VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

COMPUTER NETWORKS

Submitted by

DIVAKAR BABU M P(1BM22CS093)

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 **DIVAKAR BABU M P(1BM22CS093),** 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.

Dr. Latha N.R.
Associate Professor,
Department of CSE,
BMSCE, Bengaluru

Dr. Kavitha Sooda
Professor and Head,
Department of CSE
BMSCE, Bengaluru

INDEX

CYCLE 1

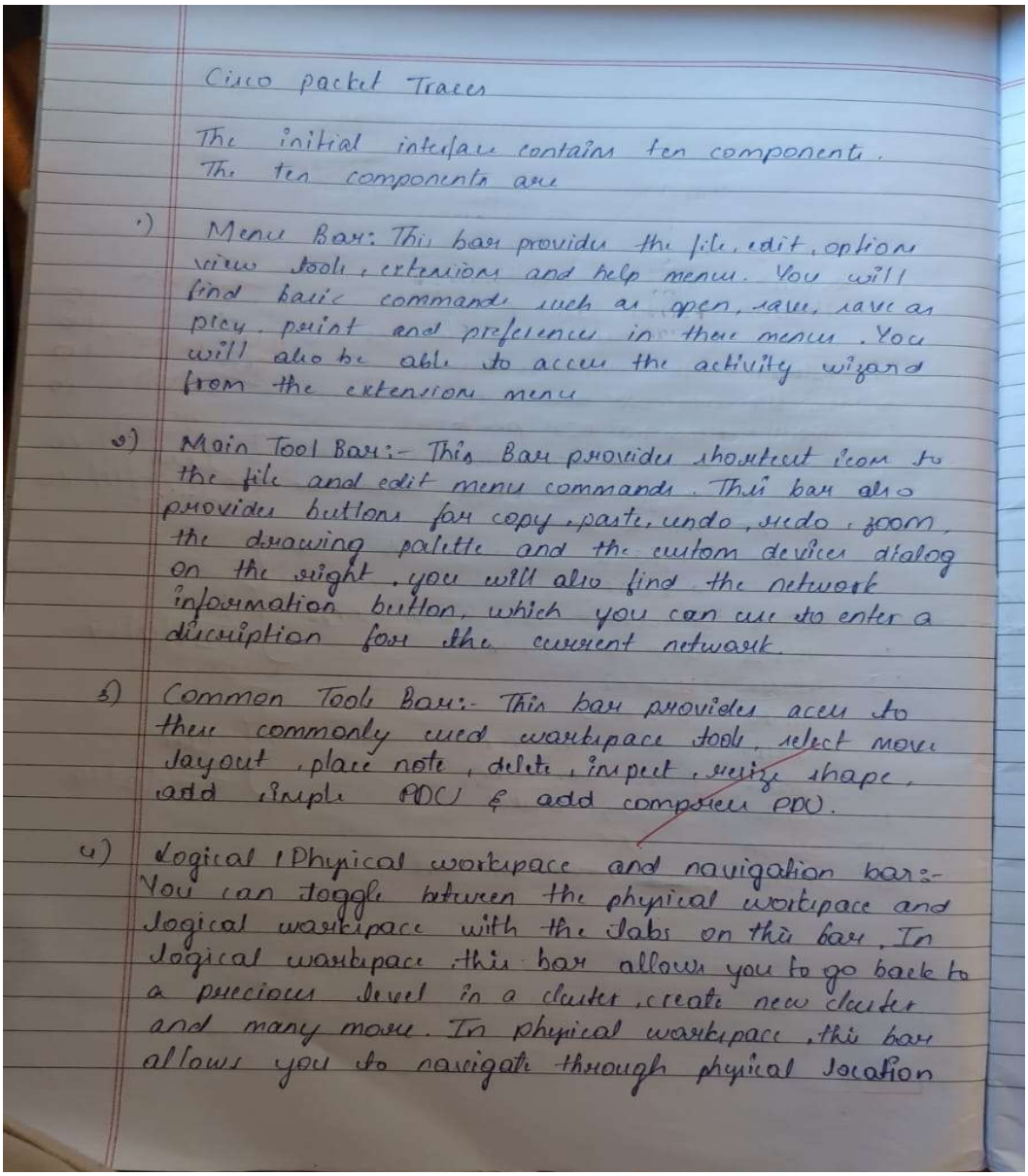
INDEX

CYCLE 2

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



Cisco packet Tracer

The initial interface contains ten components. The ten components are

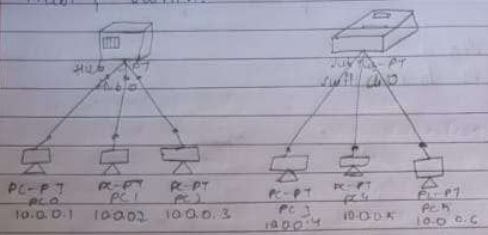
1) Menu Bar: This bar provides the file, edit, option view tools, extensions and help menus. You will find basic commands such as open, save, save as play, print and preferences in these menus. You will also be able to access the activity wizard from the extension menu

2) Main Tool Bar:- This Bar provides shortcut icon to the file and edit menu commands. This bar also provides buttons for copy, paste, undo, redo, zoom, the drawing palette and the custom devices dialog on the right. you will also find the network information button, which you can use to enter a discription for the current network.

3) Common Tools Bar:- This bar provides acess to these commonly used workspace tools, select move layout, place note, delete, inspect, resize shape, add simple PDU & add complex PDU.

4) Logical / Physical workspace and navigation bar:- You can toggle between the physical workspace and logical workspace with the tabs on this bar. In logical workspace this bar allows you to go back to a previous level in a cluster, create new cluster and many more. In physical workspace, this bar allows you to navigate through physical location

2) Hub & Switch.



Aim: To create a simple network consisting of three PCs connected to a central hub and another network with three PCs connected to a switch. This configuration will help observe the behaviour of data transmission using hub & switch device.

Topology:

1. Hub Network:- Three PCs (PC0, PC1, PC2) are connected to a hub (Hub0) using straight-through ethernet cables
   IP addresses:- PC0 = 10.0.0.1, PC1 = 10.0.0.2, PC2 = 10.0.0.3
2. Switch Network:- Three PCs (PC3, PC4, PC5) are connected to a switch (Switch0) using straight-through ethernet cables.
   IP addresses: PC3 = 10.0.0.4, PC4 = 10.0.0.5, PC5 = 10.0.0.6

Procedure:

1. Add 1 hub 1 switch and 6 PC, PC0, PC1, PC2 for the hub; PC3, PC4, PC5 for the switch to the Cisco packet tracer workspace.
2. Use copper straight through cables to connect PC0, PC1 and PC2 to Hub0. Similarly connect PC3, PC4 & PC5 to switch 0 using same type of cables.
3. Assign IP addresses to each PC & obtain subnet mask.
4. Switch to simulation mode to observe data traffic behaviour when packets are sent between the devices.
5. In the hub network, notice how the hub broadcasts packets at all devices, causing potential traffic overload
   In the switch network observe how the switch forwards packets only to the intended recipient reducing unnecessary traffic
6. The hub broadcasts data at all connected devices leading to more network congestion, while the switch efficiently sends data only to the correct device optimizing performance.

Observation:

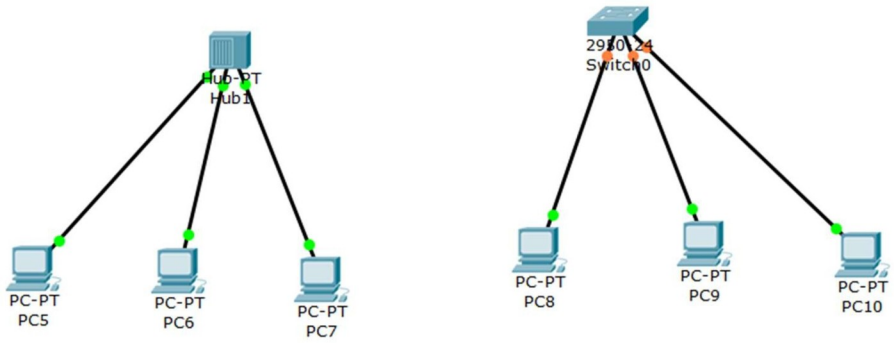
1. The hub broadcasts packets to all devices, which many cause unnecessary traffic.
2. The switch forwards packets only to the appropriate device by learning MAC addresses, making it more efficient in reducing traffic.

Diff between Hubs & Switches.

Hubs	Switches
1. Hubs broadcast data to all devices	1. Switches send it only to the destination
2. Hub create more traffic	2. Switches reduces traffic by directing data
3. Hubs work at physical layer	3. Switches operate at the data link layer
4. Hubs are slower due to shared bandwidth	4. Switches are faster with dedicated bandwidth
5. Hubs are cheaper	5. Switches are more expensive but more efficient

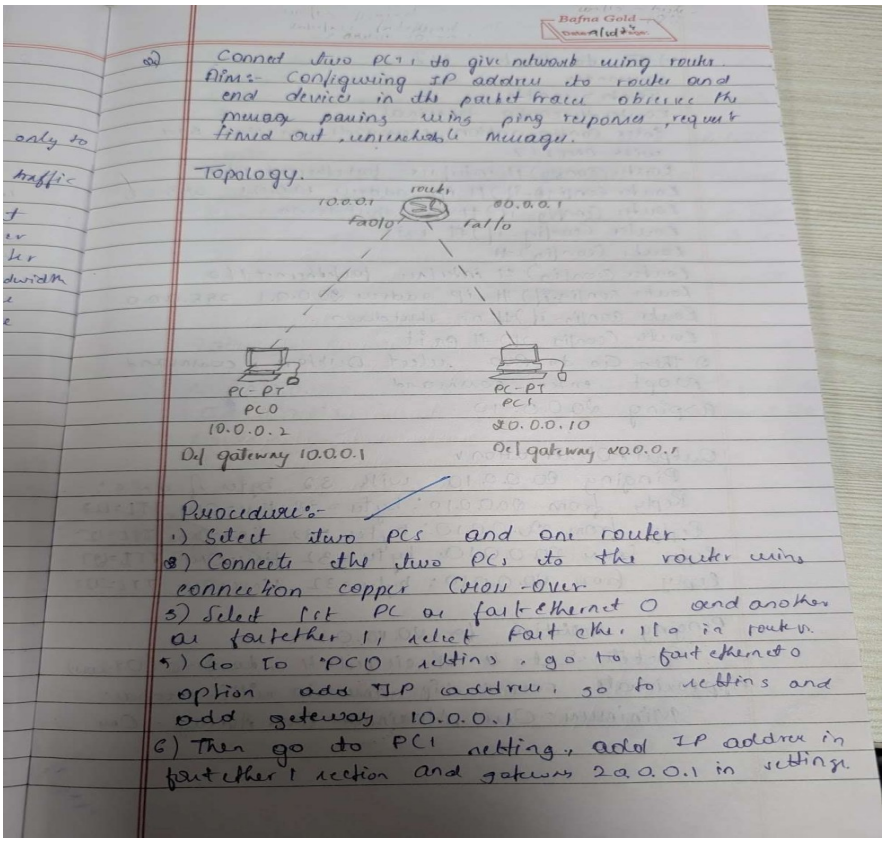
9/10/24

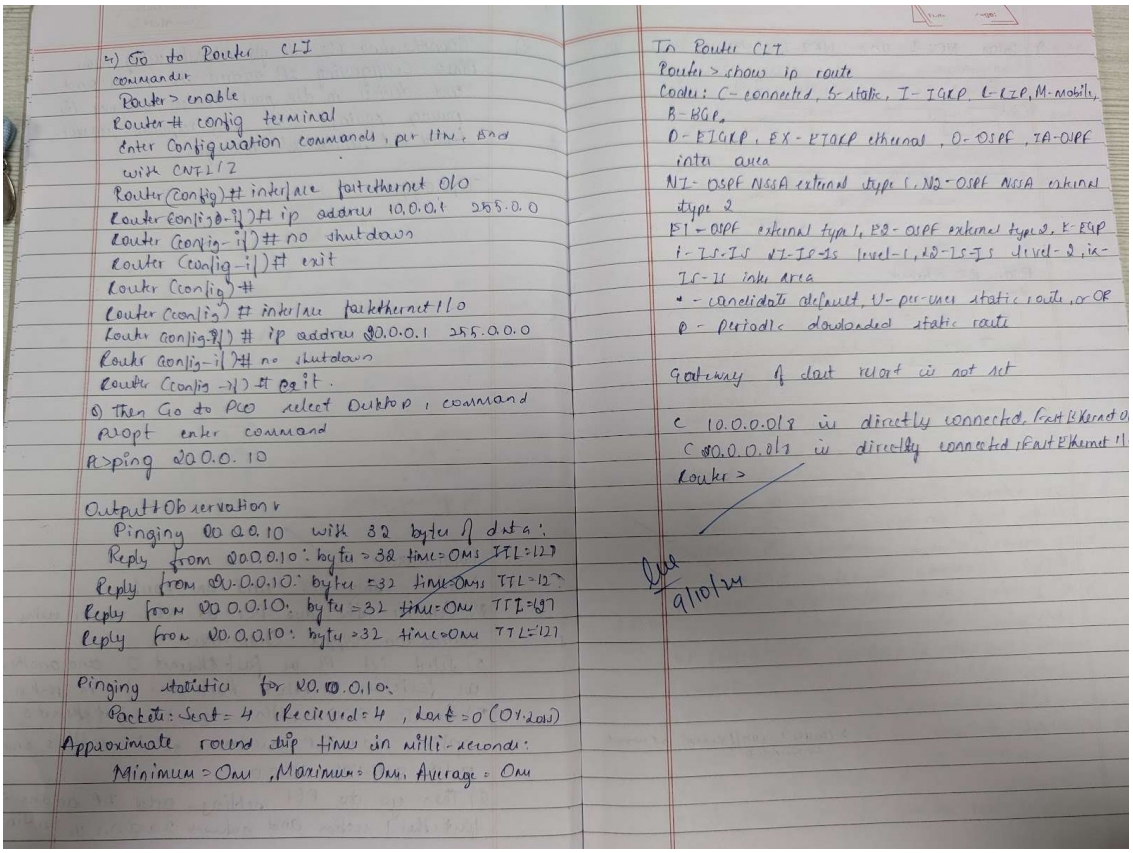
TOPOLOGY:



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:





4) Go to Router CLI
command.
Router> enable
Router# config terminal
Enter Configuration commands, one per line. End
with CNTL/Z
Router(config)# interface fastethernet 0/0
Router(config-if)# ip address 10.0.0.1 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)#
Router(config)# interface fastethernet 1/0
Router(config-if)# ip address 20.0.0.1 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit.
5) Then Go to PC0 select Desktop, command
prompt enter command
A>ping 20.0.0.10

Output & Observation:
Pinging 20.0.0.10 with 32 bytes of data:
Reply from 20.0.0.10: bytes=32 time=0ms TTL=127
Reply from 20.0.0.10: bytes=32 time=0ms TTL=127
Reply from 20.0.0.10: bytes=32 time=0ms TTL=127
Reply from 20.0.0.10: bytes=32 time=0ms TTL=127

Pinging statistics for 20.0.0.10:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms

In Router CLI
Router> show ip route
Codes: C - connected, S - static, I - IGRP, L - LIP, M - mobile,
B - BGP,
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF
inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external
type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, su - IS-IS level-1, L2 - IS-IS level-2, ia -
IS-IS inter area
* - candidate default, U - per-user static route, o - OR
P - periodic downloaded static route

Gateway of last resort is not set

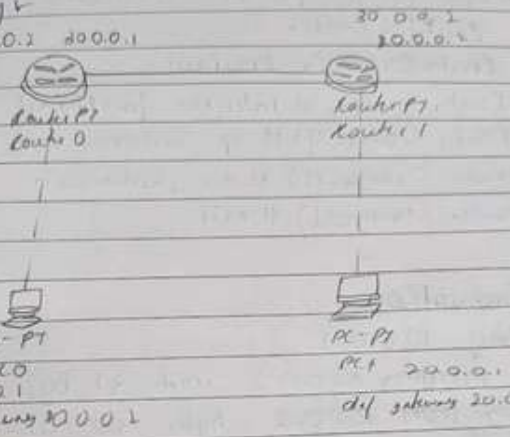
C 10.0.0.0/8 is directly connected, FastEthernet0
C 20.0.0.0/8 is directly connected, FastEthernet1
Router>

9/10/24

8b) Configuring IP address to routers in packet tracer.

Aim:- Configuring the IP address between two routers showing ping response, destination unreachable and request timed out reply.

Topology:

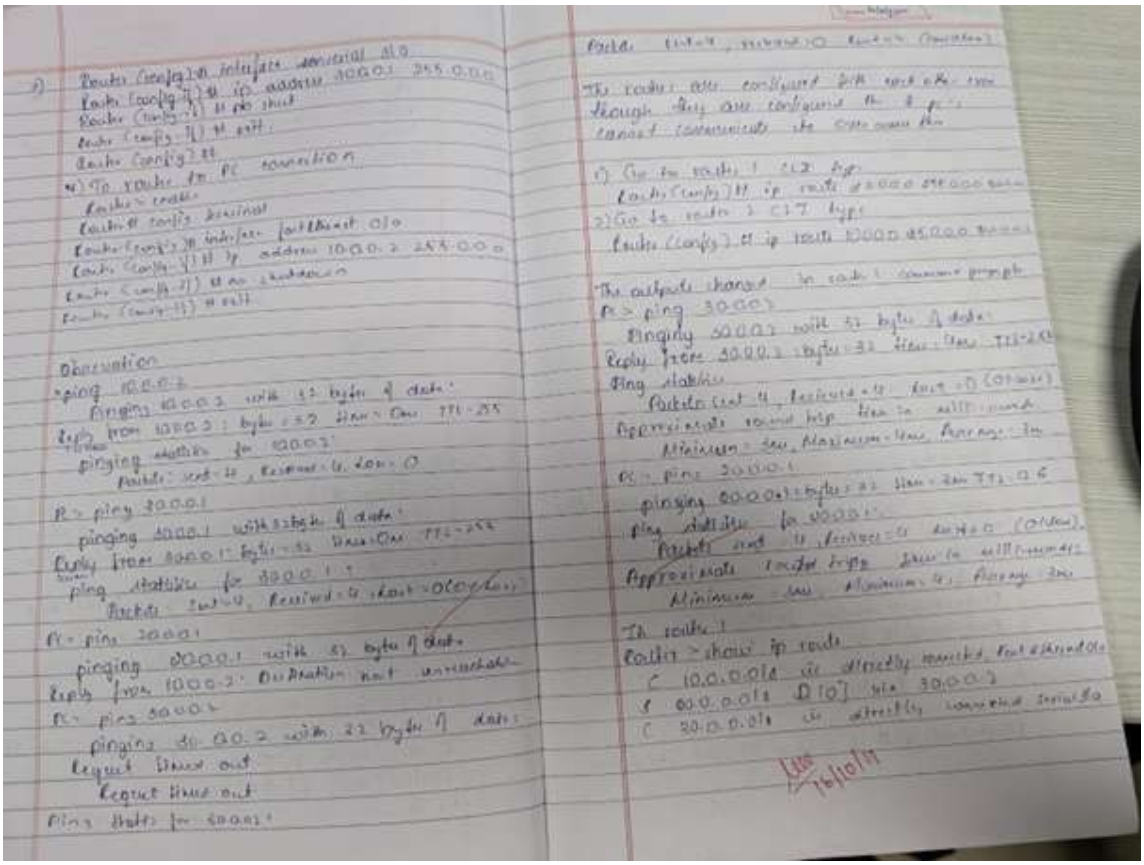


Procedure:

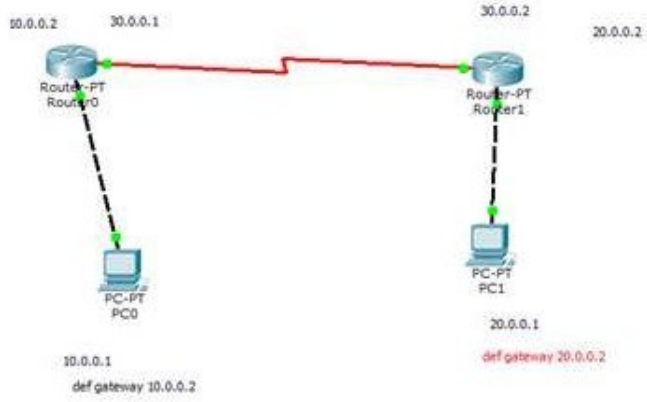
1) Setup the 2 routers and connect each of them to the 2 general end device
2) Setup the connection between router and end device using the command before
3) Connect two routers using Serial DCE given one serial to other serial d/c
4) Config the router1 in the CLI using commands.

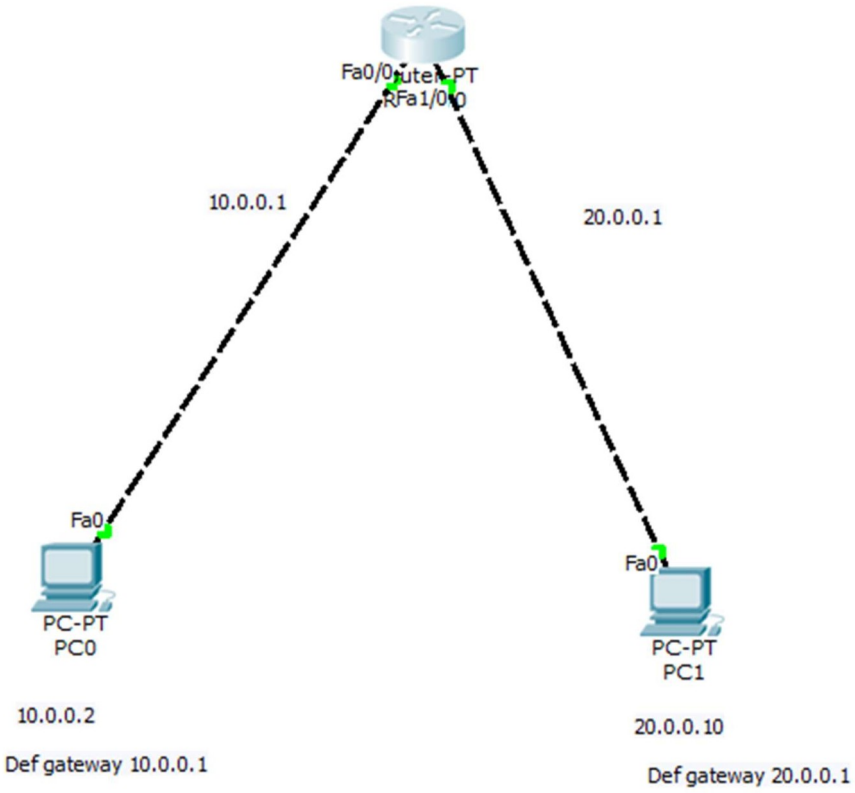
Router (config) # interface Serial 0/0
Router (config-if) # ip address 30.0.0.2 255.0.0
Router (config-if) # no shutdown
Router (config-if) # exit

5) The same way in another router0 CLI

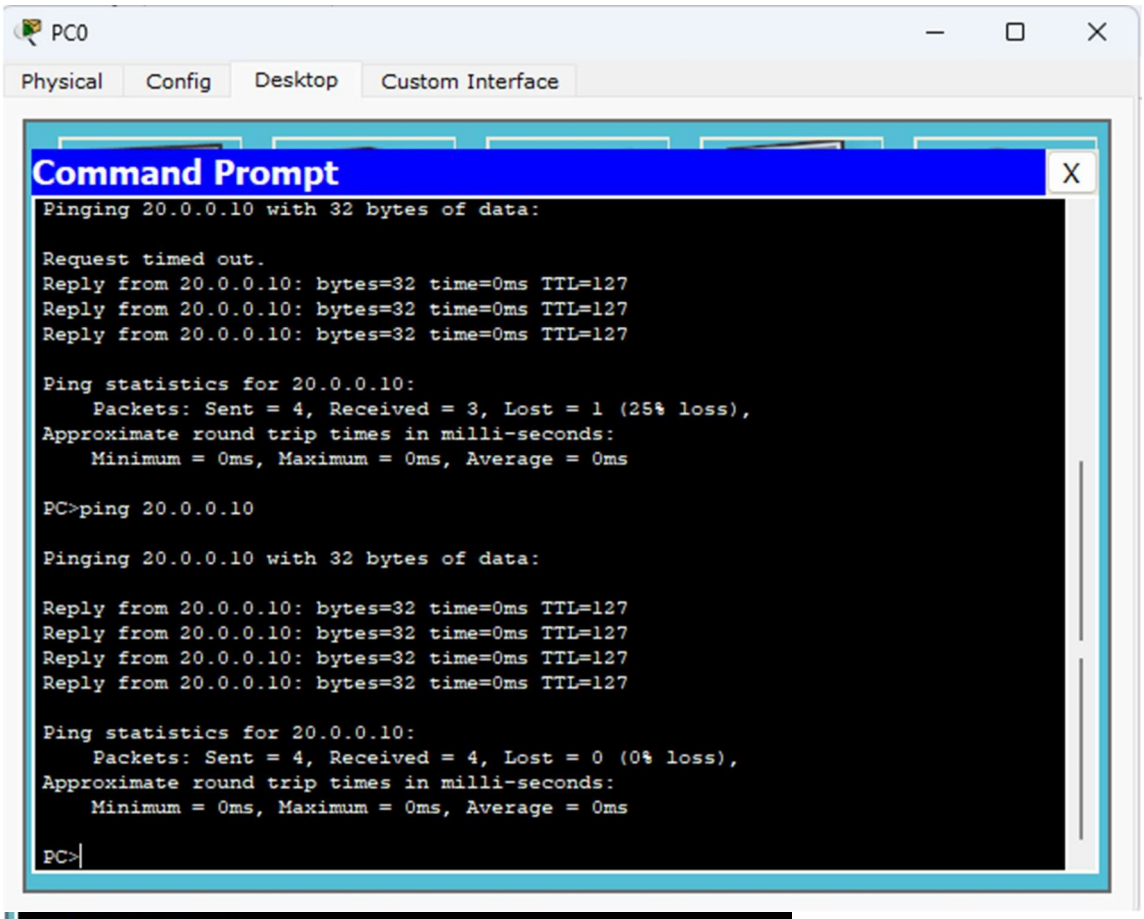


TOPOLOGY:

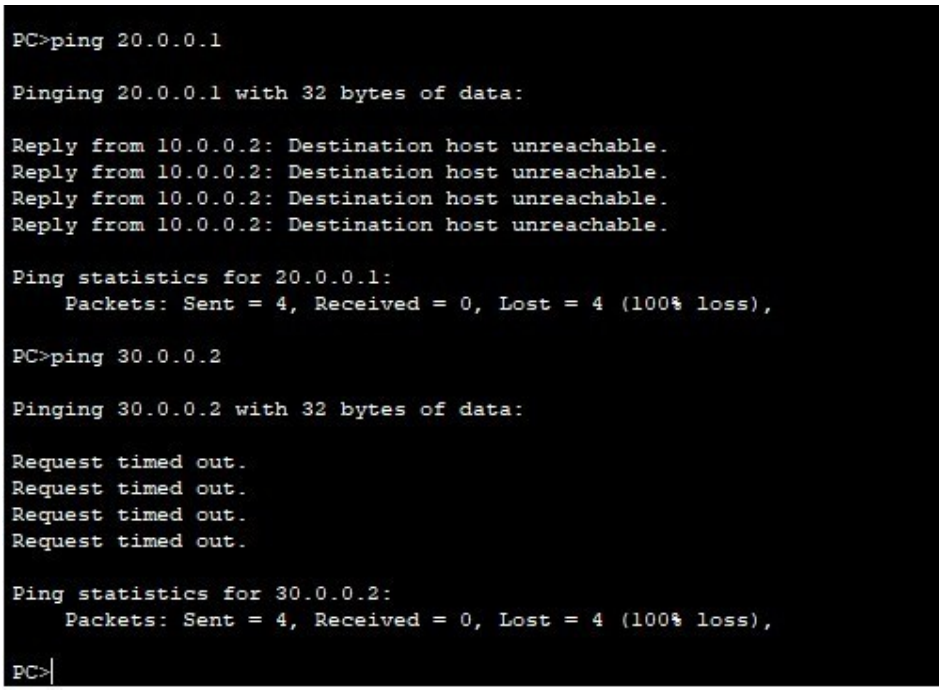

10.0.0.2
30.0.0.1
30.0.0.2
20.0.0.2
Router-PT
Router0
Router-PT
Router1
PC-PT
PC0
PC-PT
PC1
20.0.0.1
10.0.0.1
def gateway 10.0.0.2
def gateway 20.0.0.2


Fa0/0
Fa1/0
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

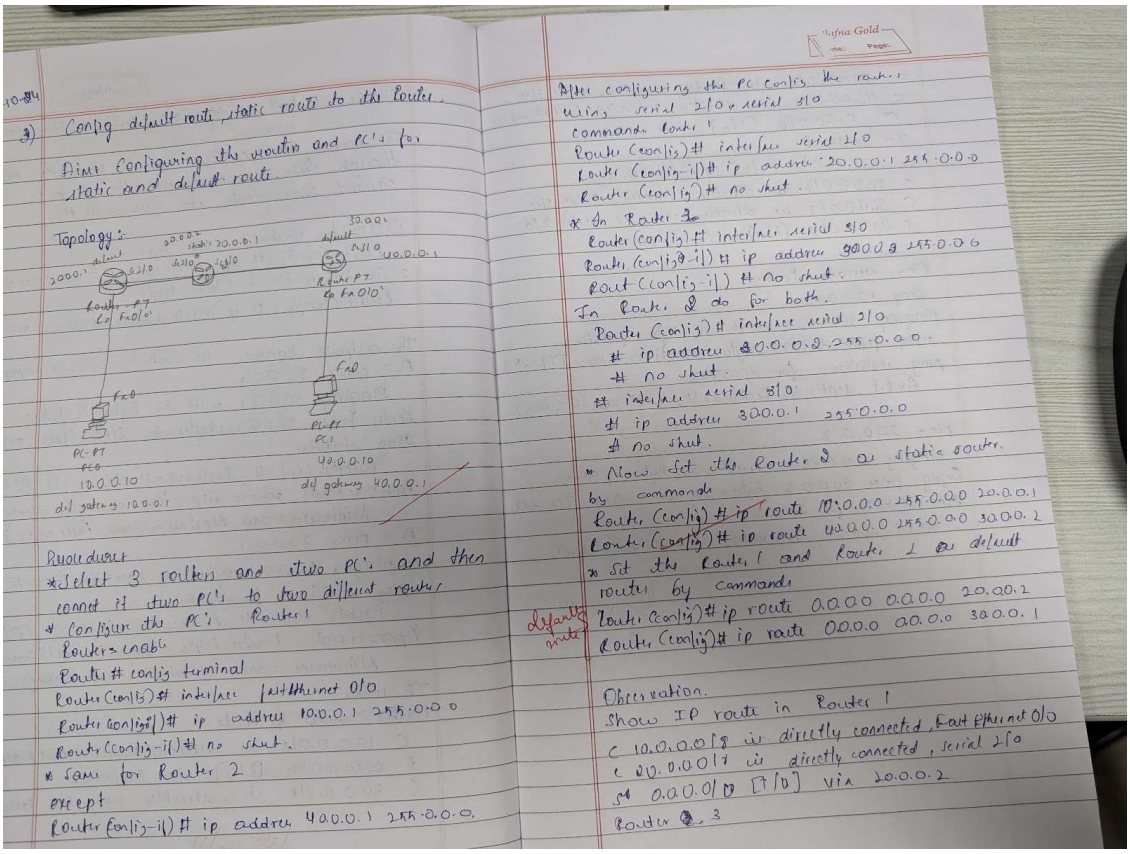
OUTPUT:




```
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
```



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



3) Config default route, static route to the Routers.

Direct Configuring the routers and PC's for static and default route.

Topology :

Procedure

*Select 3 routers and two PC's and then connect it two PC's to two different routers

* Configure the PC's Router 1

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 shut.

* same for Router 2

except

Router(config-if)# ip address 40.0.0.1 255.0.0.0

After configuring the PC config the routers using serial 2/0, serial 3/0

Commands for Router 1

Router(config)# interface serial 2/0

Router(config-if)# ip address 20.0.0.1 255.0.0.0

Router(config-if)# no shut.

* In Router 2

Router(config)# interface serial 3/0

Router(config-if)# ip address 30.0.0.2 255.0.0.0

Router(config-if)# no shut.

In Router 2 do for both.

Router(config)# interface serial 2/0

ip address 20.0.0.2 255.0.0.0

no shut.

interface serial 3/0

ip address 30.0.0.1 255.0.0.0

no shut.

* Now set the Router 2 as static router. by commands

Router(config)# ip route 10.0.0.0 255.0.0.0 20.0.0.1

Router(config)# ip route 40.0.0.0 255.0.0.0 30.0.0.2

* Set the Router 1 and Router 3 as default router by commands

default route

Router(config)# ip route 0.0.0.0 0.0.0.0 20.0.0.2

Router(config)# ip route 0.0.0.0 0.0.0.0 30.0.0.1

Observation.

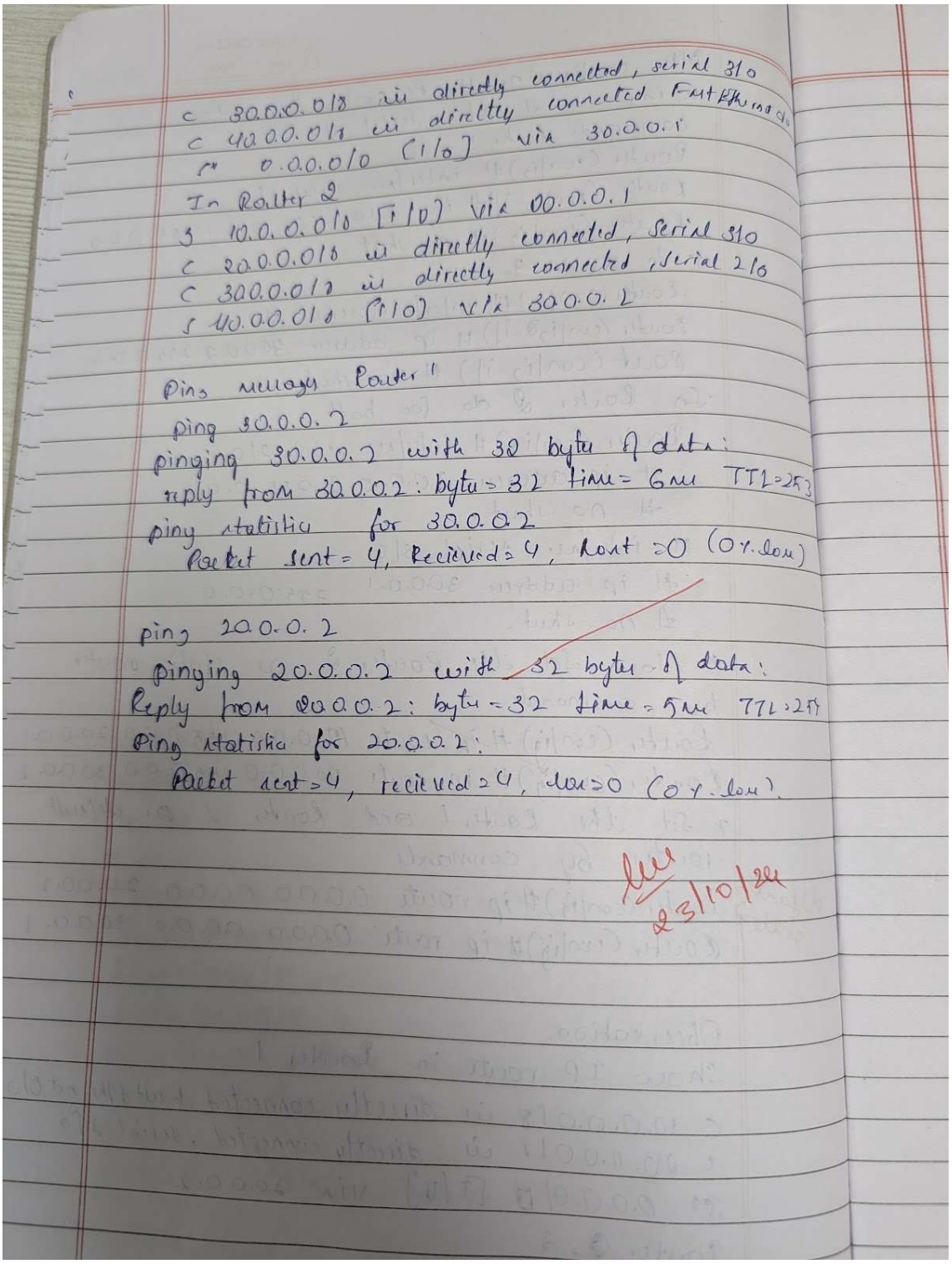
Show IP route in Router 1

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

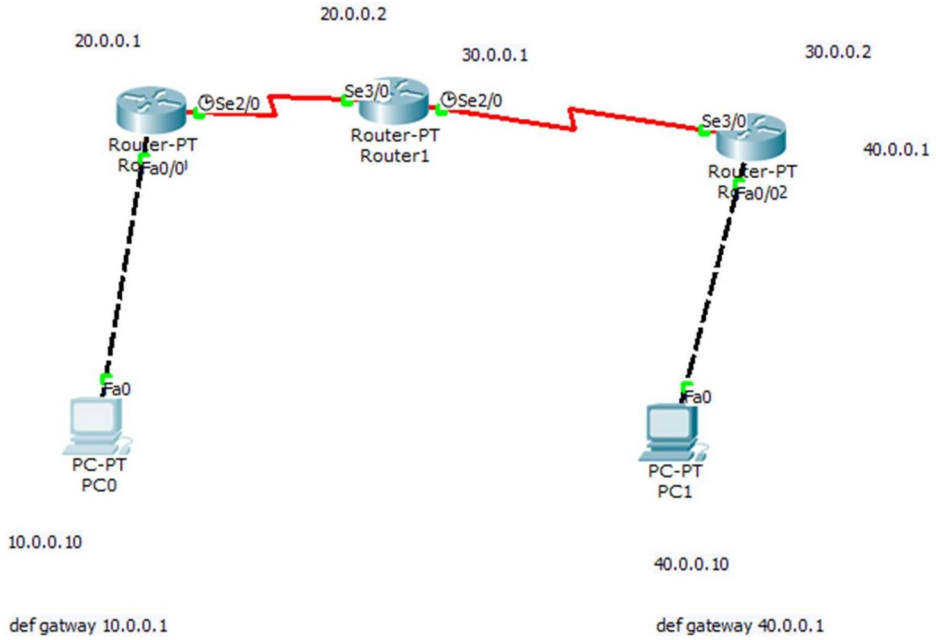
C 20.0.0.0/8 is directly connected, Serial 2/0

S* 0.0.0.0/0 [1/0] via 20.0.0.2

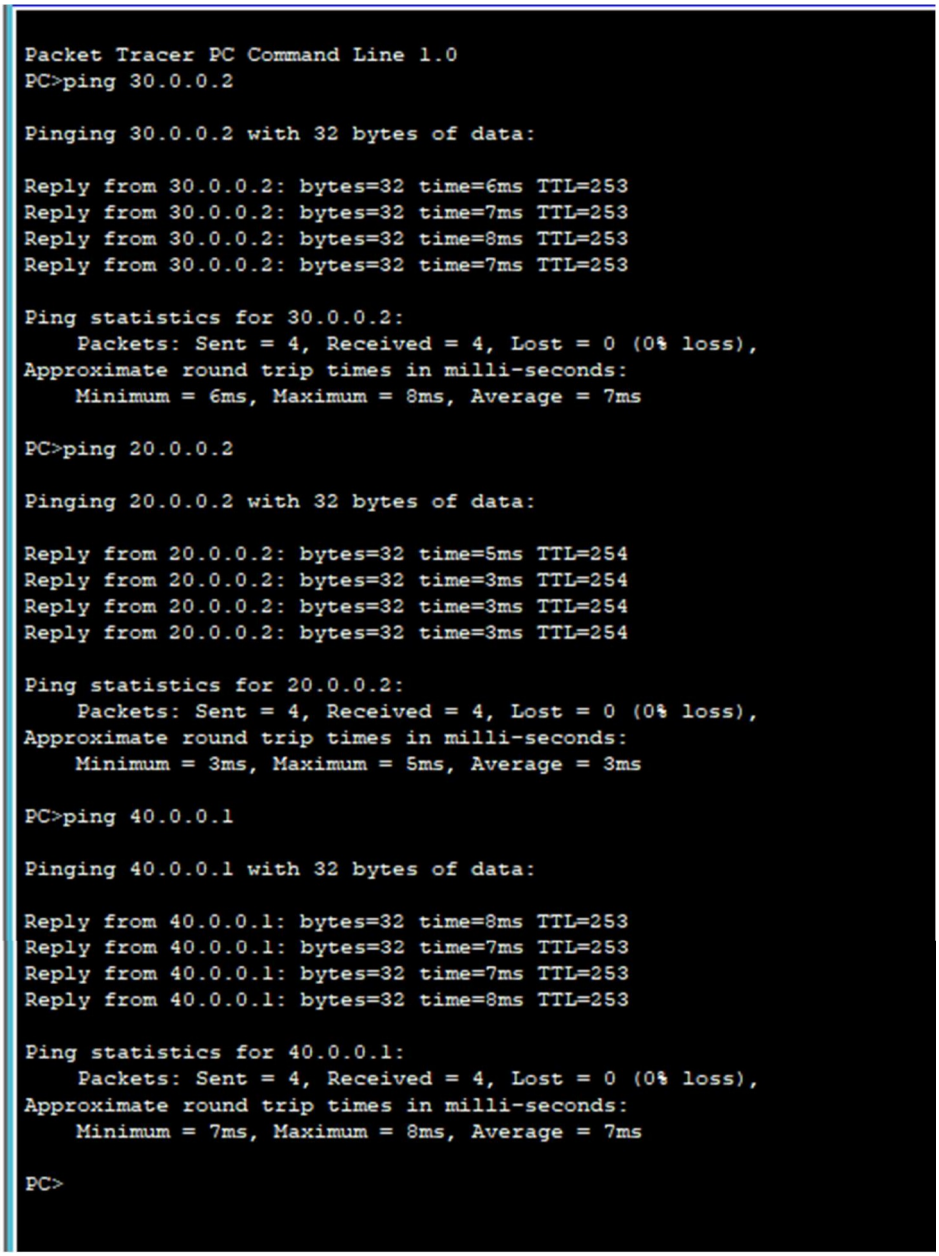
Router 3


C 30.0.0.0/8 is directly connected, serial 3/0
C 40.0.0.0/8 is directly connected FastEthernet
R 0.0.0.0/0 [1/0] via 30.0.0.1
In Router 2
S 10.0.0.0/8 [1/0] via 00.0.0.1
C 20.0.0.0/8 is directly connected, Serial 3/0
C 300.0.0.0/8 is directly connected, Serial 2/0
S 40.0.0.0/8 [1/0] via 300.0.0.2
Pins message Router1
ping 30.0.0.2
pinging 30.0.0.2 with 32 byte of data:
reply from 30.0.0.2: byte=32 time=6ms TTL=253
ping statistic for 30.0.0.2
Packet sent=4, Received=4, Lost=0 (0% loss)
ping 20.0.0.2
pinging 20.0.0.2 with 32 byte of data:
Reply from 20.0.0.2: byte=32 time=5ms TTL=254
Ping statistic for 20.0.0.2:
Packet sent=4, received=4, lost=0 (0% loss).
23/10/24

TOPOLOGY:


20.0.0.2
20.0.0.1
30.0.0.1
30.0.0.2
Se2/0
Se3/0
Se2/0
Se3/0
Router-PT
Router1
Router-PT
Router-PT
40.0.0.1
Fa0/01
Fa0/02
Fa0
Fa0
PC-PT
PC0
PC-PT
PC1
10.0.0.10
40.0.0.10
def gatway 10.0.0.1
def gateway 40.0.0.1

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:

Bafna Gold

Design a DHCP within LAN and enable outside LAN

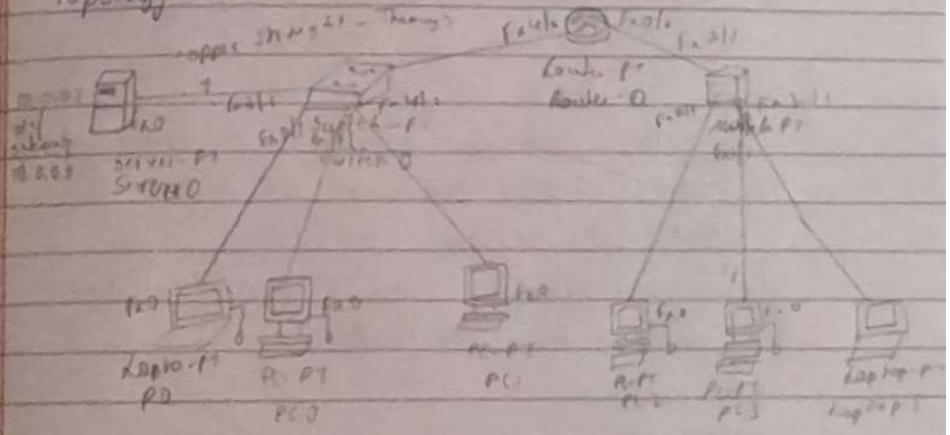
DHCP - Dynamic host control protocol, within ... server network configured

... IP address ...

... end devices ...

Aim: Connecting within and outside of LAN

Topology:



Procedure:

→ Take switch, 3 end devices and one server as end devices

→ Go To server desktop, config, service, DHCP

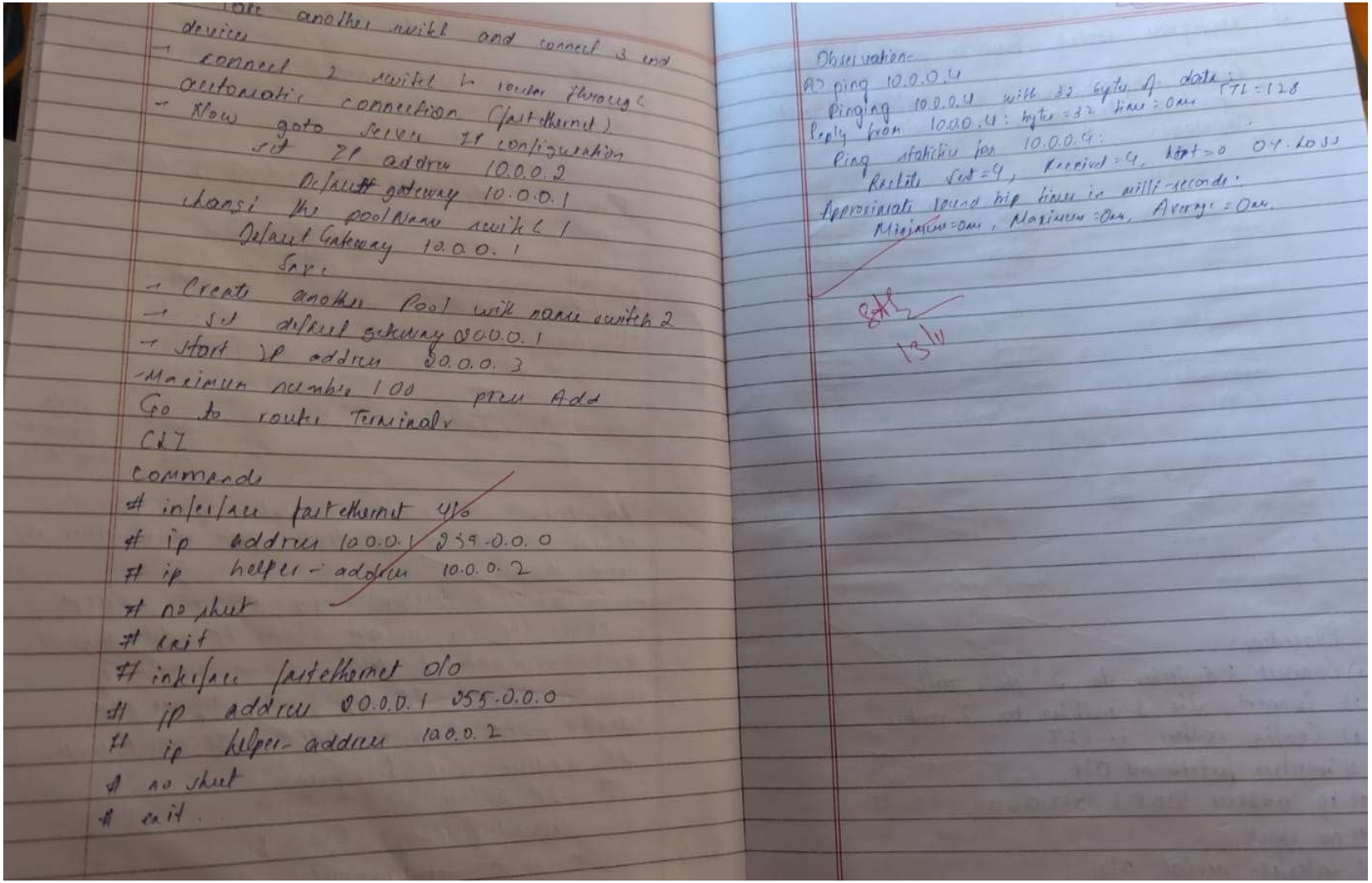
→ change the service on, change IP address, Maximum number of devices and give add

→ Go to desktop IP configuration of all end devices and turn on DHCP. The automatic IP address will be given

The IP address = 10.0.0.3

Default gateway = 10.0.0.1

Go to config → Services DHCP set pool Name Default gateway 10.0.0.0 Start IP address 10.0.0.3 and Max number 100

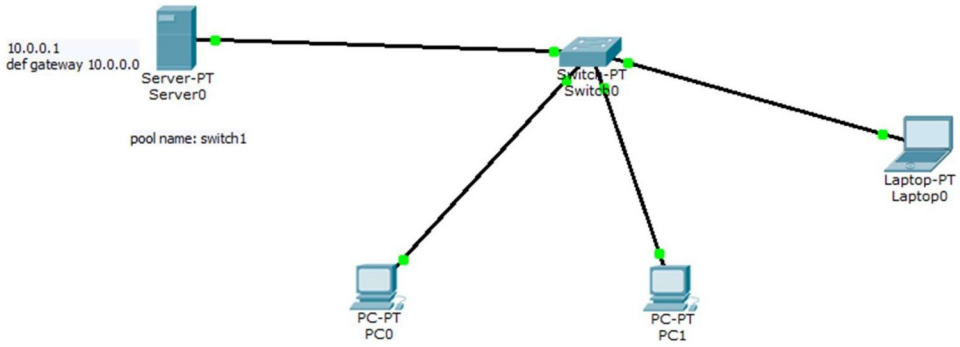


Take another switch and connect 3 end devices
- connect 2 switch to router through automatic connection (fastethernet)
- Now goto server IP configuration
  set IP address 10.0.0.2
  Default gateway 10.0.0.1
- Change the pool Name switch 1
  Default Gateway 10.0.0.1
  Save
- Create another Pool with name switch 2
- Set default gateway 10.0.0.1
- Start IP address 10.0.0.3
- Maximum number 100 then Add
- Go to router Terminal
- CLI

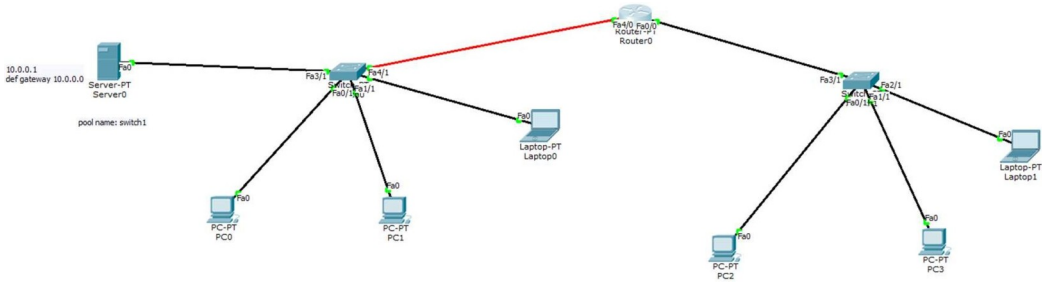
Commands
interface fastethernet 4/0
ip address 10.0.0.1 255.0.0.0
ip helper-address 10.0.0.2
no shut
exit
interface fastethernet 0/0
ip address 10.0.0.1 255.0.0.0
ip helper-address 10.0.0.2
no shut
exit

Observation-
C:\> ping 10.0.0.4
Pinging 10.0.0.4 with 32 bytes of data:
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Ping statistics for 10.0.0.4:
Packets: Sent=4, Received=4, Lost=0 0% Loss
Approximate round trip times in milli-seconds:
Minimum=0ms, Maximum=0ms, Average=0ms

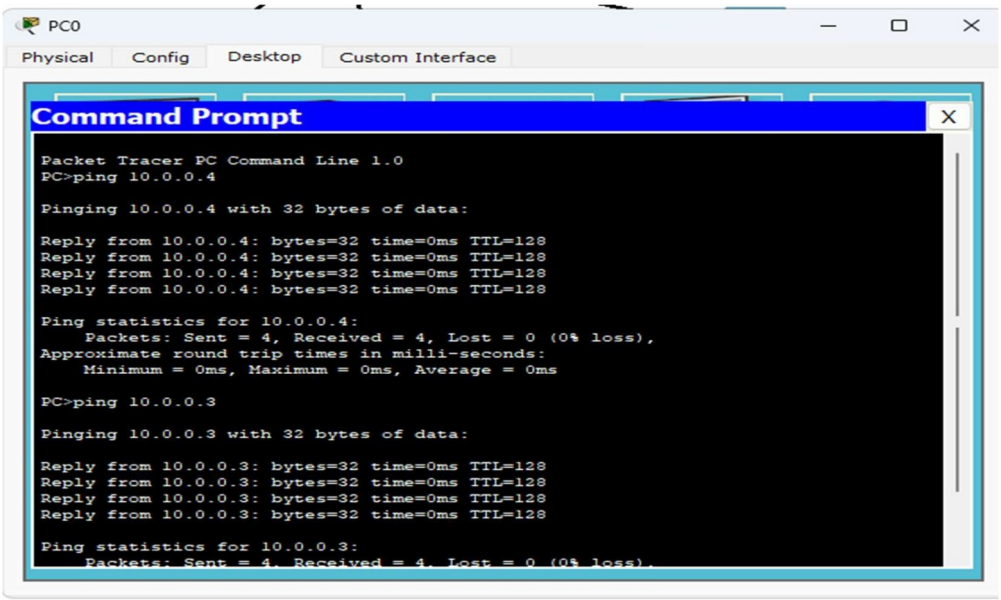
TOPOLOGY:
Within lan



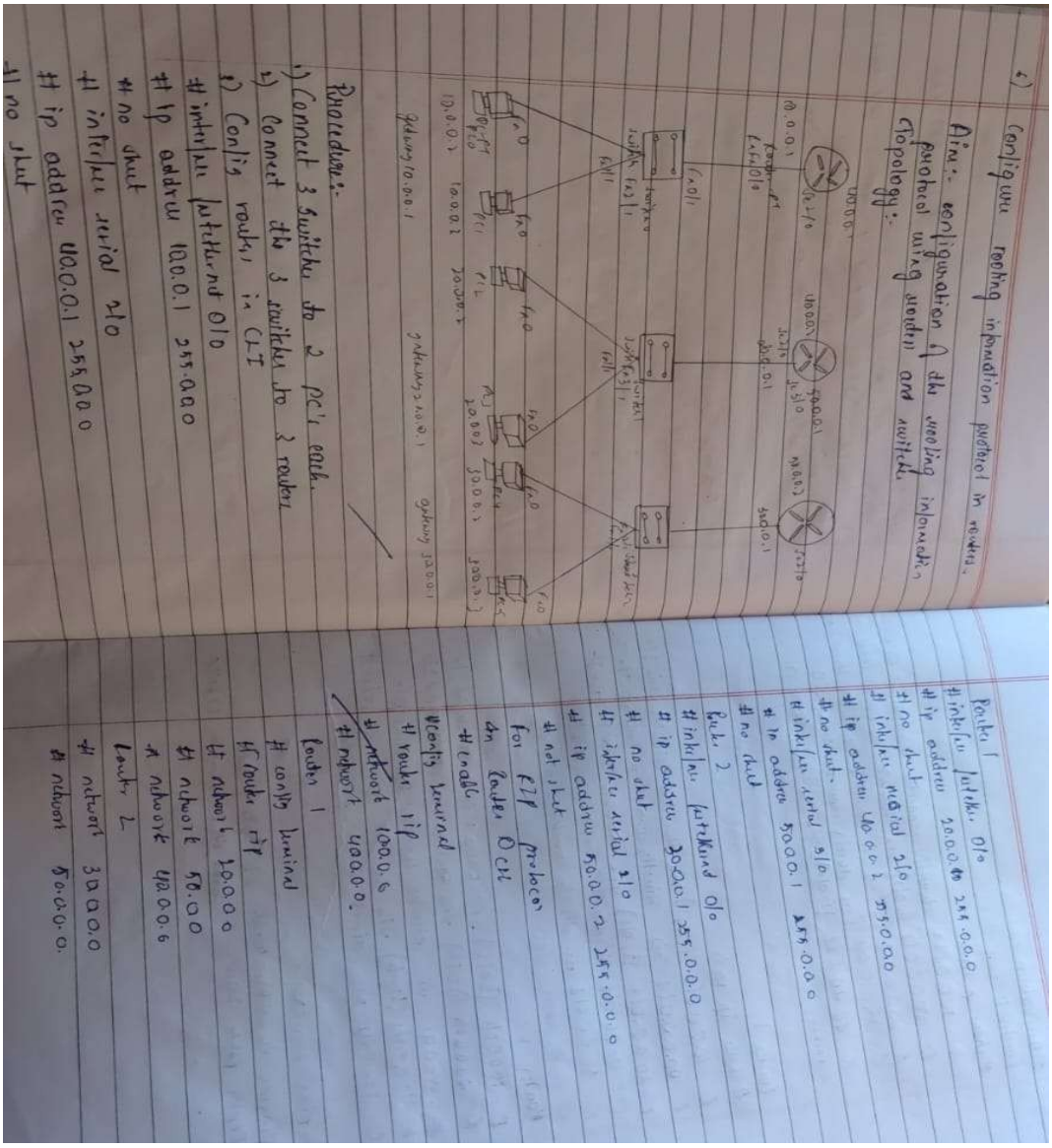
Outside lan



OUTPUT:



PROGRAM5: Configure RIP routing Protocol in Routers
OBSERVATION:



Observation.

Router 0

#show ip route

C 10.0.0.0/8 directly connected, FastEthernet 0/0

R 20.0.0.0/8 [120/1] via 40.0.0.2, 00:00:09 Serial 2/0

R 30.0.0.0/8 [120/2] via 40.0.0.2 00.00.09, Serial 2/0

C 40.0.0.0/8 is directly connected Serial 2/0

R 50.0.0.0/8 [120/1] via 40.0.0.2, 00:00:09 Serial 2/0.

Router 1

#show ip route

R 10.0.0.0/8 [120/1] via 40.0.0.1 00:00:13, Serial 2/0

C 20.0.0.0/8 is directly connected FastEthernet 0/0

R 30.0.0.0/8 [120/1] via 50.0.0.2 00:00:13, Serial 3/0

C 40.0.0.0/8 is directly connected, Serial 2/0

C 50.0.0.0/8 is directly connected, Serial 3/0

Router 2

R 10.0.0.0/8 [120/2) via 50.0.0.1 00:00:01, Serial 2/0

R 20.0.0.0/8 [120/1) via 50.0.0.1 00:00:01, Serial 2/0

C 50.0.0.0/8 is directly connected FastEthernet 0/0

R 40.0.0.0/8 [120/1] via 50.0.0.1 00:00:01 Serial 2/0

C 50.0.0.0/8 is direct connected Serial 2/0

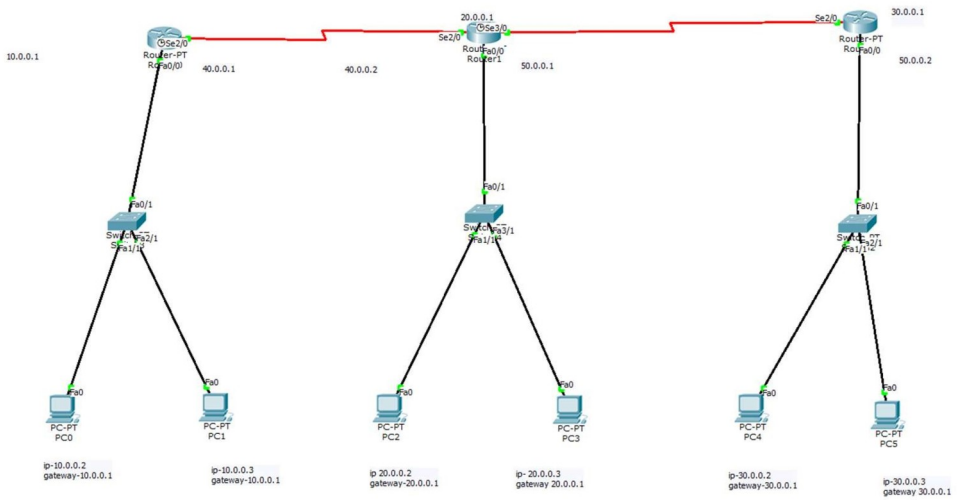
>ping 20.0.0.2

Pinging 20.0.0.2 with 32 bytes of data

Reply from 20.0.0.2 bytes=32 time=5ms TTL=126

.

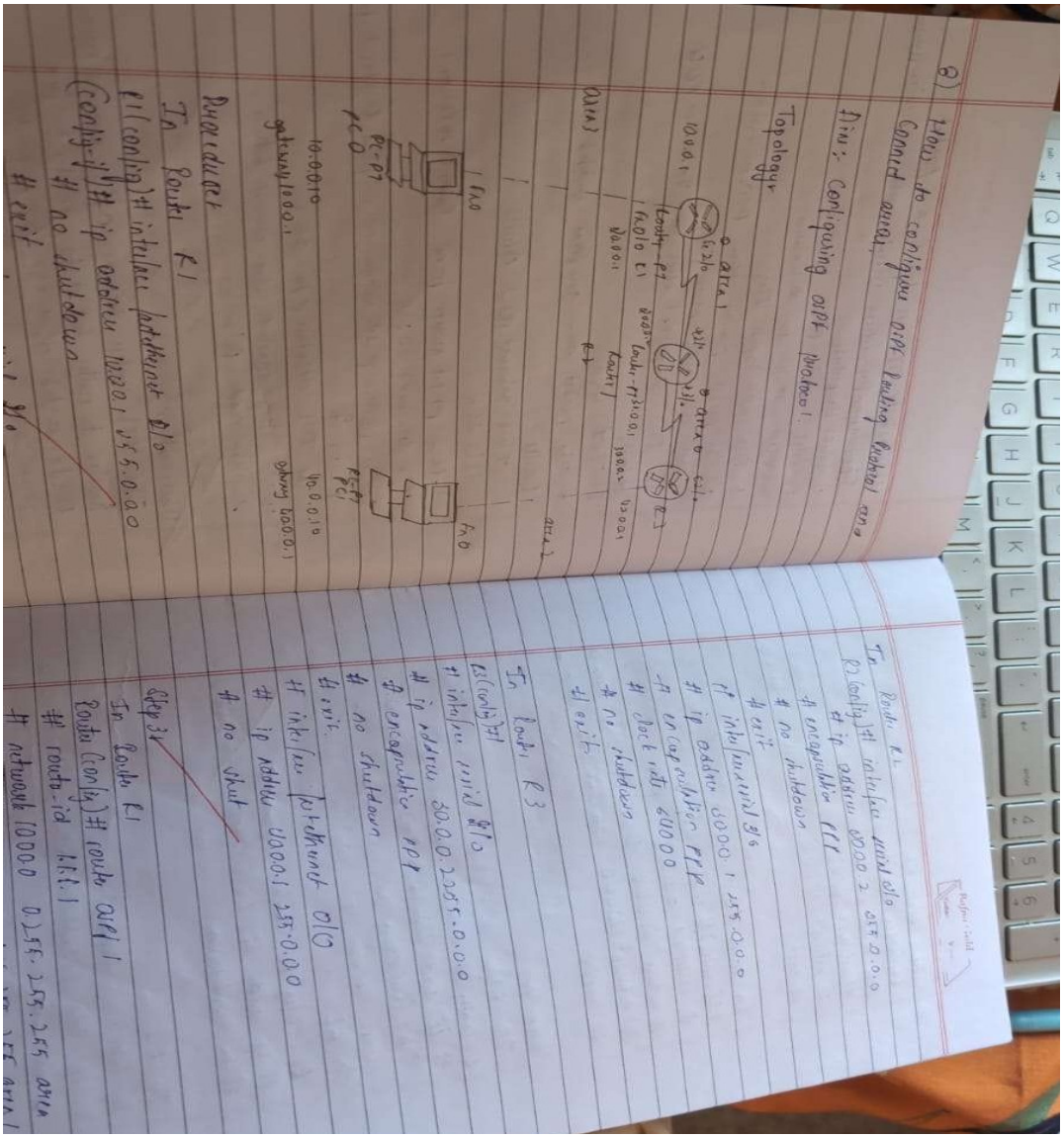
TOPOLOGY:

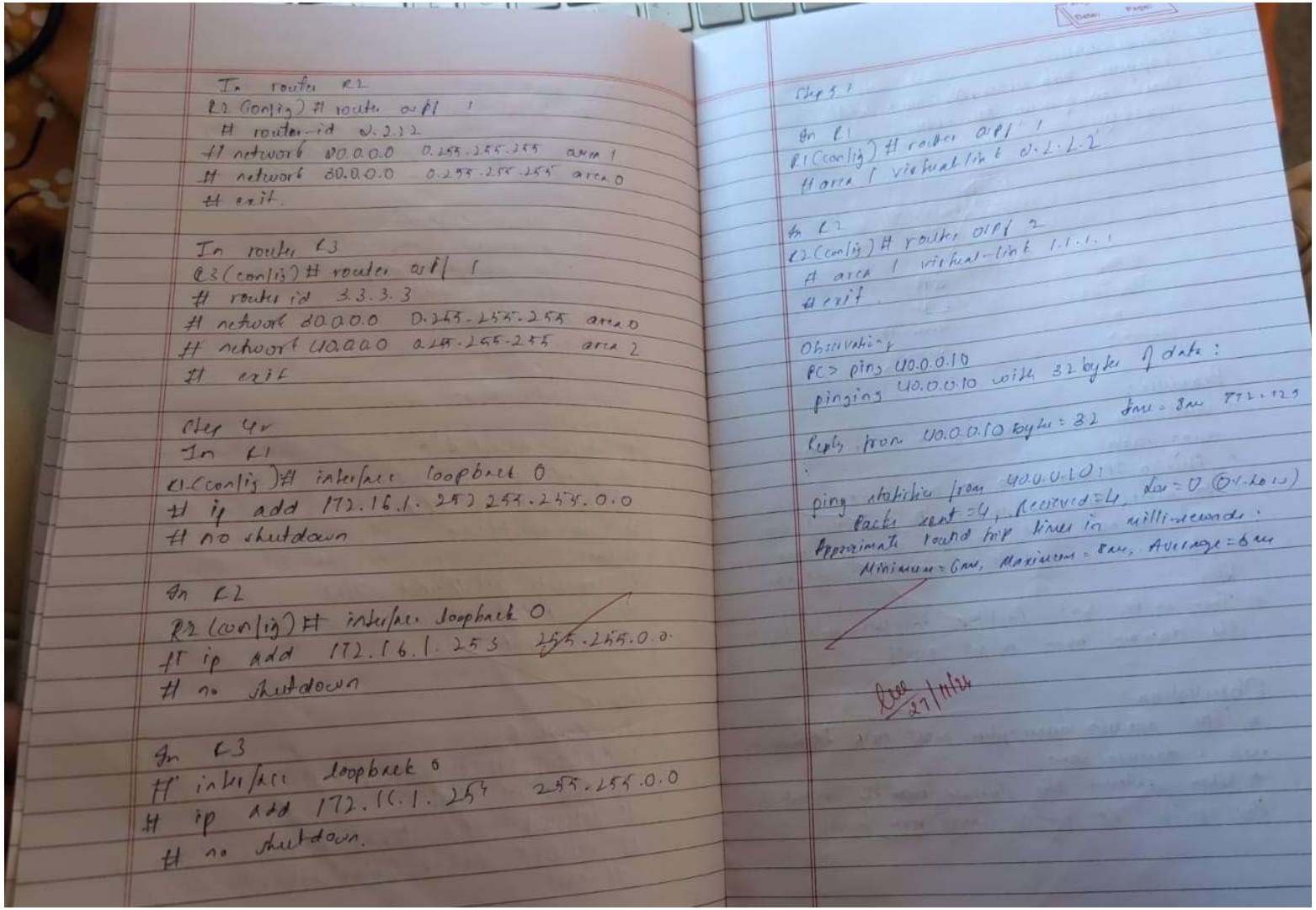


OUTPUT:

```
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:





In router R2
R2 (config) # 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.

In router R3
R3 (config) # 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 4:
In R1
R1 (config) # interface loopback 0
ip add 172.16.1.252 255.255.0.0
no shutdown

In R2
R2 (config) # interface loopback 0
ip add 172.16.1.253 255.255.0.0
no shutdown

In R3
interface loopback 0
ip add 172.16.1.254 255.255.0.0
no shutdown.

Step 5:

In R1
R1 (config) # router ospf 1
area 1 virtual-link 2.2.2.2

In R2
R2 (config) # router ospf 2
area 1 virtual-link 1.1.1.1
exit

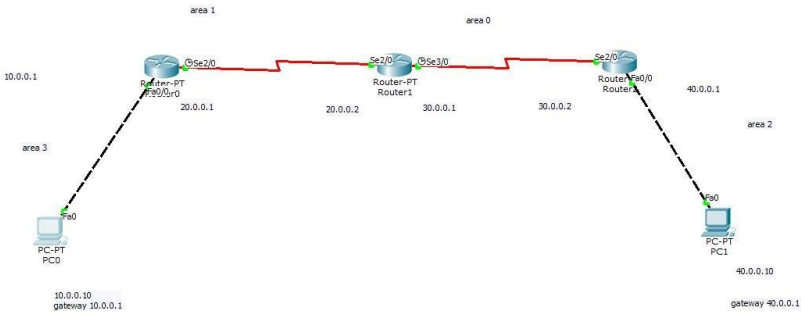
Observation:
PC> ping 40.0.0.10
pinging 40.0.0.10 with 32 bytes of data:

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

ping statistics for 40.0.0.10:
Packets: sent=4, Received=4, Lost=0 (0% loss)
Approximate round trip times in milliseconds:
Minimum=6ms, Maximum=8ms, Average=6ms

Lab 27/11/24

TOPOLOGY:

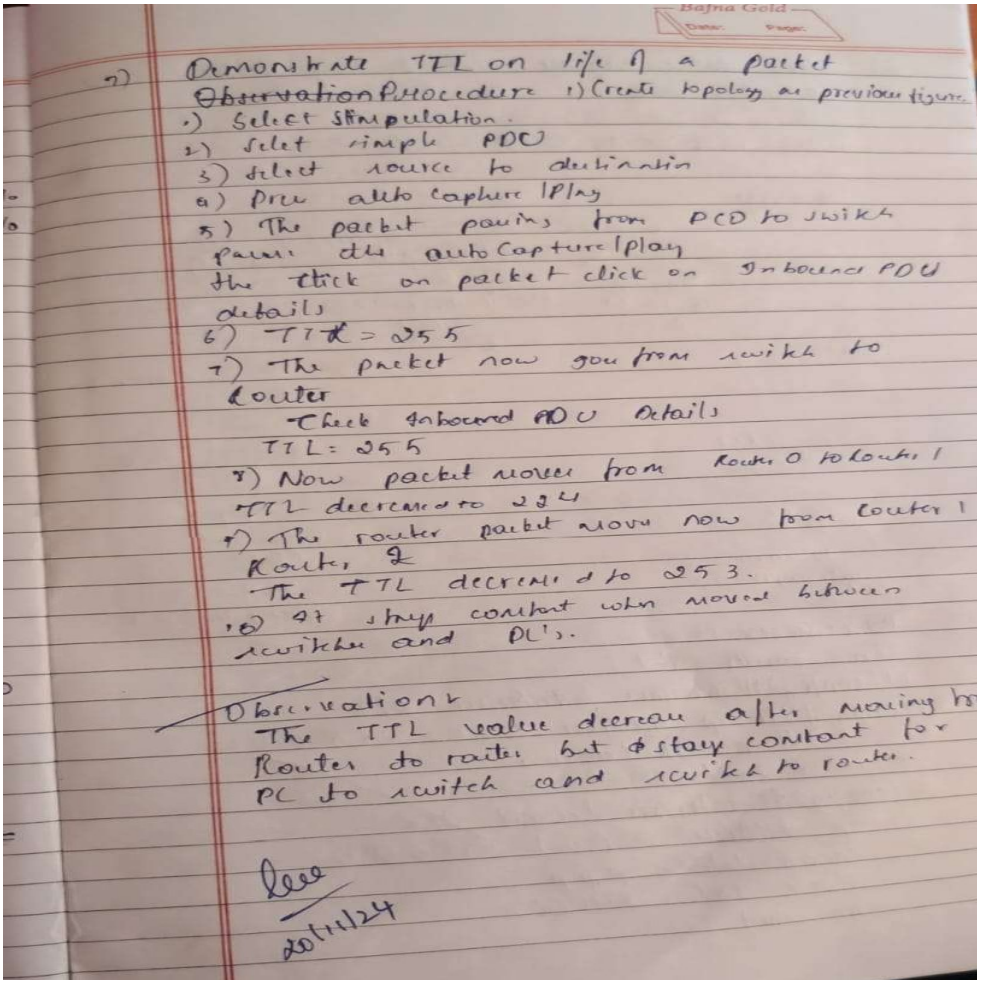


OUTPUT:

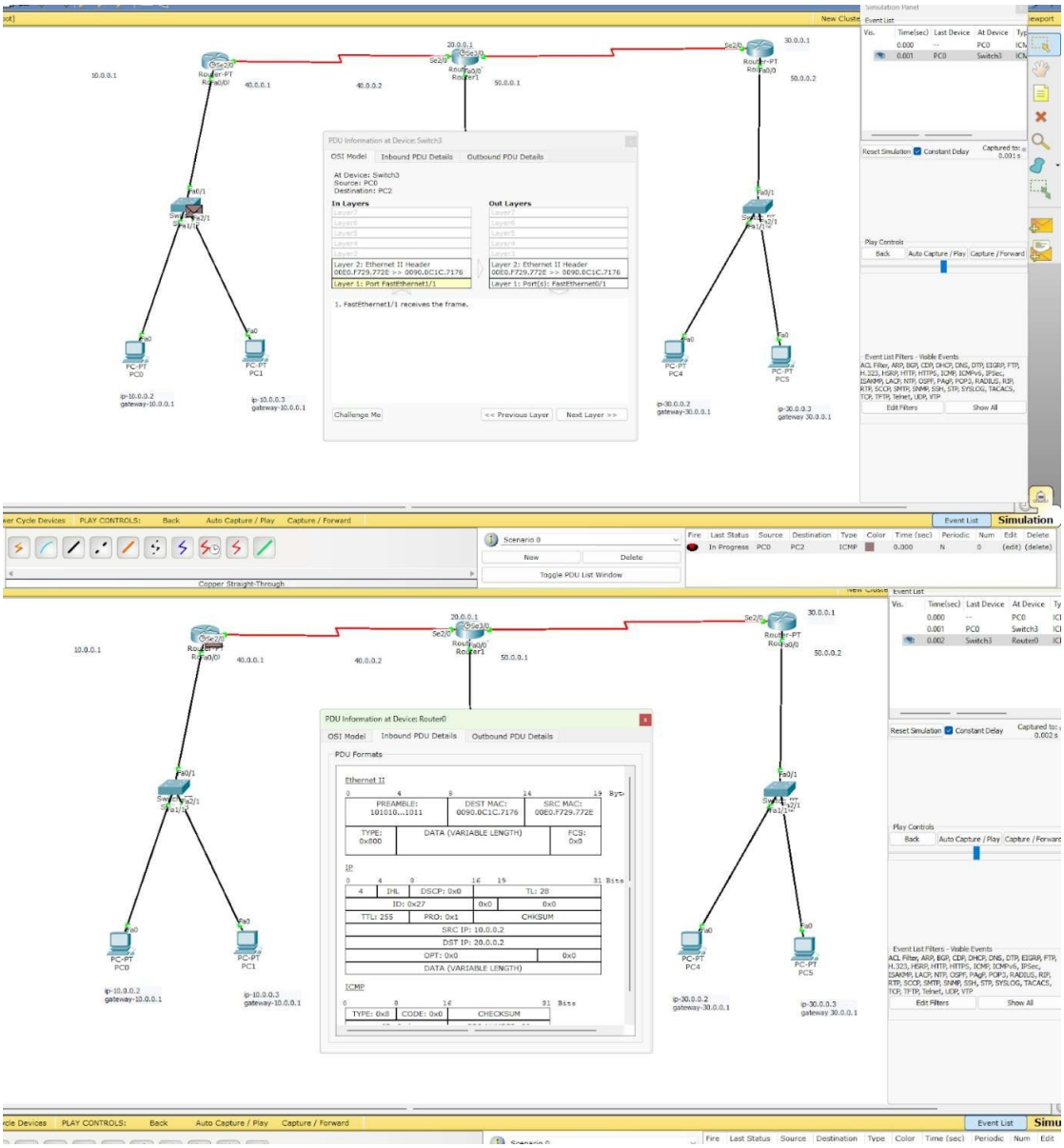
```
Packet Tracer PC Command Line 1.0
PC>ping 40.0.0.10

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
```

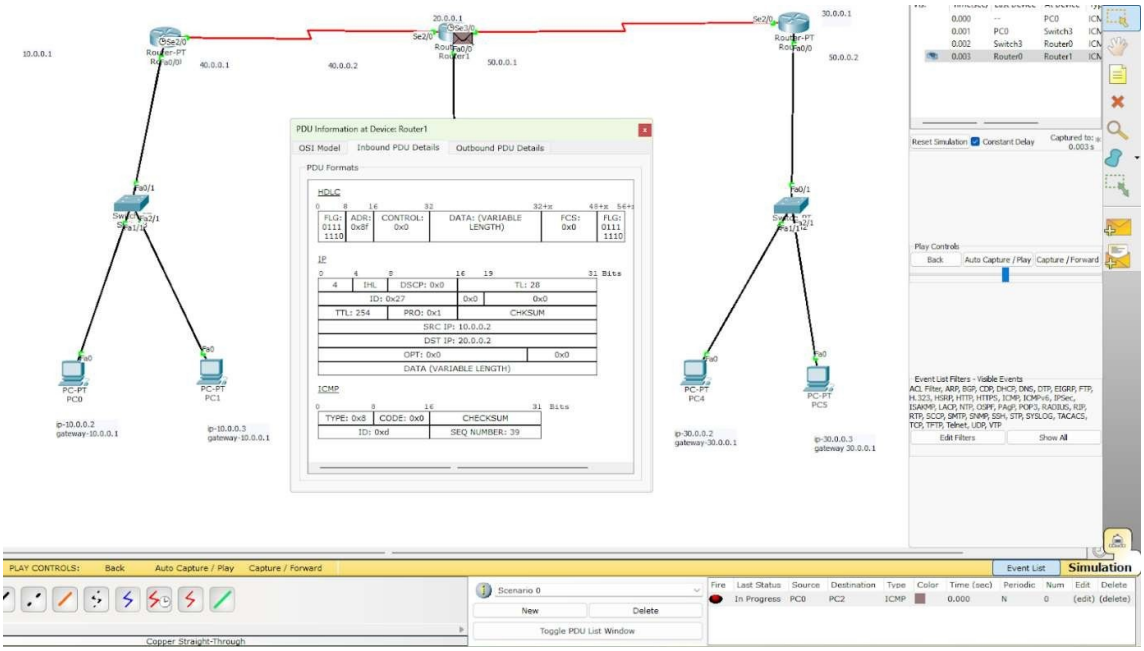
PROGRAM7: Demonstrate the TTL/ Life of a Packet

OBSERVATION:



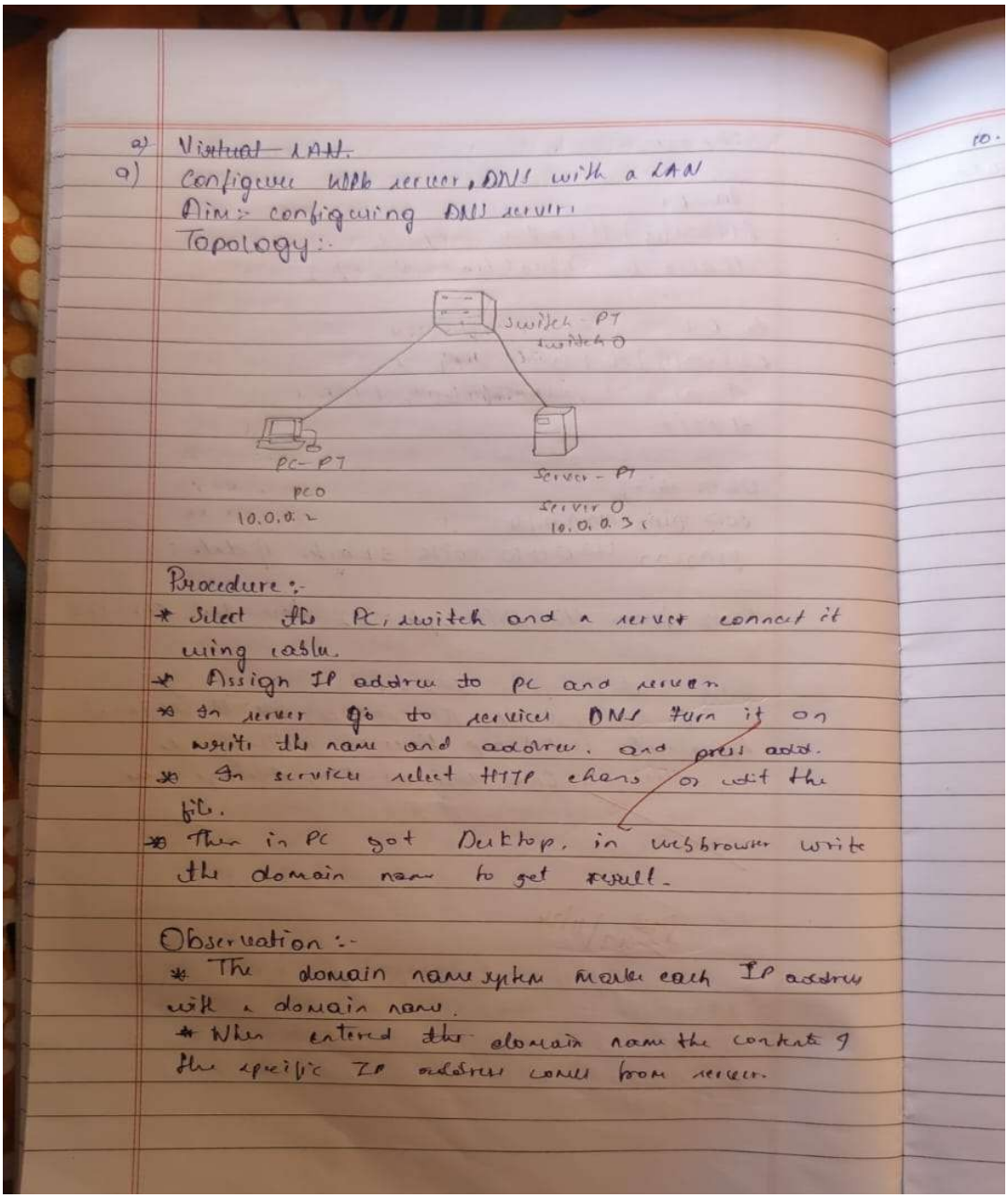
TOPOLOGY:


PDU Information at Device: Switch3
OSI Model
Inbound PDU Details
Outbound PDU Details
At Device: Switch3
Source: PC0
Destination: PC2
In Layers
Out Layers
Layer 2: Ethernet II Header 00E0.F729.772E >> 0090.0C1C.7176
Layer 1: Port FastEthernet1/1
Layer 2: Ethernet II Header 00E0.F729.772E >> 0090.0C1C.7176
Layer 1: Port(s): FastEthernet0/1
1. FastEthernet1/1 receives the frame.
Challenge Me
<< Previous Layer
Next Layer >>
PDU Information at Device: Router0
PDU Formats
Ethernet II
PREAMBLE: 101010...1011
DEST MAC: 0090.0C1C.7176
SRC MAC: 00E0.F729.772E
TYPE: 0x800
DATA (VARIABLE LENGTH)
FCS: 0x0
IP
ICMP
TYPE: 0x8
CODE: 0x0
CHECKSUM

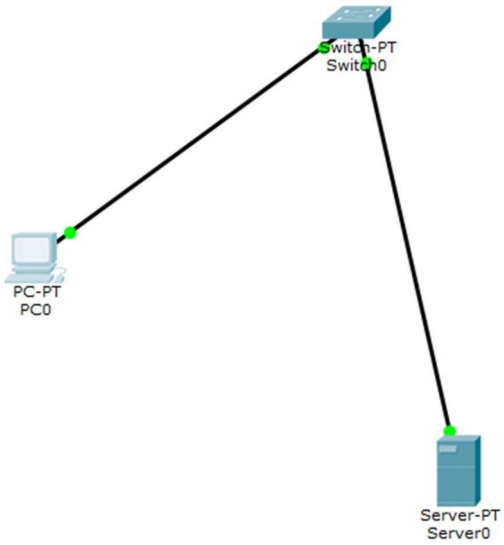


PROGRAM8: Configure Web Server, DNS within a LAN.

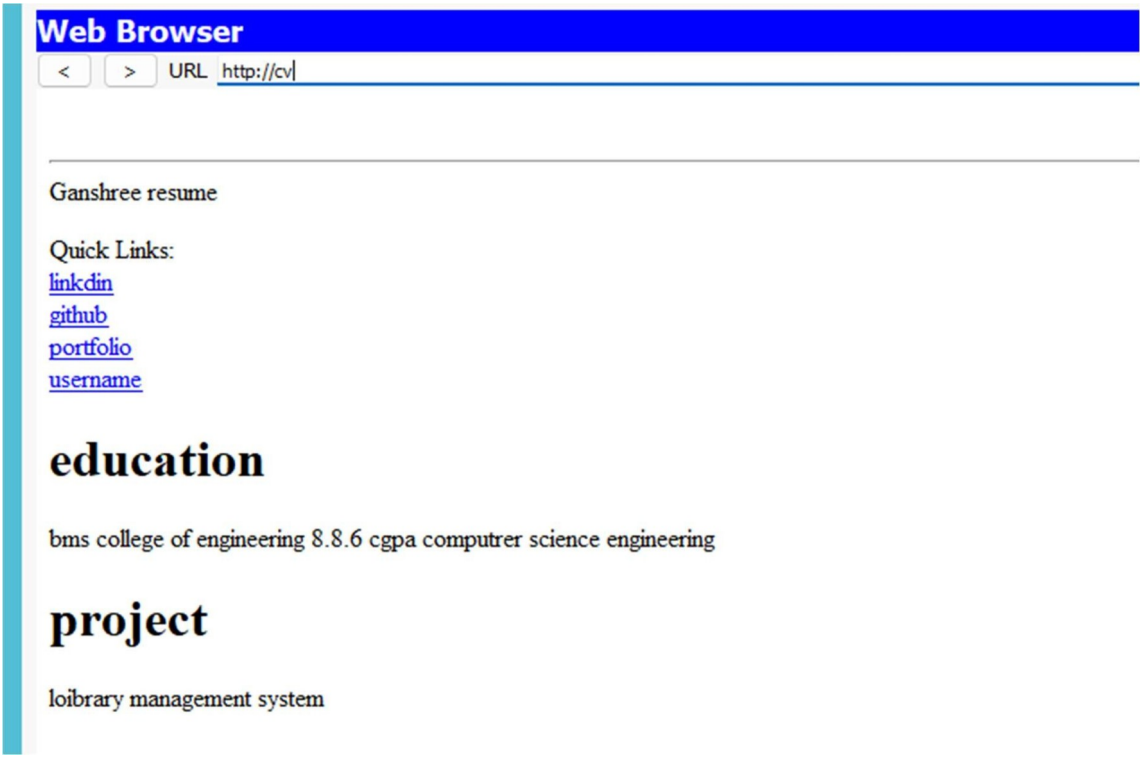
OBSERVATION:


a) Virtual LAN.
a) Configure Web server, DNS with a LAN
Aim:- configuring DNS server.
Topology:-
Switch - PT
Switch 0
PC - PT
PC 0
10.0.0.2
Server - PT
Server 0
10.0.0.3
Procedure:-
* Select the PC, switch and a server connect it using cable.
* Assign IP address to pc and server.
* In server go to services DNS turn it on write the name and address, and press add.
* In services select HTTP chers on edit the file.
* Then in PC got Desktop, in web browser write the domain name to get result.
Observation:-
* The domain name system works each IP address with a domain name.
* When entered the domain name the contents of the specific IP address comes from server.

TOPOLOGY:

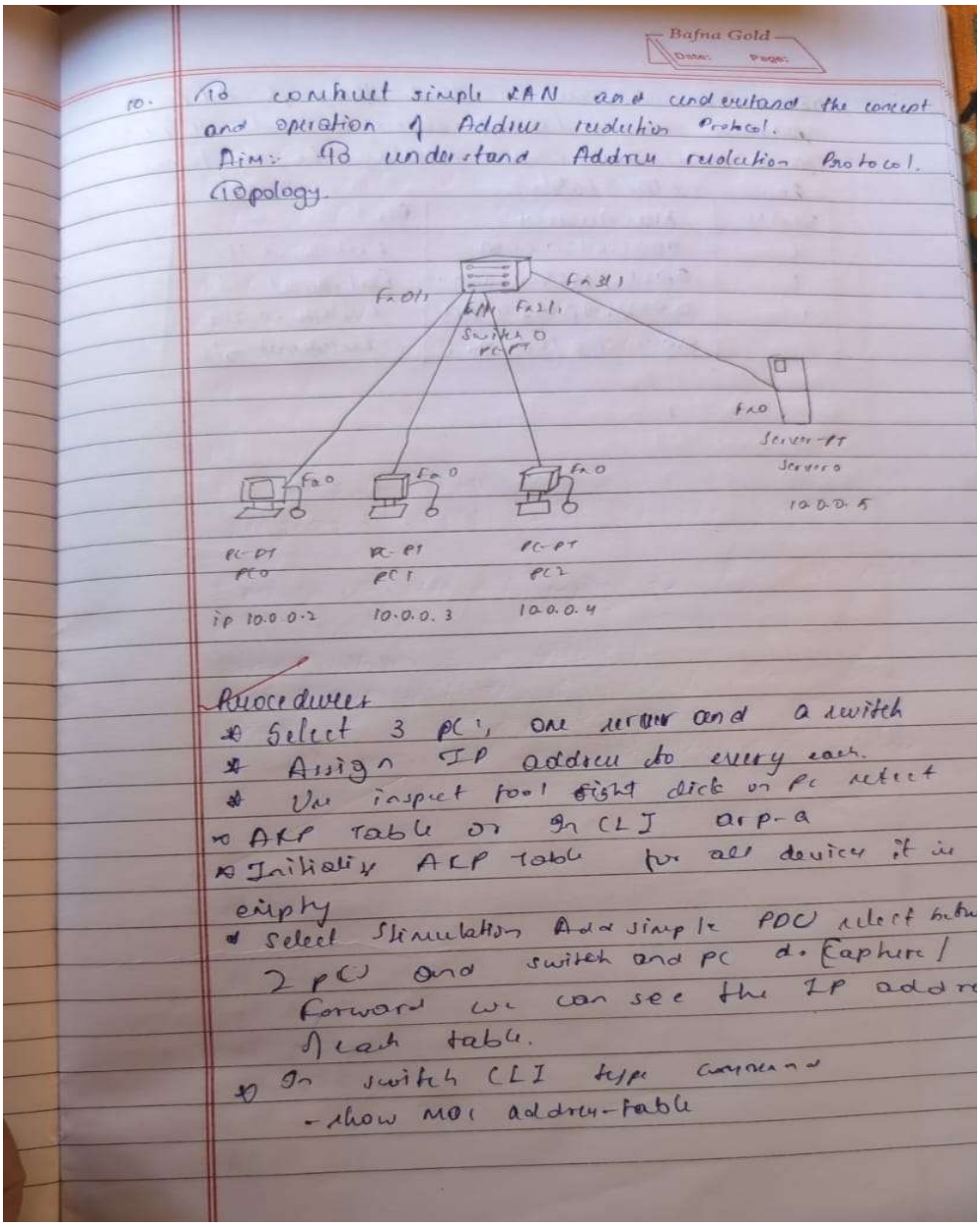

Switch-PT
Switch0
PC-PT
PC0
Server-PT
Server0

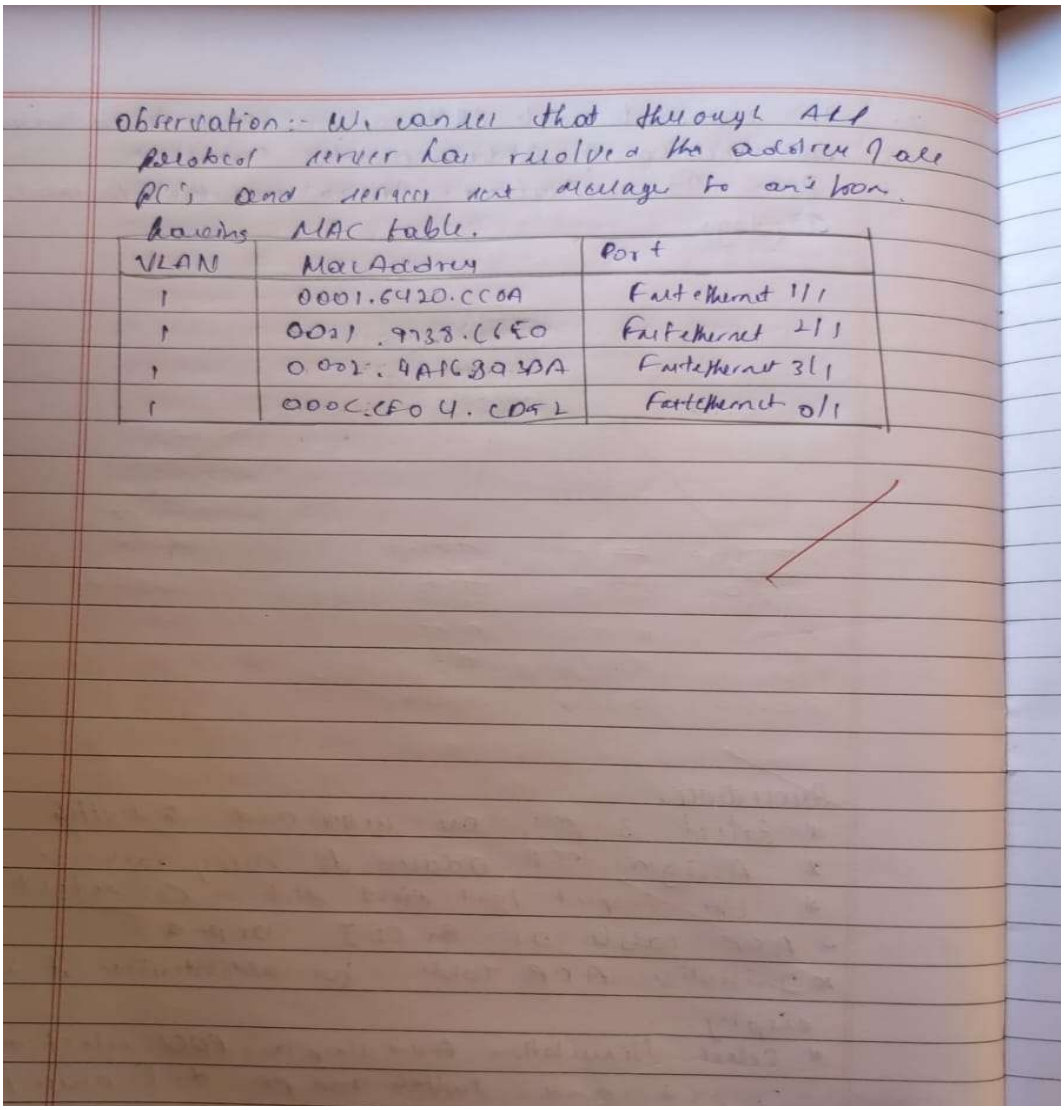
OUTPUT:


Web Browser
< > URL http://cv
Ganshree resume
Quick Links:
linkdin
github
portfolio
username
education
bms college of engineering 8.8.6 cgpa computrer science engineering
project
loibrary management system

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

OBSERVATION:

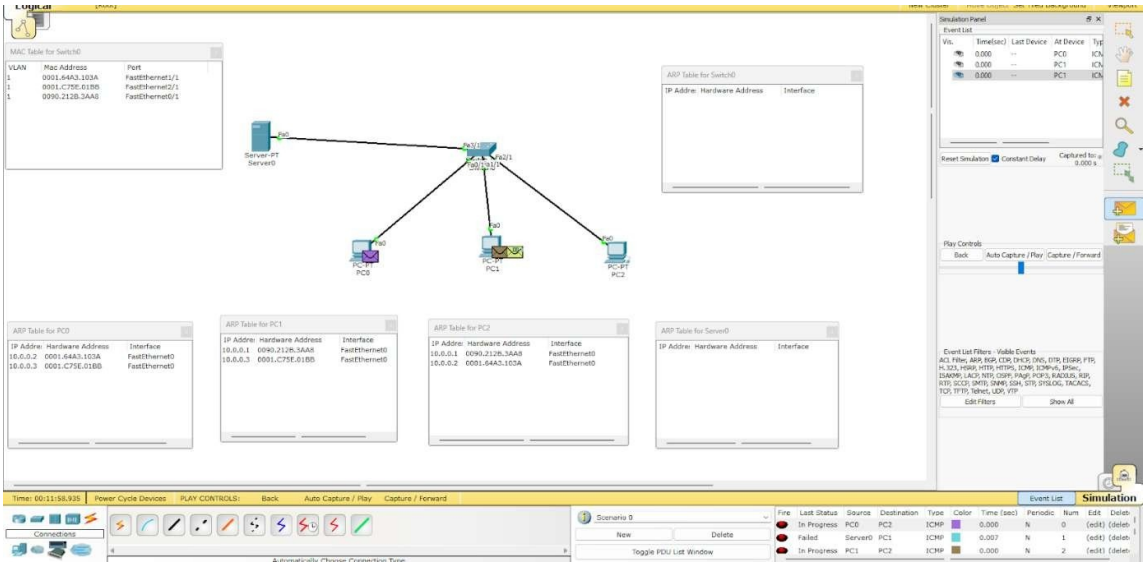




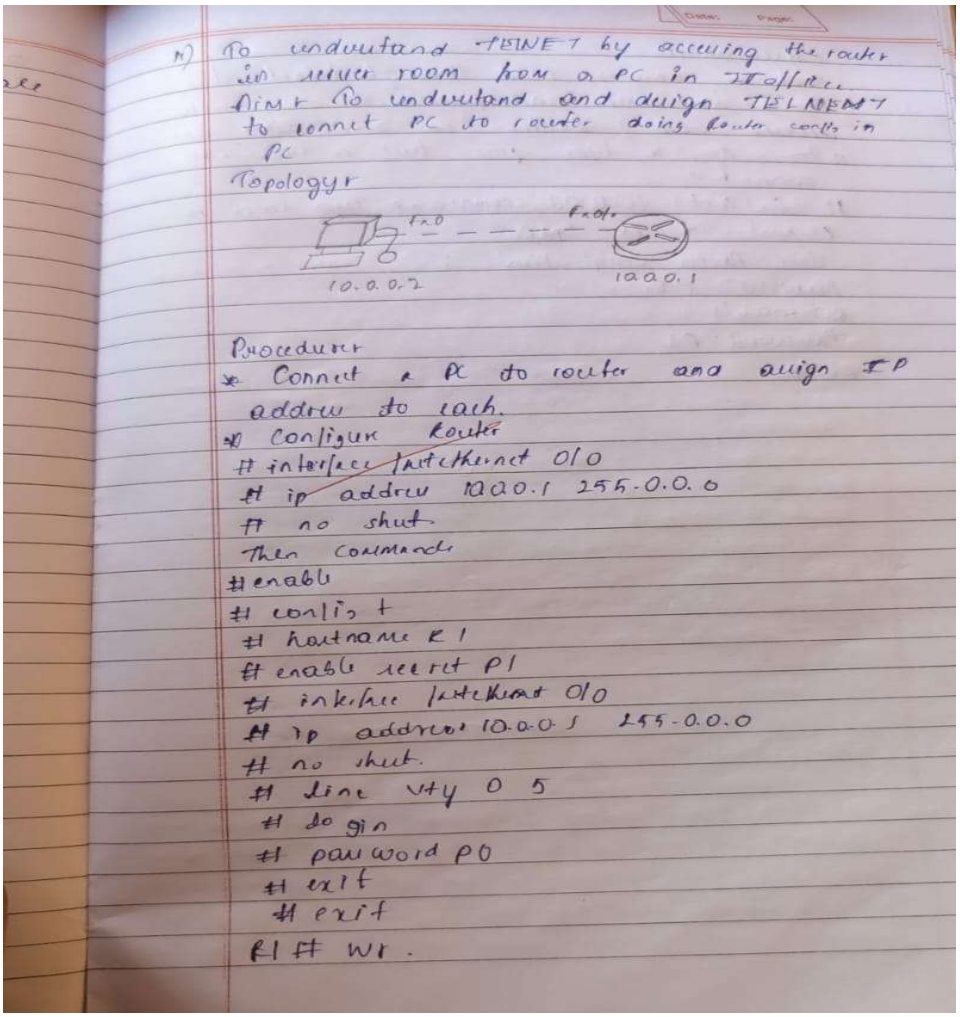
Observation:- We can see that through ARP protocol server has resolved the address of all PC's and server sent message to and from. having MAC table.

VLAN	MacAddress	Port
1	0001.6420.CC0A	FastEthernet 1/1
1	0021.9738.CCE0	FastEthernet 2/1
1	0002.4AC8.9DA	FastEthernet 3/1
1	000C.CF04.CD5L	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:

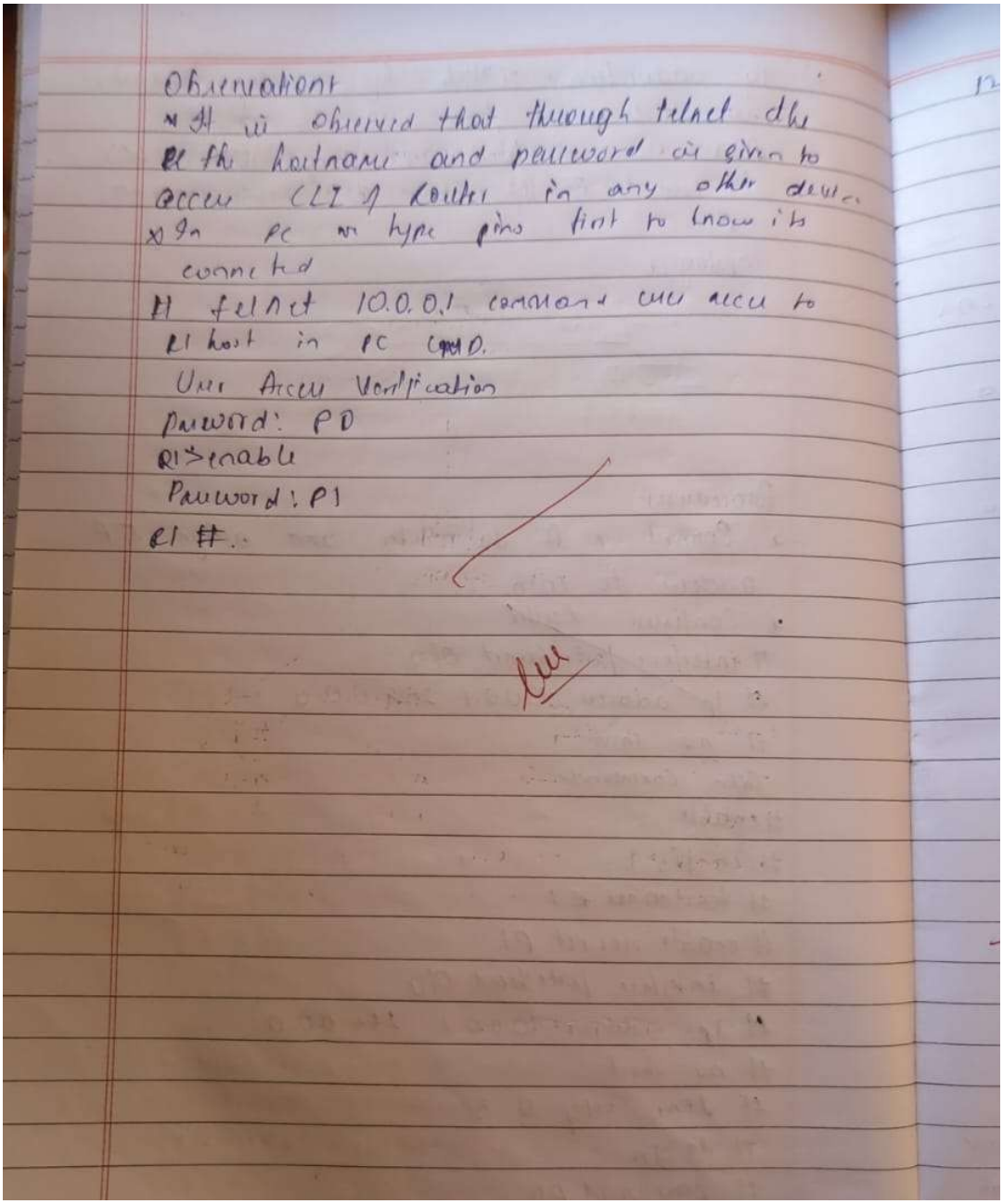


11) To understand TELNET by accessing the router in server room from a PC in IT office
Aim: To understand and design TELNET to connect PC to router doing Router configs in PC

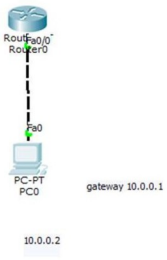
Topology:

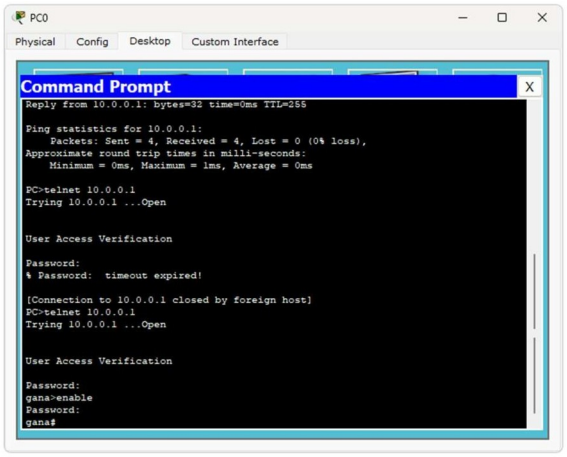
Procedure:

* Connect a PC to router and assign IP address to each.
# Configure Router
# interface fastethernet 0/0
# ip address 1000.1 255.0.0.0
# no shut
Then Commands
#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 0 5
login
password P0
exit
exit
R1 # wr.


Observation
* It is observed that through telnet the hostname and password is given to access CLI of router in any other device.
* In PC we type ping first to know it is connected
* telnet 10.0.0.1 command use access to router in PC cmd.
User Access Verification
Password: PD
R1>enable
Password: P1
R1#.

TOPOLOGY and OUPUT:





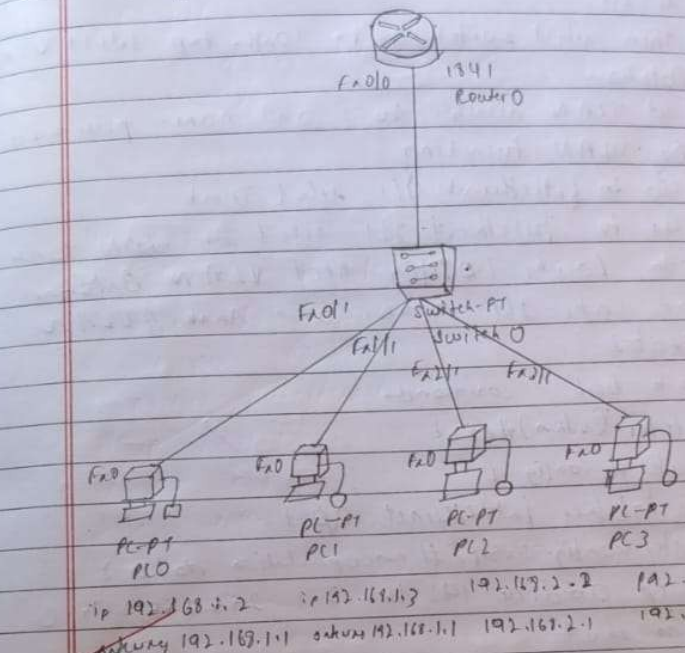
PROGRAM11: To construct a VLAN and make the PC's communicate among a VLAN
OBSERVATION:

12

Creating a new VLAN and make the PC's communicate among a VLAN.

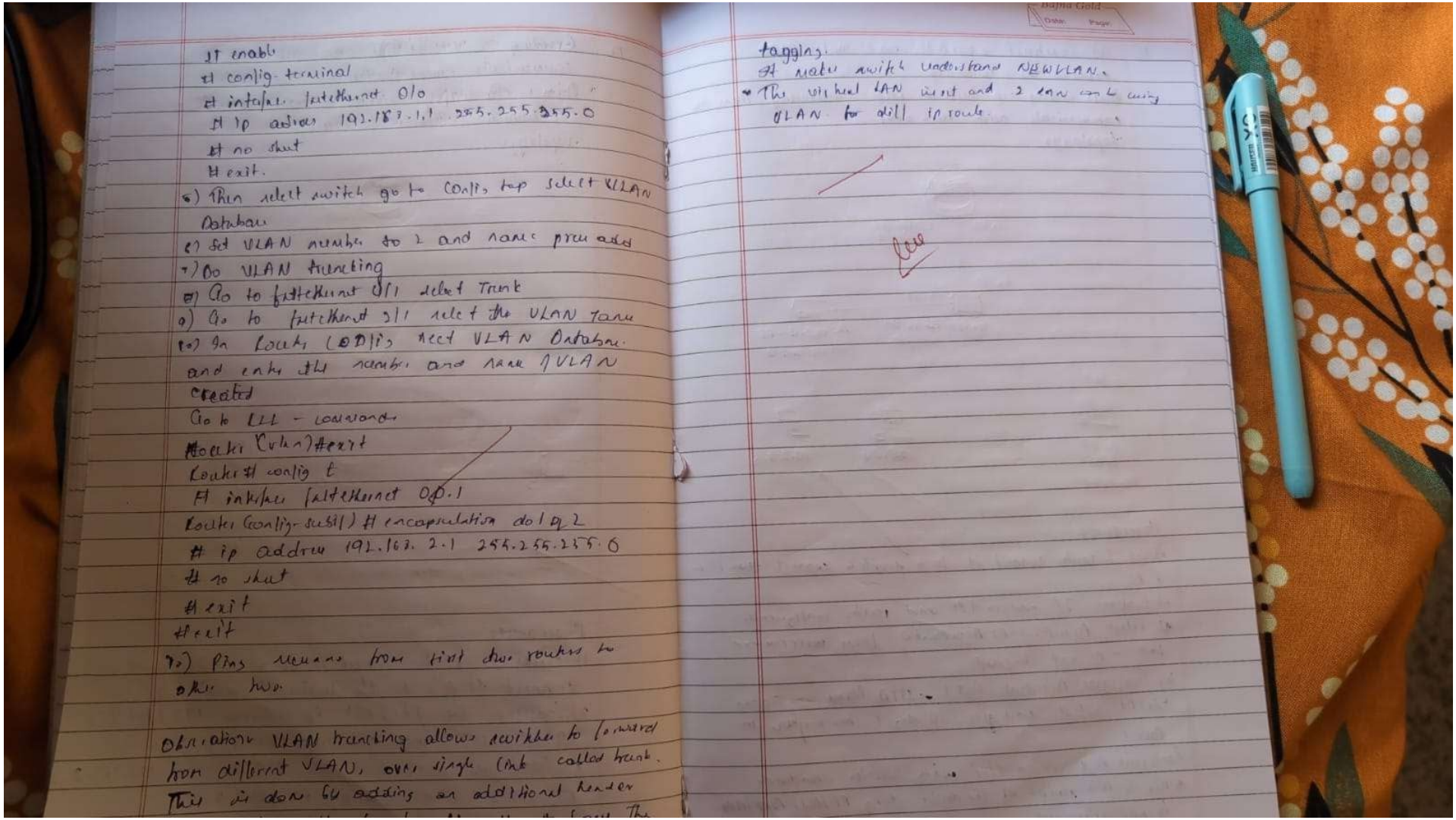
Aim: To configure VLAN and make PC's communicate with each other.

Topology:



Procedure:

1) Take a router connect it to a switch and connect 4 PC's to the switch-PT switch 0
2) Configure the 1st 2 PC's ip address 192.168.1.2 192.168.1.3 and gateway 192.168.1.1
3) Configure next two PC's IP address 192.168.2.3 and gateway 192.168.2.2
4) Go to router configure on first 2 PC's the router commands

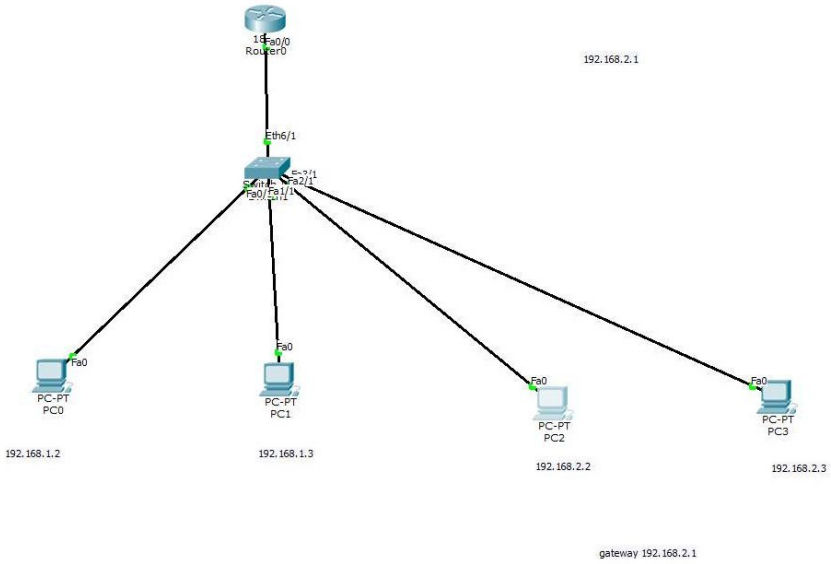


#1 enable
#1 config terminal
#1 interface fastethernet 0/0
#1 ip address 192.168.1.1 255.255.255.0
#1 no shut
#1 exit.
5) Then select switch go to config tab select VLAN Database
6) Set VLAN number to 2 and name previously added
7) Do VLAN trunking
8) Go to fastethernet 0/1 select Trunk
9) Go to fastethernet 2/1 select the VLAN name
10) In Router (CLI) is next VLAN Database and enter the number and name of VLAN created
Go to CLI - command
Router(vlan)#exit
Router#1 config t
#1 interface fastethernet 0/0.1
Router(config-subif)#1 encapsulation dot1q 2
#1 ip address 192.168.2.1 255.255.255.0
#1 no shut
#1 exit
#1 exit
12) Ping command from first two routers to other two.

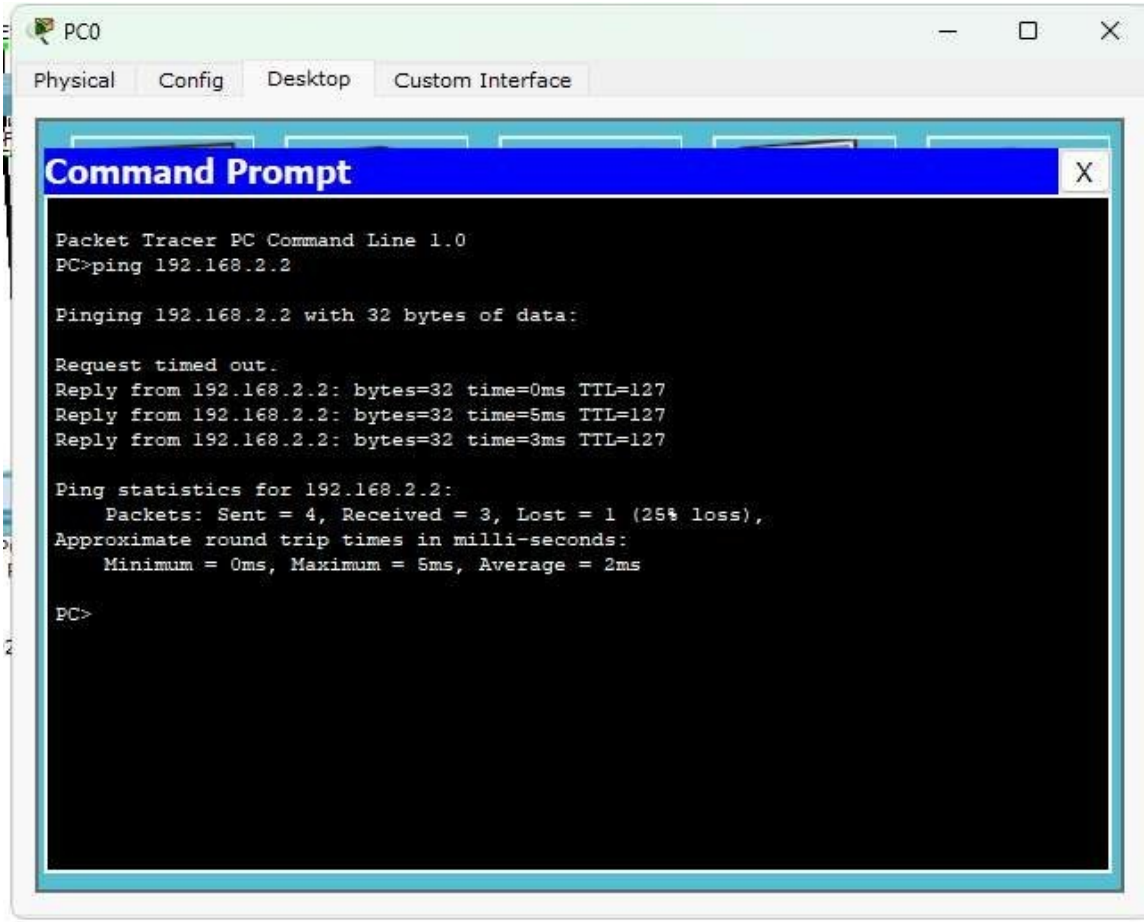
Observation VLAN trunking allows switches to forward from different VLAN, over single link called trunk. This is done by adding an additional header

tagging.
#1 make switch understand NEW VLAN.
* The virtual LAN must and 2 LAN on 1 using VLAN for dill in route.

TOPOLOGY:



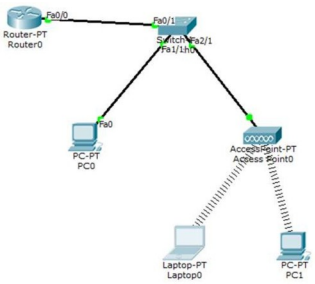
OUTPUT:

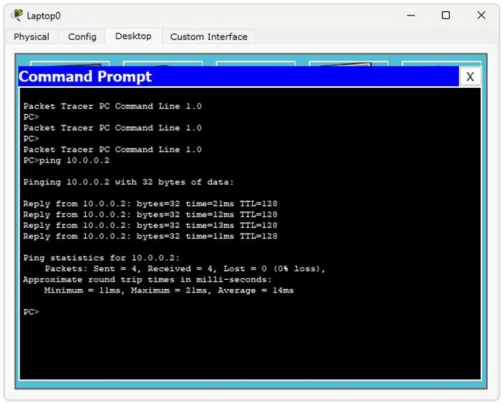


PROGRAM12: To construct a WLAN and make the nodes communicate wirelessly To construct a WLAN and make the nodes communicate wirelessly

OBSERVATION:

TOPOLOGY AND OUPUT:

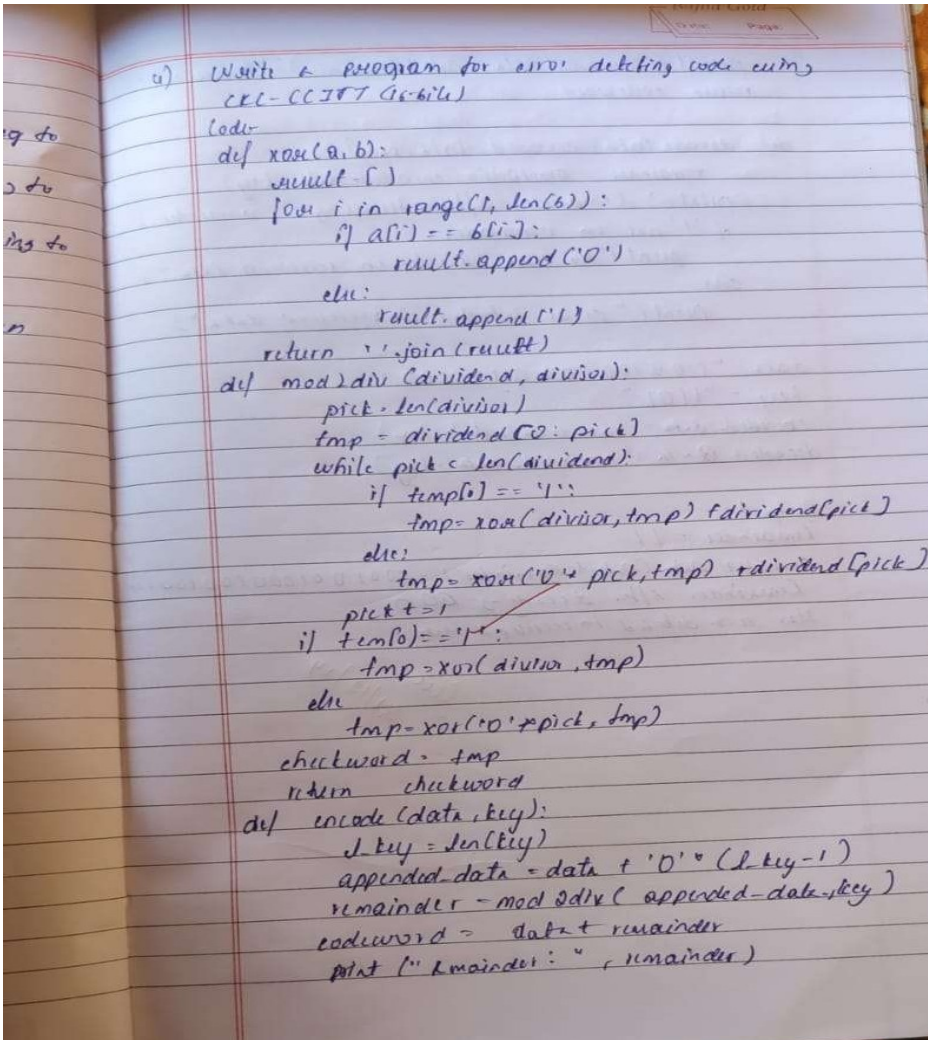

Fa0/0
Fa0/1
Router-PT
Router0
Switch-PT
Fa1/1
Fa2/1
Fa0
PC-PT
PC0
Access Point-PT
Access Point0
Laptop-PT
Laptop0
PC-PT
PC1


Laptop0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>
Packet Tracer PC Command Line 1.0
PC>
Packet Tracer PC Command Line 1.0
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=21ms TTL=128
Reply from 10.0.0.2: bytes=32 time=12ms TTL=128
Reply from 10.0.0.2: bytes=32 time=13ms TTL=128
Reply from 10.0.0.2: bytes=32 time=11ms TTL=128
Ping statistics for 10.0.0.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 11ms, Maximum = 21ms, Average = 14ms
PC>

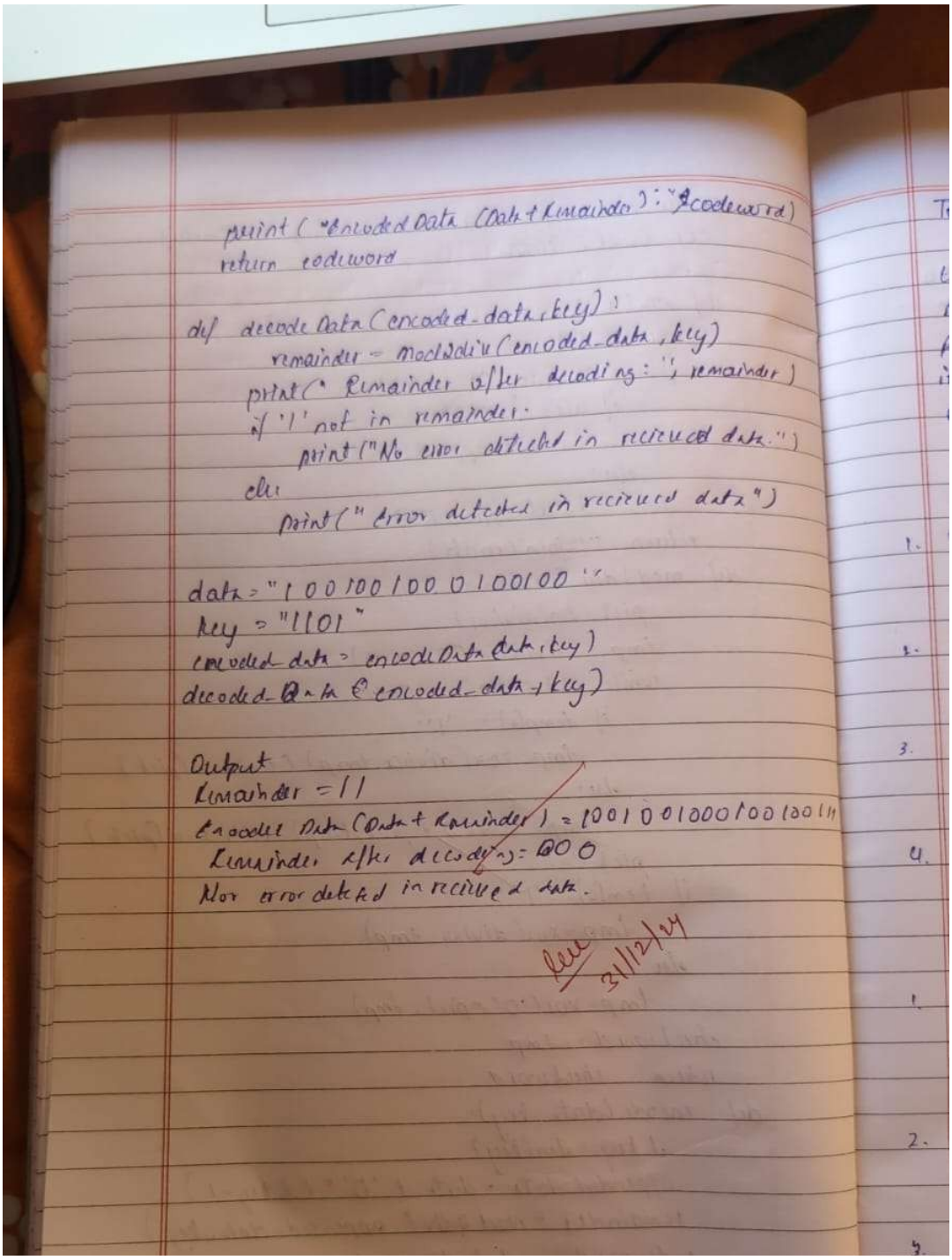
CYCLE-2

PROGRAM1: Write a program for error detecting code using CRC-CCITT (16-bits)

OBSERVATION:



```
a) Write a program for error detecting code using
   CRC-CCITT (16-bits)
Code
def xor(a,b):
    result=[]
    for i in range(1,len(b)):
        if a[i]==b[i]:
            result.append('0')
        else:
            result.append('1')
    return ''.join(result)
def mod2div(dividend,divisor):
    pick=len(divisor)
    tmp=dividend[0:pick]
    while pick<len(dividend):
        if tmp[0]=='1':
            tmp=xor(divisor,tmp)+dividend[pick]
        else:
            tmp=xor('0'*pick,tmp)+dividend[pick]
        pick+=1
    if tmp[0]=='1':
        tmp=xor(divisor,tmp)
    else
        tmp=xor('0'*pick,tmp)
    checkword=tmp
    return checkword
def encode(data,key):
    l_key=len(key)
    appended_data=data+'0'*(l_key-1)
    remainder=mod2div(appended_data,key)
    codeword=data+remainder
    print("Remainder:",remainder)
```



```
print("Encoded Data (Data + Remainder): ", codeword)
return codeword

def decodeData(encoded_data, key):
    remainder = modulo2div(encoded_data, key)
    print("Remainder after decoding: ", remainder)
    if '1' not in remainder:
        print("No error detected in received data.")
    else:
        print("Error detected in received data")

data = "100100100100100"
key = "1101"
encoded_data = encodeData(data, key)
decodeData(encoded_data, key)
```

Output

Remainder = 11

Encoded Data (Data + Remainder) = 10010010001001011

Remainder after decoding = 000

Non error detected in received data.

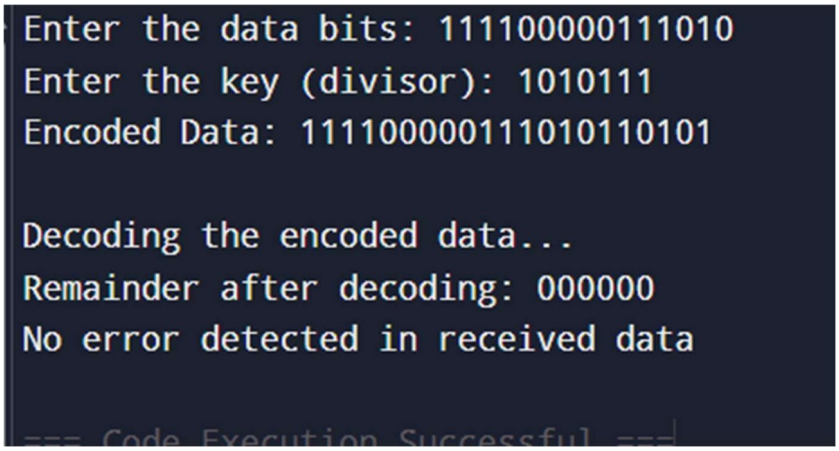
31/12/24

CODE:

```
def xor(a, b):
    result = []
    for i in range(1, len(b)):
        if a[i] == b[i]:
            result.append('0')
        else:
            result.append('1')
    return ''.join(result)


def mod2div(dividend, divisor):
    pick = len(divisor)
    tmp = dividend[0:pick]

    while pick < len(dividend):
        if tmp[0] == '1':
            tmp = xor(divisor, tmp) + dividend[pick]
        else:
            tmp = xor('0' * pick, tmp) + dividend[pick]
        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
```

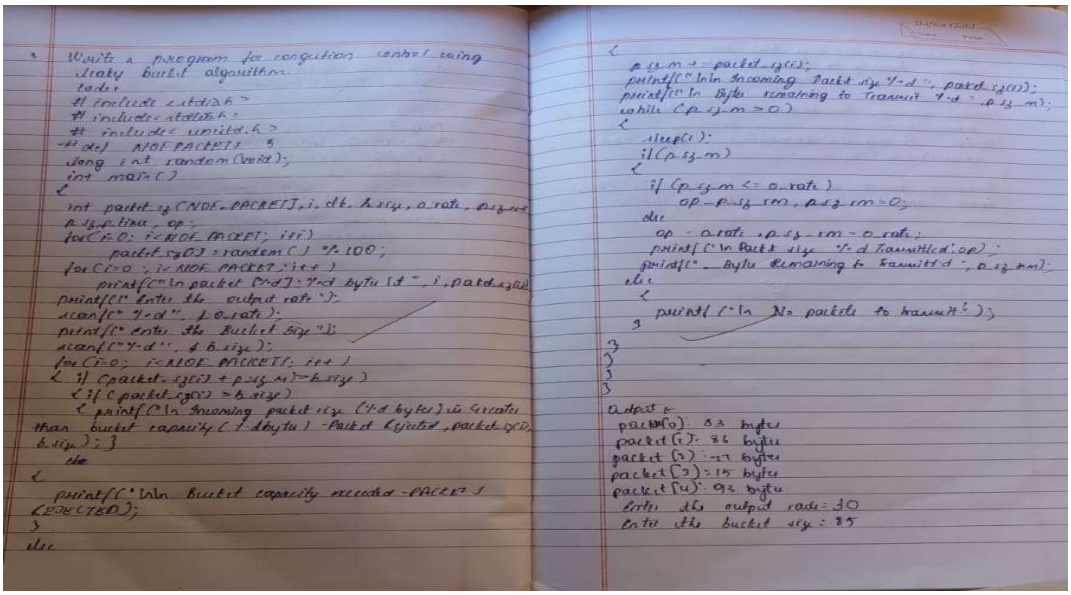
```
if __name___== "__main__":
    data = input("Enter the  data bits:  ")
    key = input("Enter the key (divisor): ")

    # Encoding
    encoded_data = encode(data, key)

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

OUTPUT:



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

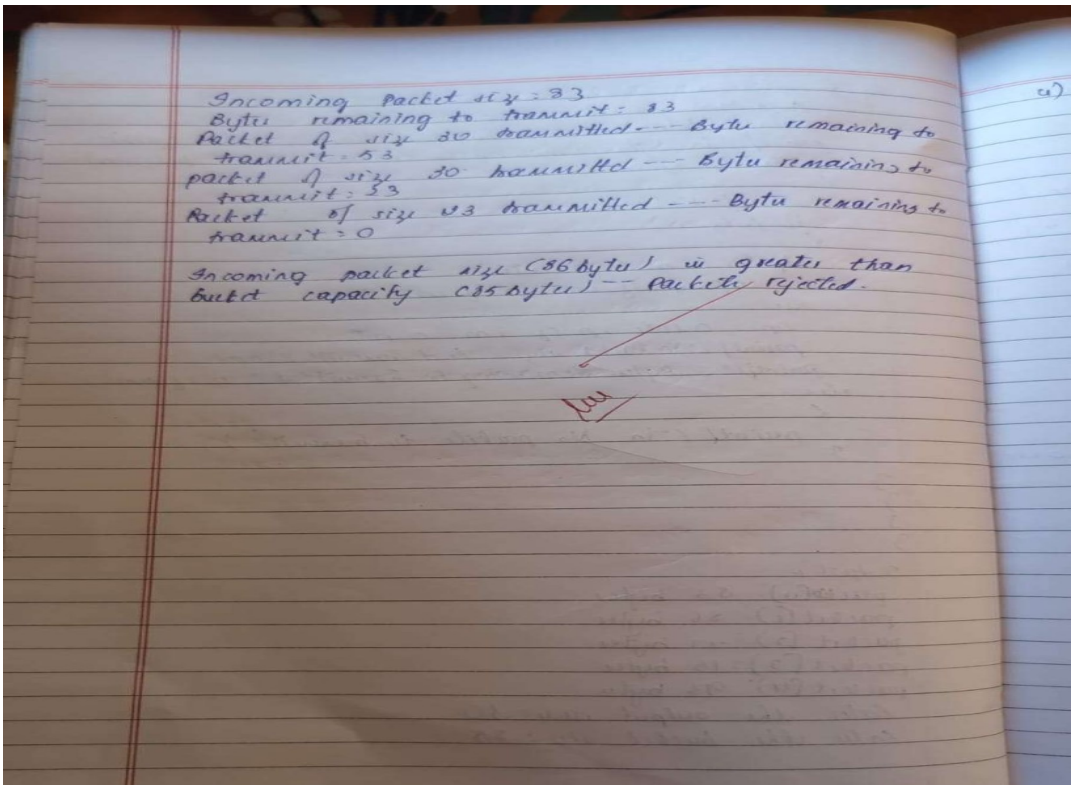
OBSERVATION:



```
3. Write a program for congestion control using
   leaky bucket algorithm.
   code:
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#define NOF_PACKETS 5
long int random(void);
int main()
{
 int packet_sz[NOF_PACKETS],i,clk,b_size,o_rate,p_sz_rm,
 p_sz,p_time,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 the output rate:");
 scanf("%d",&o_rate);
 printf("Enter the 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)
   printf("\n Incoming packet size (%d bytes) is Greater
than bucket capacity(%d bytes) -Packet %d rejected",packet_sz[i],
b_size);}
  else
 {
  printf("\n Bucket capacity exceeded -PACKET %
d REJECTED",i);
 }
else
{
 p_sz_rm+=packet_sz[i];
 printf("\n\n Incoming Packet size %d",packet_sz[i]);
 printf("\n Bytes remaining to Transmit %d",p_sz_rm);
 while(p_sz_rm>0)
 {
  sleep(3);
  if(p_sz_rm)
  {
   if(p_sz_rm<=o_rate)
     op=p_sz_rm,p_sz_rm=0;
   else
    op=o_rate,p_sz_rm=o_rate;
    printf("\n Packet size %d transmitted",op);
    printf("\n Bytes Remaining to Transmitted",p_sz_rm);
  }else
   {
    printf("\n No 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



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:

CODE:

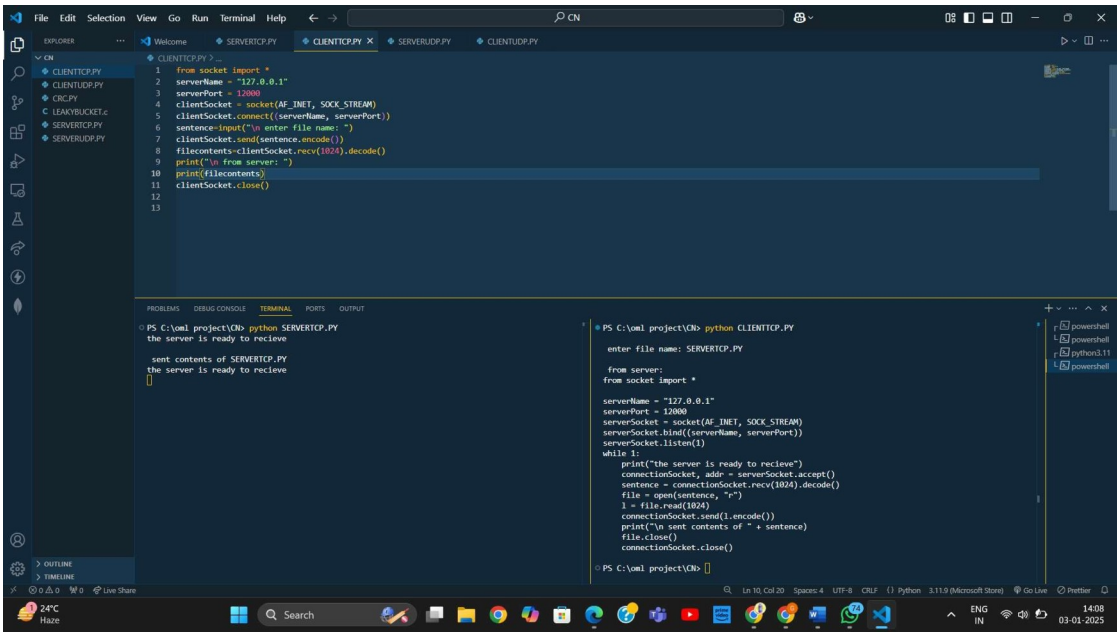
SERVERTCP.PY:

```
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:

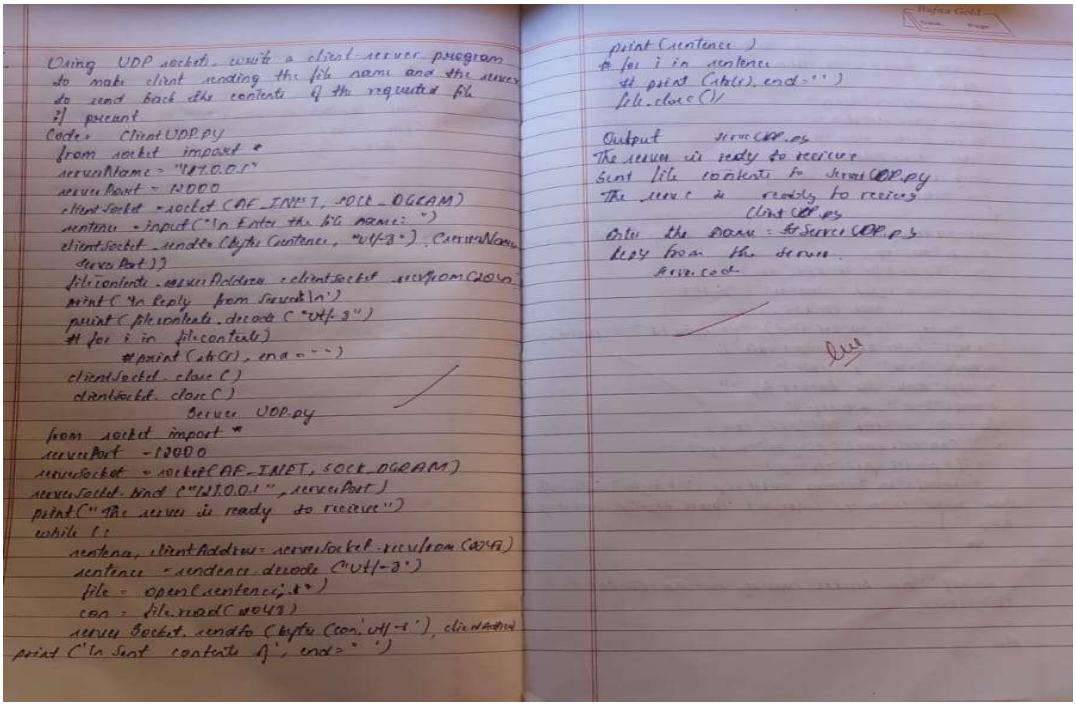
```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence=input("\n enter file name: ")
clientSocket.send(sentence.encode())
filecontents=clientSocket.recv(1024).decode()
print("\n from server: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

OUTPUT:


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:



CODE:

SERVERUDP.PY

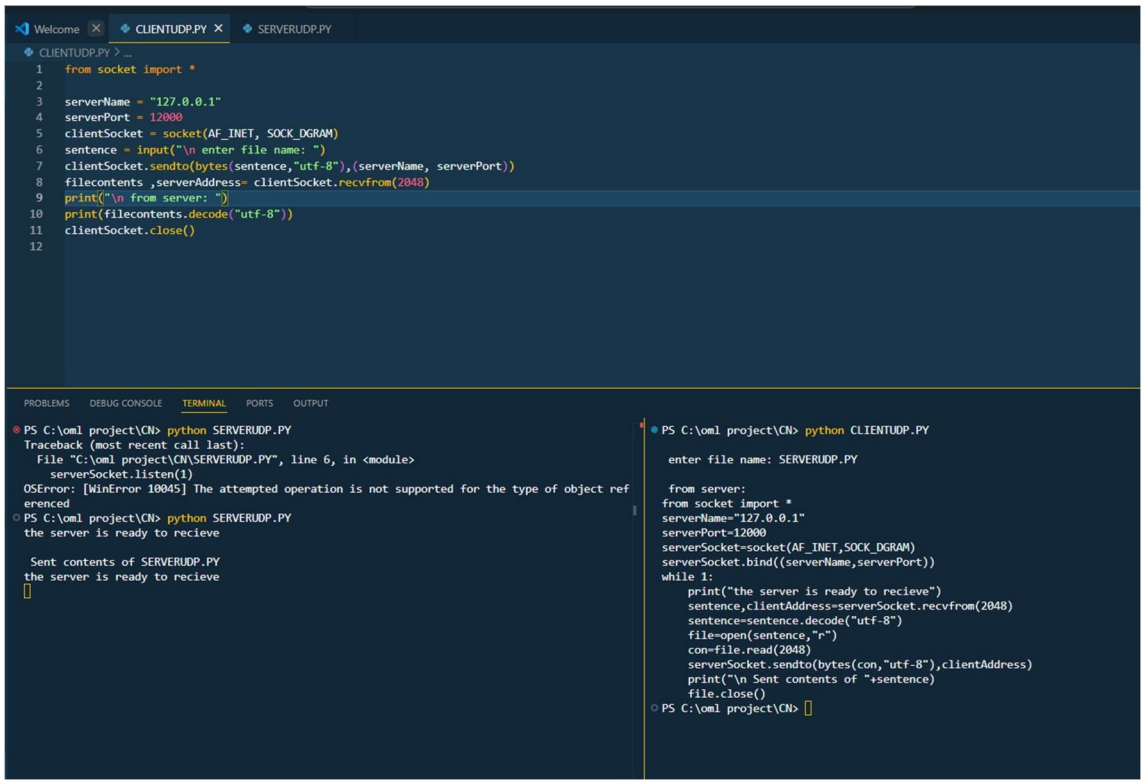
```
from socket import *
serverName="127.0.0.1"
```

```
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()
```

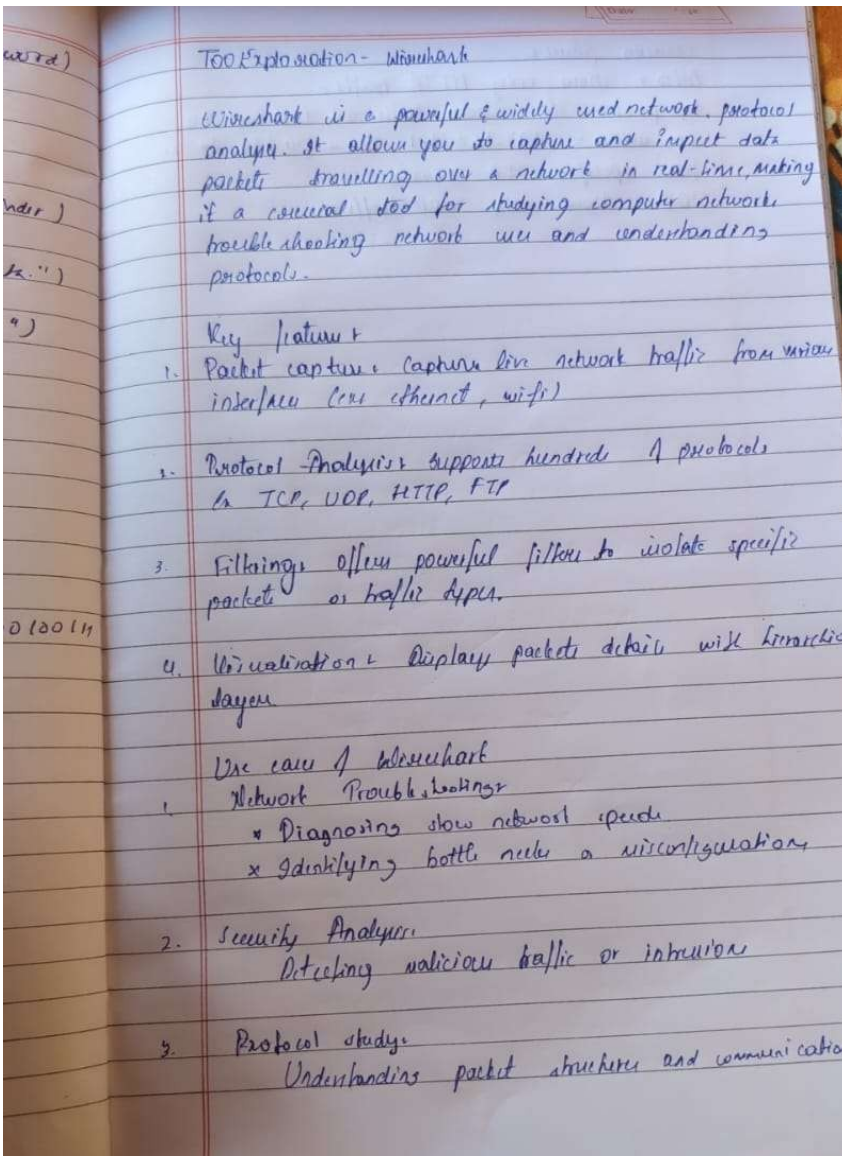
CLIENTUDP.PY:

```
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 from server: ")
print(filecontents.decode("utf-8"))
clientSocket.close()
```

OUTPUT:


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 ref
erenced
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:



Tool Exploration - Wireshark

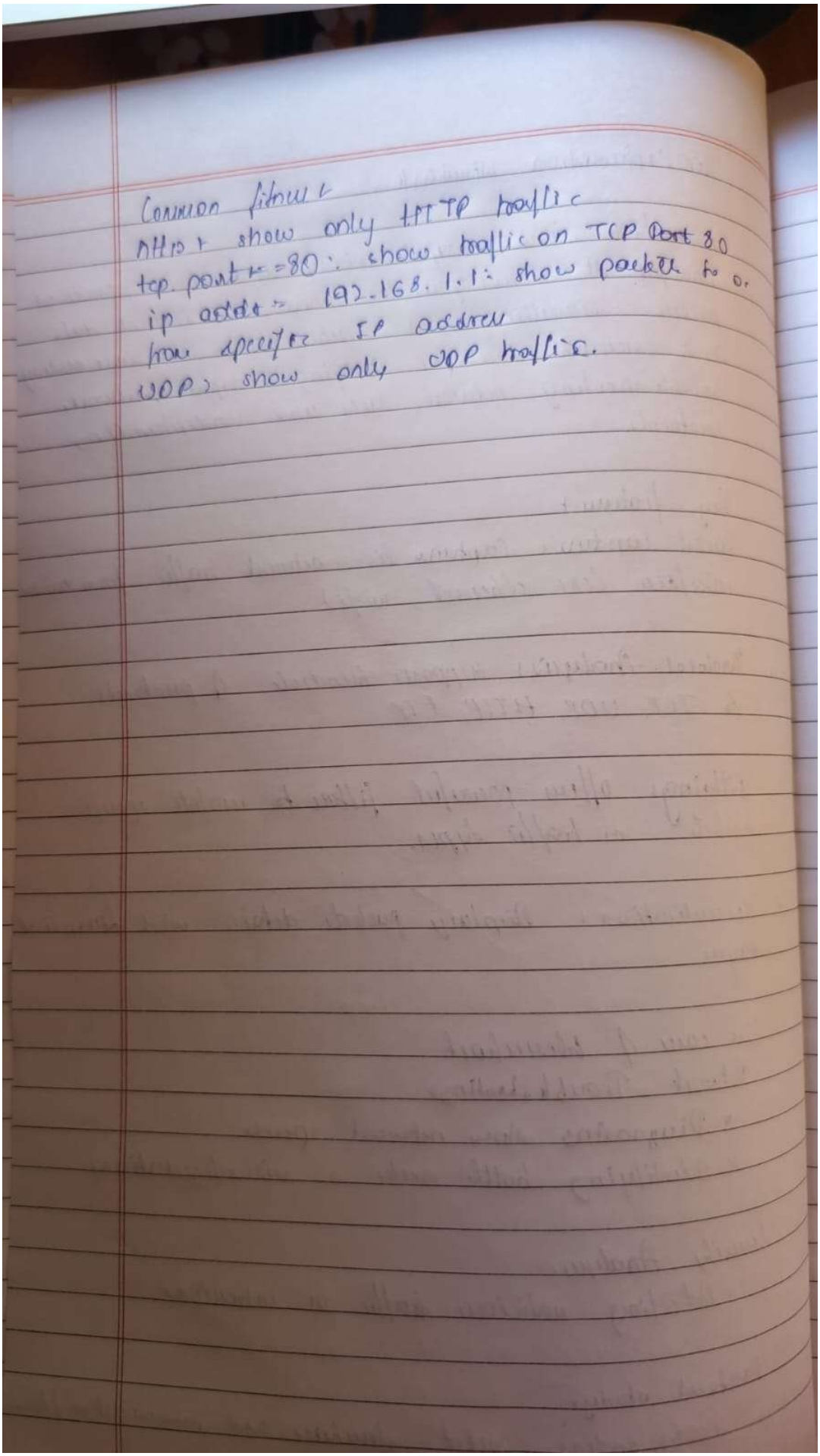
Wireshark is a powerful & widely used network protocol analyzer. It allows you to capture and inspect data packets travelling over a network in real-time, making it a essential tool for studying computer networks, troubleshooting network issues and understanding protocols.

Key features:

1. Packet capture: Capture live network traffic from various interfaces (like ethernet, wifi)
2. Protocol Analysis: Supports hundreds of protocols (e.g. TCP, UDP, HTTP, FTP)
3. Filtering: Offers powerful filters to isolate specific packets or traffic types.
4. Visualization: Displays packets details with hierarchical layers

Use cases of Wireshark

1. Network Troubleshooting
   - Diagnosing slow network speeds
   - Identifying bottle necks & misconfigurations
2. Security Analysis
   Detecting malicious traffic or intrusions
3. Protocol study:
   Understanding packet structures and communicatio


Common filters:
http: show only HTTP traffic
tcp.port == 80: show traffic on TCP Port 80
ip.addr == 192.168.1.1: show packets to or
from specific IP address
udp: show only UDP traffic.