VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

Bhoomi Suresh Kota (1BM23CS065)

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 **Bhoomi Suresh Kota (1BM23CS065),** 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.

Spoorthi D M Assistant Professor Department of CSE, BMSCE	**Dr. Kavitha Sooda** Professor & HOD Department of CSE, BMSCE

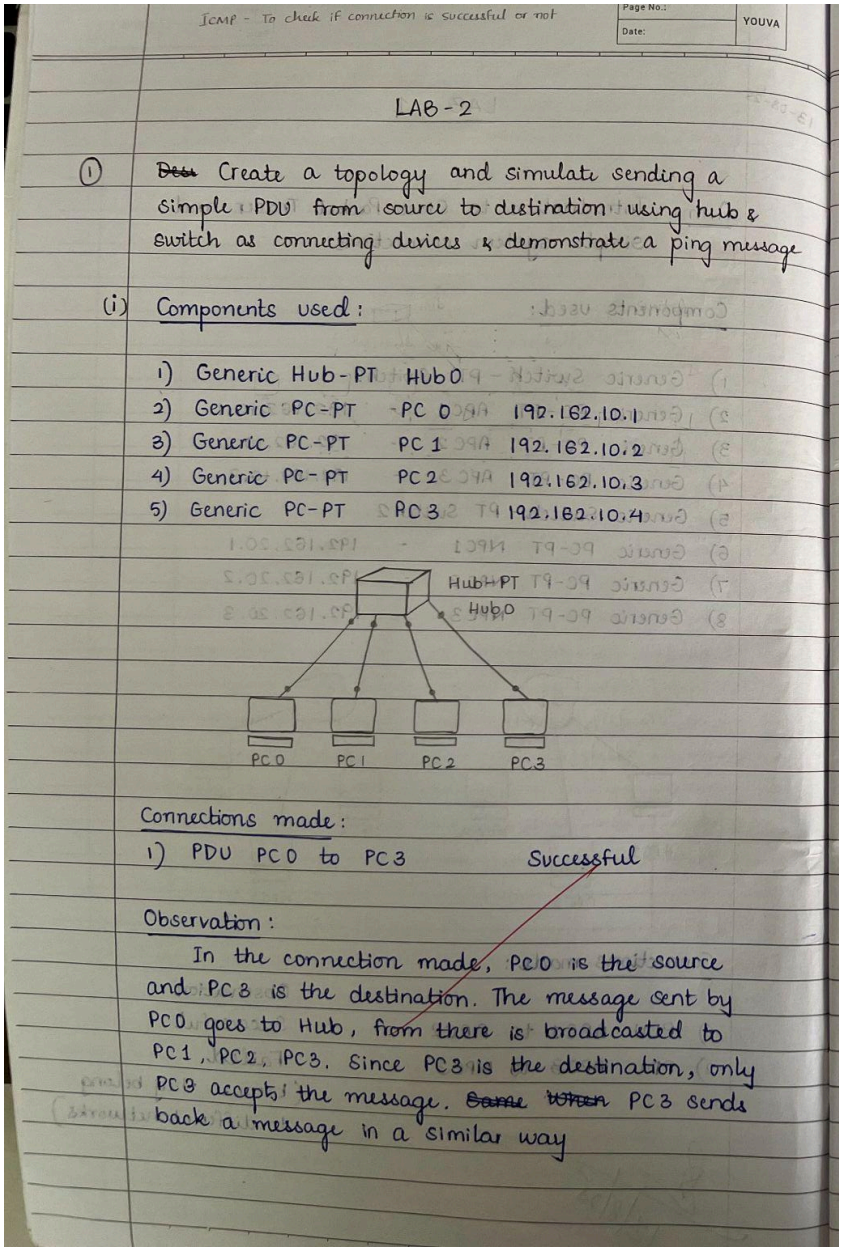
Index

Github Link:
https://github.com/BhoomiSuresh/CN-Lab.git

Program 1

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

Observation:



ICMP - To check if connection is successful or not

Page No.:
Date:
YOUVA

LAB - 2

① ~~Desc~~ Create a topology and simulate sending a simple PDU from source to destination using hub & switch as connecting devices & demonstrate a ping message

(i) Components used:

1)	Generic Hub-PT	Hub 0	
2)	Generic PC-PT	PC 0	192.162.10.1
3)	Generic PC-PT	PC 1	192.162.10.2
4)	Generic PC-PT	PC 2	192.162.10.3
5)	Generic PC-PT	PC 3	192.162.10.4

Connections made:

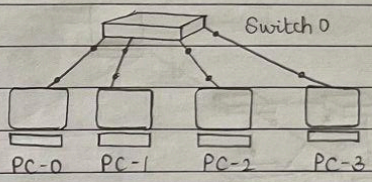
1) PDU PC 0 to PC 3 Successful

Observation:

In the connection made, PC 0 is the source and PC 3 is the destination. The message sent by PC 0 goes to Hub, from there is broadcasted to PC 1, PC 2, PC 3. Since PC 3 is the destination, only PC 3 accepts the message. ~~Same when~~ PC 3 sends back a message in a similar way

(ii) Components used:

1) Generic Switch - PT Switch 0
2) Generic PC - PT PC 0 192.162.10.1
3) Generic PC-PT PC 1 192.162.10.2
4) Generic PC-PT PC 2 192.162.10.3
5) Generic PC-PT PC 3 192.162.10.4



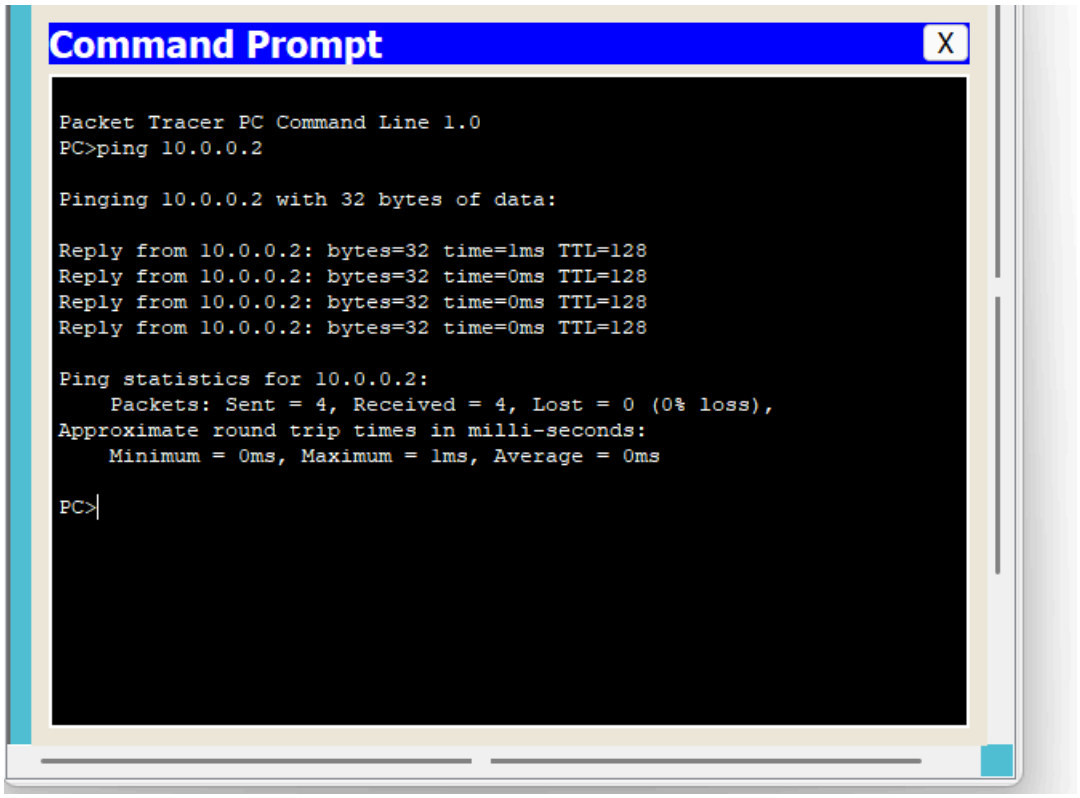
Connections made:

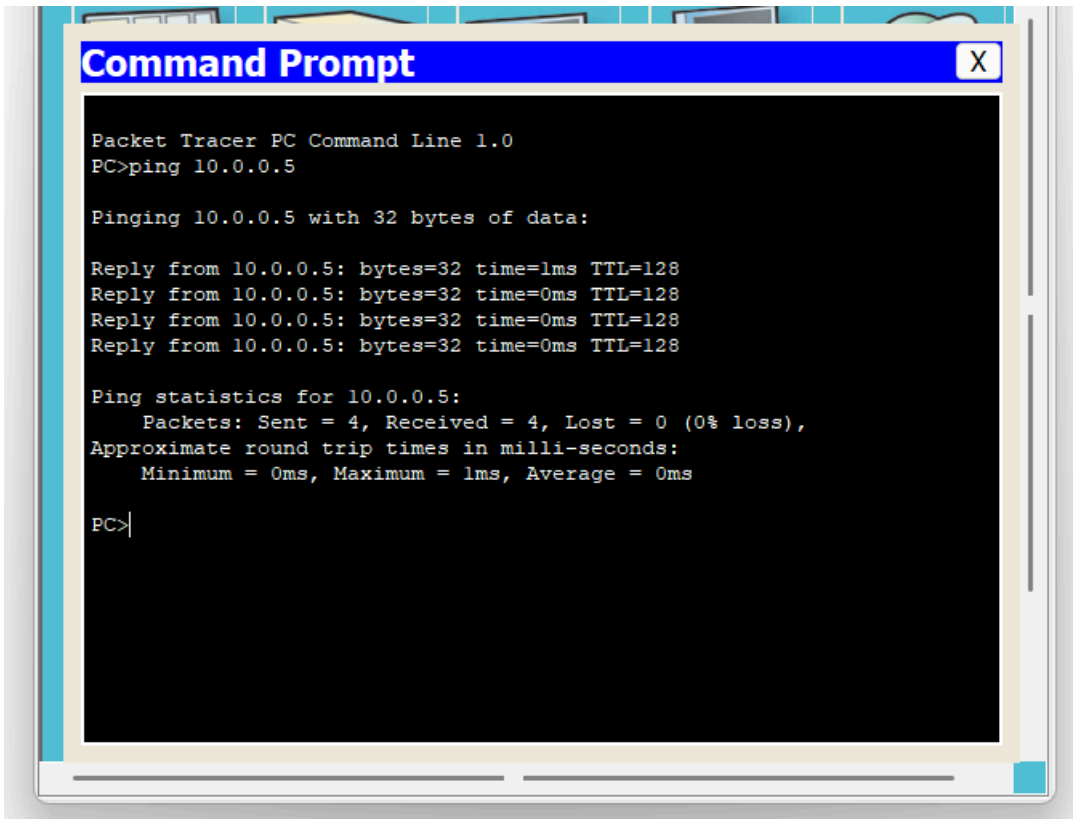
PDU PC 0 to PC 3 Successful

Observation:

In the connection made, PC 0 is the source & PC 3 is the destination. The message sent from PC 0 goes to switch and is received only by PC 3. PC 3 sends back a message in a similar way.

Output:

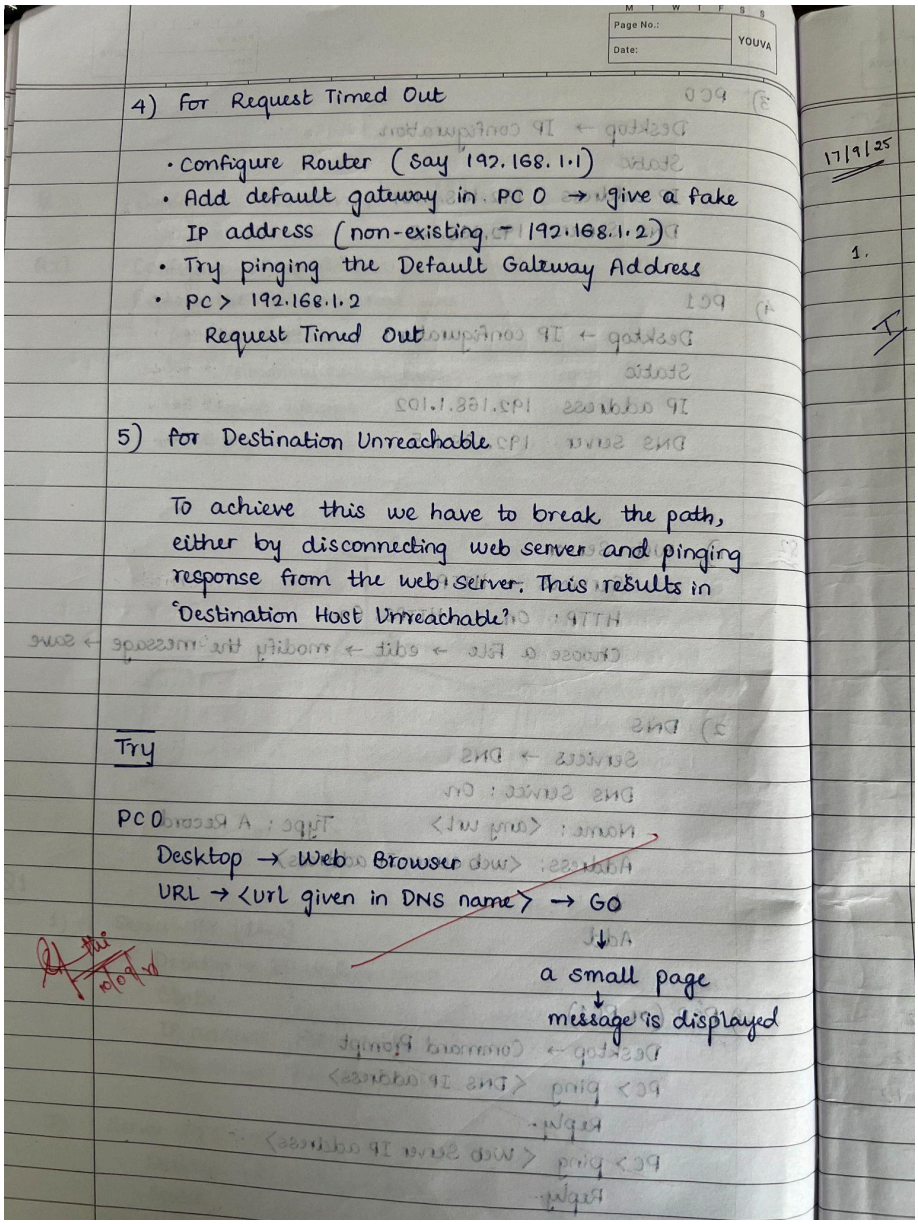

Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.2
Pinging 10.0.0.2 with 32 bytes of data:
Reply from 10.0.0.2: bytes=32 time=1ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128
Ping statistics for 10.0.0.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 1ms, Average = 0ms
PC>


Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.5
Pinging 10.0.0.5 with 32 bytes of data:
Reply from 10.0.0.5: bytes=32 time=1ms TTL=128
Reply from 10.0.0.5: bytes=32 time=0ms TTL=128
Reply from 10.0.0.5: bytes=32 time=0ms TTL=128
Reply from 10.0.0.5: bytes=32 time=0ms TTL=128
Ping statistics for 10.0.0.5:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 1ms, Average = 0ms
PC>

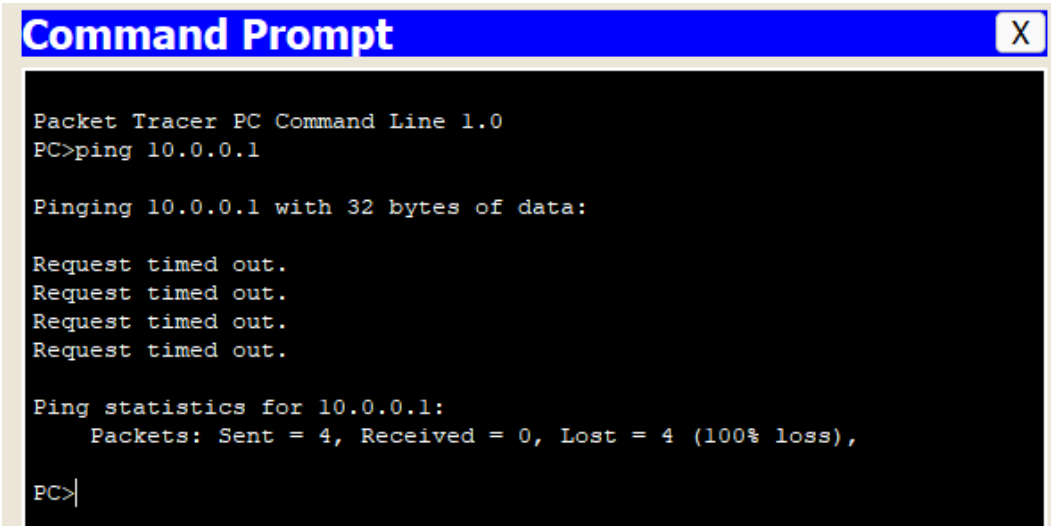
Program 2

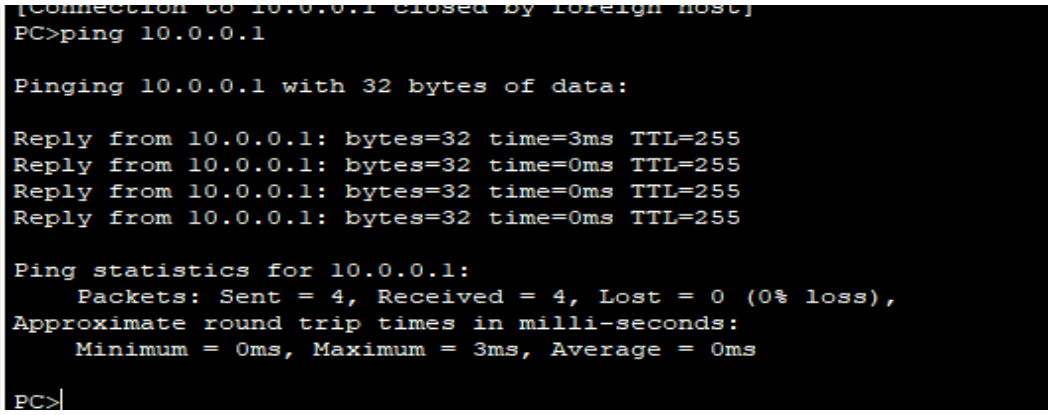
Aim: Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply

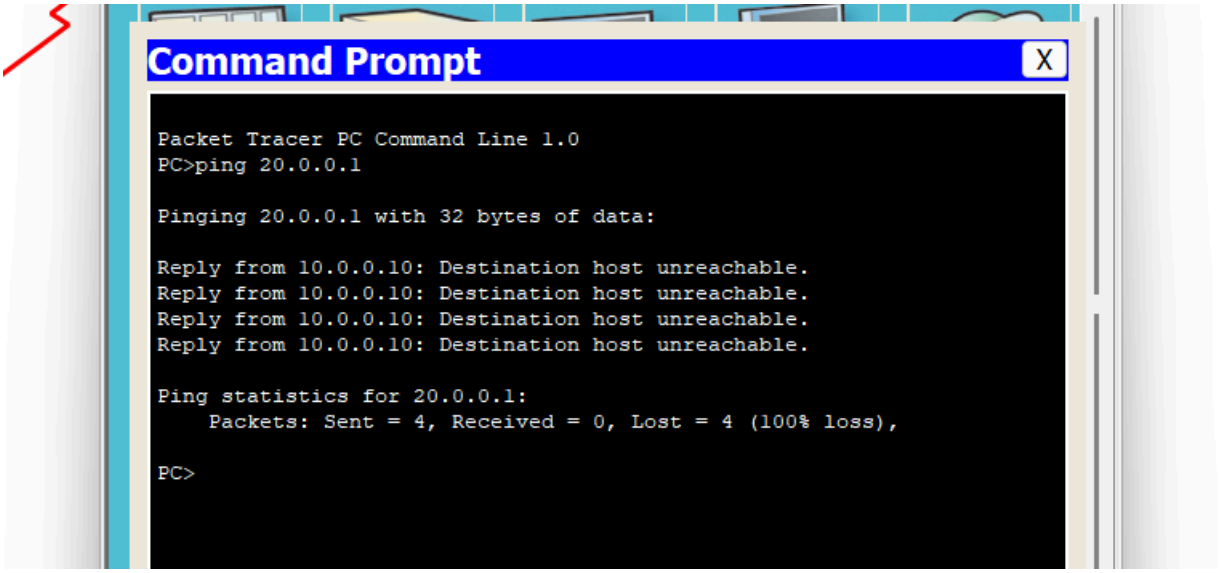
Observation:



Output:


Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.1
Pinging 10.0.0.1 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 10.0.0.1:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>

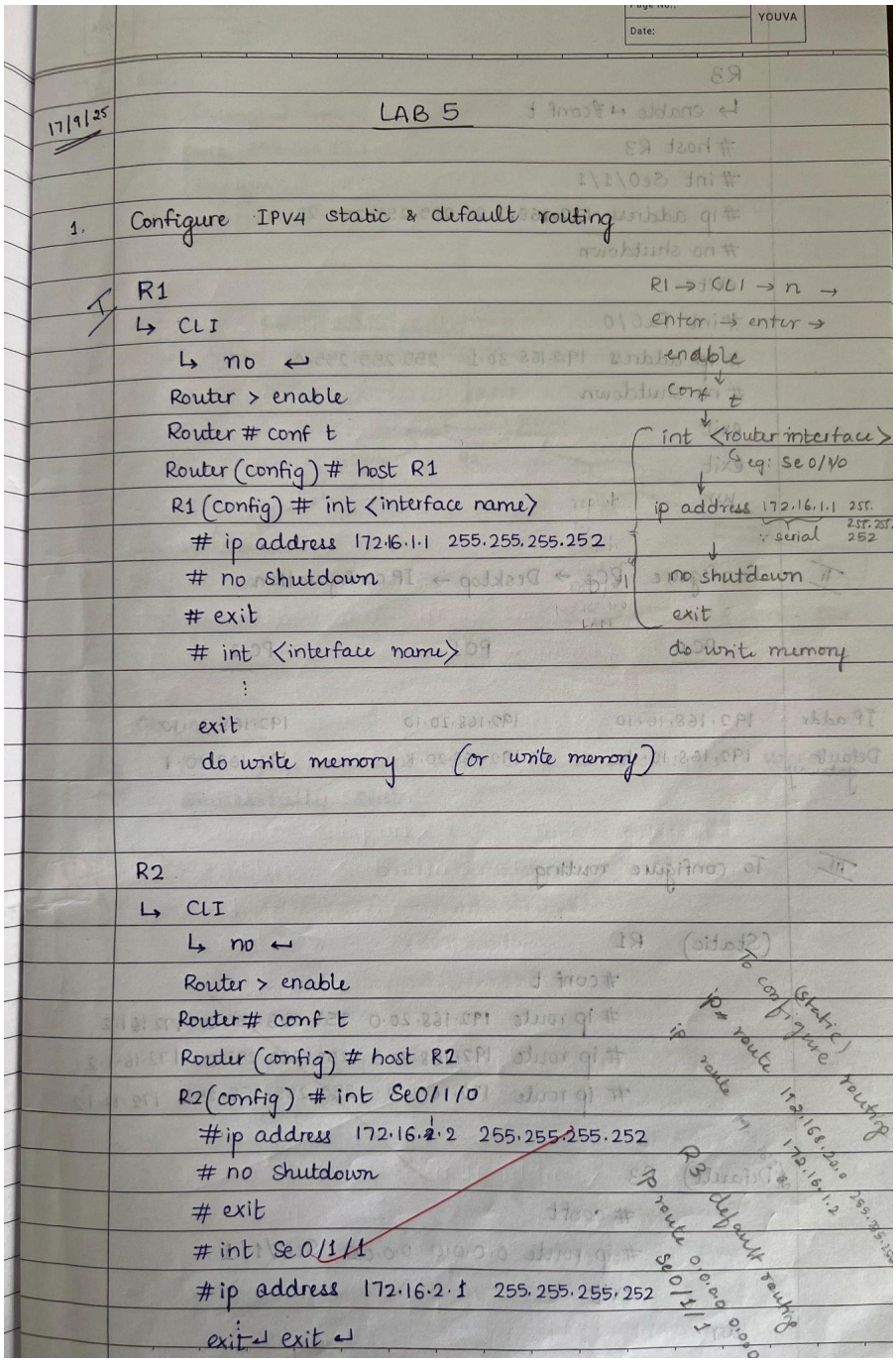

PC>ping 10.0.0.1
Pinging 10.0.0.1 with 32 bytes of data:
Reply from 10.0.0.1: bytes=32 time=3ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Ping statistics for 10.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 3ms, Average = 0ms
PC>


Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 20.0.0.1
Pinging 20.0.0.1 with 32 bytes of data:
Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: Destination host unreachable.
Ping statistics for 20.0.0.1:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>

Program 3

Aim: Configure default route, static route to the Router

Observation:



R3

↳ enable ↵ #conf t

#host R3

#int Se0/1/1

#ip address 172.168.2.2 255.255.255.252

#no shutdown

exit

#int Fa0/0

#ip address 192.168.30.1 255.255.255.0

#no shutdown

exit

exit

wr

II Configure PCs → Desktop → IP configuration

	PC0	PC1	PC2
IP addr	192.168.10.10	192.168.20.10	192.168.30.10
Default gateway	192.168.10.1	192.168.20.1	192.168.30.1

III To configure routing

(Static) R1

#conf t

#ip route 192.168.20.0 255.255.255.0 172.16.1.2

#ip route 192.168.30.0 255.255.255.0 172.16.1.2

#ip route 172.16.2.0 255.255.255.252 172.16.1.2

(Default) R3

#conf t

#ip route 0.0.0.0 0.0.0.0 Se0/1/1

exit

wr

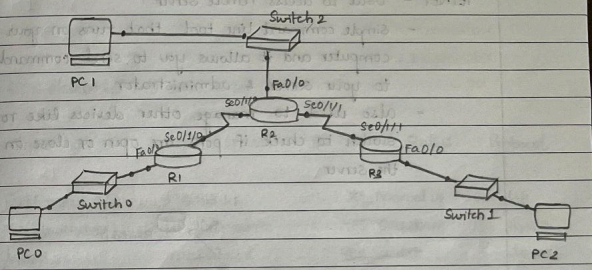
IV PC

↳ Command Prompt

> ping 192.168.20.1

Reply...

Successful

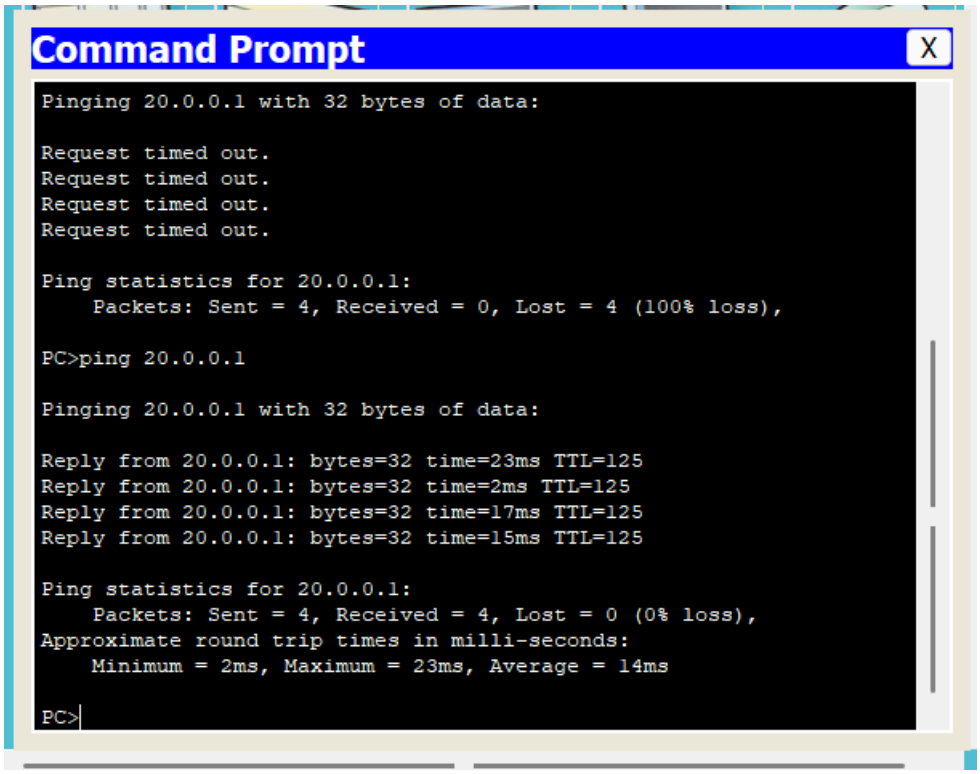


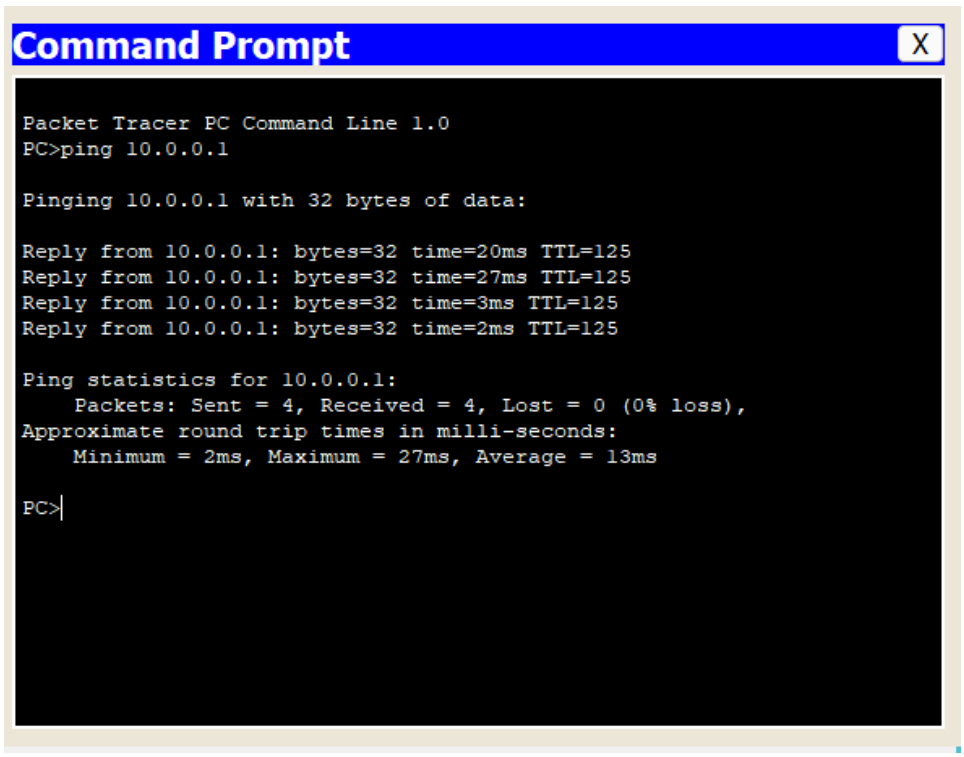
Observation:

Messages were sent from 1 PC to another. All were successfully sent

17/10/25

Output:

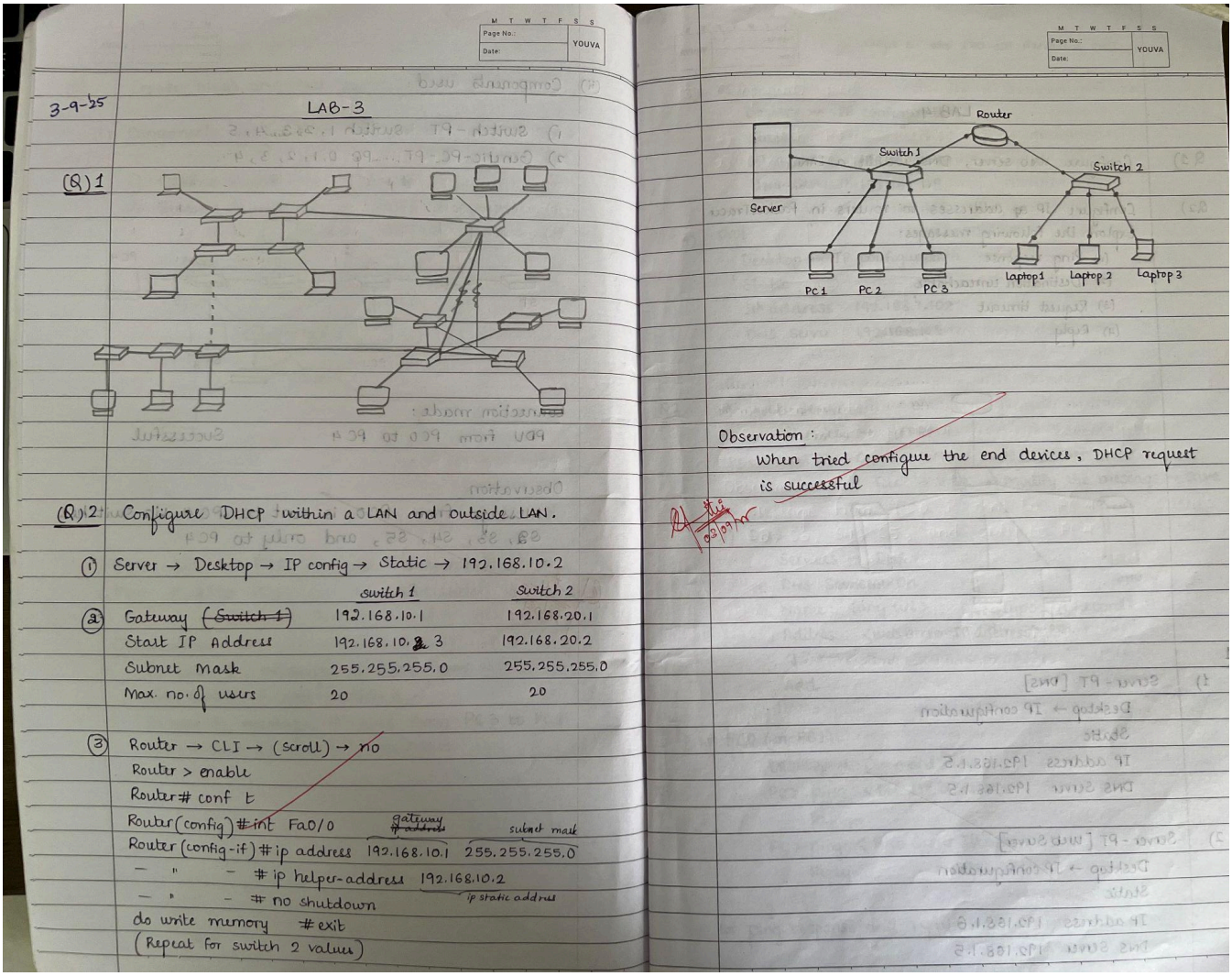

Command Prompt
X
Pinging 20.0.0.1 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 20.0.0.1:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>ping 20.0.0.1
Pinging 20.0.0.1 with 32 bytes of data:
Reply from 20.0.0.1: bytes=32 time=23ms TTL=125
Reply from 20.0.0.1: bytes=32 time=2ms TTL=125
Reply from 20.0.0.1: bytes=32 time=17ms TTL=125
Reply from 20.0.0.1: bytes=32 time=15ms TTL=125
Ping statistics for 20.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 23ms, Average = 14ms
PC>


Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.1
Pinging 10.0.0.1 with 32 bytes of data:
Reply from 10.0.0.1: bytes=32 time=20ms TTL=125
Reply from 10.0.0.1: bytes=32 time=27ms TTL=125
Reply from 10.0.0.1: bytes=32 time=3ms TTL=125
Reply from 10.0.0.1: bytes=32 time=2ms TTL=125
Ping statistics for 10.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 27ms, Average = 13ms
PC>

Program 4

Aim: Configure DHCP within a LAN and outside LAN.

Observation:



3-9-25

LAB-3

(Q) 1

(Q) 2 Configure DHCP within a LAN and outside LAN.

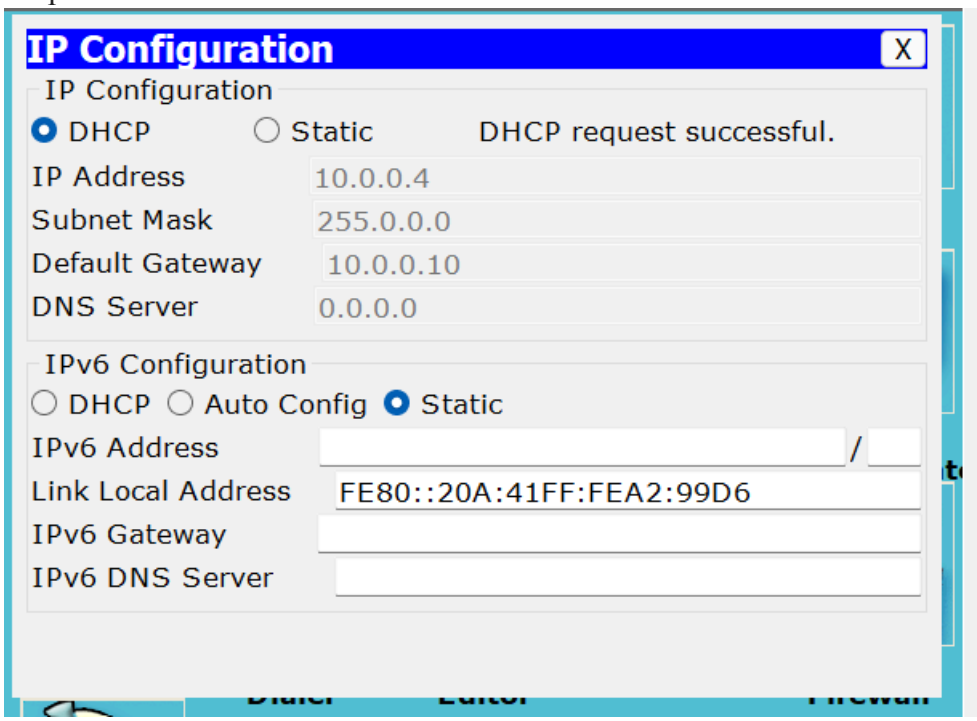
① Server → Desktop → IP config → Static → 192.168.10.2

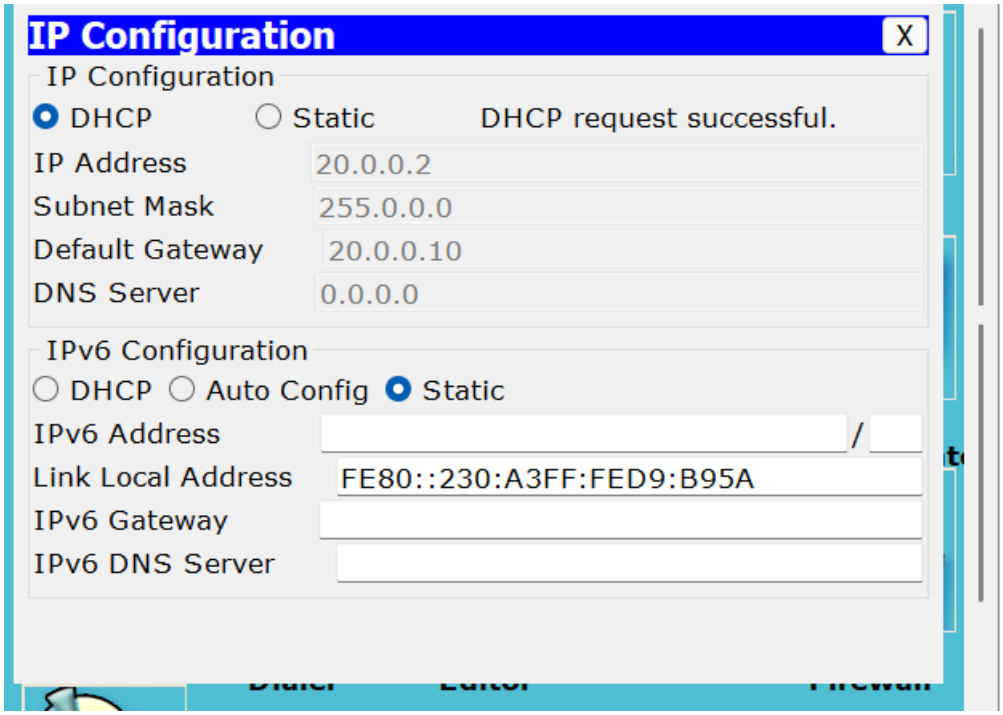
		switch 1	switch 2
②	Gateway ~~(Switch 1)~~	192.168.10.1	192.168.20.1
	Start IP Address	192.168.10.3	192.168.20.2
	Subnet Mask	255.255.255.0	255.255.255.0
	Max. no. of users	20	20

③ Router → CLI → (scroll) → no
Router > enable
Router# conf t
Router(config)#int Fa0/0
Router(config-if)# ip address 192.168.10.1 255.255.255.0 (gateway # address; subnet mask)
— " — # ip helper-address 192.168.10.2 (ip static address)
— " — # no shutdown
do write memory #exit
(Repeat for switch 2 values)

Observation:
When tried configure the end devices, DHCP request is successful

Output:

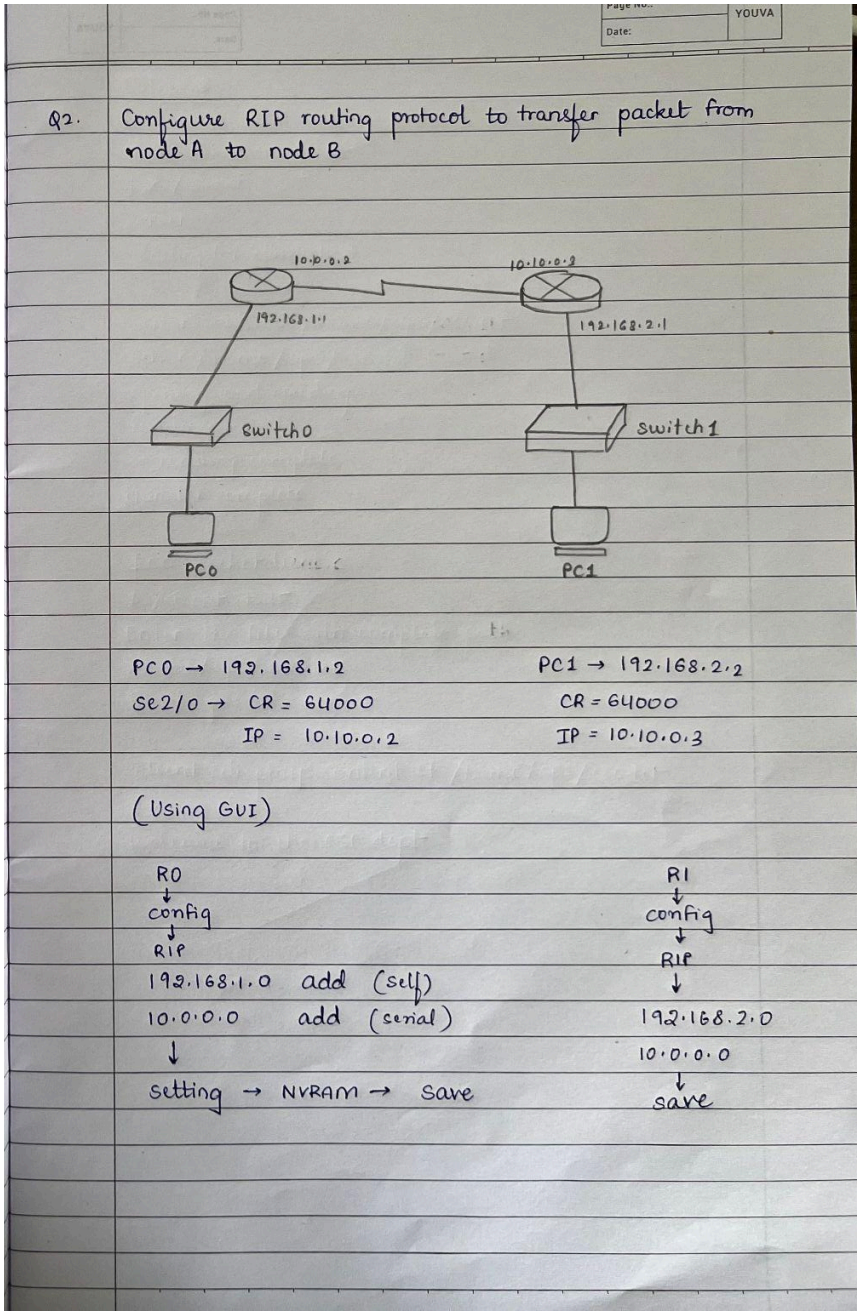

IP Configuration
X
IP Configuration
DHCP
Static
DHCP request successful.
IP Address
10.0.0.4
Subnet Mask
255.0.0.0
Default Gateway
10.0.0.10
DNS Server
0.0.0.0
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
/
Link Local Address
FE80::20A:41FF:FEA2:99D6
IPv6 Gateway
IPv6 DNS Server


IP Configuration
X
IP Configuration
DHCP
Static
DHCP request successful.
IP Address
20.0.0.2
Subnet Mask
255.0.0.0
Default Gateway
20.0.0.10
DNS Server
0.0.0.0
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
/
Link Local Address
FE80::230:A3FF:FED9:B95A
IPv6 Gateway
IPv6 DNS Server

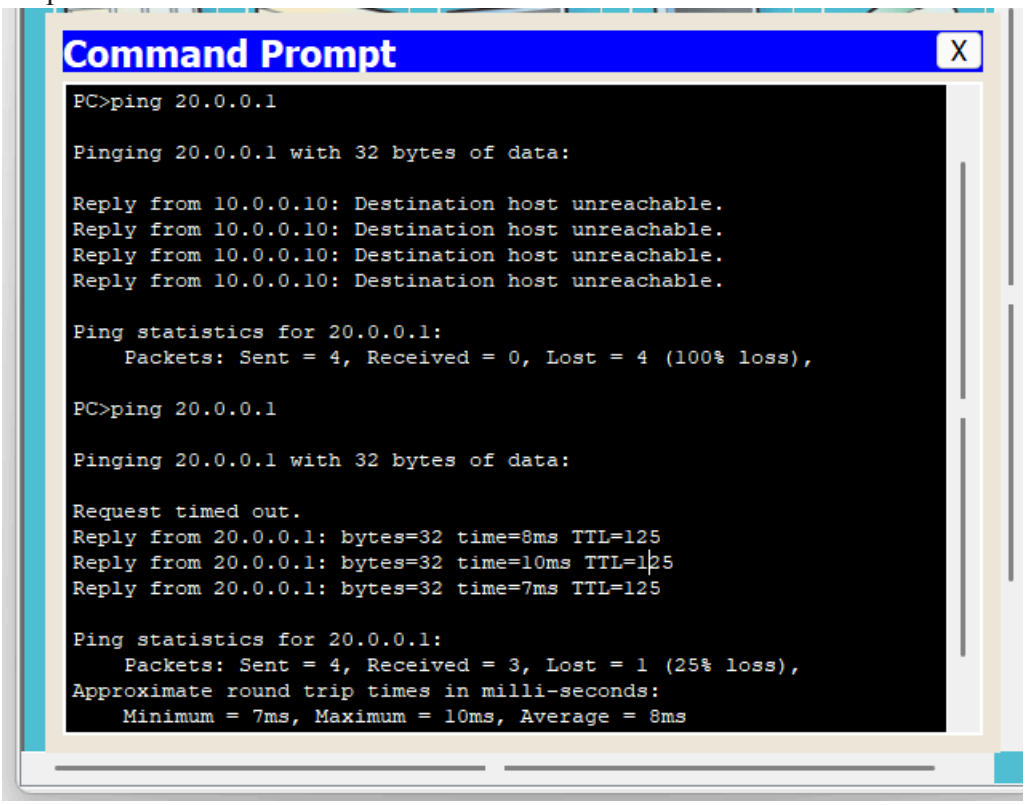
Program 5

Aim: Configure RIP routing Protocol in Routers

Observation:



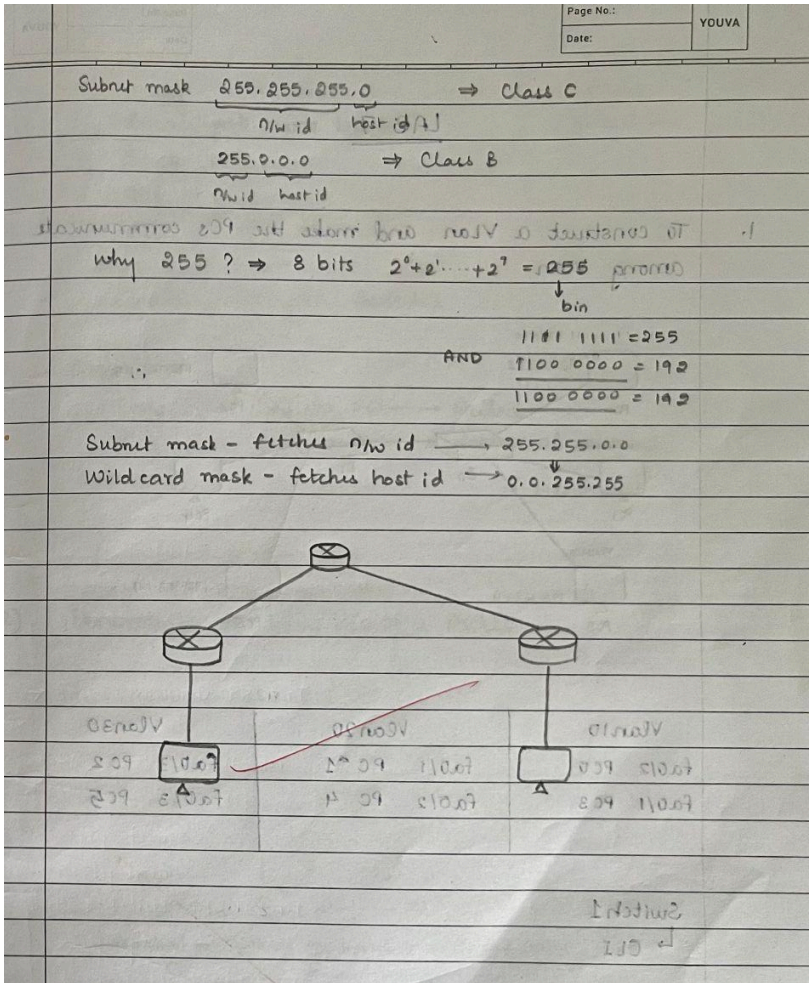
Output:


Command Prompt
X
PC>ping 20.0.0.1
Pinging 20.0.0.1 with 32 bytes of data:
Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: Destination host unreachable.
Reply from 10.0.0.10: 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:
Request timed out.
Reply from 20.0.0.1: bytes=32 time=8ms TTL=125
Reply from 20.0.0.1: bytes=32 time=10ms TTL=125
Reply from 20.0.0.1: bytes=32 time=7ms TTL=125
Ping statistics for 20.0.0.1:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 7ms, Maximum = 10ms, Average = 8ms

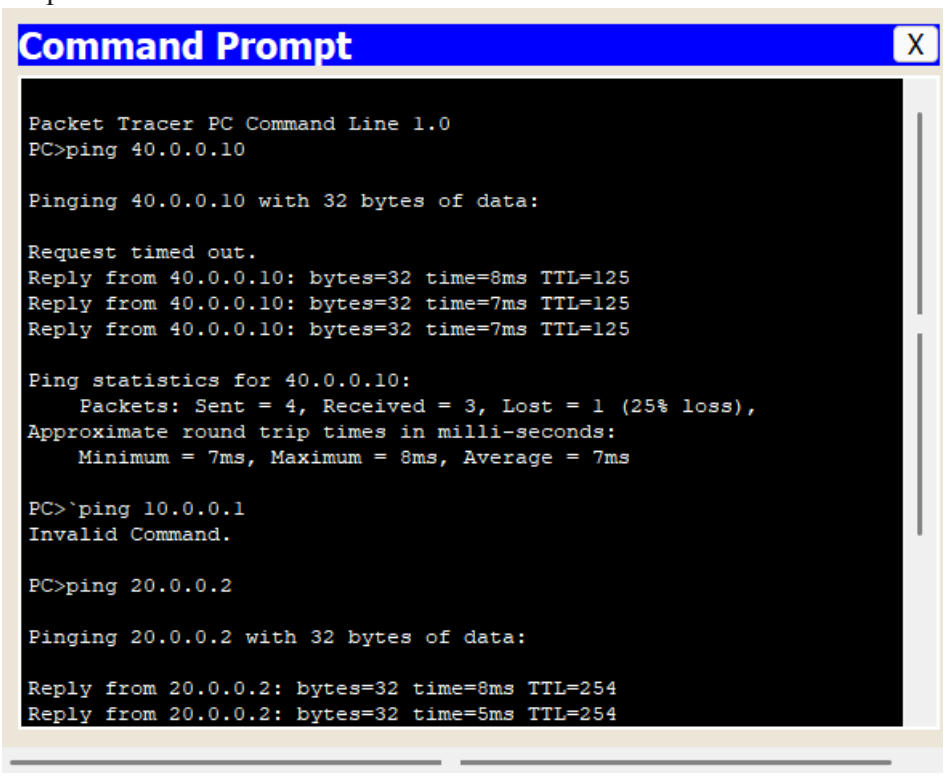
Program 6

Aim: Configure OSPF routing protocol

Observation:



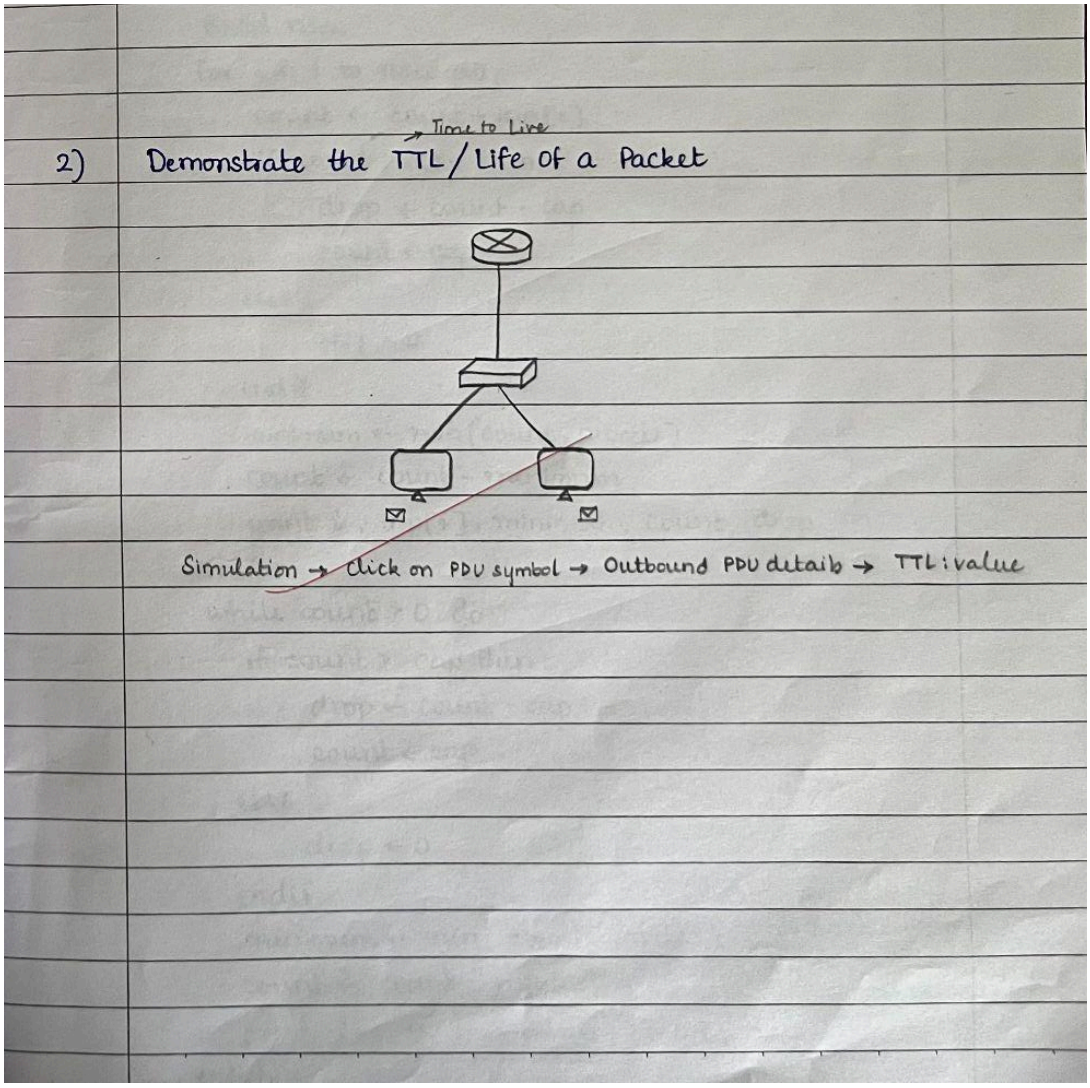
Output:


Command Prompt
X
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=8ms TTL=125
Reply from 40.0.0.10: bytes=32 time=7ms TTL=125
Reply from 40.0.0.10: bytes=32 time=7ms 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 = 7ms, Maximum = 8ms, Average = 7ms
PC>`ping 10.0.0.1
Invalid Command.
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=8ms TTL=254
Reply from 20.0.0.2: bytes=32 time=5ms TTL=254

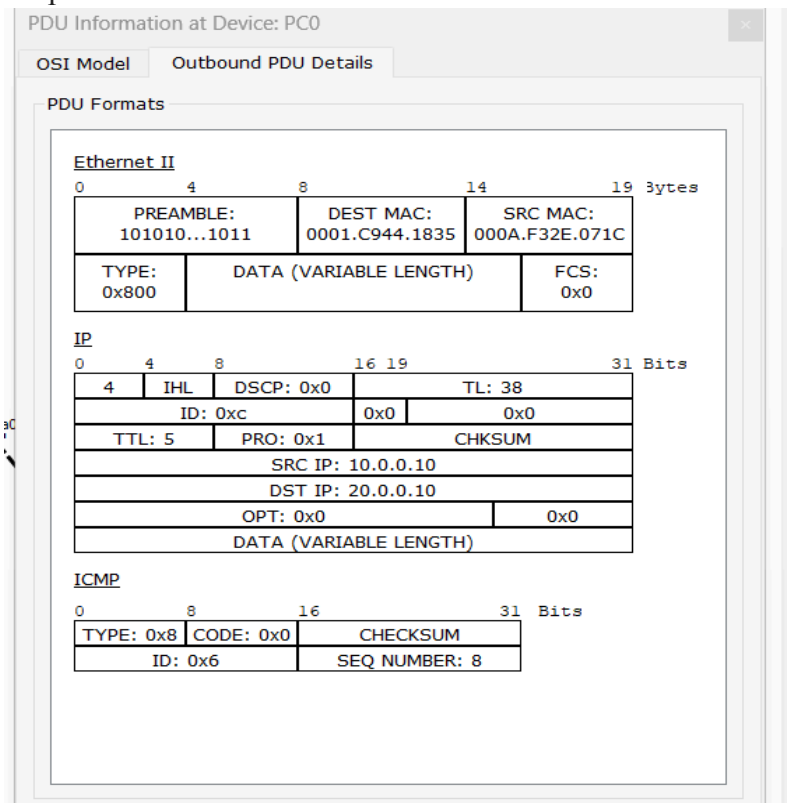
Program 7

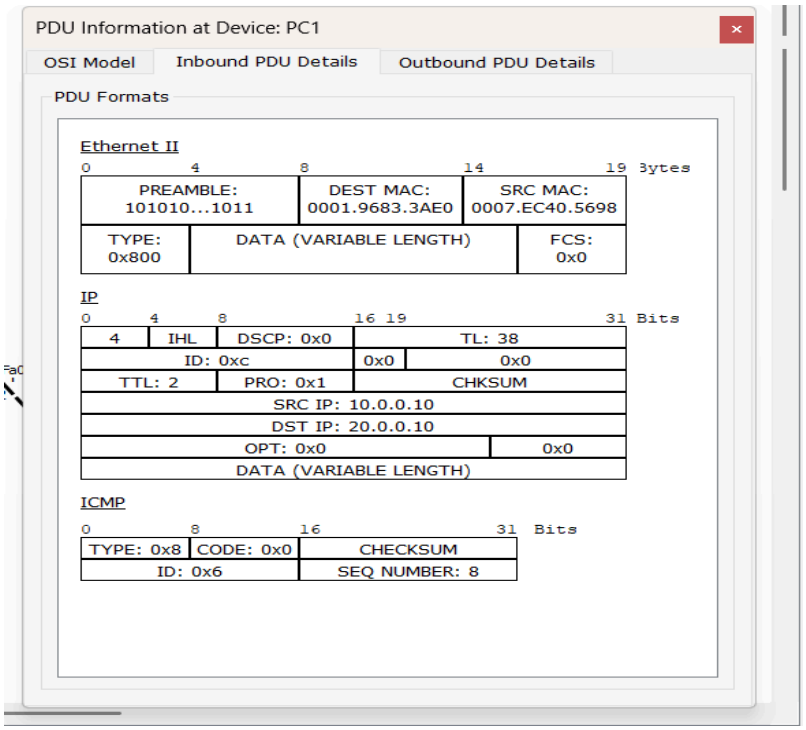
Aim: Demonstrate the TTL/ Life of a Packet

Observation:



Output:

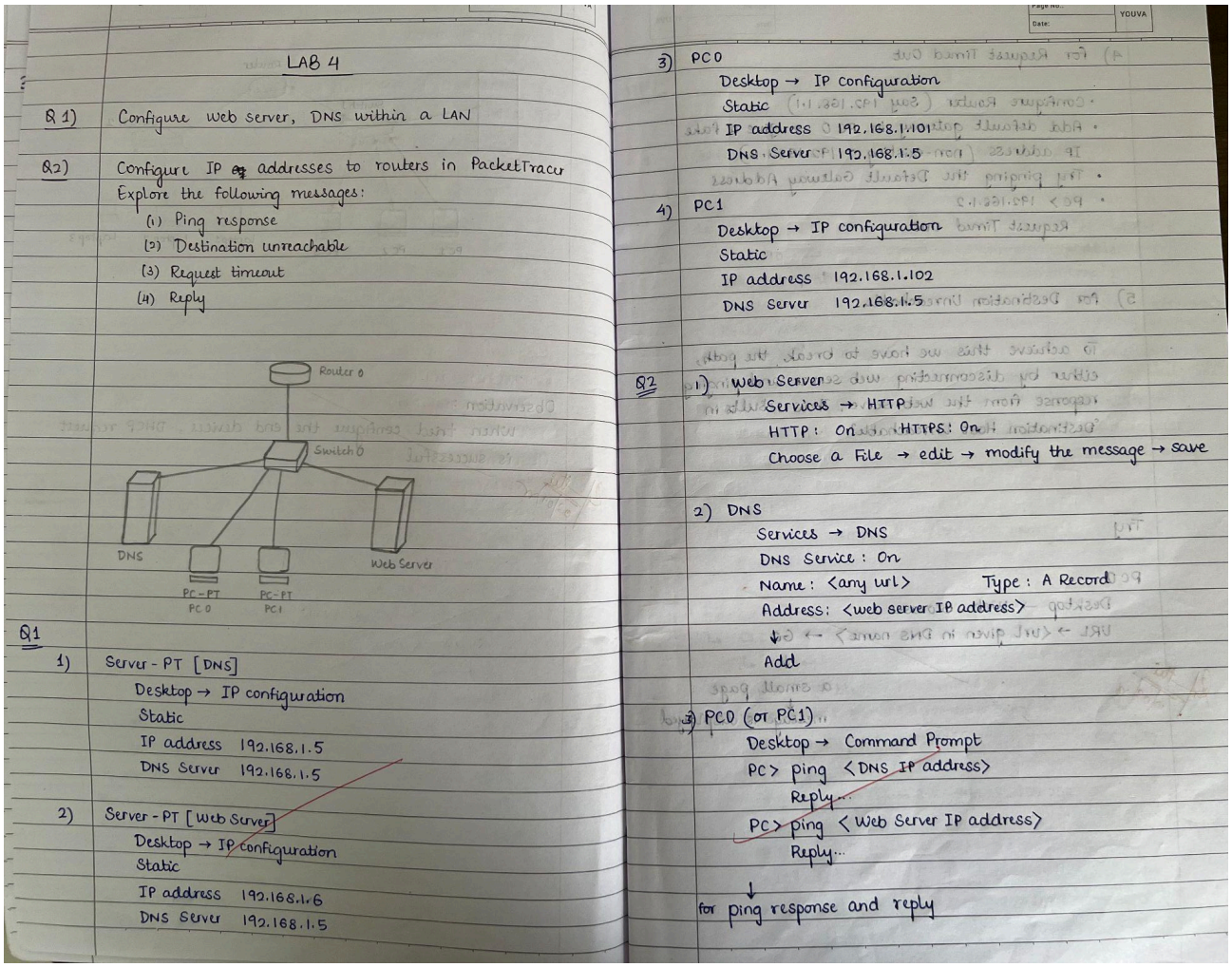

PDU Information at Device: PC0
OSI Model
Outbound PDU Details
PDU Formats
Ethernet II
0 4 8 14 19 Bytes
PREAMBLE: 101010...1011
DEST MAC: 0001.C944.1835
SRC MAC: 000A.F32E.071C
TYPE: 0x800
DATA (VARIABLE LENGTH)
FCS: 0x0
IP
0 4 8 16 19 31 Bits
4
IHL
DSCP: 0x0
TL: 38
ID: 0xc
0x0
0x0
TTL: 5
PRO: 0x1
CHKSUM
SRC IP: 10.0.0.10
DST IP: 20.0.0.10
OPT: 0x0
0x0
DATA (VARIABLE LENGTH)
ICMP
0 8 16 31 Bits
TYPE: 0x8
CODE: 0x0
CHECKSUM
ID: 0x6
SEQ NUMBER: 8


PDU Information at Device: PC1
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
Ethernet II
0 4 8 14 19 Bytes
PREAMBLE: 101010...1011
DEST MAC: 0001.9683.3AE0
SRC MAC: 0007.EC40.5698
TYPE: 0x800
DATA (VARIABLE LENGTH)
FCS: 0x0
IP
0 4 8 16 19 31 Bits
4
IHL
DSCP: 0x0
TL: 38
ID: 0xc
0x0
0x0
TTL: 2
PRO: 0x1
CHKSUM
SRC IP: 10.0.0.10
DST IP: 20.0.0.10
OPT: 0x0
0x0
DATA (VARIABLE LENGTH)
ICMP
0 8 16 31 Bits
TYPE: 0x8
CODE: 0x0
CHECKSUM
ID: 0x6
SEQ NUMBER: 8

Program 8

Aim: Configure Web Server, DNS within a LAN.

Observation:



LAB 4

Q 1) Configure web server, DNS within a LAN

Q2) Configure IP addresses to routers in PacketTracer
Explore the following messages:
(1) Ping response
(2) Destination unreachable
(3) Request timeout
(4) Reply

Q1

1) Server - PT [DNS]
Desktop → IP configuration
Static
IP address 192.168.1.5
DNS Server 192.168.1.5

2) Server - PT [Web Server]
Desktop → IP configuration
Static
IP address 192.168.1.6
DNS Server 192.168.1.5

3) PC0
Desktop → IP configuration
Static
IP address 192.168.1.101
DNS Server 192.168.1.5

4) PC1
Desktop → IP configuration
Static
IP address 192.168.1.102
DNS Server 192.168.1.5

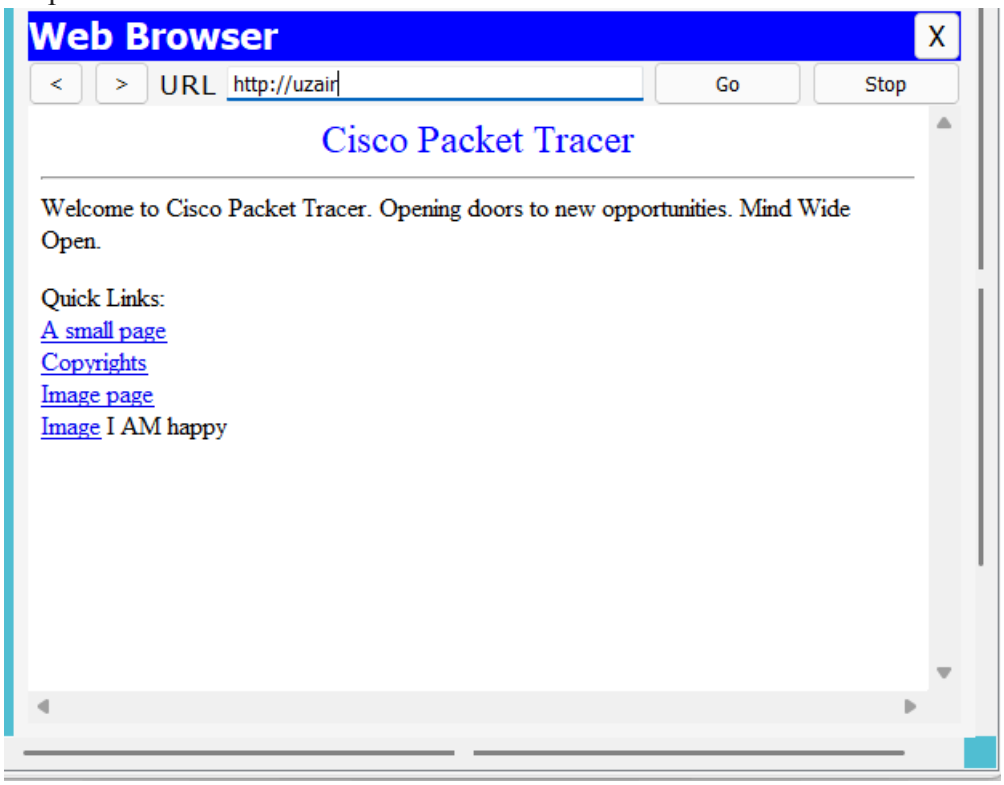
Q2 1) Web Server
Services → HTTP
HTTP: On HTTPS: On
Choose a File → edit → modify the message → save

2) DNS
Services → DNS
DNS Service : On
Name: <any url> Type : A Record
Address: <web server IP address>
Add

3) PC0 (or PC1)
Desktop → Command Prompt
PC> ping <DNS IP address>
Reply...
PC> ping <Web Server IP address>
Reply...

for ping response and reply

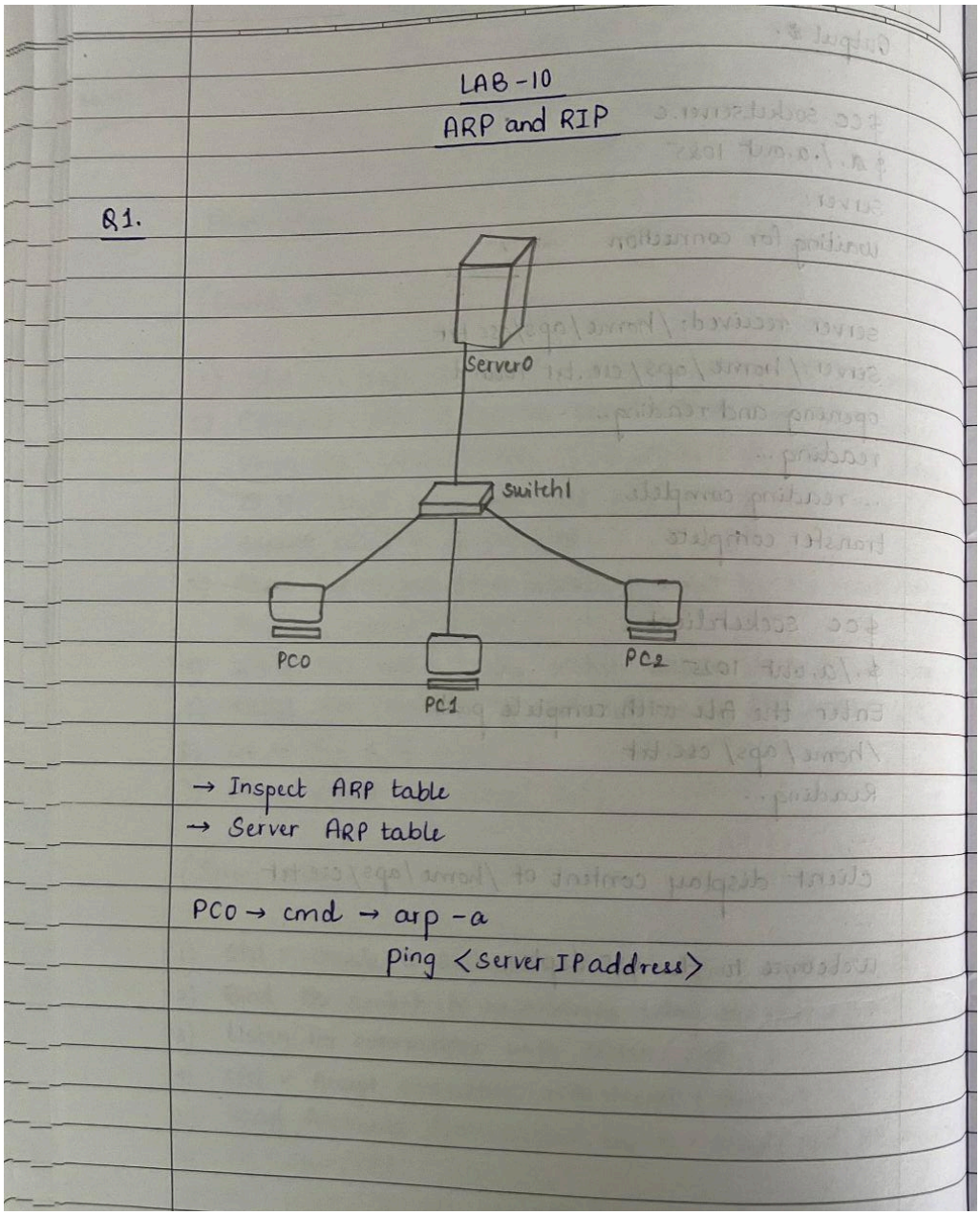
Output:


Web Browser
X
<
>
URL
http://uzair
Go
Stop
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 I AM happy

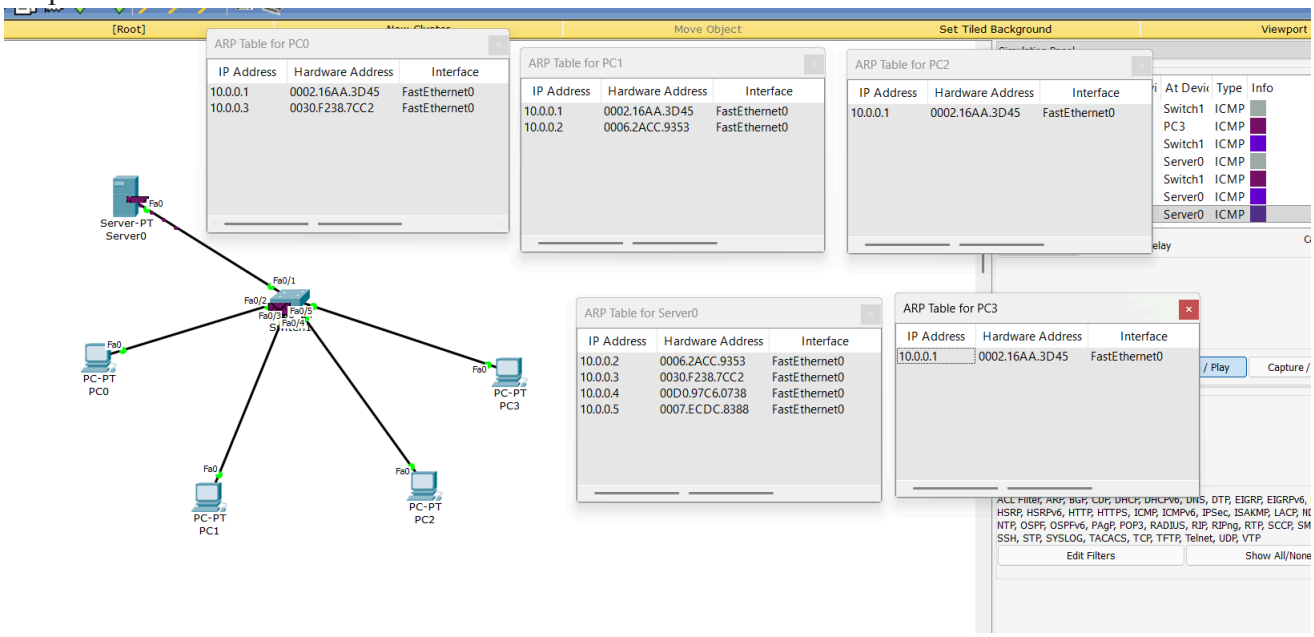
Program 9

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

Observation:



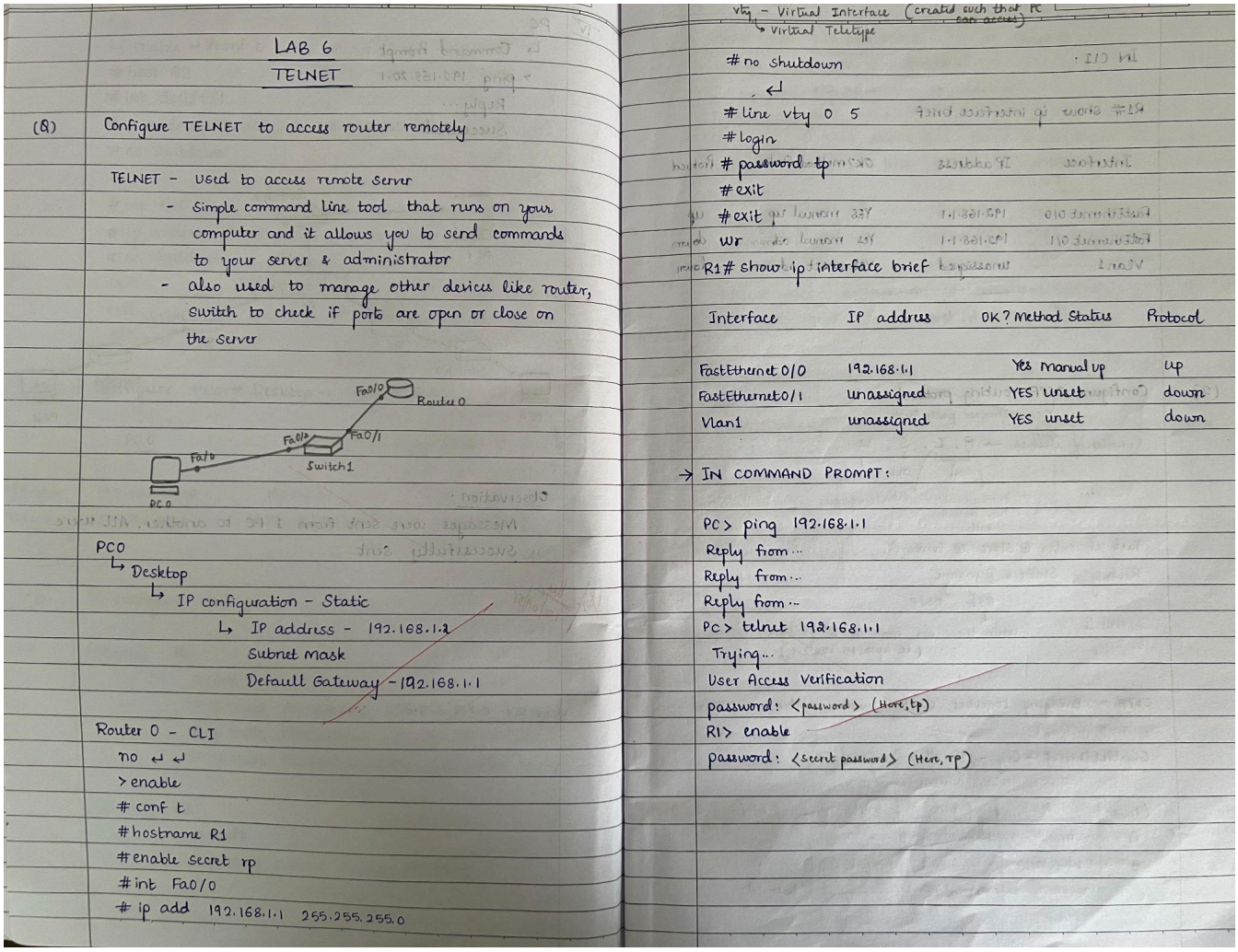
Output:


[Root]
Move Object
Set Tiled Background
Viewport
ARP Table for PC0
IP Address Hardware Address Interface
10.0.0.1 0002.16AA.3D45 FastEthernet0
10.0.0.3 0030.F238.7CC2 FastEthernet0
ARP Table for PC1
IP Address Hardware Address Interface
10.0.0.1 0002.16AA.3D45 FastEthernet0
10.0.0.2 0006.2ACC.9353 FastEthernet0
ARP Table for PC2
IP Address Hardware Address Interface
10.0.0.1 0002.16AA.3D45 FastEthernet0
At Devic Type Info
Switch1 ICMP
PC3 ICMP
Switch1 ICMP
Server0 ICMP
Switch1 ICMP
Server0 ICMP
Server0 ICMP
Server-PT Server0
Fa0
Fa0/1
Fa0/2
Fa0/3
Fa0/5
Fa0/4
PC-PT PC0
PC-PT PC3
PC-PT PC1
PC-PT PC2
ARP Table for Server0
IP Address Hardware Address Interface
10.0.0.2 0006.2ACC.9353 FastEthernet0
10.0.0.3 0030.F238.7CC2 FastEthernet0
10.0.0.4 00D0.97C6.0738 FastEthernet0
10.0.0.5 0007.ECDC.8388 FastEthernet0
ARP Table for PC3
IP Address Hardware Address Interface
10.0.0.1 0002.16AA.3D45 FastEthernet0
/ Play
Capture /
Edit Filters
Show All/None

Program 10

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

Observation:



LAB 6

TELNET

(8) Configure TELNET to access router remotely

TELNET - Used to access remote server
- Simple command line tool that runs on your computer and it allows you to send commands to your server & administrator
- also used to manage other devices like router, switch to check if ports are open or close on the server

PC0
↳ Desktop
↳ IP configuration - Static
↳ IP address - 192.168.1.2
Subnet Mask
Default Gateway - 192.168.1.1

Router 0 - CLI
no ↵ ↵
>enable
conf t
#hostname R1
#enable secret rp
#int Fa0/0
ip add 192.168.1.1 255.255.255.0

vty - Virtual Interface (created such that PC can access)
↳ Virtual Teletype

#no shutdown
↵
#line vty 0 5
#login
#password tp
#exit
#exit
wr
R1# show ip interface brief

Interface	IP address	OK?	Method	Status	Protocol
FastEthernet 0/0	192.168.1.1	Yes	manual	up	up
FastEthernet0/1	unassigned	YES	unset	down	down
Vlan1	unassigned	YES	unset	down	down

→ IN COMMAND PROMPT:

PC> ping 192.168.1.1
Reply from...
Reply from...
Reply from...
PC> telnet 192.168.1.1
Trying...
User Access Verification
password: <password> (Here, tp)
R1> enable
password: <secret password> (Here, rp)

Output:

```
PC>telnet 10.0.0.3
Trying 10.0.0.3 ...
% Connection timed out; remote host not responding
```

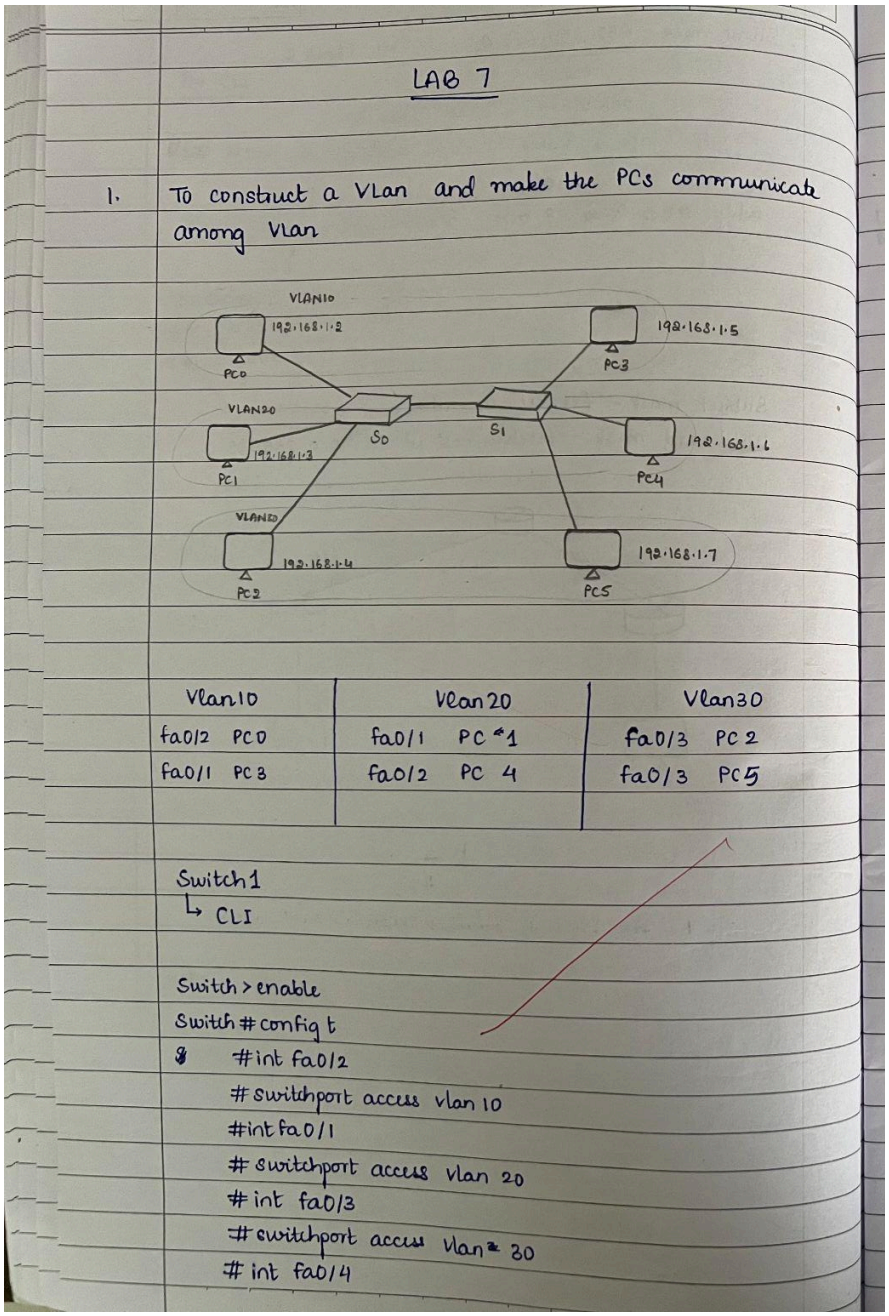
```
PC>telnet 10.0.0.1
Trying 10.0.0.1 ...Open

[Connection to 10.0.0.1 closed by foreign host]
```

```
[Connection to 10.0.0.1 closed by foreign host]
PC>telnet 10.0.0.1
Trying 10.0.0.1 ...Open


User Access Verification

Password:
r1>ping 10.0.0.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 0/0/1
ms

r1>
```

Program 11

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

Observation:



LAB 7

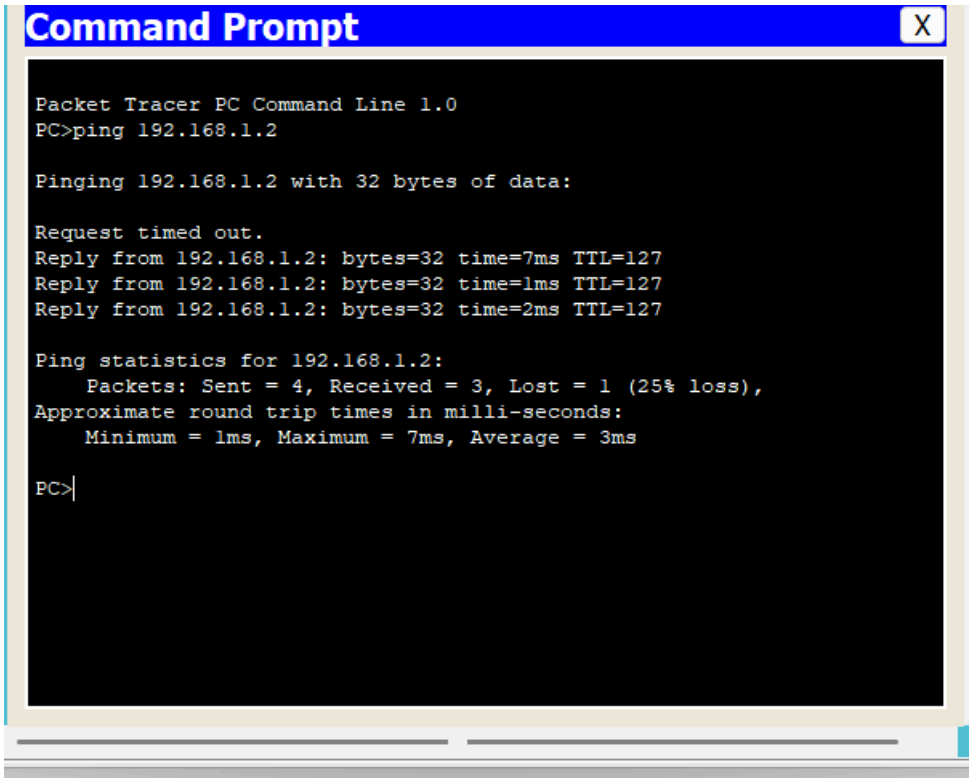
1. To construct a Vlan and make the PCs communicate among Vlan

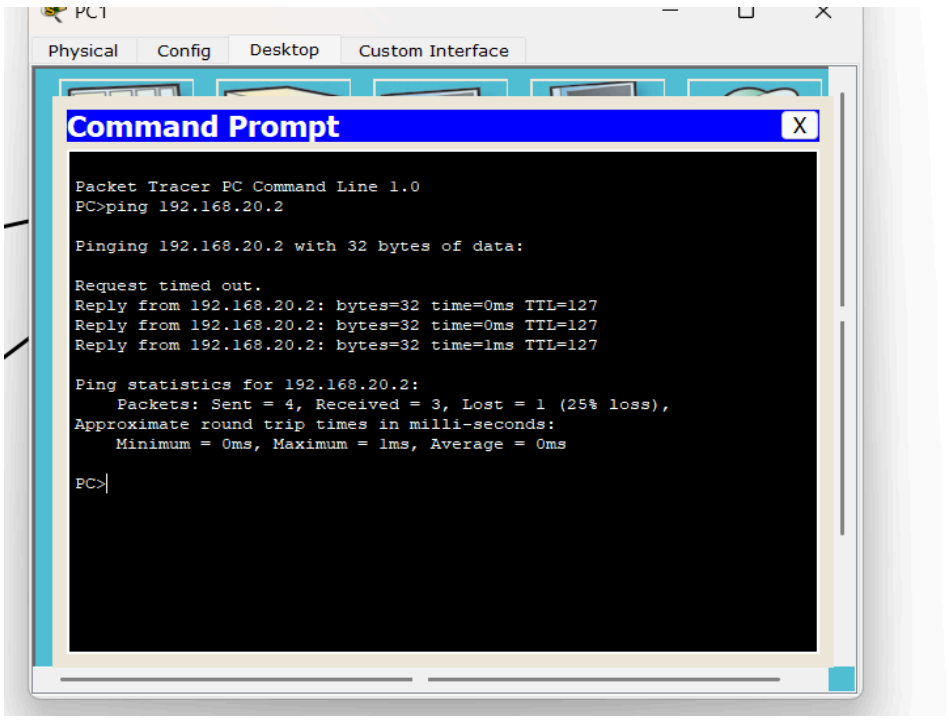
Vlan10	Vlan20	Vlan30
fa0/2 PC0	fa0/1 PC 1	fa0/3 PC 2
fa0/1 PC3	fa0/2 PC 4	fa0/3 PC5

Switch1
↳ CLI

```
Switch>enable
Switch#config t
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)#int fa0/2
Switch(config-if)#switchport access vlan 10
Switch(config-if)#int fa0/1
Switch(config-if)#switchport access vlan 20
Switch(config-if)#int fa0/3
Switch(config-if)#switchport access vlan 30
Switch(config-if)#int fa0/4
```

Output:

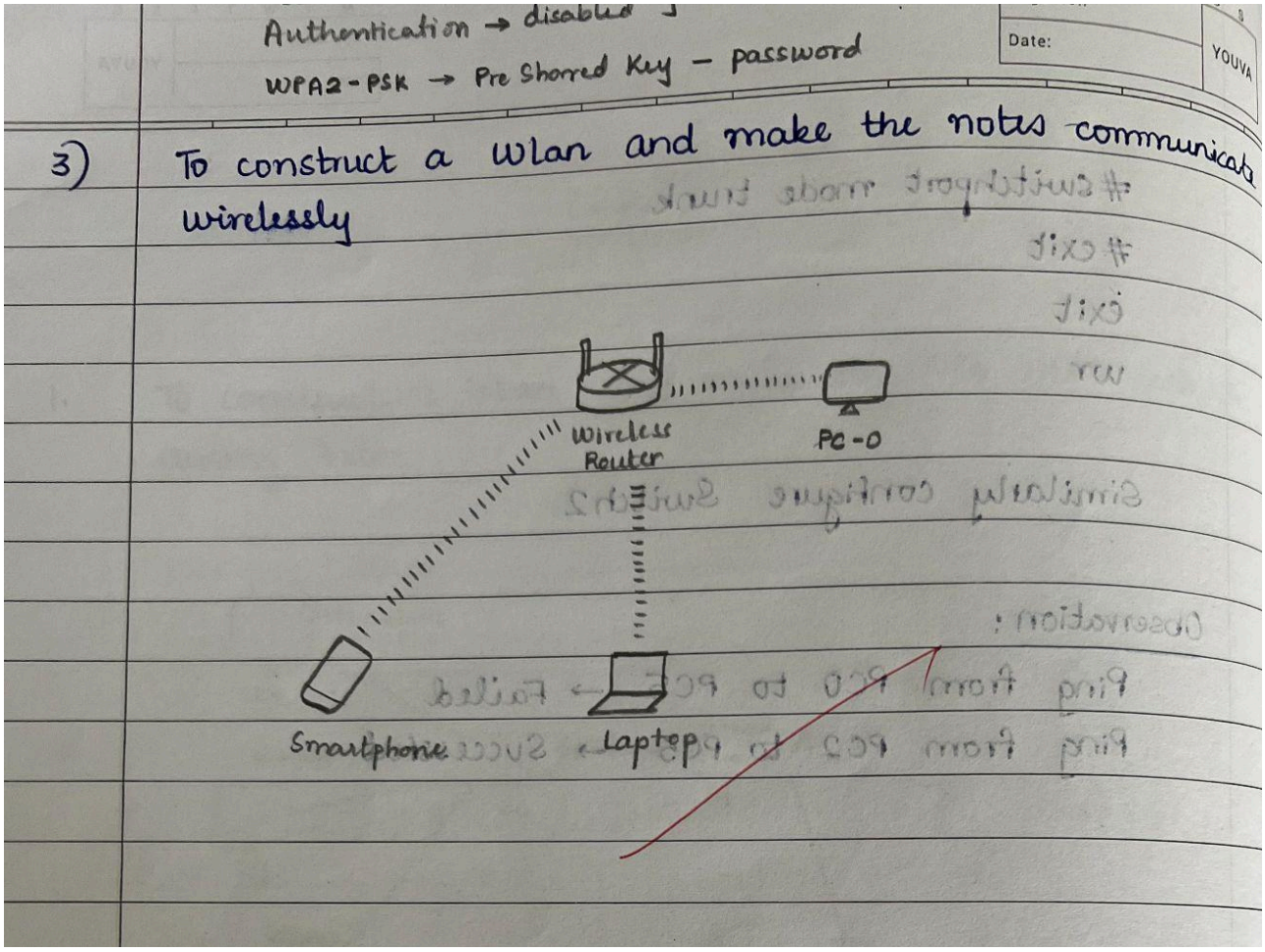

Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 192.168.1.2
Pinging 192.168.1.2 with 32 bytes of data:
Request timed out.
Reply from 192.168.1.2: bytes=32 time=7ms TTL=127
Reply from 192.168.1.2: bytes=32 time=1ms TTL=127
Reply from 192.168.1.2: bytes=32 time=2ms TTL=127
Ping statistics for 192.168.1.2:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 1ms, Maximum = 7ms, Average = 3ms
PC>


PC1
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 192.168.20.2
Pinging 192.168.20.2 with 32 bytes of data:
Request timed out.
Reply from 192.168.20.2: bytes=32 time=0ms TTL=127
Reply from 192.168.20.2: bytes=32 time=0ms TTL=127
Reply from 192.168.20.2: bytes=32 time=1ms TTL=127
Ping statistics for 192.168.20.2:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 1ms, Average = 0ms
PC>

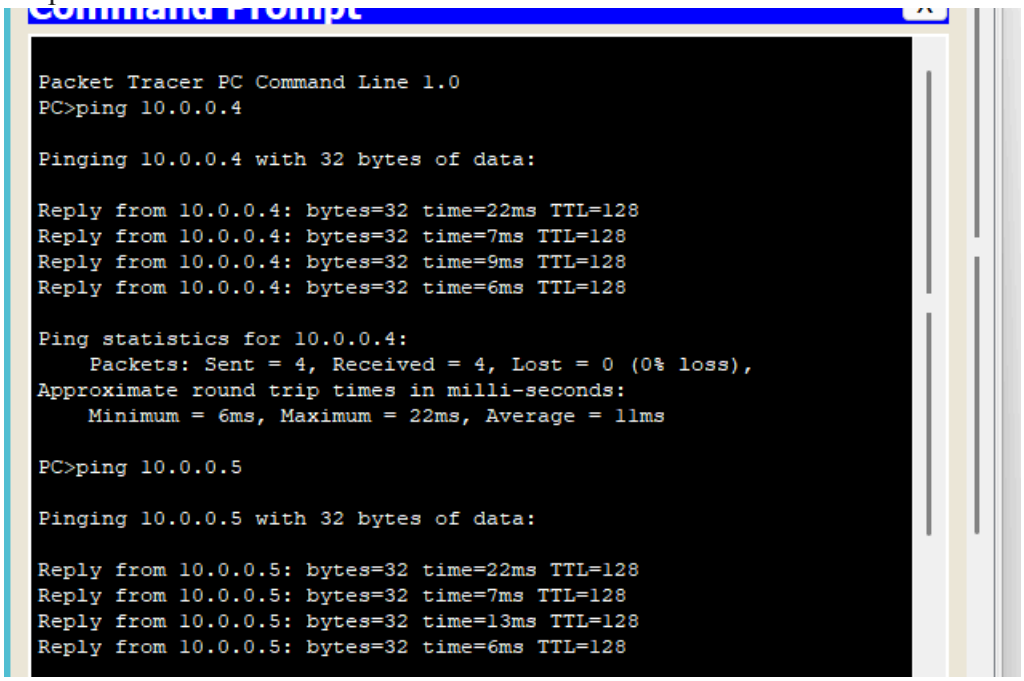
Program 12

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

Observation:



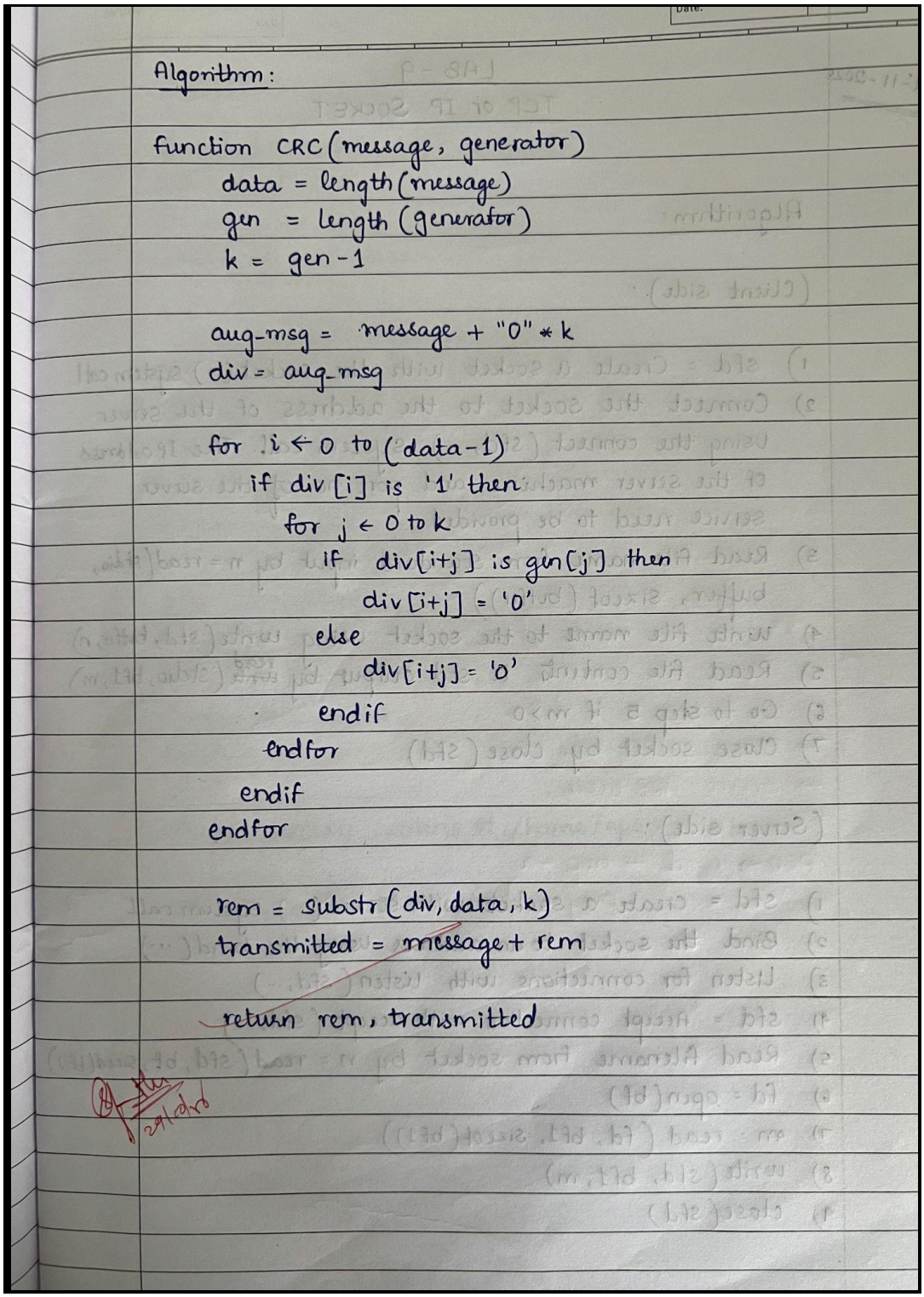
Output:


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=22ms TTL=128
Reply from 10.0.0.4: bytes=32 time=7ms TTL=128
Reply from 10.0.0.4: bytes=32 time=9ms TTL=128
Reply from 10.0.0.4: bytes=32 time=6ms 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 = 6ms, Maximum = 22ms, Average = 11ms
PC>ping 10.0.0.5
Pinging 10.0.0.5 with 32 bytes of data:
Reply from 10.0.0.5: bytes=32 time=22ms TTL=128
Reply from 10.0.0.5: bytes=32 time=7ms TTL=128
Reply from 10.0.0.5: bytes=32 time=13ms TTL=128
Reply from 10.0.0.5: bytes=32 time=6ms TTL=128

Program 13

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

Observation:



Algorithm:

```
function CRC(message, generator)
    data = length(message)
    gen  = length(generator)
    k = gen - 1

    aug_msg = message + "0" * k
    div = aug_msg

    for i ← 0 to (data-1)
      if div[i] is '1' then
        for j ← 0 to k
          if div[i+j] is gen[j] then
            div[i+j] = '0'
          else
            div[i+j] = '0'
          endif
        endfor
      endif
    endfor

    rem = substr(div, data, k)
    transmitted = message + rem

    return rem, transmitted
```

Code:

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

void computeCRC(char data[], char poly[], char remainder[])
{
    char temp[200];
    int dataLen = strlen(data);
    int polyLen = strlen(poly);

    strcpy(temp, data);

    // Append (polyLen - 1) zeros to data
    for (int i = 1; i < polyLen; i++)
        strcat(temp, "0");

    printf("\nData padded (message + zeros): %s\n", temp);

    // Perform binary division
    for (int i = 0; i <= strlen(temp) - polyLen; i++)
    {
        if (temp[i] == '1')
        {
            for (int j = 0; j < polyLen; j++)
                temp[i + j] = (temp[i + j] == poly[j]) ? '0' : '1';
        }
    }
    strcpy(remainder, temp + (strlen(temp) - (polyLen - 1)));
}

int checkReceived(char received[], char poly[])
{
    char temp[200];
    int polyLen = strlen(poly);

    strcpy(temp, received);

    // Perform division on received message
    for (int i = 0; i <= strlen(temp) - polyLen; i++)
    {
        if (temp[i] == '1')
        {
            for (int j = 0; j < polyLen; j++)
                temp[i + j] = (temp[i + j] == poly[j]) ? '0' : '1';
        }
    }
    for (int i = strlen(temp) - (polyLen - 1); i < strlen(temp); i++)
        if (temp[i] == '1')
            return 0; // Error detected
```

```
    return 1; // No error
}

int main()
{
    char message[100], poly[30], crc[30], transmitted[150], received[150];

    printf("Enter the message bits       : ");
    scanf("%s", message);

    printf("Enter the polynomial (g(x))  : ");
    scanf("%s", poly);

    computeCRC(message, poly, crc);

    printf("CRC value (remainder)        : %s\n", crc);

    // Form transmitted message
    strcpy(transmitted, message);
    strcat(transmitted, crc);

    printf("Message to be transmitted    : %s\n", transmitted);

    // Receiver side
    printf("\nEnter received message bits (can modify 1 bit to test error): ");
    scanf("%s", received);

    if (checkReceived(received, poly))
        printf("\n✅ No Error: Message received correctly.\n");
    else
        printf("\n❌ ERROR detected in received message!\n");

    return 0;
}
```

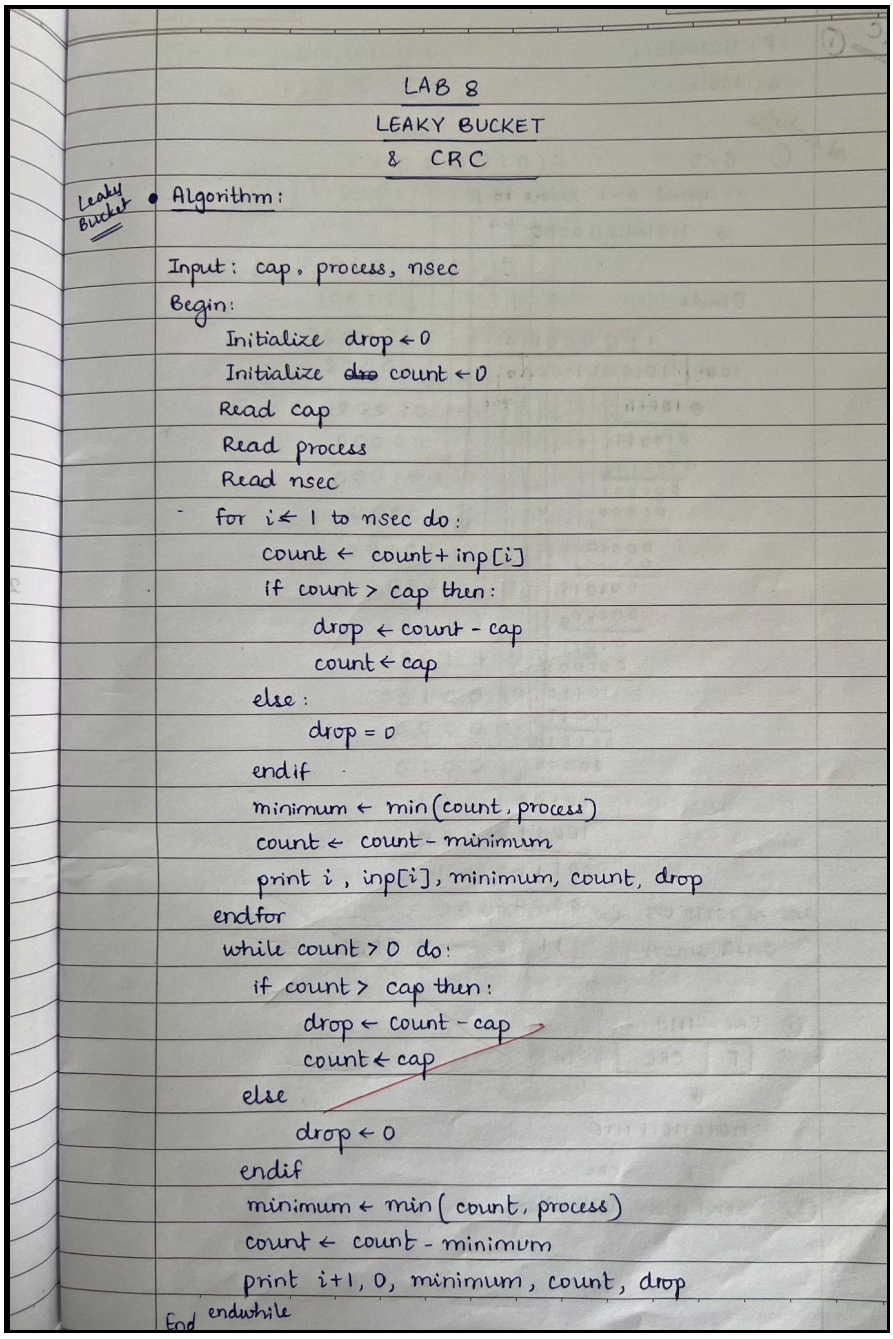
Output:

```
Enter the message bits       : 101101
Enter the polynomial (g(x))  : 1101

Data padded (message + zeros): 101101000
CRC value (remainder)        : 101
Message to be transmitted    : 101101101

Enter received message bits (can modify 1 bit to test error): 101101101

✅ No Error: Message received correctly.
```

Program 14

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

Observation:



LAB 8

LEAKY BUCKET

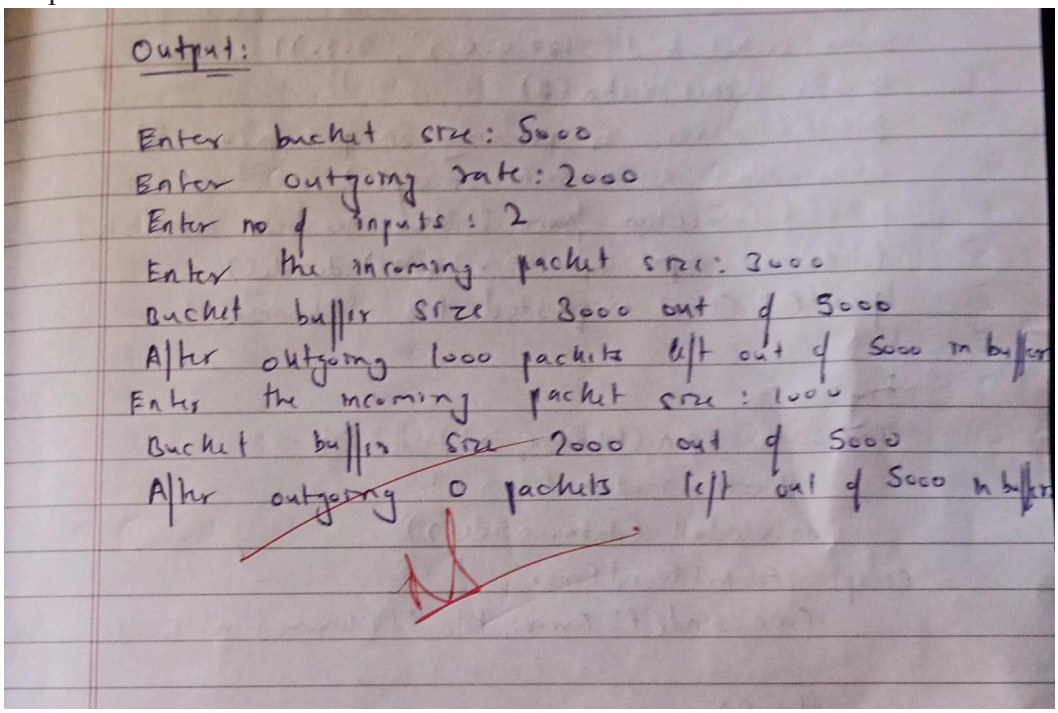
& CRC

Leaky Bucket

• Algorithm:

```
Input: cap, process, nsec
Begin:
    Initialize drop ← 0
    Initialize count ← 0
    Read cap
    Read process
    Read nsec
    for i ← 1 to nsec do:
        count ← count + inp[i]
        if count > cap then:
            drop ← count - cap
            count ← cap
        else:
            drop = 0
        endif
        minimum ← min(count, process)
        count ← count - minimum
        print i , inp[i], minimum, count, drop
    endfor
    while count > 0 do:
        if count > cap then:
            drop ← count - cap
            count ← cap
        else
            drop ← 0
        endif
        minimum ← min( count, process)
        count ← count - minimum
        print i+1, 0, minimum, count, drop
End endwhile
```

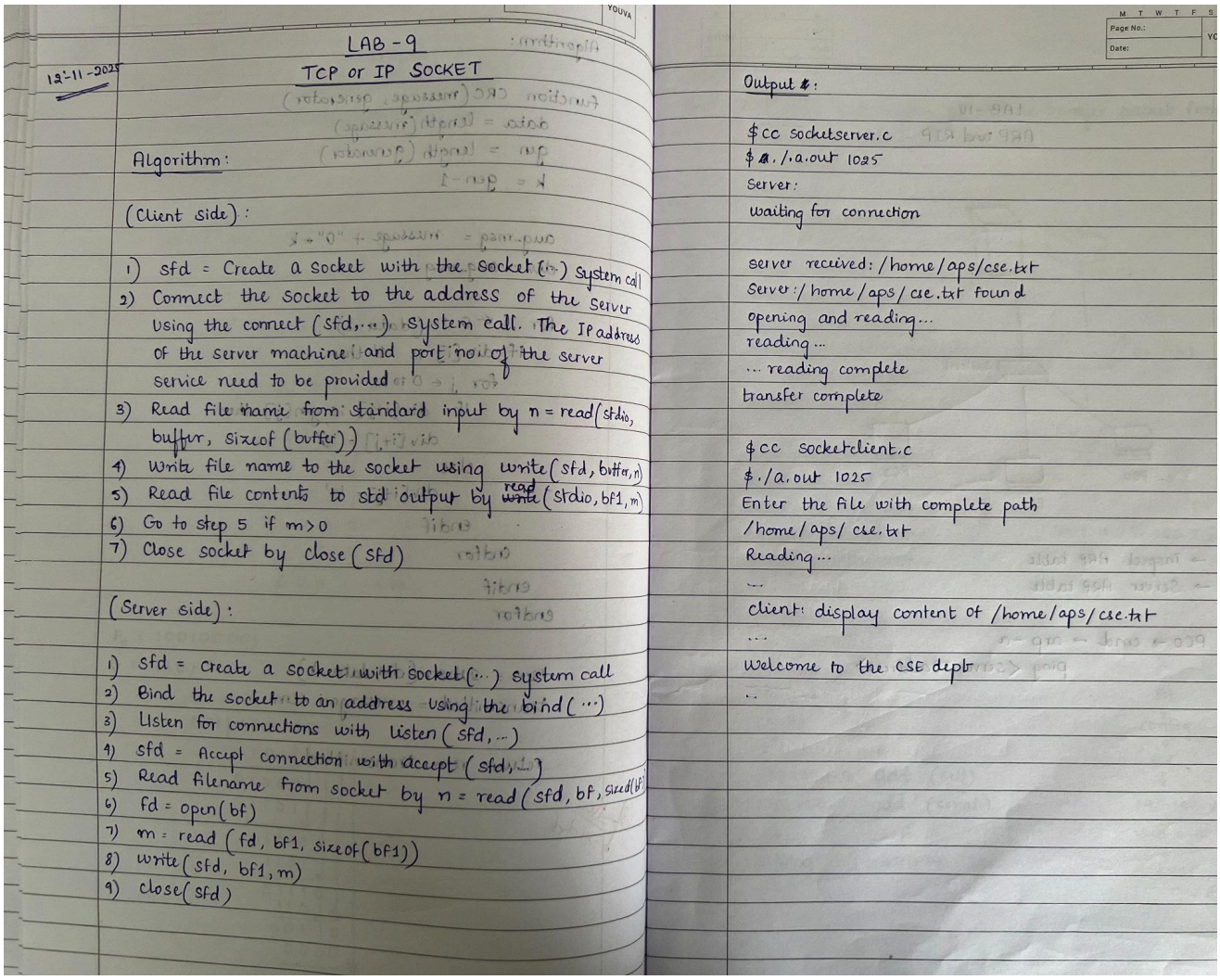
Output:


Output:
Enter bucket size: 5000
Enter outgoing rate: 2000
Enter no of inputs: 2
Enter the incoming packet size: 3000
Bucket buffer size 3000 out of 5000
After outgoing 1000 packets left out of 5000 in buffer
Enter the incoming packet size: 1000
Bucket buffer size 2000 out of 5000
After outgoing 0 packets left out of 5000 in buffer

Program 15

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

Observation + Output:



LAB - 9

12-11-2025

TCP or IP SOCKET

Algorithm:

(Client side):

1) sfd = Create a socket with the socket(...) system call
2) Connect the socket to the address of the server using the connect(sfd,...) system call. The IP address of the server machine and port no. of the server service need to be provided
3) Read file name from standard input by n = read(stdio, buffer, sizeof(buffer))
4) Write file name to the socket using write(sfd, buffer, n)
5) Read file contents to std output by ~~write~~ read(stdio, bf1, m)
6) Go to step 5 if m>0
7) Close socket by close(sfd)

(Server side):

1) sfd = Create a socket with socket(...) system call
2) Bind the socket to an address using the bind(...)
3) Listen for connections with listen(sfd,...)
4) sfd = Accept connection with accept(sfd,...)
5) Read filename from socket by n = read(sfd, bf, sizeof(bf))
6) fd = open(bf)
7) m = read(fd, bf1, sizeof(bf1))
8) write(sfd, bf1, m)
9) close(sfd)

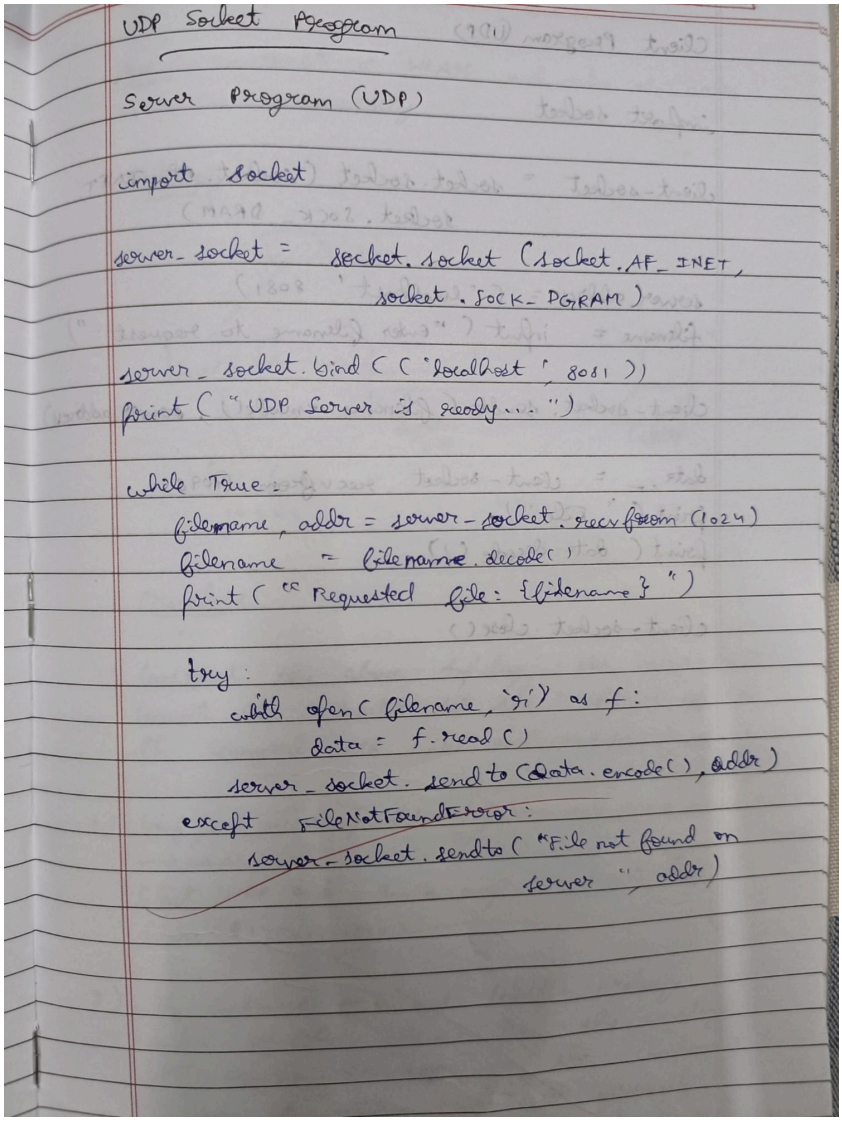
Output:

```
$cc socketserver.c
$./a.out 1025
Server:
Waiting for connection

server received: /home/aps/cse.txt
Server: /home/aps/cse.txt found
opening and reading...
reading...
... reading complete
transfer complete

$cc socketclient.c
$./a.out 1025
Enter the file with complete path
/home/aps/cse.txt
Reading...
...
client: display content of /home/aps/cse.txt
...
Welcome to the CSE dept
..
```

Program 16

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

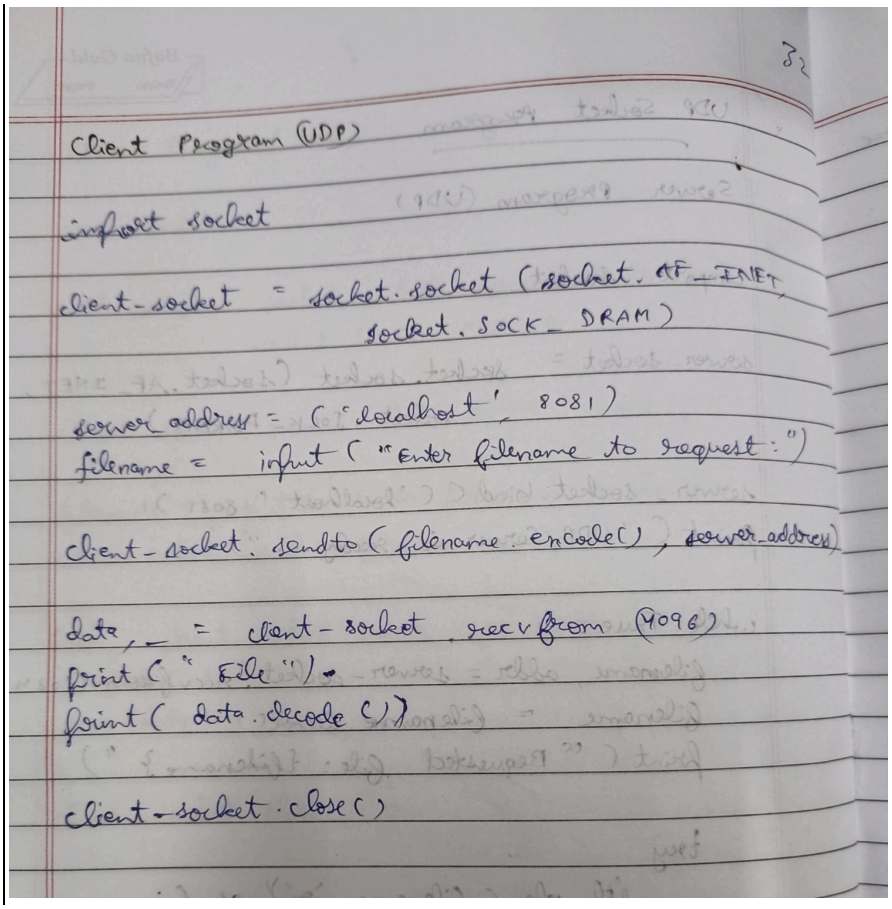
Observation:



UDP Socket Program

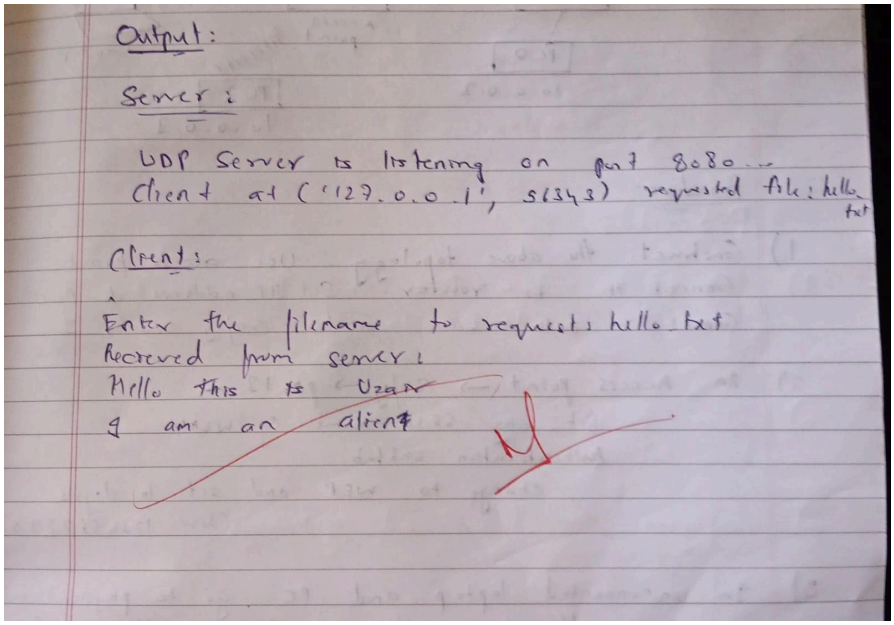
Server Program (UDP)

```
import socket

server_socket = socket.socket(socket.AF_INET,
                              socket.SOCK_DGRAM)

server_socket.bind(('localhost', 8081))
print("UDP Server is ready...")

while True:
    filename, addr = server_socket.recvfrom(1024)
    filename = filename.decode()
    print("Requested file: {filename}")

    try:
        with open(filename, 'r') as f:
            data = f.read()
        server_socket.sendto(data.encode(), addr)
    except FileNotFoundError:
        server_socket.sendto("File not found on
                             server", addr)
```



```
Client Program (UDP)

import socket

client_socket = socket.socket (socket.AF_INET,
                               socket.SOCK_DRAM)

server_address = ('localhost', 8081)
filename = input ("Enter filename to request:")

client_socket.sendto (filename.encode(), server_address)

data, _ = client_socket.recvfrom (4096)
print ("File:")
print (data.decode())

client_socket.close()
```

Output:



```
Output:

Server:

UDP Server is listening on port 8080...
Client at ('127.0.0.1', 51343) requested file: hello.txt

Client:

Enter the filename to request: hello.txt
Received from server:
Hello this is Uzair
I am an alien
```