VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

Darshna Mimrot(1BM22CS081)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
Academic Year 2024-25(Odd)

B.M.S. College of Engineering

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled " Computer Network(23CS5PCCON)" carried out by **Darshna Mimrot(1BM22CS081),** who is a bonafide student of **B.M.S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements of the above-mentioned subject and the work prescribed for the said degree.

Sandhya A Kulkarni Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index

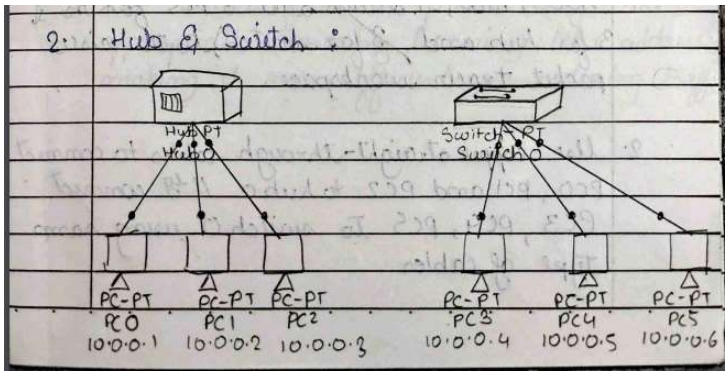
19	18-12-24	Tool exploration - wireshark	52-55

Github Link: https://github.com/Darshna27/cn-lab-

Program 1

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

Topology , Procedure and Observation:



→ Aim: To create a simple network consisting of 3 PCs connected to a central hub and another network with 3 PCs connected to a switch. This configuration will help observe the behaviour of data transmission using hub & switch devices.

→ Topology:

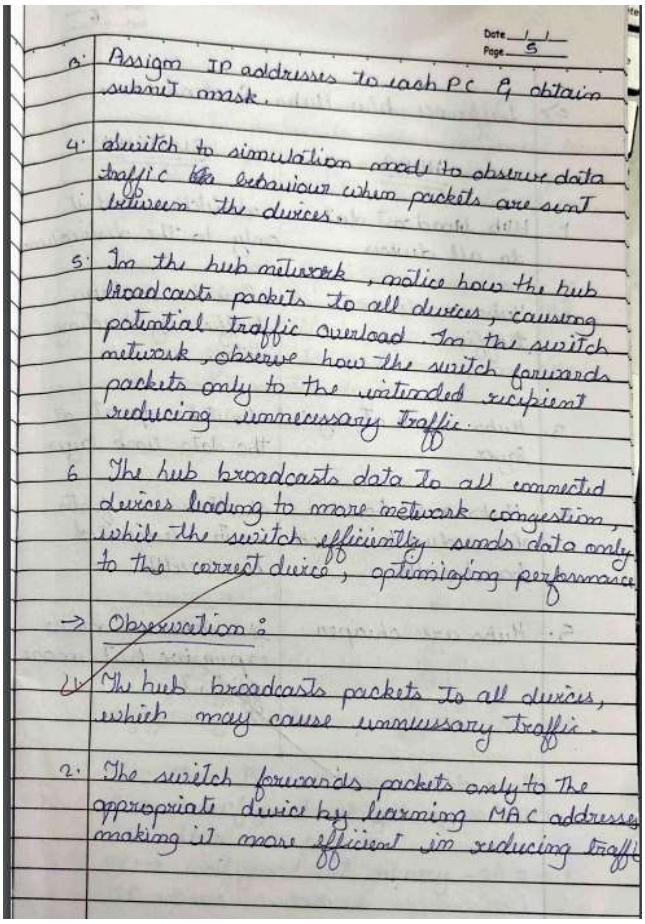
1. Hub network: Three PCs (PC0, PC1, PC2) are connected to a hub (HUB0) using straight-through Ethernet cables.
IP addresses: PC0=10.0.0.1, PC1=10.0.0.2, PC2=10.0.0.3.

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 PCs (as 3 for hub and 3 for switch) to the Cisco packet tracer workspace.

2. Use copper straight-through cables to connect PC0, PC1 and PC2 to hub0 and connect PC3, PC4, PC5 to switch0 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.

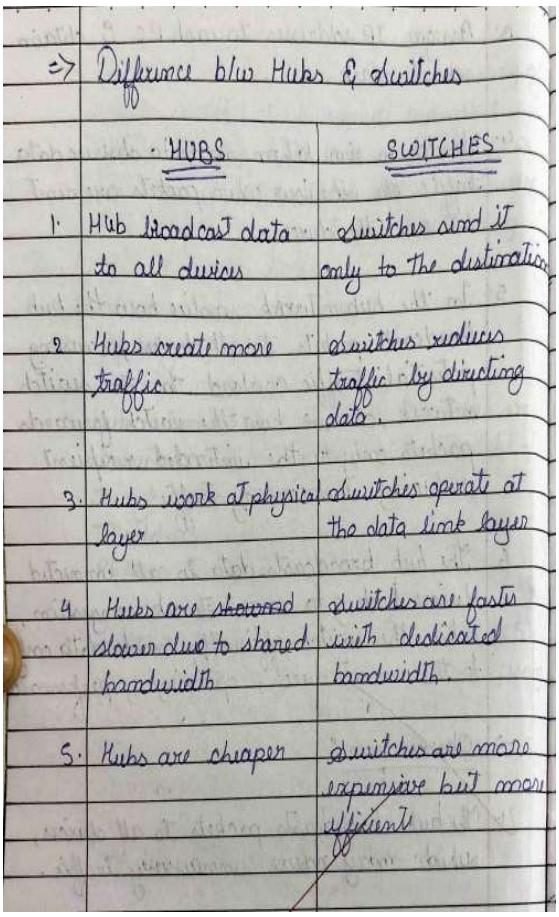
5. In the hub network, notice how the hub broadcasts packets to all devices, causing potential traffic overload. In the switch network, observe how the switch forwards packets only to the intended recipient reducing unnecessary traffic.

6. The hub broadcasts data to 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 may cause unnecessary traffic.

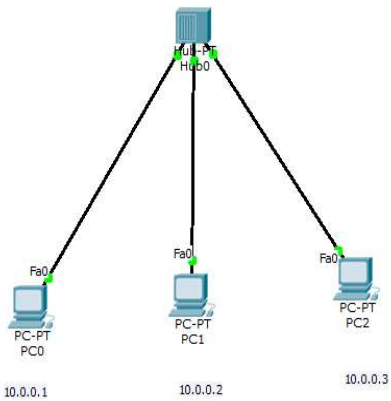
2. The switch forwards packets only to the appropriate device by learning MAC addresses making it more efficient in reducing traffic

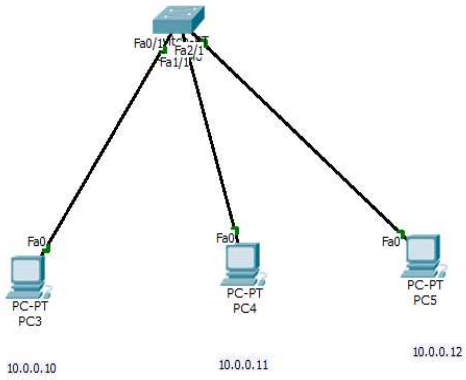


=> Difference b/w Hubs & Switches

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

Screen Shots:

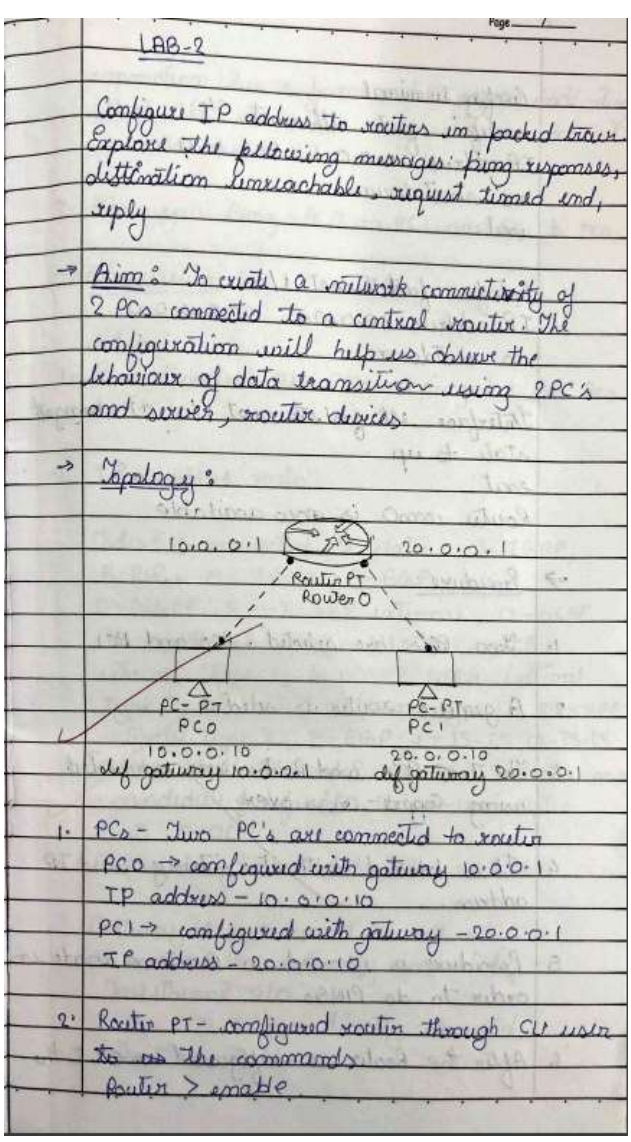




Program 2

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

Topology , Procedure and Observation:



LAB-2

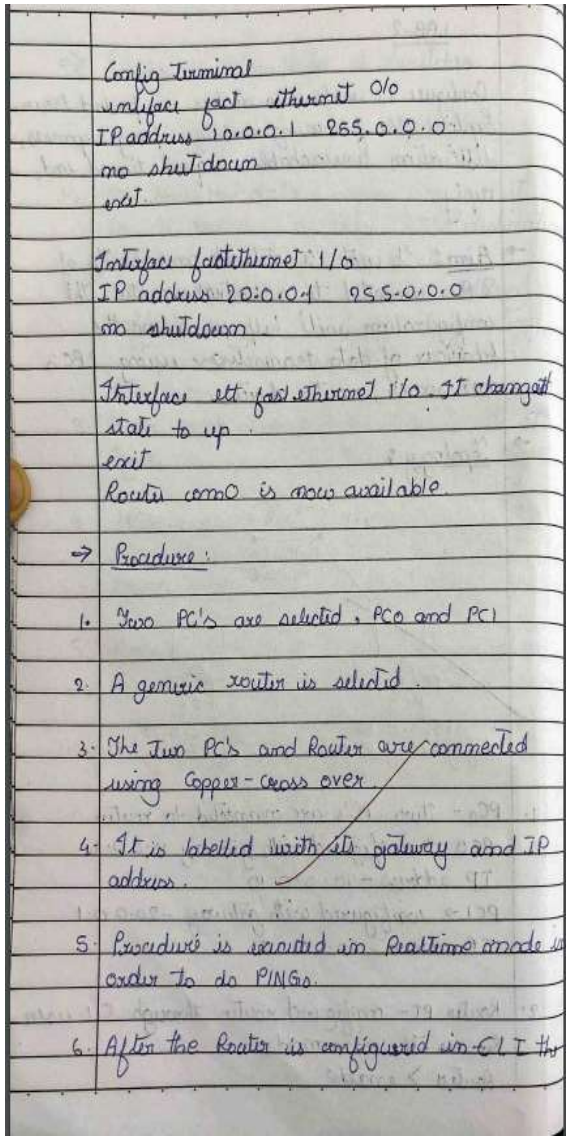
Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply

→ Aim: To create a network connectivity of 2 PCs connected to a central router. The configuration will help us observe the behaviour of data transition using 2 PC's and server, router devices

→ Topology:

1. PCs - Two PC's are connected to router
PC0 → configured with gateway 10.0.0.1
IP address - 10.0.0.10
PC1 → configured with gateway - 20.0.0.1
IP address - 20.0.0.10

2. Router PT - configured router through CLI user to use the commands
Router > enable



Config Terminal
unifaci fast ethernet 0/0
IP address 10.0.0.1 255.0.0.0
no shut down
exit

Interface fastethernet 1/0
IP address 20.0.0.1 255.0.0.0
no shutdown

Interface at fastethernet 1/0. It changed state to up
exit
Router com0 is now available

→ Procedure:

1. Two PC's are selected, PC0 and PC1
2. A generic router is selected.
3. The two PC's and Router are connected using Copper-Cross over
4. It is labelled with its gateway and IP address.
5. Procedure is executed in Realtime mode in order to do PINGs.
6. After the Router is configured in CLI the

connection turns from red to green indicating that the router is configured and connection is stabalised.

7. Then you Ping PC0 to PC1 and PC1 to PC0.

→ Observations :

Setup :

1. Ping Results : The PC's are pinged from PC0 to PC1.

2. IP ~~results~~ & routs :

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 NSSA external type 2, E1 - OSPF external type 1, E2 - 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.

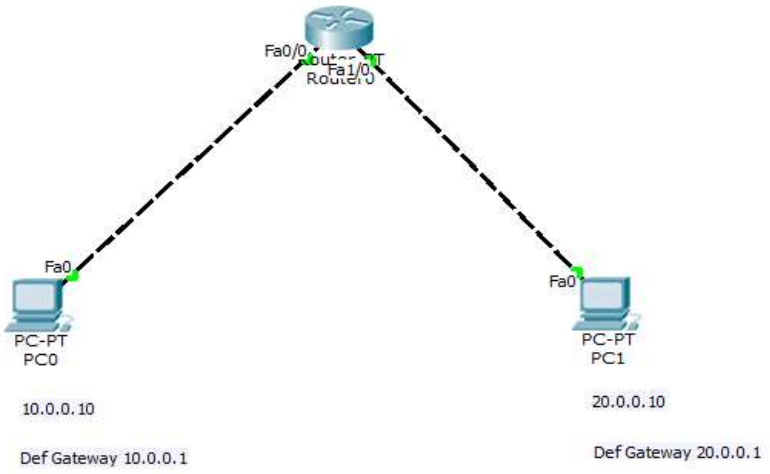
Date / /
Page 10

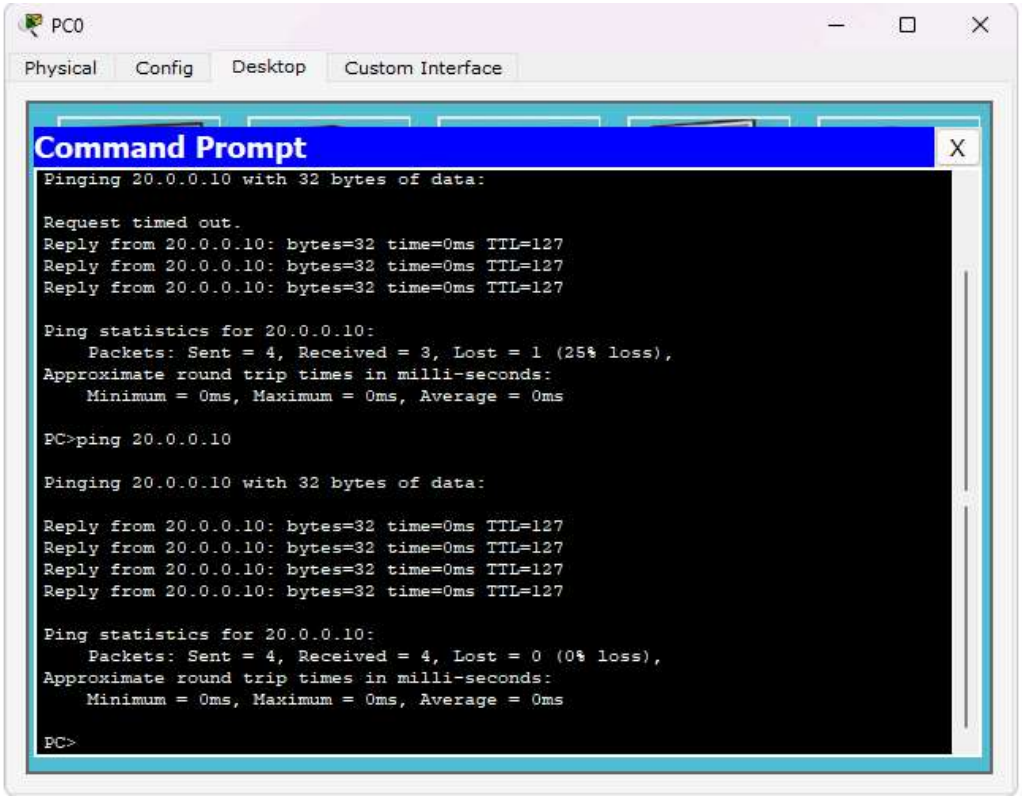
3. If the configurations and cabling are correct, you will receive successful ping b/w two PC's

4. If there is no connection, troubleshoot by verifying : correct IP addressing, cabling type, both router interfaces are up & running.

5. Then you can view the IP route data via CLI.

Screen Shots:

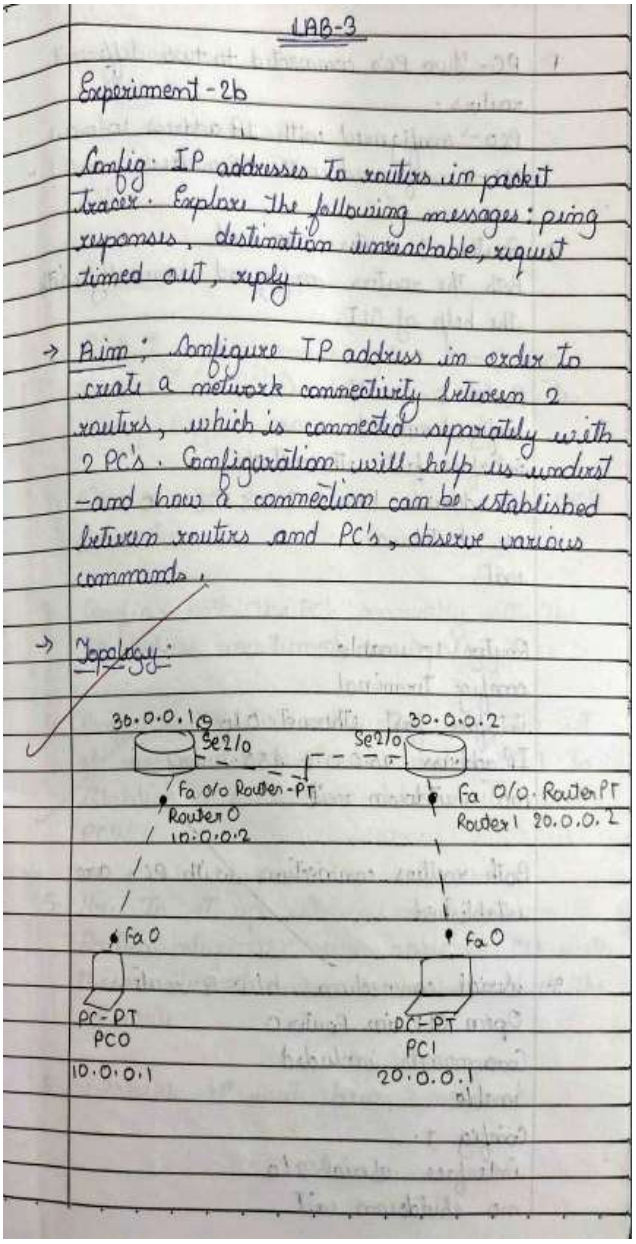




Program 3

Aim: Configure default route, static route to the Router(Part 1).

Topology , Procedure and Observation:



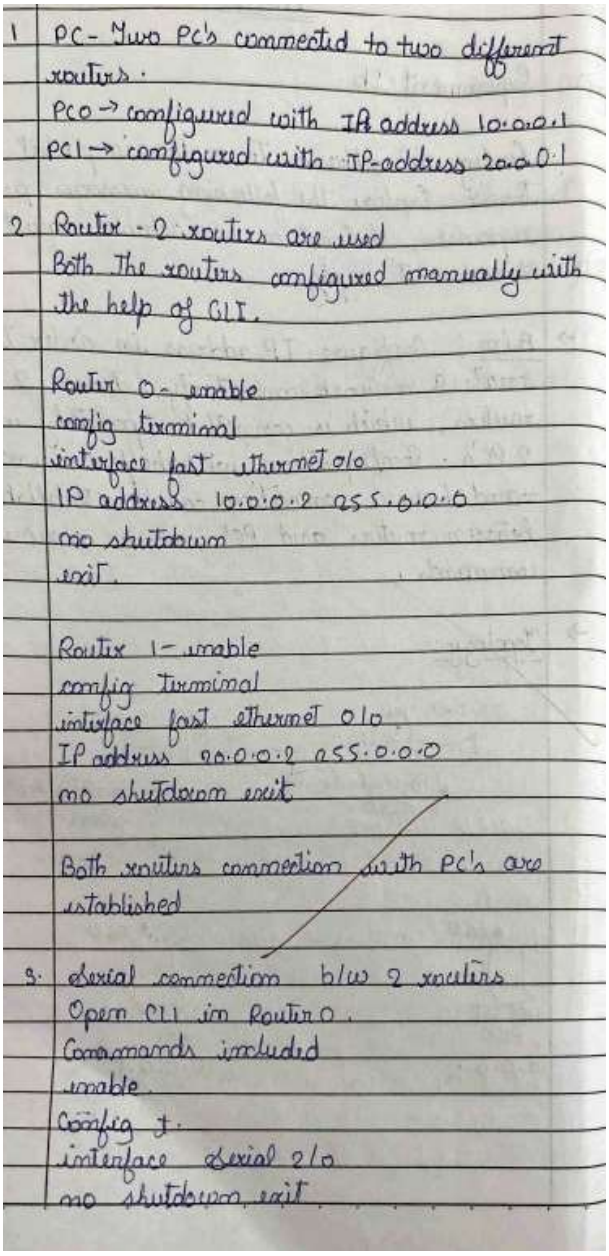
LAB-3

Experiment - 2b

Config: IP addresses To routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply

→ Aim: Configure IP address in order to create a network connectivity between 2 routers, which is connected separately with 2 PC's. Configuration will help us understand how a connection can be established between routers and PC's, observe various commands.

→ Topology:



1 PC - Two PC's connected to two different routers.
PC0 → configured with IP address 10.0.0.1
PC1 → configured with IP-address 20.0.0.1

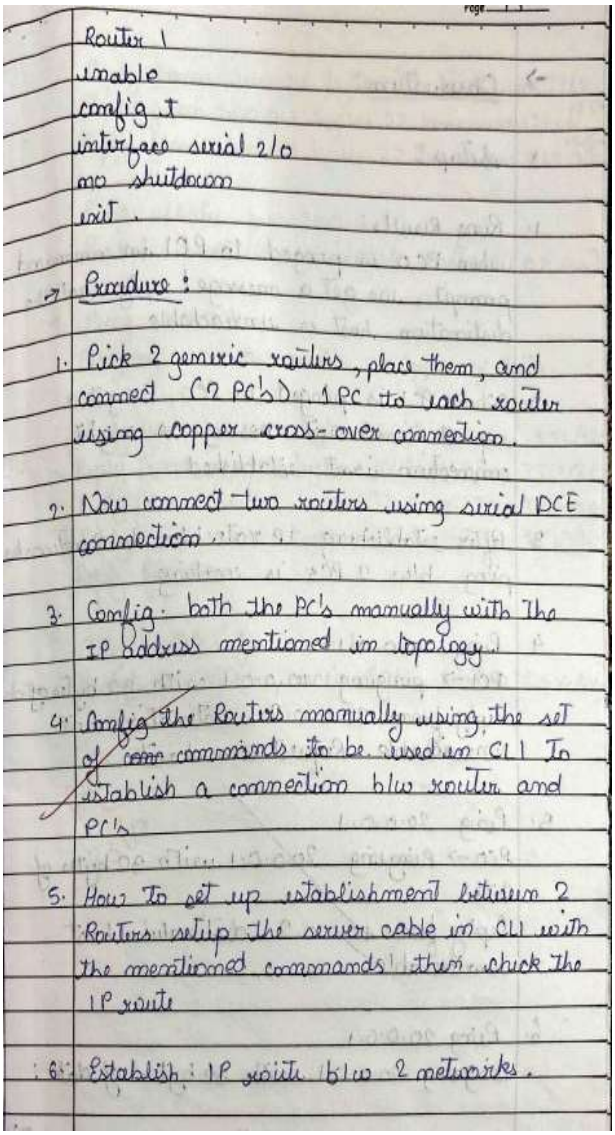
2 Router - 2 routers are used
Both the routers configured manually with the help of CLI.

Router 0 - enable
config terminal
interface fast ethernet 0/0
IP address 10.0.0.2 255.0.0.0
no shutdown
exit.

Router 1 - enable
config Terminal
interface fast ethernet 0/0
IP address 20.0.0.2 255.0.0.0
no shutdown exit

Both routers connection with PC's are established

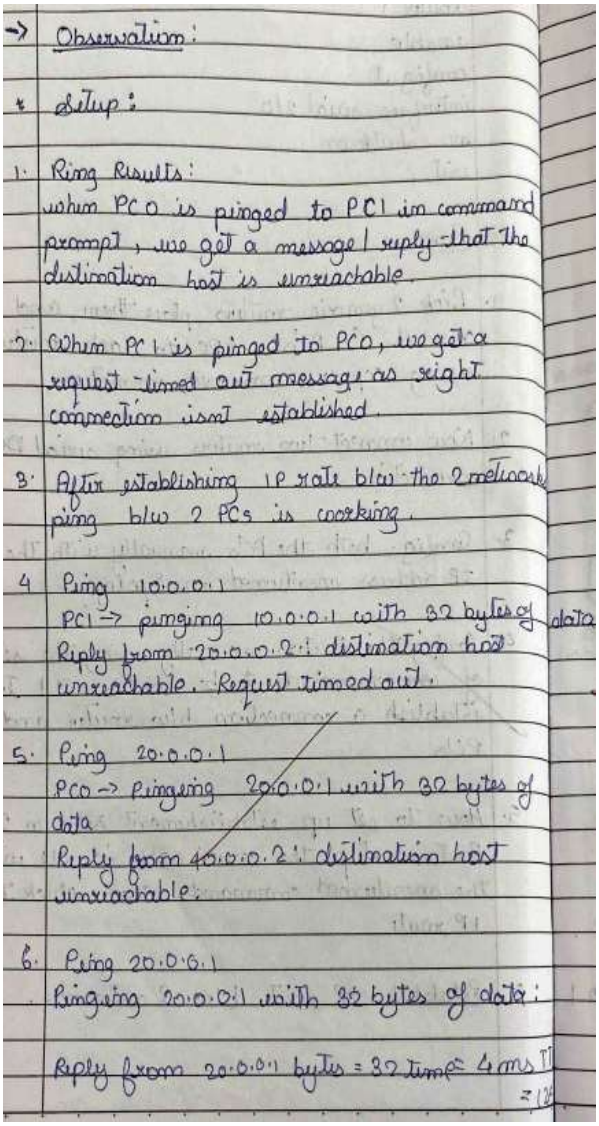
3. Serial connection b/w 2 routers
Open CLI in Router 0.
Commands included
enable
Config t.
interface Serial 2/0
no shutdown exit



Router 1
enable
config t
interface serial 2/0
no shutdown
exit

→ Procedure:

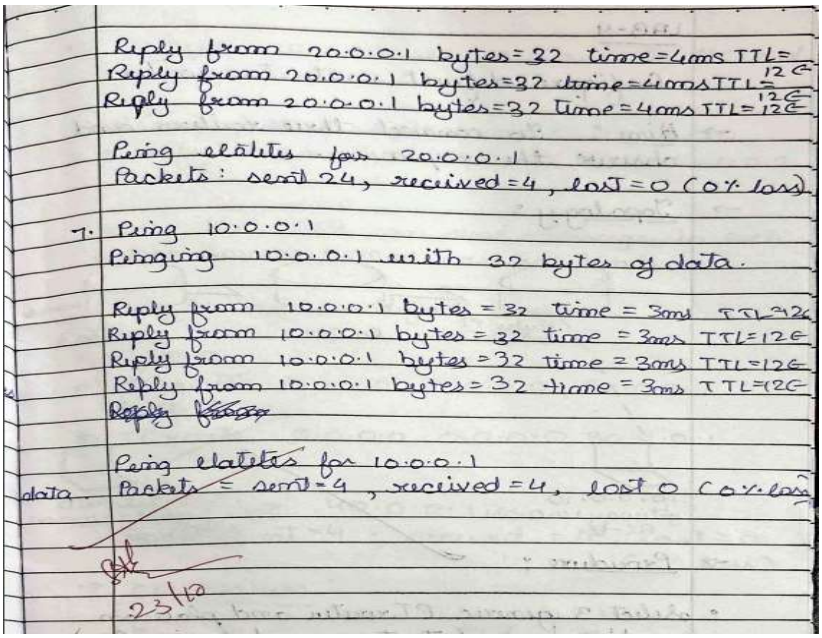
1. Pick 2 generic routers, place them, and connect (2 PC's). 1 PC to each router using copper cross-over connection.
2. Now connect two routers using serial DCE connection.
3. Config. both the PC's manually with the IP address mentioned in topology.
4. Config the Routers manually using the set of commands to be used in CLI To establish a connection b/w router and PC's
5. How To set up establishment between 2 Routers setup the server cable in CLI with the mentioned commands then check the IP route
6. Establish IP route b/w 2 networks.



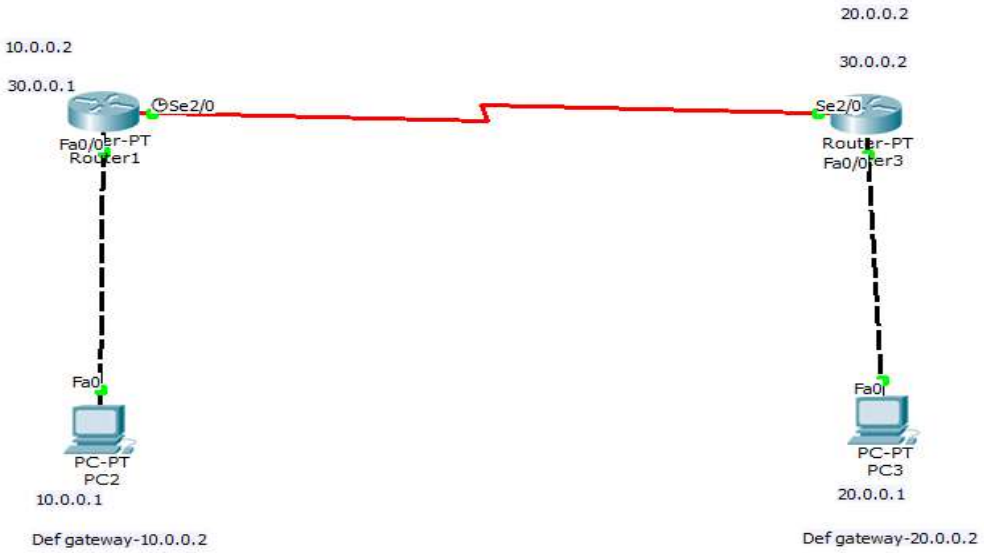
→ Observation:

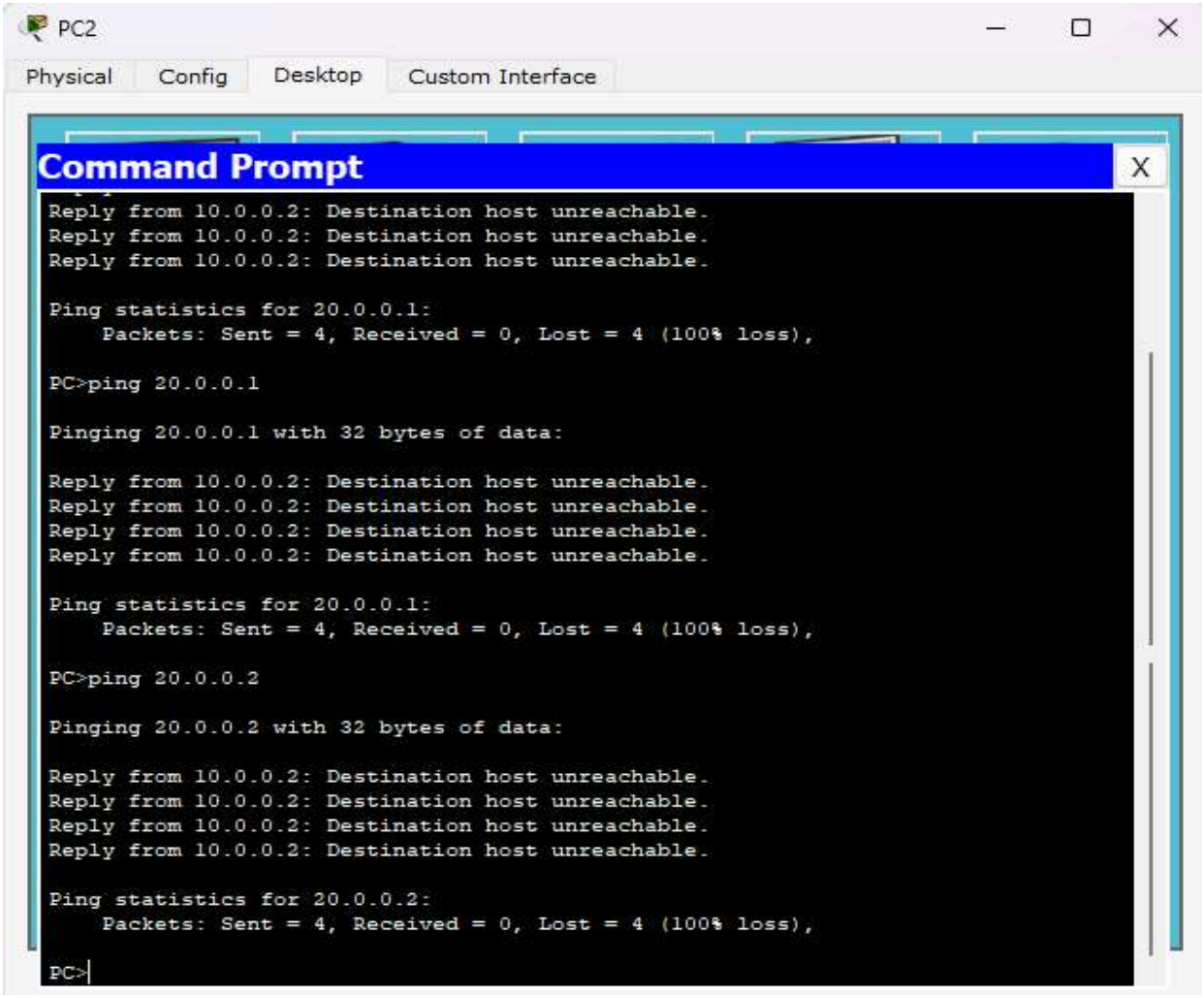
* Setup:

1. Ping Results:
when PC0 is pinged to PC1 in command prompt, we get a message / reply that the distination host is unreachable.
2. When PC1 is pinged to PC0, we get a request timed out message as right connection isn't established.
3. After establishing IP route b/w the 2 networks ping b/w 2 PCs is working.
4. Ping 10.0.0.1
PC1 → pinging 10.0.0.1 with 32 bytes of data
Reply from 20.0.0.2: distination host unreachable. Request timed out.
5. Ping 20.0.0.1
PC0 → Pinging 20.0.0.1 with 32 bytes of data
Reply from 20.0.0.2: distination host unreachable.
6. Ping 20.0.0.1
Pinging 20.0.0.1 with 32 bytes of data:
Reply from 20.0.0.1 bytes = 32 time = 4ms TT



Screen Shots:

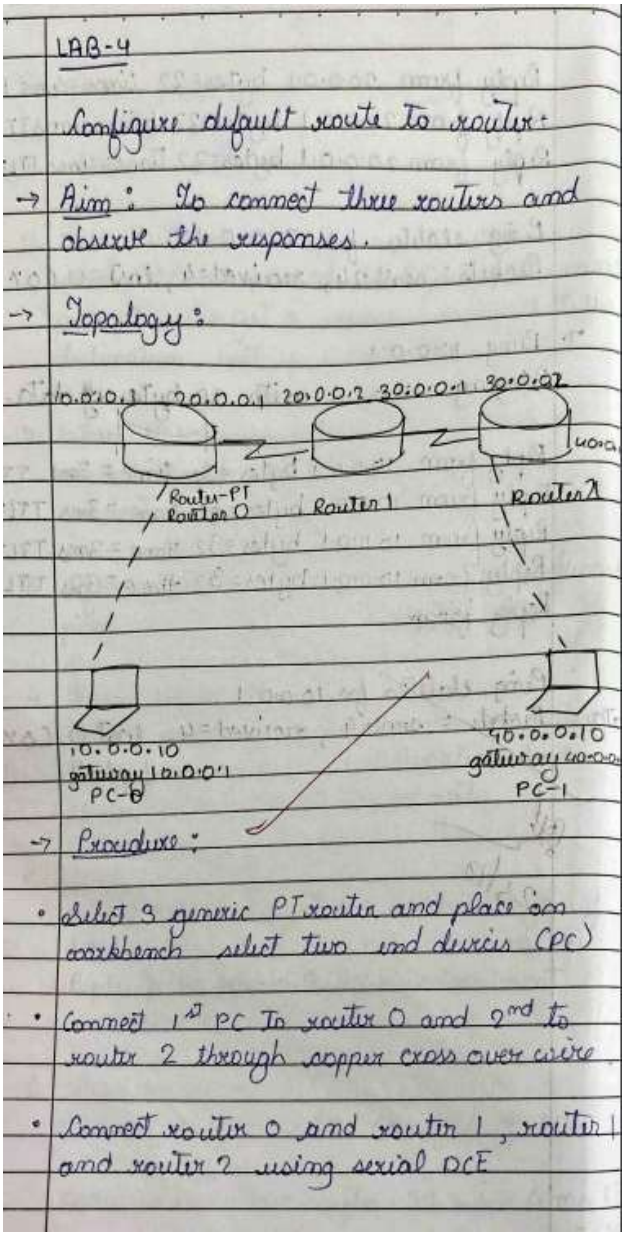


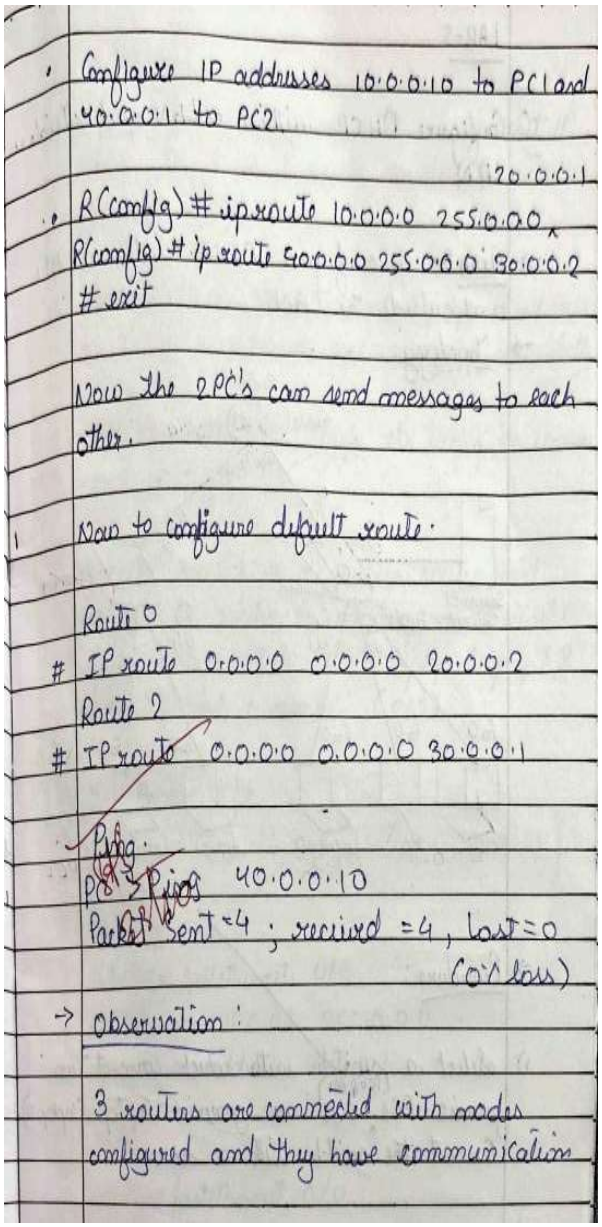

PC2
Physical
Config
Desktop
Custom Interface
Command Prompt
Reply from 10.0.0.2: Destination host unreachable.
Reply from 10.0.0.2: Destination host unreachable.
Reply from 10.0.0.2: Destination host unreachable.
Ping statistics for 20.0.0.1:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>ping 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 20.0.0.2
Pinging 20.0.0.2 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 20.0.0.2:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>

Program 4

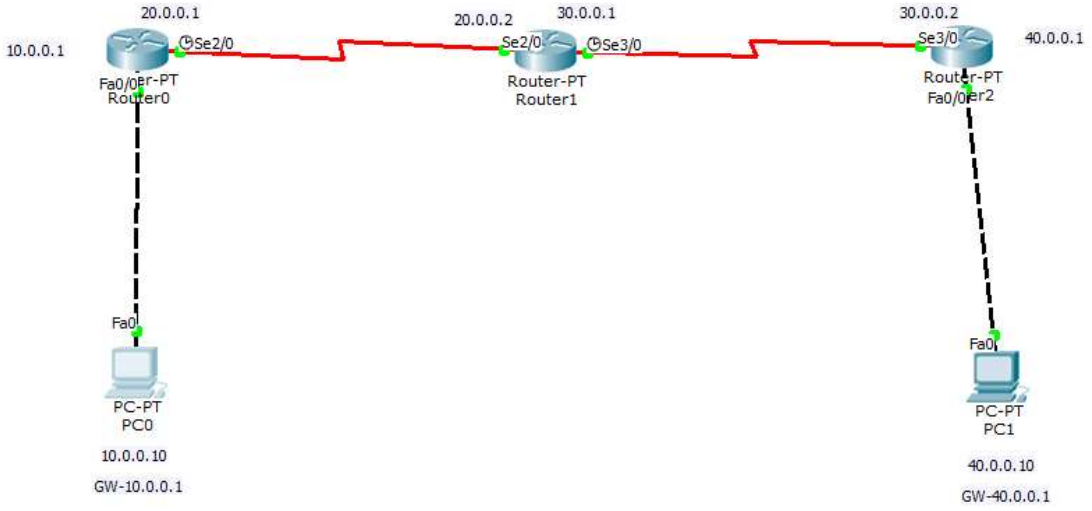
Aim: Configure default route, static route to the Router(Part 2).

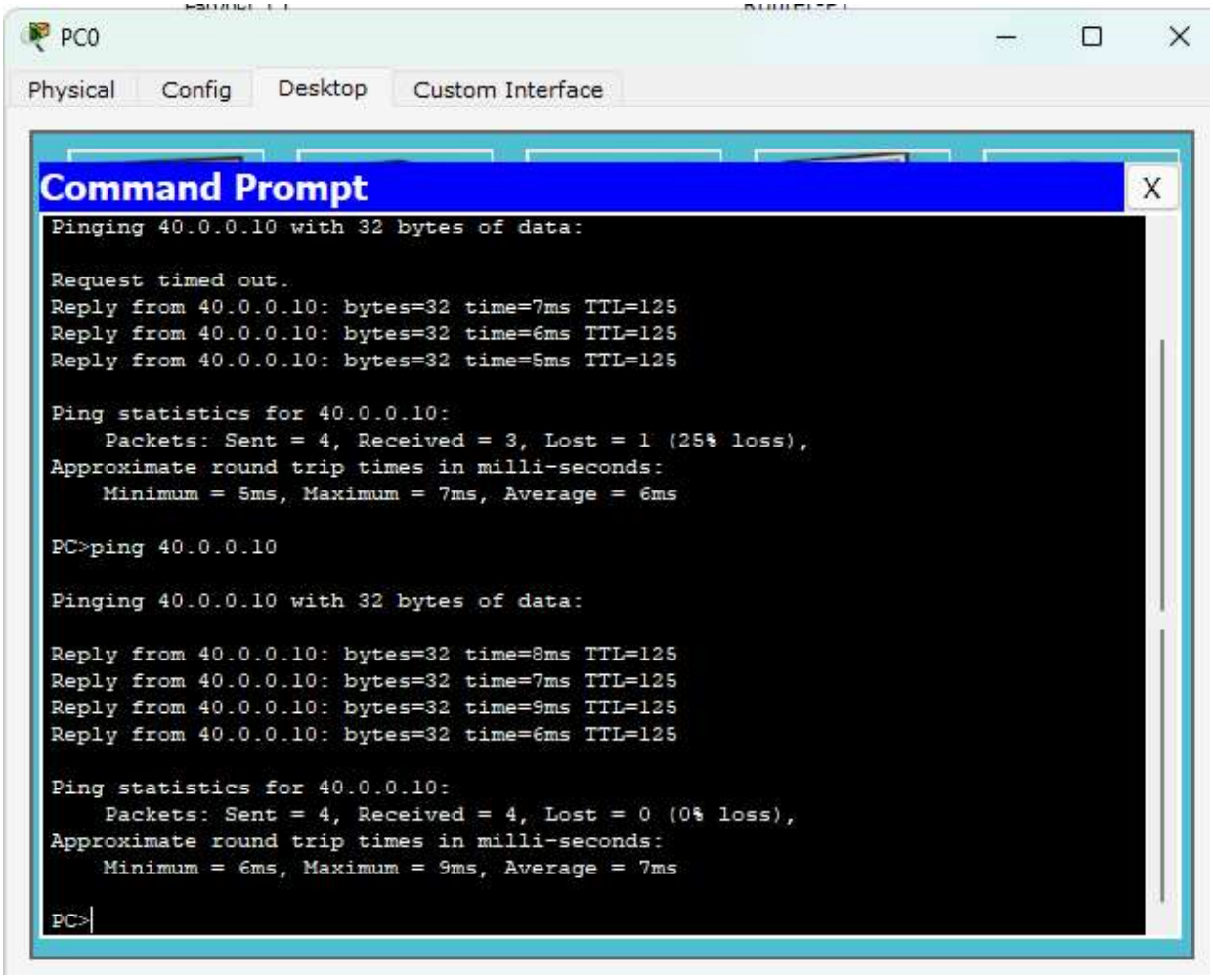
Topology , Procedure and Observation:





Screen Shots:

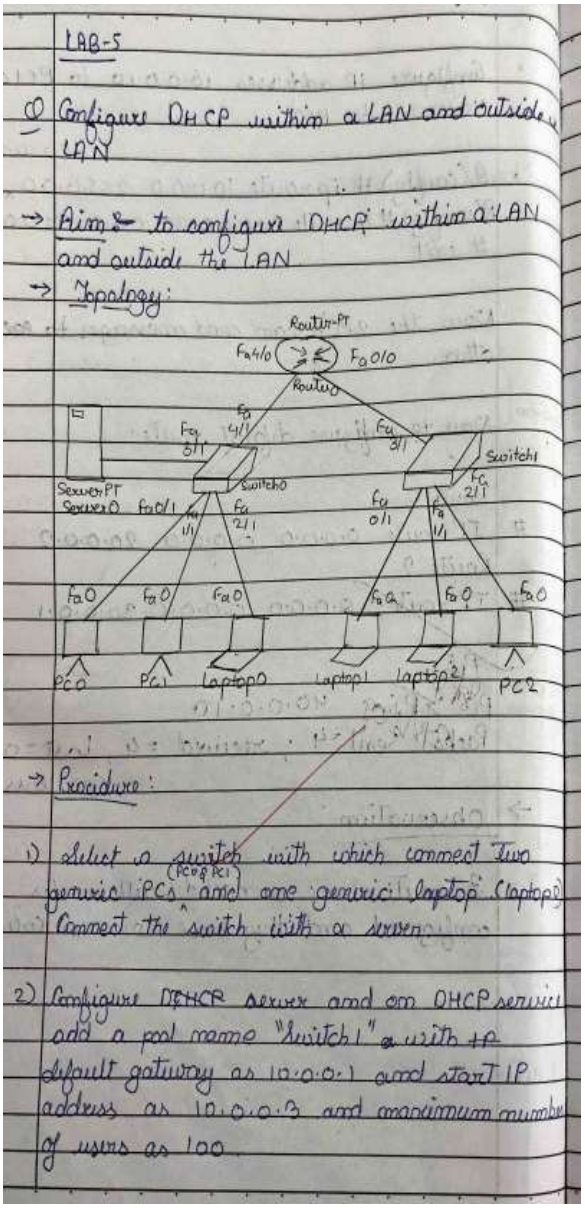


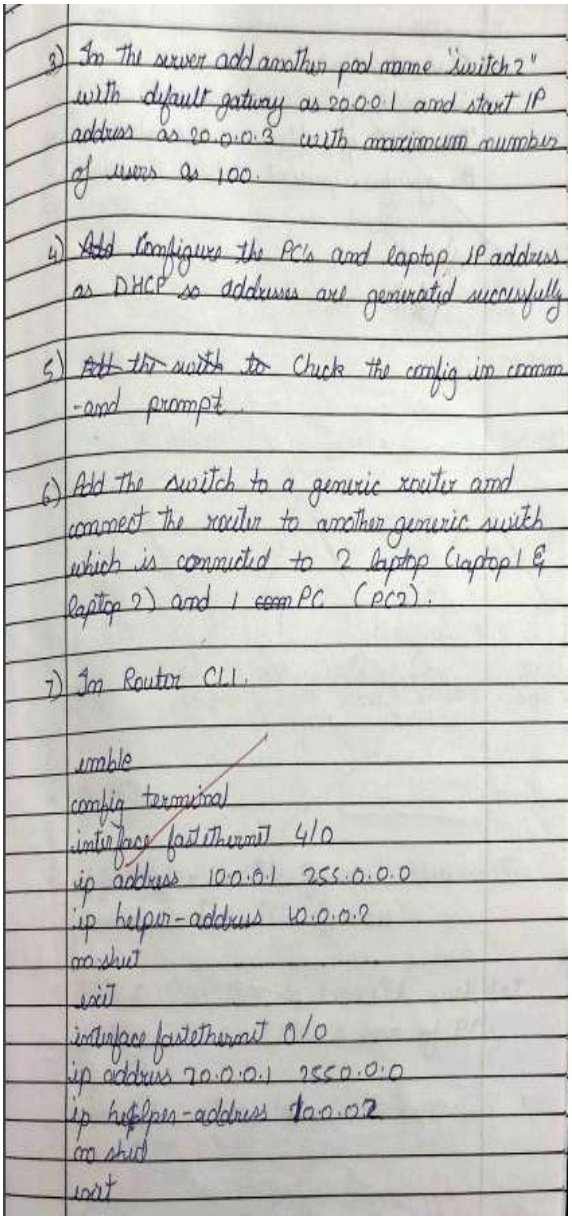


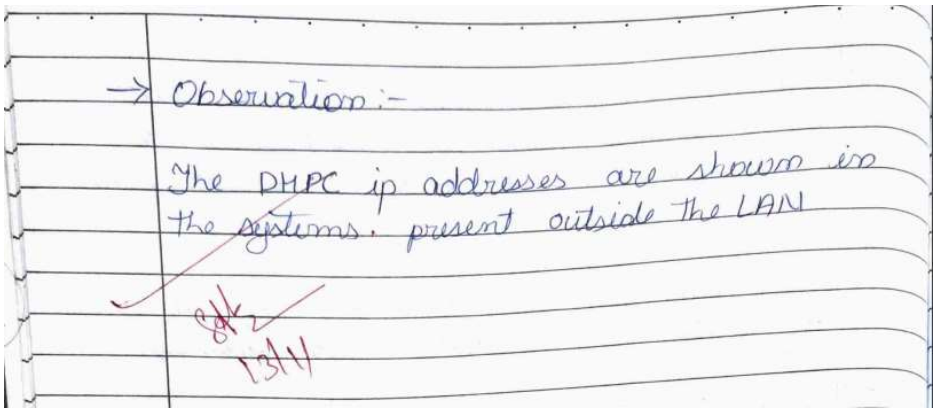
Program 5

Aim: Configure DHCP within a LAN and outside LAN.

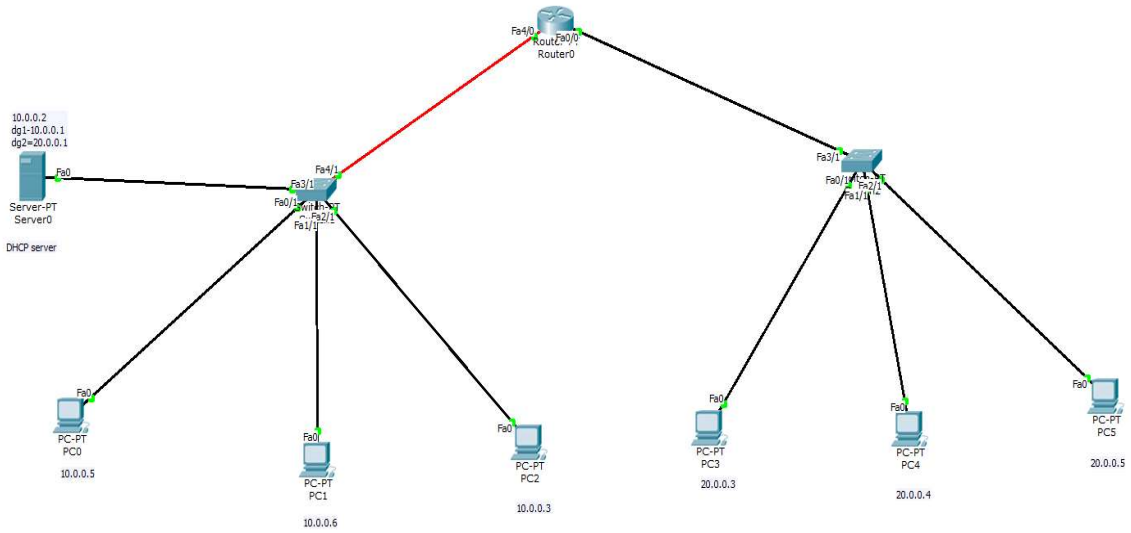
Topology , Procedure and Observation:

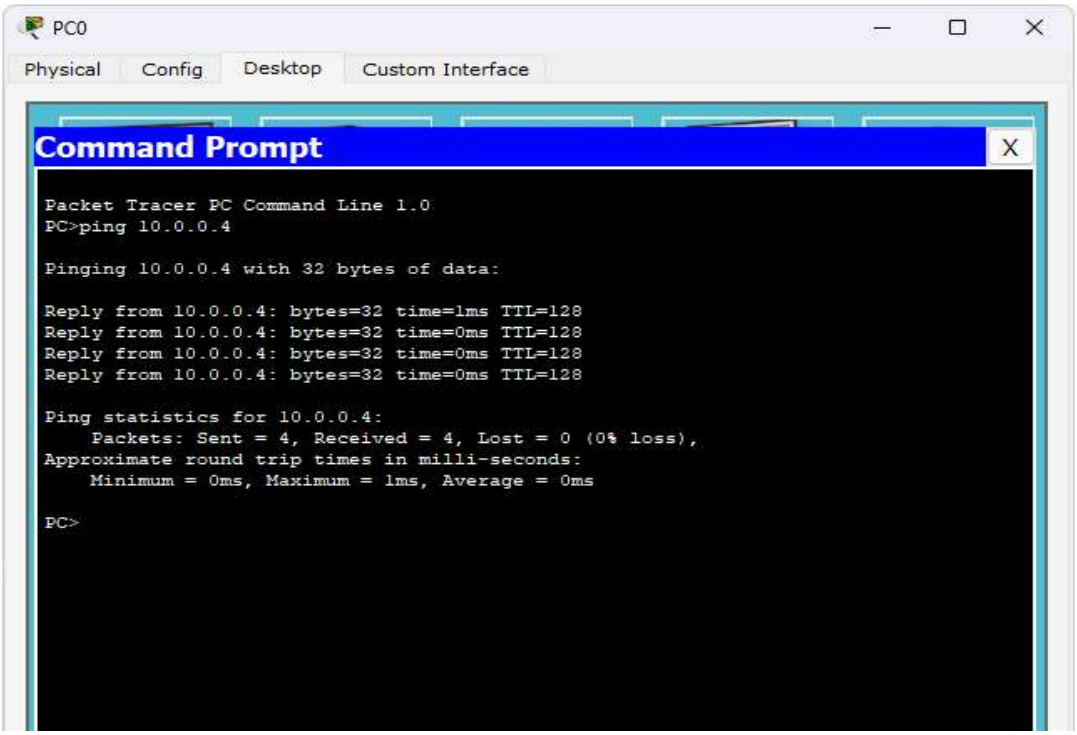






Screen Shots:

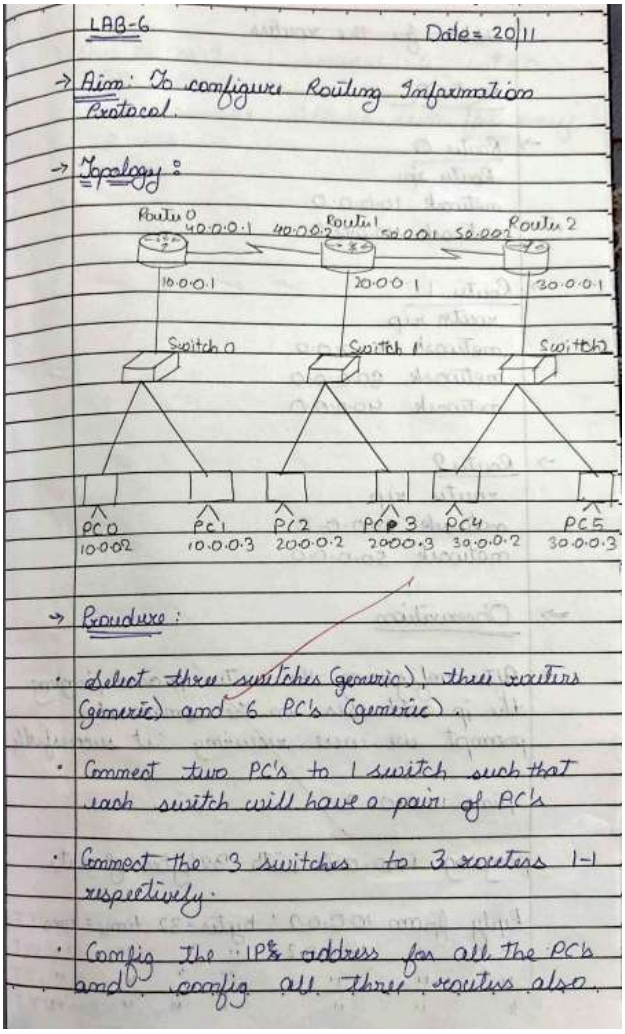


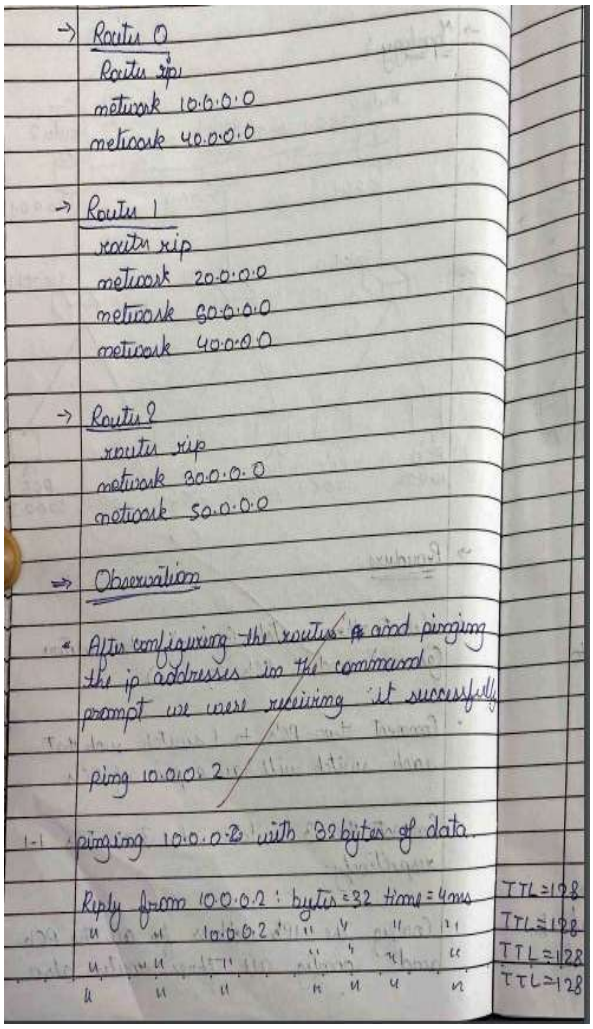

PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.4
Pinging 10.0.0.4 with 32 bytes of data:
Reply from 10.0.0.4: bytes=32 time=1ms TTL=128
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Reply from 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 = 1ms, Average = 0ms
PC>

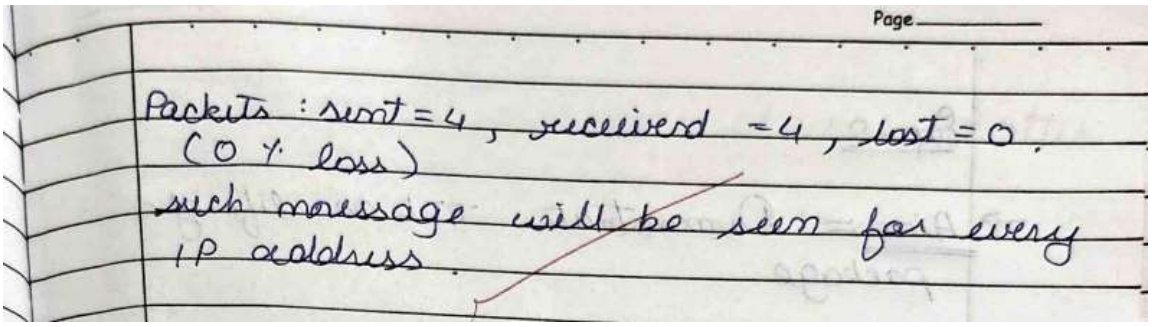
Program 6

Aim: Configure RIP routing Protocol in Routers .

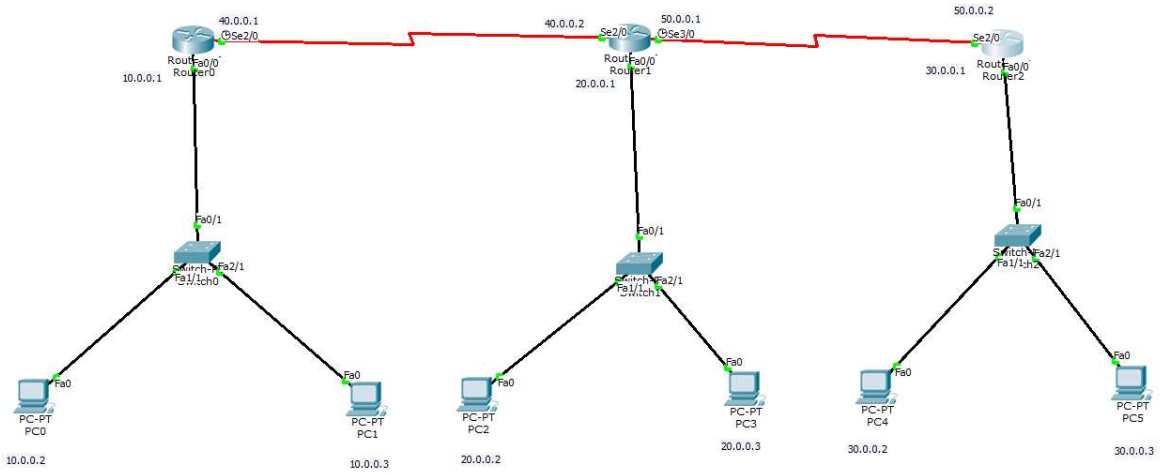
Topology , Procedure and Observation:

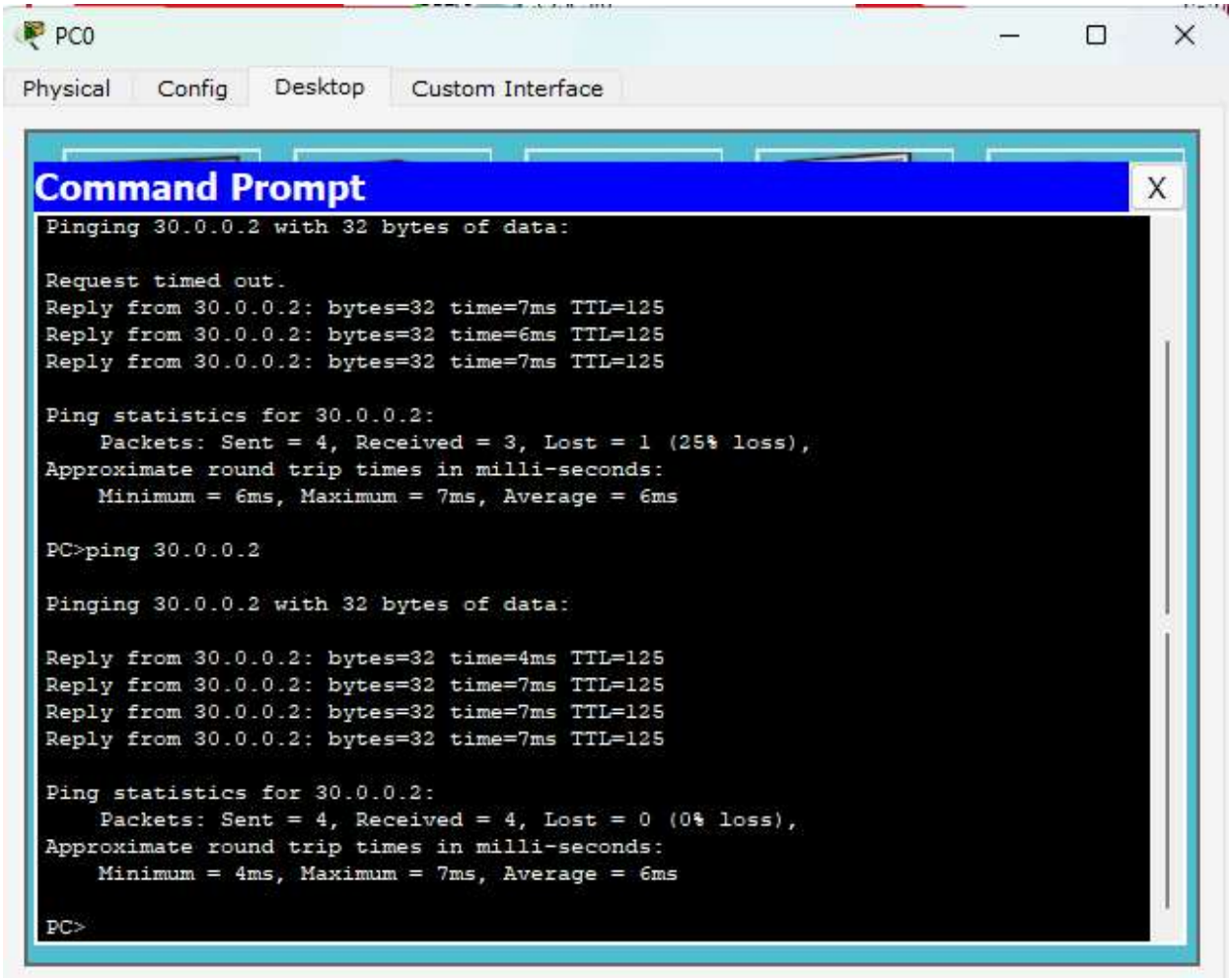






Screen Shots:

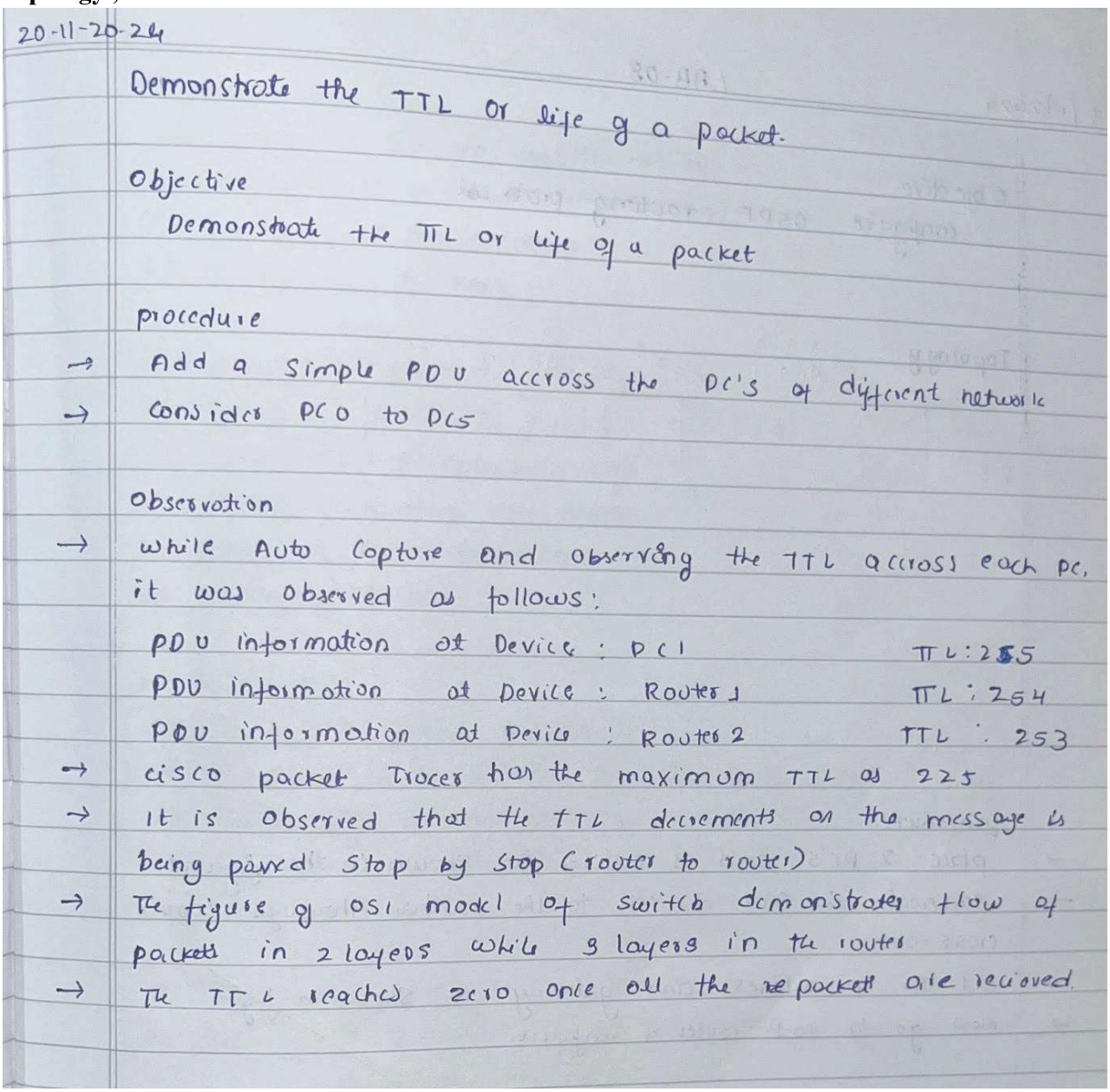




Program 7

Aim: Demonstrate the TTL/ Life of a Packet .

Topology , Procedure and Observation:



20-11-2024

Demonstrate the TTL or life of a packet.

Objective

Demonstrate the TTL or life of a packet

procedure

→ Add a simple PDU accross the PC's of different network

→ Consider PC0 to PC5

Observation

→ while Auto Capture and observing the TTL across each PC, it was observed as follows:

PDU information at Device : PC1 TTL: 255

PDU information at Device : Router1 TTL : 254

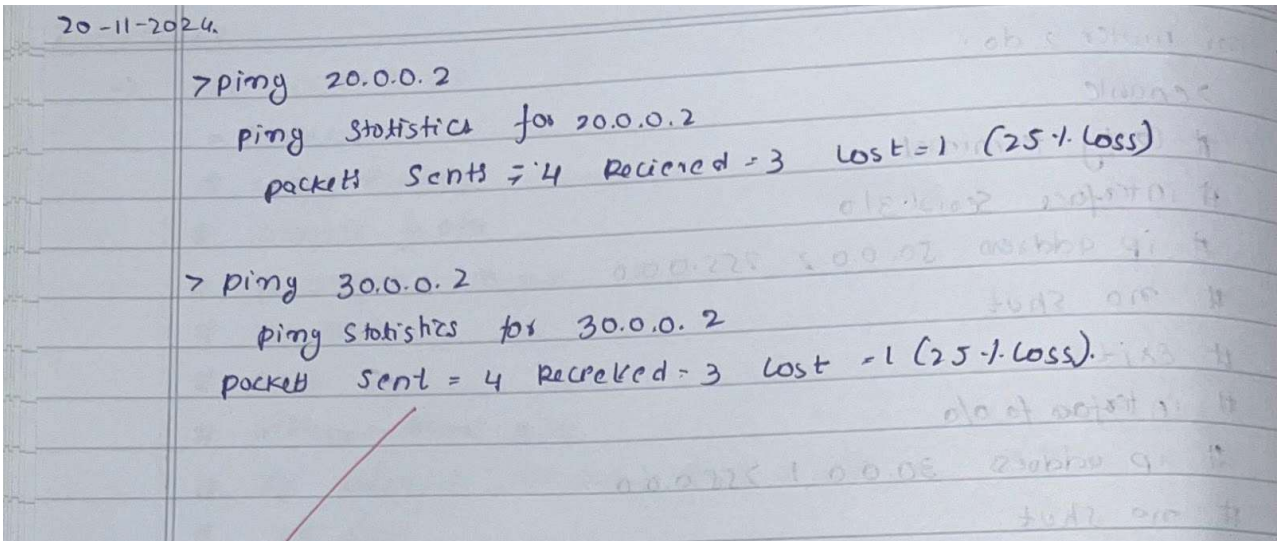
PDU information at Device : Router2 TTL : 253

→ cisco packet Tracer has the maximum TTL as 225

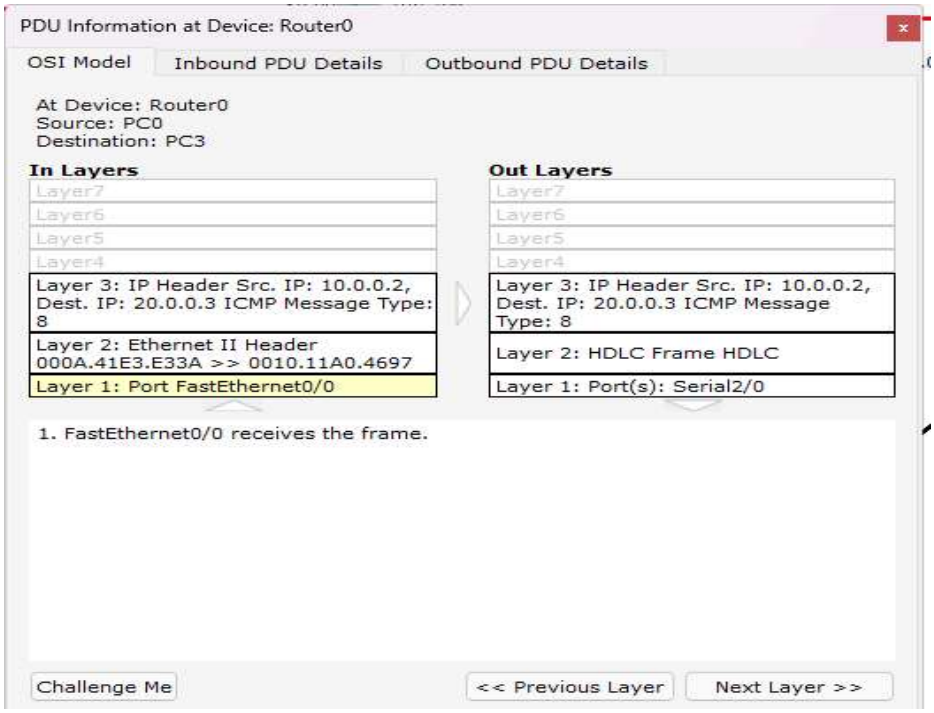
→ It is observed that the TTL decrements as the message is being passed Stop by Stop (router to router)

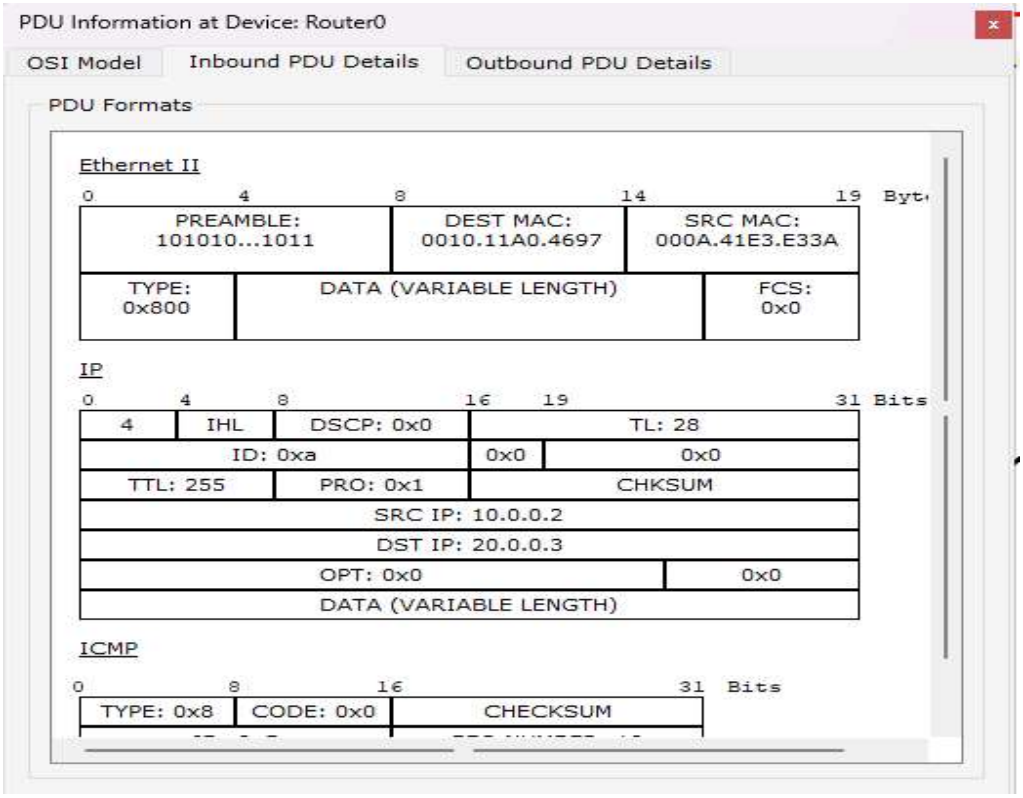
→ The figure of OSI model of switch demonstrates flow of packets in 2 layers while 3 layers in the routers

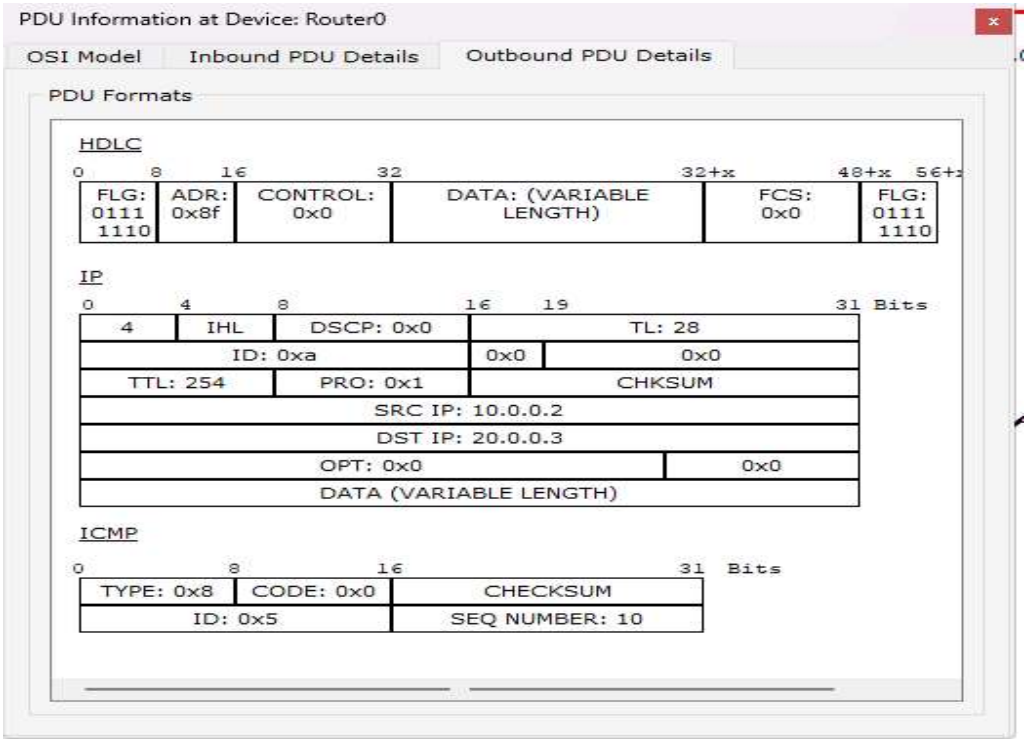
→ The TTL reaches zero once all the packets are recieved.



Screen Shots:



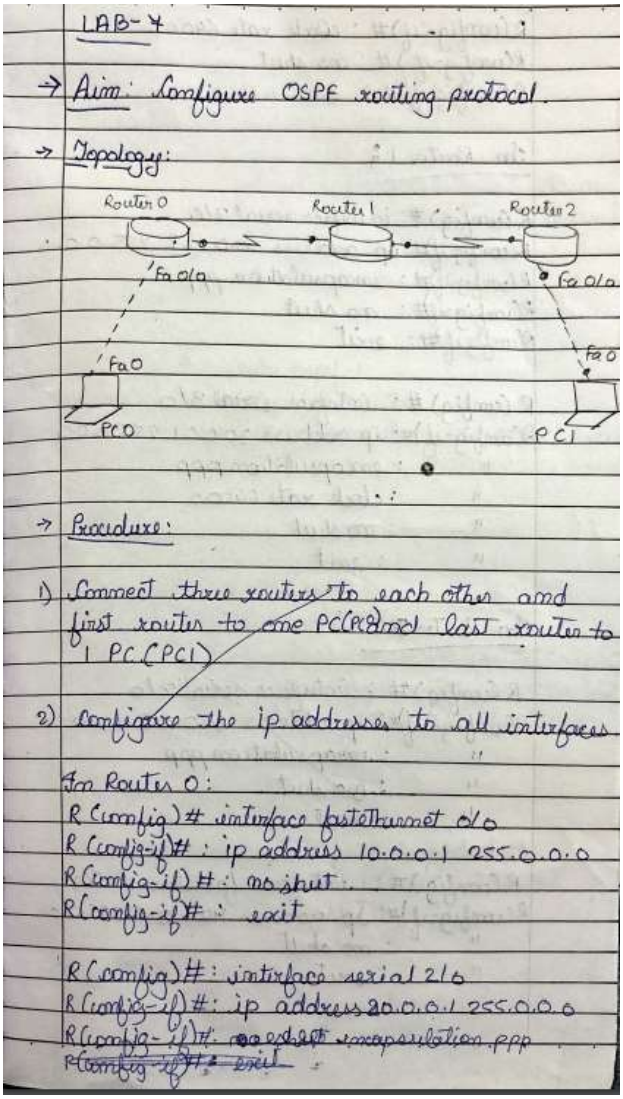

PDU Information at Device: Router0
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
Ethernet II
0 4 8 14 19 Byt
PREAMBLE: 101010...1011
DEST MAC: 0010.11A0.4697
SRC MAC: 000A.41E3.E33A
TYPE: 0x800
DATA (VARIABLE LENGTH)
FCS: 0x0
IP
0 4 8 16 19 31 Bits
4
IHL
DSCP: 0x0
TL: 28
ID: 0xa
0x0
0x0
TTL: 255
PRO: 0x1
CHKSUM
SRC IP: 10.0.0.2
DST IP: 20.0.0.3
OPT: 0x0
0x0
DATA (VARIABLE LENGTH)
ICMP
0 8 16 31 Bits
TYPE: 0x8
CODE: 0x0
CHECKSUM

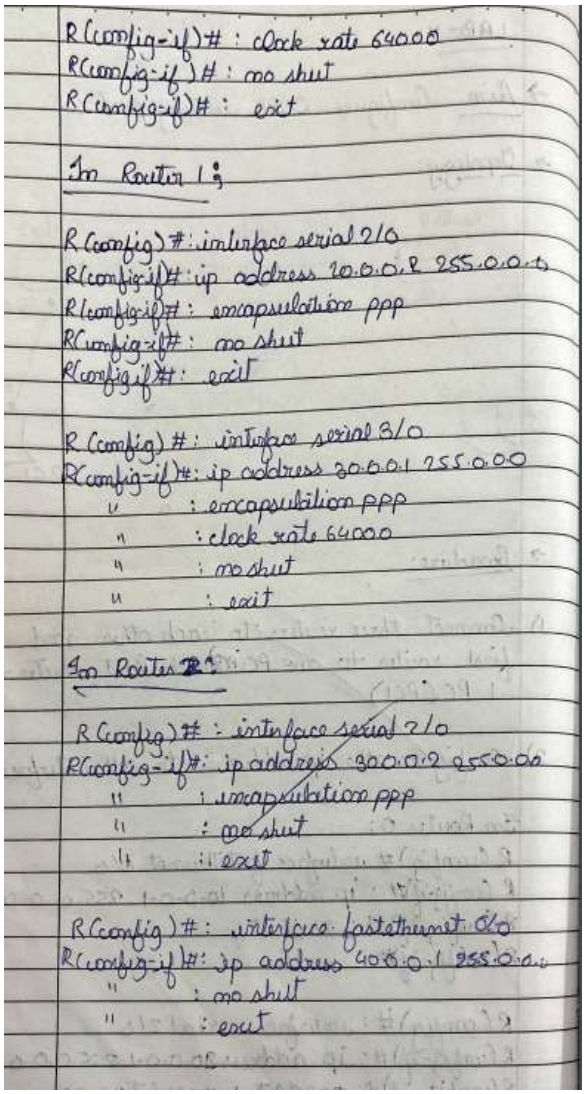

PDU Information at Device: Router0
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
HDLC
0 8 16 32 32+x 48+x
FLG: 0111 1110
ADR: 0x8f
CONTROL: 0x0
DATA: (VARIABLE LENGTH)
FCS: 0x0
FLG: 0111 1110
IP
0 4 8 16 19 31 Bits
4
IHL
DSCP: 0x0
TL: 28
ID: 0xa
0x0
0x0
TTL: 254
PRO: 0x1
CHKSUM
SRC IP: 10.0.0.2
DST IP: 20.0.0.3
OPT: 0x0
0x0
DATA (VARIABLE LENGTH)
ICMP
0 8 16 31 Bits
TYPE: 0x8
CODE: 0x0
CHECKSUM
ID: 0x5
SEQ NUMBER: 10

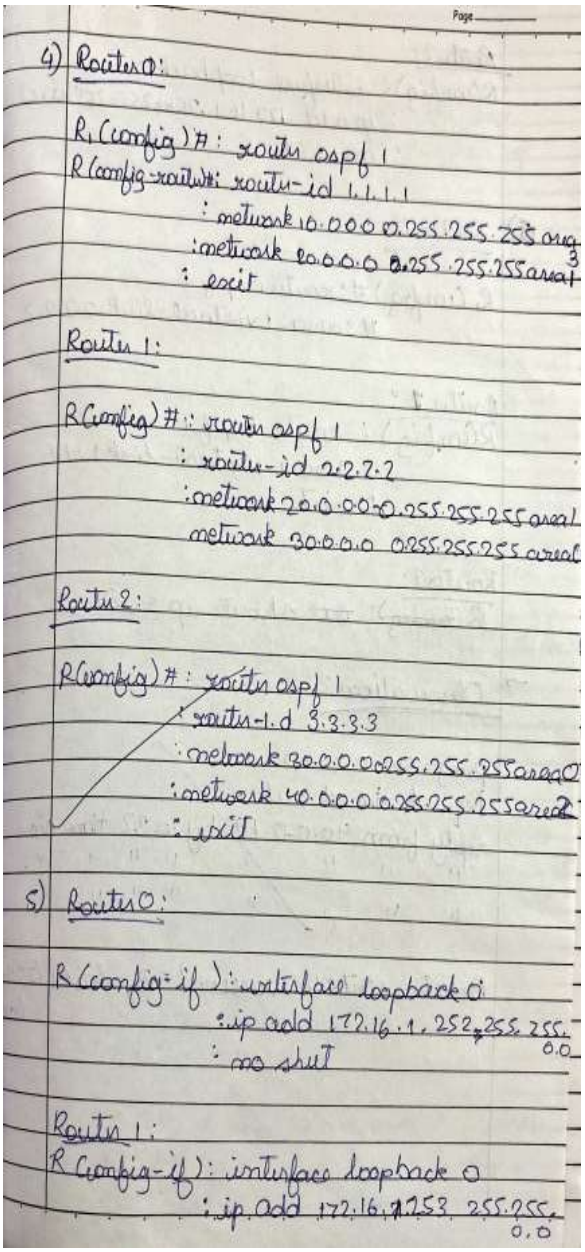
Program 8

Aim: Configure OSPF routing protocol .

Topology , Procedure and Observation:







4) Router 0:

R1(config)#: router ospf 1
R(config-router)#: router-id 1.1.1.1
: network 10.0.0.0 0.255.255.255 area 3
: network 20.0.0.0 0.255.255.255 area 1
: exit

Router 1:

R(config)#: router ospf 1
: router-id 2.2.2.2
: network 20.0.00-0.255.255.255 area 1
network 30.0.0.0 0.255.255.255 area 1

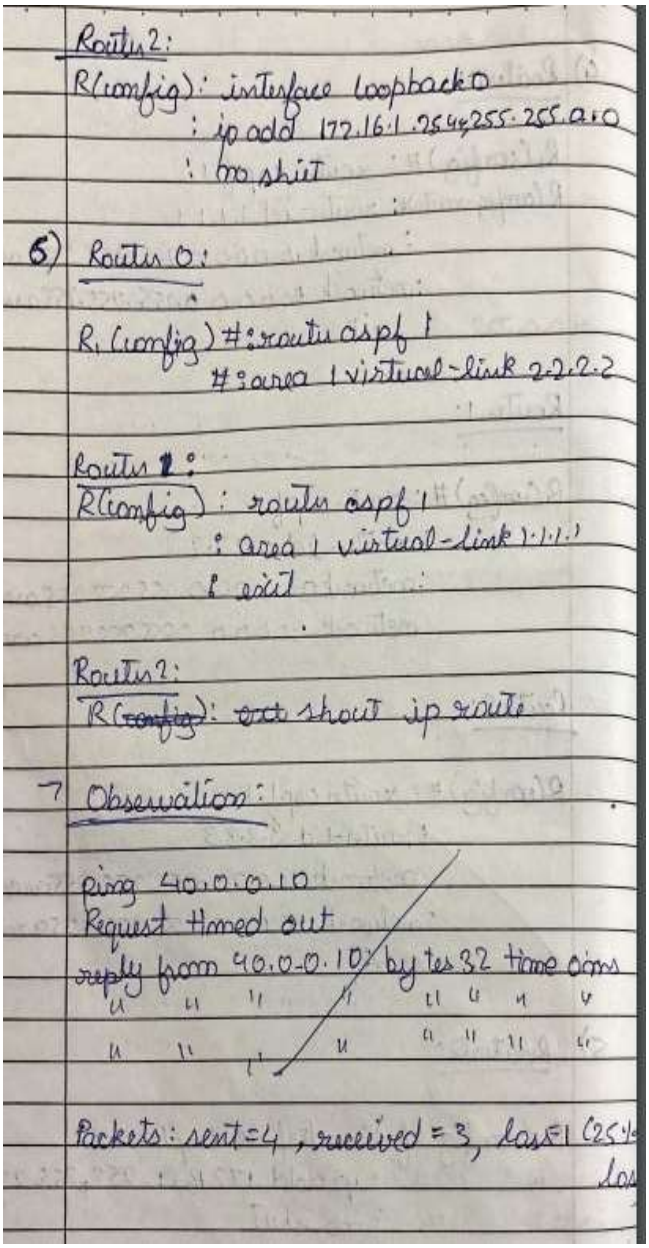
Router 2:

R(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

5) Router 0:

R(config-if): interface loopback 0
: ip add 172.16.1.252 255.255.255.0
: no shut

Router 1:
R(config-if): interface loopback 0
: ip add 172.16.1.253 255.255.255.0



Router 2:
R(config): interface loopback 0
: ip add 172.16.1.254 255.255.255.0
: no shut

6) Router 0:

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

Router 1:
R(config): router ospf 1
: area 1 virtual-link 1.1.1.1
: exit

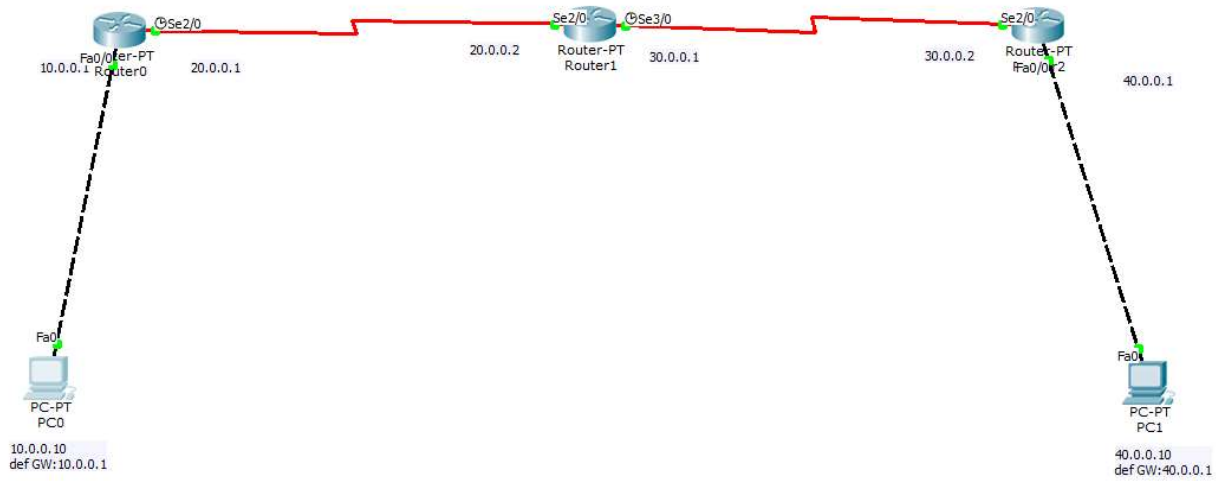
Router 2:
R(config): ~~ctrl~~ shout ip route

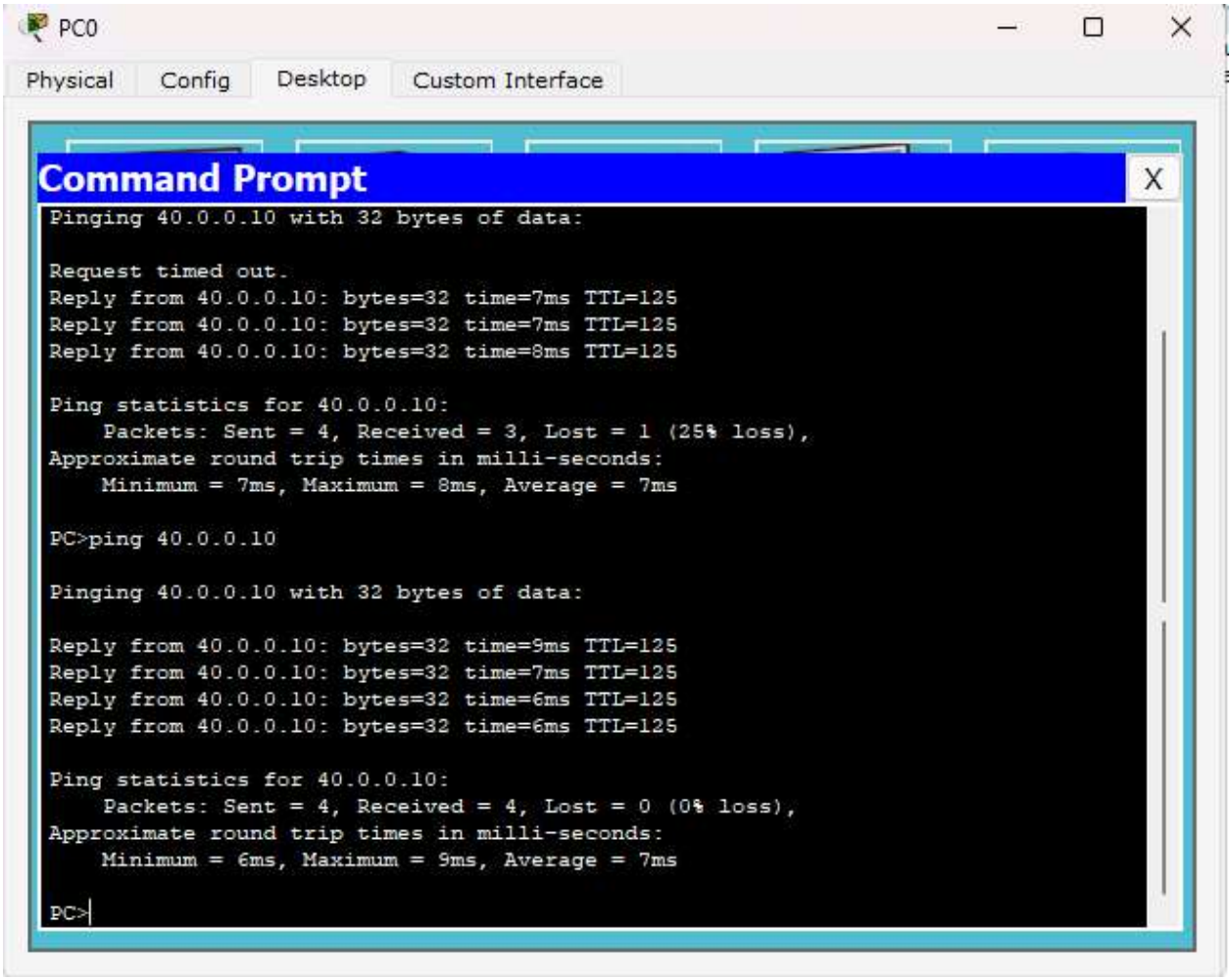
7 Observation:

ping 40.0.0.10
Request timed out
reply from 40.0.0.10 by tes 32 time oms

Packets: sent=4, received = 3, loss=1 (25%

Screen Shots:

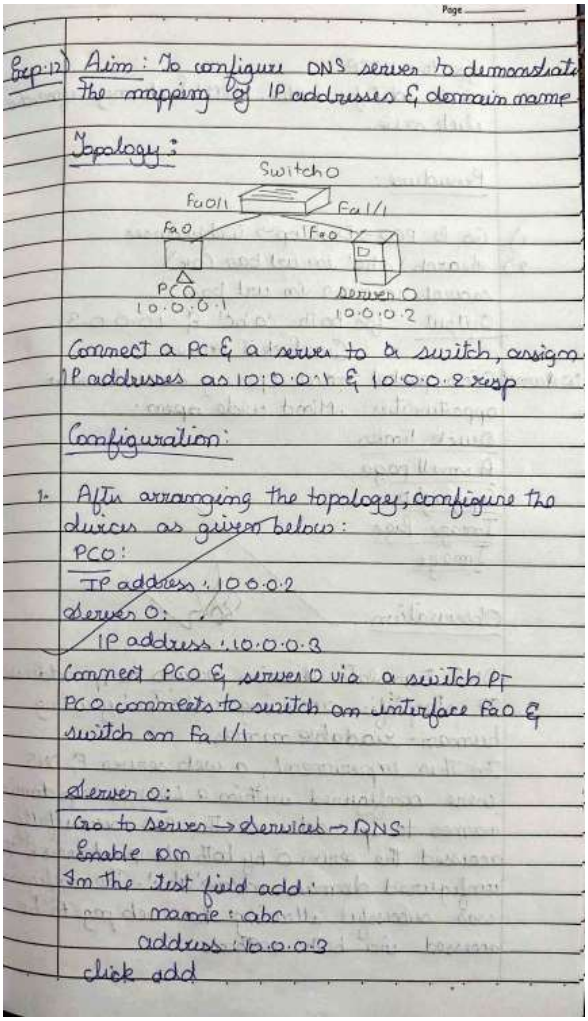




Program 9

Aim: Configure Web Server, DNS within a LAN.

Topology , Procedure and Observation:



Exp:12) Aim: To configure DNS server to demonstrate the mapping of IP addresses & domain name

Topology:

Connect a PC & a server to a switch, assign IP addresses as 10.0.0.1 & 10.0.0.2 resp

Configuration:

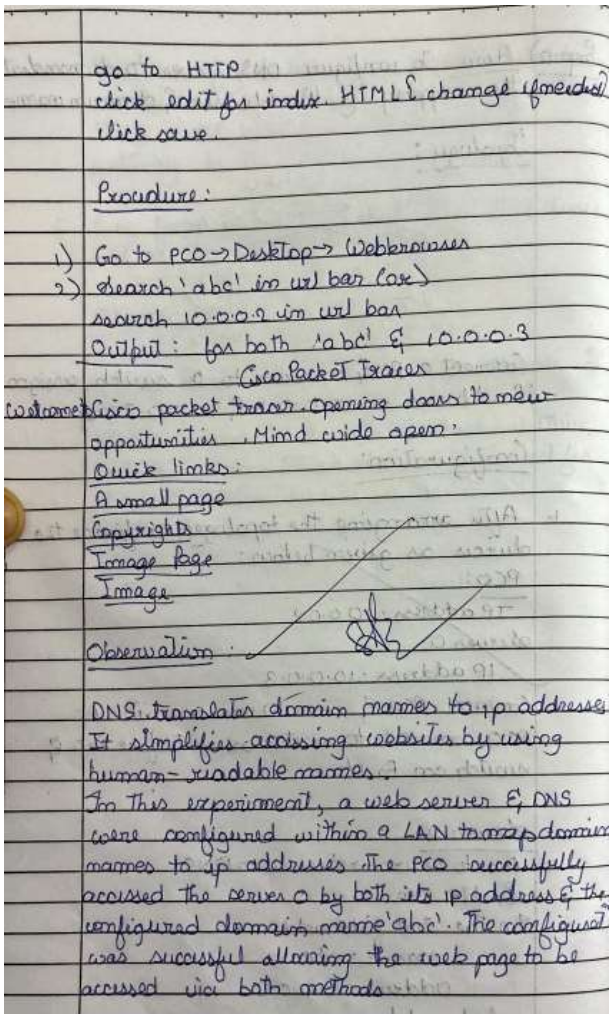
1. After arranging the topology, configure the devices as given below:

PC0:
IP address: 10.0.0.2

Server 0:
IP address: 10.0.0.3

Connect PC0 & server 0 via a switch PT
PC0 connects to switch on interface Fa0 &
switch on Fa 1/1

Server 0:
Go to Server → Services → DNS
Enable DNS
In the first field add:
name: abc
address: 10.0.0.3
click add



go to HTTP
click edit for index.HTML & change if needed
click save.

Procedure:

1) Go to PC0 → Desktop → Webbrowser
2) Search 'abc' in url bar (or)
search 10.0.0.2 in url bar

Output: for both 'abc' & 10.0.0.3

Cisco Packet Tracer

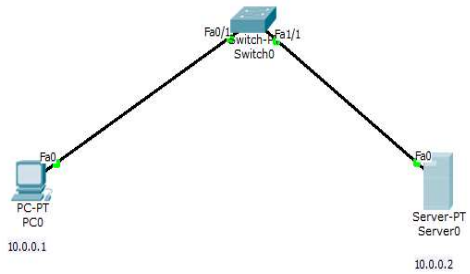
Welcome to Cisco packet tracer. Opening doors to new opportunities. Mind wide open.

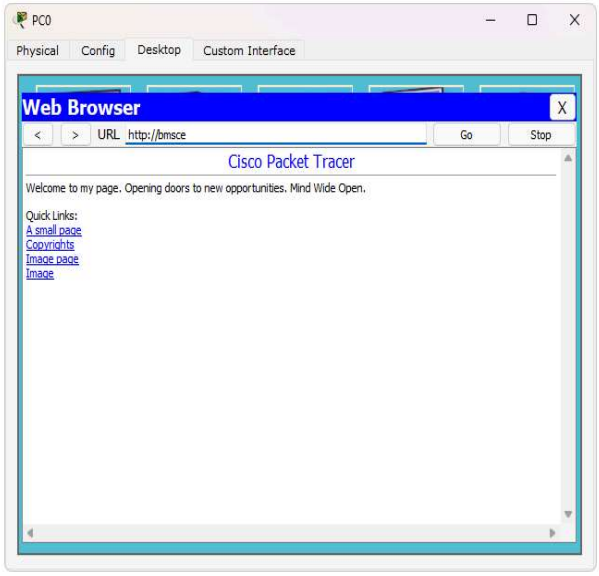
Quick links:
A small page
Copyrights
Image Page
Image

Observation:

DNS translates domain names to ip addresses. It simplifies accessing websites by using human-readable names.
In this experiment, a web server & DNS were configured within a LAN to map domain names to ip addresses. The PC0 successfully accessed the server 0 by both its ip address & the configured domain name 'abc'. The configuration was successful allowing the web page to be accessed via both methods.

Screen Shots:

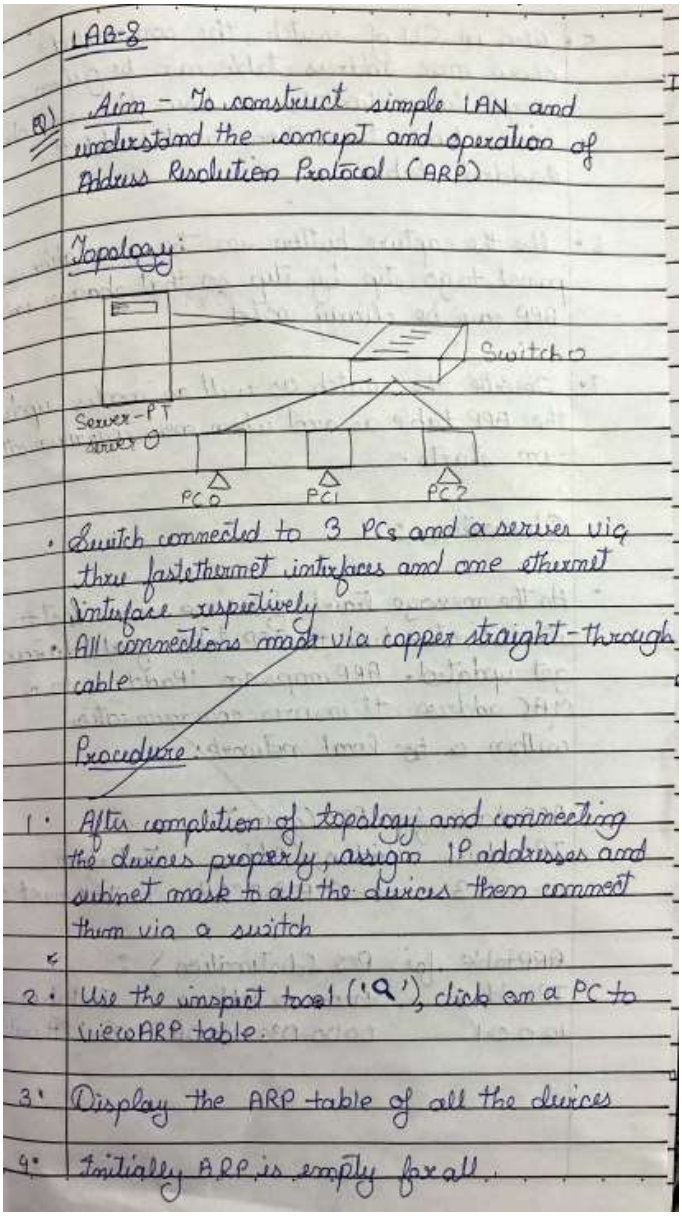


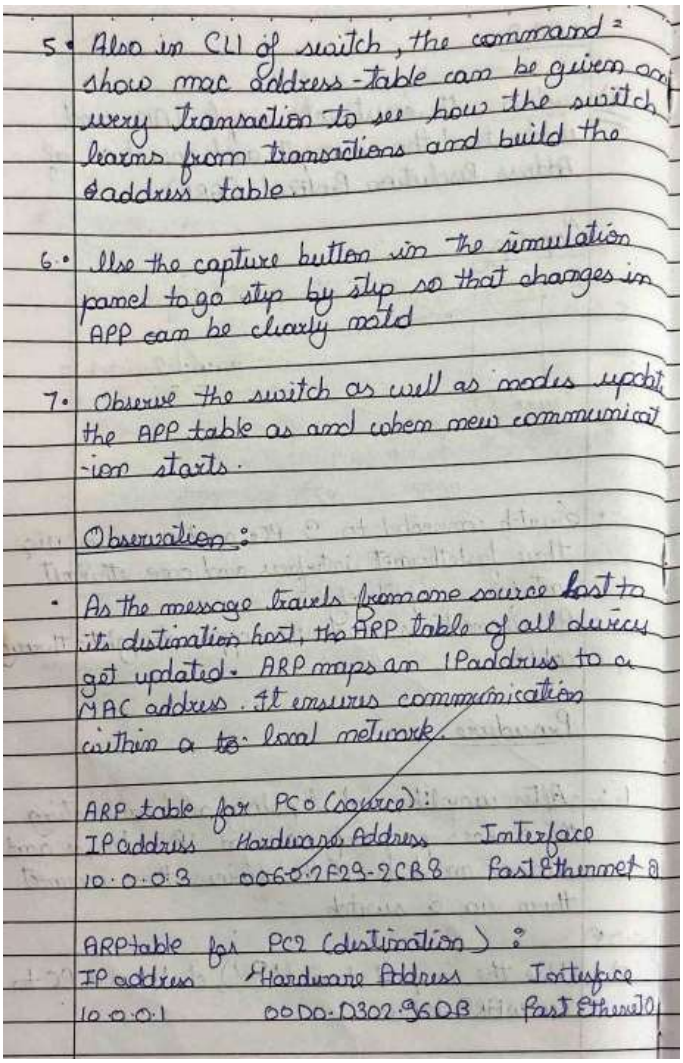


Program 10

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

Topology , Procedure and Observation:





5. Also in CLI of switch, the command = show mac address-table can be given and every transaction to see how the switch learns from transactions and build the address table.

6. Use the capture button in the simulation panel to go step by step so that changes in APP can be clearly noted

7. Observe the switch as well as modes update the APP table as and when new communication starts.

Observation:

- As the message travels from one source host to its distination host, the ARP table of all devices get updated. ARP maps an IP address to a MAC address. It ensures communication within a local network.

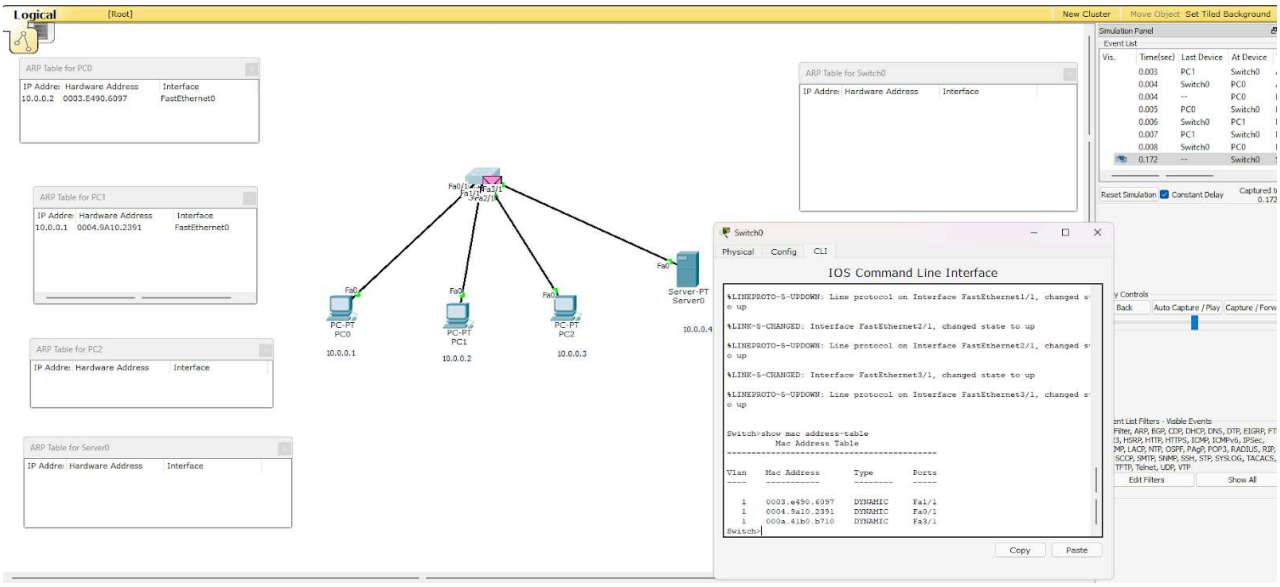
ARP table for PC0 (source):

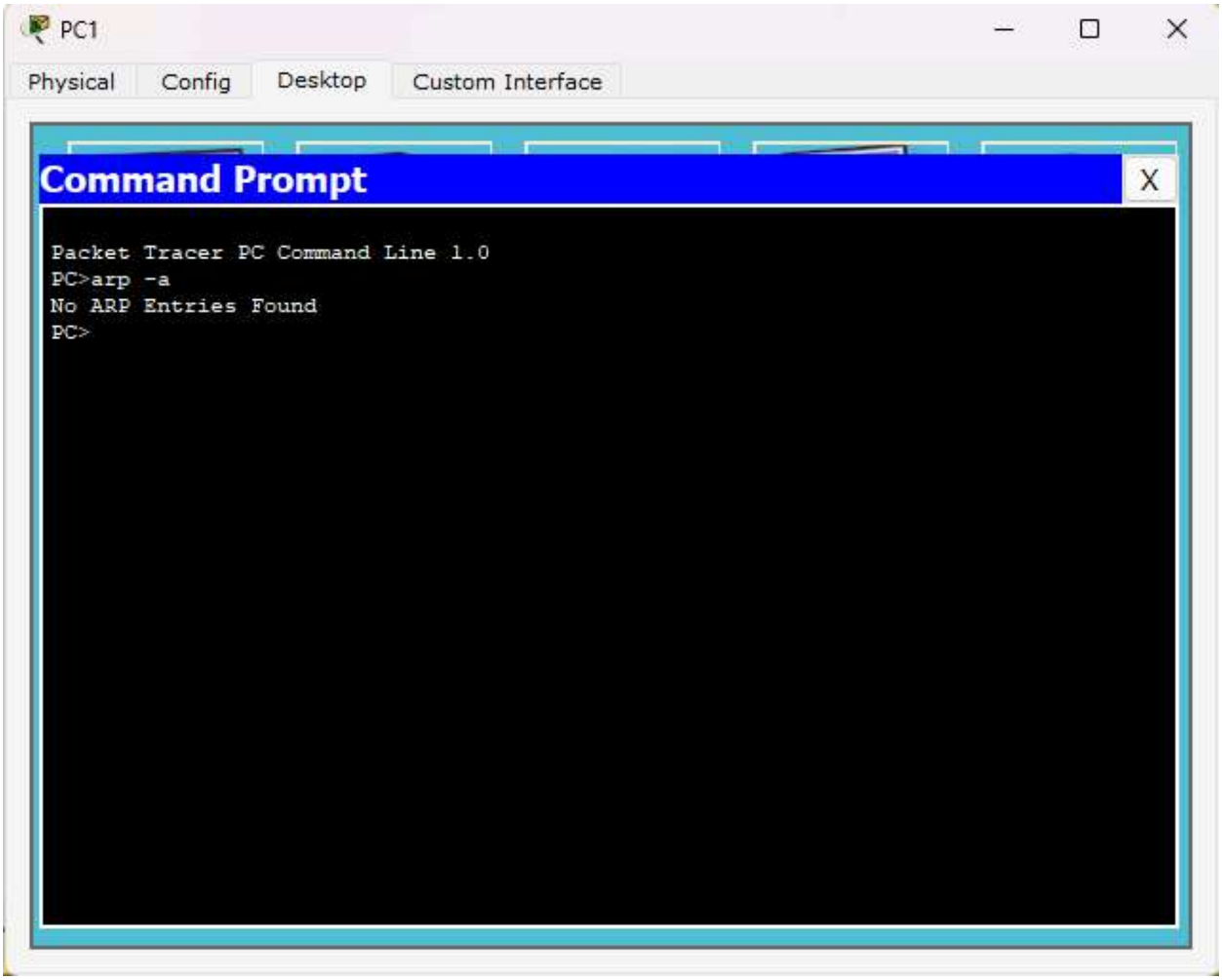
IP address	Hardware Address	Interface
10.0.0.3	0060.2F29.2CB8	FastEthernet 0

ARP table for PC2 (distination):

IP address	Hardware Address	Interface
10.0.0.1	00D0.D302.960B	FastEthernet0

Screen Shots:

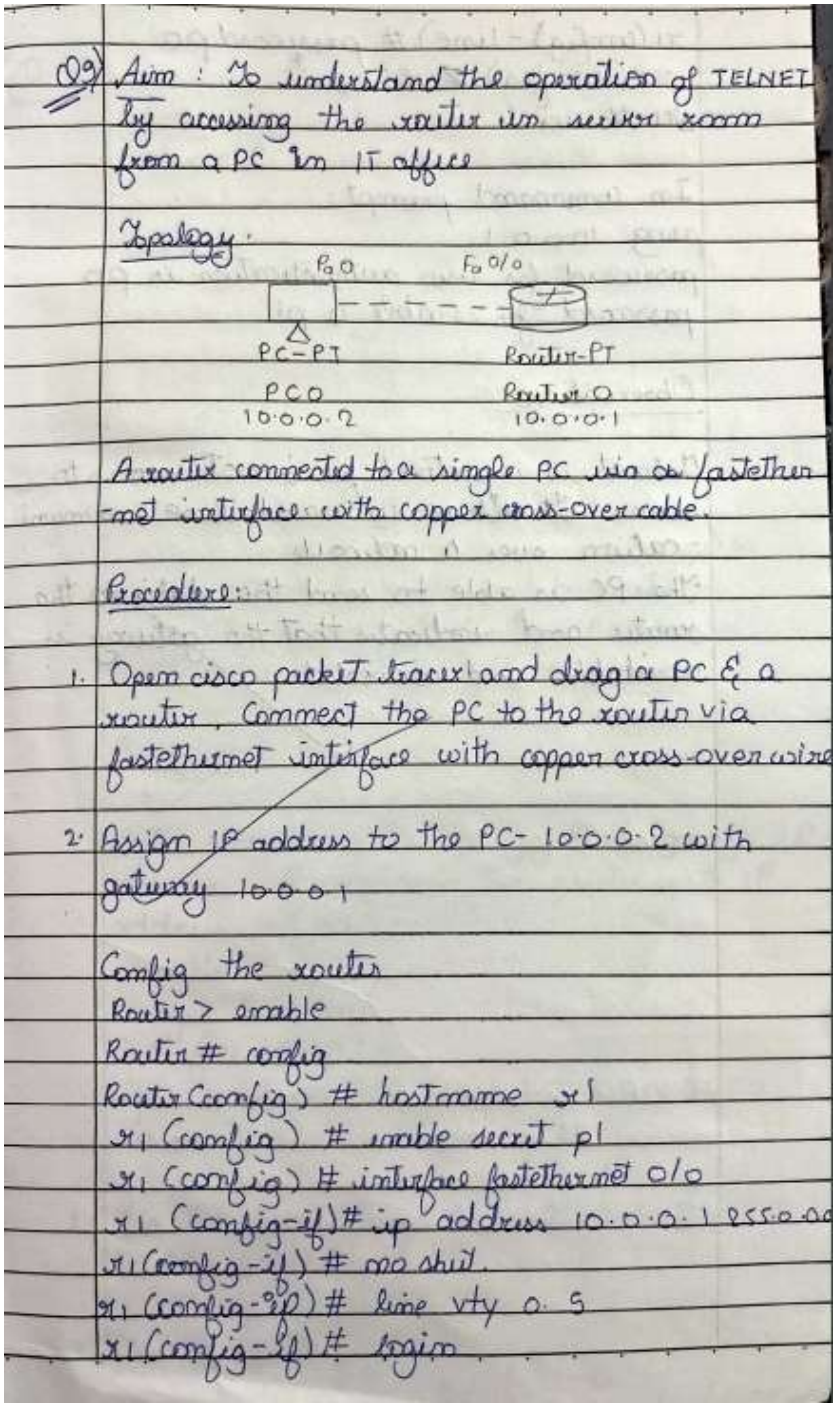


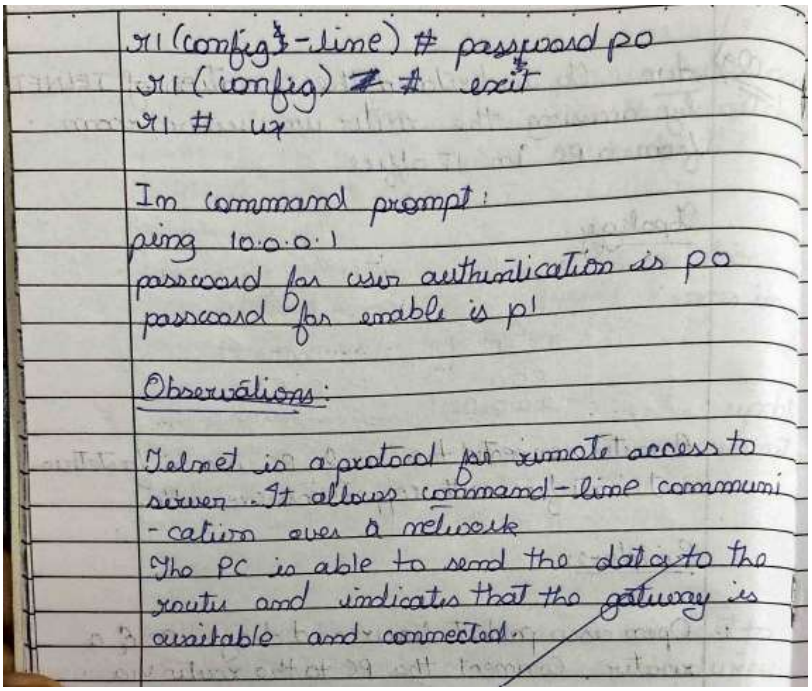


Program 11

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

Topology , Procedure and Observation:





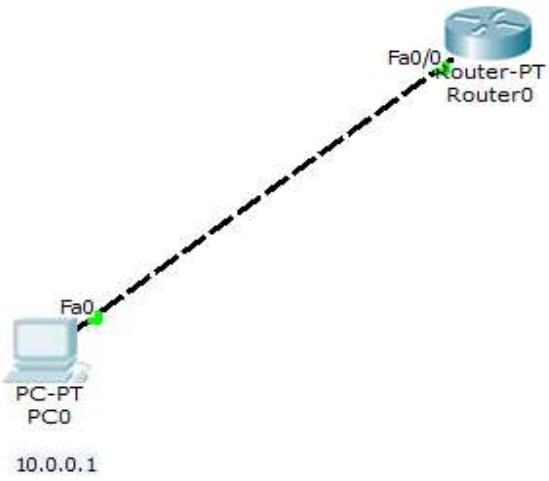
r1(config)-line) # password po
r1(config) # exit
r1 # ux

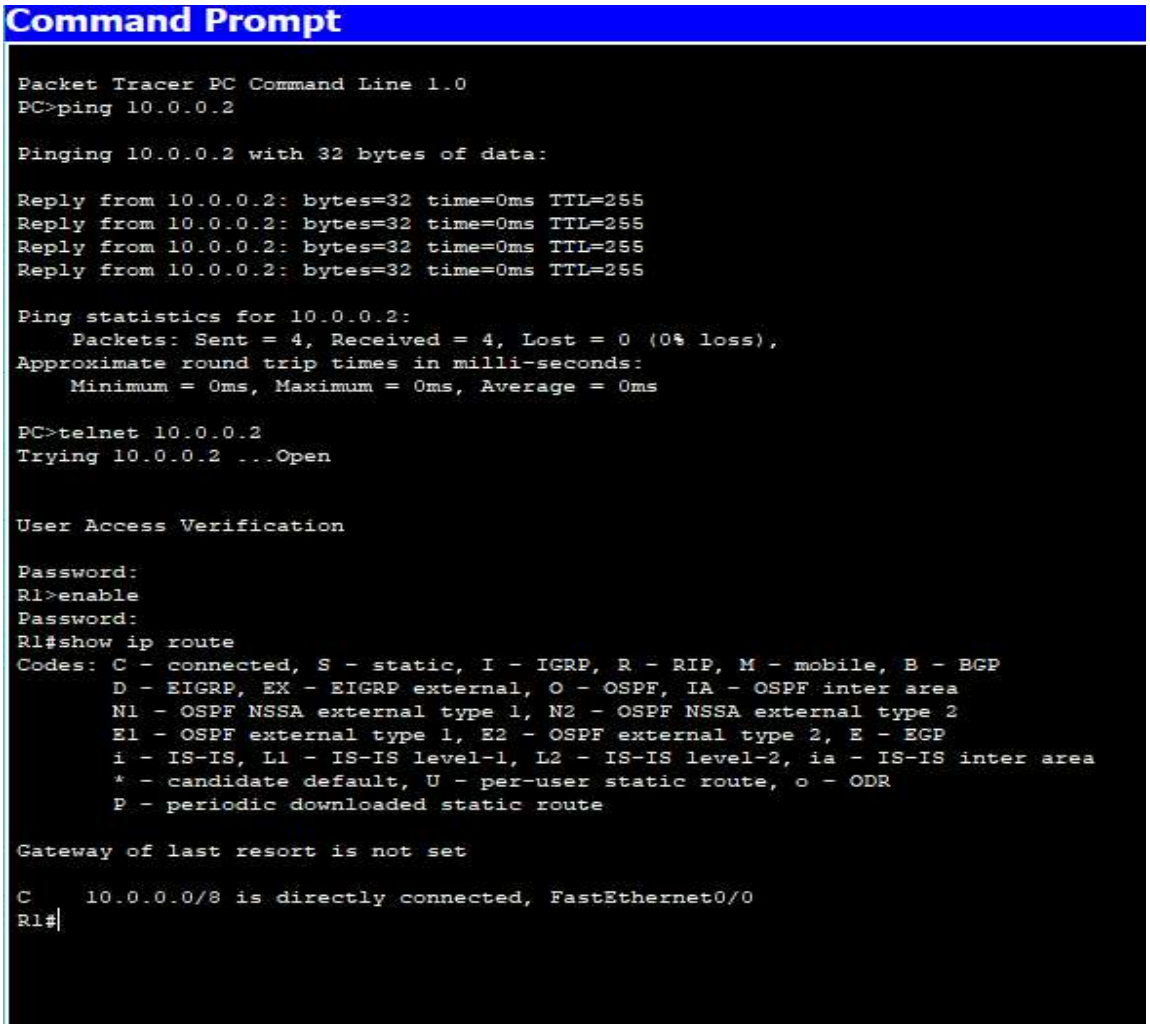
In command prompt:
ping 10.0.0.1
password for user authentication is po
password for enable is p1

Observations:

Telnet is a protocol for remote access to server. It allows command-line communication over a network
The PC is able to send the data to the router and indicates that the gateway is available and connected.

Screen Shots:

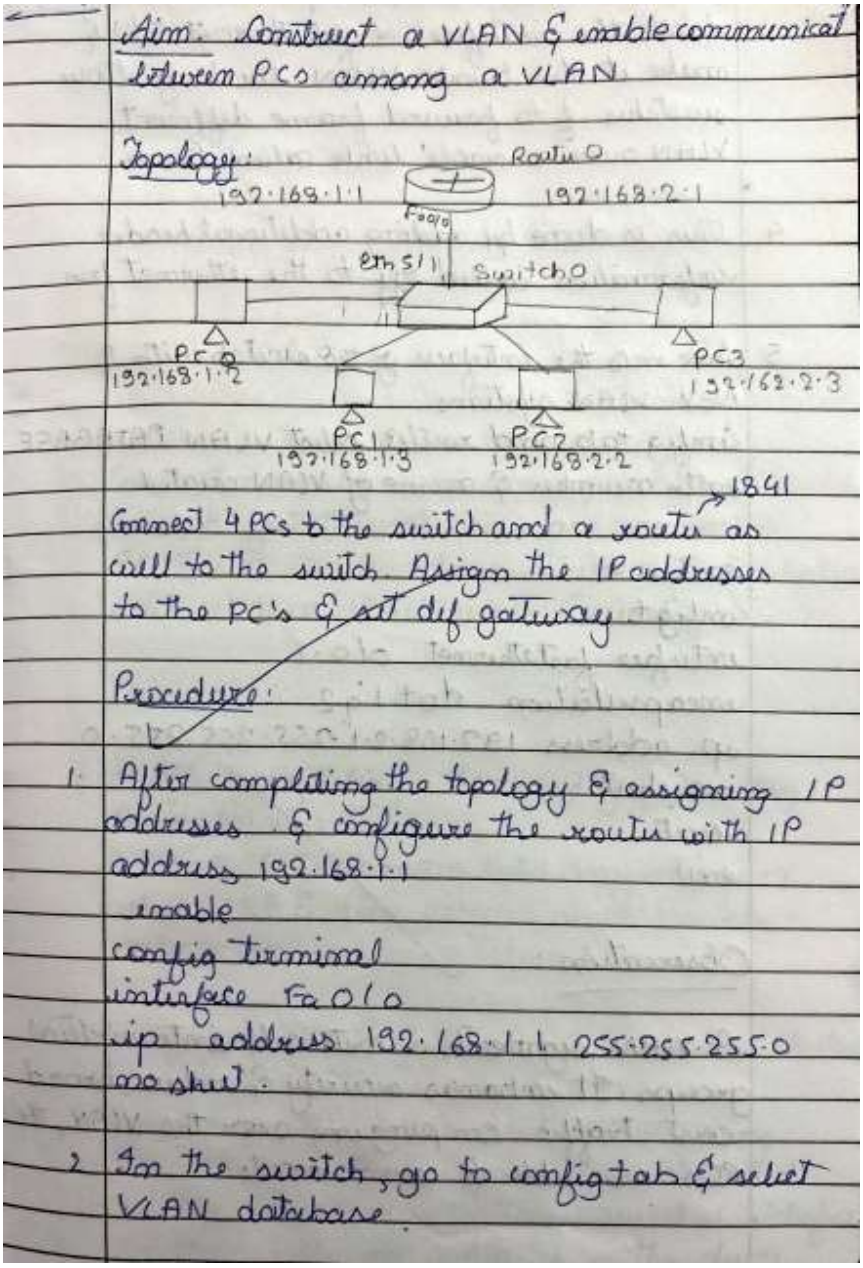


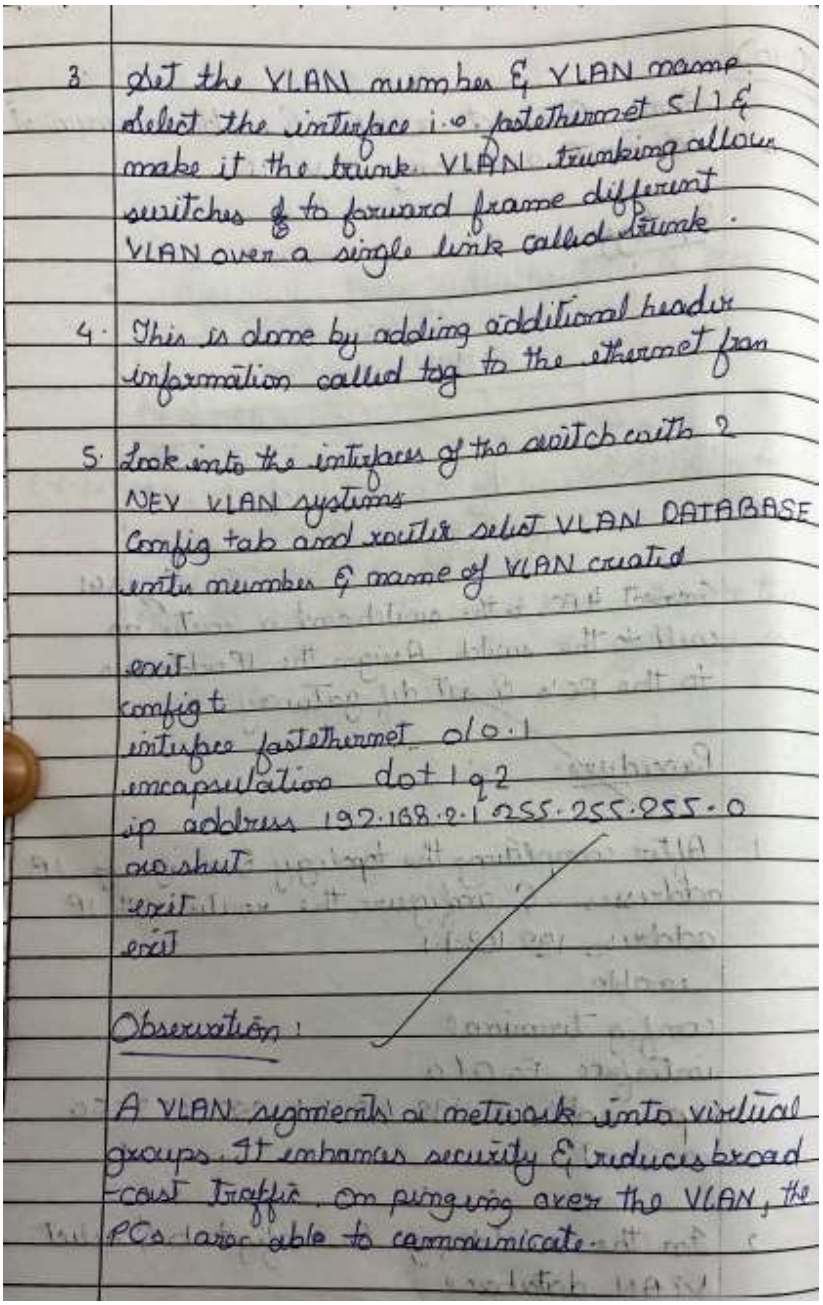

Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.2
Pinging 10.0.0.2 with 32 bytes of data:
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Reply from 10.0.0.2: bytes=32 time=0ms TTL=255
Ping statistics 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>telnet 10.0.0.2
Trying 10.0.0.2 ...Open
User Access Verification
Password:
R1>enable
Password:
R1#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 NSSA external type 2
E1 - OSPF external type 1, E2 - 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, FastEthernet0/0
R1#

Program 12

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

Topology , Procedure and Observation:





3. Set the VLAN number & VLAN name. Select the interface i.e. fastethernet 5/1 & make it the trunk. VLAN trunking allows switches & to forward frame different VLAN over a single link called trunk.

4. This is done by adding additional header information called tag to the ethernet frame.

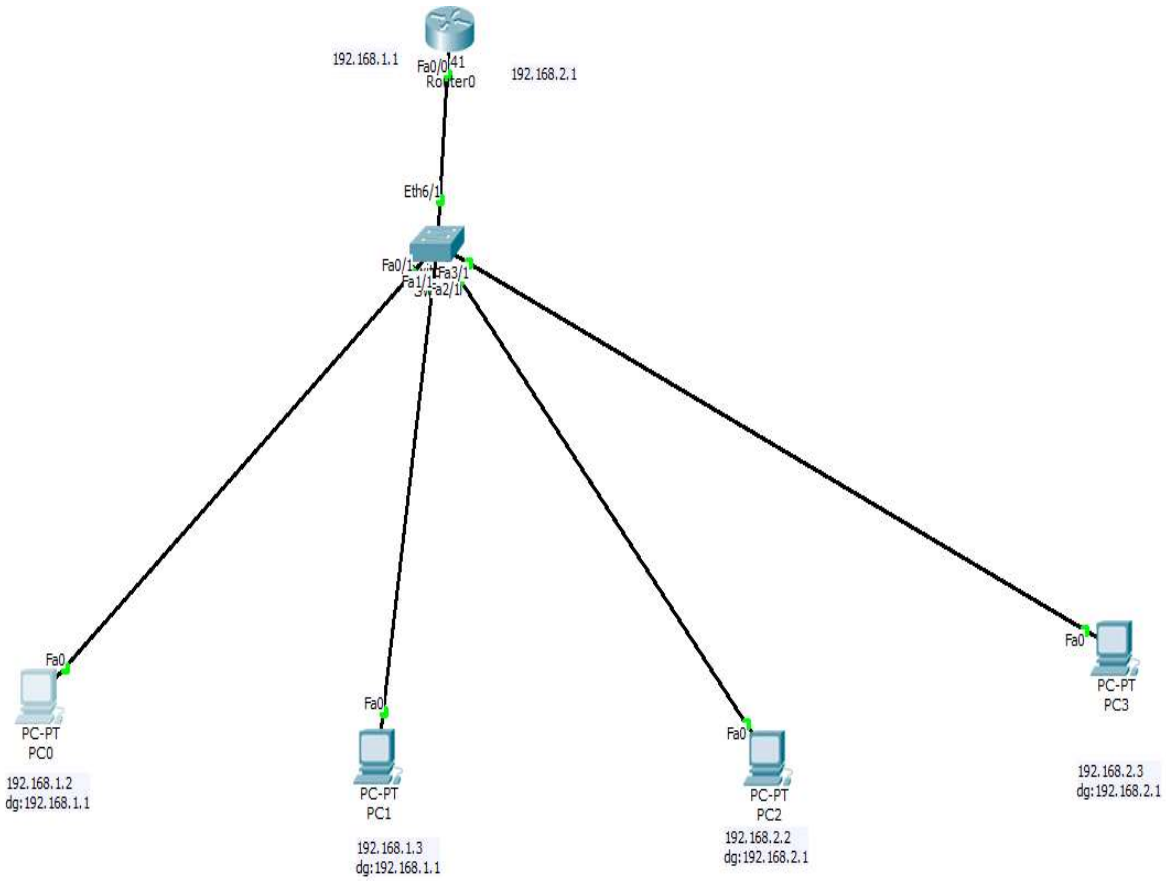
5. Look into the interfaces of the switch with 2 NEW VLAN systems
Config tab and router select VLAN DATABASE with number & name of VLAN created

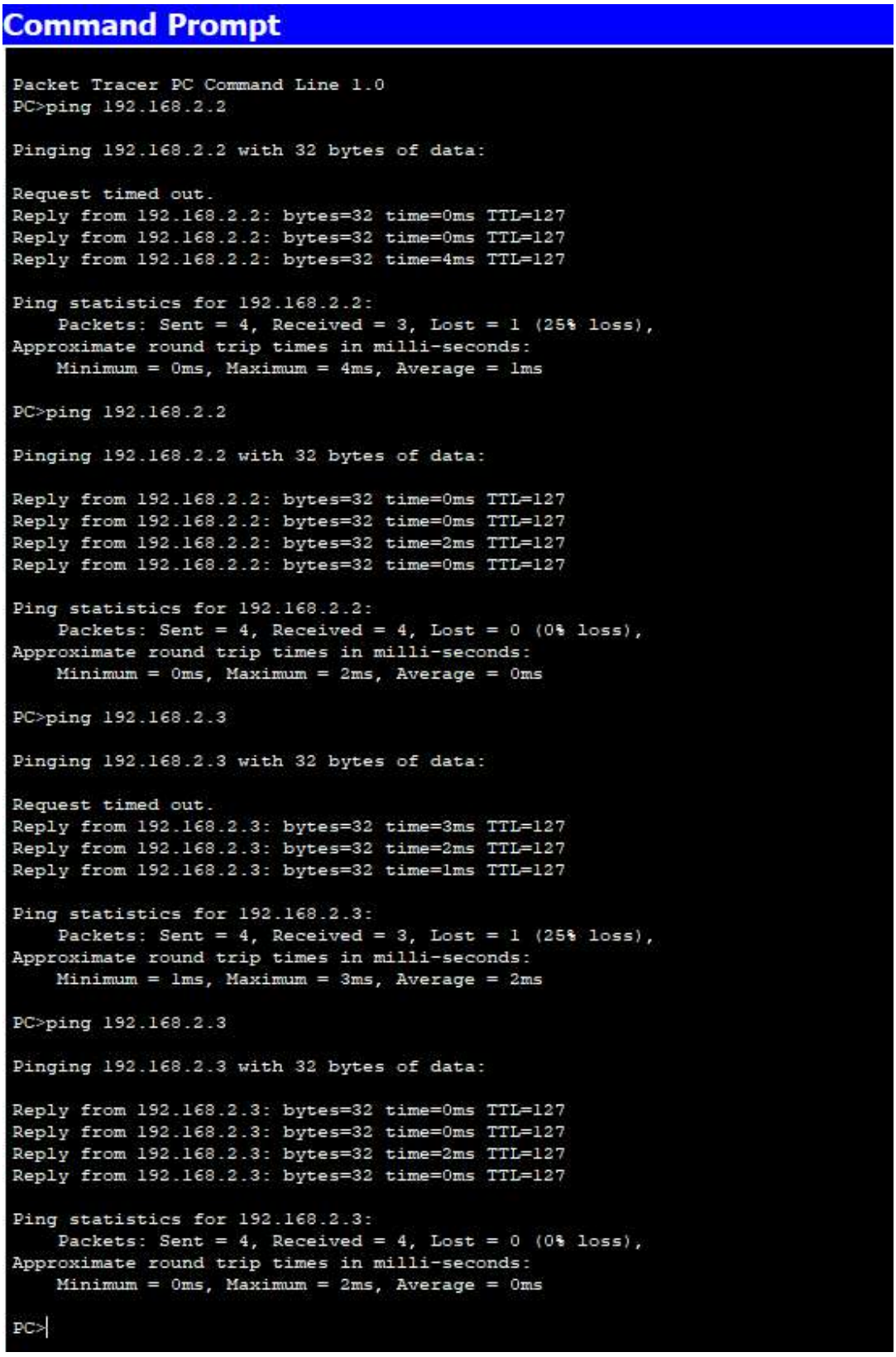
exit
config t
interface fastethernet 0/0.1
encapsulation dot1q2
ip address 192.168.2.1 255.255.255.0
no shut
exit
exit

Observation:

A VLAN segments a network into virtual groups. It enhances security & reduces broad cast traffic. On pinging over the VLAN, the PCs are able to communicate.

Screen Shots:

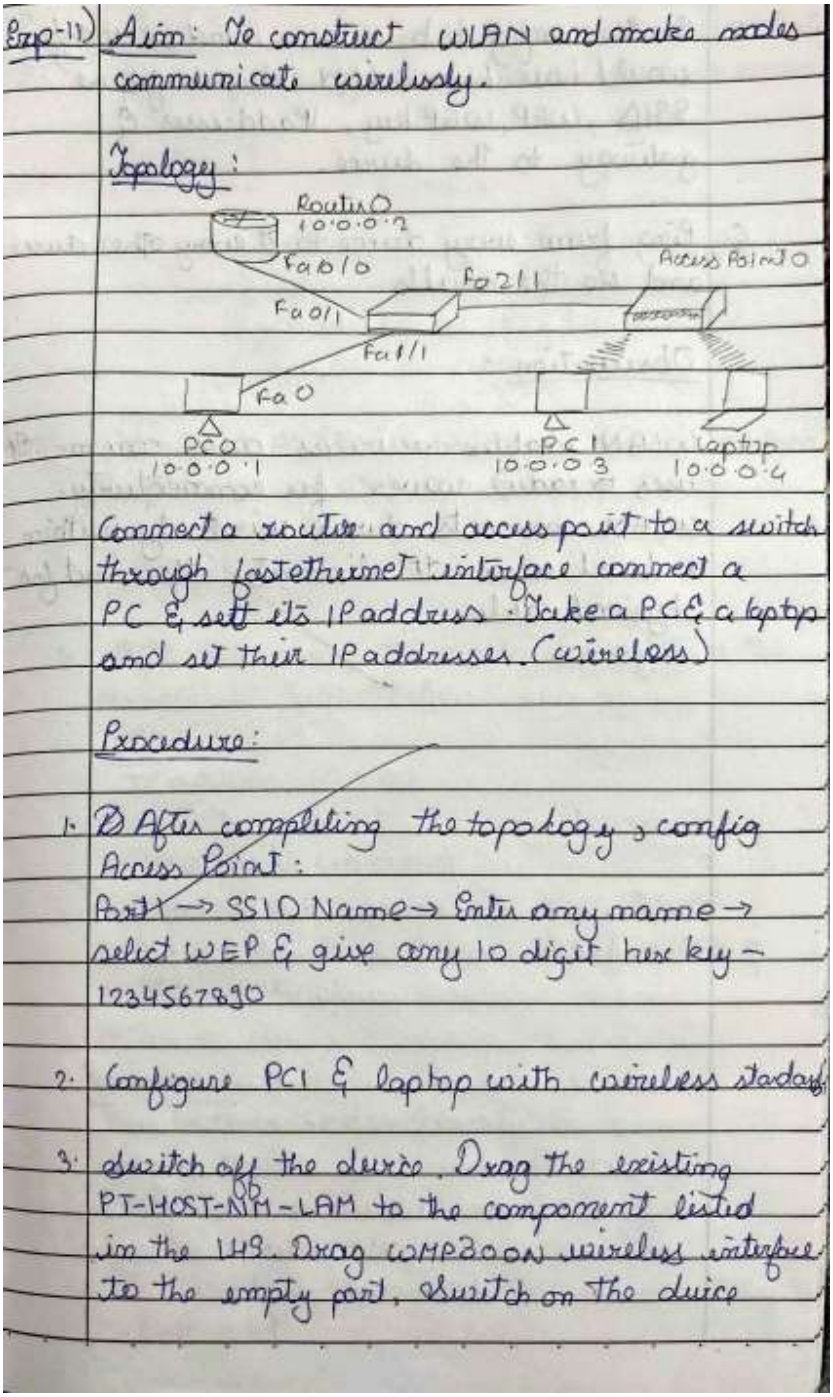



Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 192.168.2.2
Pinging 192.168.2.2 with 32 bytes of data:
Request timed out.
Reply from 192.168.2.2: bytes=32 time=0ms TTL=127
Reply from 192.168.2.2: bytes=32 time=0ms TTL=127
Reply from 192.168.2.2: bytes=32 time=4ms TTL=127
Ping statistics for 192.168.2.2:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 4ms, Average = 1ms
PC>ping 192.168.2.2
Pinging 192.168.2.2 with 32 bytes of data:
Reply from 192.168.2.2: bytes=32 time=0ms TTL=127
Reply from 192.168.2.2: bytes=32 time=0ms TTL=127
Reply from 192.168.2.2: bytes=32 time=2ms TTL=127
Reply from 192.168.2.2: bytes=32 time=0ms TTL=127
Ping statistics for 192.168.2.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 2ms, Average = 0ms
PC>ping 192.168.2.3
Pinging 192.168.2.3 with 32 bytes of data:
Request timed out.
Reply from 192.168.2.3: bytes=32 time=3ms TTL=127
Reply from 192.168.2.3: bytes=32 time=2ms TTL=127
Reply from 192.168.2.3: bytes=32 time=1ms TTL=127
Ping statistics for 192.168.2.3:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 1ms, Maximum = 3ms, Average = 2ms
PC>ping 192.168.2.3
Pinging 192.168.2.3 with 32 bytes of data:
Reply from 192.168.2.3: bytes=32 time=0ms TTL=127
Reply from 192.168.2.3: bytes=32 time=0ms TTL=127
Reply from 192.168.2.3: bytes=32 time=2ms TTL=127
Reply from 192.168.2.3: bytes=32 time=0ms TTL=127
Ping statistics for 192.168.2.3:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 2ms, Average = 0ms
PC>

Program 13

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

Topology , Procedure and Observation:



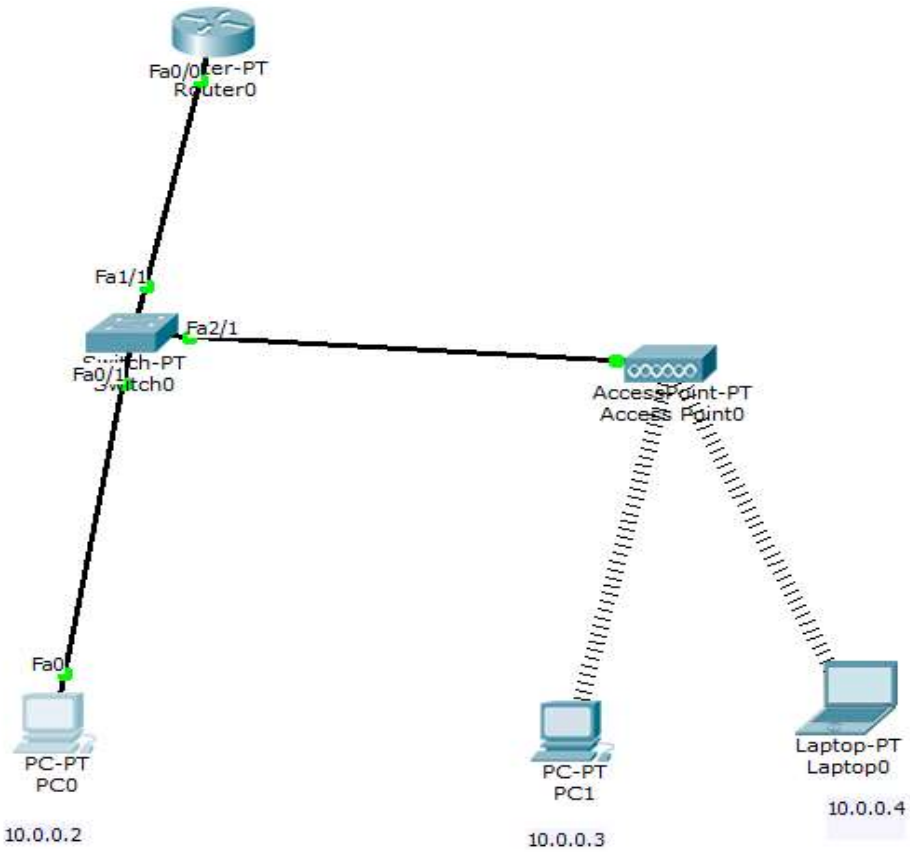
4. In the config tab, a new wireless interface would have been added. Now configure SSID, WEP, WEP key, IP addresses & gateway to the device.

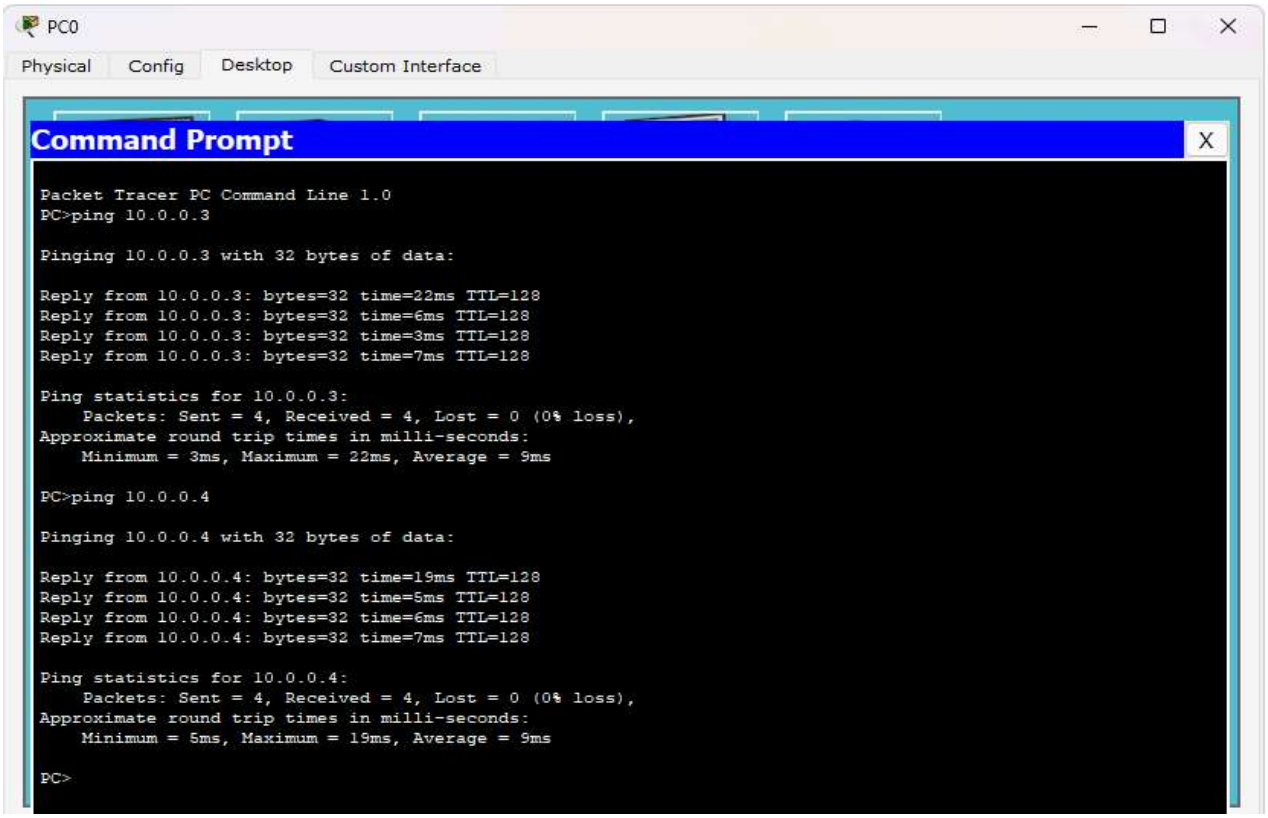
5. Ping from every device to every other device and see the results.

Observation :

WLAN enables wireless n/w comm. It uses radio waves for connectivity. WLAN connects devices wirelessly within a local area. It eliminates the need for physical cables.

Screen Shots:

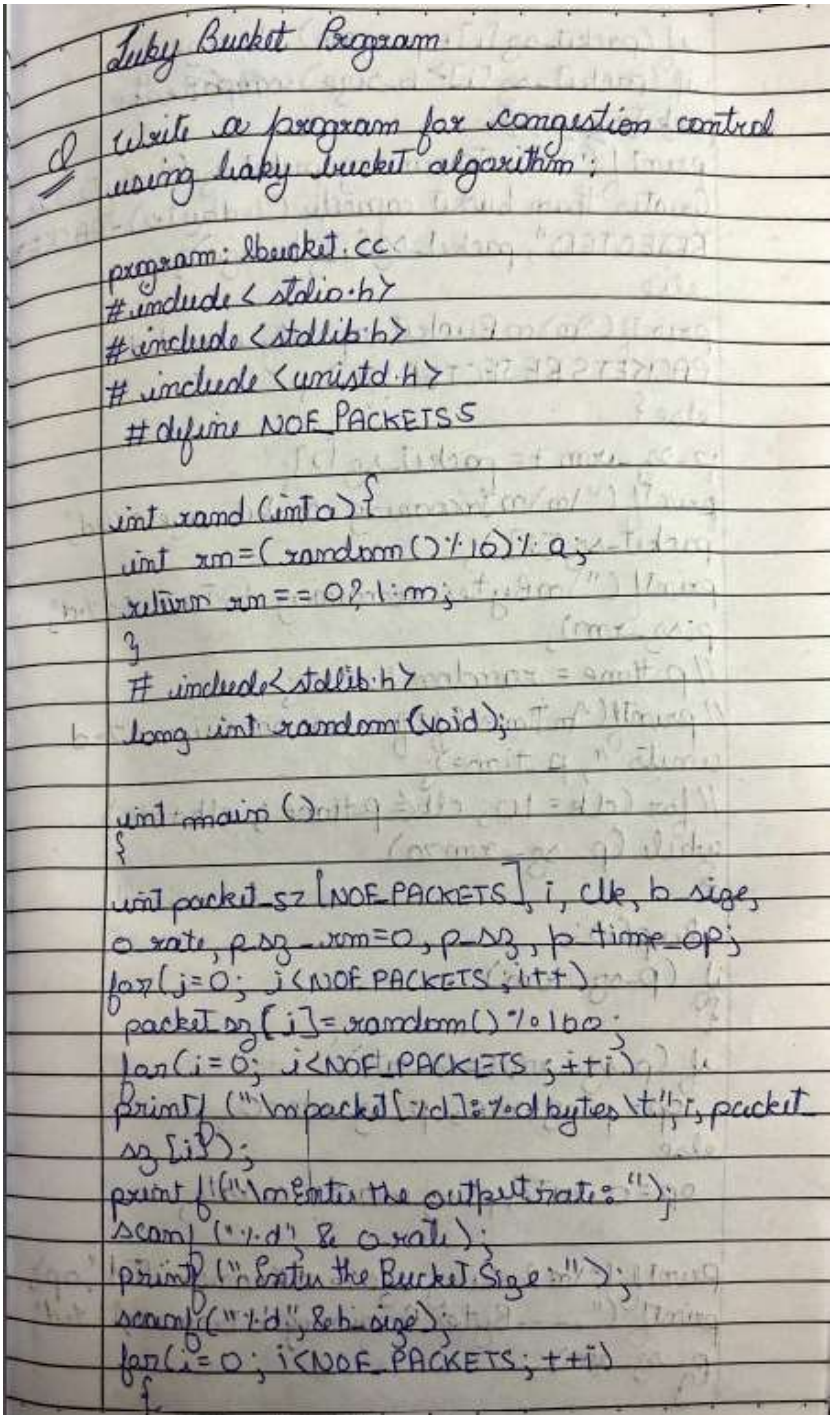



PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.3
Pinging 10.0.0.3 with 32 bytes of data:
Reply from 10.0.0.3: bytes=32 time=22ms TTL=128
Reply from 10.0.0.3: bytes=32 time=6ms TTL=128
Reply from 10.0.0.3: bytes=32 time=3ms TTL=128
Reply from 10.0.0.3: bytes=32 time=7ms TTL=128
Ping statistics for 10.0.0.3:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 3ms, Maximum = 22ms, Average = 9ms
PC>ping 10.0.0.4
Pinging 10.0.0.4 with 32 bytes of data:
Reply from 10.0.0.4: bytes=32 time=19ms TTL=128
Reply from 10.0.0.4: bytes=32 time=5ms TTL=128
Reply from 10.0.0.4: bytes=32 time=6ms TTL=128
Reply from 10.0.0.4: bytes=32 time=7ms 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 = 5ms, Maximum = 19ms, Average = 9ms
PC>

PART-B

Program 14

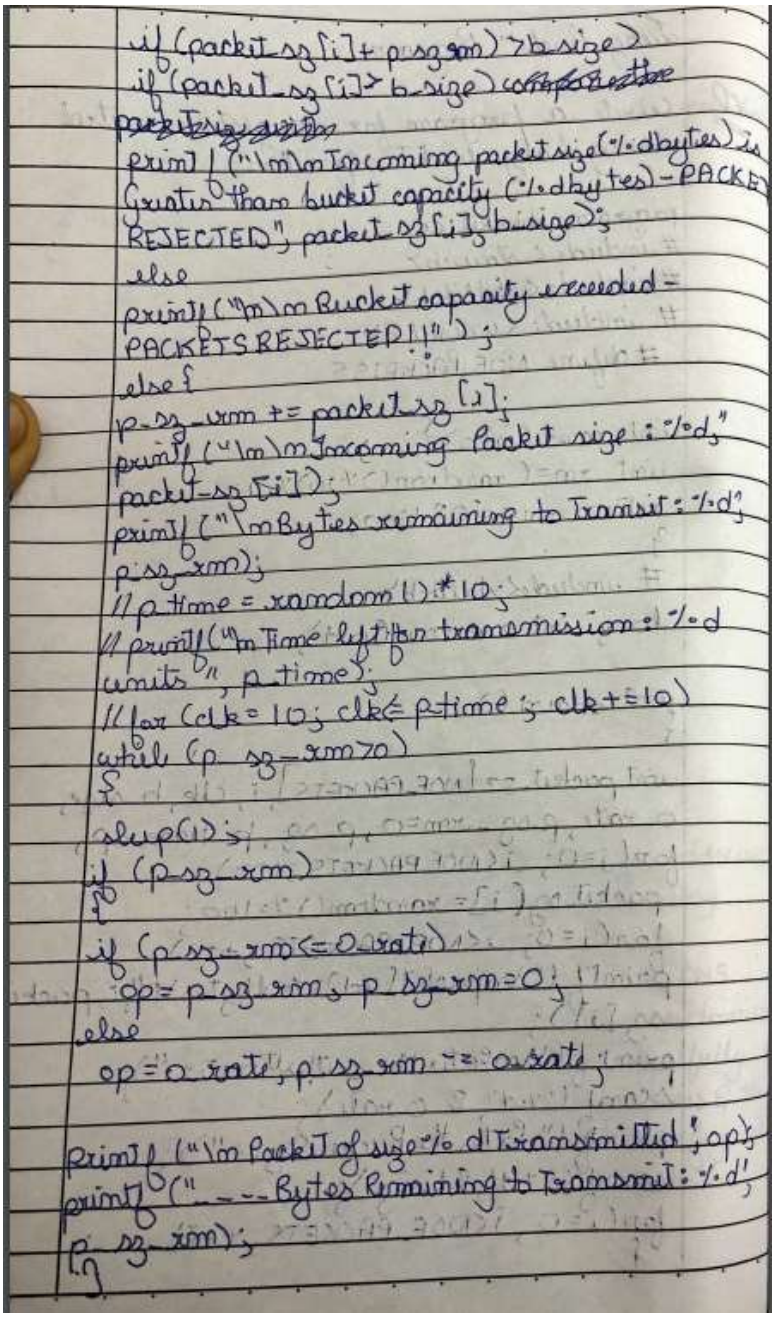
Write a program for congestion control using Leaky bucket algorithm.



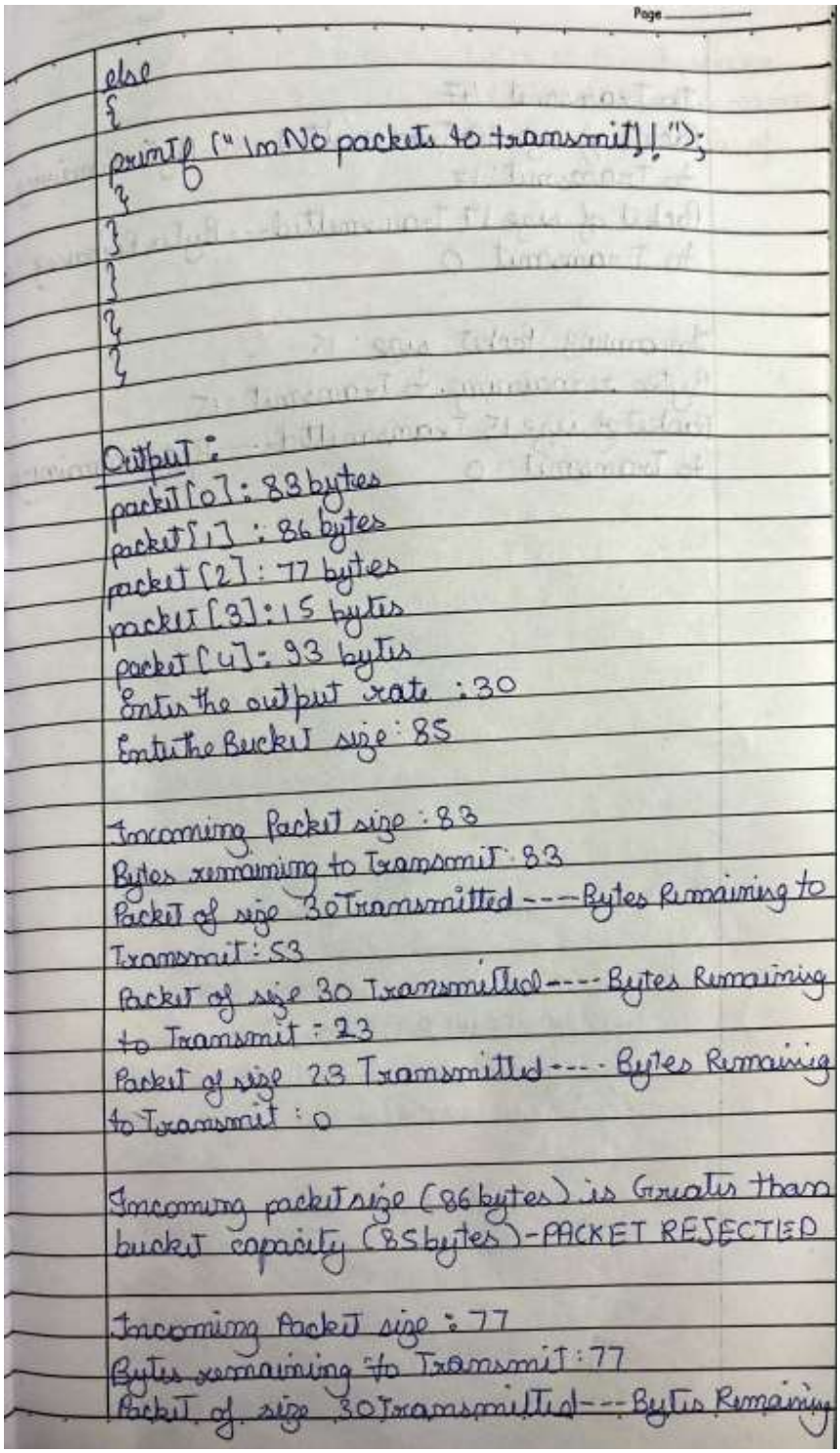
Leaky Bucket Program

Q) Write a program for congestion control using leaky bucket algorithm;

```
program: lbucket.cc
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define NOF_PACKETS 5

int rand(int a){
int rm=(random()%10)%a;
return rm==0?1:rm;
}
#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_time_op;
for(j=0; j<NOF_PACKETS;j++)
packet_sz[j]=random()%100;
for(i=0; i<NOF_PACKETS;++i)
printf("\npacket[%d]:%d bytes\t",i,packet_
sz[i]);
printf("\nEnter the output rate:");
scanf("%d",&o_rate);
printf("\nEnter the Bucket Size:");
scanf("%d",&b_size);
for(i=0; i<NOF_PACKETS;++i)
{
```



```
if (packet_sz[i]+ p_sz_rem) > b_size)
if (packet_sz[i] > b_size) compare the
packet size with
printf ("\n\nIncoming packet size(%d bytes) is
Greater than bucket capacity (%d bytes) - PACKET
REJECTED", packet_sz[i], b_size);
else
printf("\n\nBucket capacity exceeded -
PACKETS REJECTED!!");
else {
p_sz_rem += packet_sz[i];
printf("\n\nIncoming Packet size : %d",
packet_sz[i]);
printf("\nBytes remaining to Transmit : %d",
p_sz_rem);
// p_time = random(1) * 10;
// printf("\nTime 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)
op = p_sz_rem; p_sz_rem = 0;
else
op = o_rate; p_sz_rem -= o_rate;

printf("\nPacket of size %d Transmitted", op);
printf("\n---Bytes Remaining to Transmit : %d",
p_sz_rem);
}
```



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

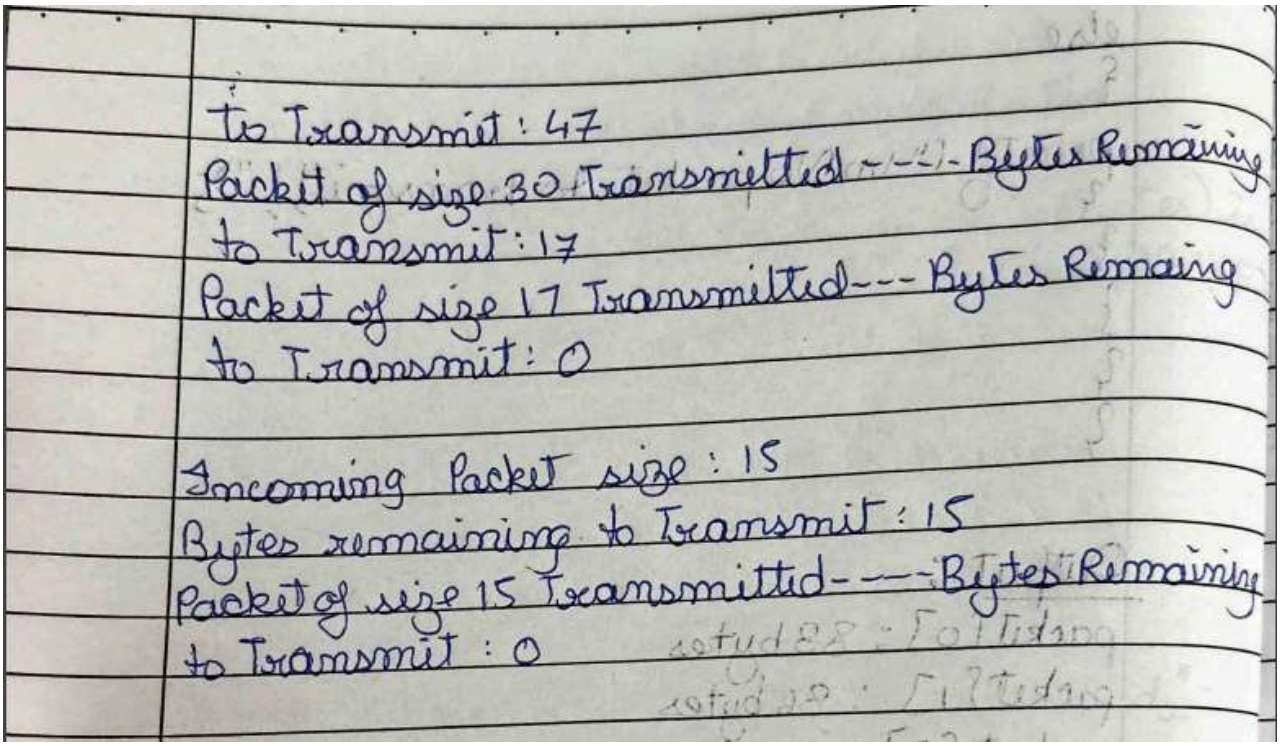
Output:

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

Incoming Packet size: 83
Bytes remaining to Transmit: 83
Packet of size 30 Transmitted --- Bytes Remaining to Transmit: 53
Packet of size 30 Transmitted --- Bytes Remaining to Transmit: 23
Packet of size 23 Transmitted --- Bytes Remaining to Transmit: 0

Incoming packet size (86 bytes) is Greater than bucket capacity (85 bytes) - PACKET REJECTED

Incoming Packet size: 77
Bytes remaining to Transmit: 77
Packet of size 30 Transmitted --- Bytes Remaining



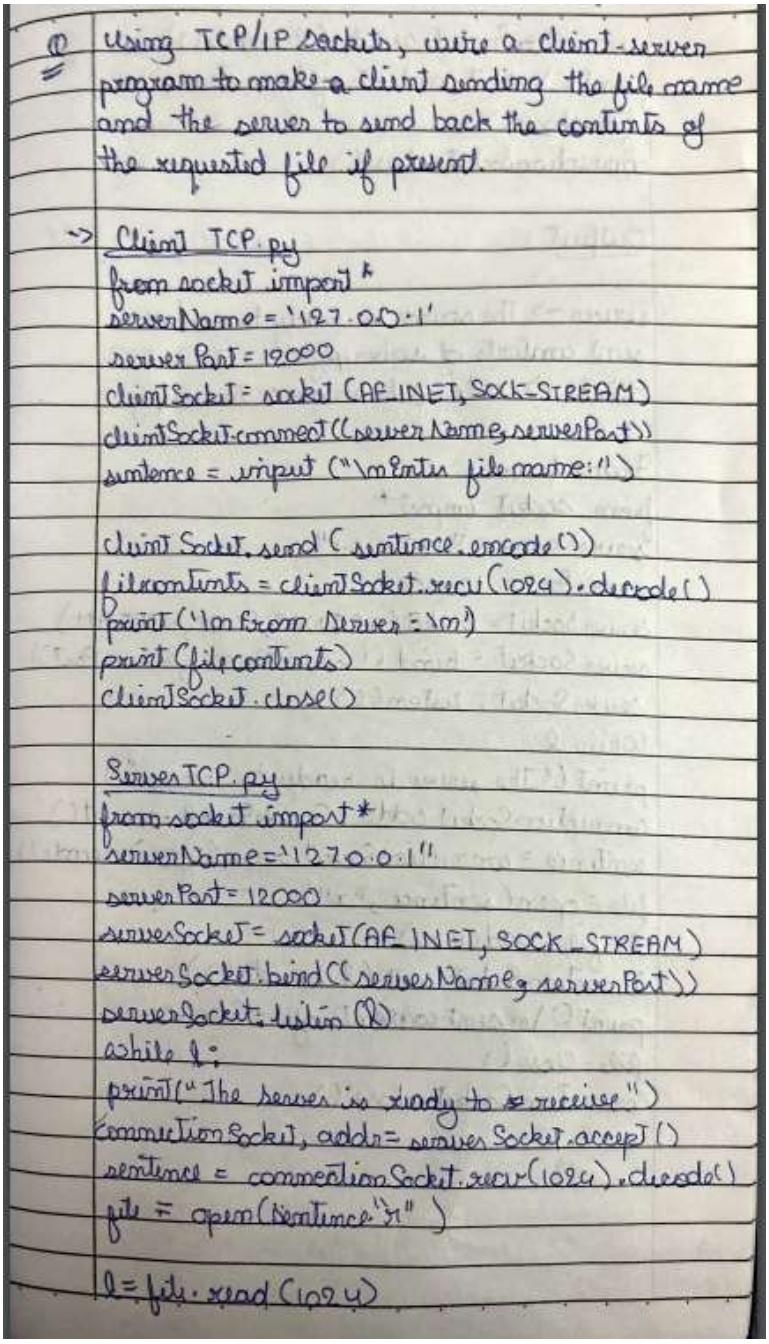
Output Clear

```
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 Remaining to Transmit: 53
Packet of size 30 Transmitted---Bytes Remaining to Transmit: 23
Packet of size 23 Transmitted---Bytes Remaining to Transmit: 0

Incoming Packet size: 86
Incoming packet size (86bytes) is Greater than bucket capacity (85bytes)-PACKET REJECTED

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 Remaining to Transmit: 17
Packet of size 17 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
Incoming packet size (93bytes) is Greater than bucket capacity (85bytes)-PACKET REJECTED
```

Program 15

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

Code and Output:



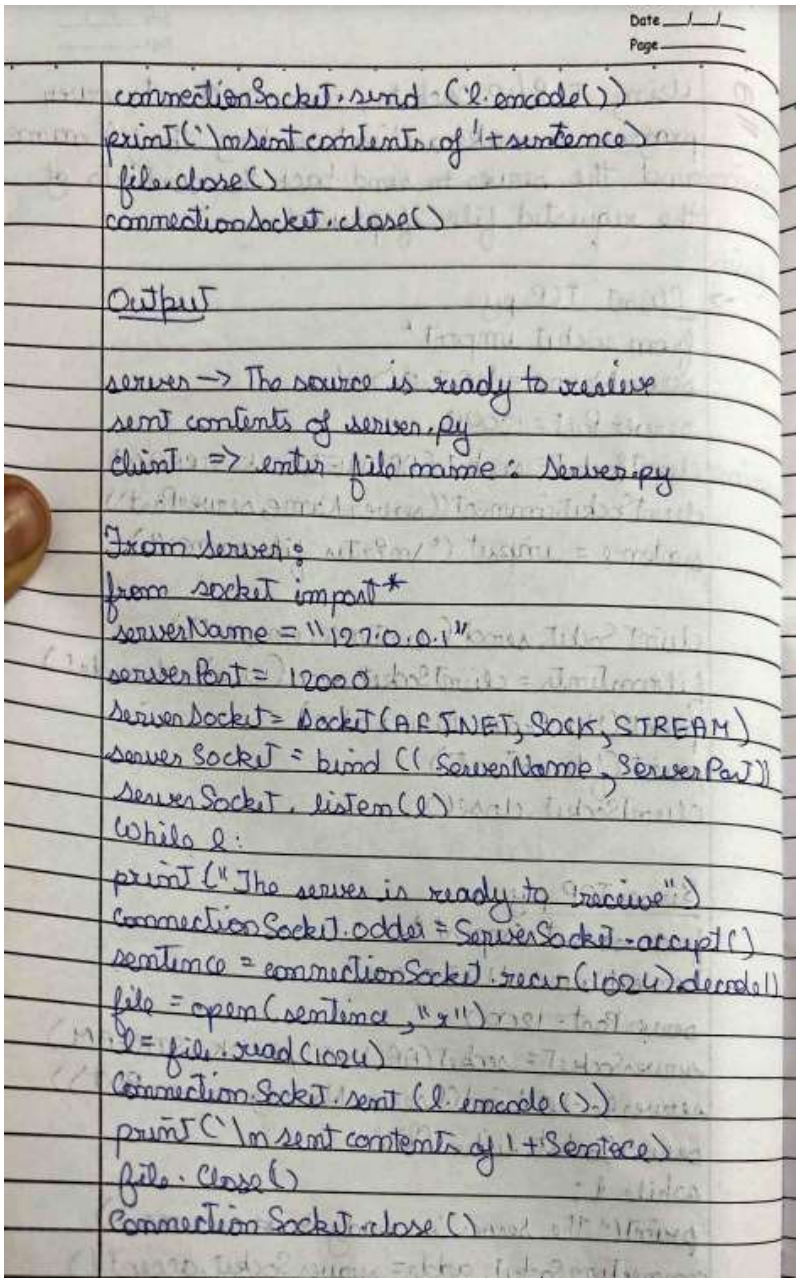
Using TCP/IP sockets, write a client-server program to make a client sending the file name and the server to send back the contents of the requested file if present.

-> Client TCP.py

```
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket (AF_INET, SOCK_STREAM)
clientSocket.connect ((serverName, serverPort))
sentence = input ("\nEnter file name:")

clientSocket.send (sentence.encode())
fileContents = clientSocket.recv(1024).decode()
print ("\nFrom Server :\n")
print (fileContents)
clientSocket.close()
```

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 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()
```

Output

server -> The source 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()
```

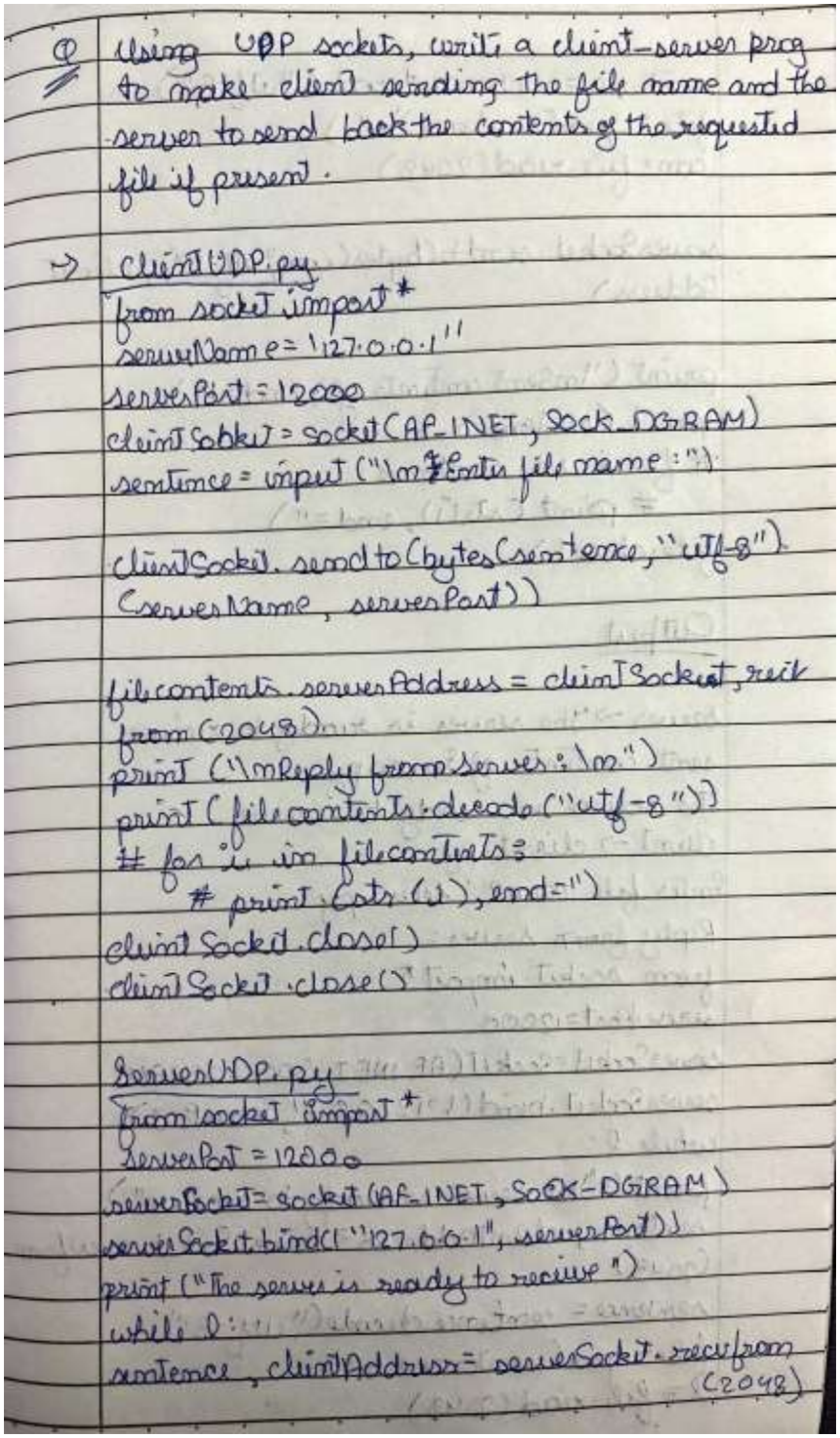
```
PS C:\Users\I AM HP\CN> python ClientTCP.py

 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 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()
PS C:\Users\I AM HP\CN>
```

```
DEBUG CONSOLE   PROBLEMS   TERMINAL   PORTS

PS C:\Users\I AM HP\CN> python ServerTCP.py
The server is ready to receive

 Sent contents ofServerTCP.py
The server is ready to receive
```

Program 16

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

Code and Output:



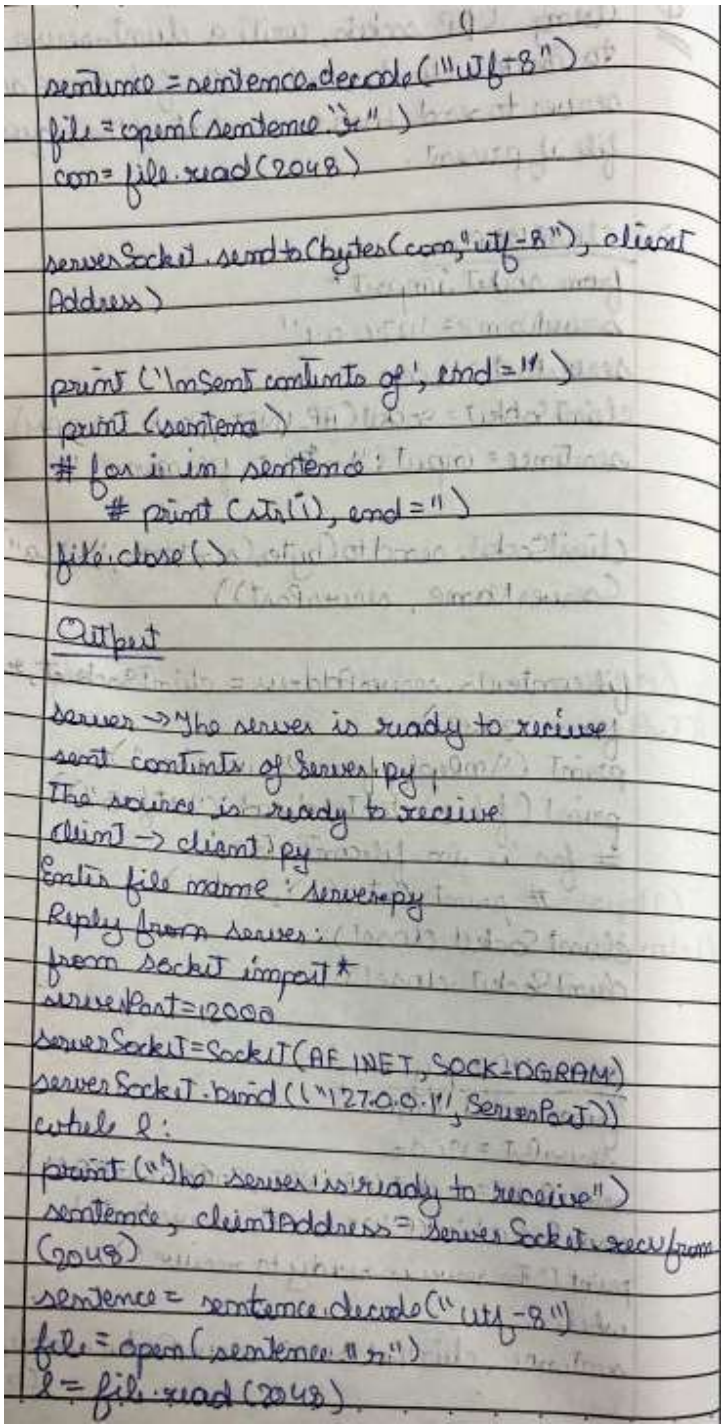
Using UDP sockets, write a client-server prog to make client sending the file name and the server to send back the contents of the requested file if present.

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

ServerUDP.py

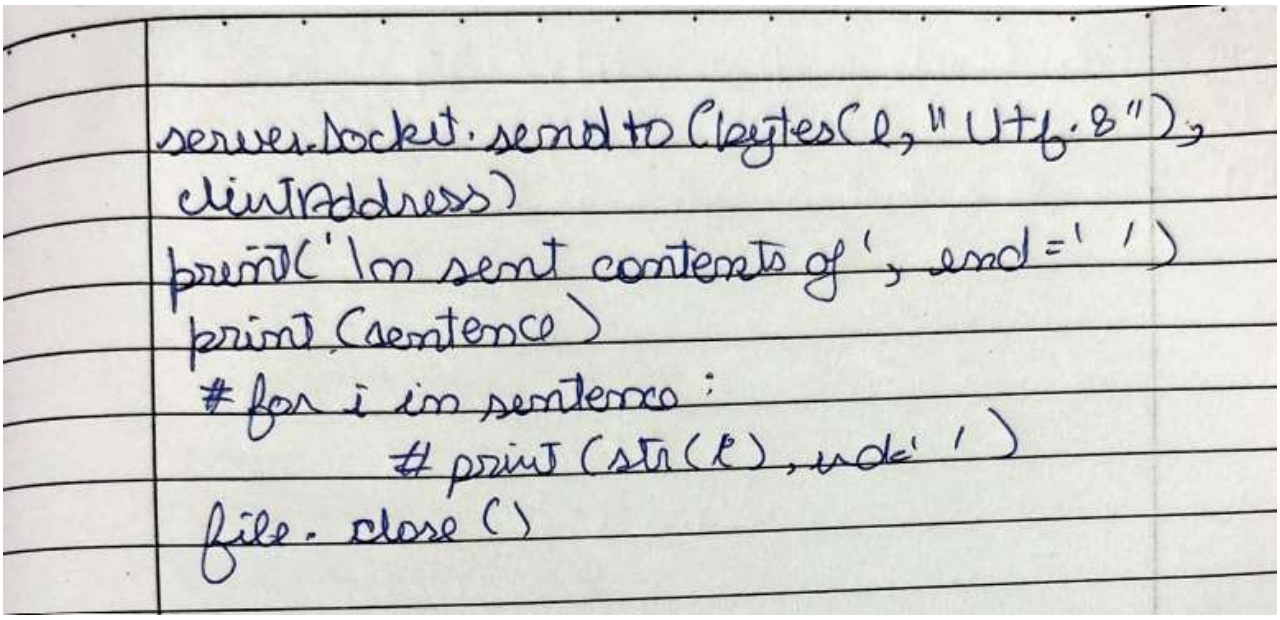
```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print ("The server is ready to receive")
while 1:
sentence, clientAddress = serverSocket.recvfrom
(2048)
```



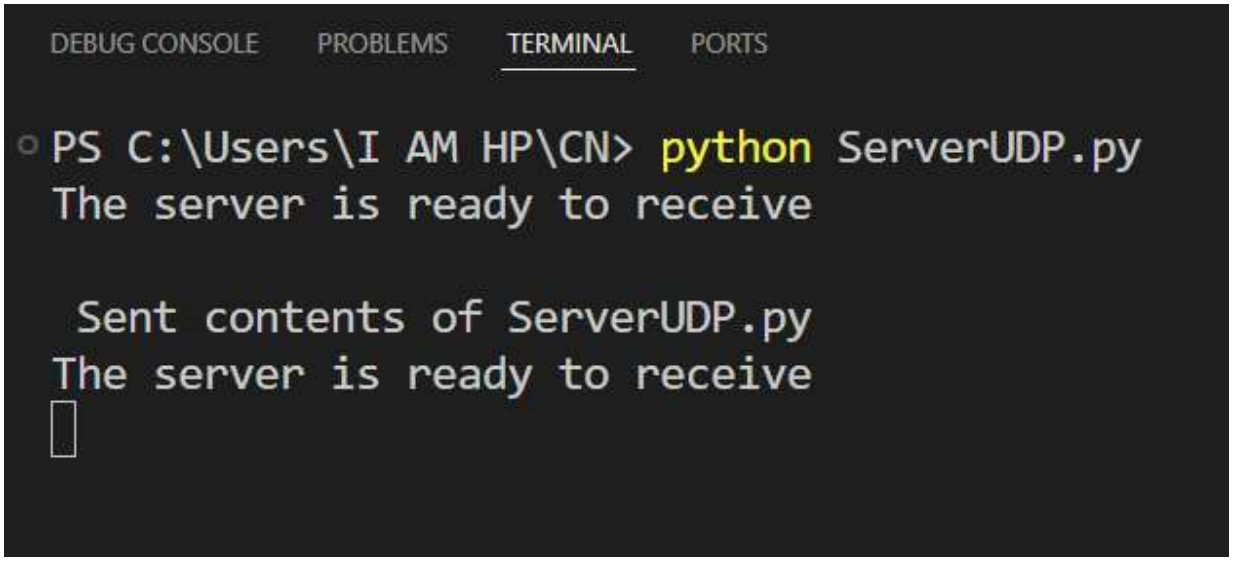
```
sentence = sentence.decode("utf-8")
file = open(sentence,"r")
con = file.read(2048)

serverSocket.sendto(bytes(con,"utf-8"), clientAddress)

print ('\nSent 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 source 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()
```

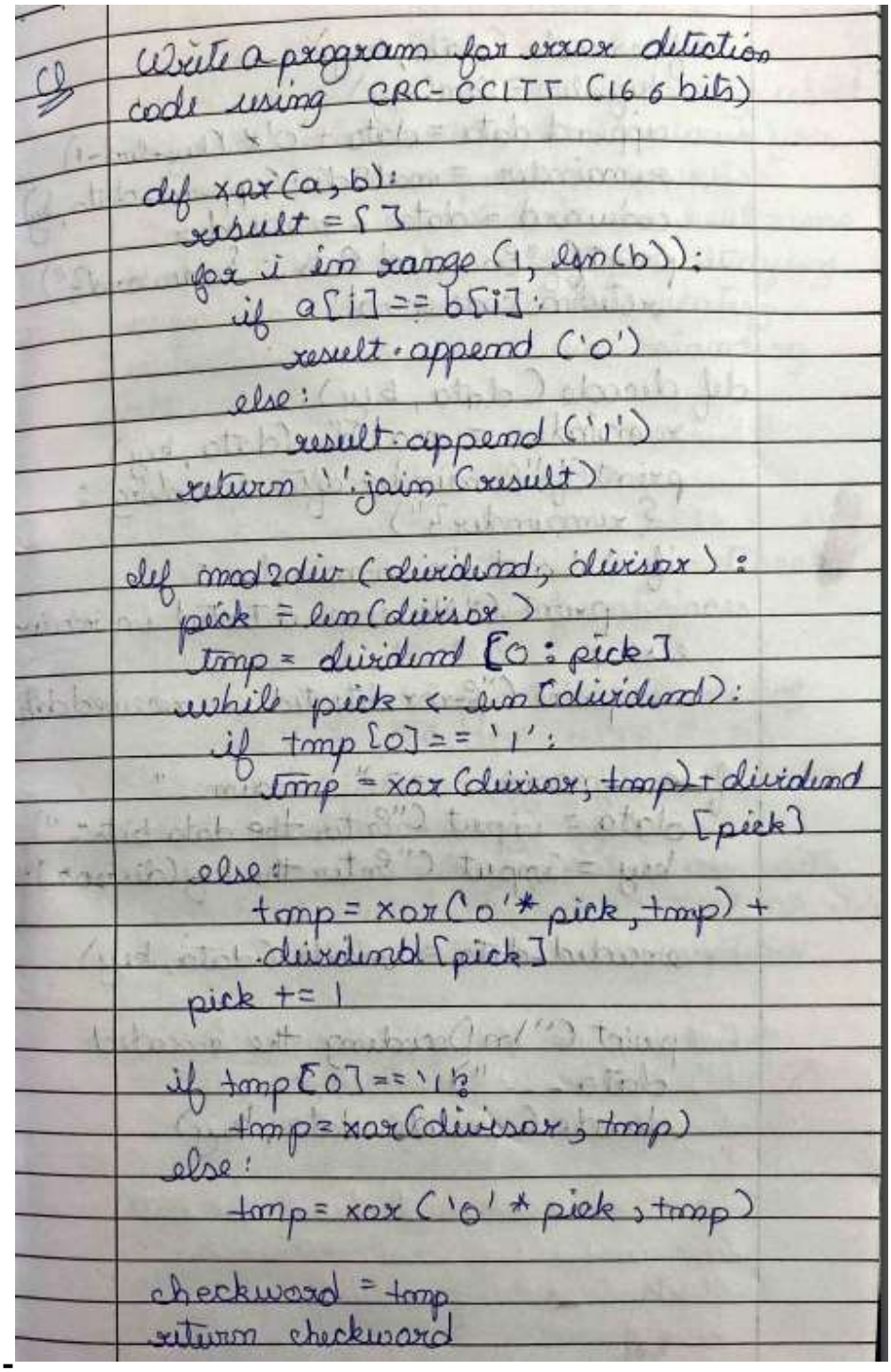


```
DEBUG CONSOLE  PROBLEMS  TERMINAL  PORTS

PS C:\Users\I AM HP\CN> python ServerUDP.py
The server is ready to receive

 Sent contents of ServerUDP.py
The server is ready to receive
```

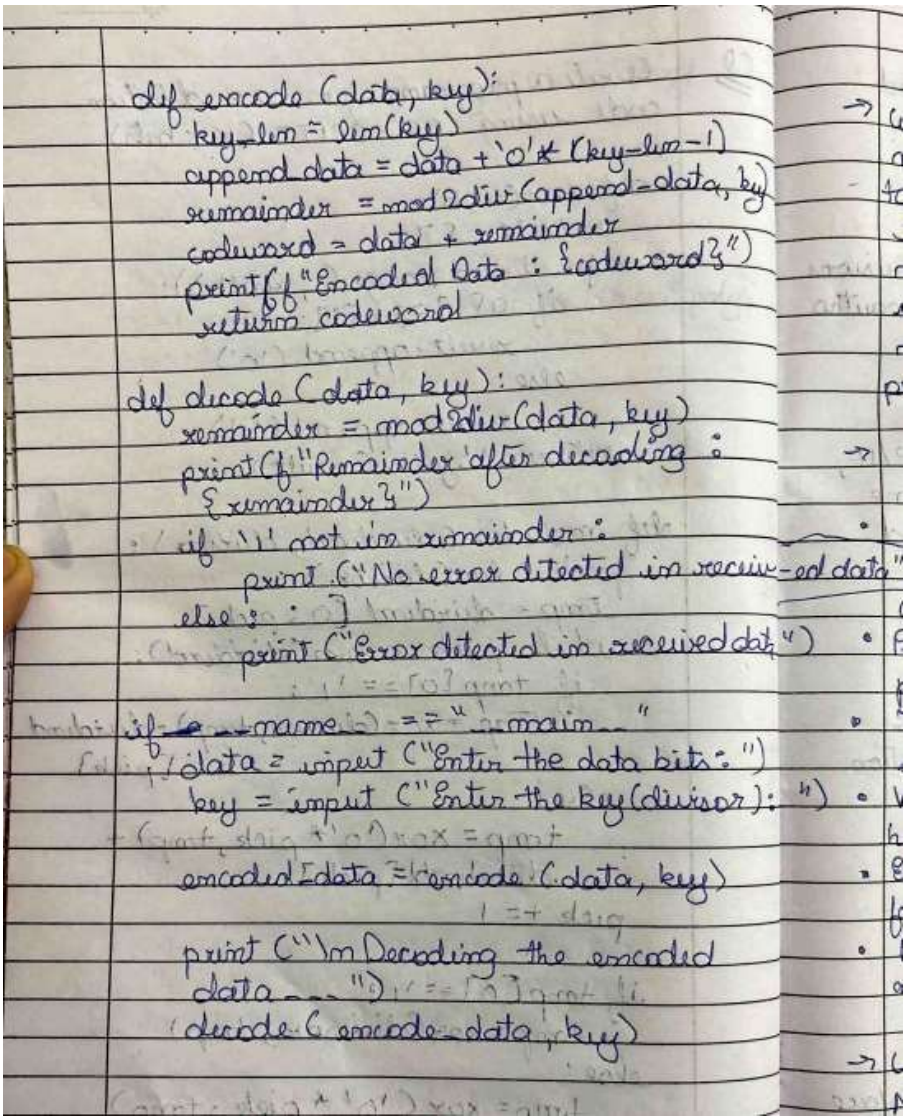
```
PS C:\Users\I AM HP\CN> python ClientUDP.py

 Enter File Name:ServerUDP.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")
    con=file.read(2048)
    serverSocket.sendto(bytes(con,"utf-8"),clientAddre
ss)
    print("\n Sent contents of "+sentence)
    file.close()
PS C:\Users\I AM HP\CN>
```

Program 17:

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



```
Q) Write a program for error detection
code using CRC-CCITT (16 & bits)

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
```



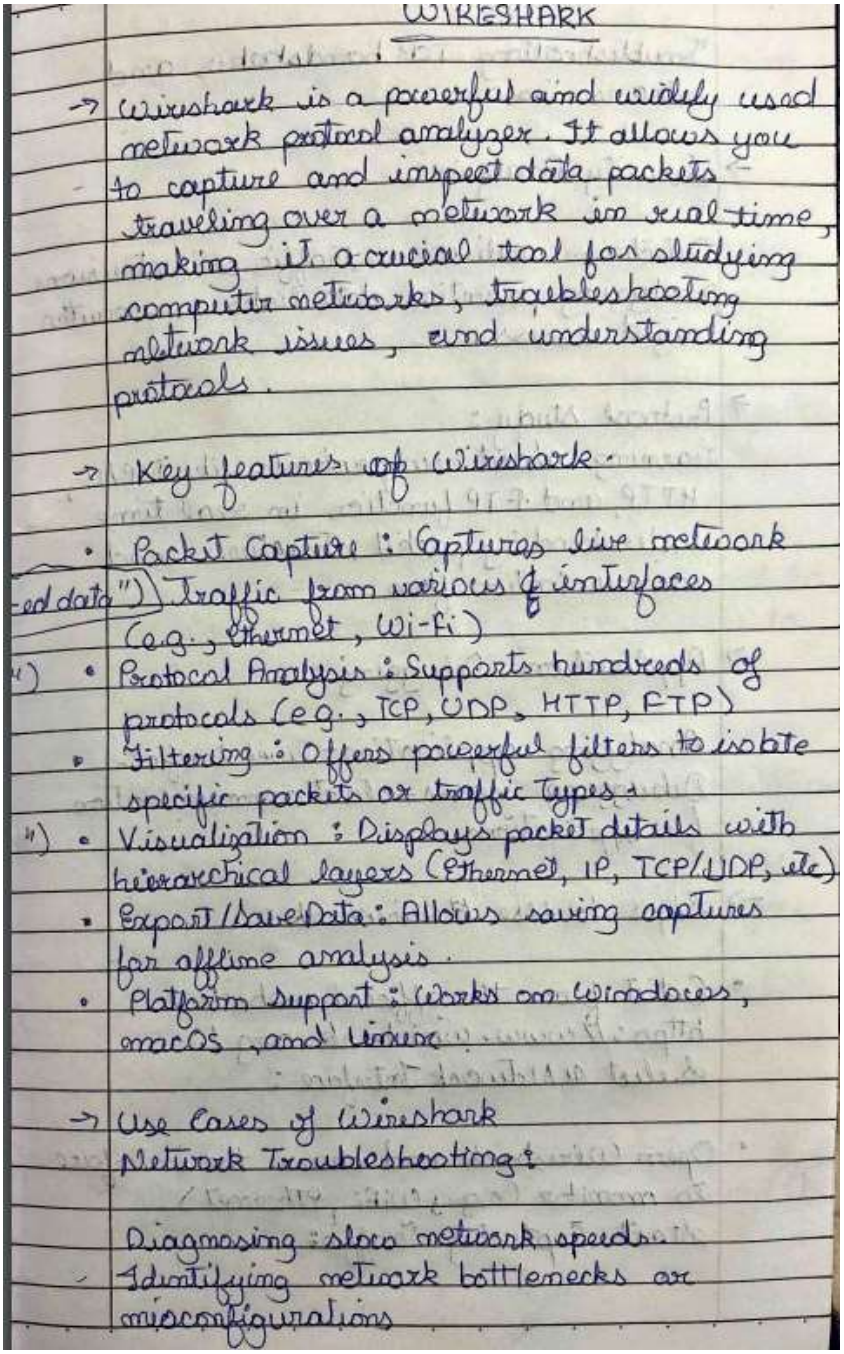
```
def encode(data, key):
    key_len = len(key)
    append_data = data + '0' * (key_len - 1)
    remainder = mod2div(append_data, key)
    codeword = data + remainder
    print(f"Encoded Data : {codeword}")
    return codeword

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")

if __name__ == "__main__":
    data = input("Enter the data bits: ")
    key = input("Enter the key (divisor): ")
    encoded_data = encode(data, key)
    print("\nDecoding the encoded data...")
    decode(encode_data, key)
```

Output Clear

```
Enter the data bits: 100100100100100
Enter the key (divisor): 10101
Encoded Data: 1001001001001000001

Decoding the encoded data...
Remainder after decoding: 0000
No error detected in received data

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

Program 18:

Tool exploration : wireshark



WIRESHARK

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

→ Key features of Wireshark.

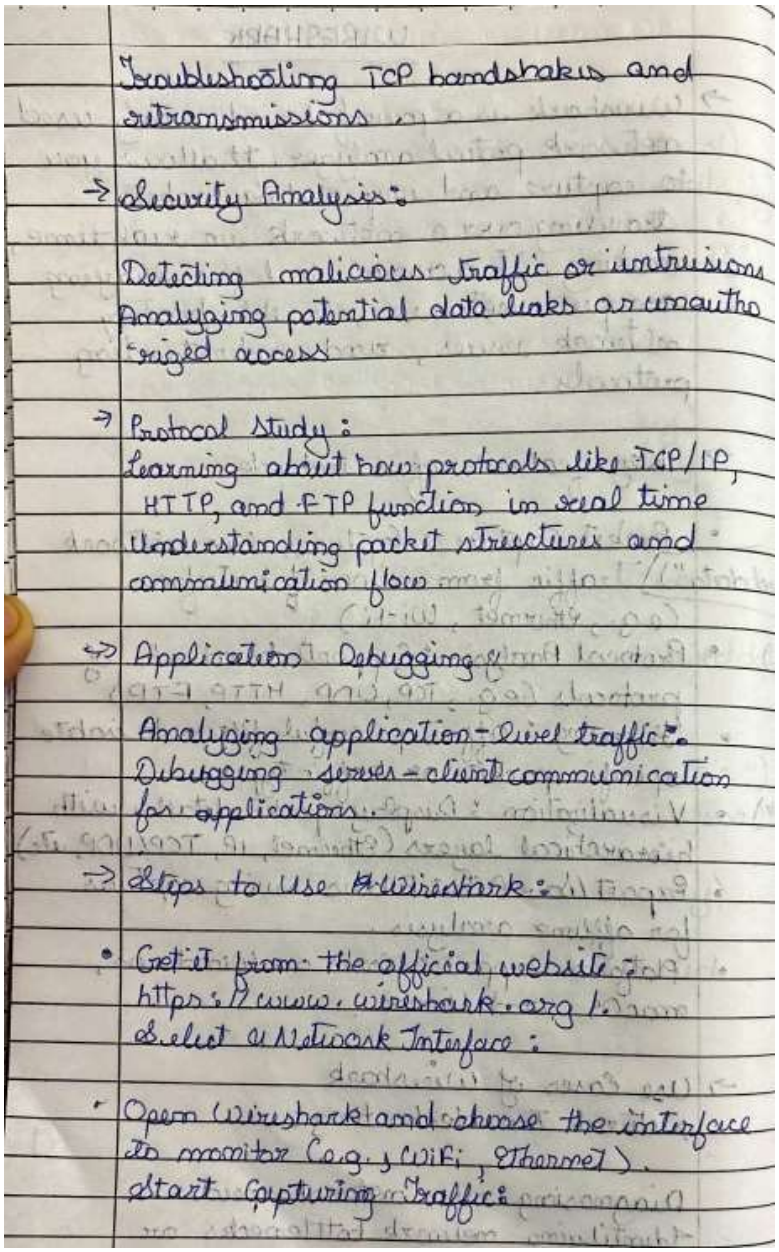
- Packet Capture : Captures live network traffic from various interfaces (e.g., Ethernet, Wi-Fi)
- Protocol Analysis : Supports hundreds of protocols (e.g., TCP, UDP, HTTP, FTP)
- Filtering : Offers powerful filters to isolate specific packets or traffic types.
- Visualization : Displays packet details with hierarchical layers (Ethernet, IP, TCP/UDP, etc)
- Export/Save Data : Allows saving captures for offline analysis.
- Platform Support : Works on Windows, macOS, and Linux.

→ Use Cases of Wireshark

Network Troubleshooting :

Diagnosing slow network speeds.

Identifying network bottlenecks or misconfigurations



Troubleshooting TCP handshakes and retransmissions.

→ Security Analysis:

Detecting malicious traffic or intrusions
Analyzing potential data leaks or unauthorized access

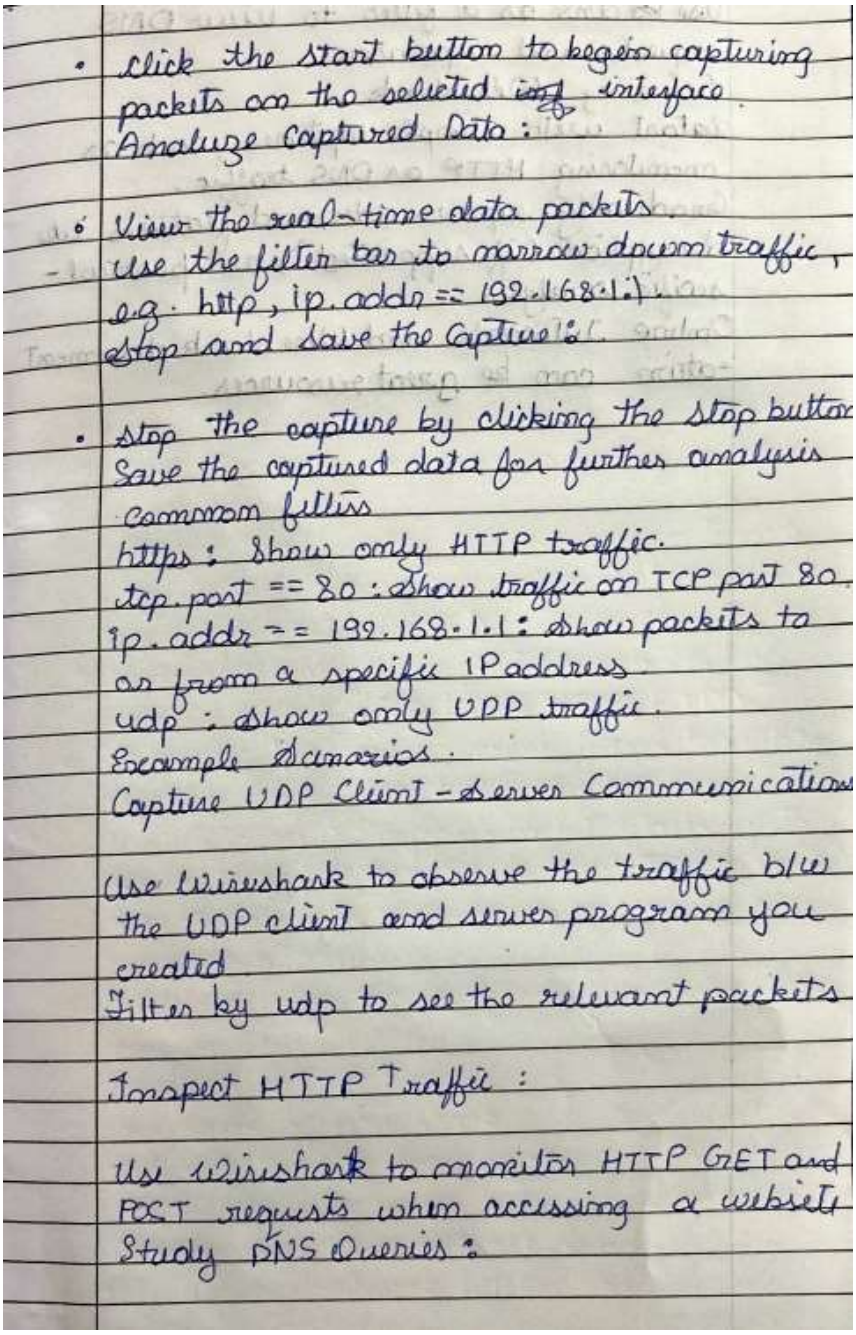
→ Protocol Study:
Learning about how protocols like TCP/IP, HTTP, and FTP function in real time
Understanding packet structures and communication flow

→ Application Debugging:

Analyzing application-level traffic.
Debugging server-client communication for applications.

→ Steps to Use Wireshark:

- Get it from the official website:
https://www.wireshark.org/
Select a Network Interface:
- Open Wireshark and choose the interface to monitor (e.g., WiFi, Ethernet).
Start Capturing Traffic:



- Click the start button to begin capturing packets on the selected interface.

Analyze Captured Data:

- View the real-time data packets.
- Use the filter bar to narrow down traffic, e.g. http, ip.addr == 192.168.1.1.
- Stop and Save the Capture:
- Stop the capture by clicking the stop button.

Save the captured data for further analysis.

Common filters

https: Show only HTTP traffic.

tcp.port == 80: Show traffic on TCP port 80.

ip.addr == 192.168.1.1: Show packets to or from a specific IP address.

udp: Show only UDP traffic.

Example Scenarios:

Capture UDP Client-Server Communications

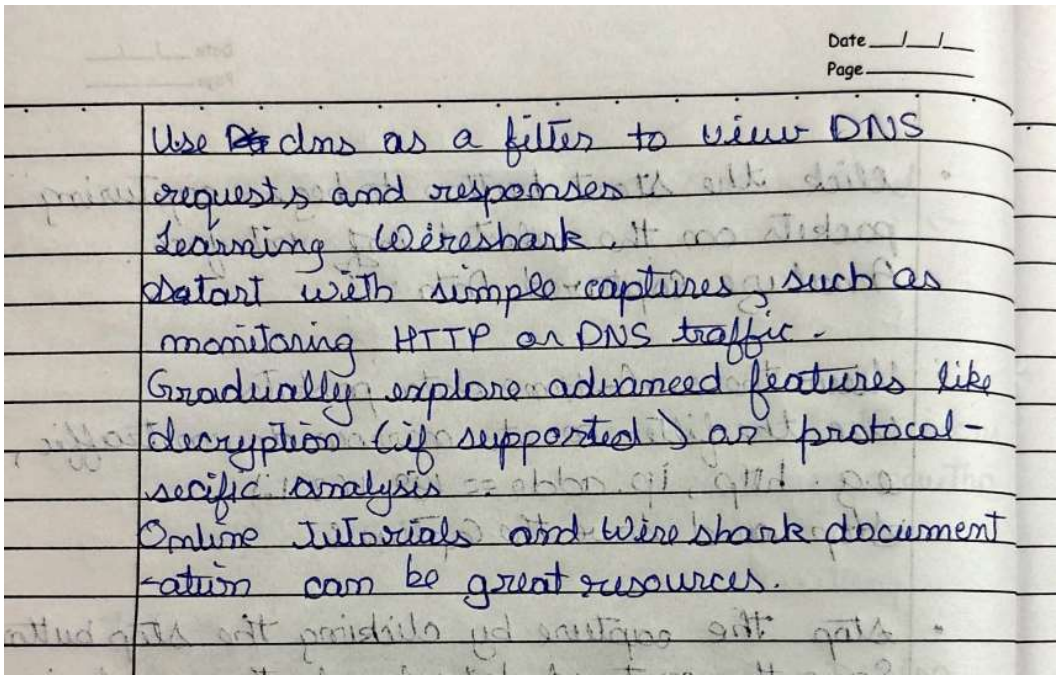
Use Wireshark to observe the traffic b/w the UDP client and server program you created.

Filter by udp to see the relevant packets.

Inspect HTTP Traffic:

Use Wireshark to monitor HTTP GET and POST requests when accessing a website.

Study DNS Queries:



Date __/__/__
Page _____

Use dns as a filter to view DNS requests and responses in Wireshark. It can be tricky initially because DNS queries and responses are usually quick and numerous. You might want to start with simple captures, such as monitoring HTTP or DNS traffic. Gradually explore advanced features like decryption (if supported) and protocol-specific analysis. Online tutorials and Wireshark documentation can be great resources.