.1
-> go to RIP Routing and add the respective ip addresses which was set for the ports

Step 04:- Configuring Routers connected to 2 other Rout
-> go to port serial 2/0 and set the ip addr
ress as 30.0.0.4 with Subnet Mask 255.0.0.
-> go to port serial 3/0 and set the ip addr
as 40.0.0.4 with Subnet mask 255.0.0.
-> Add the respective ip addresses to the RIP
Routing and setup the connection

Step 05:- go to CLI for pc0 and execute ping 20.0
111 us for pc1, execute ping 10.0.0.1 and
check for the successful ping message

Topology:-




II Aim:- Configuring same topology with default Routing.

Steps:

Step 01: Set up the devices:
drag and drop :- 3 Routers, 2 pc's

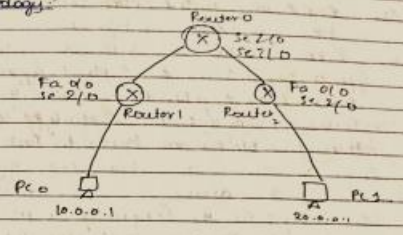
Step 02:- Configure pc's:
-> Configure pc1 with the ip address 10.0.0.1
and for pc2, ip address -> 20.0.0.1
-> Enable the gateway for pc1 as 10.0.0.2
and for pc2, with gateway 20.0.0.2

Step 03:- Configuring Routers connected with pc's
-> go to Router 1 and configure Serial 2/0
and the ip address as 10.0.0.2 [gateway
for pc1], go to Router 2 and configure

-> In case of default, configure pc's and
routers as same as in the topology
-> go to Router in CLI, type interface
serial 2/0, 0.0.0.0, 0.0.0.0 30.0.0.2
in Router R1
-> In Router 2
in CLI, type interface serial 2/0,
ip Route 0.0.0.0 0.0.0.0 40.0.0.2
-> In Router R0
In CLI, type Interface 2/0,
ip Route 0.0.0.0 0.0.0.0 30.0.0.1
Interface Serial 3/0
ip Route 0.0.0.0 0.0.0.0 40.0.0.1

Step 04:- go to CLI for pc0 and execute ping
20.0.0.1 111 us for pc1, execute ping 10.0.0.1
and check for the successful ping message

Topology:-

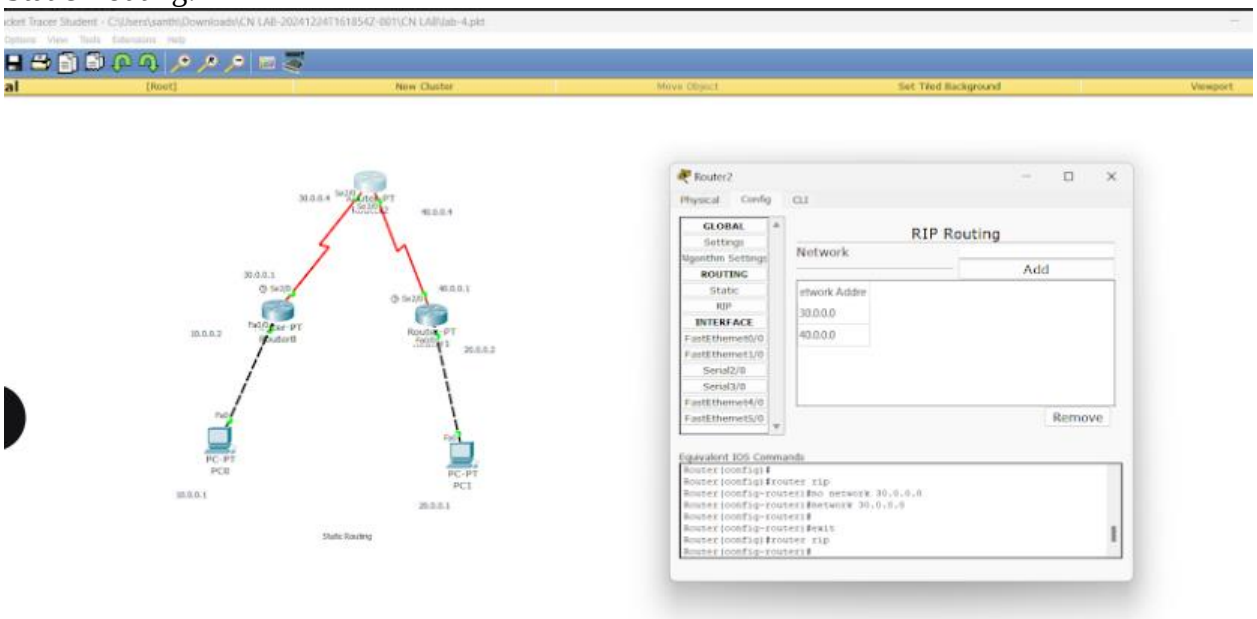


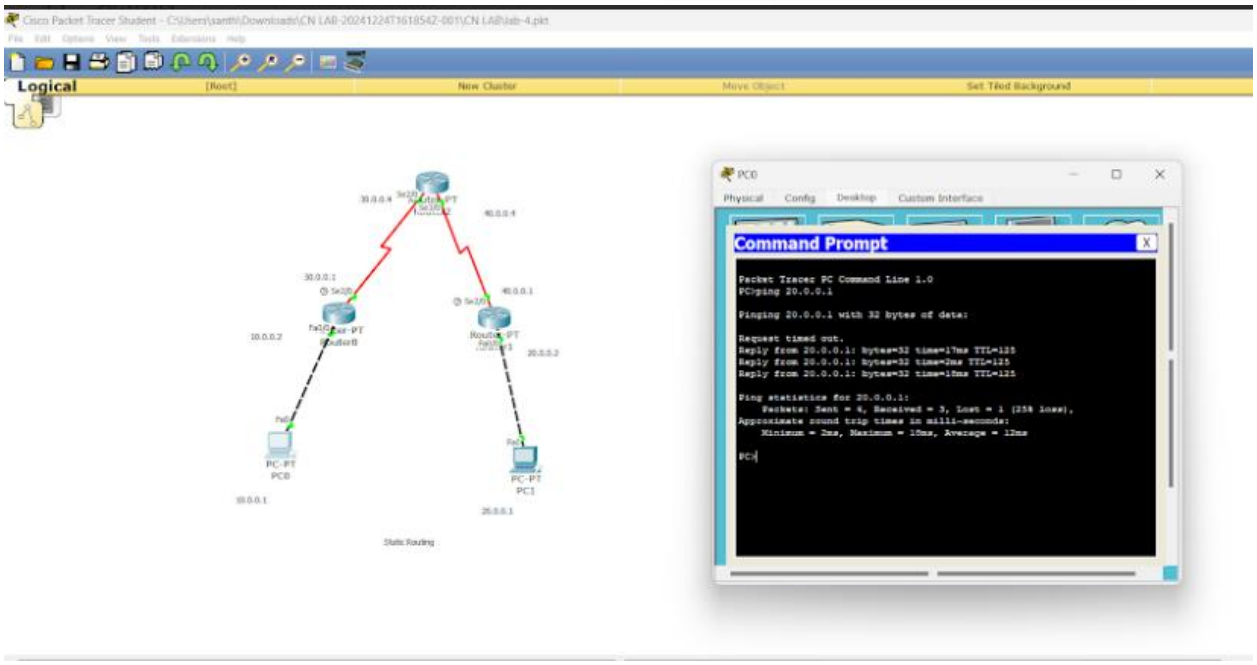

observation:-
Executed ping command to test the successful connection of the Router network connected to a single Router using default configuration

29/10/24

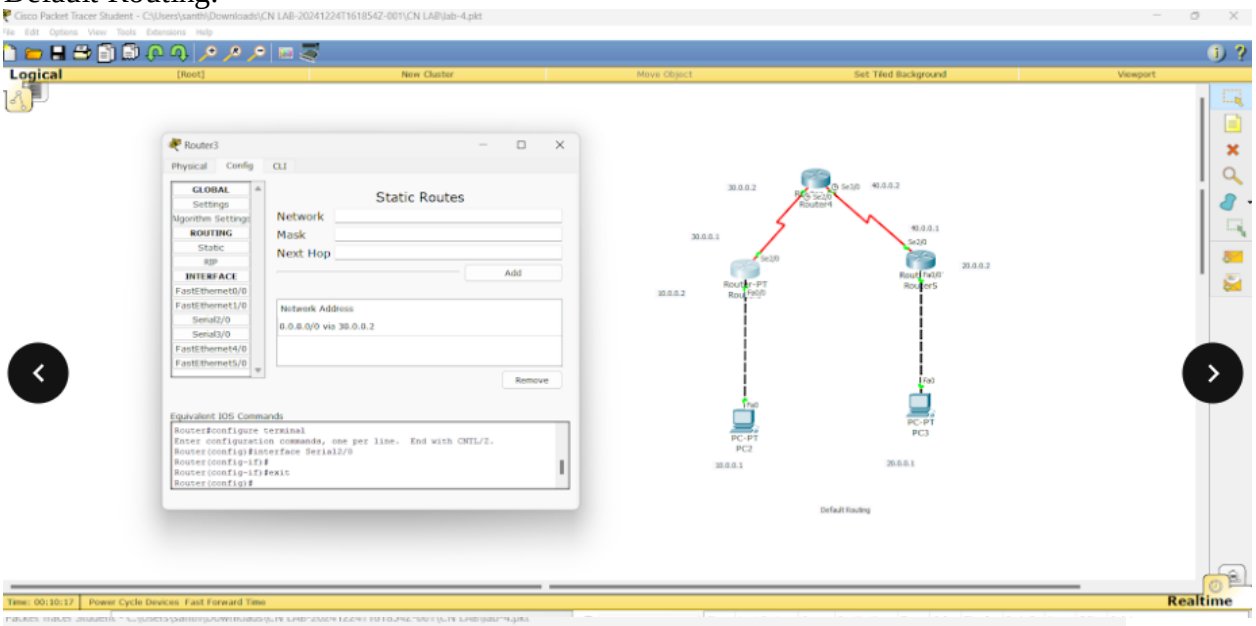
iii. Screen shots/ output:

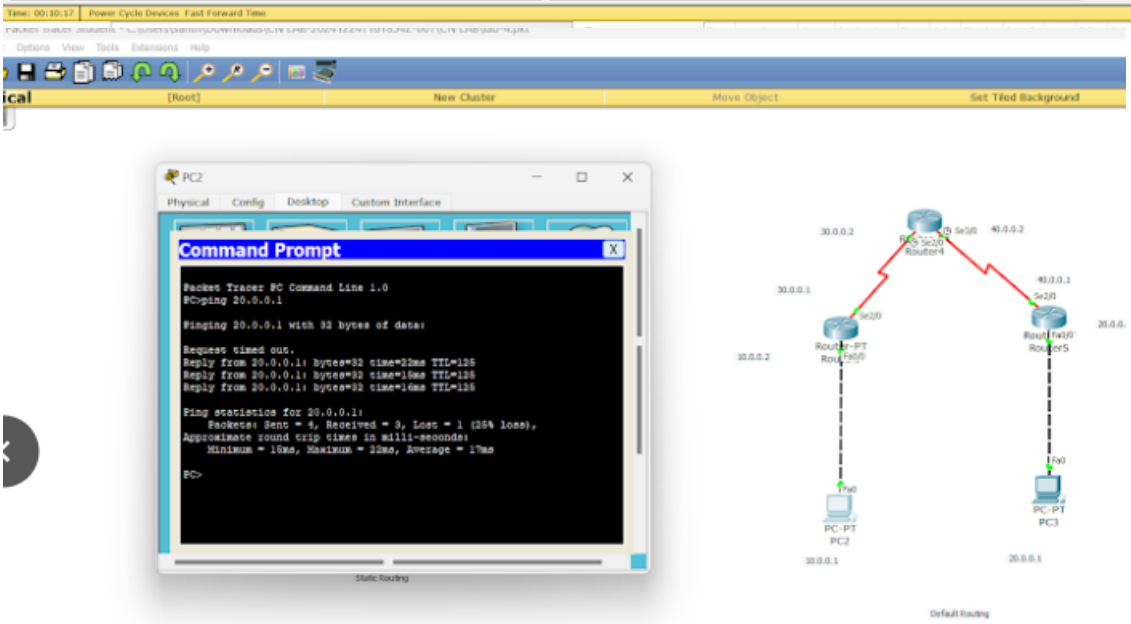
Static Routing:



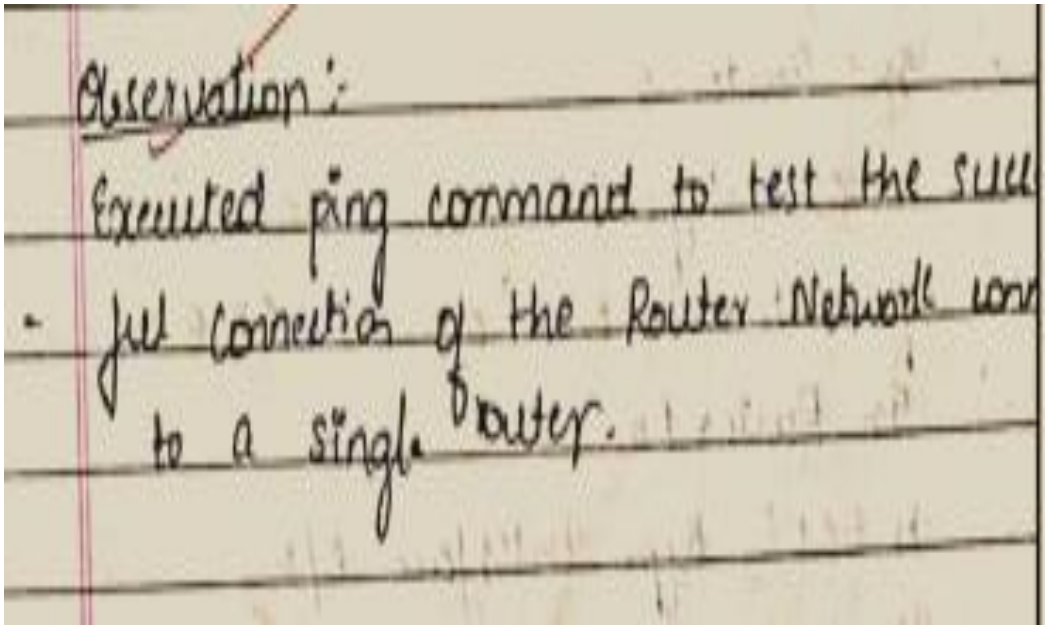



Default Routing:



iv. Observation



Observation :

Executed ping command to test the succ

- full connection of the Router Network con

to a single router.

**Program 5**

i. Aim of the program: To configure DHCP services for both LAN and across multiple LANs using a router in Cisco Packet Tracer

ii. Procedure along with the topology

III Basic DHCP configuration for a single N/w

Step 1:- Set up LAN Topology [router, switch, server, PC]

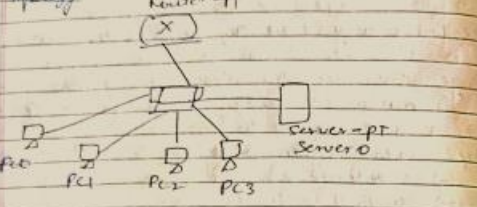
Step 2:- Configure the router interface with an ip address (10.0.0.1) & subnet mask (255.0.0.0)

Step 3:- Enter the commands
- -) enable
- -) config t
- -) interface fastethernet 0/0
- -) ip address 10.0.0.1 255.0.0.0
- -) no shutdown
- -) exit

Step 4:- In DHCP tab, configure the following
- -) Default gateway : 10.0.0.1
- -) DNS Server : 10.0.0.2
- -) Start ip Address : 10.0.0.10
- -) Maximum users : set it to 500
- -) TFTP Server : 10.0.0.2

Step 5:- Configure pc to use DHCP
- -) click on a pc, go to "Desktop" -> "ip configuration",
- -) Select DHCP
- -) The pc will automatically recieve & ip from the DHCP server.

Topology:




IV Using IP helper address to get DHCP from a Remote Network

Step1: Create Another N/W (with a different subnet eg 20.0.0.0/8 connected to a different router interface (fastethernet0/1)

Step 2:- Configure router interfaces with following
Interface 0/0, 10.0.0.1 (for '10.0.0.0/8' network)
Interface 0/1, 20.0.0.1 (for the 20.0.0.0/8 network)

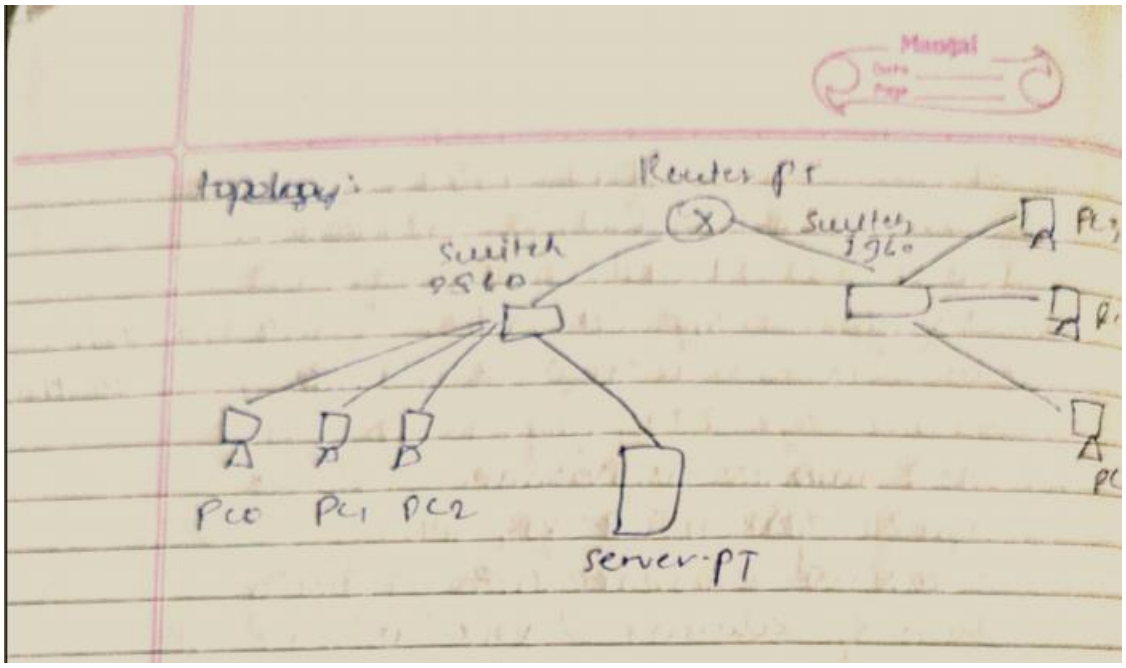
Step3: enter the following commands:-
-> enable
-> config t
-> Interface fastethernet 0/0
-> ip address 10.0.0.1 255.0.0.0
-> no shutdown
-> exit
-> Interface fastethernet 0/1
-> ip address 20.0.0.1 255.0.0.0
-> no shutdown
-> exit

Step 4: Configure DHCP Server for 20.0.0.0/8 N/W:
-> On server, create a new DHCP pool for the 20.0.0 N/W
-> pool name: 20 Network
-> Default Gateway: 20.0.0.1
-> DNS Server: 10.0.0.2
-> Start IP address: 20.0.0.10
-> Subnet Mask: 255.0.0.0
-> Maximum users: 100
-> TFTP Server: 10.0.0.2
Click add & Save

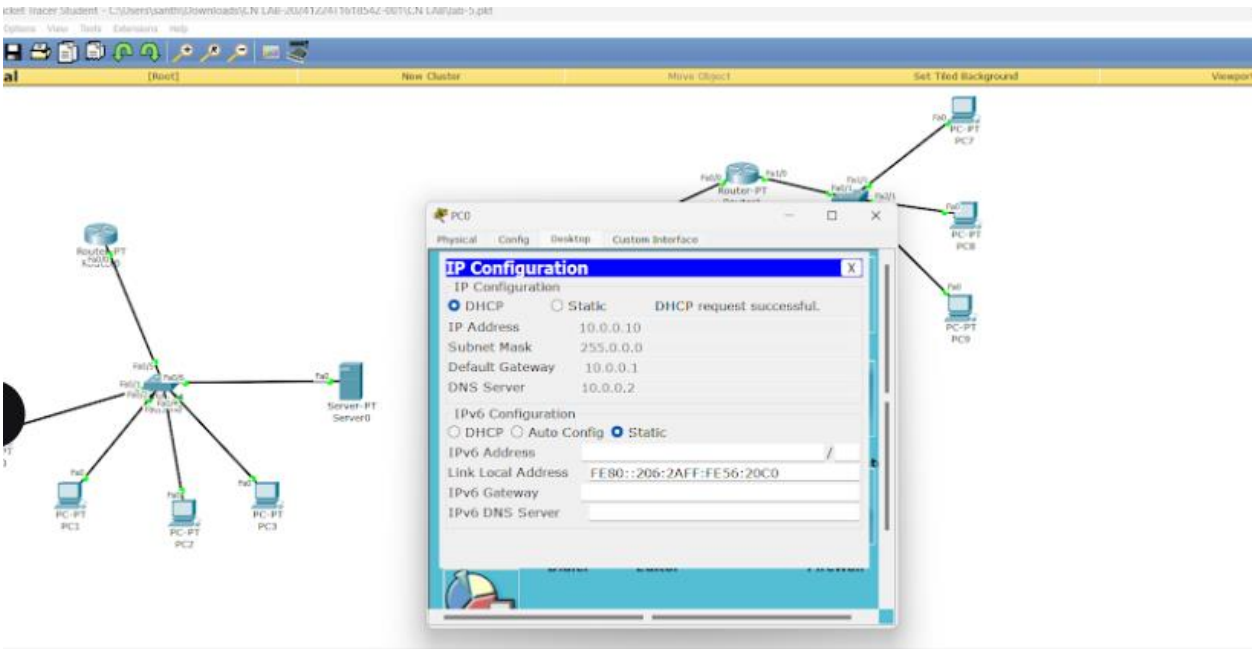
Step 4: To allow 20.0.0.0 N/W to access DHCP Server on 10.0.0.2 configure the IP-helper address on the router interface connected to the 20.0.0.0 N/W

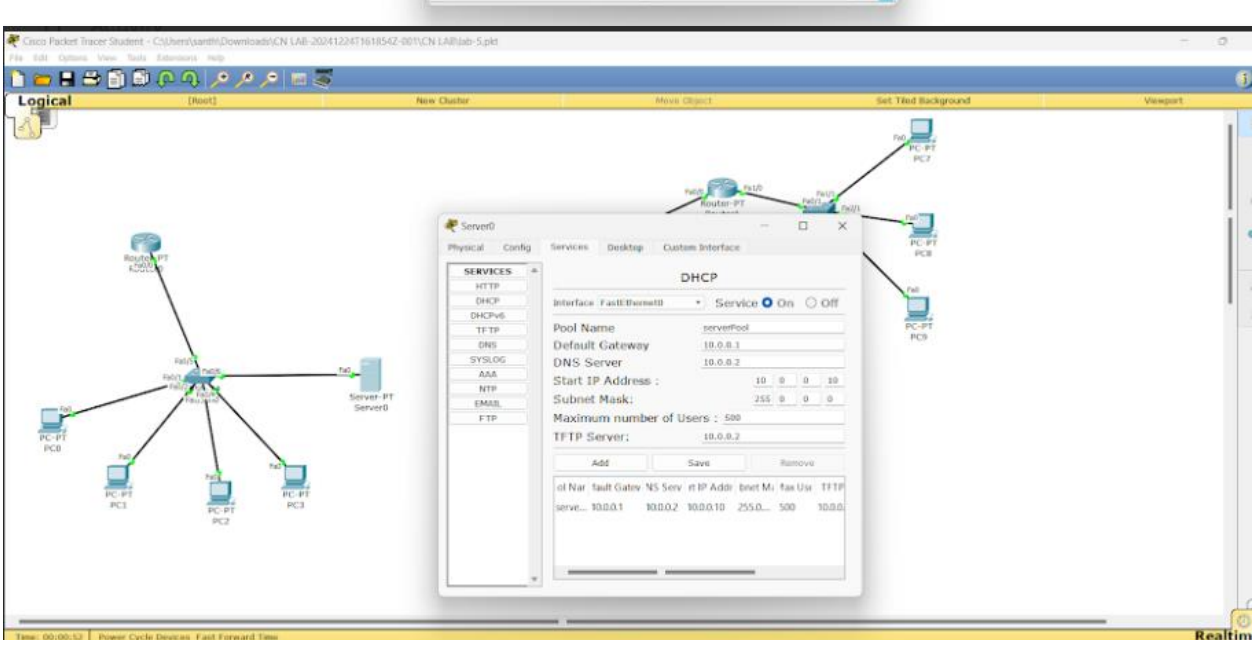
# Interface fast ethernet 1/0
IP helper address 10.0.0.2
exit

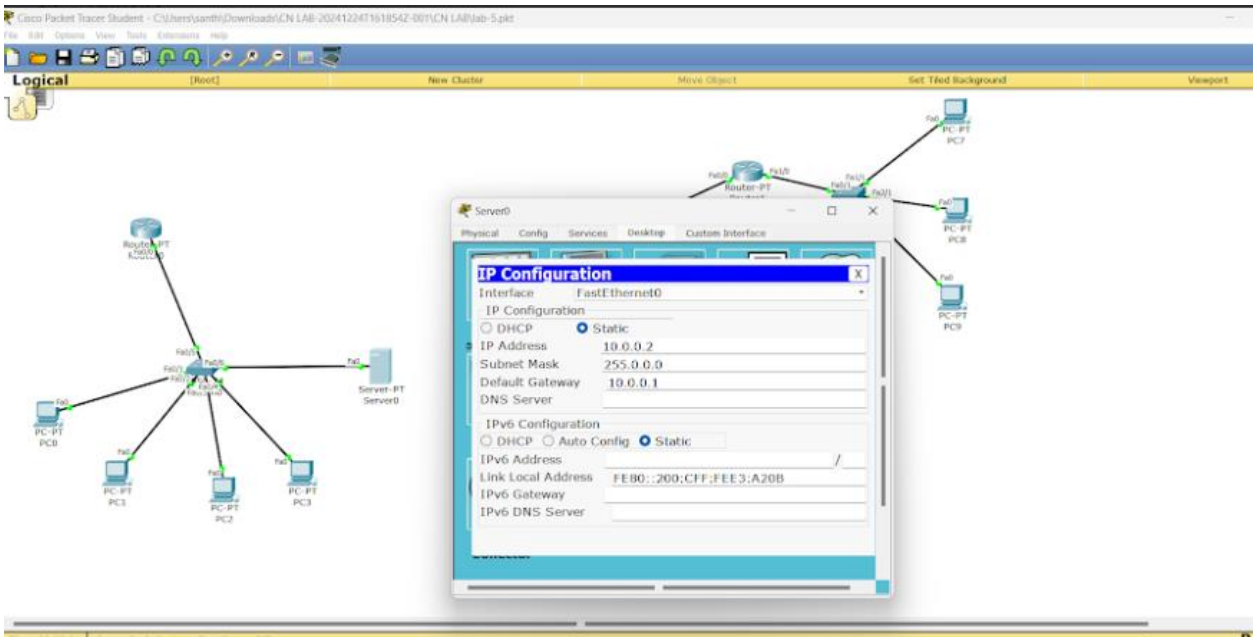
Step5:- On any PC in the 20.0.0.0 N/W go to desktop -> IP configuration, Select DHCP
-> The PC will recieve an IP from the DHCP server on the 10.0.0.0 N/W



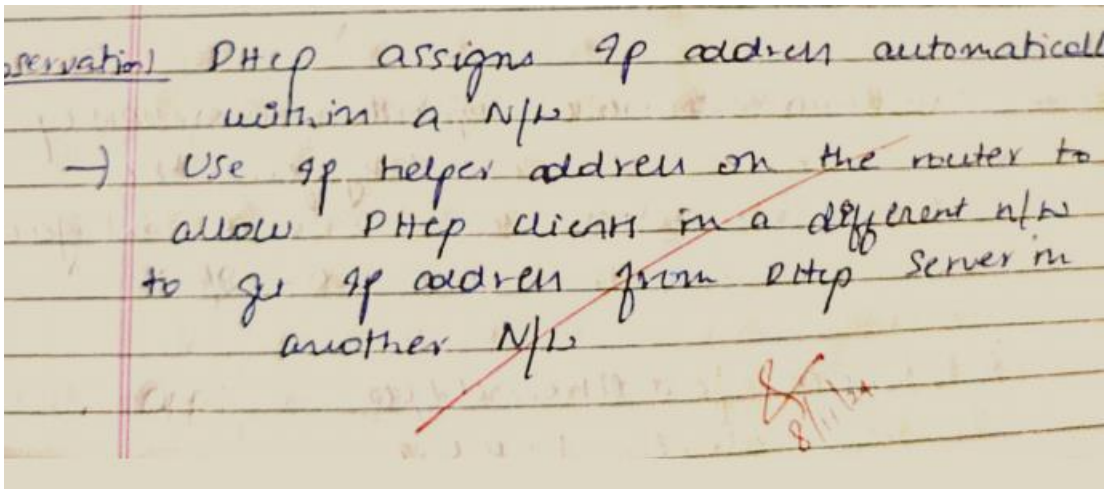

iii. Screen shots/ output:


PC0
Physical
Config
Desktop
Custom Interface
IP Configuration
IP Configuration
DHCP
Static
DHCP request successful.
IP Address 10.0.0.10
Subnet Mask 255.0.0.0
Default Gateway 10.0.0.1
DNS Server 10.0.0.2
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
Link Local Address FE80::206:2AFF:FE56:20C0
IPv6 Gateway
IPv6 DNS Server
Server-PT
Server0
PC-PT
PC1
PC2
PC3
PC7
PC8
PC9


Logical
Server0
Physical
Config
Services
Desktop
Custom Interface
SERVICES
HTTP
DHCP
DHCPv6
TFTP
DNS
SYSLOG
AAA
NTP
EMAIL
FTP
DHCP
Service On Off
Pool Name serverPool
Default Gateway 10.0.0.1
DNS Server 10.0.0.2
Start IP Address : 10 0 0 10
Subnet Mask: 255 0 0 0
Maximum number of Users : 500
TFTP Server: 10.0.0.2
Add
Save
Remove
Server-PT
Server0
PC-PT
PC0
PC1
PC2
PC3
PC7
PC8
PC9




iv. Observation:



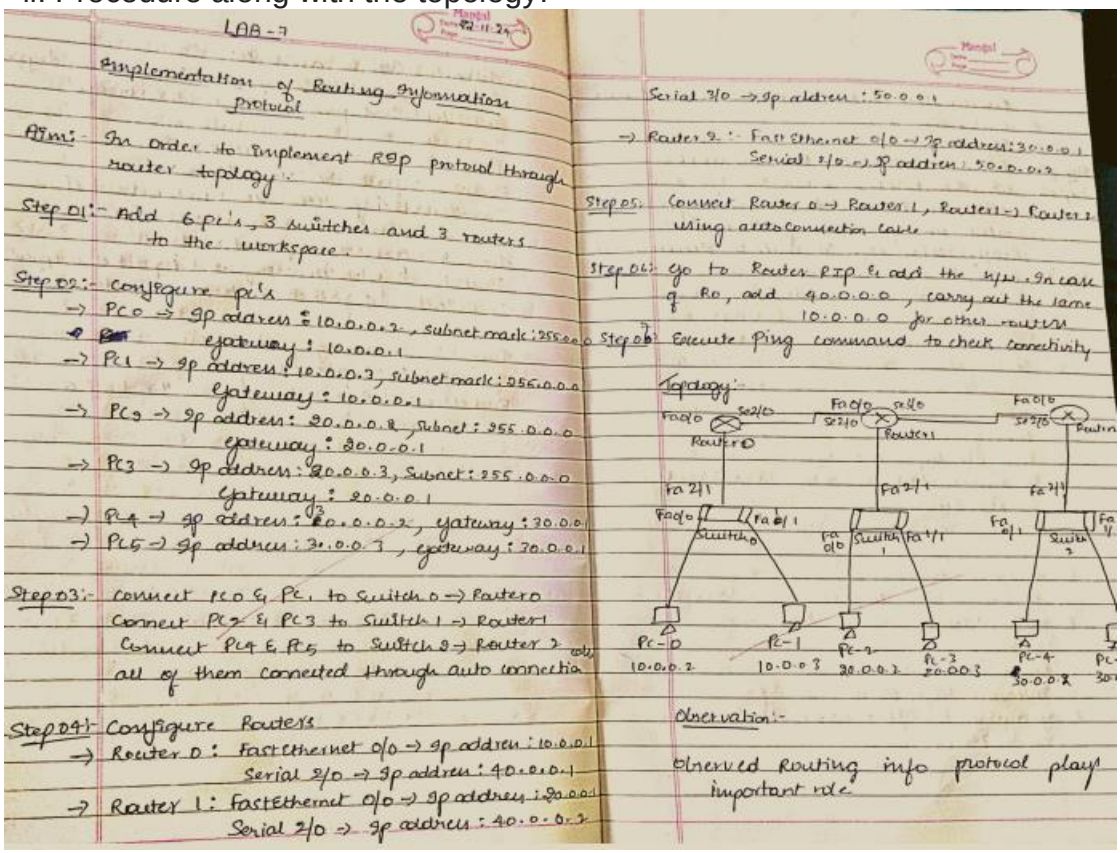
Observation: DHCP assigns IP address automatically within a N/W

→ Use IP helper address on the router to allow DHCP client in a different n/w to get IP address from DHCP server in another N/W

**Program 6**

i. Aim of the program: Configuring RIP(Routing Information Protocol on routers to enable dynamic routing between routers allowing them to share routing information

ii. Procedure along with the topology:



LAB-7

Implementation of Routing Information protocol

Aim: In order to implement RIP protocol through router topology.

Step 01:- Add 6 pc's, 3 switches and 3 routers to the workspace.

Step 02:- Configure pc's
-> Pc0 -> ip address: 10.0.0.2, subnet mask: 255.0.0.0
gateway: 10.0.0.1
-> Pc1 -> ip address: 10.0.0.3, subnet mask: 255.0.0.0
gateway: 10.0.0.1
-> Pc2 -> ip address: 20.0.0.2, subnet: 255.0.0.0
gateway: 20.0.0.1
-> Pc3 -> ip address: 20.0.0.3, subnet: 255.0.0.0
gateway: 20.0.0.1
-> Pc4 -> ip address: 30.0.0.2, gateway: 30.0.0.1
-> Pc5 -> ip address: 30.0.0.3, gateway: 30.0.0.1

Step 03:- Connect Pc0 & Pc1 to Switch 0 -> Router 0
Connect Pc2 & Pc3 to Switch 1 -> Router 1
Connect Pc4 & Pc5 to Switch 2 -> Router 2
all of them connected through auto connection

Step 04:- Configure Routers
-> Router 0: FastEthernet 0/0 -> ip address: 10.0.0.1
Serial 2/0 -> ip address: 40.0.0.1
-> Router 1: FastEthernet 0/0 -> ip address: 20.0.0.1
Serial 2/0 -> ip address: 40.0.0.2
Serial 3/0 -> ip address: 50.0.0.1
-> Router 2: FastEthernet 0/0 -> ip address: 30.0.0.1
Serial 2/0 -> ip address: 50.0.0.2

Step 05: Connect Router 0 -> Router 1, Router 1 -> Router 2 using autoconnection cable

Step 06: go to Router RIP & add the n/w. In case of R0, add 40.0.0.0, carry out the same 10.0.0.0 for other routers

Step 07: Execute Ping command to check connectivity

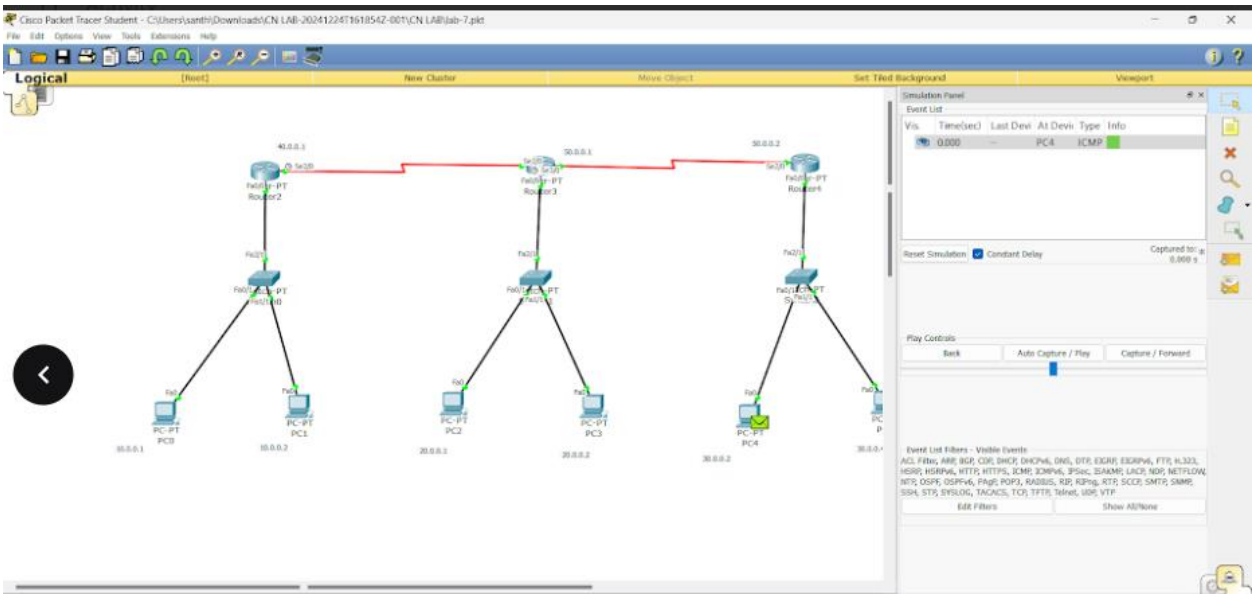
Topology:

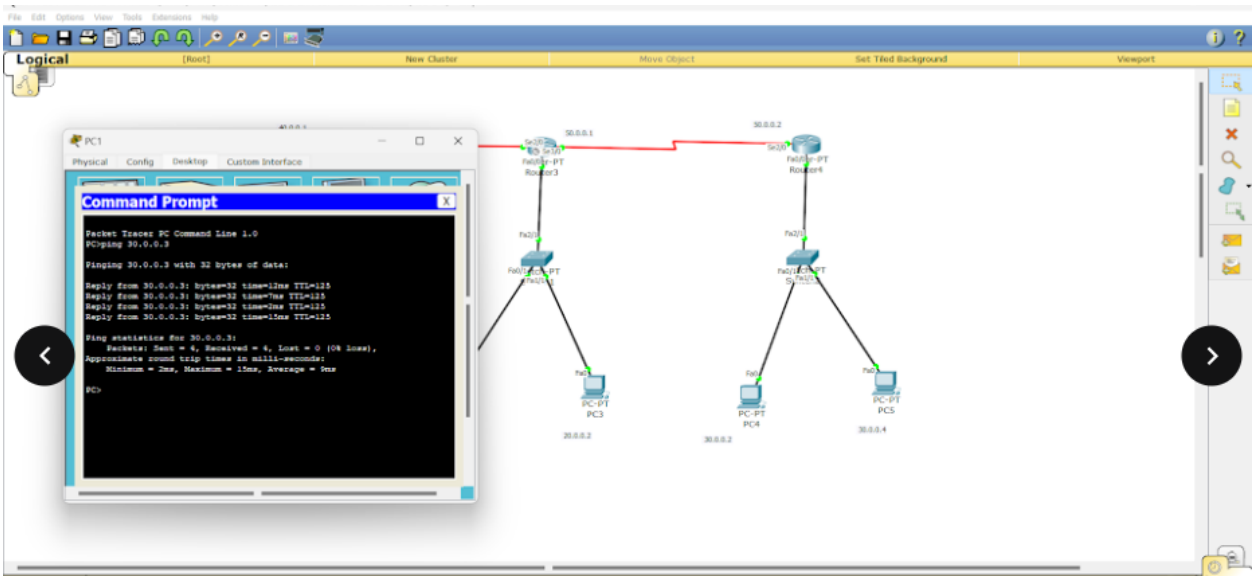


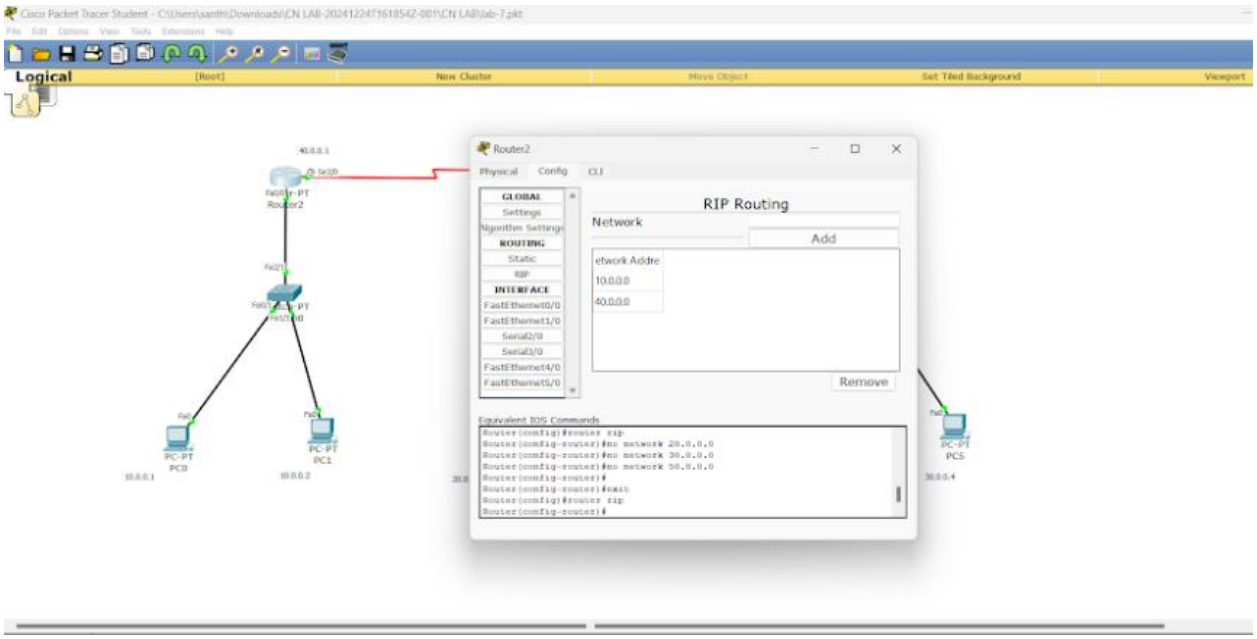
Observation:-

Observed Routing info protocol plays important role.

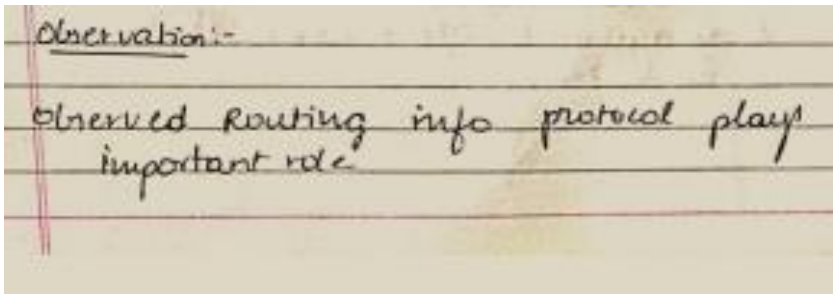
iii. Screen shots/ output:


Logical
Simulation Panel
Event List
Play Controls


Logical
PC1
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 30.0.0.3
Pinging 30.0.0.3 with 32 bytes of data:
Reply from 30.0.0.3: bytes=32 time=12ms TTL=125
Reply from 30.0.0.3: bytes=32 time=7ms TTL=125
Reply from 30.0.0.3: bytes=32 time=2ms TTL=125
Reply from 30.0.0.3: bytes=32 time=15ms TTL=125
Ping statistics for 30.0.0.3:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 15ms, Average = 9ms
PC>




iv. Observation:



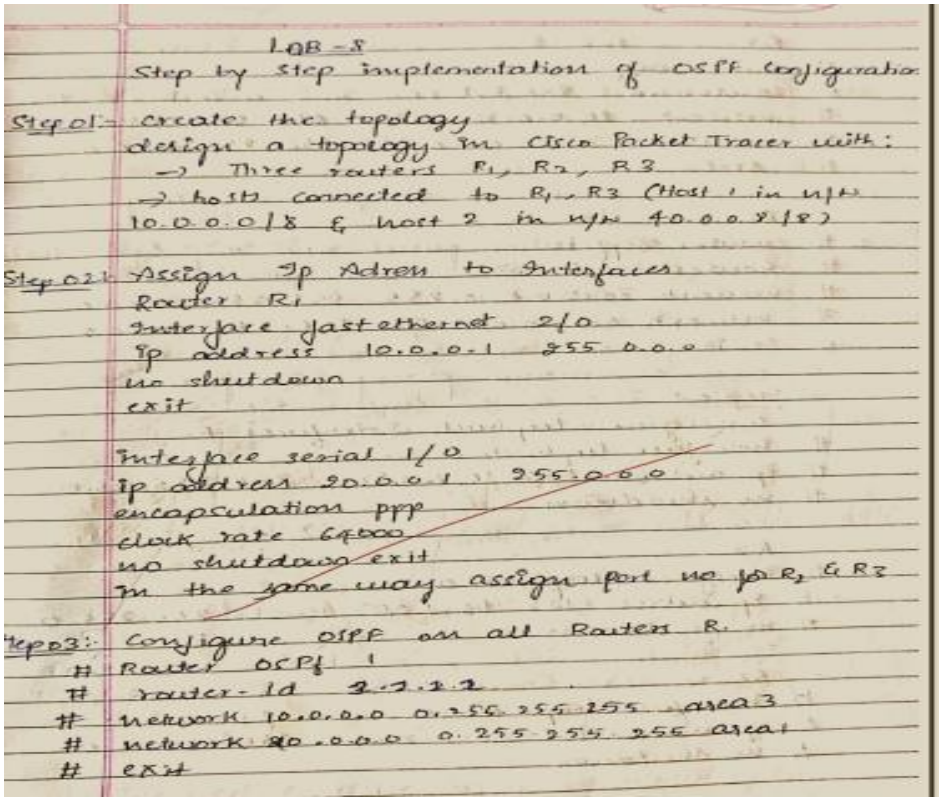
Observation:-

Observed Routing info protocol plays important role

**Program 7**

i. Aim of the program: Configuring OSPF to enable efficient and dynamic routing between routers using a link-state routing protocol

ii. Procedure along with the topology:



LAB - 8

Step by step implementation of OSPF configuration

Step 01:- create the topology

design a topology in Cisco Packet Tracer with:

→ Three routers R1, R2, R3

→ hosts connected to R1, R3 (Host 1 in n/w 10.0.0.0/8 & host 2 in n/w 40.0.0.0/8)

Step 02:- Assign IP Adress to Interfaces

Router R1

Interface fastethernet 2/0

ip address 10.0.0.1 255.0.0.0

no shutdown

exit

Interface serial 1/0

ip address 20.0.0.1 255.0.0.0

encapsulation ppp

clock rate 64000

no shutdown exit

In the same way assign port no for R2 & R3

Step 03:- Configure OSPF on all Routers R.

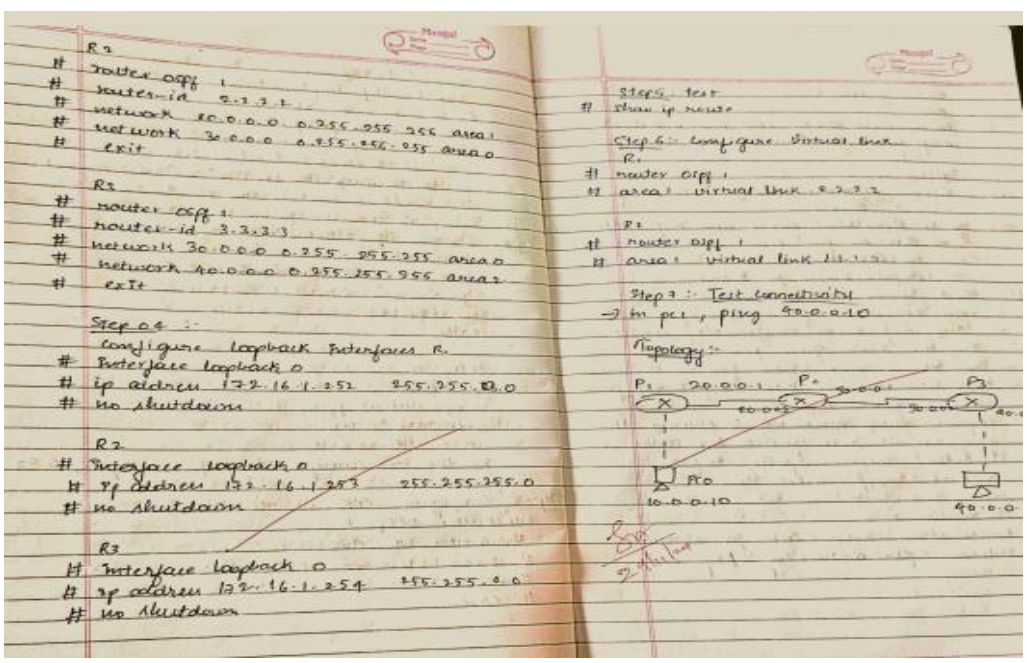
# Router OSPF 1

# router-id 2.2.2.2

# network 10.0.0.0 0.255.255.255 area 3

# network 20.0.0.0 0.255.255.255 area 1

# exit



R2

# router ospf 1

# router-id 2.2.2.2

# network 20.0.0.0 0.255.255.255 area 1

# network 30.0.0.0 0.255.255.255 area 0

# exit

R3

# router ospf 1

# router-id 3.3.3.3

# network 30.0.0.0 0.255.255.255 area 0

# network 40.0.0.0 0.255.255.255 area 2

# exit

Step 04 :-

Configure loopback Interfaces R.

# Interface loopback 0

# ip address 172.16.1.252 255.255.255.0

# no shutdown

R2

# Interface loopback 0

# ip address 172.16.1.253 255.255.255.0

# no shutdown

R3

# Interface loopback 0

# ip address 172.16.1.254 255.255.0.0

# no shutdown

Step 5 test

# show ip route

Step 6 :- Configure Virtual link

R1

# router ospf 1

# area 1 virtual link 2.2.2.2

R2

# router ospf 1

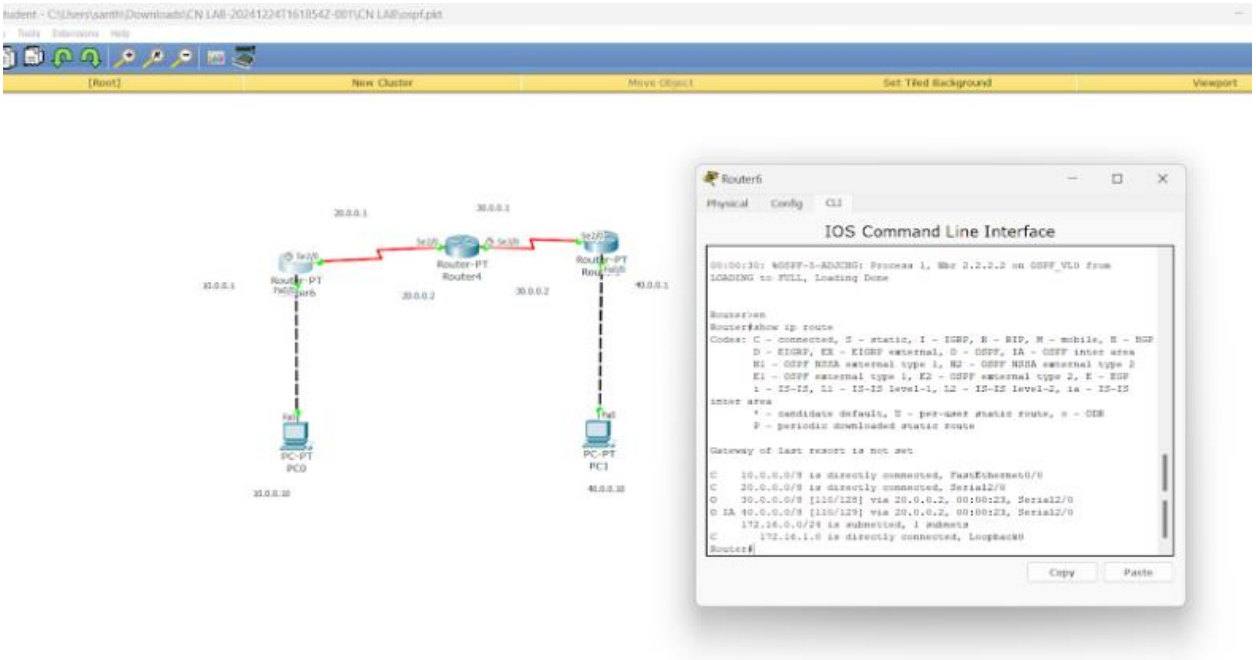
# area 1 virtual link 1.1.1.1

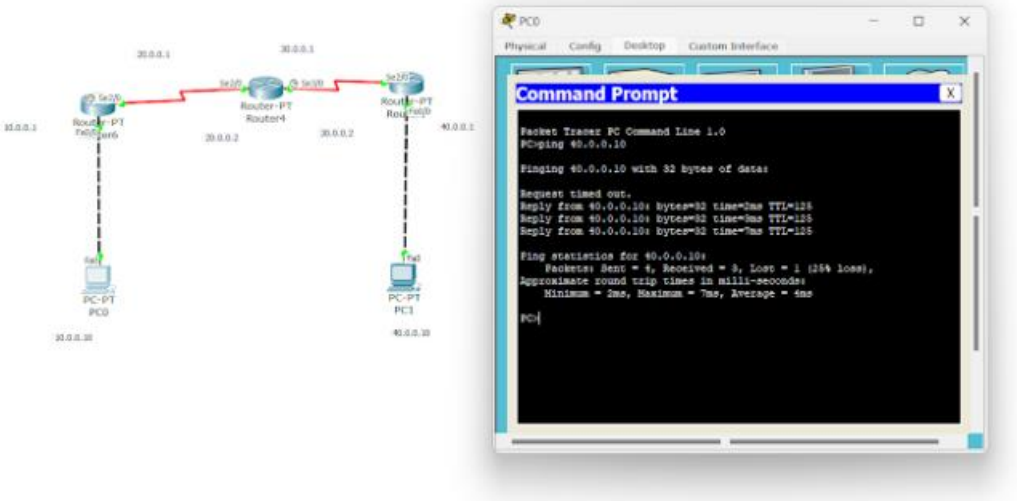
Step 7 :- Test connectivity

→ In pc1, ping 40.0.0.10

Topology :-

iii. Screen shots/ output:



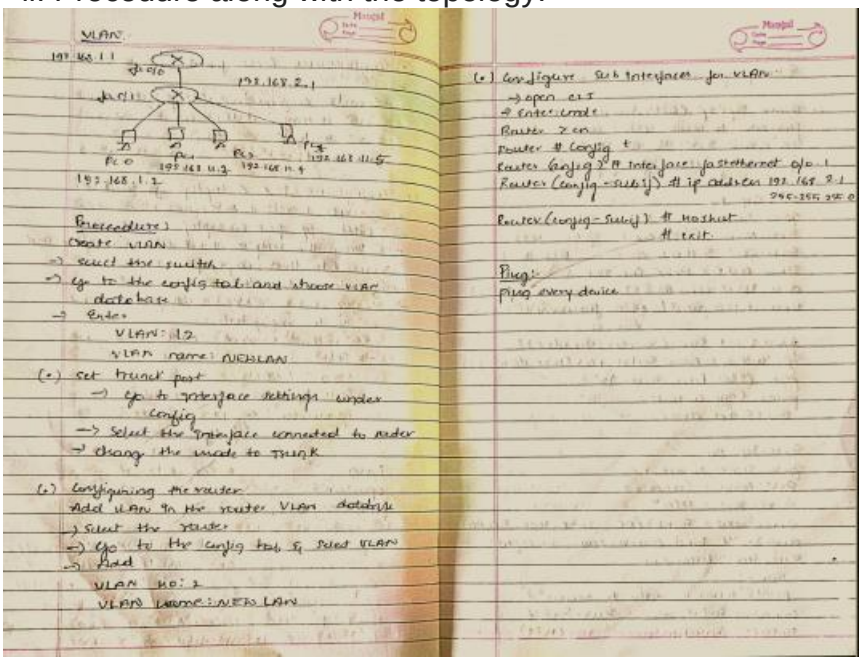

iv.Observation:

observation when configuring OSPF is that routers exchange link-state information allowing them to build a consistent and updated routing table based on the network topology.

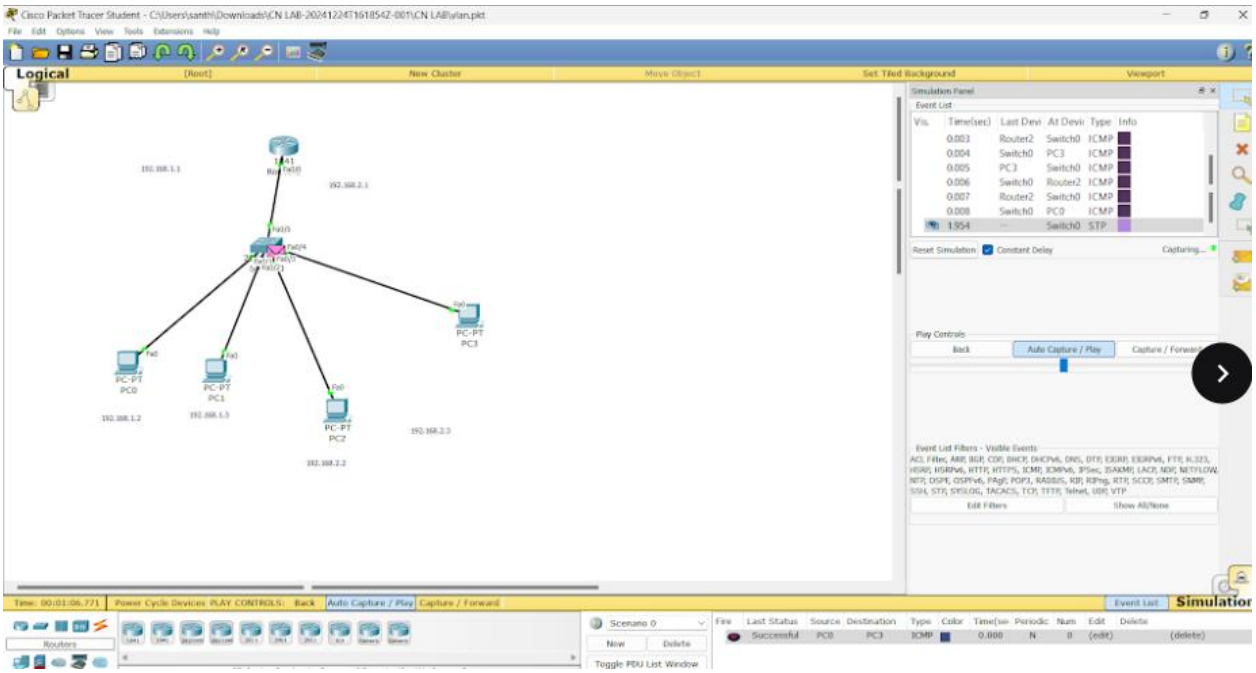
**Program 8**

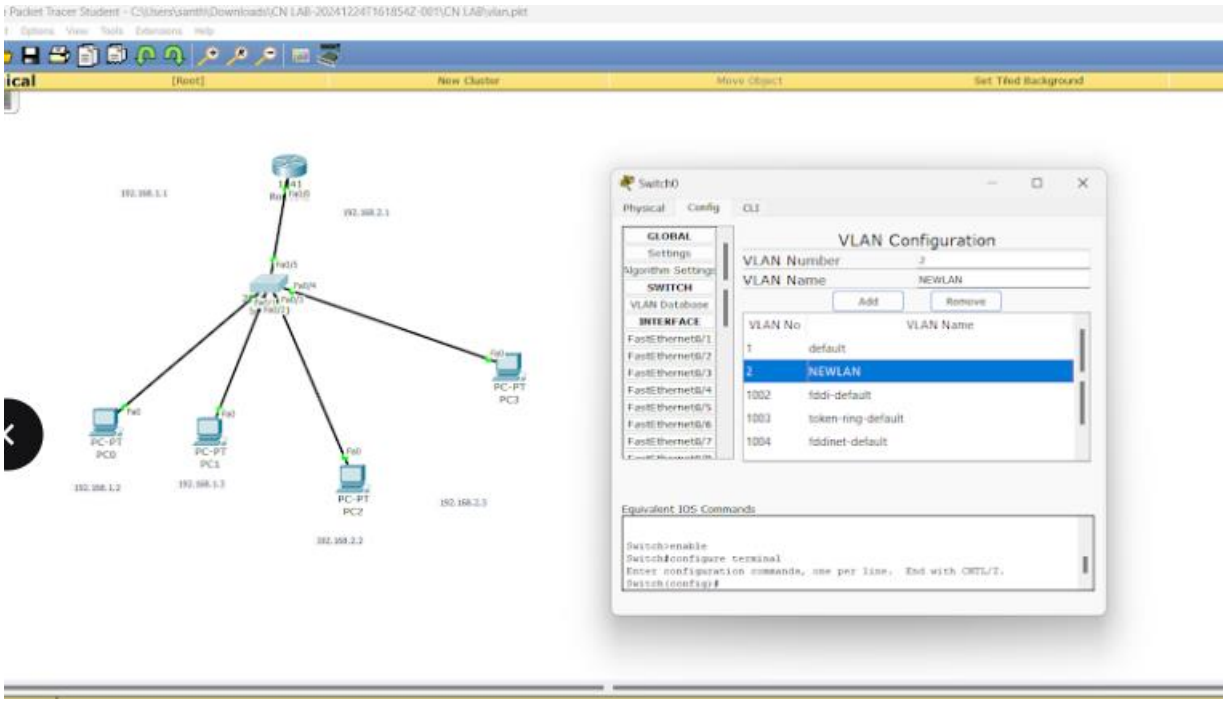
i. Aim of the program: to configure VLAN(Virtual Local Area Network) using two routers and 4 pc's

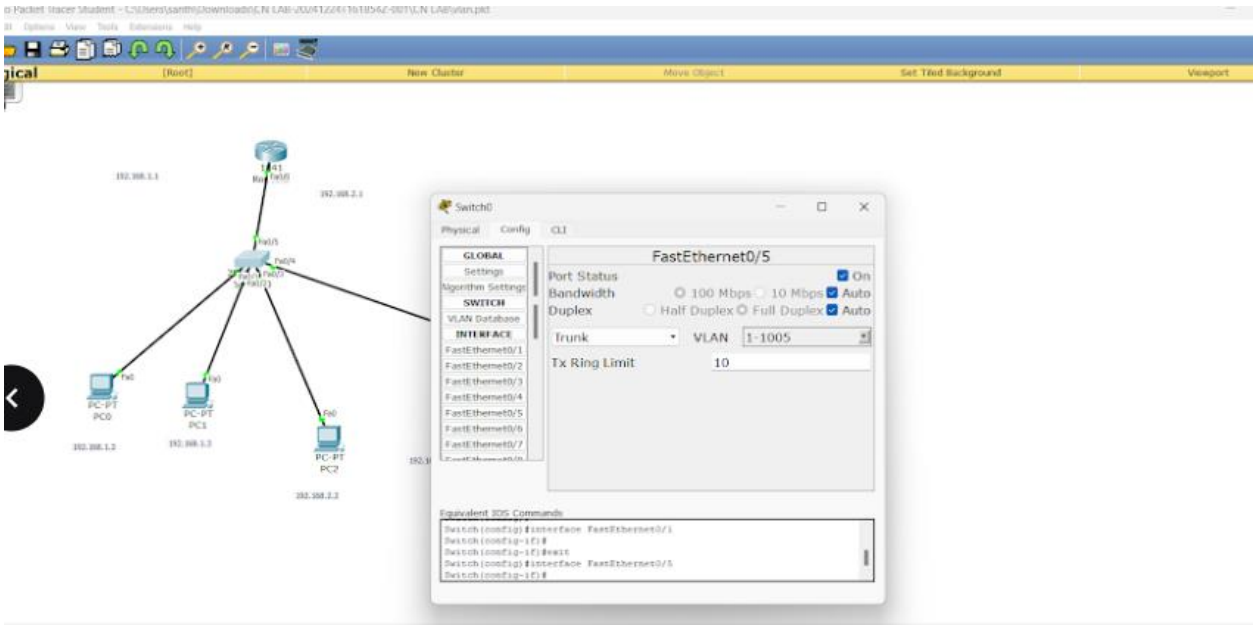
ii. Procedure along with the topology:

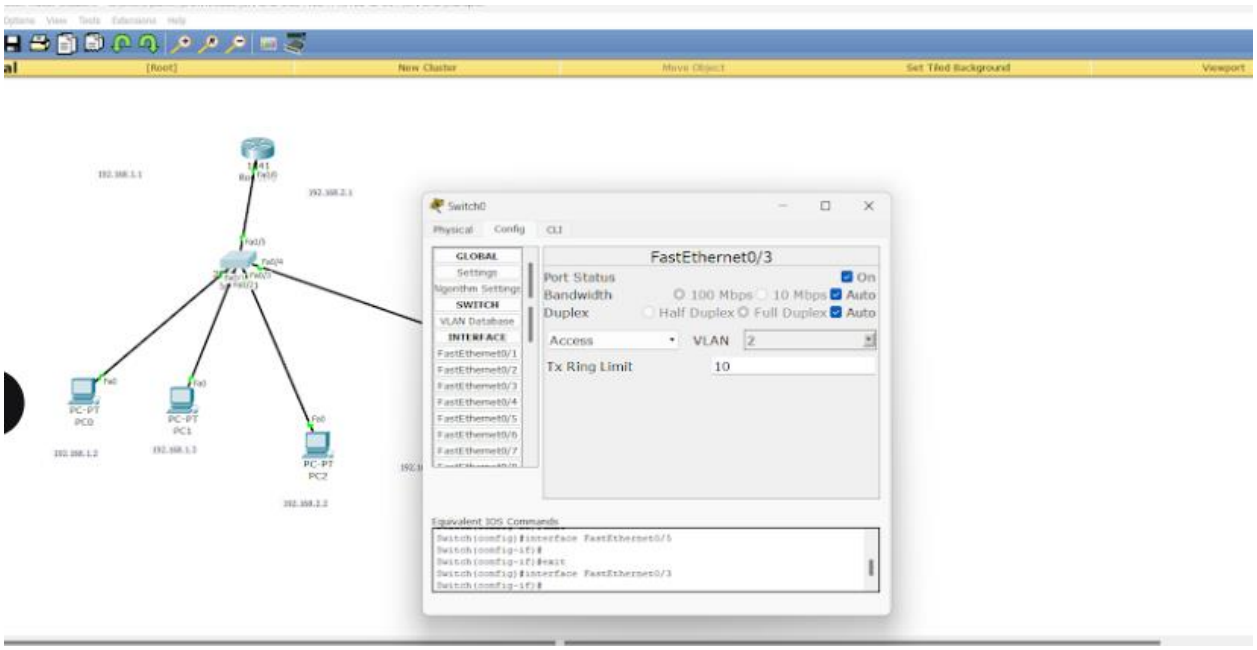


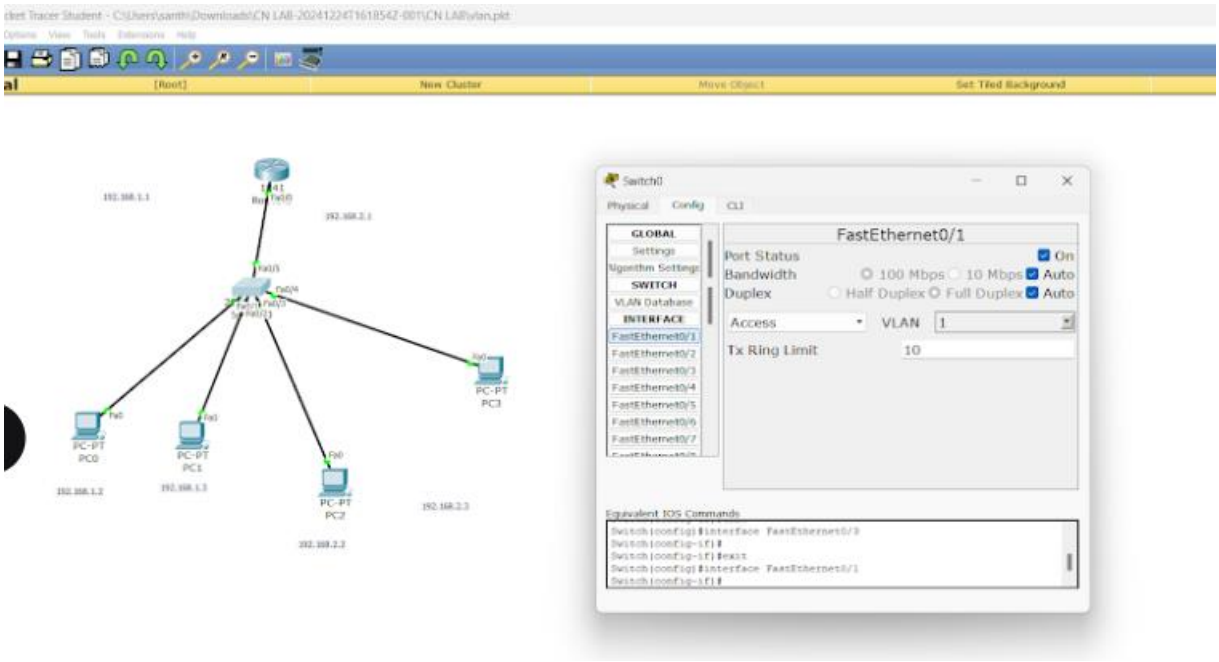

iii. Screen shots/ output:

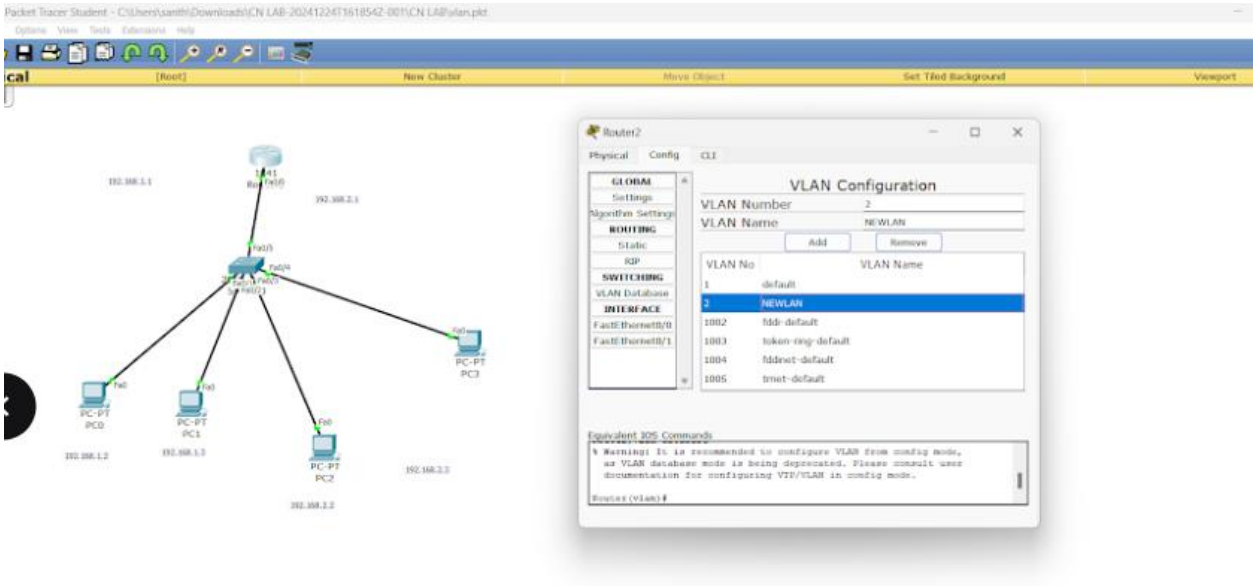


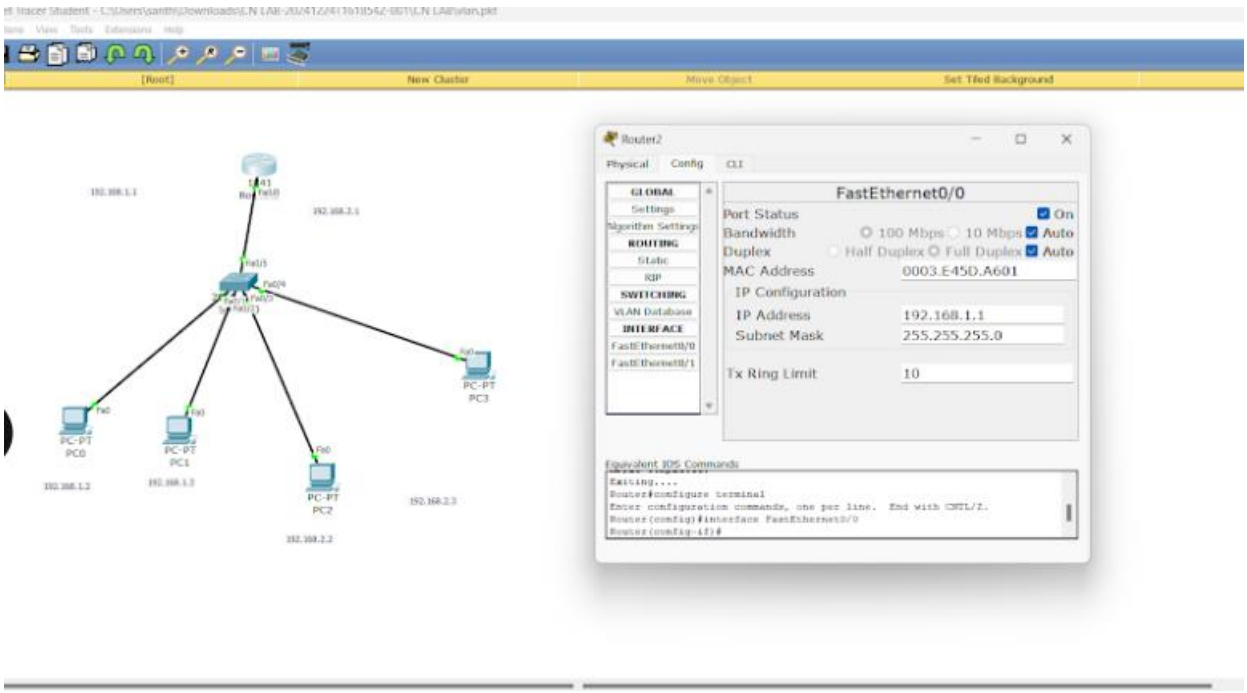

Switch0
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
SWITCH
VLAN Database
INTERFACE
VLAN Configuration
VLAN Number
VLAN Name
NEWLAN
Add
Remove
VLAN No
VLAN Name
1 default
2 NEWLAN
1002 fddi-default
1003 token-ring-default
1004 fddinet-default
Equivalent IOS Commands
Switch>enable
Switch#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)#
PC-PT PC0
PC-PT PC1
PC-PT PC2
PC-PT PC3


Switch0
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
SWITCH
VLAN Database
INTERFACE
FastEthernet0/5
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
trunk
VLAN
1-1005
Tx Ring Limit
10
Equivalent IOS Commands
Switch(config)#interface FastEthernet0/1
Switch(config-if)#
Switch(config-if)#exit
Switch(config)#interface FastEthernet0/5
Switch(config-if)#


Switch0
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
SWITCH
VLAN Database
INTERFACE
FastEthernet0/1
FastEthernet0/2
FastEthernet0/3
FastEthernet0/4
FastEthernet0/5
FastEthernet0/6
FastEthernet0/7
FastEthernet0/3
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
Access
VLAN
2
Tx Ring Limit
10
Equivalent IOS Commands
Switch(config)#interface FastEthernet0/3
Switch(config-if)#
Switch(config-if)#exit
Switch(config)#interface FastEthernet0/3
Switch(config-if)#
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2


Switch0
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
SWITCH
VLAN Database
INTERFACE
FastEthernet0/1
FastEthernet0/2
FastEthernet0/3
FastEthernet0/4
FastEthernet0/5
FastEthernet0/6
FastEthernet0/7
FastEthernet0/1
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
Access
VLAN
1
Tx Ring Limit
10
Equivalent IOS Commands
Switch(config)#interface FastEthernet0/3
Switch(config-if)#
Switch(config-if)#exit
Switch(config)#interface FastEthernet0/1
Switch(config-if)#
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC3




**iv.** Observation**:**

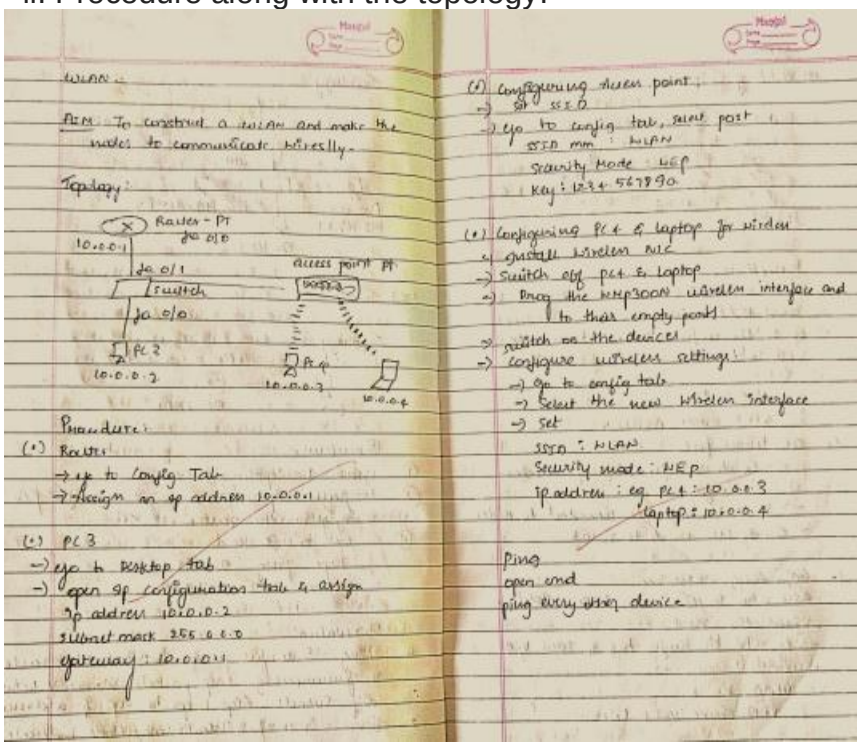
observations will help ensure a thorough understanding of VLANs and their behavior in a network simulation environment.

4o

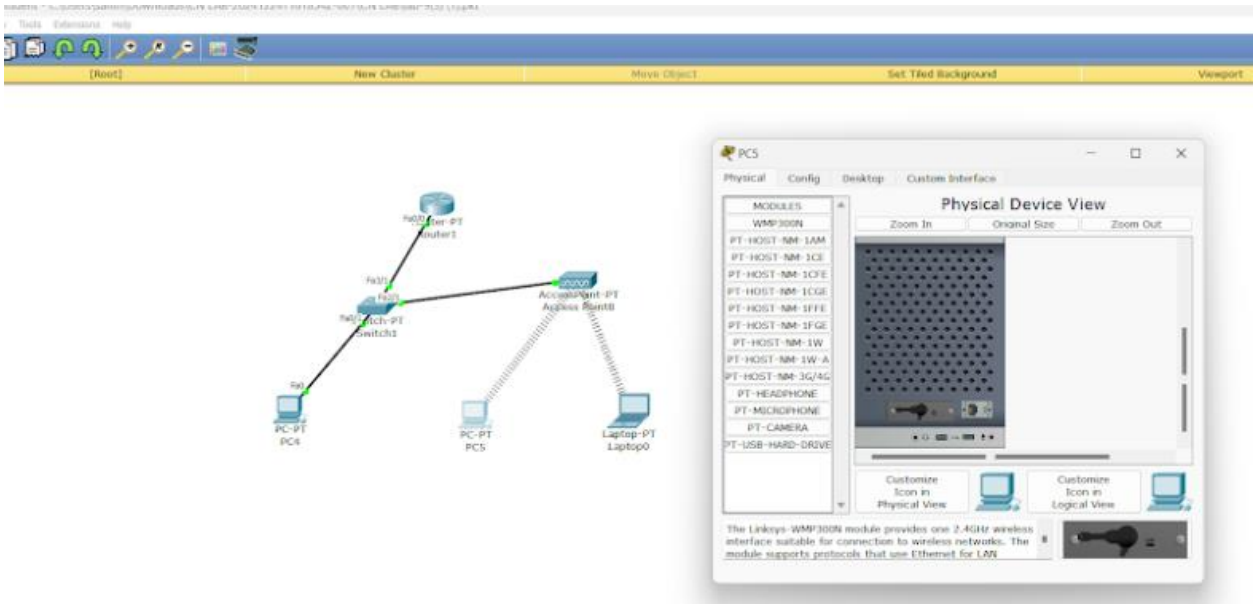
**Program 9**

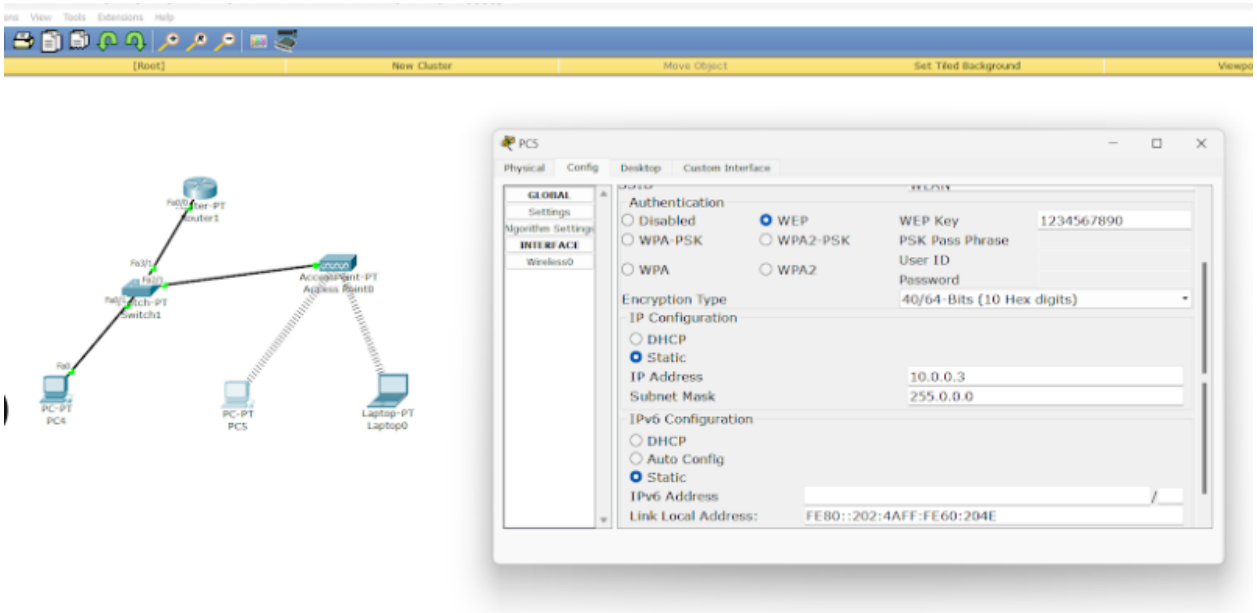
i. Aim of the program: to configure WLAN(Wireless Local Area Network) using a routers switches and through access points

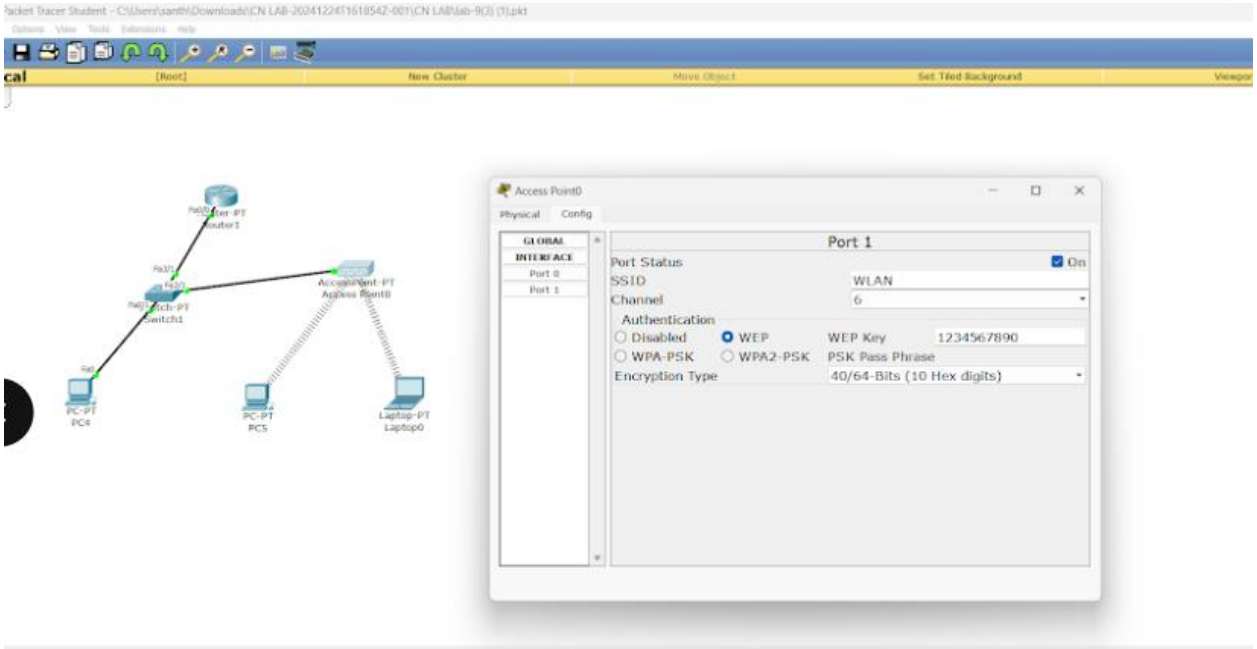
ii. Procedure along with the topology:

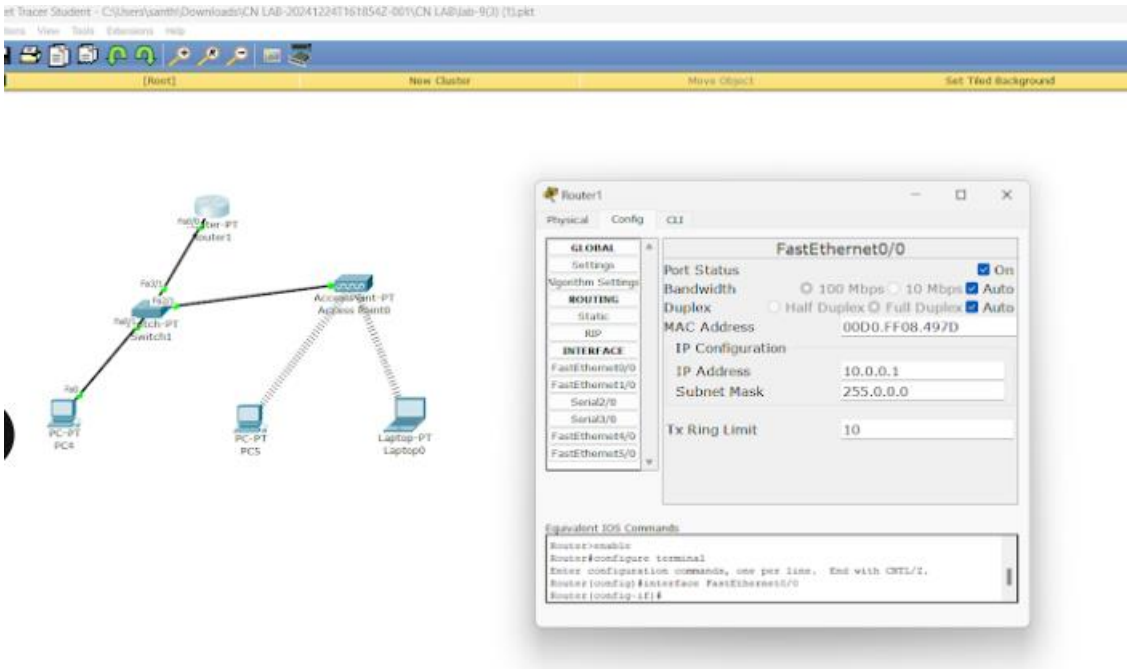



iii. Screen shots/ output:




PC5
Physical
Config
Desktop
Custom Interface
GLOBAL
Settings
INTERFACE
Wireless0
Authentication
Disabled
WEP
WPA-PSK
WPA2-PSK
WPA
WPA2
WEP Key
1234567890
PSK Pass Phrase
User ID
Password
Encryption Type
40/64-Bits (10 Hex digits)
IP Configuration
DHCP
Static
IP Address
10.0.0.3
Subnet Mask
255.0.0.0
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
Link Local Address:
FE80::202:4AFF:FE60:204E
Router-PT
Router1
Switch-PT
Switch1
Access Point-PT
Access Point0
PC-PT
PC4
PC-PT
PC5
Laptop-PT
Laptop0


Access Point0
Physical
Config
GLOBAL
INTERFACE
Port 0
Port 1
Port 1
Port Status
On
SSID
WLAN
Channel
6
Authentication
Disabled
WEP
WPA-PSK
WPA2-PSK
WEP Key
1234567890
PSK Pass Phrase
Encryption Type
40/64-Bits (10 Hex digits)
Router-PT
Router1
Switch-PT
Switch1
Access Point-PT
Access Point0
PC-PT
PC4
PC-PT
PC5
Laptop-PT
Laptop0



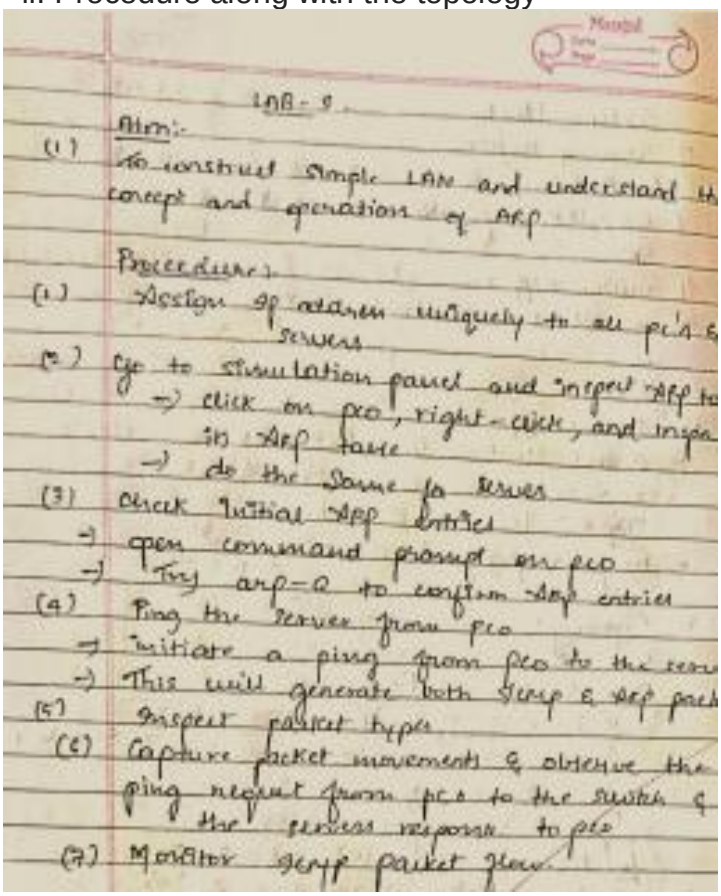

iv.Observation:

- Wireless devices (Access Points, PCs, Laptops) are connected and configured with appropriate IP settings and SSIDs.
- Signal coverage and range are visualized using coverage zones in simulation mode.

**Program 10**

i. Aim of the program: To demonstrate ARP(Adress Resolution Protocol) for observing resolved address while sending the packets

ii. Procedure along with the topology



LAB - 9

Aim:-

(1) To construct simple LAN and understand the concept and operation of ARP

Procedure:-

(1) Assign IP address uniquely to all pc's & servers

(2) Go to simulation panel and inspect ARP table
→ click on pc0, right-click, and inspect in ARP table
→ do the same for server

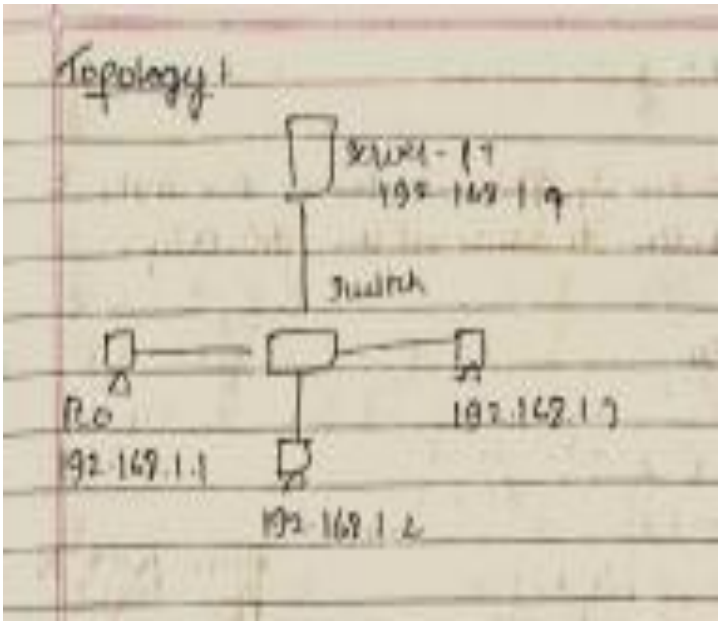
(3) Check initial ARP entries
→ open command prompt on pc0
→ Try arp-a to confirm ARP entries

(4) Ping the server from pc0
→ initiate a ping from pc0 to the server
→ This will generate both ICMP & ARP packets

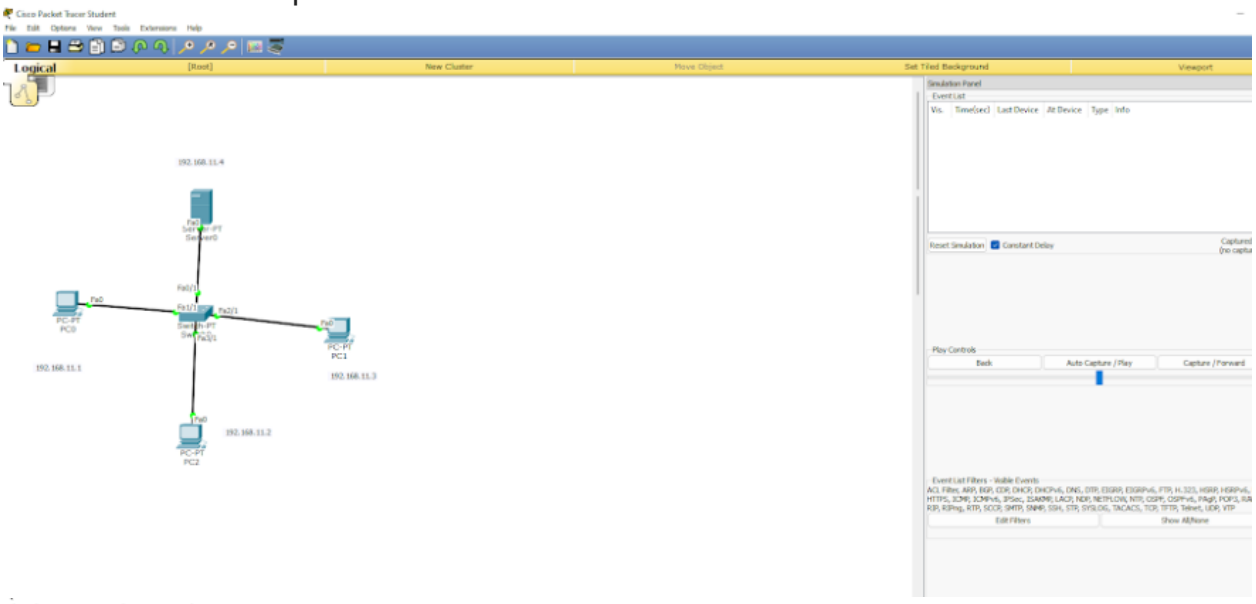
(5) Inspect packet types

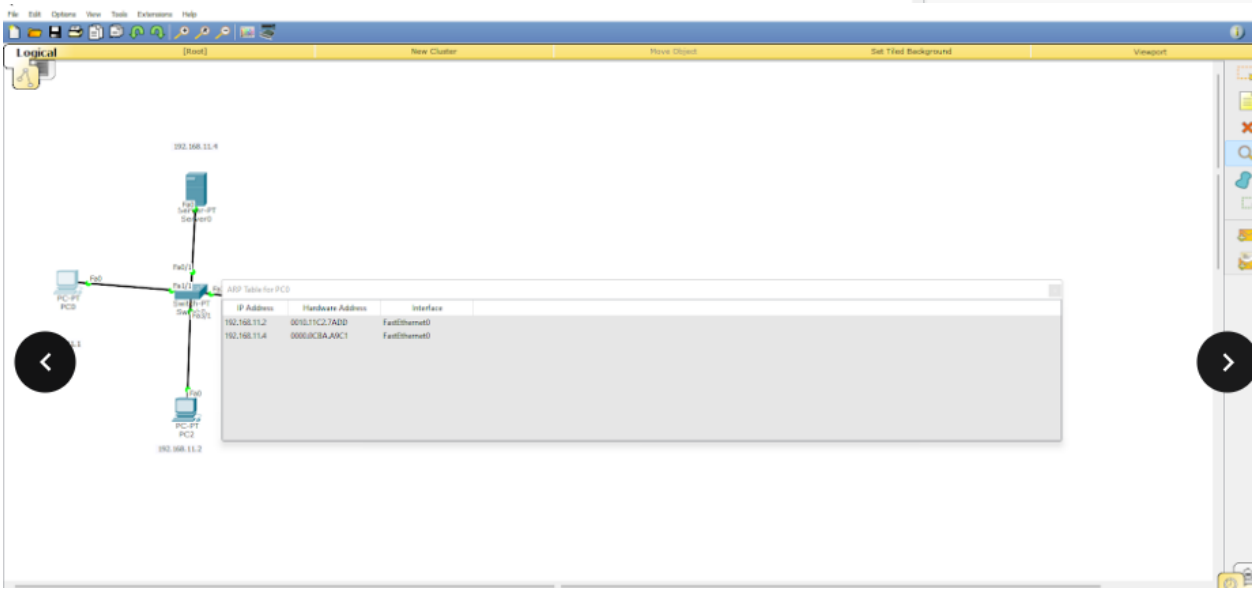
(6) Capture packet movements & observe the ping request from pc0 to the switch & the ping response to pc0

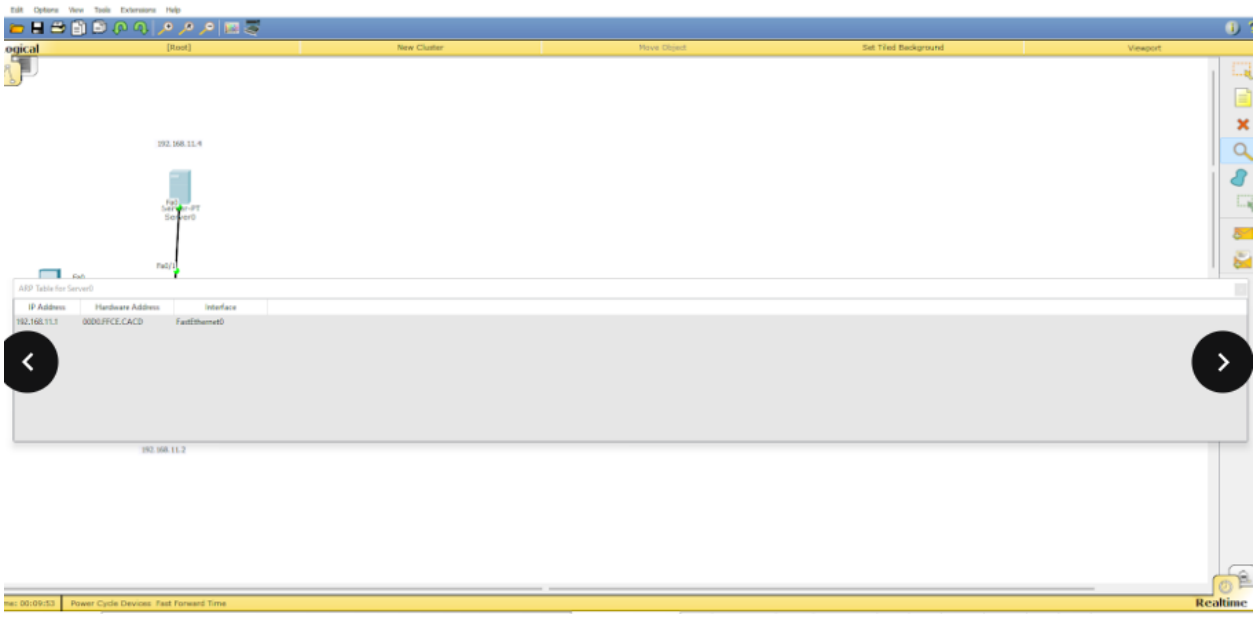
(7) Monitor ICMP packet flow.

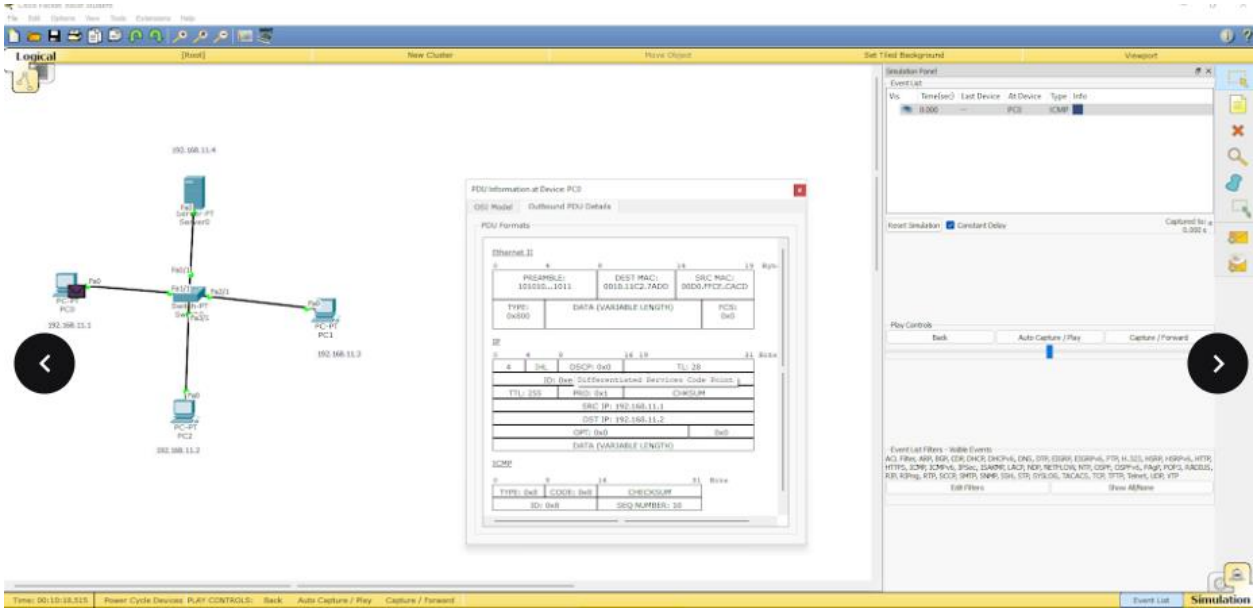


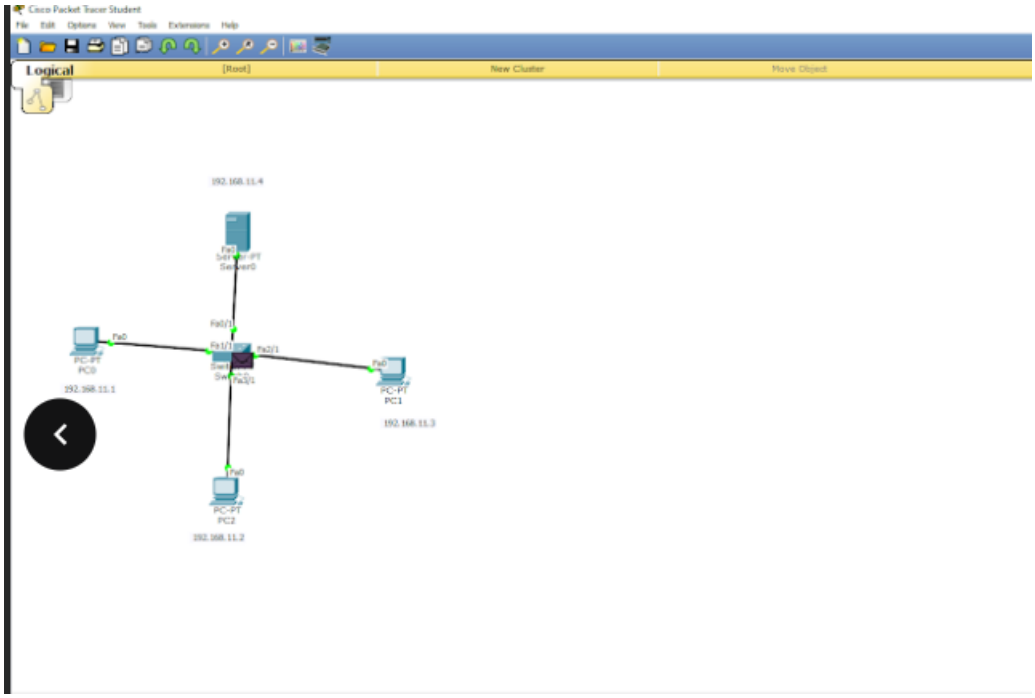
iii. Screen shots/ output:



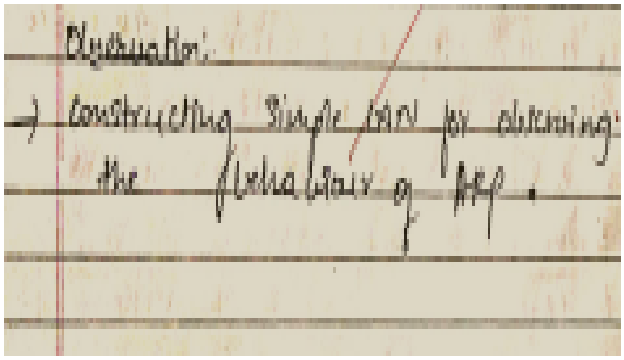




ogical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
192.168.11.4
Server0
Fa0/1
ARP Table for Server0
IP Address
Hardware Address
Interface
192.168.11.1
0000.FFCE.CACD
FastEthernet0
192.168.11.2
Power Cycle Devices Fast Forward Time
Realtime


Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
192.168.11.4
Server0
PC-PT
PC0
192.168.11.1
PC-PT
PC1
192.168.11.3
PC-PT
PC2
192.168.11.2
PDU Information at Device PC0
OSI Model
Outbound PDU Details
PDU Formats
Ethernet II
PREAMBLE: 101010...1011
DEST MAC: 0010.11C2.7AD0
SRC MAC: 0000.FFCE.CACD
TYPE: 0x800
DATA (VARIABLE LENGTH)
FCS: 0x0
IP
DSCP: 0x0
TL: 28
ID: 0x0
TTL: 255
PRO: 0x1
CHKSUM
SRC IP: 192.168.11.1
DST IP: 192.168.11.2
OPT: 0x0
DATA (VARIABLE LENGTH)
ICMP
TYPE: 0x8
CODE: 0x0
CHECKSUM
ID: 0x8
SEQ NUMBER: 10
Simulation Panel
Event List
Vis.
Time(sec)
Last Device
At Device
Type
Info
0.000
PC0
ICMP
Reset Simulation
Constant Delay
Play Controls
Back
Auto Capture / Play
Capture / Forward
Edit Filters
Show All/None
Power Cycle Devices PLAY CONTROLS: Back Auto Capture / Play Capture / Forward
Event List
Simulation




iv.Observation:



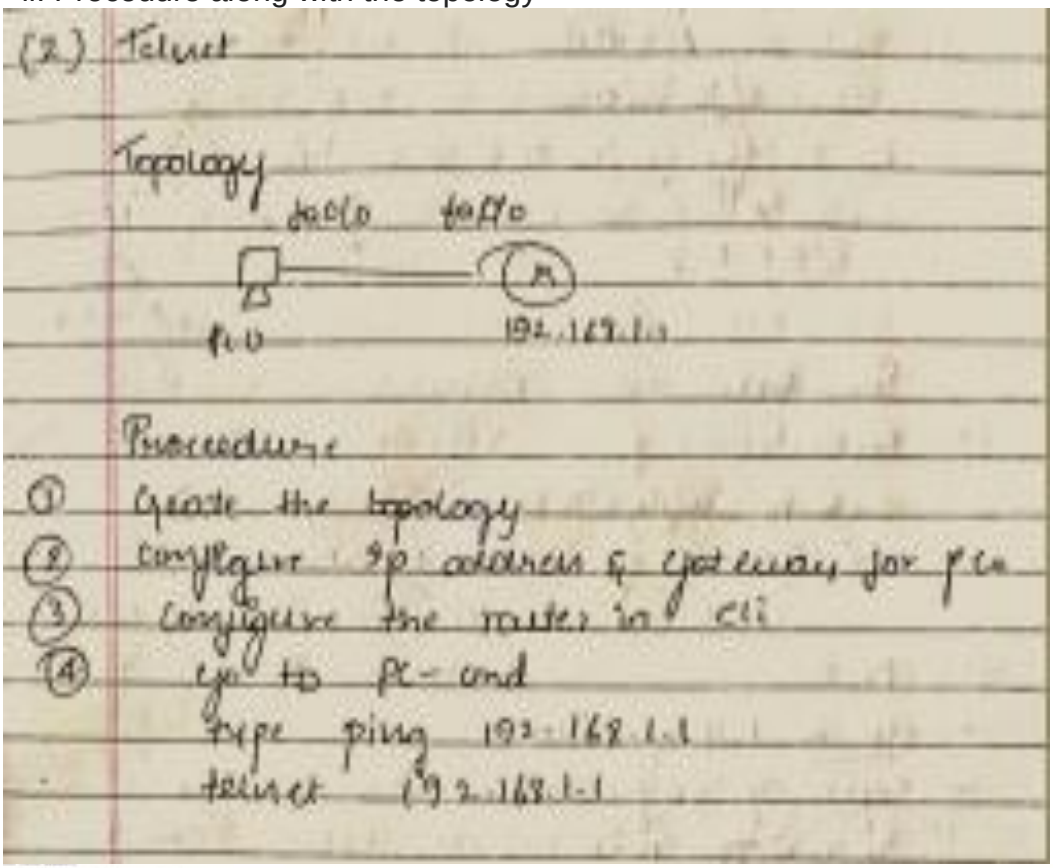
Observation:

→ Constructing Simple LAN for obtaining the functionality of ARP.

**Program 7**

i. Aim of the program: Demonstrating TELNET

ii. Procedure along with the topology



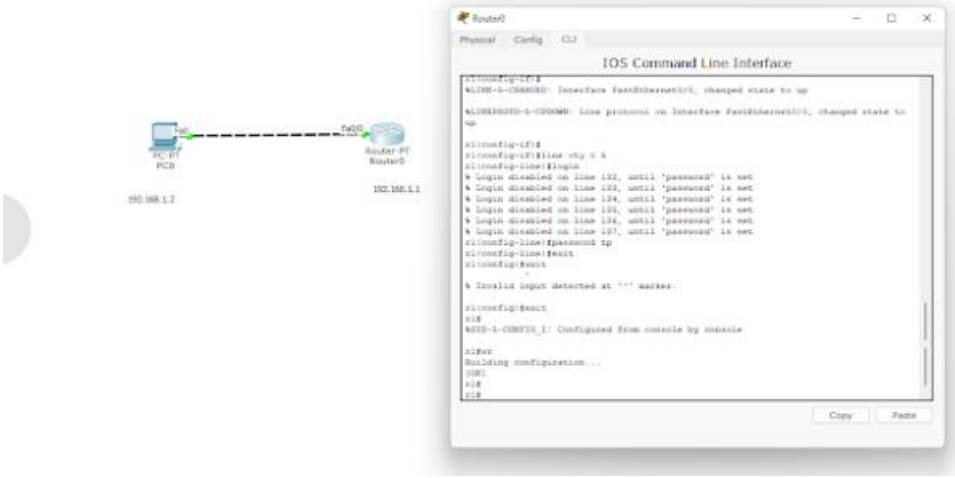
(2) Telnet

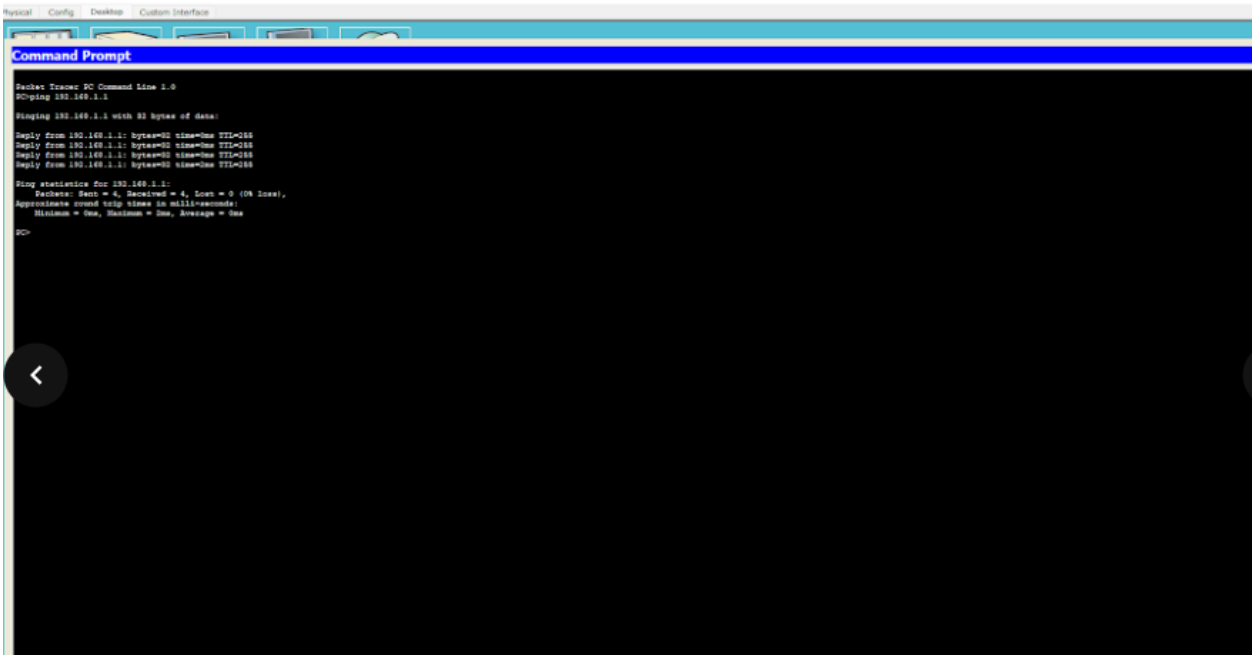
Topology

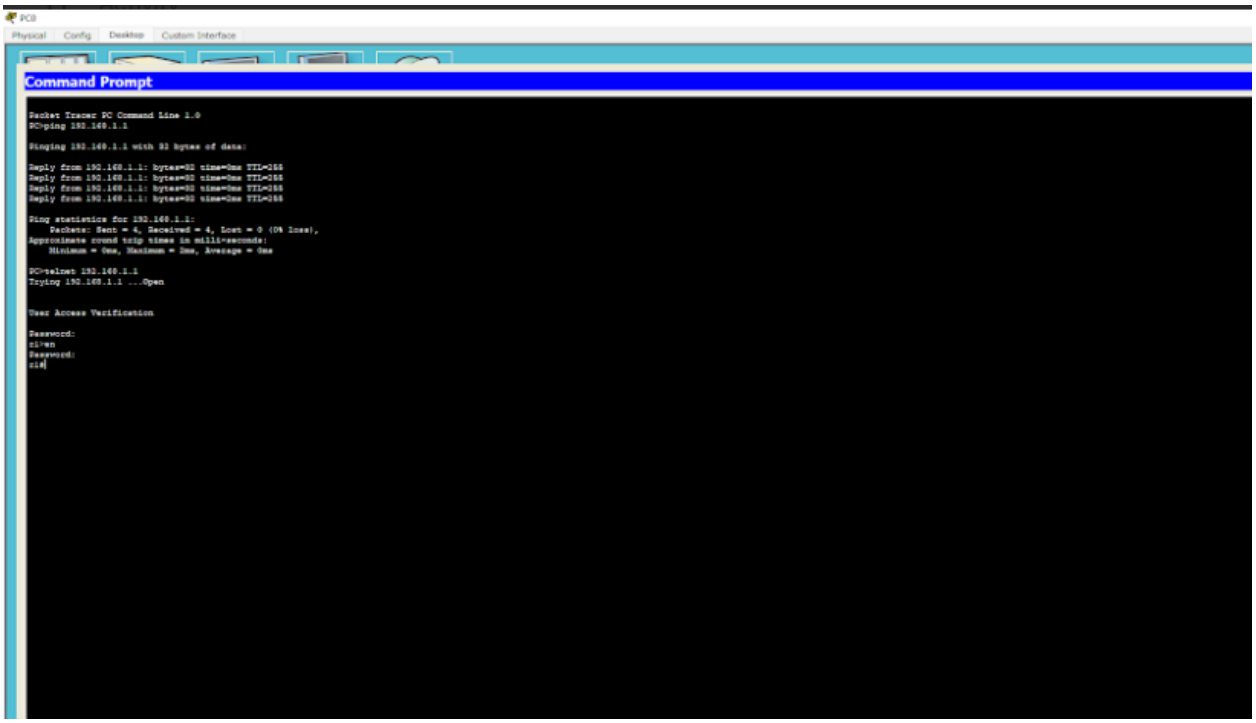


Procedure

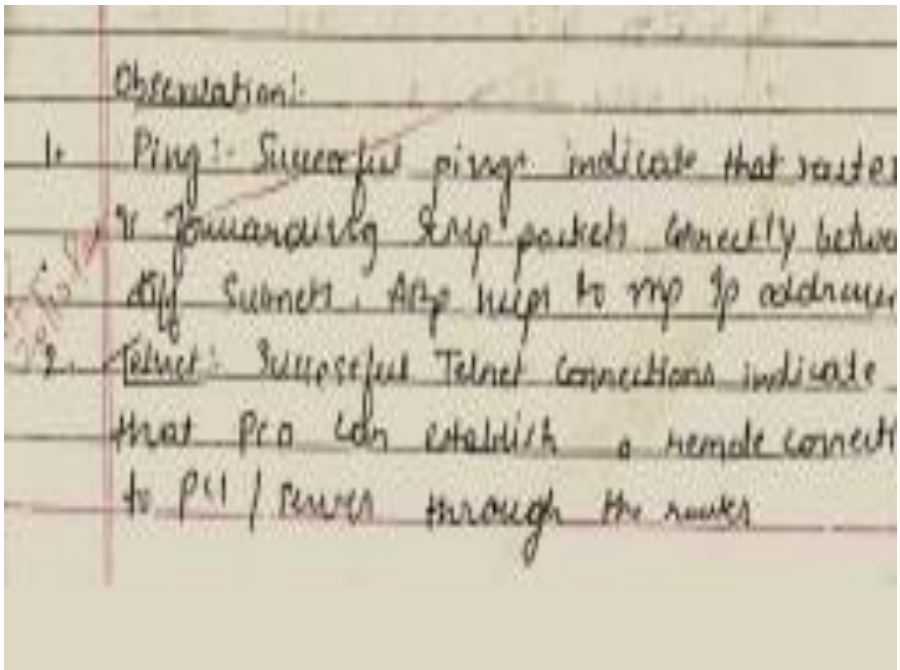
1. Create the topology
2. Configure ip address & gateway for PC
3. Configure the router in CLI
4. go to PC - cmd
   type ping 192.168.1.1
   telnet 192.168.1.1




Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 192.168.1.1
Pinging 192.168.1.1 with 32 bytes of data:
Reply from 192.168.1.1: bytes=32 time=0ms TTL=255
Reply from 192.168.1.1: bytes=32 time=0ms TTL=255
Reply from 192.168.1.1: bytes=32 time=0ms TTL=255
Reply from 192.168.1.1: bytes=32 time=0ms 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
PC>

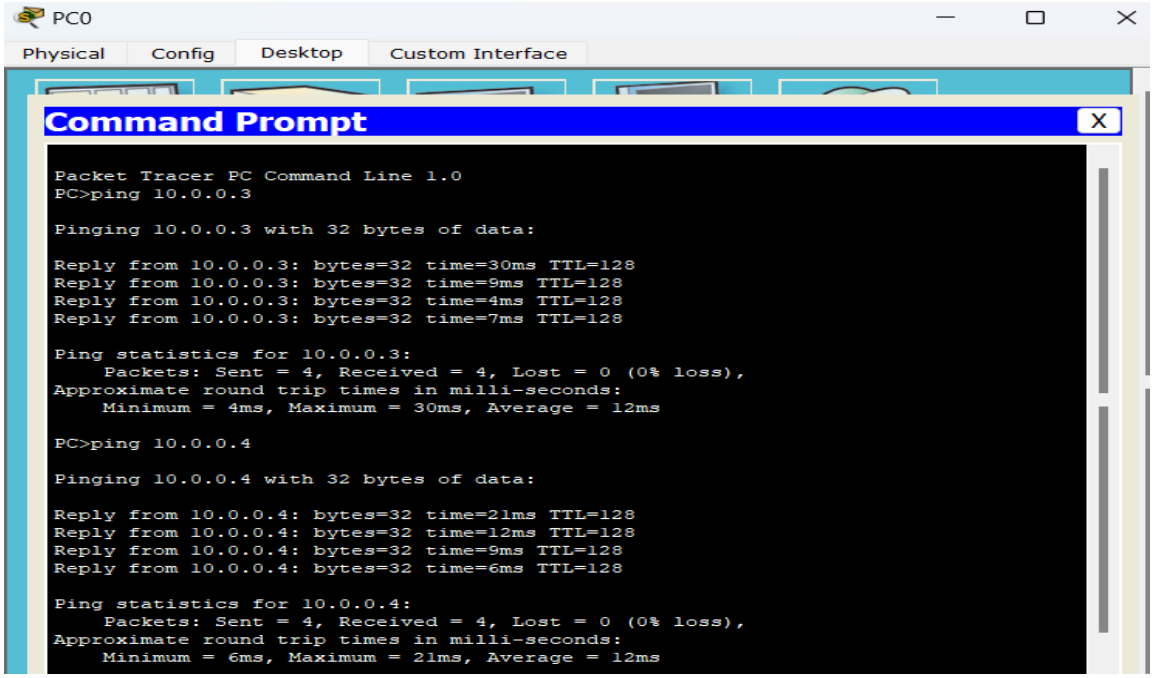

PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 192.168.1.1
Pinging 192.168.1.1 with 32 bytes of data:
Reply from 192.168.1.1: bytes=32 time=0ms TTL=255
Reply from 192.168.1.1: bytes=32 time=0ms TTL=255
Reply from 192.168.1.1: bytes=32 time=0ms TTL=255
Reply from 192.168.1.1: bytes=32 time=0ms 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
PC>telnet 192.168.1.1
Trying 192.168.1.1 ...Open
User Access Verification
Password:
ciln>en
Password:
cin#

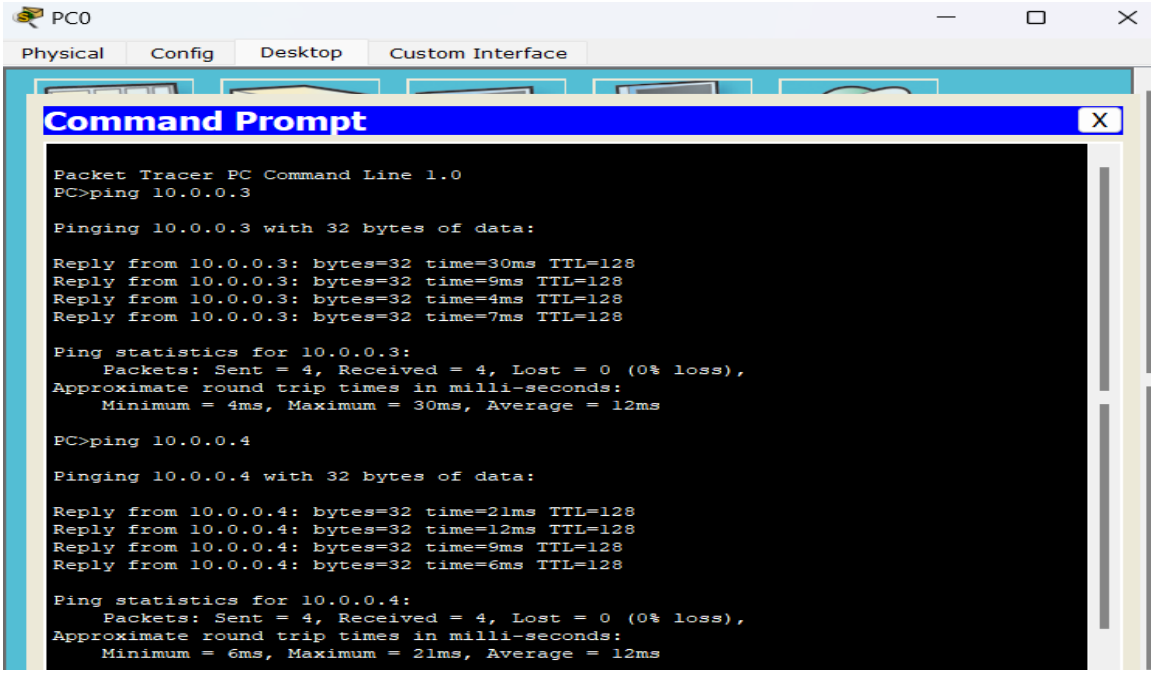
**iv.Observation:**



Observation:

1. Ping: Successful pings indicate that routers are forwarding ICMP packets correctly between different subnets. ARP helps to map IP addresses.
2. Telnet: Successful Telnet connections indicate that PC0 can establish a remote connection to PC1 / Server through the router






## CYCLE - 2

**Program 13:**

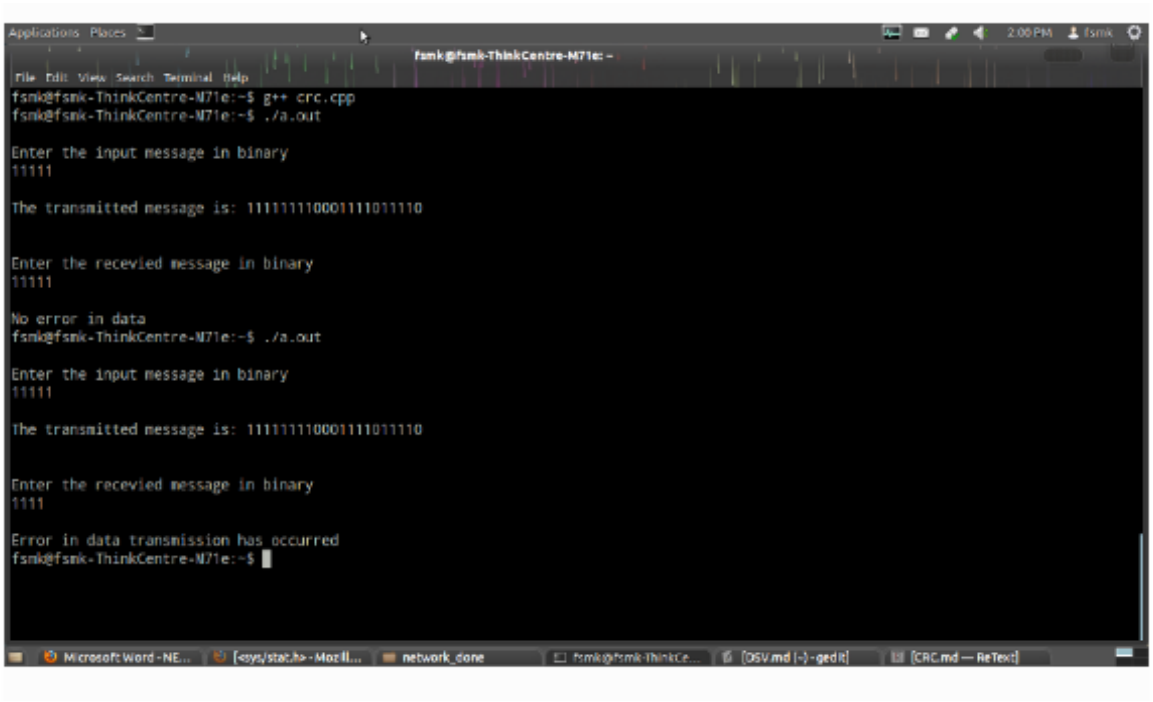
**Aim:** Write a program for error detecting code using CRC-CCITT (16-bits)

**Program:**

```
#include <stdio.h>
#include <string.h>

// Function to compute CRC
int crc(char *ip, char *op, char *poly, int mode) {
    strcpy(op, ip); // Copy input to output
    if (mode) {
        // Append zeros to the message
        for (int i = 1; i < strlen(poly); i++) {
            strcat(op, "0");
        }
    }
```

```
    // Perform division
    for (int i = 0; i <= strlen(op) - strlen(poly); i++) {
        if (op[i] == '1') {
            for (int j = 0; j < strlen(poly); j++) {
                op[i + j] = (op[i + j] == poly[j]) ? '0' : '1';
            }
        }
    }

    // Check if remainder contains only zeros
    for (int i = 0; i < strlen(op); i++) {
        if (op[i] == '1') {
            return 0; // Error detected
        }
    }
    return 1; // No error
}

int main() {
    char ip[50], op[50], recv[50];
    char poly[] = "100000111"; // Example CRC-8 polynomial

    // Input message
    printf("Enter the input message: ");
    scanf("%s", ip);

    // Compute transmitted message
    crc(ip, op, poly, 1);
    printf("Transmitted message: %s%s\n", ip, op + strlen(ip));

    // Simulate received message
    printf("Enter received message: ");
    scanf("%s", recv);

    // Check for errors
    if (crc(recv, op, poly, 0)) {
        printf("No error in data transmission\n");
    } else {
        printf("Error in data transmission\n");
    }

    return 0;
}
```

Output:




Observation:

LAB-6

Cyclic Redundancy Check [CRC] Algorithm :

```
#include <stdio.h>
#include <string.h>

int crc(char *ip, char *op, char *poly,
                    int mode){
    strcpy(op, ip);
    if (mode) {
        for (int i=1; i<strlen(poly); i++){
            strcat(op,"0")
    }   }
for (int i=0; i<strlen(ip); i++) {
    if (op[i]=='1') {
        for (int j=0; j<strlen(poly); j++) {
            if (op[i+j]==poly[j]) {
                op[i+j]='0';
            }
            else {
                op[i+j]='1'; } } } }

for (int i=0; i<strlen(op); i++) {
    if (op[i]=='1') {
        return 0;
    }
}
return 1;
}
```

```
int main() {
    char ip[50], op[50], recv[50];
    char poly[] = "10001000000100001";
    printf("Enter the i/p message :");
    scanf("%s", ip);
    crc(ip, op, poly, 1);
    printf("Transmitted message : %s %s\n",
            ip, op + strlen(ip));
    printf("Enter recieved message :");
    scanf("%s", recv);

    if (crc(recv, op, poly, 0)) {
        printf("No error in data\n");
    }
    else {
        printf("Error in data transmission\n");
    }
    return 0;
}
```

O/p:-

Enter the ip message : 11111
transmitted message : 11111 00011110
11110
Enter Recieved message : 111
Error in data transmission occurred

**Program 14:**

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

Program:

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <unistd.h>
using namespace std;

#define NOF_PACKETS 10

int rand_num(int a) {
    int rn = (rand() % 10) % a;
    return rn == 0 ? 1 : rn;
}

int main() {
    int packet_sz[NOF_PACKETS], i, clk, b_size, o_rate, p_sz_rm = 0, p_time, op;

    // Initialize random seed
    srand(time(0));

    // Generate random packet sizes
    for (i = 0; i < NOF_PACKETS; ++i)
        packet_sz[i] = rand_num(6) * 10;

    // Display packet sizes
    for (i = 0; i < NOF_PACKETS; ++i)
        cout << "\nPacket[" << i << "]: " << packet_sz[i] << " bytes\t";

    // Input output rate and bucket size
    cout << "\nEnter the Output rate: ";
    cin >> o_rate;
    cout << "Enter the Bucket Size: ";
    cin >> b_size;

    // Process each packet
    for (i = 0; i < NOF_PACKETS; ++i) {
        if ((packet_sz[i] + p_sz_rm) > b_size) {
            if (packet_sz[i] > b_size) // Packet size larger than bucket
                cout << "\n\nIncoming packet size (" << packet_sz[i]
                    << " bytes) is greater than bucket capacity ("
                    << b_size << " bytes) - PACKET REJECTED";
            else
                cout << "\n\nBucket capacity exceeded - PACKETS REJECTED!!";
```

```
        } else {
            p_sz_rm += packet_sz[i];
            cout << "\n\nIncoming Packet size: " << packet_sz[i];
            cout << "\nBytes remaining to Transmit: " << p_sz_rm;

            p_time = rand_num(4) * 10;
            cout << "\nTime left for transmission: " << p_time << " units";

            // Transmission simulation
            for (clk = 10; clk <= p_time; clk += 10) {
                sleep(1); // Simulate time passing
                if (p_sz_rm > 0) {
                    if (p_sz_rm <= o_rate) { // Check if remaining size is less than or equal to output rate
                        op = p_sz_rm;
                        p_sz_rm = 0;
                    } else {
                        op = o_rate;
                        p_sz_rm -= o_rate;
                    }
                    cout << "\nPacket of size " << op << " Transmitted";
                    cout << "\tBytes Remaining to Transmit: " << p_sz_rm;
                } else {
                    cout << "\nTime left for transmission: " << (p_time - clk)
                         << " units";
                    cout << "\nNo packets to transmit!!";
                    break;
                }
            }
        }
    }
    return 0;
}
```

Output:

```
Packet[0]: 30 bytes
Packet[1]: 20 bytes
Packet[2]: 10 bytes
Packet[3]: 50 bytes
Packet[4]: 20 bytes
Packet[5]: 30 bytes
Packet[6]: 60 bytes
Packet[7]: 40 bytes
Packet[8]: 50 bytes
Packet[9]: 20 bytes

Enter the Output rate: 25
```

```
Enter the Bucket Size: 60

Incoming Packet size: 30
Bytes remaining to Transmit: 30
Time left for transmission: 30 units
Packet of size 25 Transmitted   Bytes Remaining to Transmit: 5
Packet of size 5 Transmitted   Bytes Remaining to Transmit: 0

Incoming Packet size: 20
Bytes remaining to Transmit: 20
Time left for transmission: 20 units
Packet of size 20 Transmitted   Bytes Remaining to Transmit: 0

Incoming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 20 units
Packet of size 10 Transmitted   Bytes Remaining to Transmit: 0

Incoming packet size (50 bytes) is greater than bucket capacity (60 bytes) - PACKET REJECTED

Incoming Packet size: 20
Bytes remaining to Transmit: 20
Time left for transmission: 30 units
Packet of size 20 Transmitted   Bytes Remaining to Transmit: 0

Incoming Packet size: 30
Bytes remaining to Transmit: 30
Time left for transmission: 40 units
Packet of size 25 Transmitted   Bytes Remaining to Transmit: 5
Packet of size 5 Transmitted   Bytes Remaining to Transmit: 0

Incoming Packet size: 60
Bytes remaining to Transmit: 60
Time left for transmission: 30 units
Packet of size 25 Transmitted   Bytes Remaining to Transmit: 35
Packet of size 25 Transmitted   Bytes Remaining to Transmit: 10
Packet of size 10 Transmitted   Bytes Remaining to Transmit: 0

Incoming packet size (40 bytes) is greater than bucket capacity (60 bytes) - PACKET REJECTED

Incoming packet size (50 bytes) is greater than bucket capacity (60 bytes) - PACKET REJECTED

Incoming Packet size: 20
Bytes remaining to Transmit: 20
Time left for transmission: 20 units
Packet of size 20 Transmitted   Bytes Remaining to Transmit: 0
```

Observation:

Leaky Bucket Algorithm :

```
#include <stdio.h>

#define MAX_CAPACITY 5
#define LEAK_RATE 1

void leakyBucket (int incomingPackets[], int
                numPackets) {
    int bucket = 0;
    int time = 0;

    for (int i = 0; i < NumPackets ; i++) {
        printf(" Time %d : Incoming packet = %d
                \n", time, incomingPackets[i]);

        if (bucket + incomingPackets[i] <= MAX_CAPA
                CITY) {
            bucket += incomingPackets[i];
            printf(" Bucket : %d Packets Accepted
                \n", bucket);
        }
        else {
            int droppedPackets = incomingPackets[i] -
                (MAX_CAPACITY - bucket);
            bucket = MAX_CAPACITY;
            printf(" Bucket : %d packets Overflow
                %d packets dropped \n", bucket,
                droppedPackets);
        }
```

```
bucket -= LEAK_RATE;
if (bucket < 0) bucket = 0;
printf("Bucket leaks: %d", bucket);

time++;
printf("\n");
}
}

int main() {
int incomingPackets[] = {2, 3, 4, 1, 2, 3, 1, 5, 2, 1, 3};
int numPackets = sizeof(incomingPackets) / sizeof(incomingPackets[0]);

printf("Leaky Bucket Algorithm");

leakyBucket(incomingPackets, numPackets);
return 0;
}
```

O/P:

Leaky Bucket Algorithm

Time 0: Incoming packet = 2
Bucket : 2 packets (accepted)
Bucket leaks : 1 packet left in the bucket

Time 1 : Incoming packet = 3
Bucket : 4 packets (accepted)
Bucket leaks : 3 packets left in bucket.

Time 2 : Incoming packet = 4
Bucket : 5 packets (overflow) 2 packets are dropped
Bucket leaks : 4 packets left in the bucket

Time 3 : Incoming packet = 1
Bucket : 5 packets (accepted)
Bucket leaks 4 packets left in the bucket

Time 4 : Incoming packet = 2
Bucket : 5 packets (overflow) 1 packet dropped
Bucket leaks 4 packets left in bucket

Time 5 : Incoming packet = 3
Bucket : 5 packets (accepted)
Bucket leaks : 4 packets left in the bucket.

**Program 15:**

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

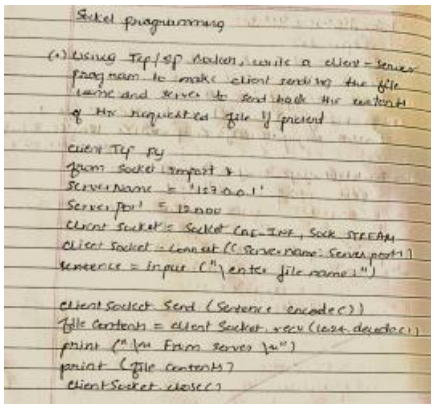
Program:

```
   from socket import *

serverName = '127.0.0.1'
serverPort = 12000

# Create a socket
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))

# Get the file name from the user
sentence = input("\nEnter file name: ")

# Send the file name to the server
clientSocket.send(sentence.encode())

# Receive the file contents from the server
filecontents = clientSocket.recv(1024).decode()

# Print the contents received from the server
print('\nFrom Server:\n')
print(filecontents)

# Close the socket connection
clientSocket.close()
```

Output:

Server is Online.
Requesting for file : test.txt
Request sent.
Client is connected to server
Enter file name : test.txt
Received Response
Hello World.

Observation:


Socket programming

(1) Using Tcp/ip sockets, write a client-server program to make client sending the file name and server to send back the contents of the requested file if present

client.Tcp.py
from socket import *
ServerName = '127.0.0.1'
ServerPort = 12000
clientSocket = Socket(AF_INET, SOCK_STREAM)
clientSocket.connect((ServerName, ServerPort))
sentence = input("enter file name:")

clientSocket.send(sentence.encode())
fileContents = clientSocket.recv(1024.decode())
print("\n From server \n")
print(fileContents)
clientSocket.close()

Program 15:

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

Program:

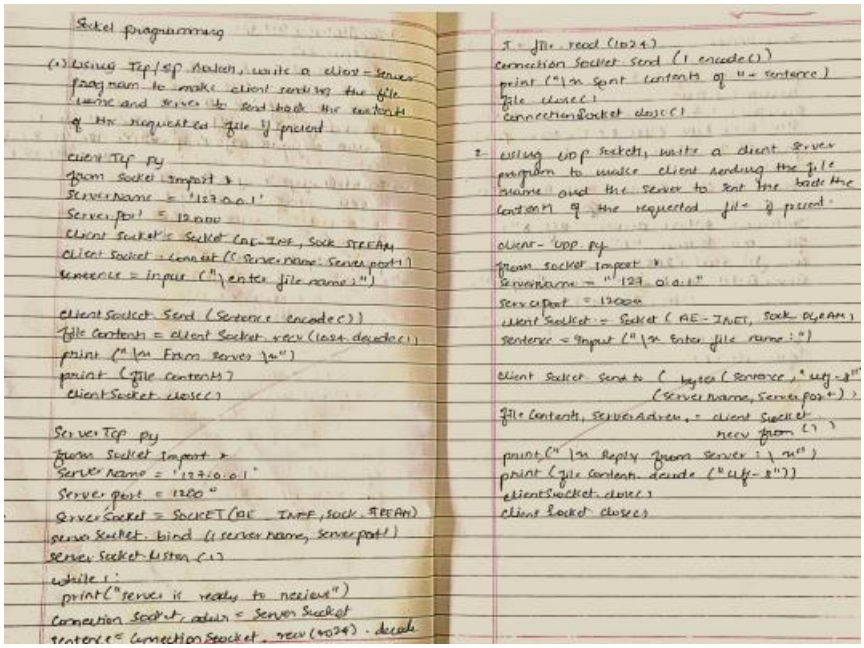
```
from socket import *

serverName = "127.0.0.1"
serverPort = 12000

# Create the server socket
serverSocket = socket(AF_INET, SOCK_STREAM)

# Bind the server to the specified address and port
serverSocket.bind((serverName, serverPort))

# Start listening for client connections
serverSocket.listen(1)

print("The server is ready to receive")

while True:
    # Accept a connection from the client
    connectionSocket, addr = serverSocket.accept()

    # Receive the file name from the client
    sentence = connectionSocket.recv(1024).decode()

    # Open the file
    try:
        with open(sentence, 'r') as file:
            # Read the file contents
            fileContents = file.read(1024)
            # Send the file contents back to the client
```

```
        connectionSocket.send(fileContents.encode())
        print(f"\nSent contents of {sentence}")
    except FileNotFoundError:
        # If the file is not found, send an error message
        connectionSocket.send(b"File not found.")

    # Close the connection with the client
    connectionSocket.close()
```

OUTPUT:

Server is Online.
Requesting for file : test.txt
Request sent.
Client is connected to server
Enter file name : test.txt
Received Response
Hello World.

Observation:



Socket programming

(1) Using Tcp/Ip Sockets, write a client-server program to make client sending the file name and server to send back the contents of the requested file if present.

client_Tcp.py
from socket import *
ServerName = '127.0.0.1'
ServerPort = 12000
clientSocket = Socket(AF_INET, SOCK_STREAM)
clientSocket.connect((ServerName, ServerPort))
sentence = input("\enter file name:")

clientSocket.send(Sentence.encode())
fileContents = clientSocket.recv(1024.decode())
print("\n From server \n")
print(fileContents)
clientSocket.close()

Server_Tcp.py
from Socket import *
ServerName = '127.0.0.1'
ServerPort = 1200"
serverSocket = SOCKET(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
while 1:
print("server is ready to receive")
connectionSocket, addr = serverSocket
sentence = connectionSocket.recv(1024).decode

f = file.read(1024)
connectionSocket.send(f.encode())
print("\n sent contents of " + sentence)
file.close()
connectionSocket.close()

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

client_UDP.py
from socket import *
ServerName = "127.0.0.1"
ServerPort = 12000
clientSocket = Socket(AF_INET, SOCK_DGRAM)
sentence = input("\n Enter file name:")

clientSocket.sendto(bytes(sentence, "utf-8"), (ServerName, ServerPort))
fileContents, serverAddress = clientSocket.recvfrom()

print("\n Reply from server:\n")
print(fileContents.decode("utf-8"))
clientSocket.close()
client Socket closes

**Program 16:**

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

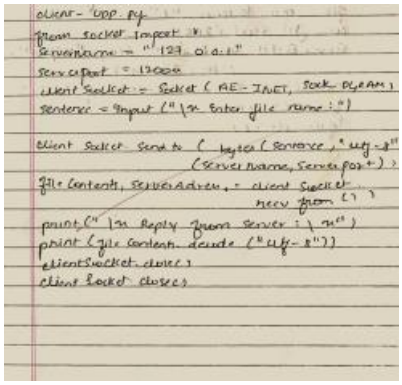
Program:

```
from socket import *

serverName = "127.0.0.1"
serverPort = 12000

clientSocket = socket(AF_INET, SOCK_DGRAM)

sentence = input("\nEnter file name: ")
clientSocket.sendto(bytes(sentence, "utf-8"), (serverName, serverPort))

filecontents, serverAddress = clientSocket.recvfrom(2048)

print('\nReply from Server:\n')
print(filecontents.decode("utf-8"))

clientSocket.close()
```

OUTPUT:

```
Server is Online.
Requesting for file : test.txt
Request sent.
Client is connected to server
Enter file name : test.txt
Received Response
Hello World.
```

Observation:



**Program 16:**

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

Program:

```
 from socket import *

serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))

print("The server is ready to receive")

while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")

    try:
        with open(sentence, "r") as file:
            con = file.read(2048)
        serverSocket.sendto(bytes(con, "utf-8"), clientAddress)
        print('\nSent contents of', sentence)
    except FileNotFoundError:
        error_msg = "Error: File not found"
        serverSocket.sendto(bytes(error_msg, "utf-8"), clientAddress)
        print("File not found:", sentence)
    except Exception as e:
        error_msg = f"Error: {str(e)}"
        serverSocket.sendto(bytes(error_msg, "utf-8"), clientAddress)
        print("Error:", str(e))
```

Output:

```
Server is Online.
Requesting for file : test.txt
Request sent.
Client is connected to server
Enter file name : test.txt
Received Response
Hello World.
```

Observation:

Server UDP.py

```
from Socket import *
Serverport = 12000
Serverpacket = Socket (AF_INET, SOCK_DGRAM)
Server socket.bind ("127.0.0.1", server_port)
print (" The server is ready to recive ")
while 1:
sentence, client Address = server Socket. recv
                                    from (2048)
sentence = sentence.decode ("utf-8")
file = open( Sentence, "r")
con = file. read (2048)
Server Socket. Send to (bytes (con, "utf-8",
                    client Address)
print ('\n Sentence contents of, 'end=" ")
print (Sentence)
file. close()
```