VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

COMPUTER NETWORKS

Submitted by

CHAITANYA N (1BM22CS076)

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 Sep
2024-Jan 2025

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 NETWORKS**" carried out by **CHAITANYA N(1BM22CS076),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of **Computer Networks Lab - (23CS5PCCON)** work prescribed for the said degree.

Sandhya A Kulkarni

Assistant Professor,
Department of CSE,
BMSCE, Bengaluru

Dr. Kavitha Sooda

Professor and Head,
Department of CSE
BMSCE, Bengaluru

INDEX

CYCLE 1

Sl. No.	Date	Experiment Title	Page No.
1	25-9-24	Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.	5-9
2	16-10-24	Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply	10-18
3	23-10-24	Configure default route, static route to the Router	19-23
4	13-11-24	Configure DHCP within a LAN and outside LAN.	24-26
5	20-11-24	Configure RIP routing Protocol in Routers	27-30
6	27-11-24	Configure OSPF routing protocol	31-35
7	20-11-24	Demonstrate the TTL/ Life of a Packet	36-41
8	18-12-24	Configure Web Server, DNS within a LAN.	42-43
9	18-12-24	To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)	44-45
10	18-12-24	To understand the operation of TELNET by accessing the router in server room from a PC in IT office.	46-48
11	18-12-24	To construct a VLAN and make the PC's communicate among a VLAN	49-50

12	18-12-24	To construct a WLAN and make the nodes communicate wirelessly	51-53

INDEX

CYCLE 2

Sl. No.	Date	Experiment Title	Page No.
1	25-12-24	Write a program for error detecting code using CRC-CCITT (16-bits)	54-57
2	25-12-24	Write a program for congestion control using Leaky bucket algorithm.	58-65
3	25-12-24	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.	66-67
4	25-12-24	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.	68-79

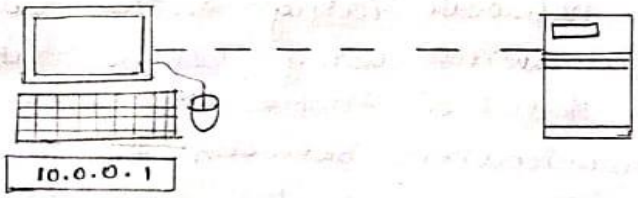
CYCLE-1

PROGRAM1:
Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

OBSERVATION

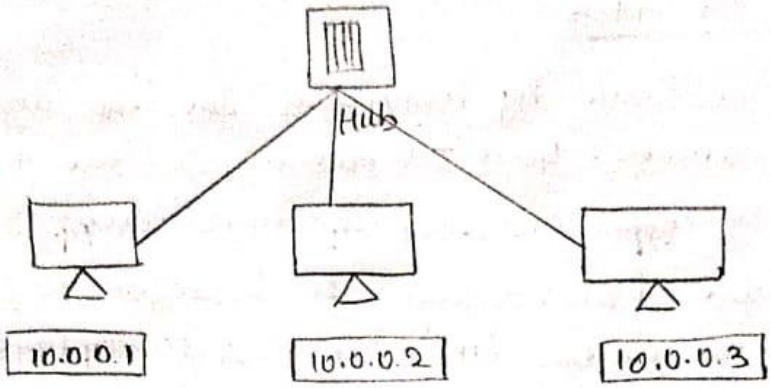
★ My First PT LAB

Created a network by connecting the end devices PC and a server. Gave IP address to the PC. Connected by copper cross-over connections. Added simple PDU message to transfer message from PC to server. Clicked on Auto Capture/Play and observed the operations.
Status showed successfully.



★ Hub - PCs.

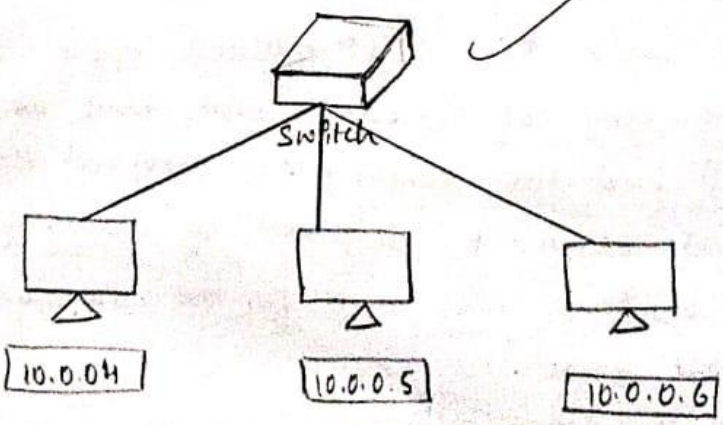
Created a local area network using HUB and end devices PCs. Connected PC to hub using Copper Straight-through cable. Gave IP address for the PCs. connected. Added a simple PDU to communicate from PC of IP address 10.0.0.1 to 10.0.0.3. The copy of message was sent to all PC connected and the correct PC accepted the message, and other PC rejected by wrong symbol. Similar observation was done for the message or acknowledgment sent.
Status shown successfully.



★ Switch - PC :-

Comm created a network using switch and PC. Connected PC to switch using Copper Straight - Through cable. Added a simple PDU to observe & message communication between two PCs. Clicked autoplay to observe the working. Initially a copy of message was sent to all devices connected to switch. Only the right device accepted the message. Now when the acknowledgment was sent back the switch transfers that message to the correct PC. Since switch has memory.

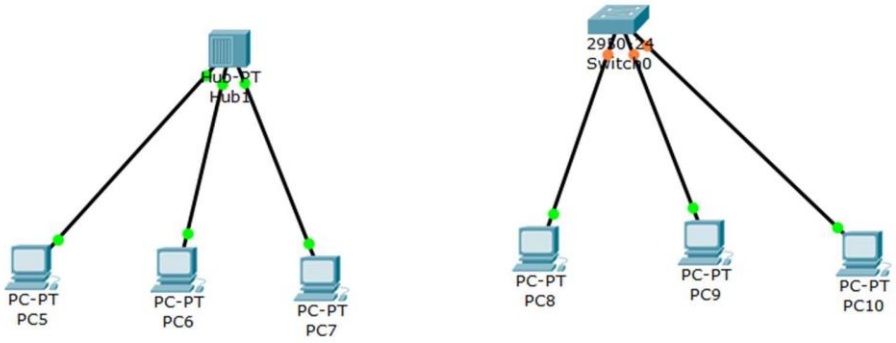
Status shown successfully.



→ Difference b/w Hub and Switch

Hub	Switch.
Hub operates on physical layer of OSI model.	Switch operates on data layer of OSI model.
Hub connects multiple pcs to single network	Switch connects multiple devices on single network.
Hub is broadcast type transmission	Switch is unicast, multicast and broadcast type transmission.

TOPOLOGY:



OUTPUT:
Pcs are connected

PROGRAM2:

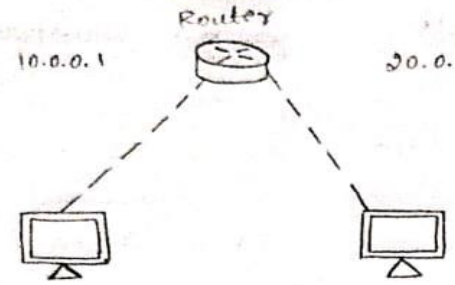
Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply

OBSERVATION:

Ex No 2a: LAB-2 = Configure IP address to 9/10/24
routers. An Explore: ping responses, destination unreachable, request timed out, reply

Aim: To connect two PC's in two different network using a router.

Topology:



Procedure:

1. Select two PC from the device type selection and place it on the workbench using Placement cursor.
2. Similarly select router (generic) and place it on the workspace.
3. Connect two devices ie PCs to router using copper cross-over cable.
4. Assigned IP address for the two PCs and the gateway.
5. Setup router configuration. An CLI

CLI:

```
Router > enable
Router # config terminal
Router (config) # interface fastethernet 0/0
Router (config-if) # ip address 10.0.0.1 255.0.0.0
Router (config-if) # no shutdown
exit
```

Router (config) # interface fastethernet 1/0
Router (config-if) # ip address 20.0.0.1 255.0.0.0
Router (config-if) # no shutdown
exit
show ip route

nodes are not Router connected to nodes through proper cable configured.

6. Click on PCs and go to Desktop and choose command line prompt, ping 20.0.0.10 to observe status of the packets.

Observation:

- The buttons on the copper wire over connection turned green indicating correct connection
- Packets were successfully transferred showing results as:

  Packets: Sent = 4, Received = 4, lost = 0

- ping 20.0.0.10

  Pinging 20.0.0.10 with 32 bytes of data:

  Ping statistics for 20.0.0.10:

  Packets: Sent = 4, Received = 4, lost = 0

  Approximate round trip times in milli-seconds:

  Minimum = 0ms, Maximum = 0ms, Average = 0ms

- Routing table results:

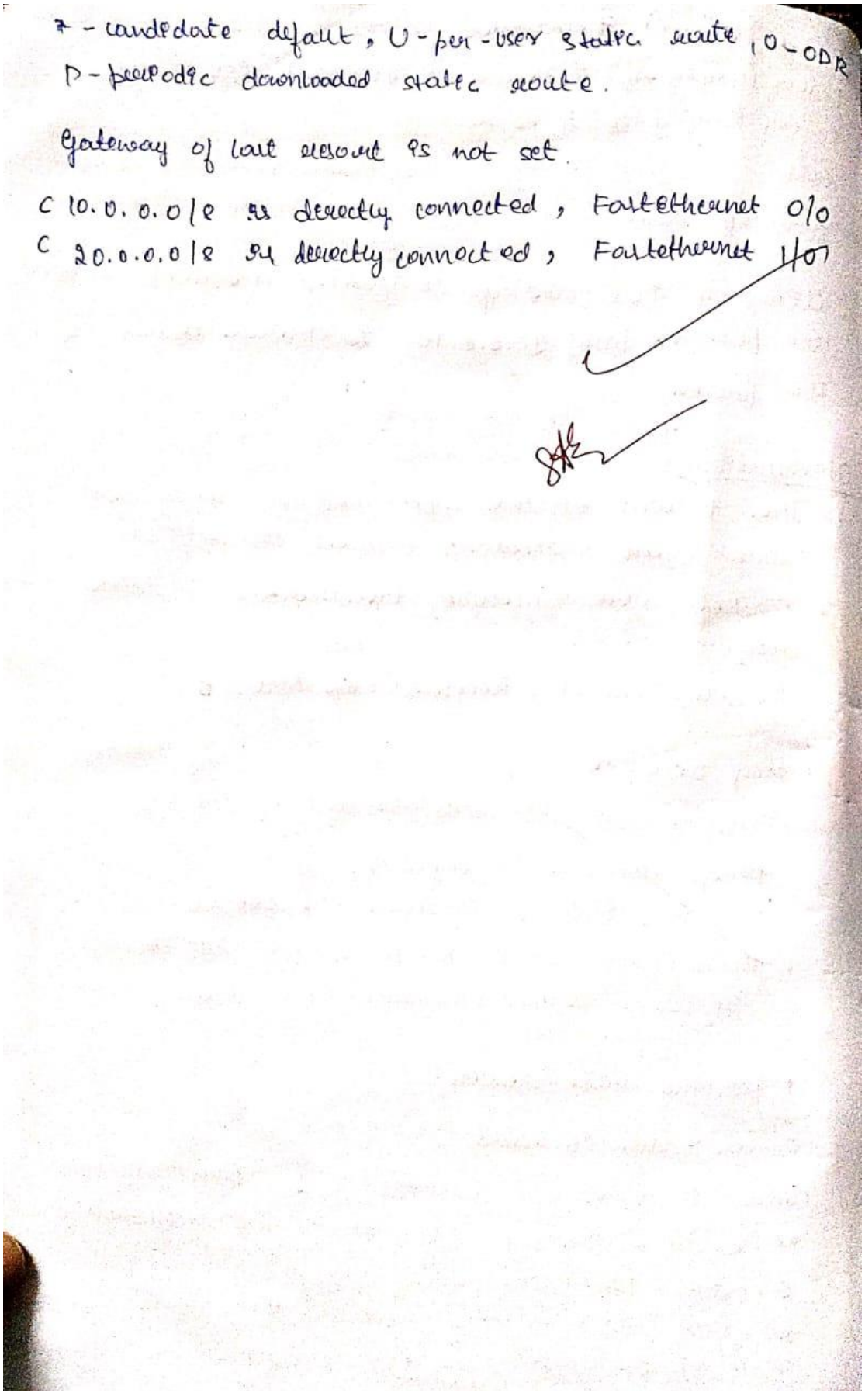
Router > show ip route

Codes: C - connected, S - static, I - IGRP, R - RIP
M - mobile, B - BGP, D - EIGRP, EX - EIGRP external,
O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF external
type 2, E - EGP i - IS-IS, L1 - IS-IS level-1,
L2 - IS-IS level-2, ia - IS-IS inter area

* - candidate default, U - per-user static route, O - ODR
P - periodic downloaded static route.

Gateway of last resort is not set.

C 10.0.0.0/8 is directly connected, FastEthernet 0/0
C 20.0.0.0/8 is directly connected, FastEthernet 1/0



Exp.: 2b

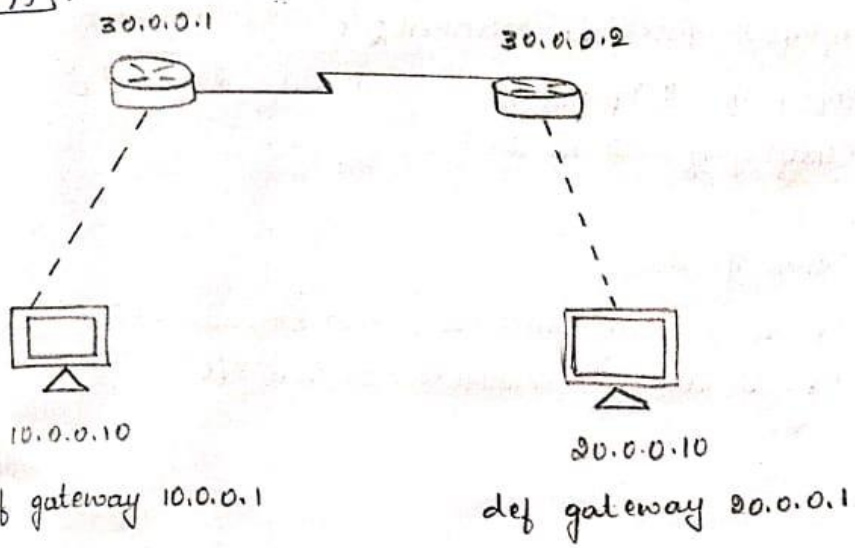
LAB-3 :

16/10/24

Q] Configure IP address to routers on packet tracer. Explore: ping responses, destination unreachable, request timed out & reply.

Aim: To connect two routers and observe the responses.

Topology :-



Procedure :

1) Select two generic routers - DT from device type selection and place it on workbench
2) Select two end devices (PC's) for and place on workbench.
3) Assign IP address 10.0.0.10 to 1st PC and 20.0.0.10 to other and set getway 10.0.0.1 and 20.0.0.1 respectively.
4) Connect PC in 10.0.0.1 IP to router 30.0.0.1 & 20.0.0.1 to 30.0.0.2 router using copper cross over cable
5) Connect 2 routers using Serial DCE cable

Router (IP 300.0.1) CLI:

Router > enable

Router # config terminal

Router (config) # interface fastethernet 0/0

Router (config-if) # ip address 10.0.0.1 255.0.0.0

Router (config-if) # no shutdown.

exit

Router (config) # interface serial 2/0

Router (config-if) # ip address 30.0.0.1 255.0.0.0

Router (config-if) # no shut

Router # show ip route

10.0.0.0/8 is directly connected, Fast Ethernet 0/0

30.0.0.0/8 is directly connected, Serial 2/0

Ping:

ping 20.0.0.10

Packets: Sent = 4, Received = 0, Lost = 4

ping 30.0.0.1

Packets: Sent = 4 Received = 4, Lost = 0

ping 30.0.0.2

Packets: Sent = 4 Received = 0, Lost = 4

Observation:

two routers are Successfully connected with nodes through proper cable.

Both routers configured.

Routers not able to communicate with those nodes.

16/10/24

Experiment 3a :

a] Configure default route, Static route to routers.

→ Same as previous experiment.

Procedure :

CLI :

Router(config)# ip route 20.0.0.0 255.0.0.0 30.0.0.2

Router # show ip route

10.0.0.0/8 is directly connected, FastEthernet 0/0
20.0.0.0/8 [1/0] via 30.0.0.2
30.0.0.0/8 is directly connected.

Ping :

ping 20.0.0.10

Packets : Sent = 4, Received = 4, lost = 0

ping 30.0.0.1

packets : Sent = 4, Received = 4, lost = 0

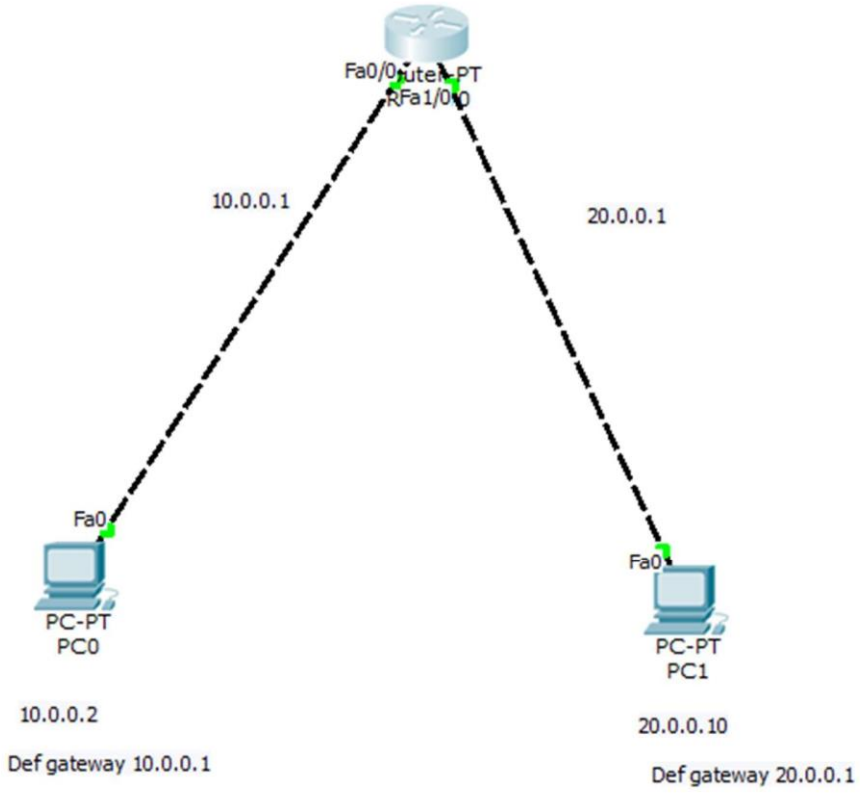
ping 30.0.0.2

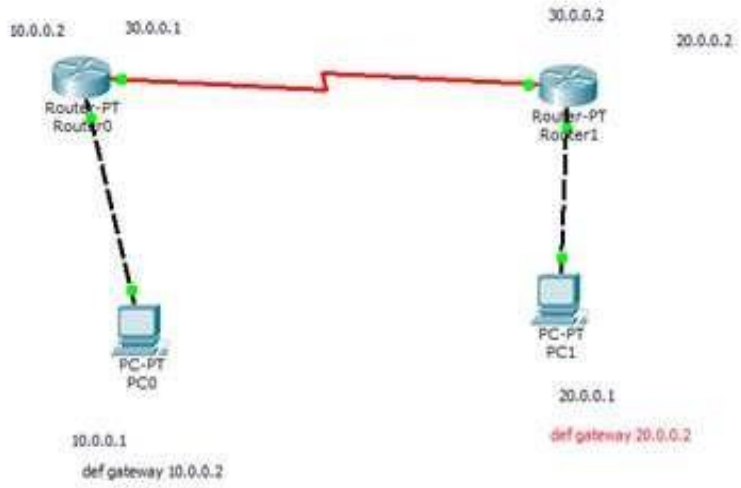
packets : Sent = 4, Received = 4, lost = 0

Observation :

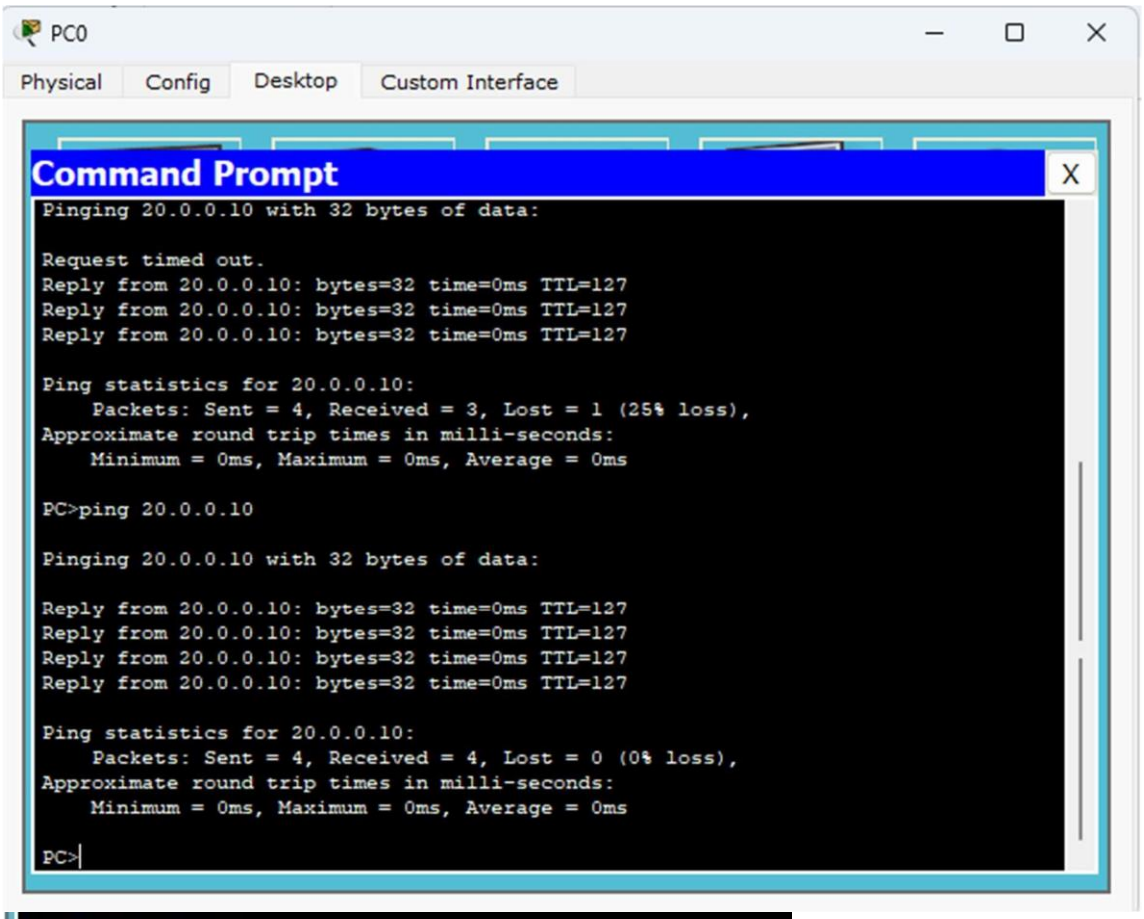
Routers can successfully communicate with their nodes.

TOPOLOGY:


Fa0/0
RFa1/0/0
10.0.0.1
20.0.0.1
Fa0
PC-PT
PC0
10.0.0.2
Def gateway 10.0.0.1
Fa0
PC-PT
PC1
20.0.0.10
Def gateway 20.0.0.1


10.0.0.2
30.0.0.1
30.0.0.2
20.0.0.2
Router-PT
Router0
Router-PT
Router1
PC-PT
PC0
PC-PT
PC1
20.0.0.1
10.0.0.1
def gateway 10.0.0.2
def gateway 20.0.0.2

OUTPUT:




```
PC>ping 10.0.0.2

Pinging 10.0.0.2 with 32 bytes of data:

Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255

Ping statistics for 10.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

PC>ping 30.0.0.1

Pinging 30.0.0.1 with 32 bytes of data:

Reply from 30.0.0.1: bytes=32 time=0ms TTL=255
Reply from 30.0.0.1: bytes=32 time=0ms TTL=255
Reply from 30.0.0.1: bytes=32 time=0ms TTL=255
Reply from 30.0.0.1: bytes=32 time=0ms TTL=255

Ping statistics for 30.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

```
PC>ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data:

Reply from 10.0.0.2: Destination host unreachable.
Reply from 10.0.0.2: Destination host unreachable.
Reply from 10.0.0.2: Destination host unreachable.
Reply from 10.0.0.2: Destination host unreachable.

Ping statistics for 20.0.0.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

PC>ping 30.0.0.2

Pinging 30.0.0.2 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 30.0.0.2:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

PC>
```

PROGRAM3: Configure default route, static route to the Router
OBSERVATION:

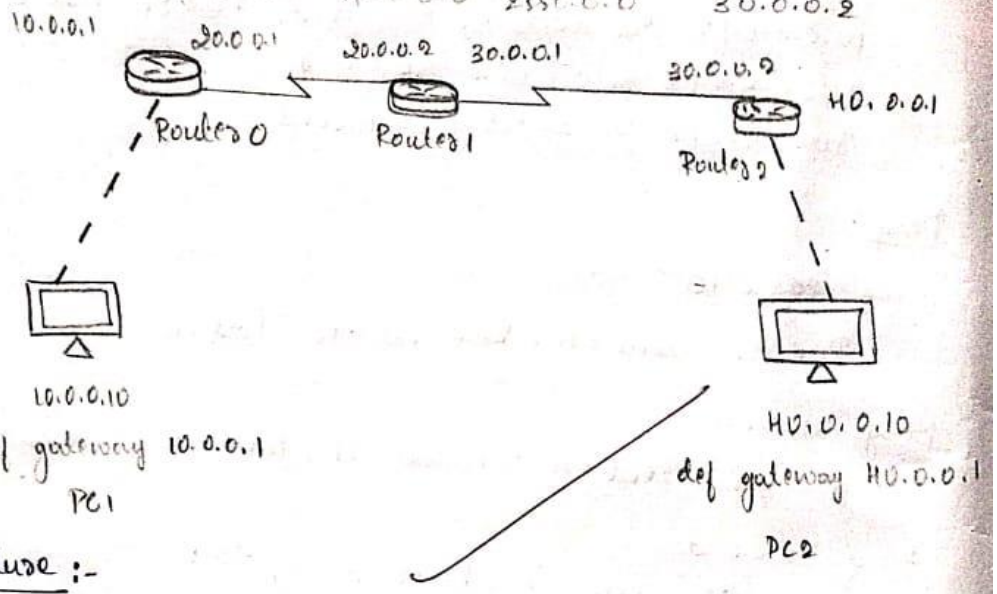
LAB-4 23/10/24

8] Configure default route, static route to router.

Aim: Two connect three routers and observe the responses.

Topology:

1) ip route 0.0.0.0 0.0.0.0 20.0.0.2
2) ip route 10.0.0.0 255.0.0.0 20.0.0.1
ip route 40.0.0.0 255.0.0.0 30.0.0.2



Procedure:-

1. Select 3 generic PT router and place on workbench.
2. Select two end devices (PC)
3. Connect 1st PC to router 0 and 2nd to router 2 through copper cross over.
4. Connect router0 and router1 & router1 and router2 using serial DCE.
5. Configure IP address 10.0.0.10 to PC1 and 40.0.0.1 to PC2.

Router 0 CLI :

i) Router > enable
Router # config terminal
Router (config) # interface fastethernet 0/0
Router (config-if) # ip address 10.0.0.1 255.0.0.0
Router (config-if) # no shutdown. 10.0.0.1 255.0.0.0
exit

Router (config) # interface serial 2/0
Router (config-if) # ip address 20.0.0.1 255.0.0.0.
Router (config-if) # no shut 30.0.0.2

Router # show ip route
10.0.0.0/8 is directly connected, FastEthernet 0/0

ii) Router (config) # ip route 0.0.0.0 0.0.0.0 20.0.0.2
Router # show ip route 30.0.0.1

C 10.0.0.0/8 is directly connected, FastEthernet 0/0
C 20.0.0.0/8 is directly connected, serial 2/0.
S* 0.0.0.0/0 [1/0] via 20.0.0.2

Router2 : same as Router 1

Router 1 CLI :-

Router (config-if) # ip ~~address~~ interface serial 2/0
Router (config-if) # ip address 20.0.0.2 255.0.0.0

Router (config) : # interface serial 3/0
Router (config-if) # ip address 30.0.0.1 255.0.0.0

Router (config) # ip route 10.0.0.0 255.0.0.0 20.0.0.1
ip route 40.0.0.0 255.0.0.0 30.0.0.2

Ping Result:

- PC1:

Ping 10.0.0.10

Packet: Sent = 4, Received = 4, Lost = 0.

- PC2:

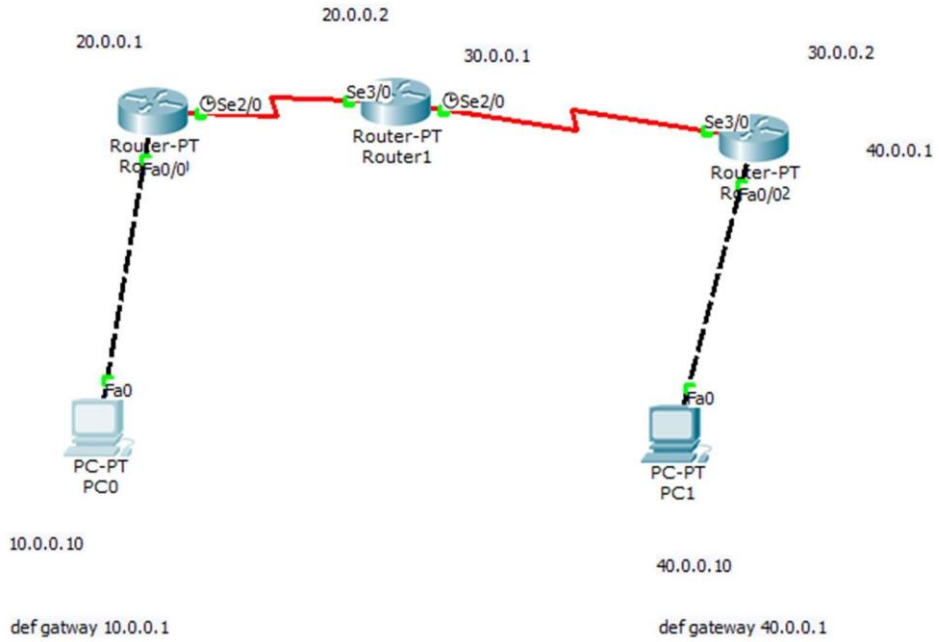
Ping 10.0.0.10

Packet: sent = 4 Received = 4, Lost = 0.

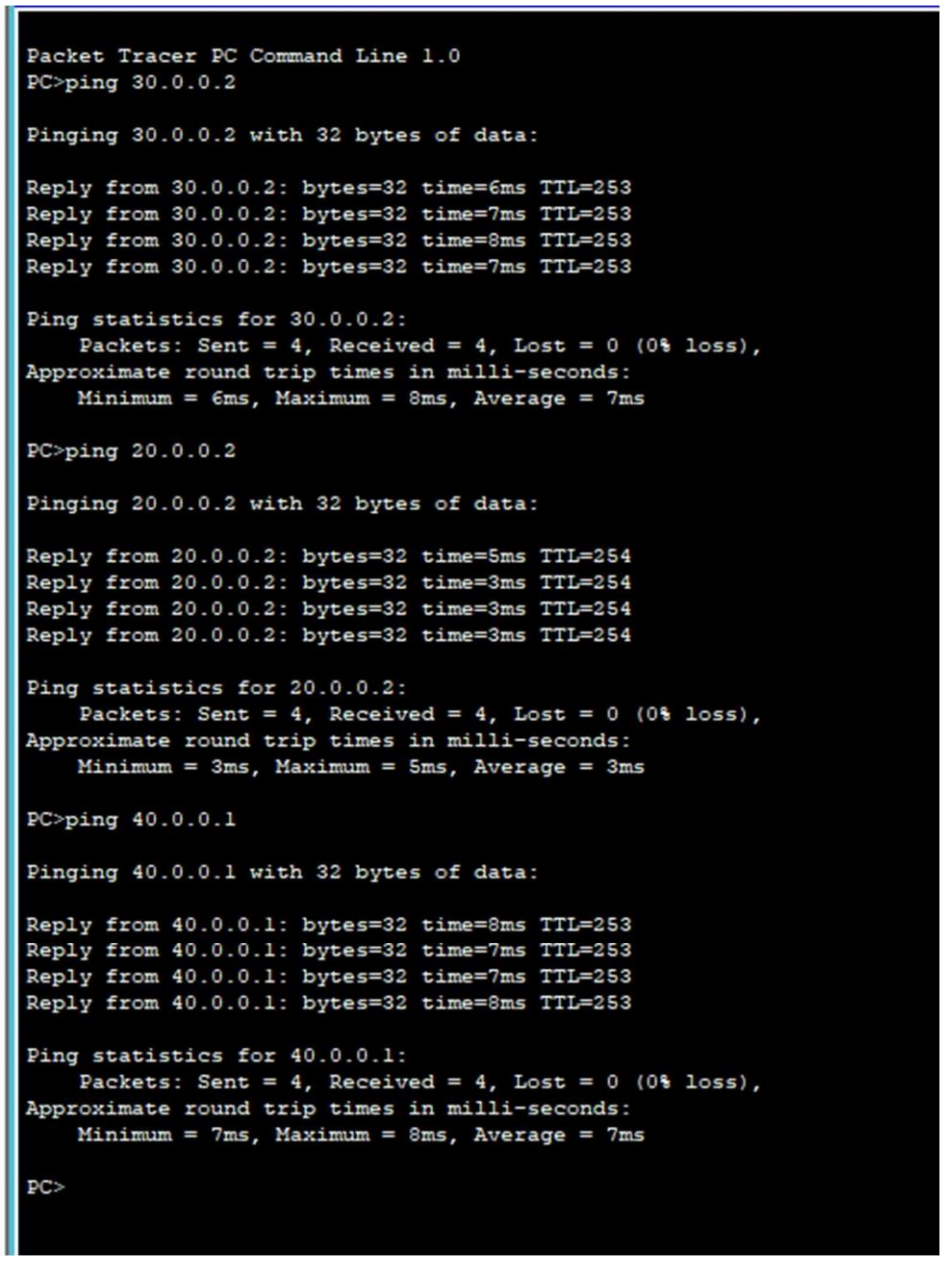
Observation:

- three routers successfully connected through proper route.
- three routers configured.
- routers are able to communicate with there route.

TOPOLOGY:



OUTPUT:


Packet Tracer PC Command Line 1.0
PC>ping 30.0.0.2
Pinging 30.0.0.2 with 32 bytes of data:
Reply from 30.0.0.2: bytes=32 time=6ms TTL=253
Reply from 30.0.0.2: bytes=32 time=7ms TTL=253
Reply from 30.0.0.2: bytes=32 time=8ms TTL=253
Reply from 30.0.0.2: bytes=32 time=7ms TTL=253
Ping statistics for 30.0.0.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 6ms, Maximum = 8ms, Average = 7ms
PC>ping 20.0.0.2
Pinging 20.0.0.2 with 32 bytes of data:
Reply from 20.0.0.2: bytes=32 time=5ms TTL=254
Reply from 20.0.0.2: bytes=32 time=3ms TTL=254
Reply from 20.0.0.2: bytes=32 time=3ms TTL=254
Reply from 20.0.0.2: bytes=32 time=3ms TTL=254
Ping statistics for 20.0.0.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 3ms, Maximum = 5ms, Average = 3ms
PC>ping 40.0.0.1
Pinging 40.0.0.1 with 32 bytes of data:
Reply from 40.0.0.1: bytes=32 time=8ms TTL=253
Reply from 40.0.0.1: bytes=32 time=7ms TTL=253
Reply from 40.0.0.1: bytes=32 time=7ms TTL=253
Reply from 40.0.0.1: bytes=32 time=8ms TTL=253
Ping statistics for 40.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 7ms, Maximum = 8ms, Average = 7ms
PC>

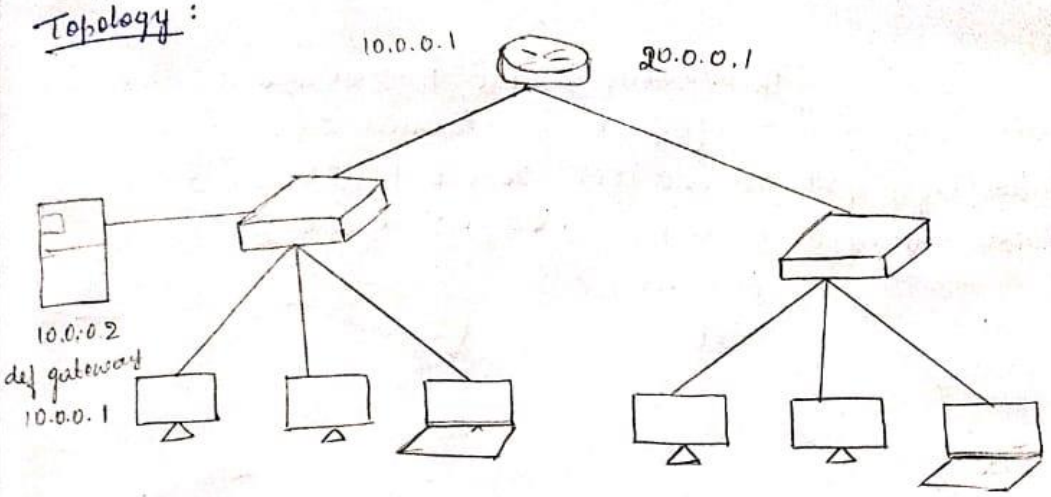
PROGRAM4: Configure DHCP within a LAN and outside LAN.
OBSERVATION:

13/11/24

6] Configure DHCP within a LAN and outside a LAN

Aim :- to configure DHCP within a LAN and outside LAN

Topology :



Procedure :

1. Select a switch from device type selection.
2. add 3 end devices on workbench and one server.
3. connect all this to switch through copper straight through cabel.
4. Repeat same topology to set other network without d/j server.
5. Place on router on workbench and connect 2 switches to it.

* In server :
  set IP address 10.0.0.1
  later set IP address 10.0.0.2 (for 2 switches set).
  Default gateway : 10.0.0.1

Router Configuration :

Router> enable
Router # config terminal

Router (config-if) # interface fastethernet 4/0
Router (config-if) # ip address 10.0.0.1 255.0.0.0
Router (config-if) # ip helper-address 20.0.0.2
no shut
exit

~~Router (config-if) # ip address 20.0.0.1 255.0.0.0~~
Router (config-if) # interface fastethernet 0/0
Router (config-if) # ip address 20.0.0.1 255.0.0.0
Router (config-if) # ip helper-address 10.0.0.2
no shut
exit.

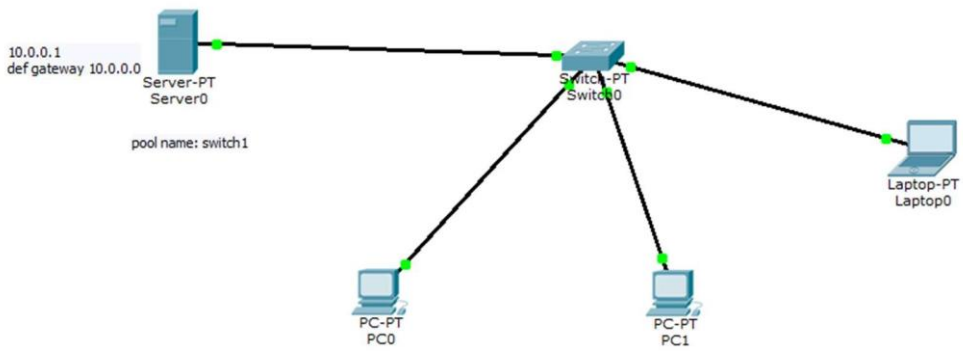
Observation?

Observation :-

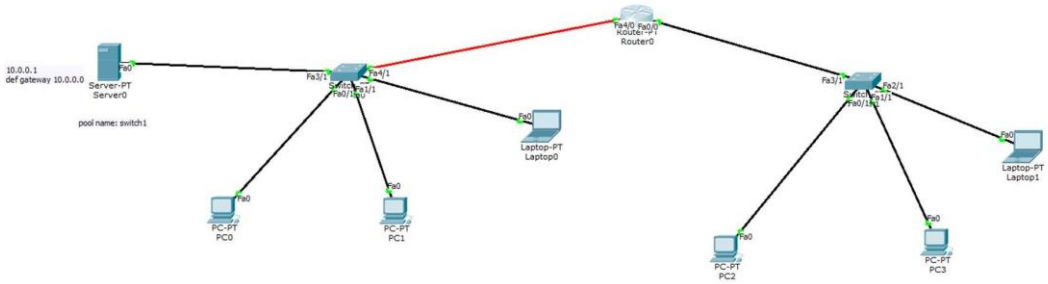
ping 10.0.0.4

packets : sent = 4, Received = 4, Lost = 0

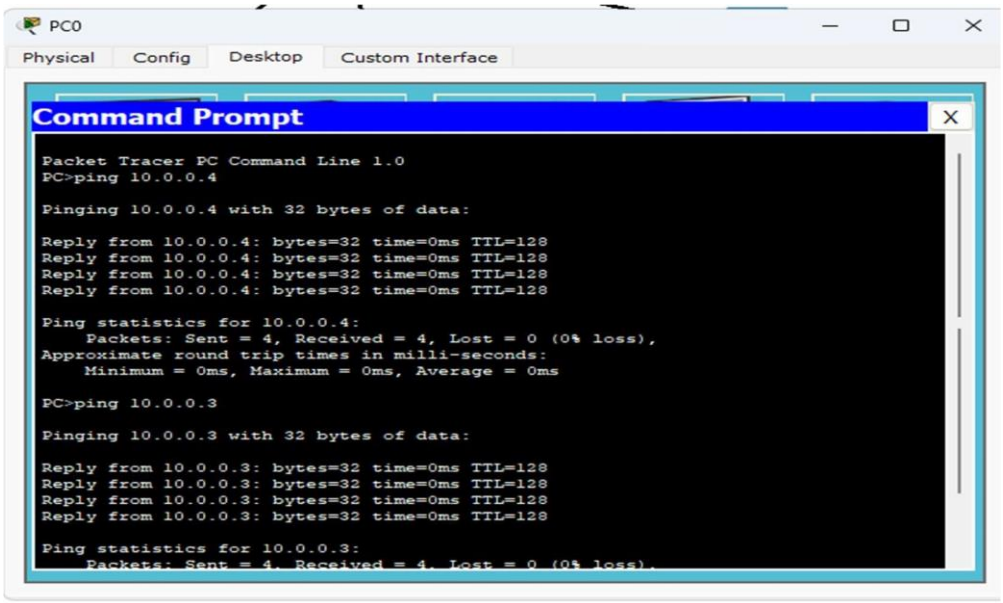
TOPOLOGY:
Within lan



Outside lan



OUTPUT:

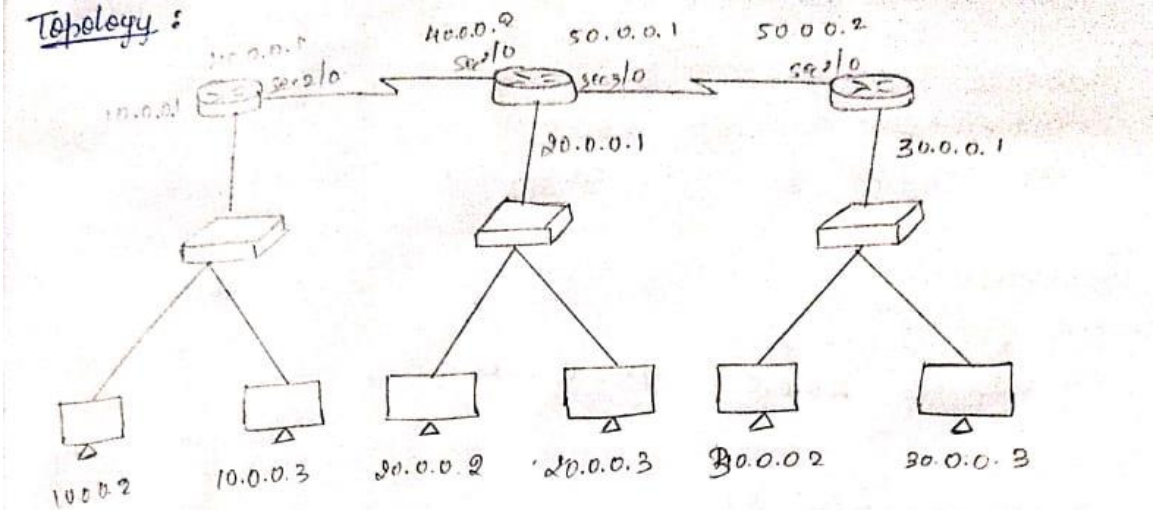


PROGRAM5: Configure RIP routing Protocol in Routers
OBSERVATION:

LAB-6 20/11/24

8] Configure Routing Information Protocol

Topology :



Procedure :

* Select 8 no end devices (Generic PC). Give ip address configuration as given in the diagram.
* Select three generic PT switch. Connect two end devices to each switch that as same default gateway.
* Select 3 routers and connect each switch to one of the routers.
* Configure IP address to end devices and add default gateway also.

Router Configuration :-

Router 0 :

Router > enable
Router # config terminal
Router (config) # interface fastethernet 0/0
Router (config-if) # ip address 10.0.0.1 255.0.0.0

Router 1 :-

Interface fast ethernet 0/0
ip address 20.0.0.1 255.0.0.0

Router 2 :-

Interface fast ethernet 0/0
ip address 30.0.0.1 255.0.0.0

Router 0 :-

Interface serial 2/0
ip address 40.0.0.1 255.0.0.0

Router 1 :

Interface serial 2/0
ip address 40.0.0.2 255.0.0.0

Router 2 :

Interface serial 3/0
ip address 50.0.0.1 255.0.0.0

Router 2 :

Interface serial 2/0
ip address 50.0.0.2 255.0.0.0

Router 0 :-

Router (config) # router rip
Router (config-router) # network 10.0.0.0
network 40.0.0.0

Router 1 :

Router (config) # router rip
Router (config-router) # network 40.0.0.0
network 50.0.0.0
network 20.0.0.0

Router 2 :

Router (config) # router rip

Router (config-router) # network 50.0.0.0

network 30.0.0.0

Observation :

* packets were succesfully transfered among end devices connected to d/f routers.

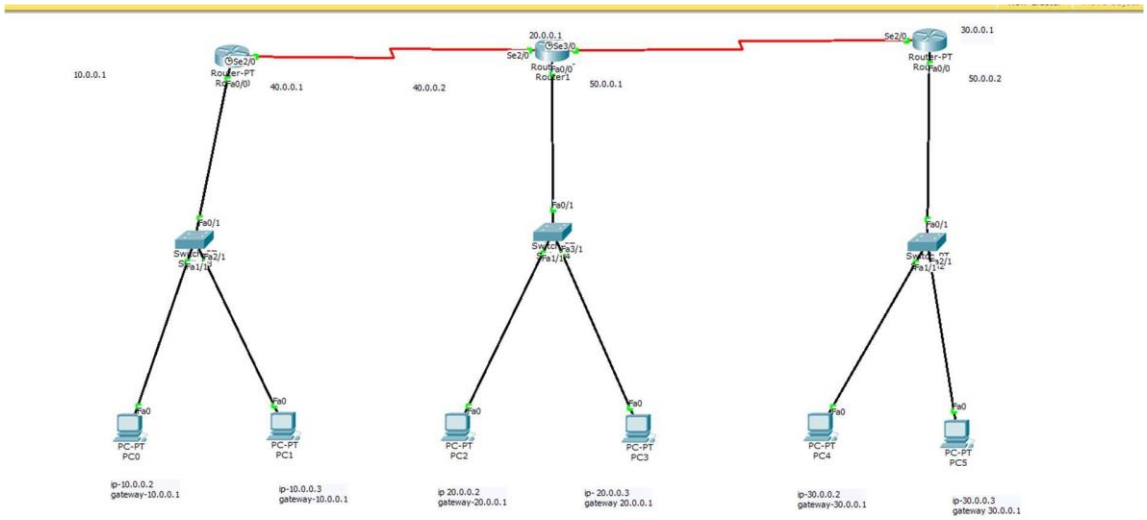
* Ping results :-

from PC1

ping 30.0.0.2

Sent = 4, Received = 4, lost = 0

TOPOLOGY:



OUTPUT:

```
Request timed out.
Reply from 20.0.0.2: bytes=32 time=2ms TTL=126
Reply from 20.0.0.2: bytes=32 time=2ms TTL=126
Reply from 20.0.0.2: bytes=32 time=4ms TTL=126

Ping statistics for 20.0.0.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 4ms, Average = 2ms

PC>ping 20.0.0.2

Pinging 20.0.0.2 with 32 bytes of data:

Reply from 20.0.0.2: bytes=32 time=5ms TTL=126
Reply from 20.0.0.2: bytes=32 time=2ms TTL=126
Reply from 20.0.0.2: bytes=32 time=1ms TTL=126
Reply from 20.0.0.2: bytes=32 time=1ms TTL=126

Ping statistics for 20.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 5ms, Average = 2ms

PC>
```

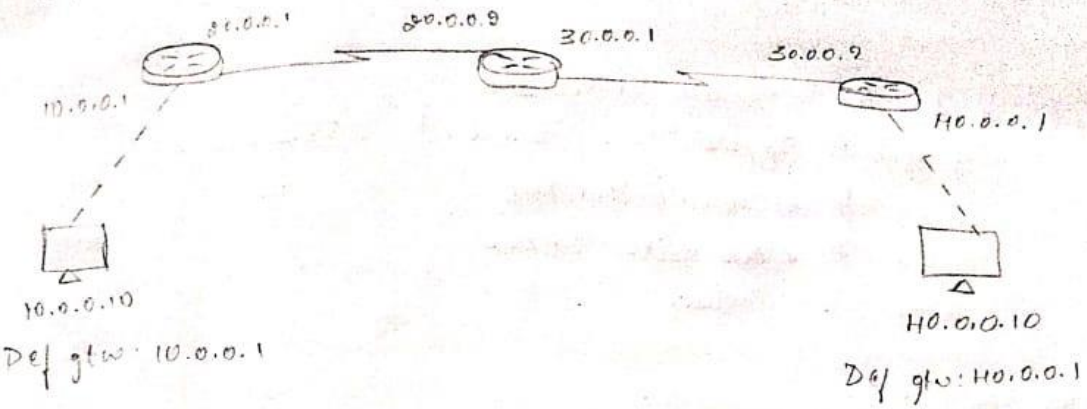
PROGRAM6: Configure OSPF routing protocol
OBSERVATION:

LAB-7

27/11/24

Aim :- Configure OSPF routing protocol.

Topology :



Procedure :-

1. Place 3 routers on the workbench.
2. Connect two general PC to router0 & 2 respectively using copper cross over.
3. Configure IP address and default gateway for both the end devices.
4. Configure router connections.

CLI :-

Router0 :

(Config) # interface fastethernet 0/0.

(config-if) # ip address 10.0.0.0 255.0.0.0

no shut.

interface serial 2/0

ip address 20.0.0.1 255.0.0.0

encapsulation ppp

clock rate 64000

no shut

Router1:

(config)# router ospf 1

(config-router)# network 20.0.0.0 0.255.255.255 area 1

network 30.0.0.0 0.255.255.255 area 0

exit

Router2:

(config)# router ospf 1

(config-router)# network 30.0.0.0 0.255.255.255 area 0

network 40.0.0.0 0.255.255.255 area 2

6. Loopback:

Router 0:

(config)# interface loopback 0

(config-if)# ip add 172.16.1.252 255.255.0.0

no shut

exit

Router1:

(config)# interface loopback 0

(config-if)# ip add 172.16.1.253 255.255.0.0

no shut

exit

Router2:

(config)# interface loopback 0

(config-if)# ip add 172.16.1.254 255.255.0.0

no shut

exit.

7. Virtual link b/w router establishment.

Router 0:

(config)# router ospf 1

(config-router) # area 1 virtual-link 2.2.2.2.

Router1 :

(config) # router ospf 1

(config-router) # area 1 virtual-link 1.1.1.1

exit.

Router2 :

show ip route.

~~IR 0.0.0~~

10.0.0.0/8 via 30.0.0.1 Serial 2/0

20.0.0.0/8 via 30.0.0.1 Serial 2/0

30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks

30.0.0.0/8 is directly connected, Serial 2/0

30.0.0.1/32 is directly connected, Serial 2/0

40.0.0.0/8 is directly connected, FastEthernet 0/0

172.16.0.0/16 is directly connected, Loopback 0

Observation :-

Successfully configured OSPF

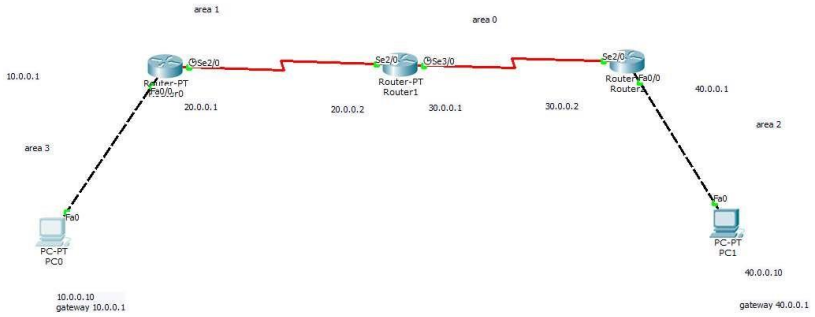
Ping results :

Ping 40.0.0.10

Ping statistics for 40.0.0.10 :

Packets : Sent = 4, Received = 4, Lost = 0 (0% loss),

TOPOLOGY:



OUTPUT:

```
Packet Tracer PC Command Line 1.0
PC>ping 40.0.0.10

Pinging 40.0.0.10 with 32 bytes of data:

Request timed out.
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125
Reply from 40.0.0.10: bytes=32 time=4ms TTL=125

Ping statistics for 40.0.0.10:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 6ms, Average = 5ms

PC>ping 40.0.0.10

Pinging 40.0.0.10 with 32 bytes of data:

Reply from 40.0.0.10: bytes=32 time=8ms TTL=125
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125

Ping statistics for 40.0.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 6ms, Maximum = 8ms, Average = 6ms
```

PROGRAM7: Demonstrate the TTL/ Life of a Packet

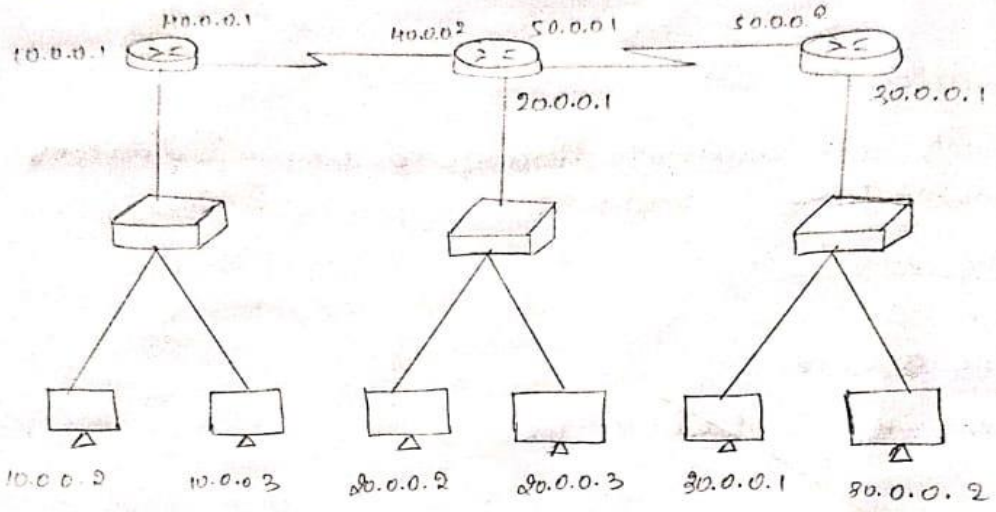
OBSERVATION:

LAB-6

20/11/24

8] Demonstrate TTL or life of a packet.

Topology:-



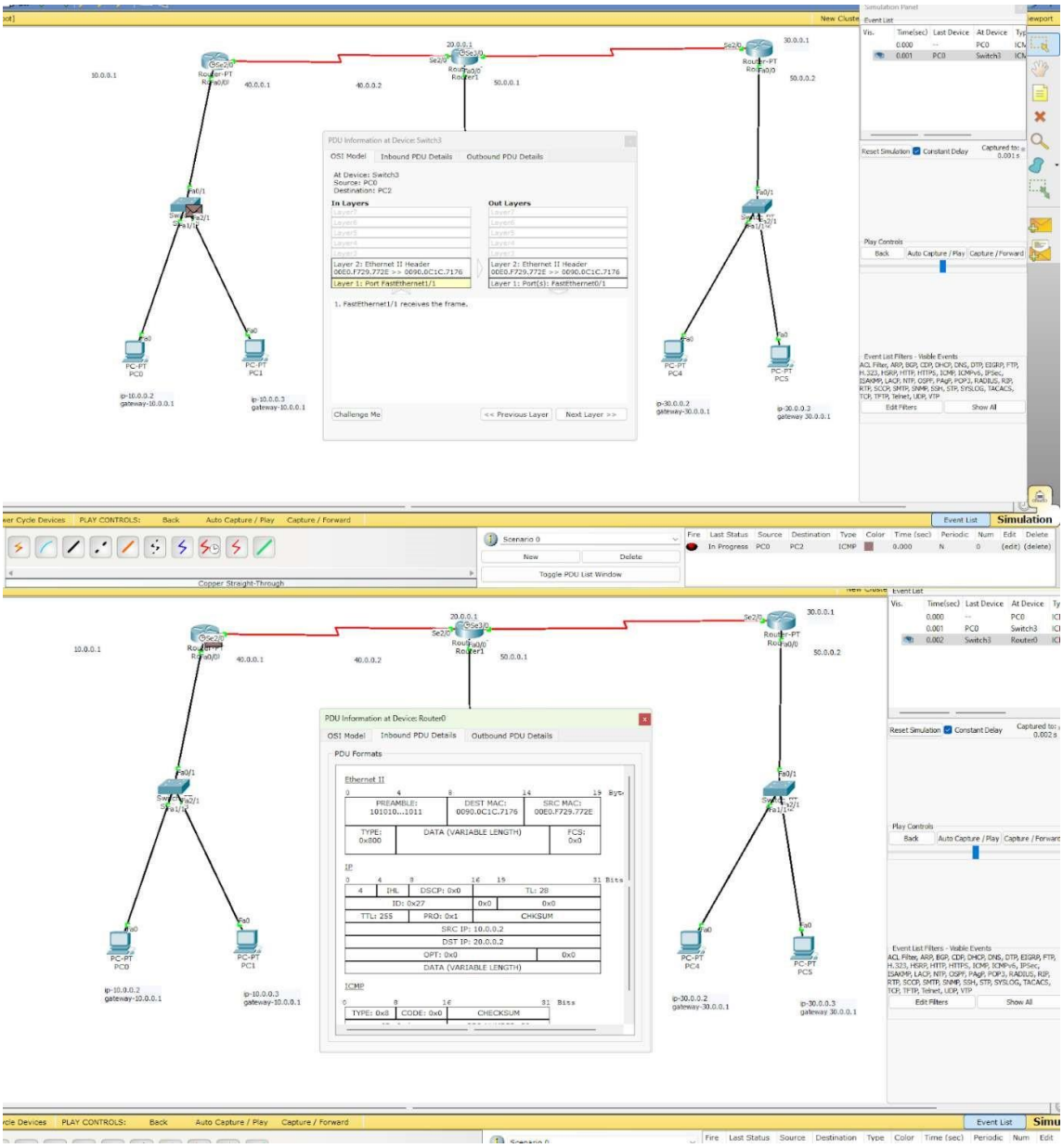
Procedure:-

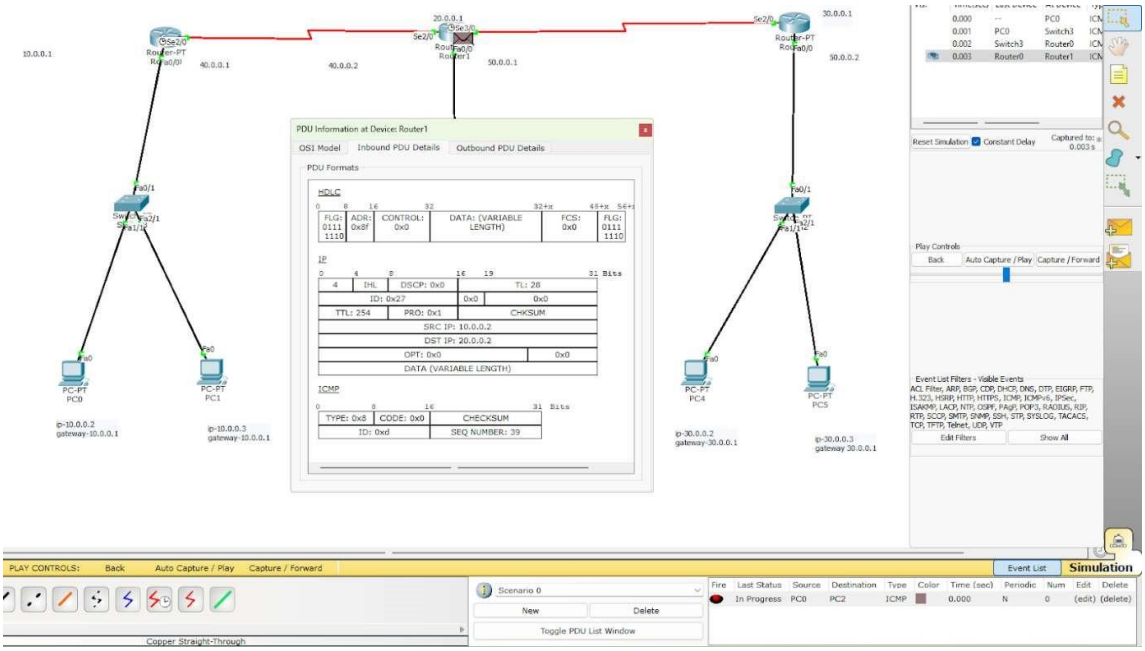
- Do configuration same as done in Routing Information Protocol experiment.
- In simulation mode, transfer a simple PDU from PC2 to PC4. Click on auto capture / play.
- This displays the PDU / packet being transferred from source to destination.

Observation:-

- The packets successfully sent from source to destination.
- The TTL starts at 255 and is decrement by 1 at each router and packets are succesfully transferred.

TOPOLOGY:




PDU Information at Device: Router1
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
HDLC
IP
ICMP
PC-PT PC0
PC-PT PC1
PC-PT PC4
PC-PT PC5
PLAY CONTROLS:
Back
Auto Capture / Play
Capture / Forward
Event List
Simulation
Scenario 0
New
Delete
Toggle PDU List Window
Copper Straight-Through

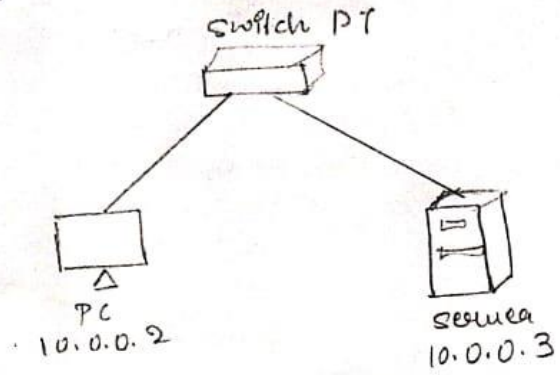
PROGRAM8: Configure Web Server, DNS within a LAN.
OBSERVATION:

LAB-8 18/12/24

8] To construct configure simple web server, DNS within LAN.

Aim :- To configure DNS server

Topology:



Procedure :

*. Select the PC, switch and server. and connect them as shown in the topology.

* Assign IP address to PC and server.

Note :- No IP address is assigned to switch.

* Go to DNS services in the server and turn it on write name and address and press add.

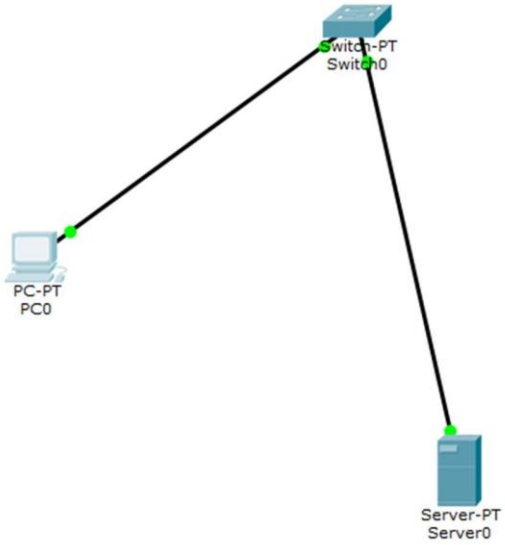
* In services select HTTP change or edit the file.

* After editing the file. Select the PC and go to desk top in the PC. In the desktop go to web browser. and there provide the domain name to get results.

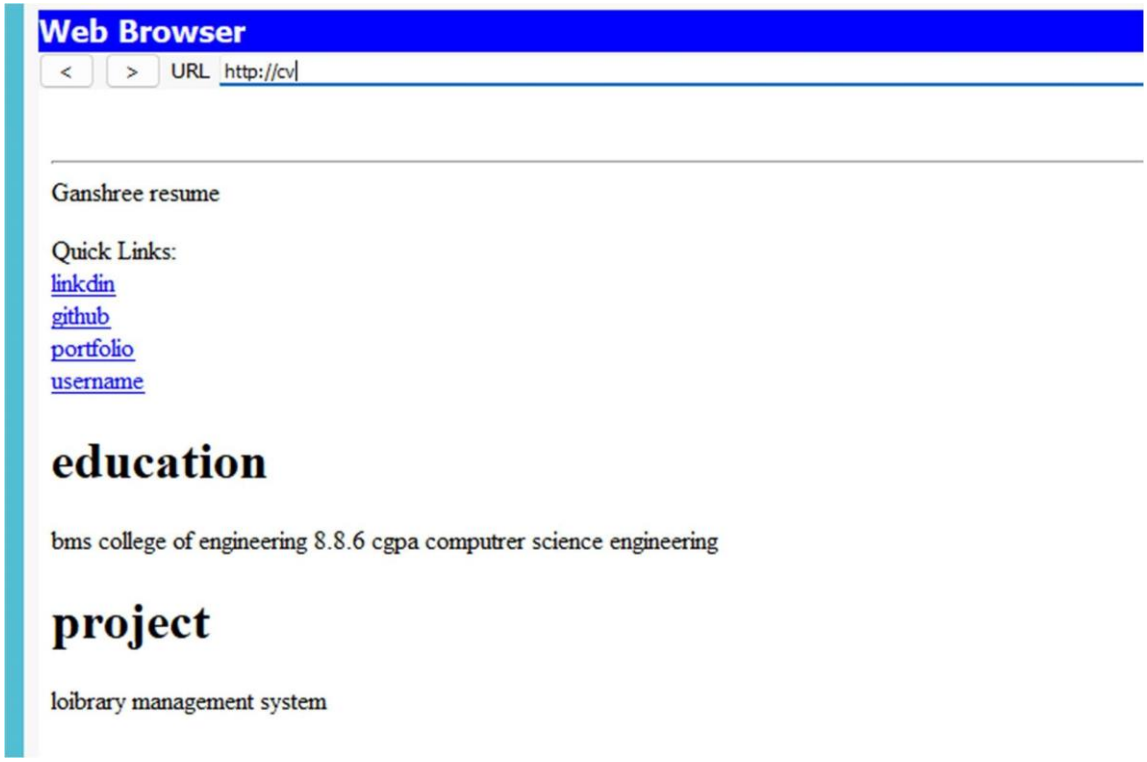
Observation :

* Connection is successfully established.
* The domain name system maps each IP address with domain name.
* When entered domain name the contents of the specific IP address comes from server.

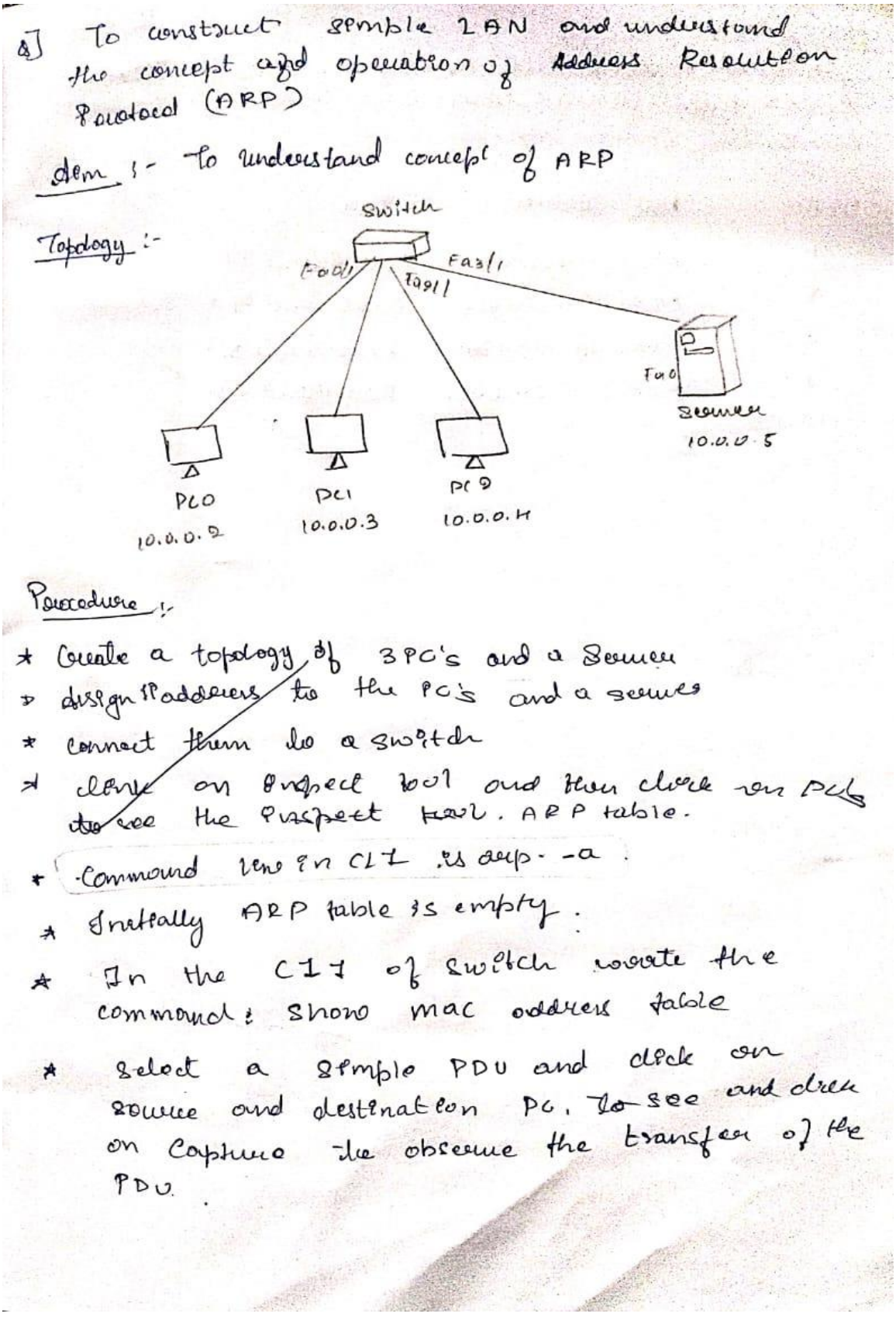
TOPOLOGY:



OUTPUT:



PROGRAM9: To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)
OBSERVATION:

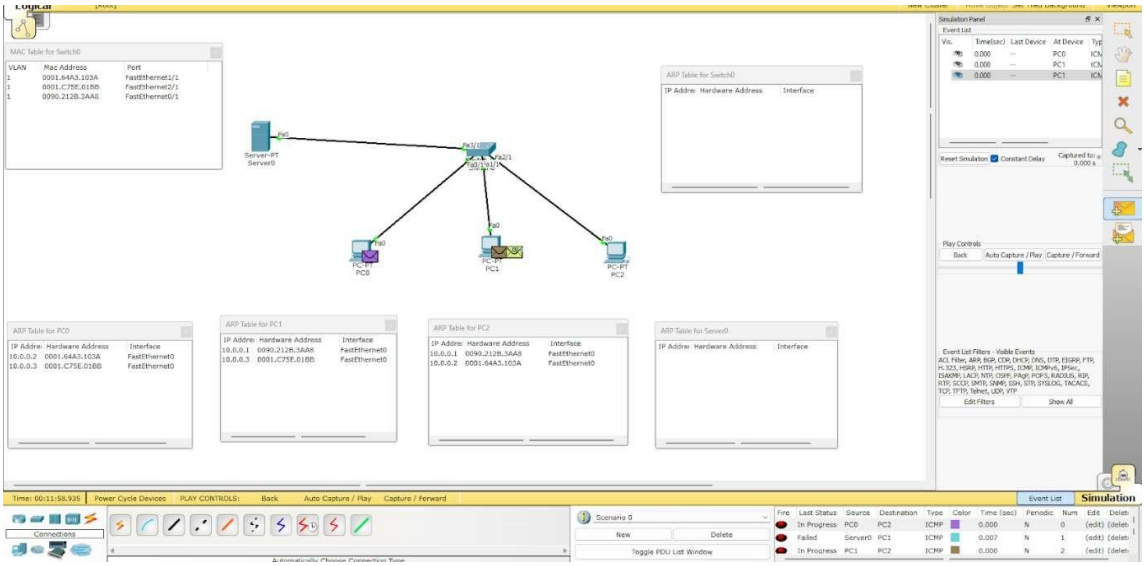


Observation:

Observed that switch as well as nodes update the ARP table as and when new communication starts.

VLAN	MAC Address	Port
1	0001.6420.CCM	FastEthernet 1/1
1	0021.9938.C650	FastEthernet 2/1
1	0002.4A1639[illegible]	FastEthernet 3/1
1	000C.CE0H.C05C	FastEthernet 0/1

TOPOLOGY:



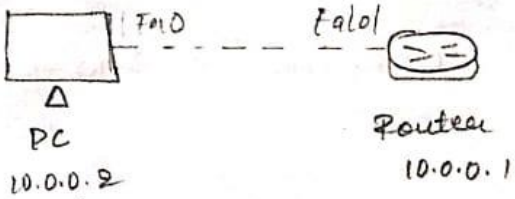
PROGRAM10: To understand the operation of TELNET by accessing the router in server room from a PC in IT office.

OBSERVATION:

Q] To understand the operation of TELNET by accessing router in server room from PC in IT office.

Aim: To understand and design TELNET to connect PC to router

Topology :-



Procedure :-

* Connect PC to the router as shown in topology.
* Assign IP address to the PC and router.
* configure Router :

Router CLI :

interface fast ethernet 0/0
ip address 10.0.0.1 255.0.0.0
no shut

Then commands : in Router:

enable
config t
hostname R1
enable secret P1
interface fastethernet 0/0
ip address 10.0.0.1 255.0.0.0
no shut
line vty 05
login

password p0
enet
enet
ht

* Command in PC:

- ping 10.0.0.1

  ping results shown:

  Packet: Sent = 4, Received = 4, Lost = 0

- telnet 10.0.0.1

  Trying 10.0.0.1 .. open

  User access verification

  Password:

  R1> enable

  Password:

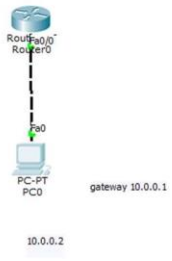
  R1# show ip route

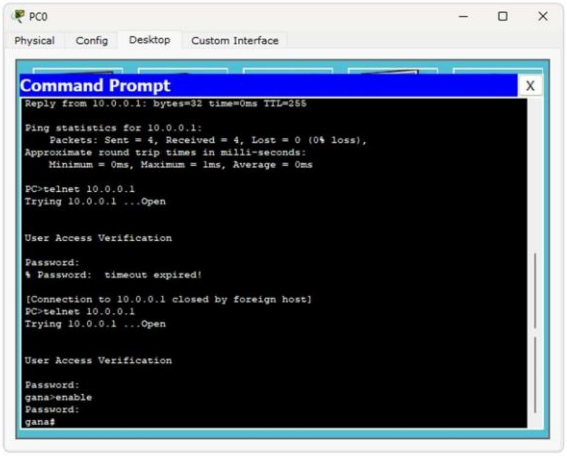
  output: 10.0.0.0/8 is directly connected.

Observation:

* Design of TELNET was successful.
* It is observed that through TELNET the PC, the hostname and password is given to access CLI of Router by any other devices.
* could successfully ping.

TOPOLOGY and OUPUT:





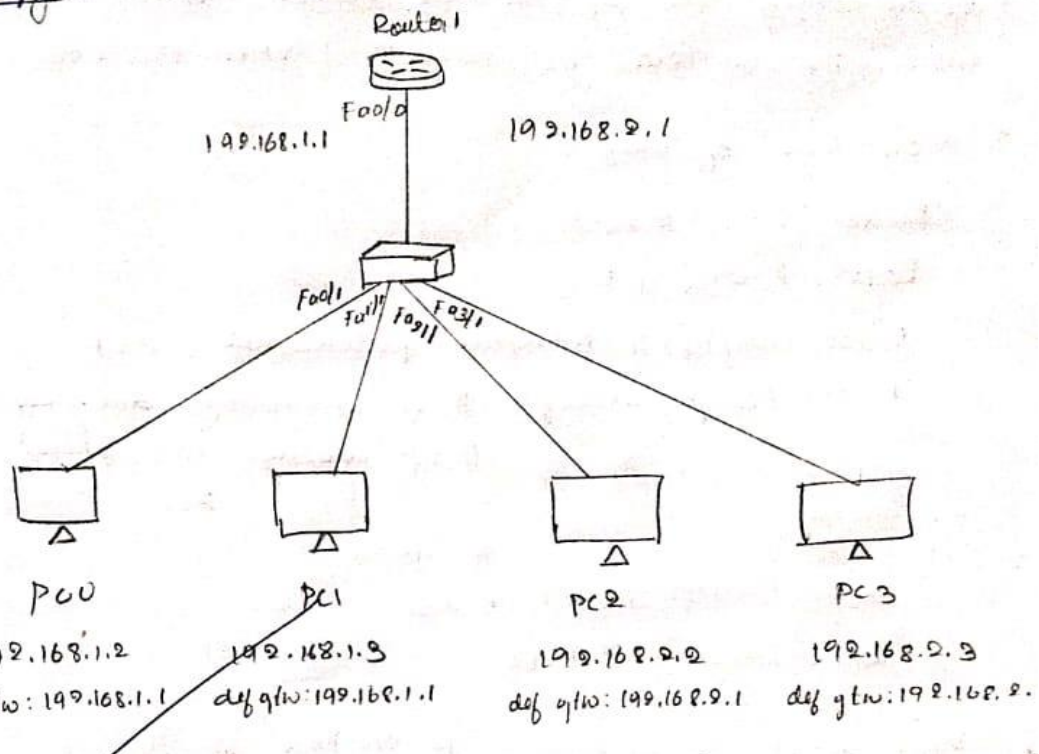
PROGRAM11: To construct a VLAN and make the PC's communicate among a VLAN

OBSERVATION:

Q] creating a VLAN and make PC's communicate among the VLAN

Aim:- To configure VLAN and make PC's communicate each other.

Topology:



Procedure:-

* Select 1 8HI routers and connect to switch.
* connect HPC's to switch.
* design IP addresses and gtw to the PC's as shown in the topology.
* configure Router for 1st 2 Pc's for def gtw: 192.168.1.1

```
# enable
  config t
  interface fastethernet 0/0
  ip address 192.168.1.1  255.255.255.0
  no shut
  exit
```

* Go to switch and then to config > select VLAN Database
* Set the VLAN number to 2
* set VLAN name bmsce and click on ADD
* Go to fastethernet 1/1 and set access as trunk
* Go to fastethernet 0/1 and set access as trunk
* Go to config tab of router select VLAN Database enter the number and name of vlan created
* Go to CLI of router:

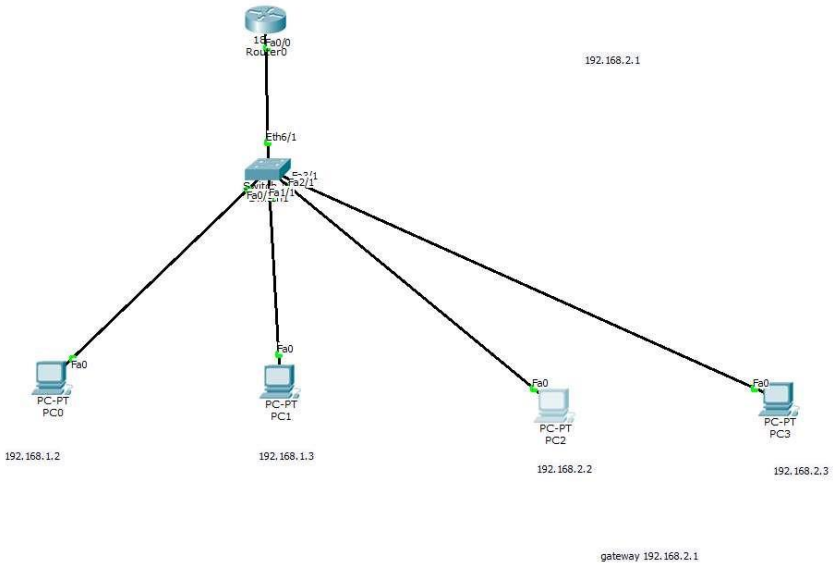
```
Router (vlan) #exit
Router # config t
Router (config) # interface fastethernet 0/0.1
Router (config-subif) # encapsulation dot1q 2
                      # ip address 192.168.2.1
                                    255.255.255.0
                      # no shut
Router (config-subif) # exit
Router (config) # exit
```

* Ping PC of one of gateway to the PCs of other gtw.

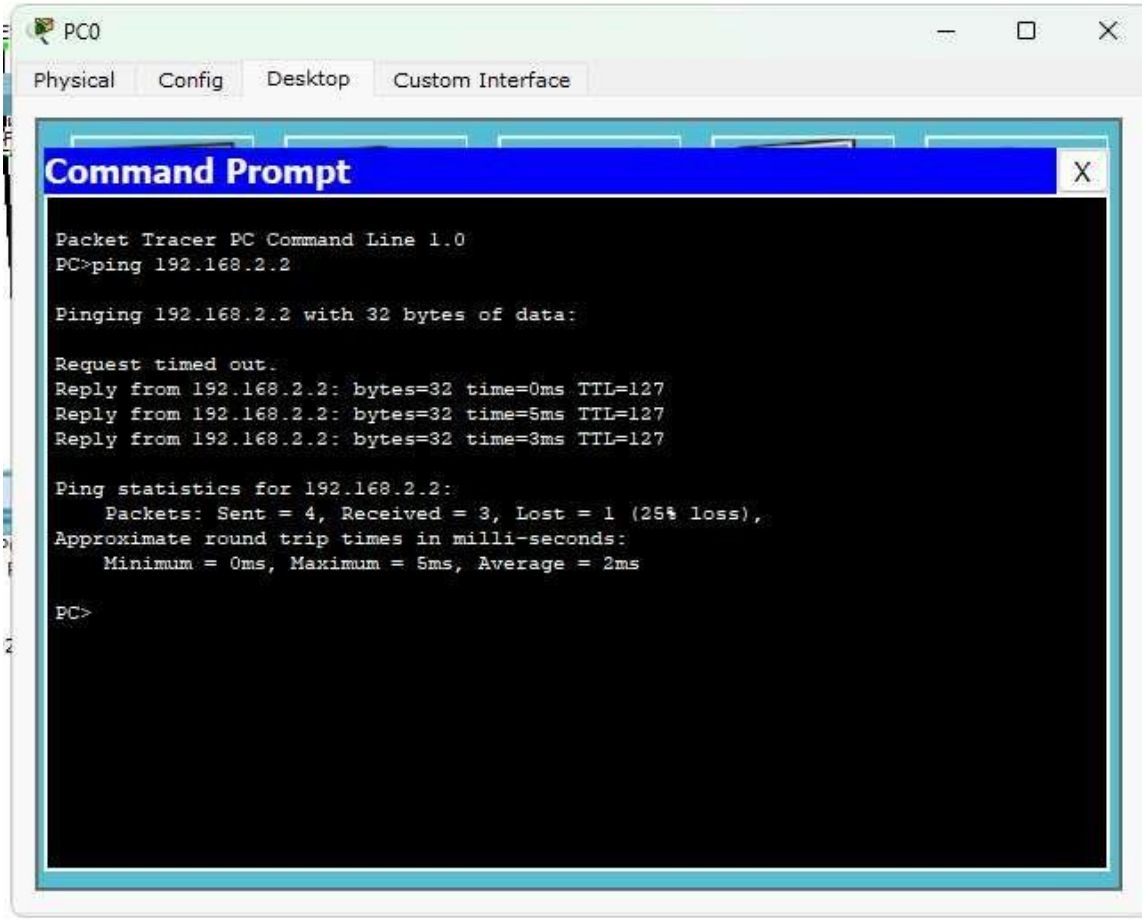
Observation :-

* VLAN tunneling allows switches to forward from d/f VLAN, over single link called trunk.
* This is done by adding adding header ftn called tag to ethernet frame. This process is called tagging
* The switch understands VLAN
* The router understands VLAN

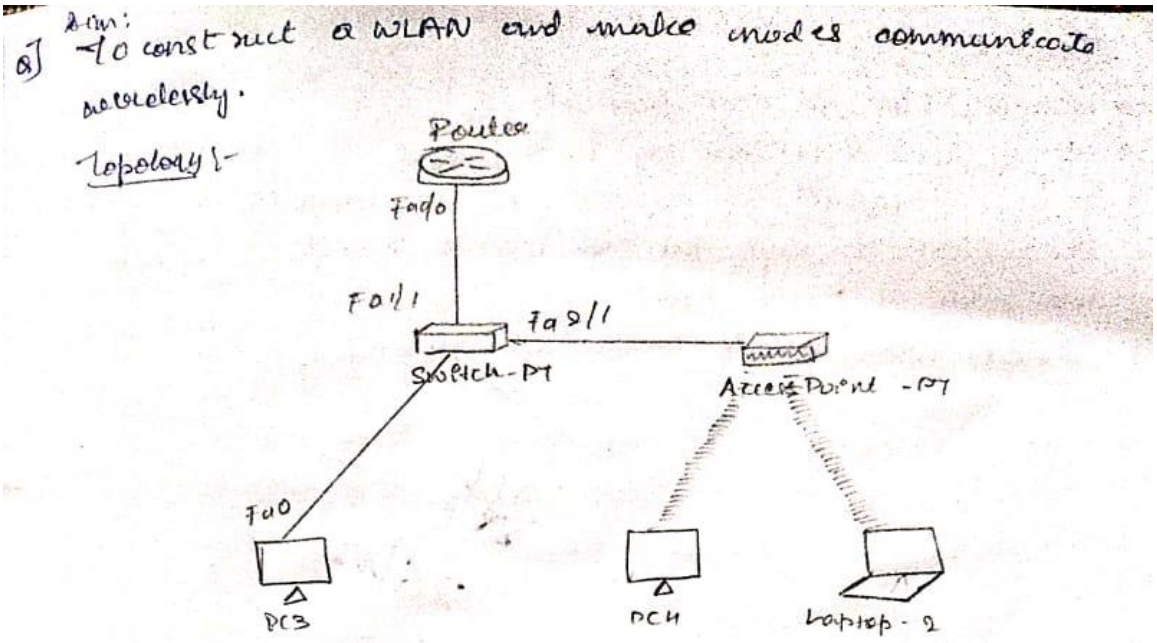
TOPOLOGY:



OUTPUT:



PROGRAM12: To construct a WLAN and make the nodes communicate wirelessly
To construct a WLAN and make the nodes communicate wirelessly
OBSERVATION:



Procedure :-

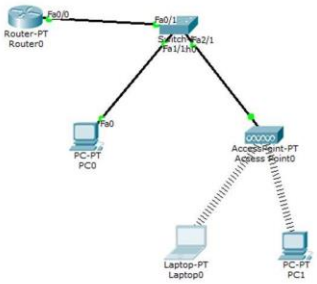
* Select the router and connect it to a switch.
* Connect PC to the switch.
* Select Access Point-PT from the wireless devices and connect it to switch.
* Configure Access Point (- Port 1 - SSID Name as name any name.
* Select WEP and give any 10 digit hexa key
* Configure PC4 and laptop with wireless standards.
* Go to PC4 switch off the device. Drag existing PT-HOST-NM-1AM to the component listed in the LHS.
* Drag WMP300N wireless interface to empty port.
* Switch on the device.
* Repeat for the same for Laptop also.
* Ping from every wireless device to every other device and observe the results.

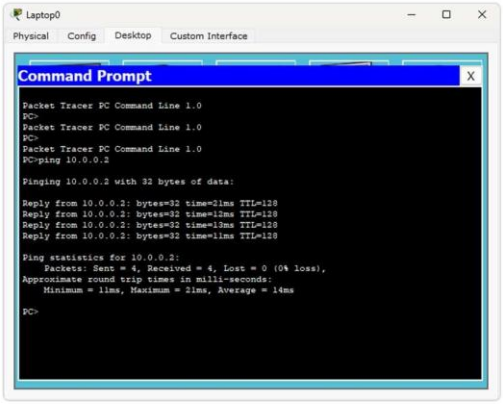
Observation :-

* Wireless LAN uses WEP protocol.
* It requires SSID and key to be present
* It uses AccessPoint to establish wireless connection.
* The wireless in config tab new wireless interface have been added to PC and laptop
* Devices were able to communicate successfully.

26/2

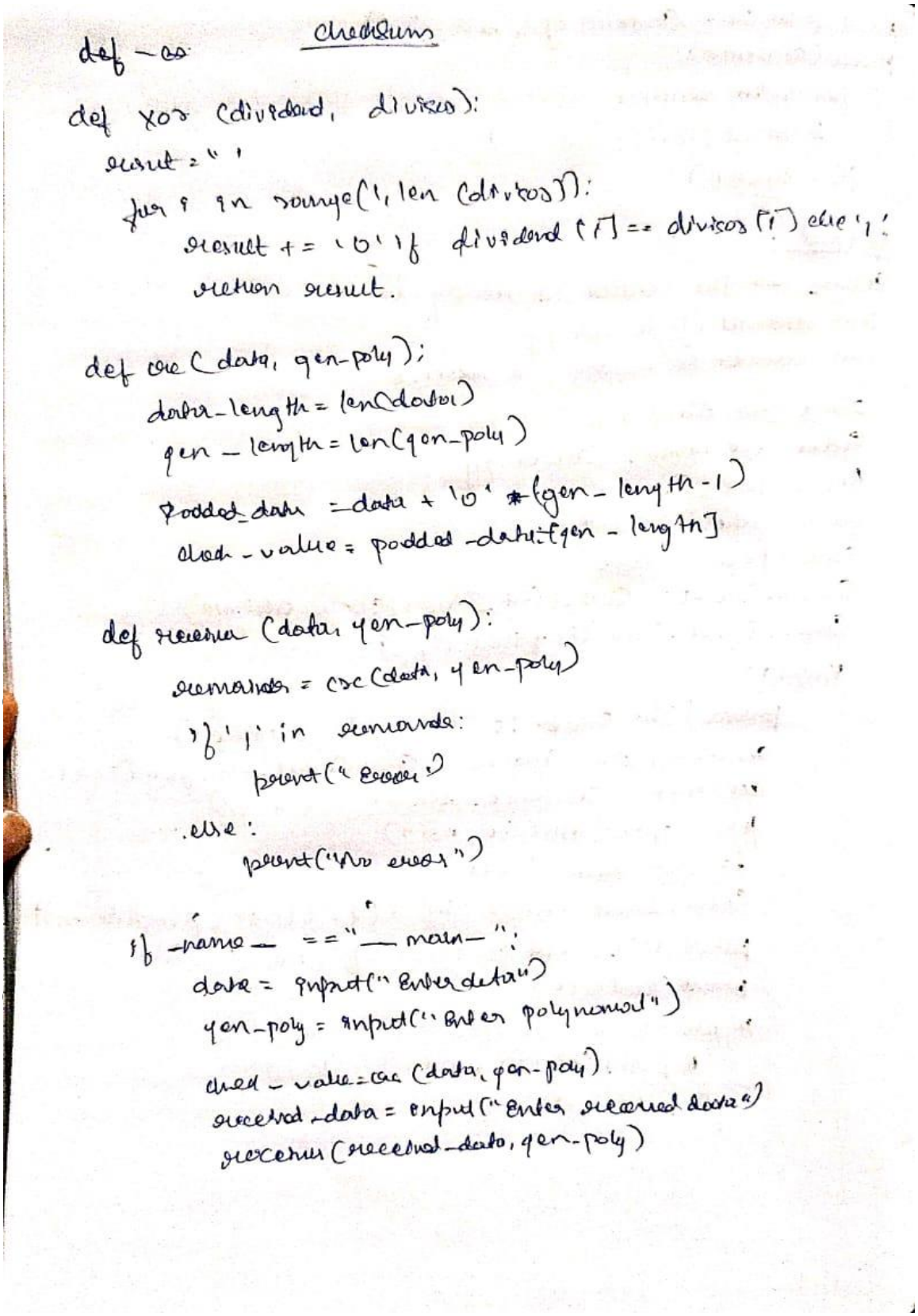
TOPOLOGY AND OUPUT:





CYCLE-2

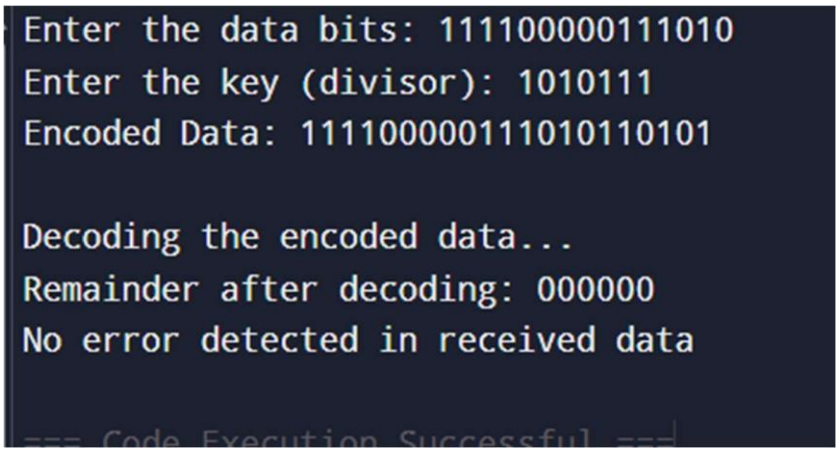
PROGRAM1: Write a program for error detecting code using CRCCCITT (16-bits)
OBSERVATION:



```
def-xo                    checksum

def xor(dividend, divisor):
    result = ""
    for i in range(1, len(divisor)):
        result += '0' if dividend[i] == divisor[i] else '1'
    return result

def crc(data, gen_poly):
    data_length = len(data)
    gen_length = len(gen_poly)
    padded_data = data + '0' * (gen_length - 1)
    check_value = padded_data[:gen_length]

def receiver(data, gen_poly):
    remainder = crc(data, gen_poly)
    if '1' in remainder:
        print("Error")
    else:
        print("No error")

if __name__ == "__main__":
    data = input("Enter data")
    gen_poly = input("Enter polynomial")
    check_value = crc(data, gen_poly)
    received_data = input("Enter received data")
    receiver(received_data, gen_poly)
```

```
        pick += 1

    if tmp[0] == '1':
        tmp = xor(divisor, tmp)
else:
        tmp = xor('0' * pick, tmp)

    checkword = tmp
return checkword


def encode(data, key):     key_len = len(key)
appended_data = data + '0' * (key_len - 1)
remainder = mod2div(appended_data, key)
codeword = data + remainder
print(f"Encoded Data: {codeword}")
return codeword


def decode(data, key):
    remainder = mod2div(data, key)
print(f"Remainder after decoding: {remainder}")
if '1' not in remainder:
        print("No error detected in received data")
else:
        print("Error detected in received data")


# Main function
if __name__ == "__main__":
    data = input("Enter the data bits: ")
    key = input("Enter the key (divisor): ")

    # Encoding
    encoded_data = encode(data, key)

    # Decoding
    print("\nDecoding the encoded data...")
decode(encoded_data, key)
```

OUTPUT:


Enter the data bits: 111100000111010
Enter the key (divisor): 1010111
Encoded Data: 11110000111010110101
Decoding the encoded data...
Remainder after decoding: 000000
No error detected in received data
=== Code Execution Successful ===

PROGRAM2: Write a program for congestion control using Leaky bucket algorithm.
OBSERVATION:

Leaky Bucket

1] Write a program for congestion control using leaky bucket algm.

Program: lbucket.cc

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
# define NOF_PACKETS
/* int rand (int a) {
    int rn= (random () % 10) % a;
    return rn == 0 ? 1 : rn;
  }
*/
/*
# include <stdlib.h>
    long int random (void);
*/
int main()
{ int packet_sz [NOF_PACKETS] , i, clk, b_size, o_rate,
  p_sz_rm=0, p_sz, p_temp_op;
  for (i=0 ; i<NOF_PACKETS; ++i)
     packet_sz[i]= random () %100;
  for (i=0 ; i < NOF_PACKETS ; ++i)
     printf("\n packet[%d] : %d bytes \t", i, packet_sz[i]);
  printf("\n Enter Output rate :");
  scanf("%d", &o_rate);
  printf("\n Enter bucket size:");
  scanf("%d", & b_size);
  for (i=0 ; i<NOF_PACKETS ; ++i)
  {
     if (packet_sz[i]+ p_sz_rm) > b_size)
        if ( packet_sz[i] > b_size)
           /* compare the packet size & bucket size */
```

```
        printf("\n\n Incoming packetsize (%d bytes) is
        greater than bucket capacity (%d bytes) - PACKET
        REJECTED", packet_sz[i], b_size);

    else:
        print("\n\n Bucket capacity exceeded - PACKETS
        REJECTED!!");

    else {
     p_sz_rem += packet_sz[i];
     printf("\n\n Incoming packet size: %d", packet_sz[i]);
     printf("\n Bytes remaining to transmit: %d", p_sz_rem);
     // p_time = random() * 10;
     // printf("\n Time left for transmission: %d units",
      p_time);
     // for (clk = 10; clk <= p_time; clk += 10)
     while (p_sz_rem > 0) {
      sleep(1);
      if (p_sz_rem) {
          if (p_sz_rem <= o_rate) /* packet size remaining
       comparing with output rate */
              op = p_sz_rem, p_sz_rem = 0;
       else
          op = o_rate, p_sz_rem = o_rate;
          printf("\n Packet of size %d transmitted", op);
          printf("\n -- Bytes remaining to transmit: %d",
          p_sz_rem);

     }
     else {
     printf("\n No of packets to transmit!!"); } } }
```

Output:

packet[0]: 83 bytes

packet[1]: 86 bytes

packet [2] : 77 bytes
packet [3] : 15 bytes
packet [4] : 93 bytes
Enter the output rate : 30
Enter the bucket size : 85

Incoming packet size : 83
Bytes remaining to transmit : 83
packet of size 30 transmitted -- Bytes Remaing to transmit : 53
packet of size 30 transmitted -- Bytes Remaing to transmit : 23
packet of size 23 transmitted -- Bytes Remaining to transmit : 0

Incoming packet size : 77
Bytes remaining to transmit : 77
packet of size 30 transmitted -- Bytes Remaining to transmit : 47
packet of size 30 transmitted -- Bytes Remaing to transmit : 17

CODE:

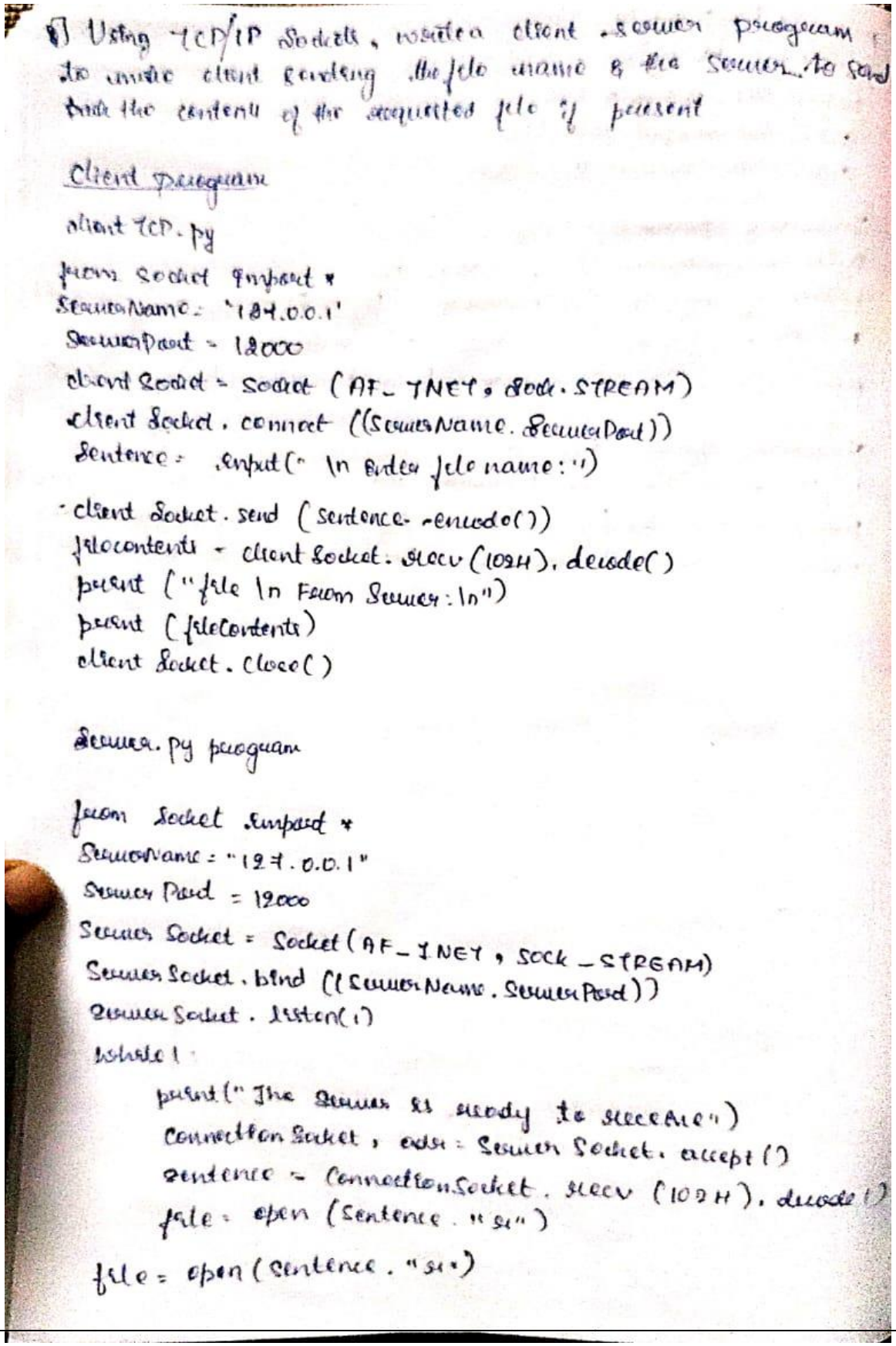
```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h> // for sleep function
#define NOF_PACKETS 5
// Function to simulate sending packets void send_packet(int
packet_size, int output_rate) {     while (packet_size > 0) {         int
sent = (packet_size < output_rate) ? packet_size : output_rate;
printf("Packet of size %d Transmitted---", sent);         packet_size -=
sent;         printf("Bytes Remaining to Transmit: %d\n",
packet_size);         sleep(1);  // Simulate time delay between packets
   }
}

int main() {
   int output_rate, bucket_size, incoming_packet_size;
   int i, packet_size[NOF_PACKETS];

   // Input number of packets and their sizes
for(i = 0; i < NOF_PACKETS; i++) {
      packet_size[i] = rand() % 100;  // Random packet size between 0 and 99
printf("packet[%d]:%d bytes\n", i, packet_size[i]);
   }

   printf("Enter the Output rate:");
   scanf("%d", &output_rate);

   printf("Enter the Bucket Size:");
   scanf("%d", &bucket_size);

   for(i = 0; i < NOF_PACKETS; i++) {         printf("\nIncoming Packet size: %d\n",
packet_size[i]);         if(packet_size[i] > bucket_size) {             printf("Incoming packet
size (%dbytes) is Greater than bucket capacity (%dbytes)-
PACKET REJECTED\n", packet_size[i], bucket_size);
continue;
      }

      printf("Bytes remaining to Transmit: %d\n", packet_size[i]);
send_packet(packet_size[i], output_rate);
   }
return 0;
}
```

OUTPUT:

```
packet[0]:83 bytes
packet[1]:86 bytes
packet[2]:77 bytes
packet[3]:15 bytes
packet[4]:93 bytes
Enter the Output rate:50
Enter the Bucket Size:300

Incoming Packet size: 83
Bytes remaining to Transmit: 83
Packet of size 50 Transmitted---Bytes Remaining to Transmit: 33
Packet of size 33 Transmitted---Bytes Remaining to Transmit: 0

Incoming Packet size: 86
Bytes remaining to Transmit: -86
Packet of size 50 Transmitted---Bytes Remaining to Transmit: 36
Packet of size 36 Transmitted---Bytes Remaining to Transmit: 0

Incoming Packet size: 77
Bytes remaining to Transmit: 77
Packet of size 50 Transmitted---Bytes Remaining to Transmit: 27
Packet of size 27 Transmitted---Bytes Remaining to Transmit: 0

Incoming Packet size: 15
Bytes remaining to Transmit: 15
Packet of size 15 Transmitted---Bytes Remaining to Transmit: 0

Incoming Packet size: 93
Bytes remaining to Transmit: 93
Packet of size 50 Transmitted---Bytes Remaining to Transmit: 43
Packet of size 43 Transmitted---Bytes Remaining to Transmit: 0


=== Code Execution Successful ===
```

PROGRAM3: 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.

OBSERVATION:



3) Using TCP/IP Sockets, write a client-server program to make client sending the file name & the server to send back the contents of the requested file if present

Client program

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("\n Enter file name:")
clientSocket.send(sentence.encode())
fileContents = clientSocket.recv(1024).decode()
print("File \n From Server:\n")
print(fileContents)
clientSocket.close()
```

Server.py program

```
from socket import *
ServerName = "127.0.0.1"
ServerPort = 12000
ServerSocket = socket(AF_INET, SOCK_STREAM)
ServerSocket.bind((ServerName, ServerPort))
ServerSocket.listen(1)
while 1:
    print("The Server is ready to receive")
    connectionSocket, addr = ServerSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    file = open(sentence, "r")
file = open(sentence, "r")
```

```
l = file.read(1024)
connectionSocket.send(l.encode())
print("\n Sent contents of" + sentence)
file.close()
connectionSocket.close()
```

Output:

server -> The server is ready to receive
sent contents of server.py
client -> Enter file name: server.py

From Server:

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
while 1:
    print("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    file = open(sentence, "r")
    l = file.read(1024)
    connectionSocket.send(l.encode())
    print("\n Sent contents of" + sentence)
    file.close()
    connectionSocket.close()
```

CODE:

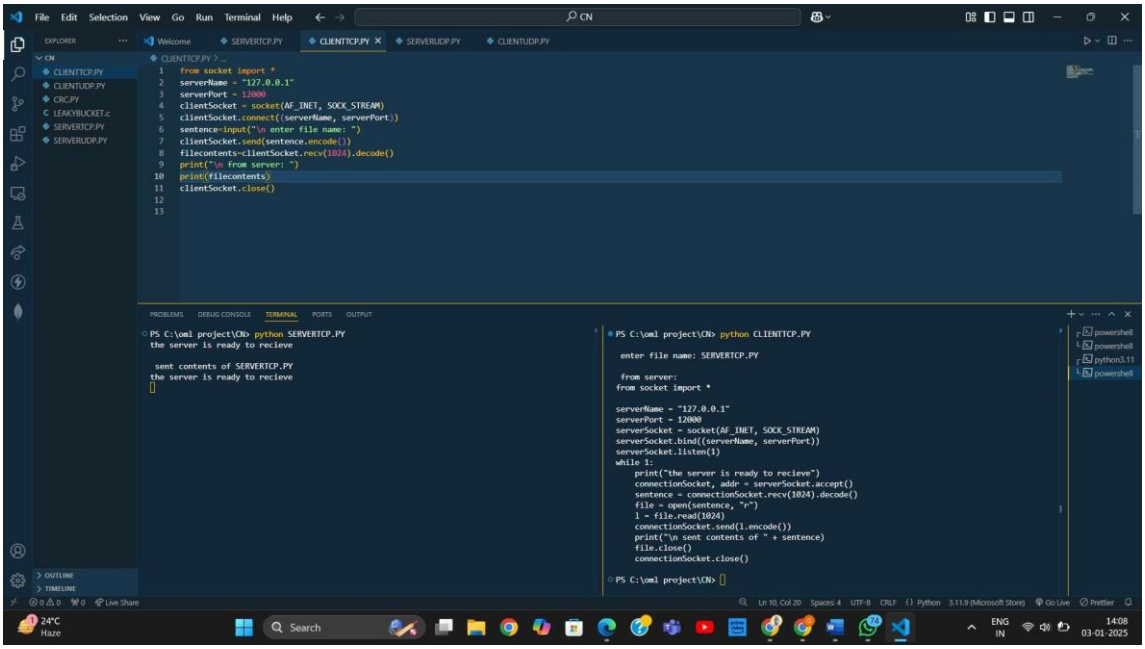
SERVERTCP.PY:

```
from socket import *

serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET,
SOCK_STREAM) serverSocket.bind((serverName,
serverPort)) serverSocket.listen(1) while 1:
   print("the server is ready to recieve")
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024).decode()
   file = open(sentence, "r")     l =
file.read(1024)
connectionSocket.send(l.encode())
print("\n sent contents of " + sentence)
   file.close()
   connectionSocket.close()
```

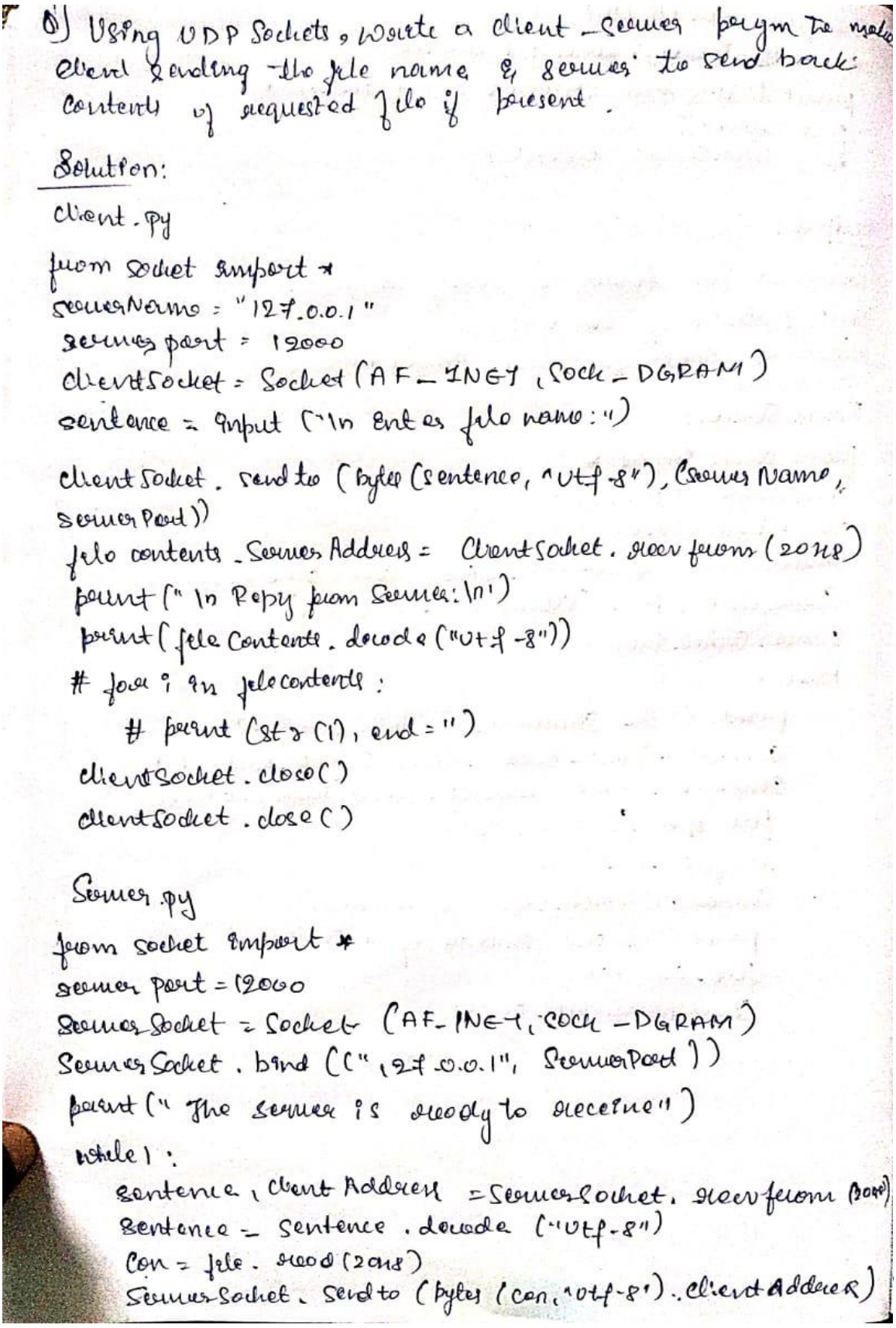
CLIENTTCP.PY:

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET,
SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence=input("\n enter file name: ")
clientSocket.send(sentence.encode())
filecontents=clientSocket.recv(1024).decode()
print("\n from server: ")
print(filecontents)
clientSocket.close()
```

OUTPUT:


CLIENTTCP.PY
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence=input("\n enter file name: ")
clientSocket.send(sentence.encode())
filecontents=clientSocket.recv(1024).decode()
print("\n from server: ")
print(filecontents)
clientSocket.close()
PS C:\oml project\CN> python SERVERTCP.PY
the server is ready to recieve
sent contents of SERVERTCP.PY
the server is ready to recieve
PS C:\oml project\CN> python CLIENTTCP.PY
enter file name: SERVERTCP.PY
from server:
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
while 1:
print("the server is ready to recieve")
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024).decode()
file = open(sentence, "r")
l = file.read(1024)
connectionSocket.send(l.encode())
print("\n sent contents of " + sentence)
file.close()
connectionSocket.close()
PS C:\oml project\CN>

PROGRAM4: 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.
OBSERVATION:



4) Using UDP Sockets, write a client-server pgm to make client sending the file name & server to send back contents of requested file if present.

Solution:

client.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(2048)
print("\n Reply from Server:\n")
print(filecontents.decode("utf-8"))
# for i in filecontents:
    # print(str(i), end="")
clientSocket.close()
clientSocket.close()
```

Server.py

```
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")
    con = file.read(2048)
    ServerSocket.sendto(bytes(con, "utf-8"), clientAddress)
```

```
print ('\n Sent Contents of ', end = ' ')
print (sentence)
# for i in sentence:
    # print (str(i), end = '')
file.close()
```

Output:

Server -> The server is ready to receive.

Sent contents of Server.py

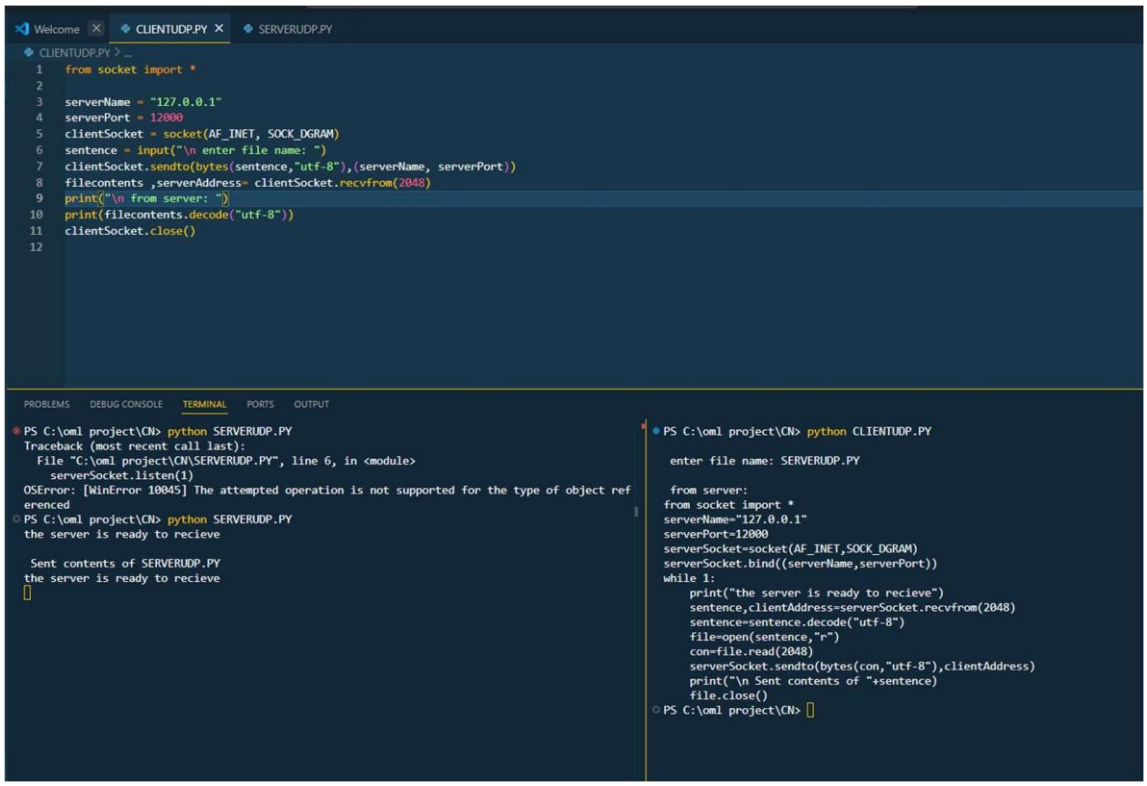
The server is ready to receive

client -> client.py

Enter file name: Server.py

Reply from Server:

```
from socket import *
serverport = 12000
ServerSocket = socket(AF_INET, SOCK_DGRAM)
ServerSocket.bind(("127.0.0.1", serverPort))
while 1:
    print ("The server is ready to receive")
    sentence, clientAddress = ServerSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file = open(sentence, "r")
    l = file.read(2048)
    ServerSocket.sendto (bytes(l, "utf-8"), clientAddress)
    print ('\n sent contents of ', end = ' ')
    print (sentence)
    # for i in sentence:
        # print (str(i), end = '')
    file.close()
```


Welcome
CLIENTUDP.PY
SERVERUDP.PY
CLIENTUDP.PY > ...
1 from socket import *
2
3 serverName = "127.0.0.1"
4 serverPort = 12000
5 clientSocket = socket(AF_INET, SOCK_DGRAM)
6 sentence = input("\n enter file name: ")
7 clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))
8 filecontents ,serverAddress= clientSocket.recvfrom(2048)
9 print("\n from server: ")
10 print(filecontents.decode("utf-8"))
11 clientSocket.close()
12
PROBLEMS
DEBUG CONSOLE
TERMINAL
PORTS
OUTPUT
PS C:\oml project\CN> python SERVERUDP.PY
Traceback (most recent call last):
File "C:\oml project\CN\SERVERUDP.PY", line 6, in <module>
serverSocket.listen(1)
OSError: [WinError 10045] The attempted operation is not supported for the type of object referenced
PS C:\oml project\CN> python SERVERUDP.PY
the server is ready to recieve
Sent contents of SERVERUDP.PY
the server is ready to recieve
PS C:\oml project\CN> python CLIENTUDP.PY
enter file name: SERVERUDP.PY
from server:
from socket import *
serverName="127.0.0.1"
serverPort=12000
serverSocket=socket(AF_INET,SOCK_DGRAM)
serverSocket.bind((serverName,serverPort))
while 1:
print("the server is ready to recieve")
sentence,clientAddress=serverSocket.recvfrom(2048)
sentence=sentence.decode("utf-8")
file=open(sentence,"r")
con=file.read(2048)
serverSocket.sendto(bytes(con,"utf-8"),clientAddress)
print("\n Sent contents of "+sentence)
file.close()
PS C:\oml project\CN>

WIRESHARK:

6) Using UDP Sockets, Write a client_server program to make client sending the file name & server to send back contents of requested file if present.

Solution:

Client.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))
file contents, ServerAddress = clientSocket.recvfrom(2048)
print("\n Reply from Server:\n")
print(file contents.decode("utf-8"))
# for i in filecontents:
    # print(str(i), end="")
clientSocket.close()
clientSocket.close()
```

Server.py

```
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")
    con = file.read(2048)
    ServerSocket.sendto(bytes(con, "utf-8"), clientAddress)
```