VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

Eashan Jain V (1BM23CS098)

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 **Eashan Jain V(1BM23CS098),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements of the above-mentioned subject and the work prescribed for the said degree.

Sarala D V Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

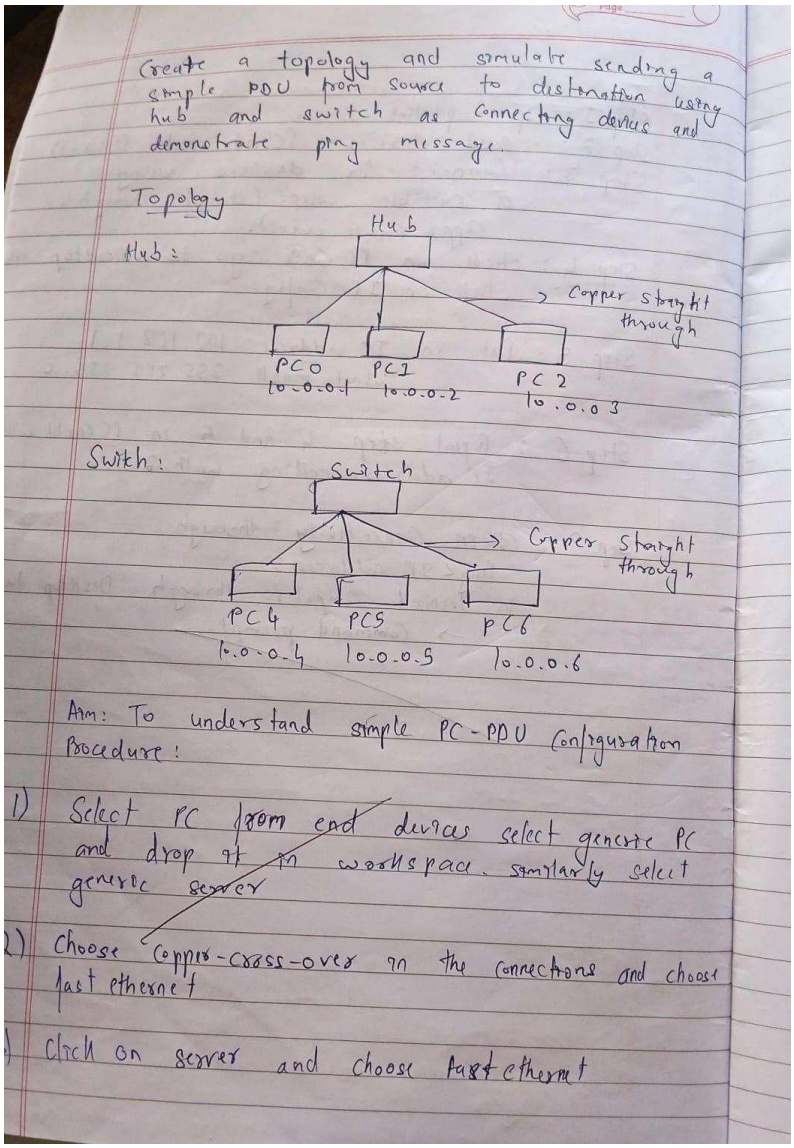
Index

Github Link : github.com/navirath/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 message.

Observation:



4) Click on PC and go to config tab. Set IP address to 10.0.0.1 and click on subnetMask
5) Repeat same step and set IP address for server
6) In simulation mode in edit filters click only ICMP
7) Add simple PDU from PC to server
8) Click on auto capture/play

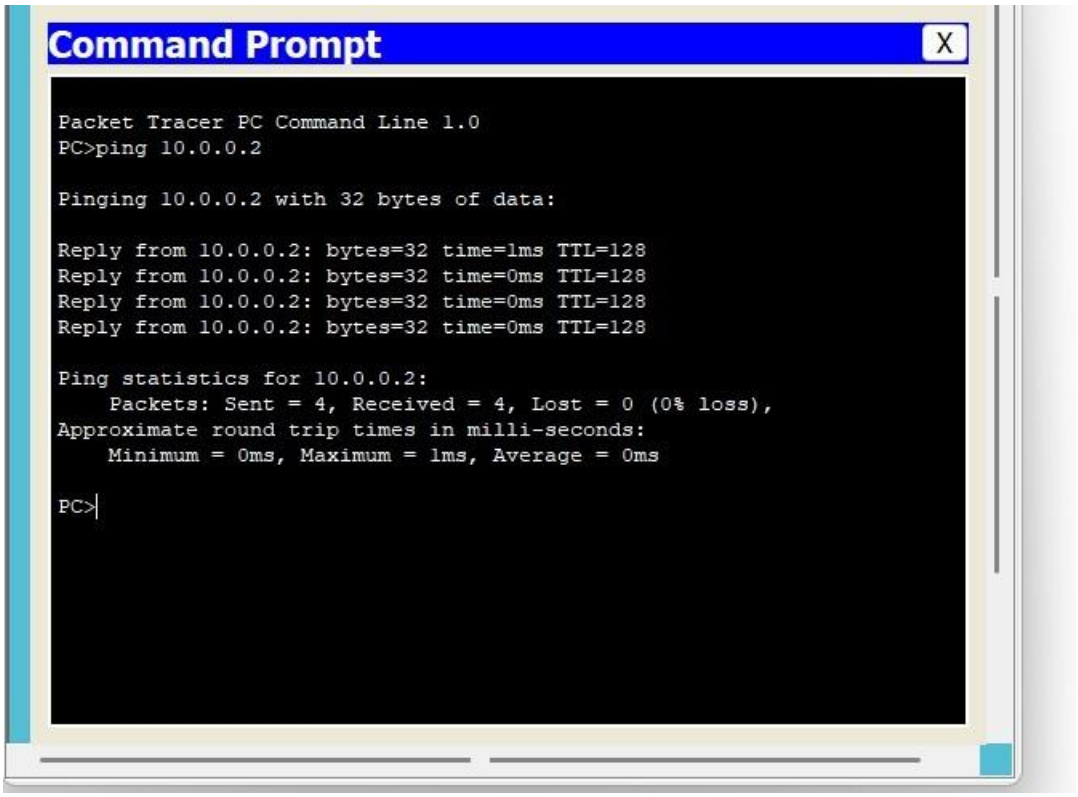
Hub -
Procedure -

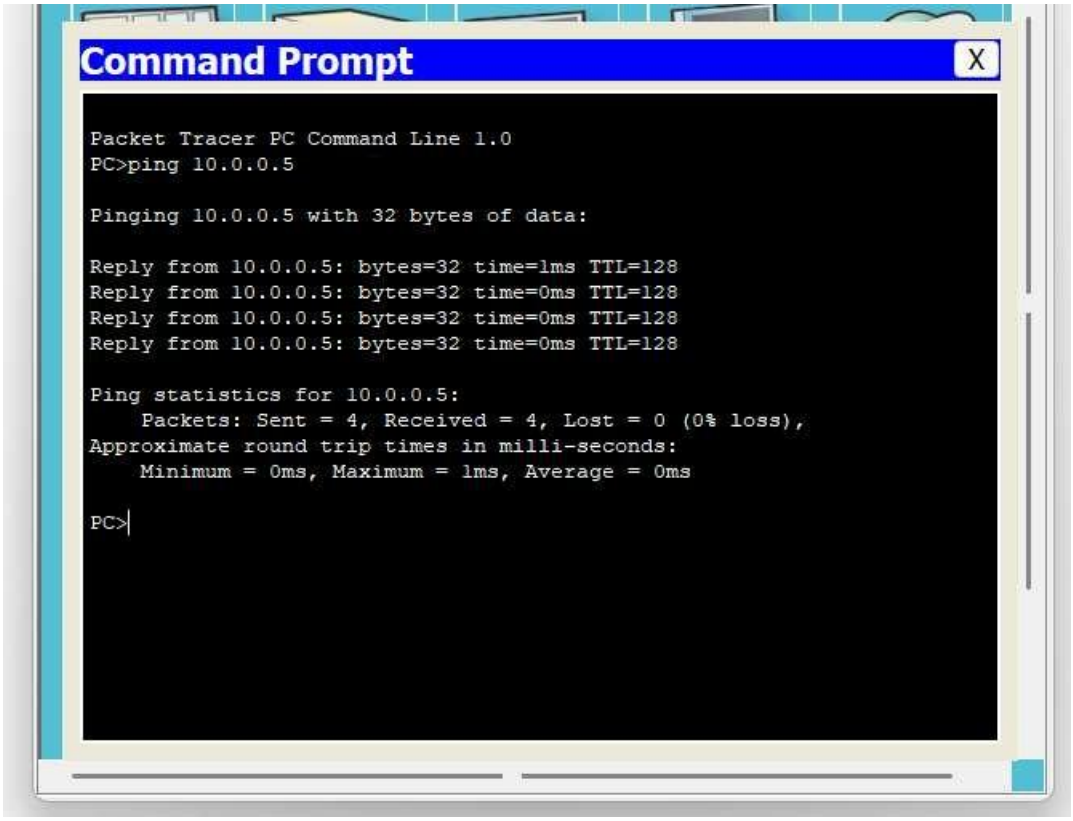
1) Select the end devices and change their IP addresses suitably
2) Select hub as the connecting device
3) Select copper straight through as the connecting wire between end devices and hub
4) Connect the fastethernet to hub ports
5) Select the message and first click on source device and destination device
6) Observe the packet transmission and acknowledgement recieving procedure.

Switch -
Procedure -

1) Select the end devices and change their IP addresses suitably
2) Select switch as the connecting device
3) Slect copper straight through as the connecting wire between end devices and hub.
4) Connect fastethernets to hub ports
5) Select the msg and first click on source device and then the distination device
6) Observe the packet transmission and acknowledgement recieving procedure

Output:

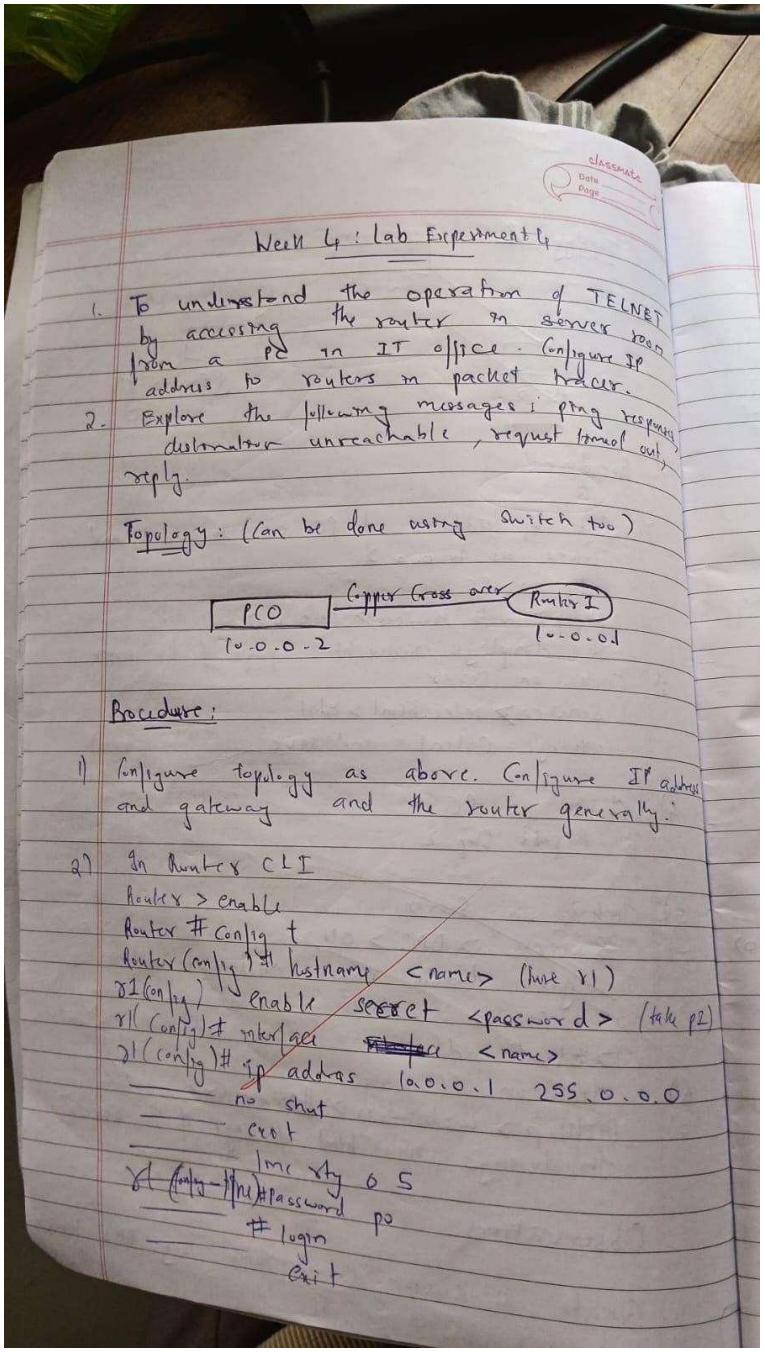

Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.2
Pinging 10.0.0.2 with 32 bytes of data:
Reply from 10.0.0.2: bytes=32 time=1ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128
Reply from 10.0.0.2: bytes=32 time=0ms TTL=128
Ping 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:



Week 4 : Lab Experiment 4

1. To understand the operation of TELNET by accessing the router in server room from a PC in IT office. Configure IP address to routers in packet tracer.
2. Explore the following messages: ping responses, destination unreachable, request timed out, reply.

Topology: (Can be done using Switch too)

PC0 — Copper Cross over — Router 1

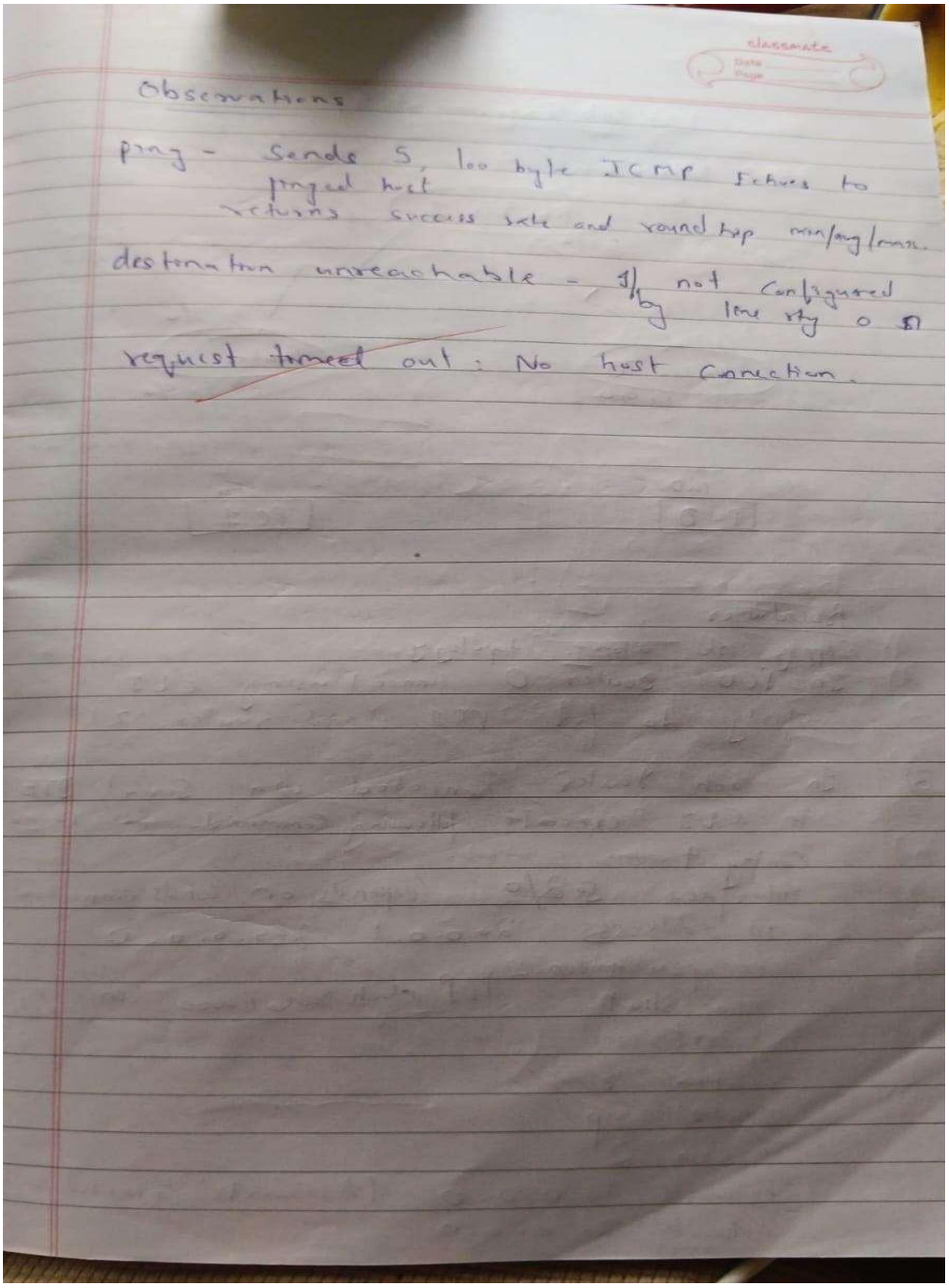
10.0.0.2 — 10.0.0.1

Procedure:

1) Configure topology as above. Configure IP address and gateway and the router generally.

2) In Router CLI

Router > enable
Router # Config t
Router (config) # hostname <name> (here r1)
r1 (config) enable secret <password> (take p2)
r1 (config) # interface <name>
r1 (config) # ip address 10.0.0.1 255.0.0.0
no shut
exit
line vty 0 5
password p0
login
exit

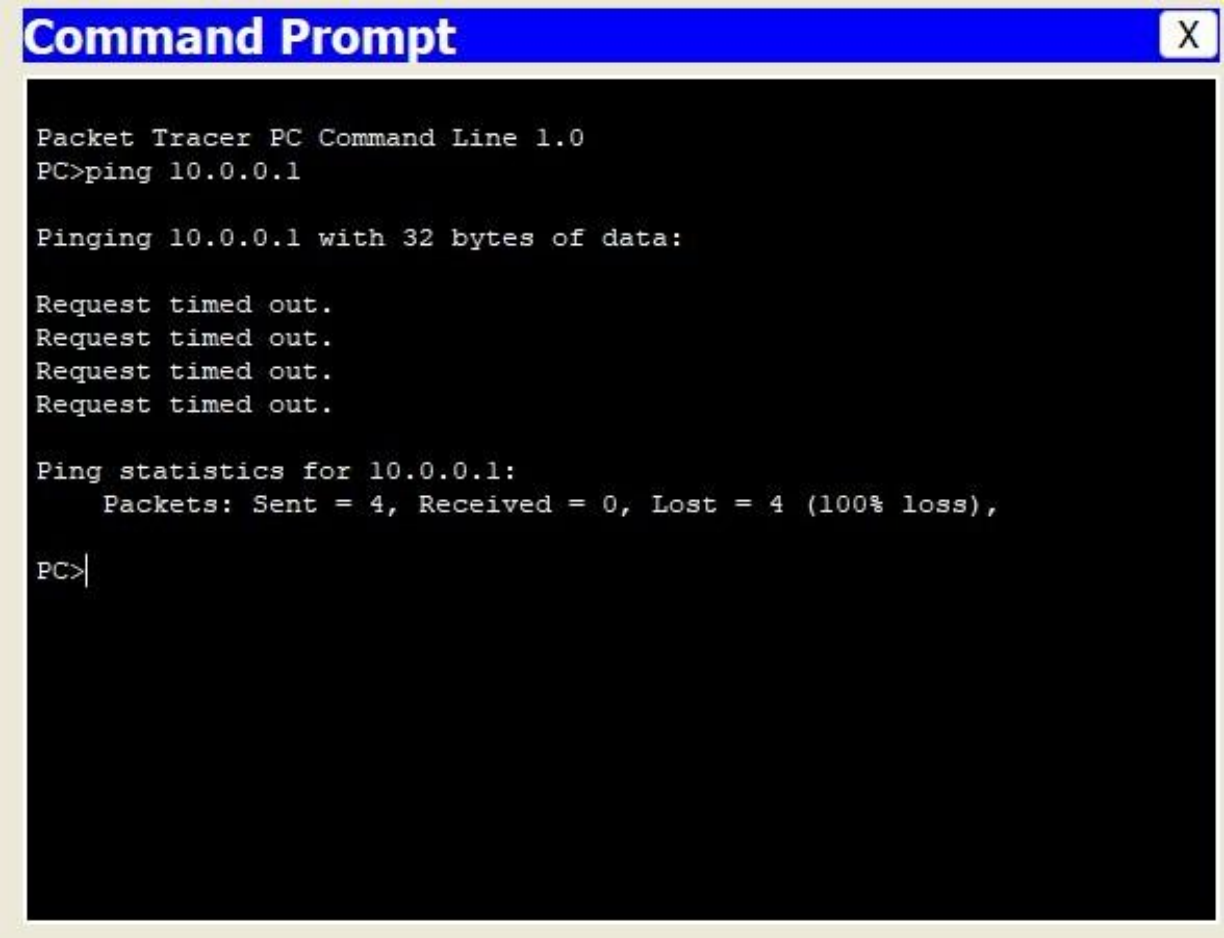

Observations

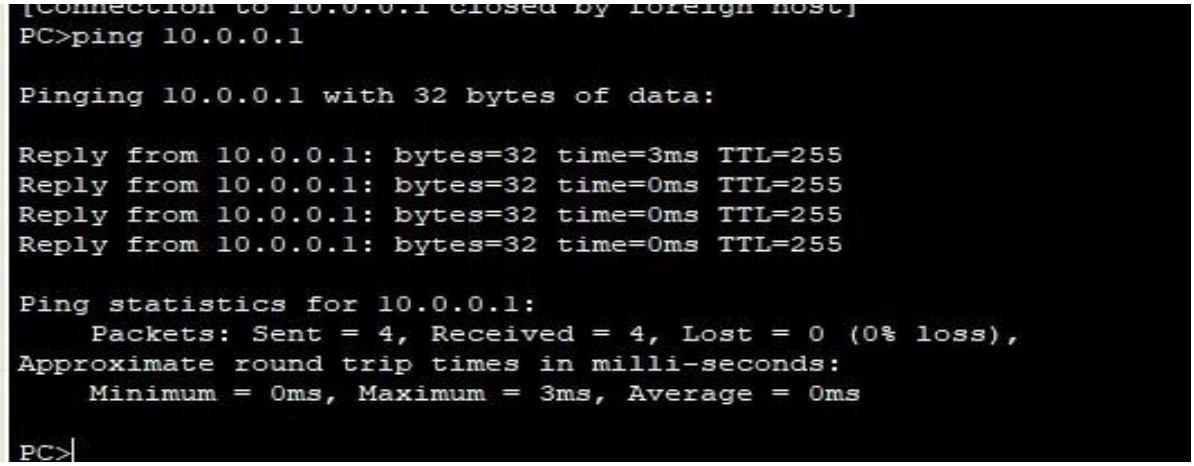
ping - Sends 5, 100 byte ICMP Echos to pinged host
returns success rate and round trip min/avg/max.

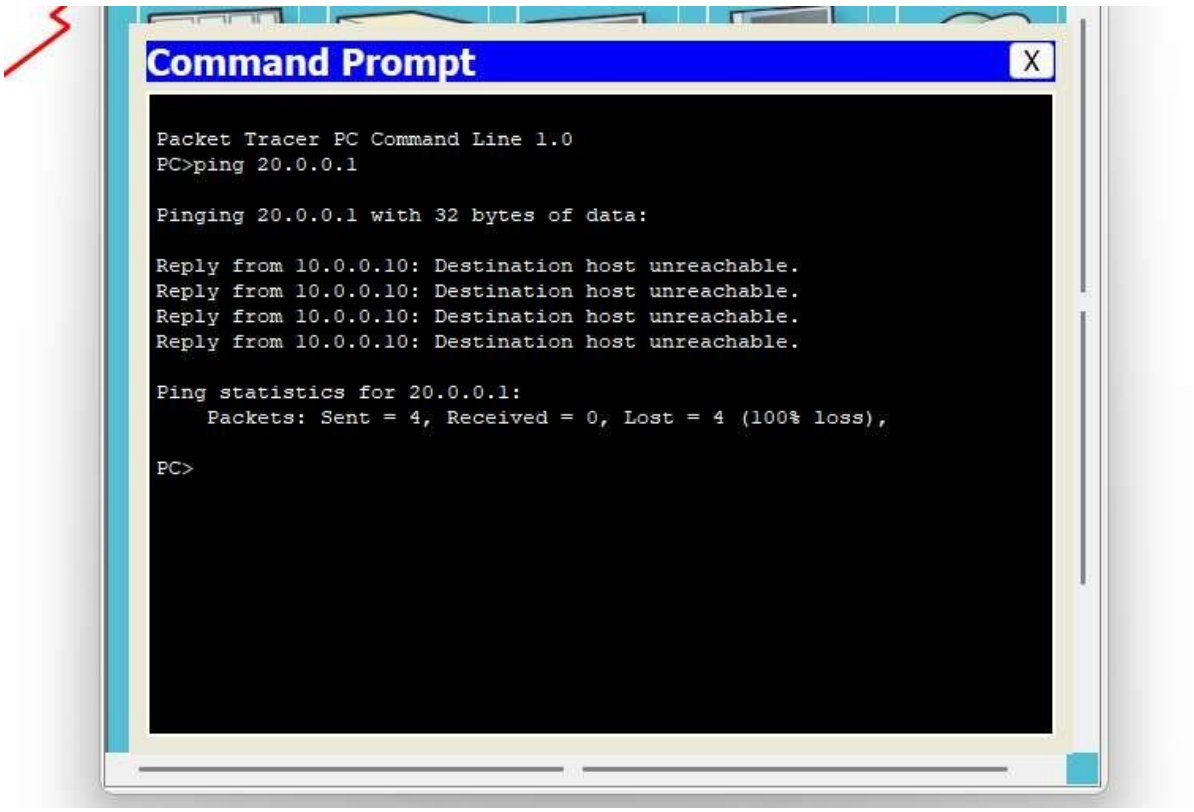
destination unreachable - if not configured by ip rip 0 0

request timed out : No host connection.

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>

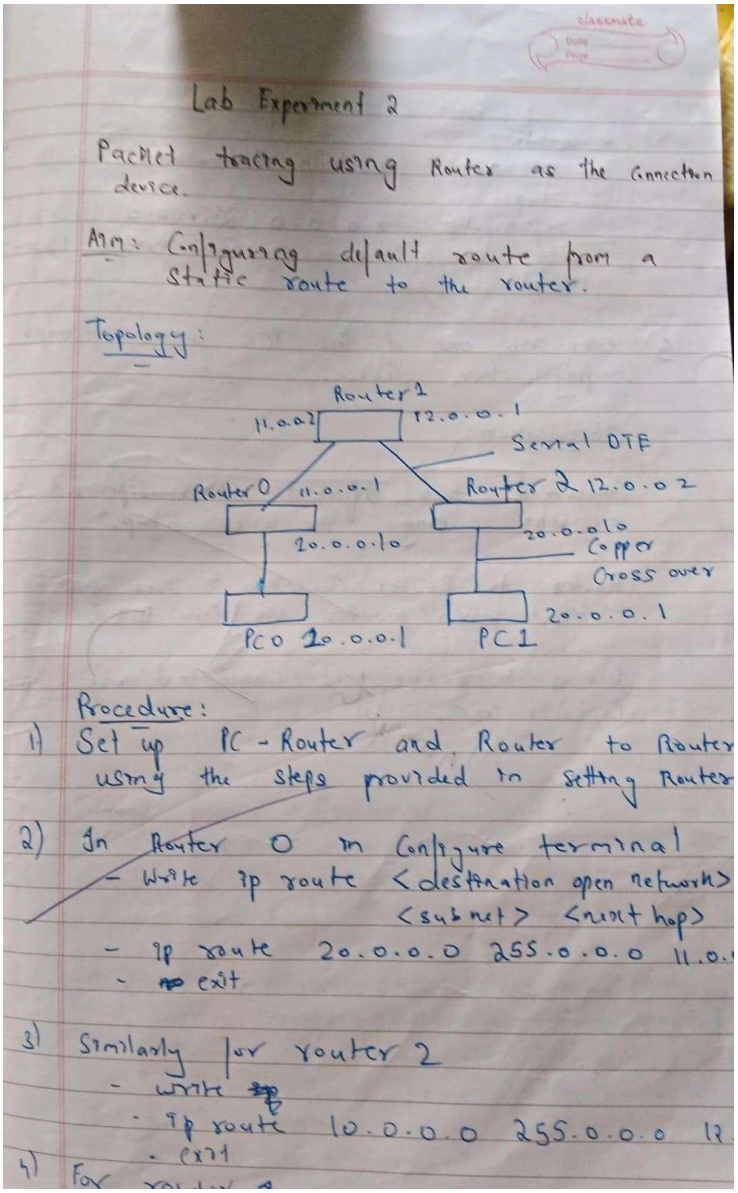

PC>ping 10.0.0.1
Pinging 10.0.0.1 with 32 bytes of data:
Reply from 10.0.0.1: bytes=32 time=3ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Reply from 10.0.0.1: bytes=32 time=0ms TTL=255
Ping statistics for 10.0.0.1:
Packets: Sent = 4, Received = 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 IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply

Observation:



Lab Experiment 2

Packet tracing using Router as the Connection device.

Aim: Configuring default route from a static route to the router.

Topology:

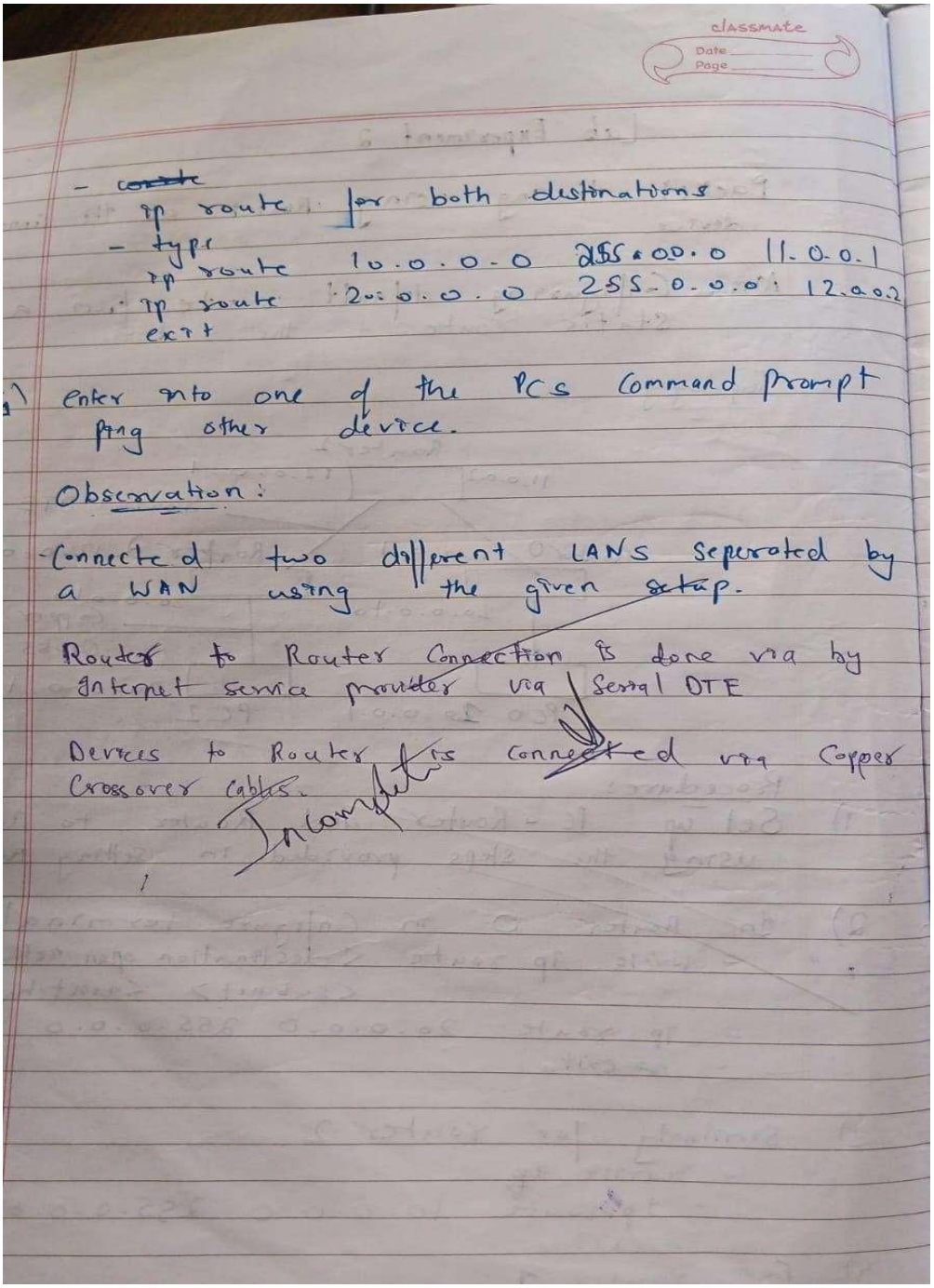
Procedure:

1) Set up PC - Router and Router to Router using the steps provided in setting Router

2) In Router 0 in Configure terminal
   - Write ip route <destination open network> <subnet> <next hop>
   - ip route 20.0.0.0 255.0.0.0 11.0.
   - exit

3) Similarly for router 2
   - write
   - ip route 10.0.0.0 255.0.0.0 12
   - exit

4) For



- ~~create~~
  ip route for both destinations
- type
  ip route 10.0.0.0 255.00.0 11.0.0.1
  ip route 20.0.0.0 255.0.0.0 12.00.2
  exit

1) Enter into one of the PCs Command prompt ping other device.

Observation:

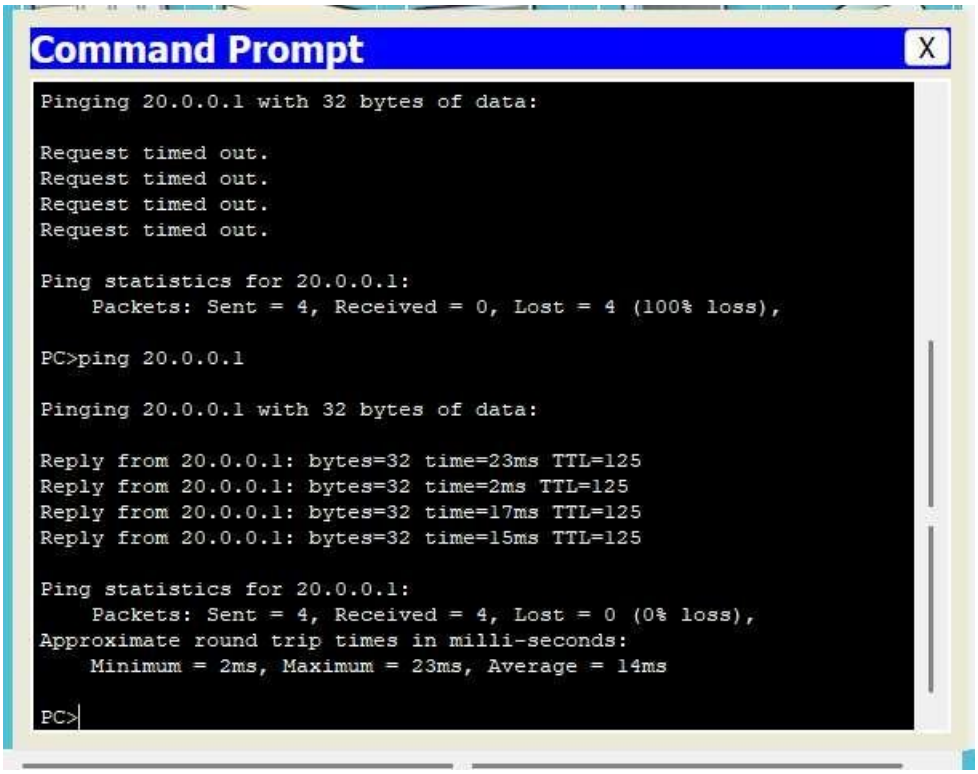
- Connected two different LANs seperated by a WAN using the given ~~setup~~.

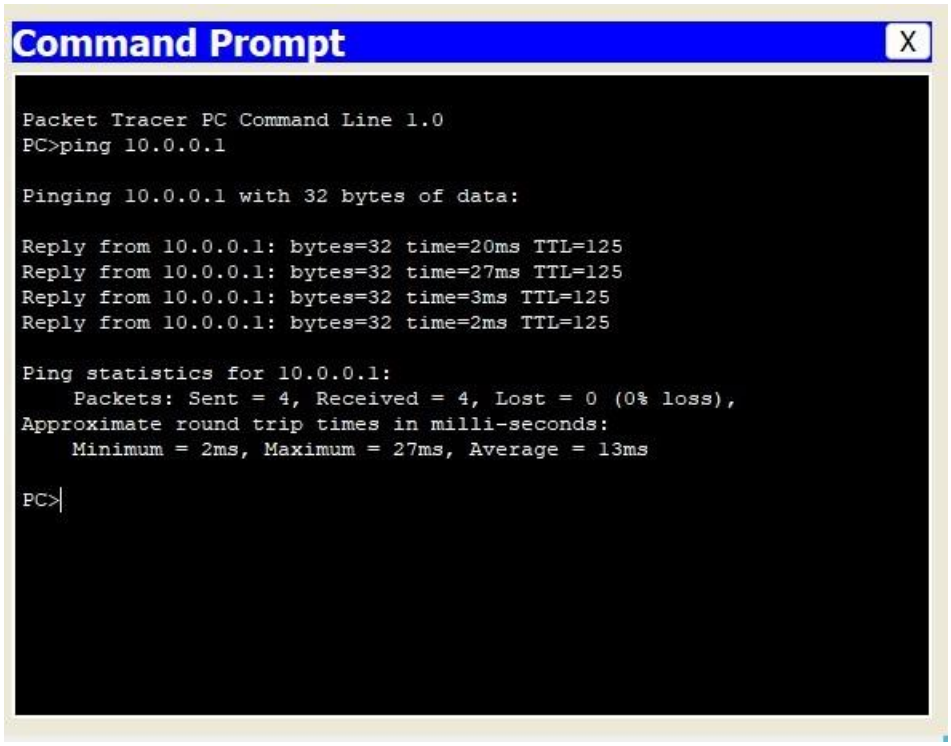
Router to Router Connection is done via by internet service provider via Serial DTE

Devices to Router is connected via Copper Crossover Cables.

Incomplete

Output:

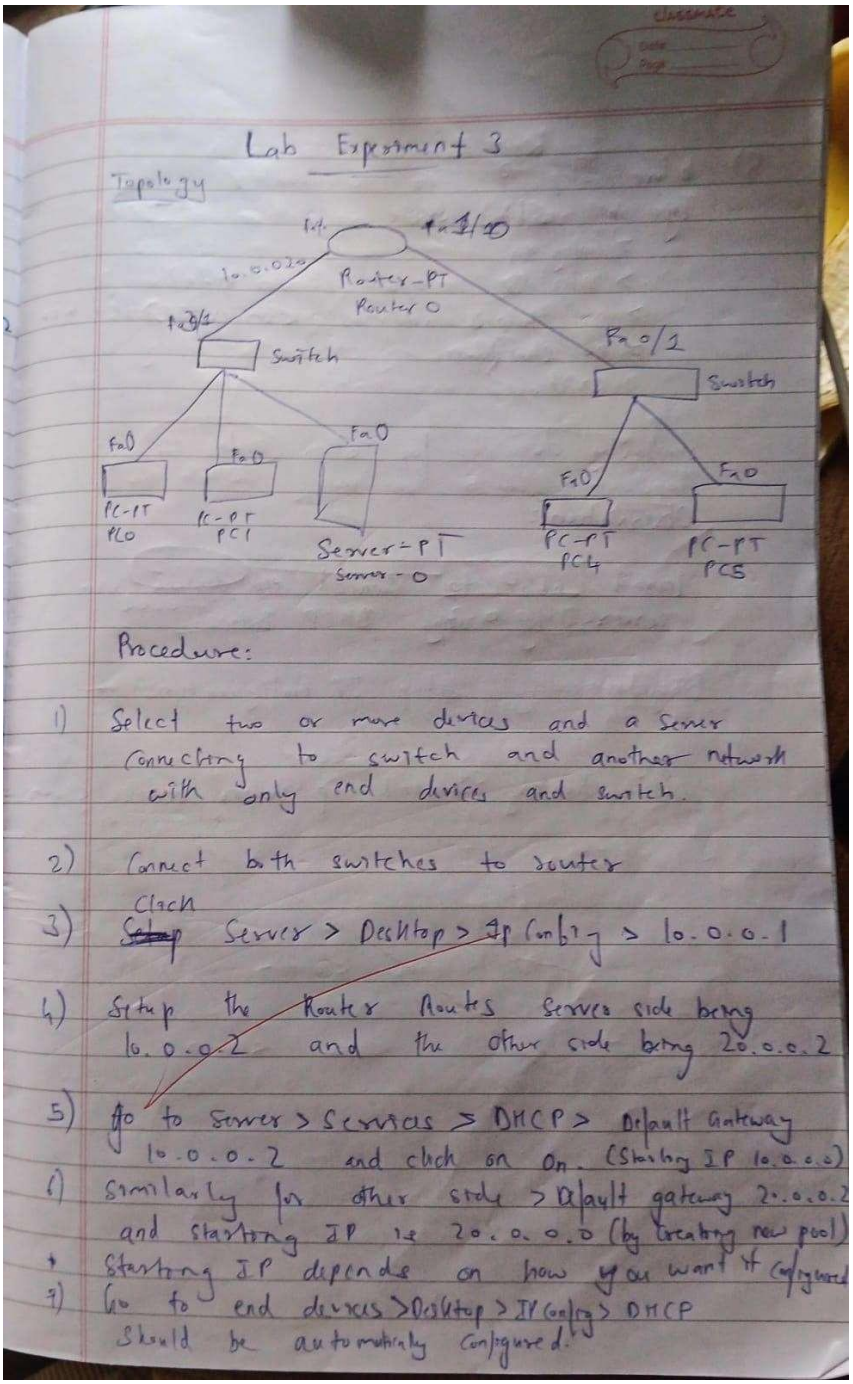

Command Prompt
X
Pinging 20.0.0.1 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 20.0.0.1:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>ping 20.0.0.1
Pinging 20.0.0.1 with 32 bytes of data:
Reply from 20.0.0.1: bytes=32 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:



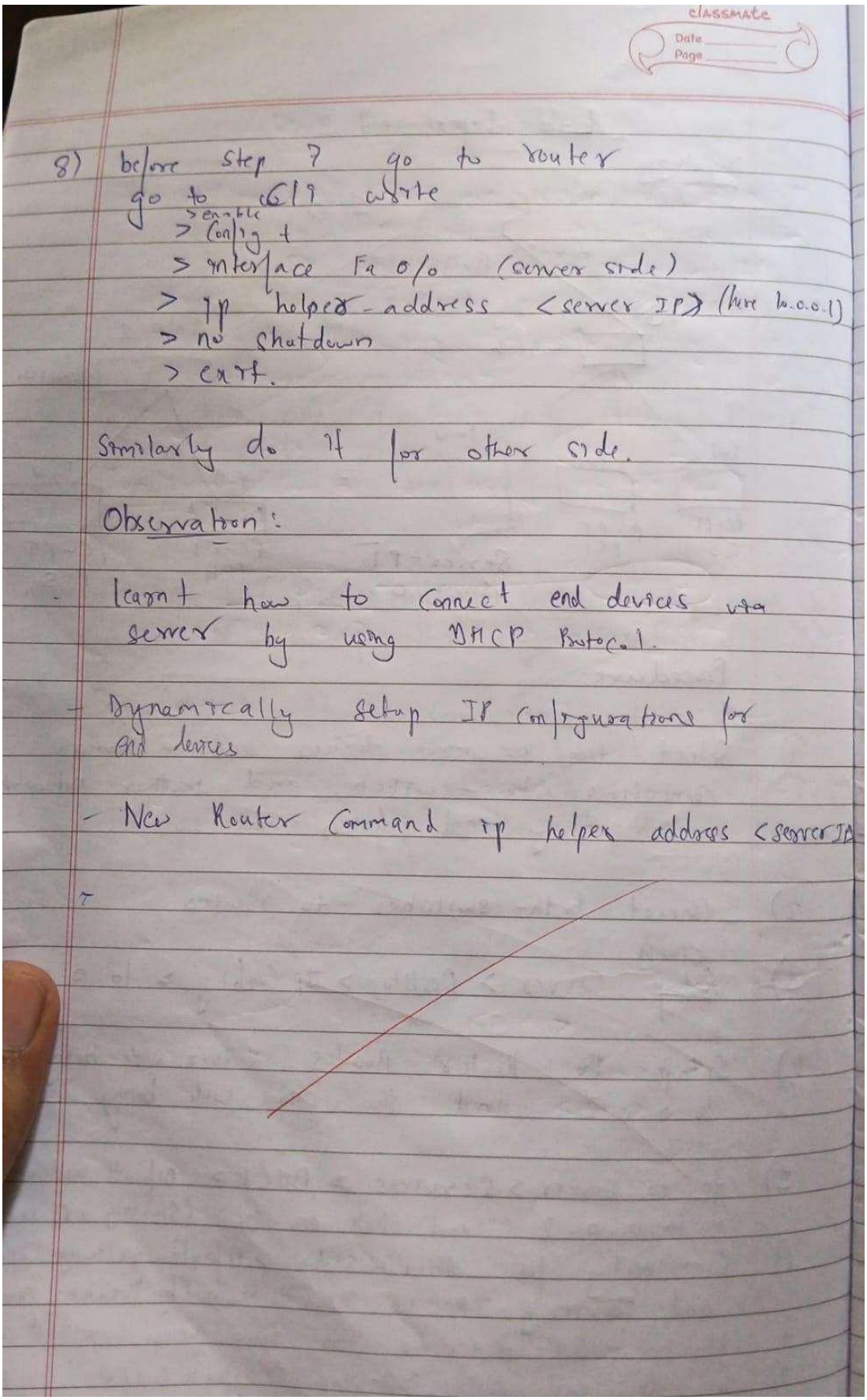
8) before step 7 go to router
go to CLI write
> enable
> config t
> interface Fa 0/0 (server side)
> ip helper-address <server IP> (here 10.0.0.1)
> no shutdown
> exit.

Similarly do it for other side.

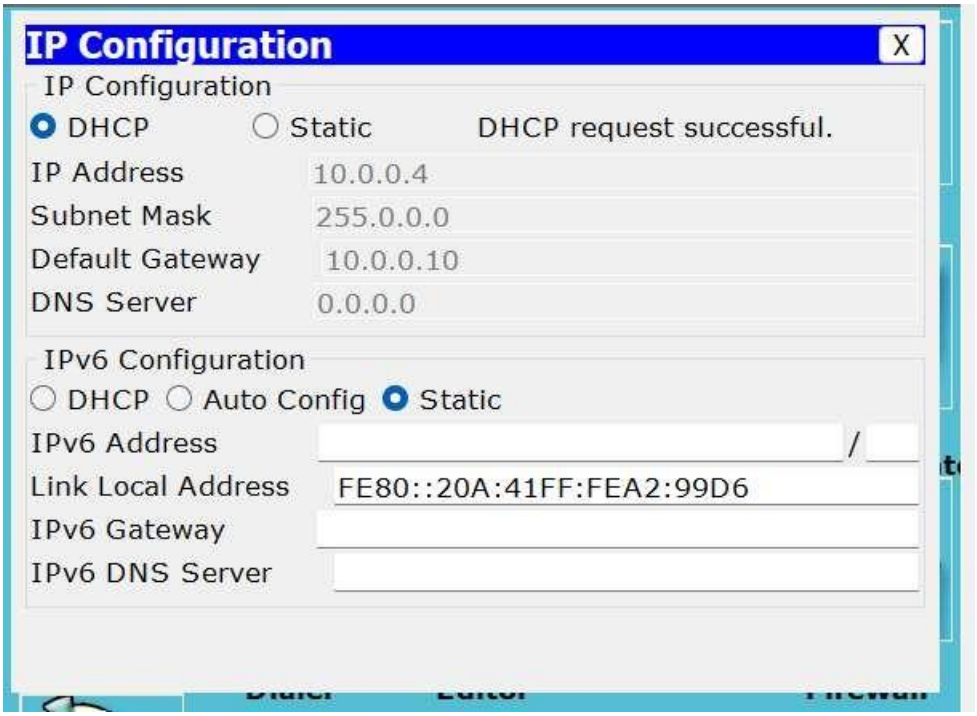
Observation:

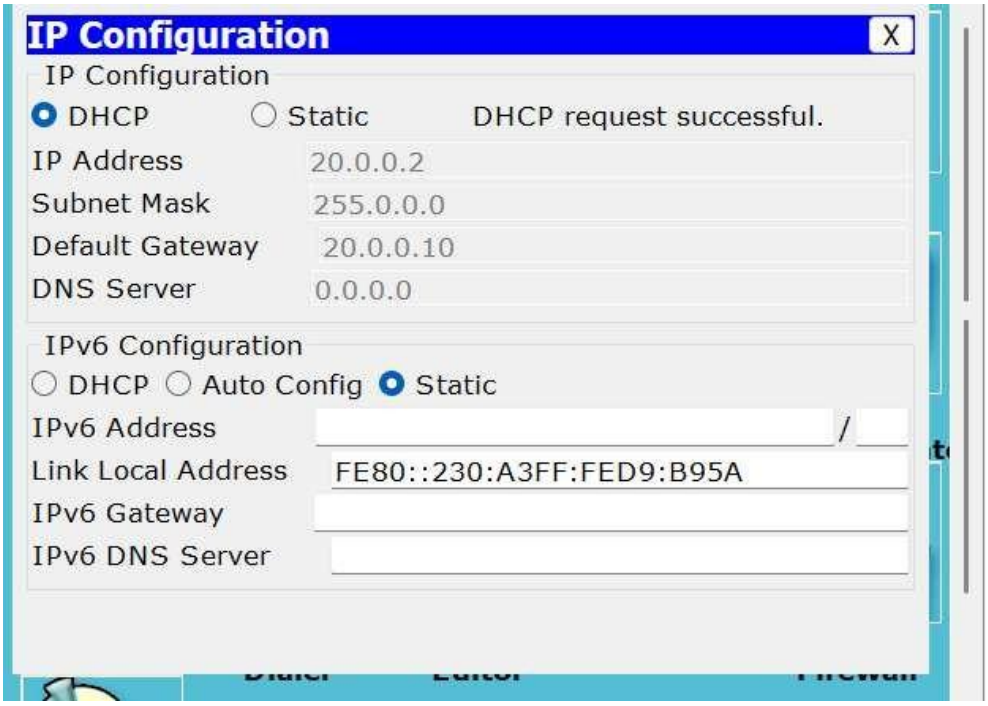
- learnt how to connect end devices via server by using DHCP Protocol.
- Dynamically setup IP configurations for end devices
- New Router Command ip helper address <server IP>

7



Output:

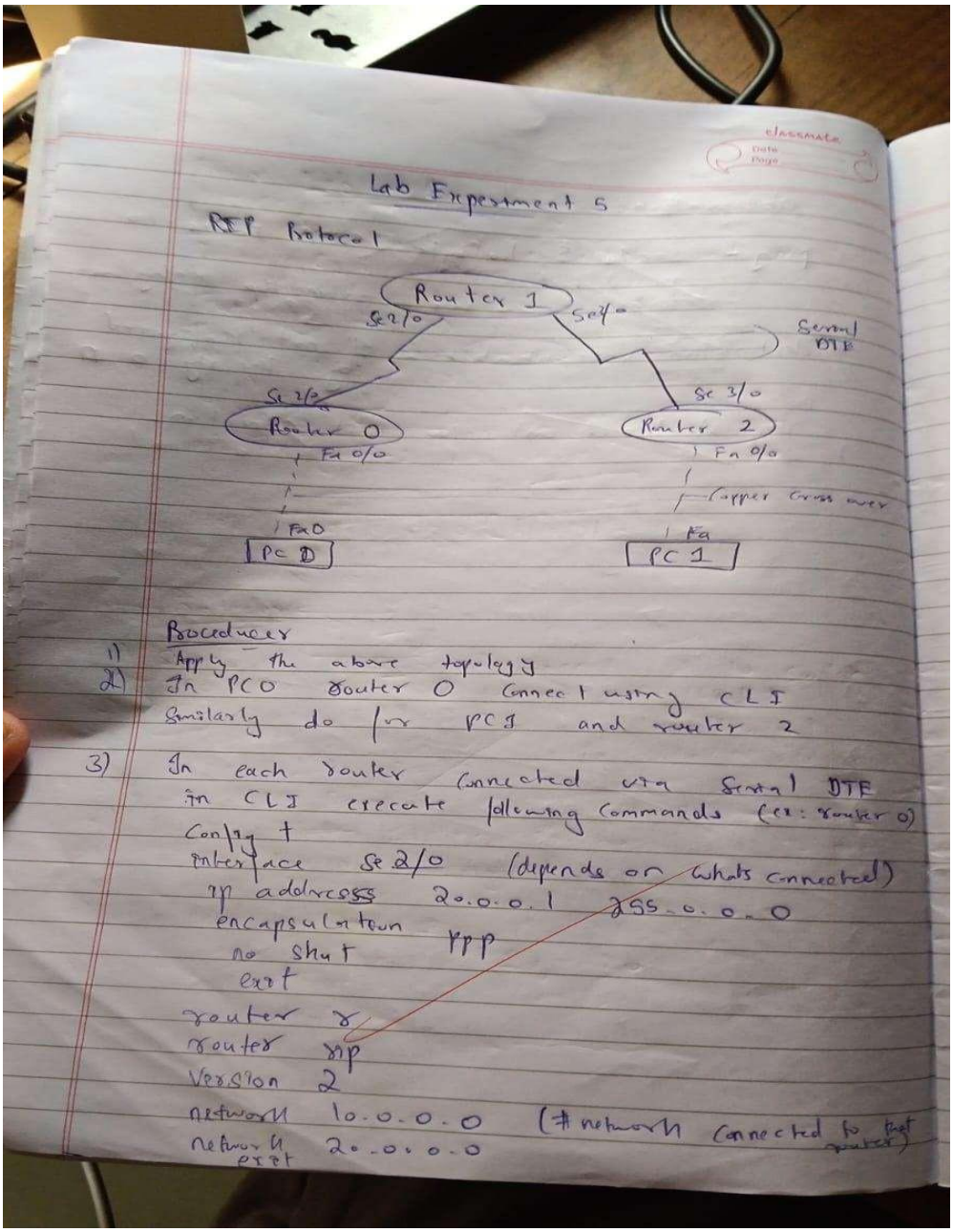

IP Configuration
X
IP Configuration
DHCP
Static
DHCP request successful.
IP Address
10.0.0.4
Subnet Mask
255.0.0.0
Default Gateway
10.0.0.10
DNS Server
0.0.0.0
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
/
Link Local Address
FE80::20A:41FF:FEA2:99D6
IPv6 Gateway
IPv6 DNS Server

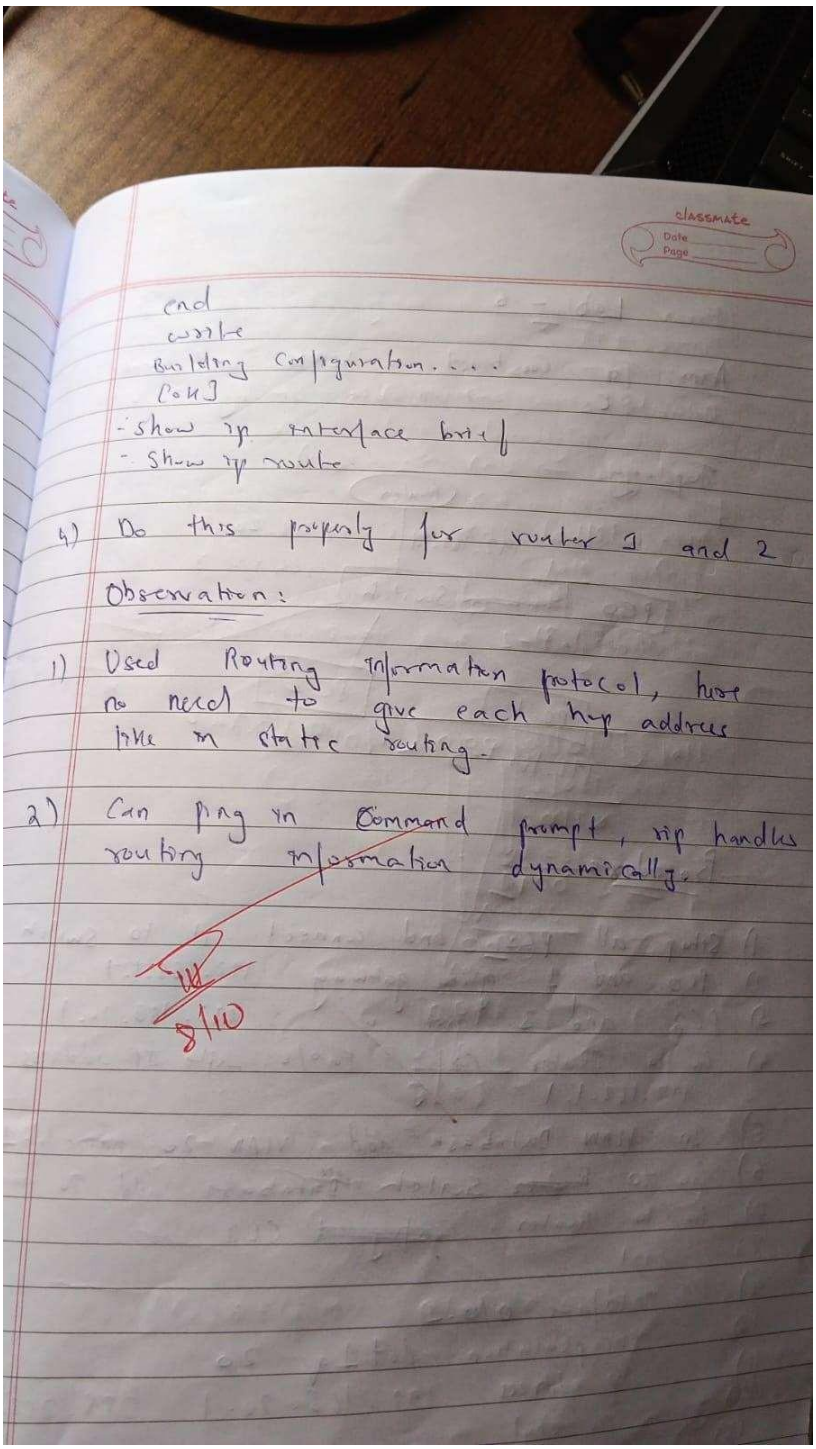

IP Configuration
X
IP Configuration
DHCP
Static
DHCP request successful.
IP Address
20.0.0.2
Subnet Mask
255.0.0.0
Default Gateway
20.0.0.10
DNS Server
0.0.0.0
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
/
Link Local Address
FE80::230:A3FF:FED9:B95A
IPv6 Gateway
IPv6 DNS Server

Program 5

Aim : Configure RIP routing Protocol in Routers

Observations :





end
write
Building Configuration....
[OK]
- show ip interface brief
- show ip route

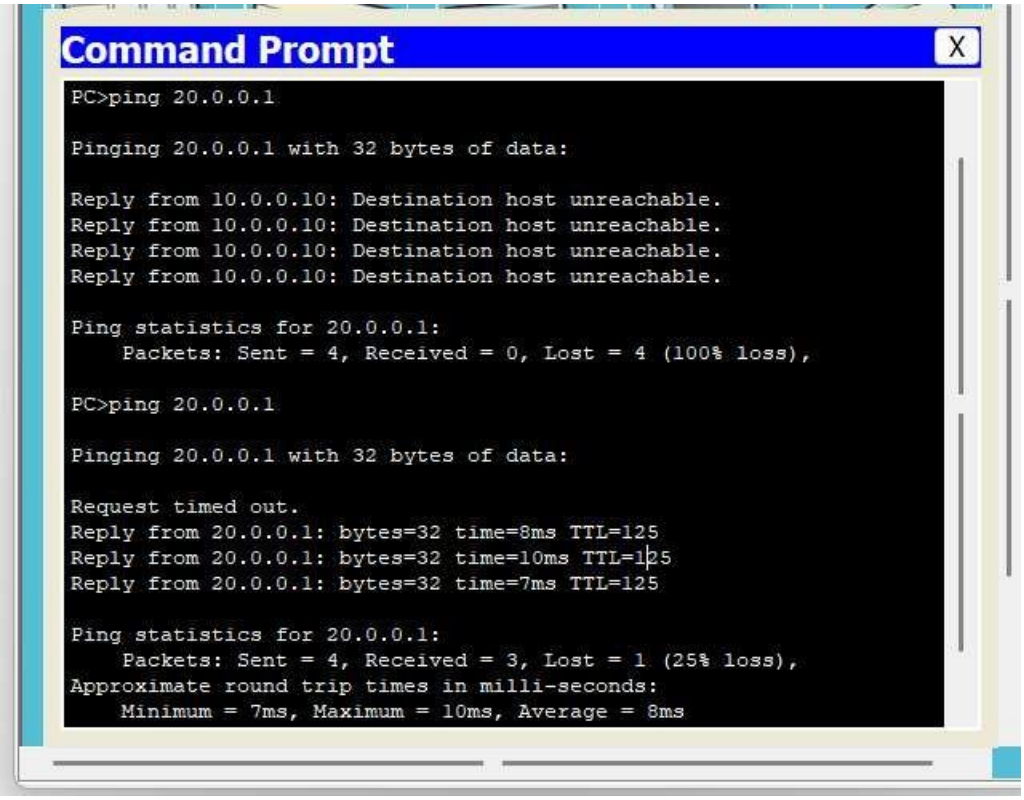
4) Do this properly for router 1 and 2

Observation:

1) Used Routing information protocol, here no need to give each hop address like in static routing.

2) Can ping in Command prompt, rip handles routing information dynamically.

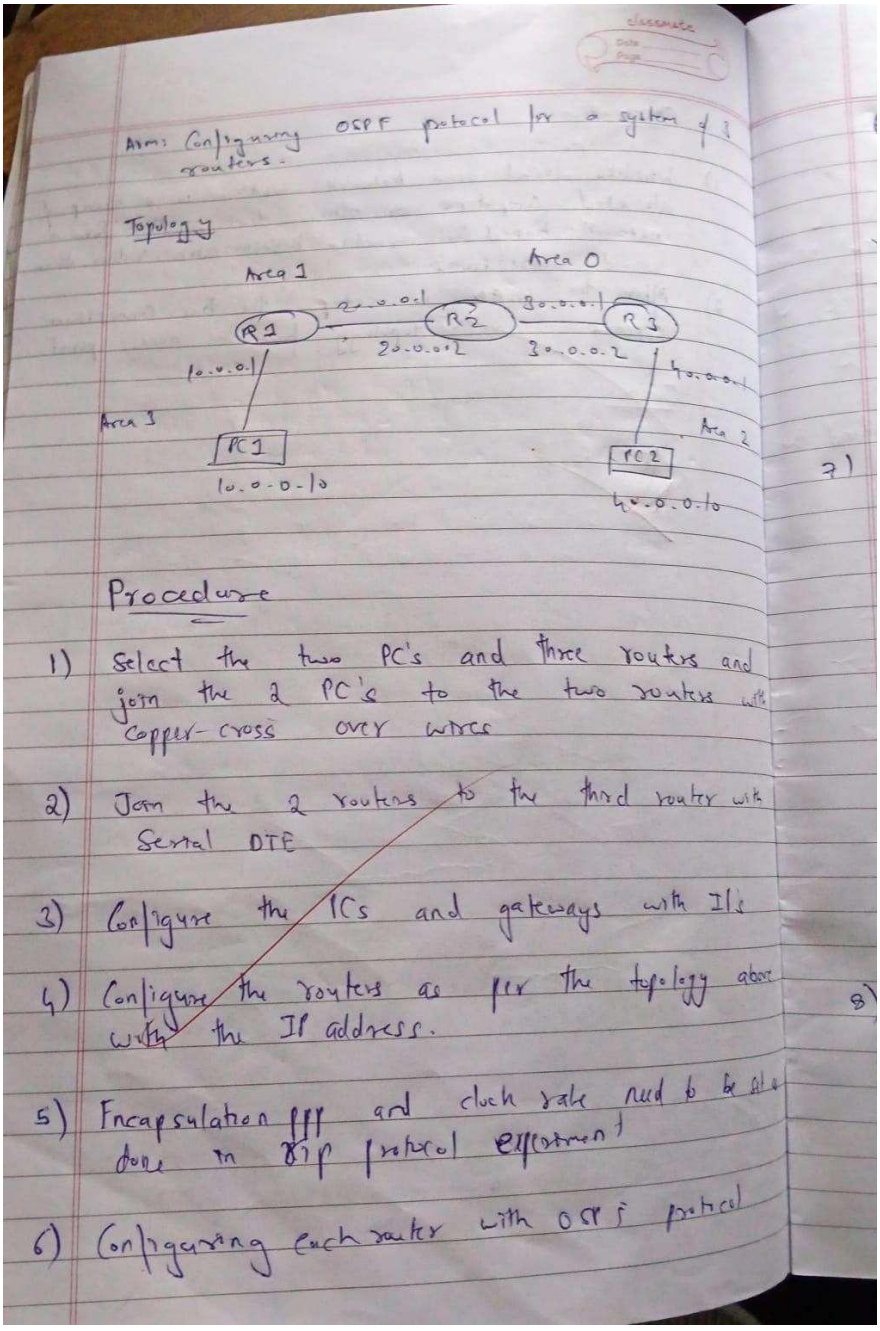
Output:


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

Observations:



Aim: Configuring OSPF protocol for a system of 3 routers.

Topology

Procedure

1) Select the two PC's and three routers and join the 2 PC's to the two routers with copper-cross over wires

2) Join the 2 routers to the third router with Serial DTE

3) Configure the PCs and gateways with IPs

4) Configure the routers as per the topology above with the IP address.

5) Encapsulation ppp and clock rate need to be done in rip protocol experiment

6) Configuring each router with OSPF protocol

For router 1

Enable
Config t
router ospf 1
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

Similarly do for other routers with id 2 & 3

7) Configure the interface

(config)
R0 # interface loopback 0
ip address 17).16.1.252 255.255.0.0
no shutdown

R1 interface loopback 0
253

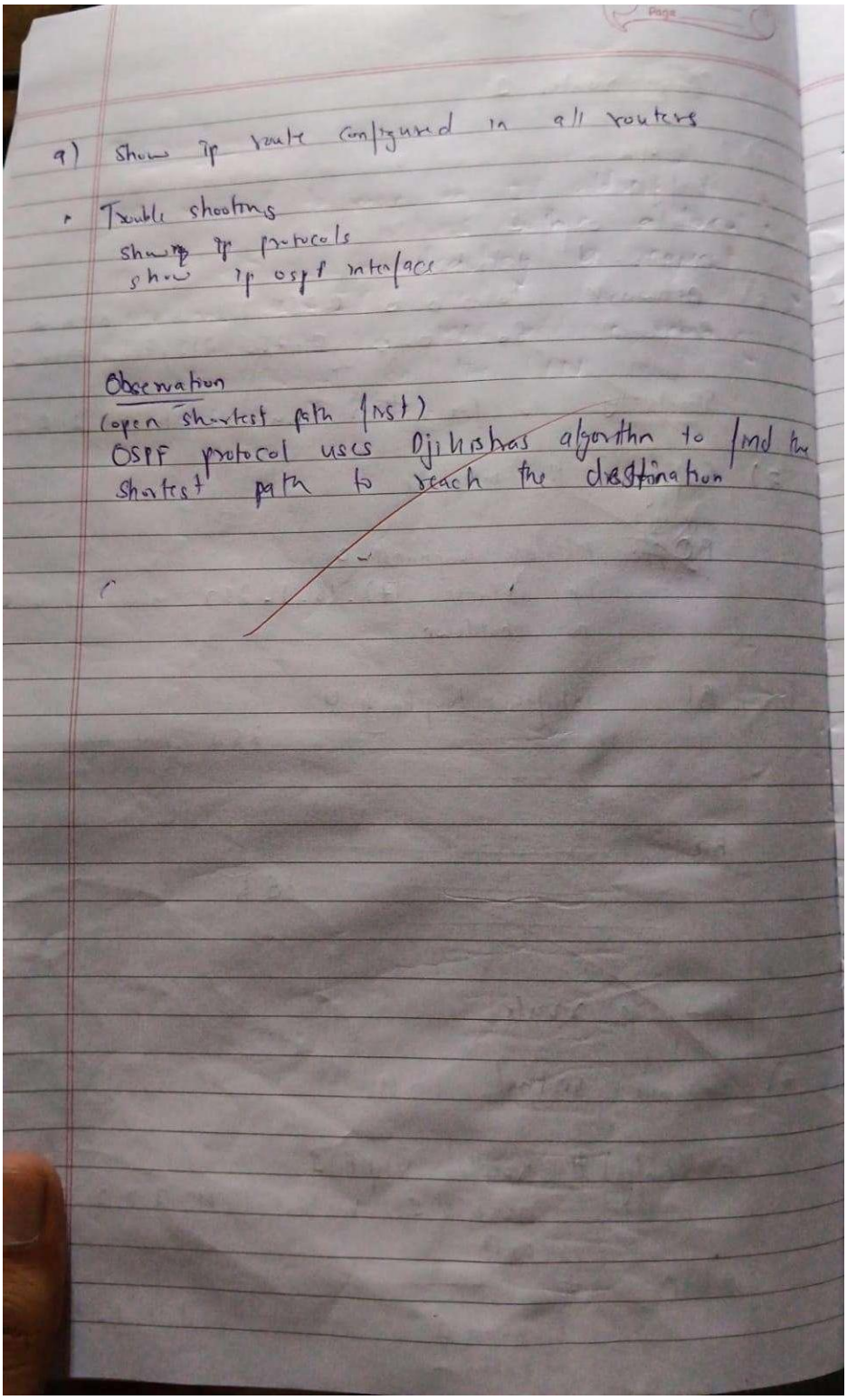
R2
254

show ip route

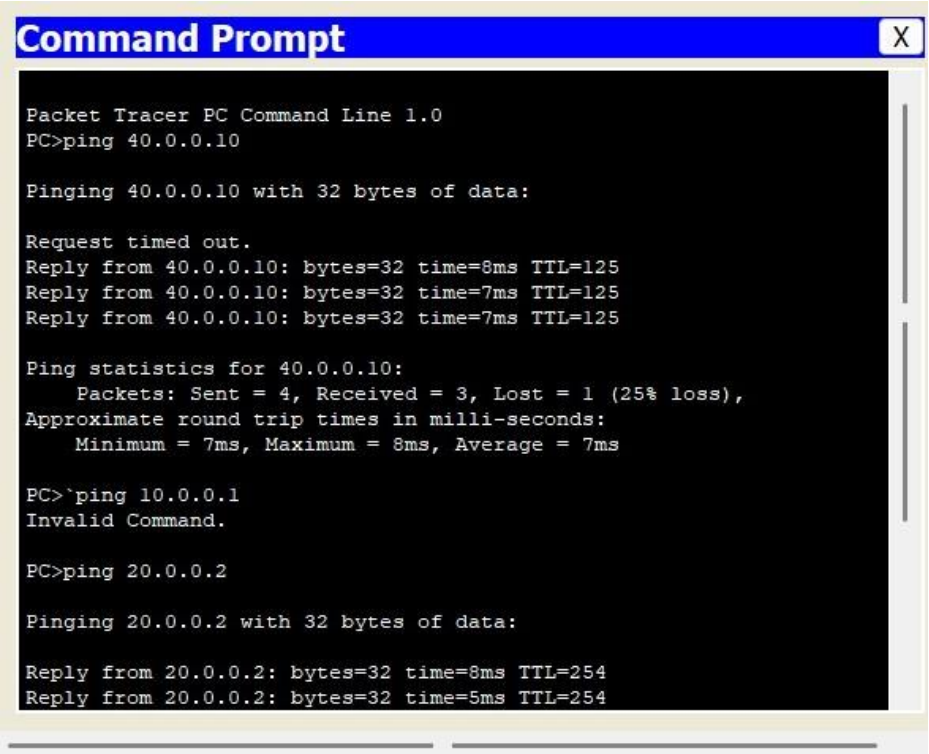
8) Configure virtual link

R0 (config) # router ospf 1
area 1 virtual-link 2.2.2.2
cntl

R1 (—) #
area 1 — 1.1.1.1
exit


9) Show ip route configured in all routers
• Trouble shooting
show ip protocols
show ip ospf interface
Observation
(open shortest path first)
OSPF protocol uses Dijkstras algorithm to find the
shortest path to reach the destination

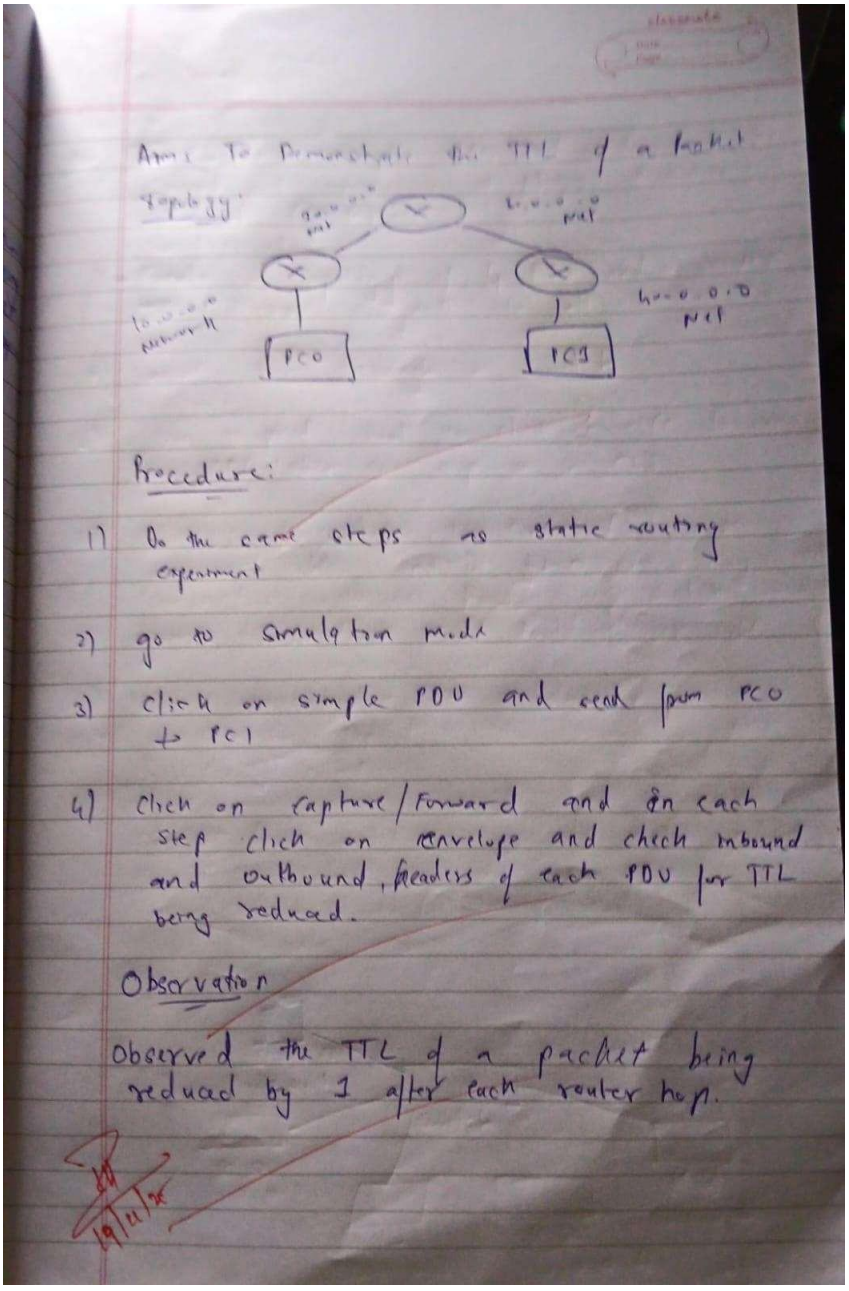
Output:


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

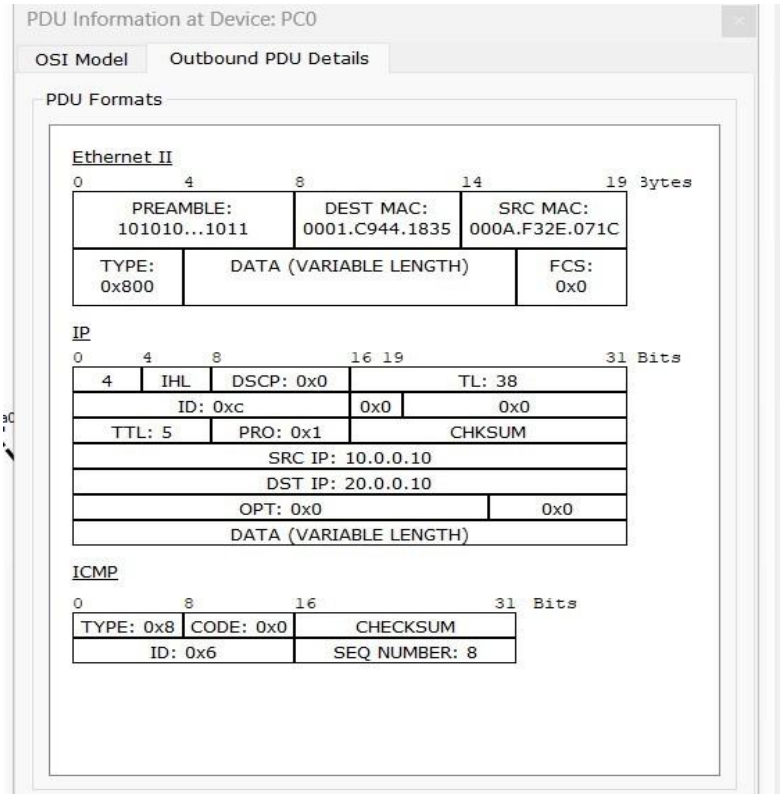
Program 7

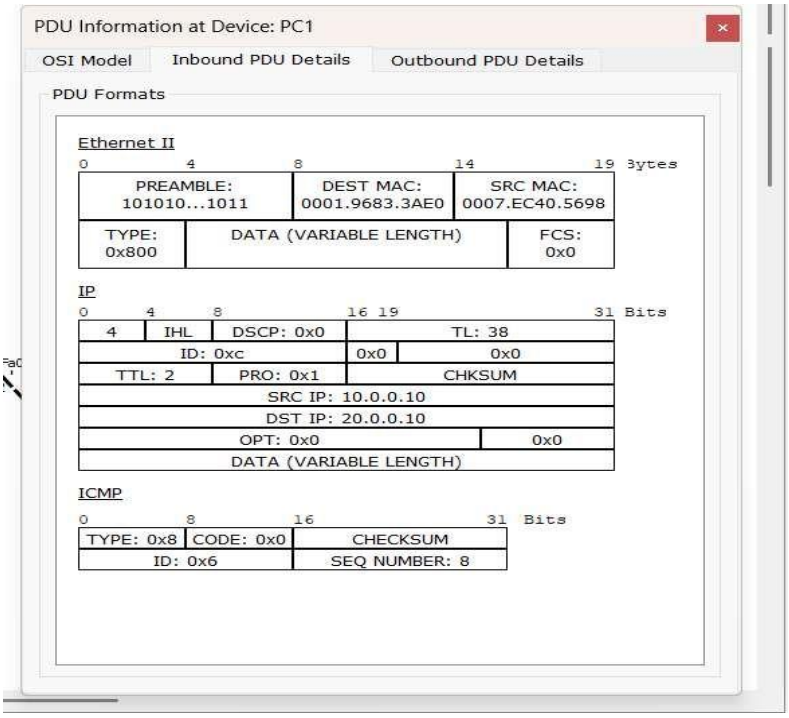
Aim : Demonstrate the TTL/ Life of a Packet

Observations:



Output:

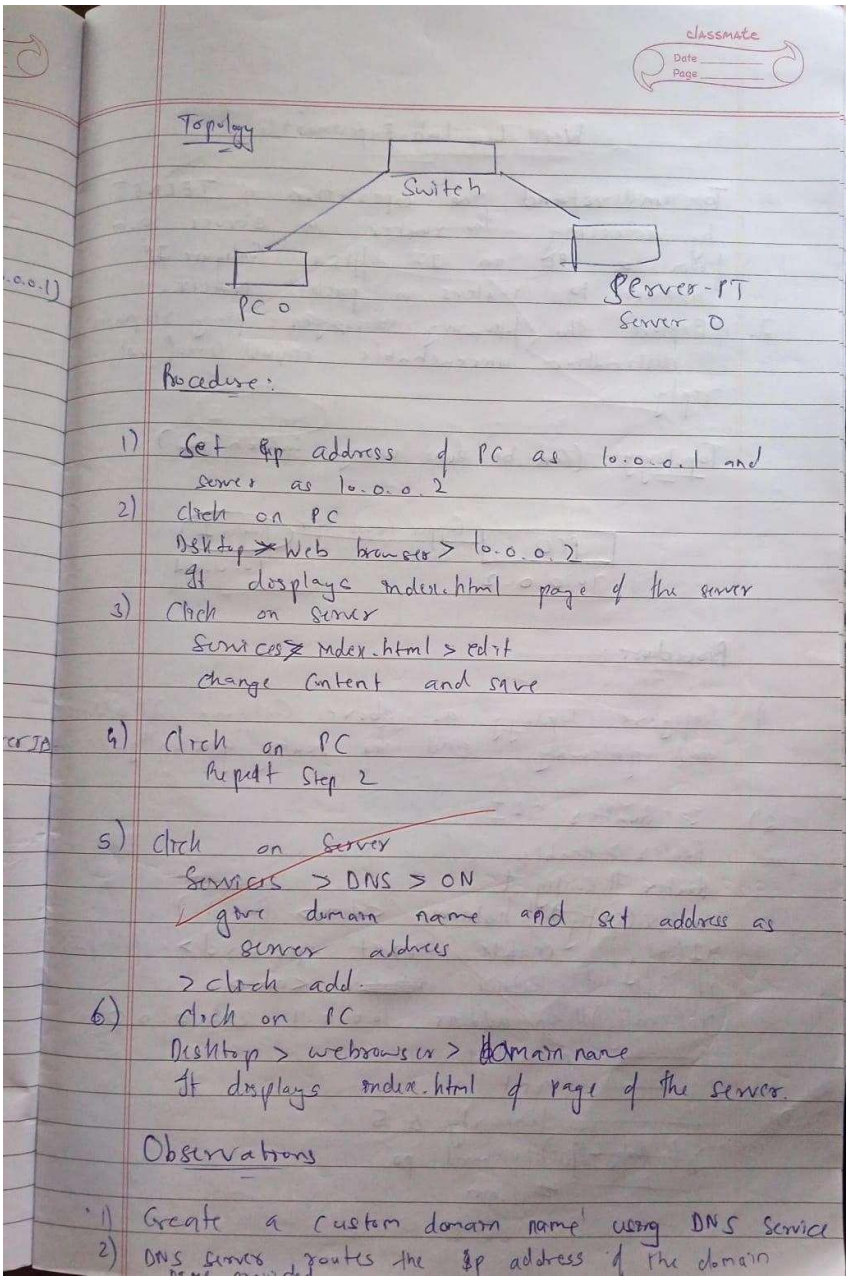

PDU Information at Device: PC0
OSI Model
Outbound PDU Details
PDU Formats
Ethernet II
0 4 8 14 19 Bytes
PREAMBLE: 101010...1011
DEST MAC: 0001.C944.1835
SRC MAC: 000A.F32E.071C
TYPE: 0x800
DATA (VARIABLE LENGTH)
FCS: 0x0
IP
0 4 8 16 19 31 Bits
4
IHL
DSCP: 0x0
TL: 38
ID: 0xc
0x0
0x0
TTL: 5
PRO: 0x1
CHKSUM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

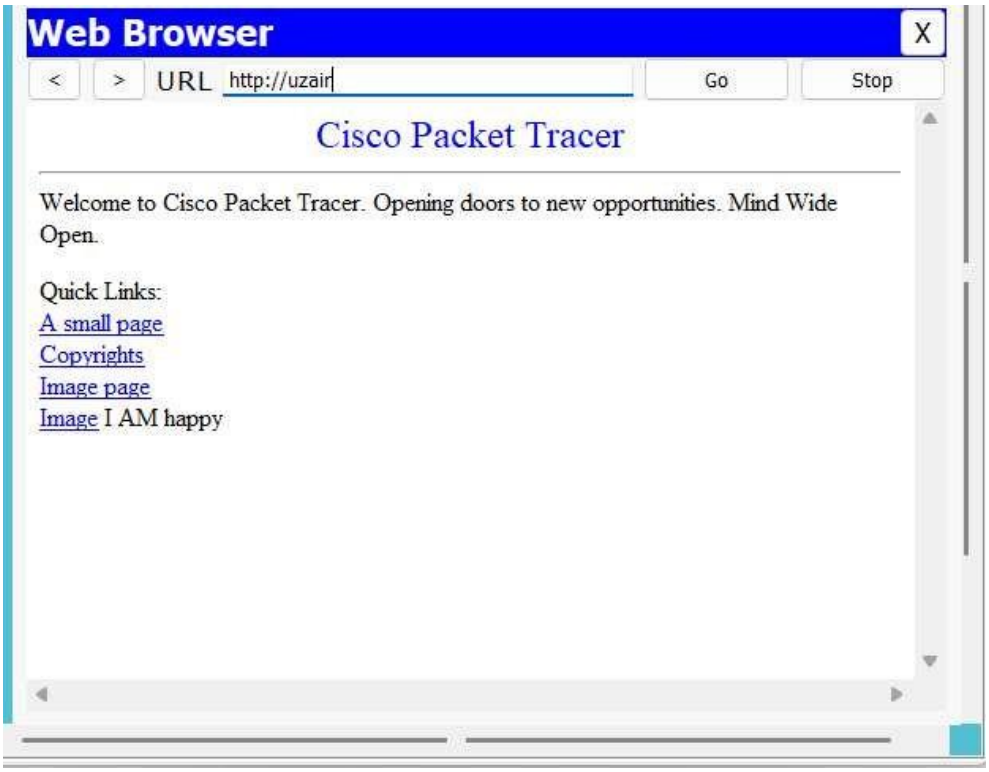
Program 8

Aim : Configure Web Server, DNS within a LAN.

Observation:



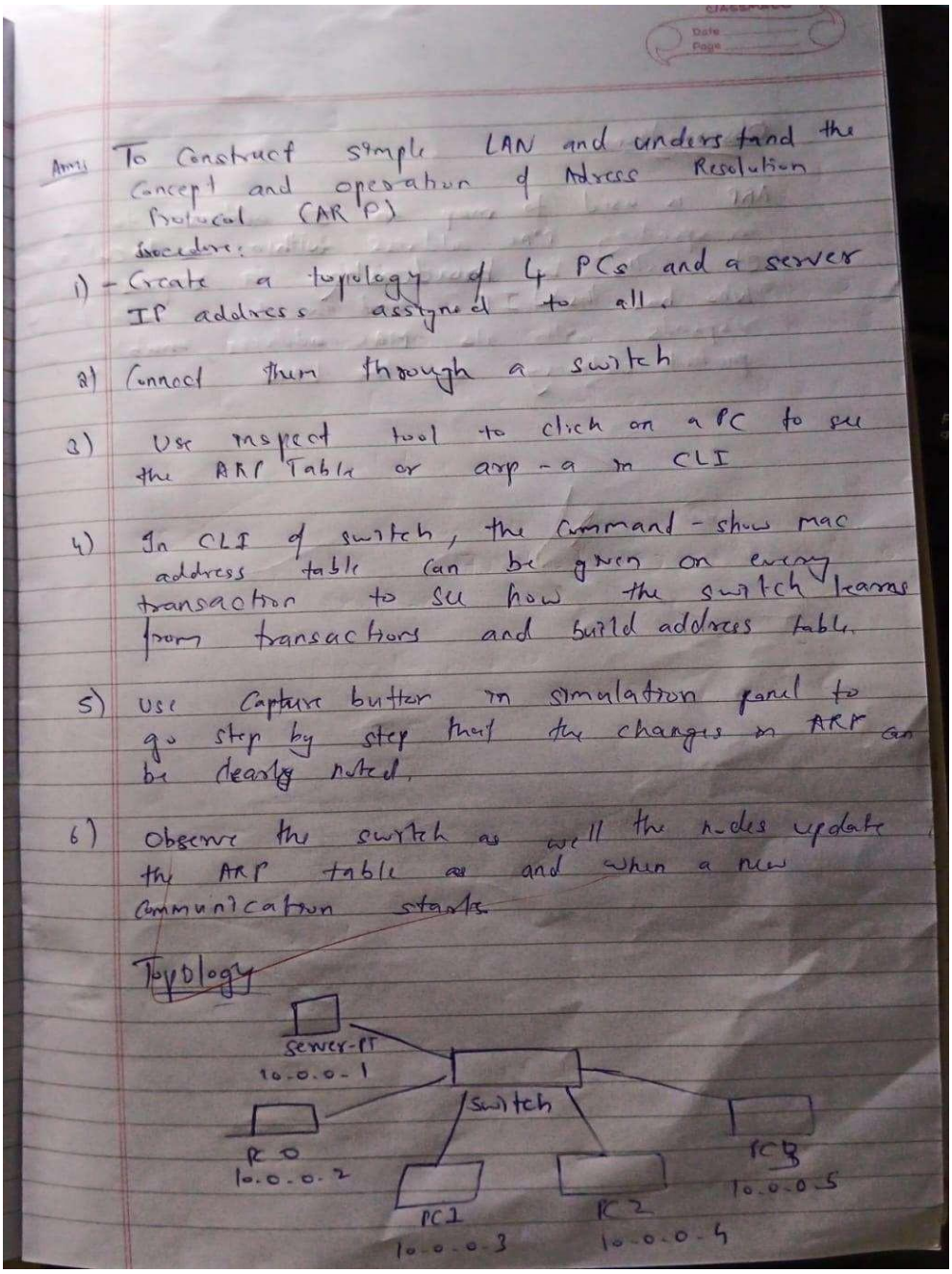
Output:

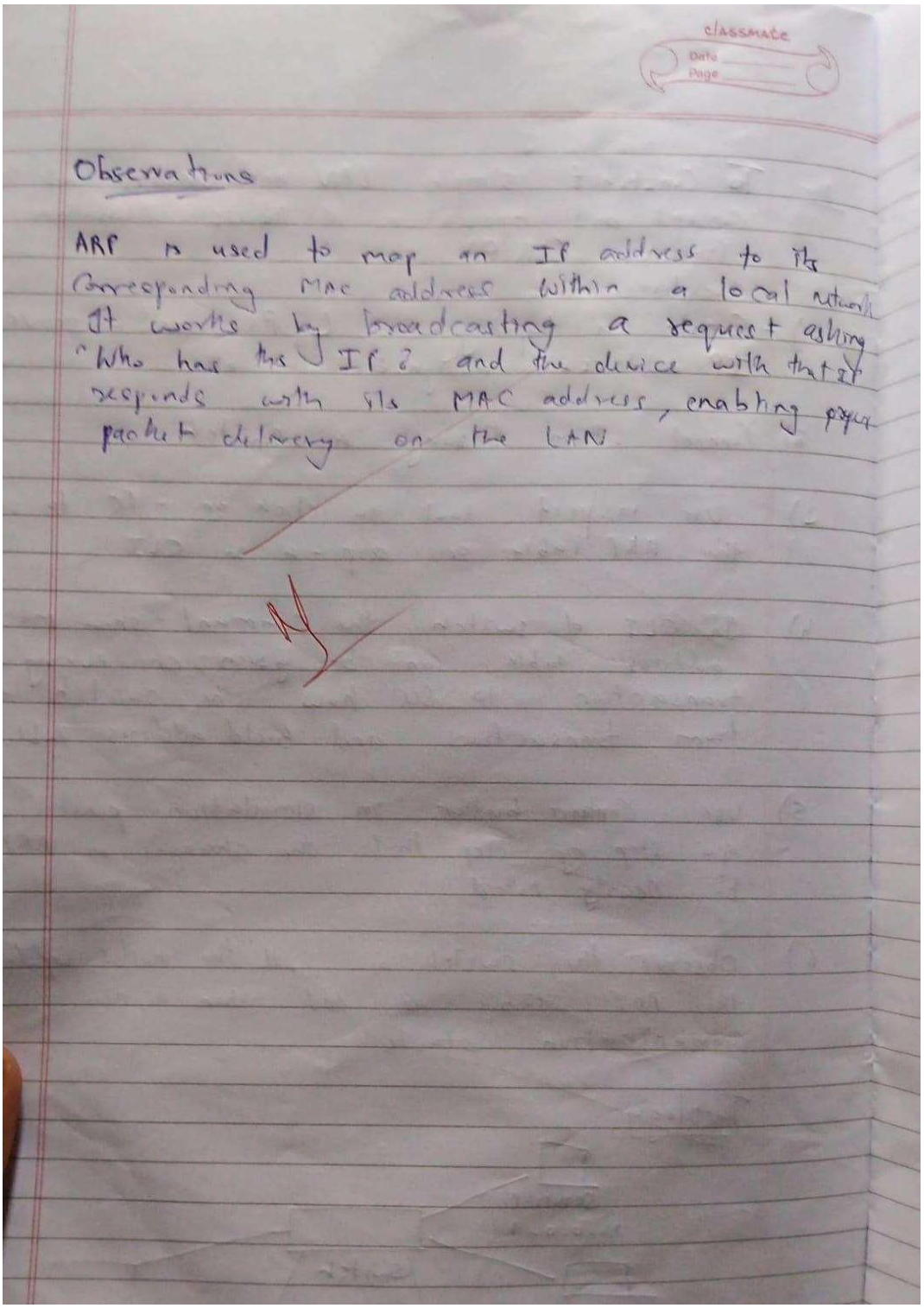

Web Browser
X
<
>
URL
http://uzair
Go
Stop
Cisco Packet Tracer
Welcome to Cisco Packet Tracer. Opening doors to new opportunities. Mind Wide Open.
Quick Links:
A small page
Copyrights
Image page
Image I AM happy

Program 9

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

Observation:

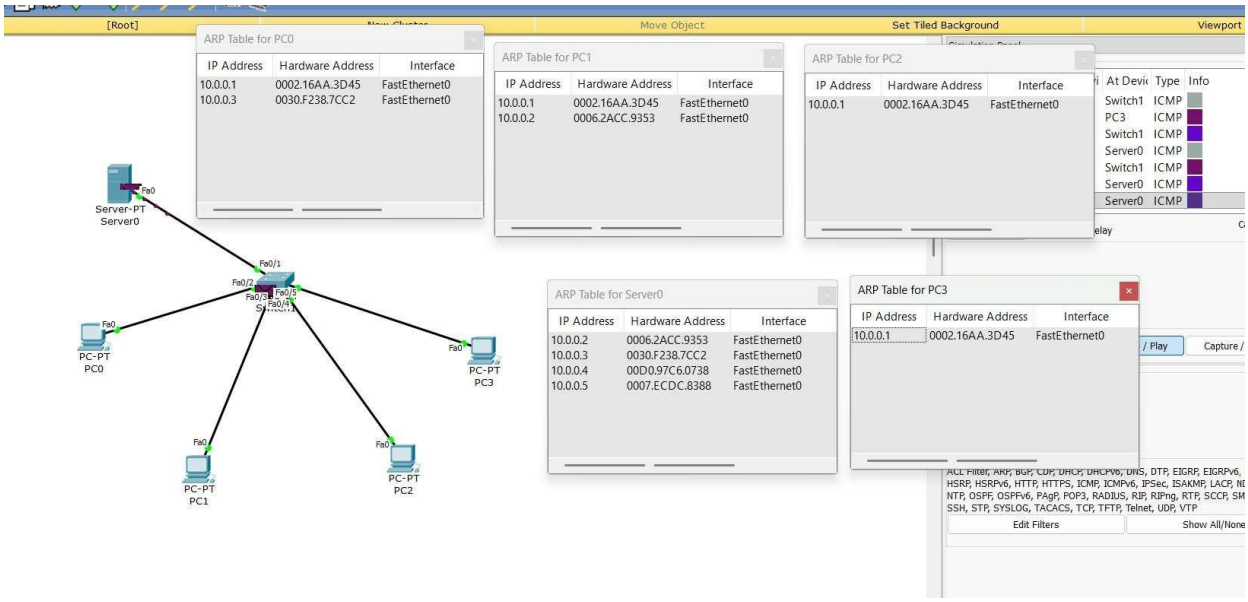




Observations

ARP is used to map an IP address to its corresponding MAC address within a local network. It works by broadcasting a request asking "Who has this IP?" and the device with that IP responds with its MAC address, enabling proper packet delivery on the LAN.

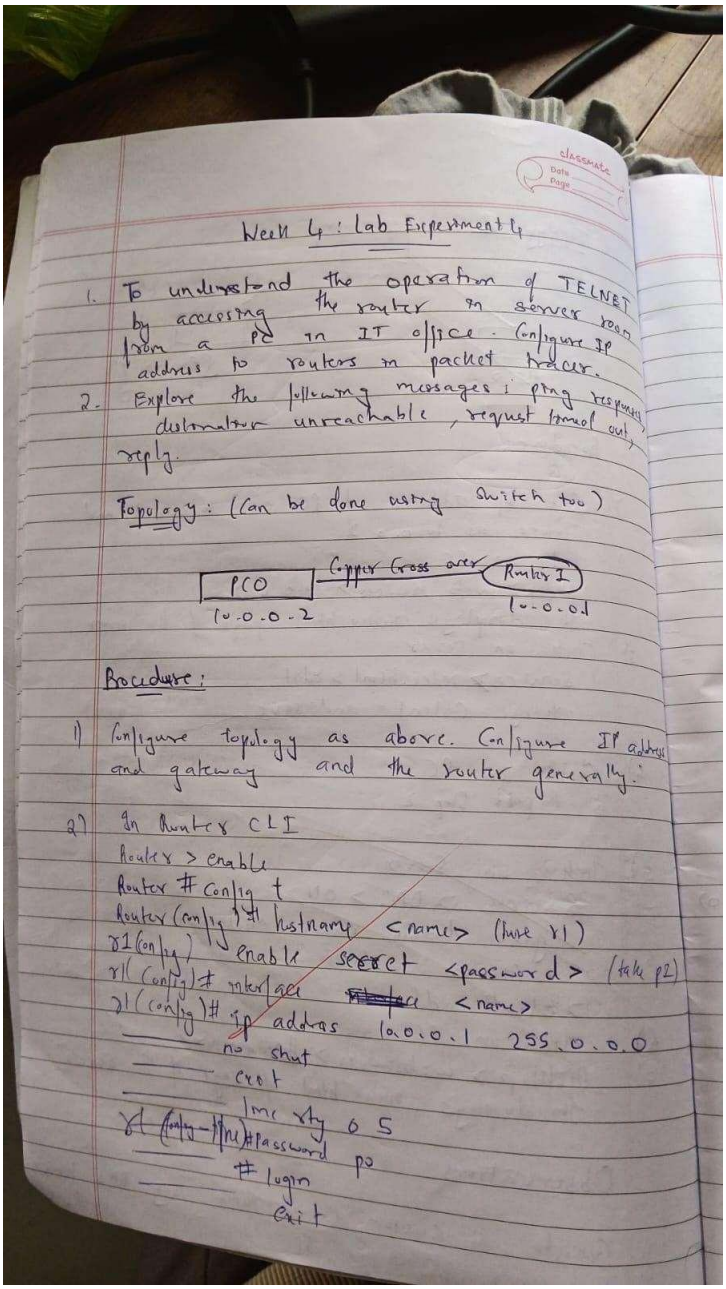
Output:


[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
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
At Devic Type Info
Switch1 ICMP
PC3 ICMP
Switch1 ICMP
Server0 ICMP
Switch1 ICMP
Server0 ICMP
Server0 ICMP
Server-PT Server0
PC-PT PC0
PC-PT PC1
PC-PT PC2
PC-PT PC3
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:



Week 4 : Lab Experiment 4

1. To understand the operation of TELNET by accessing the router in server room from a PC in IT office. Configure IP address to routers in packet tracer.
2. Explore the following messages : ping response, destination unreachable, request timed out, reply.

Topology : (can be done using Switch too)

PC0 — Copper Cross over — Router 1

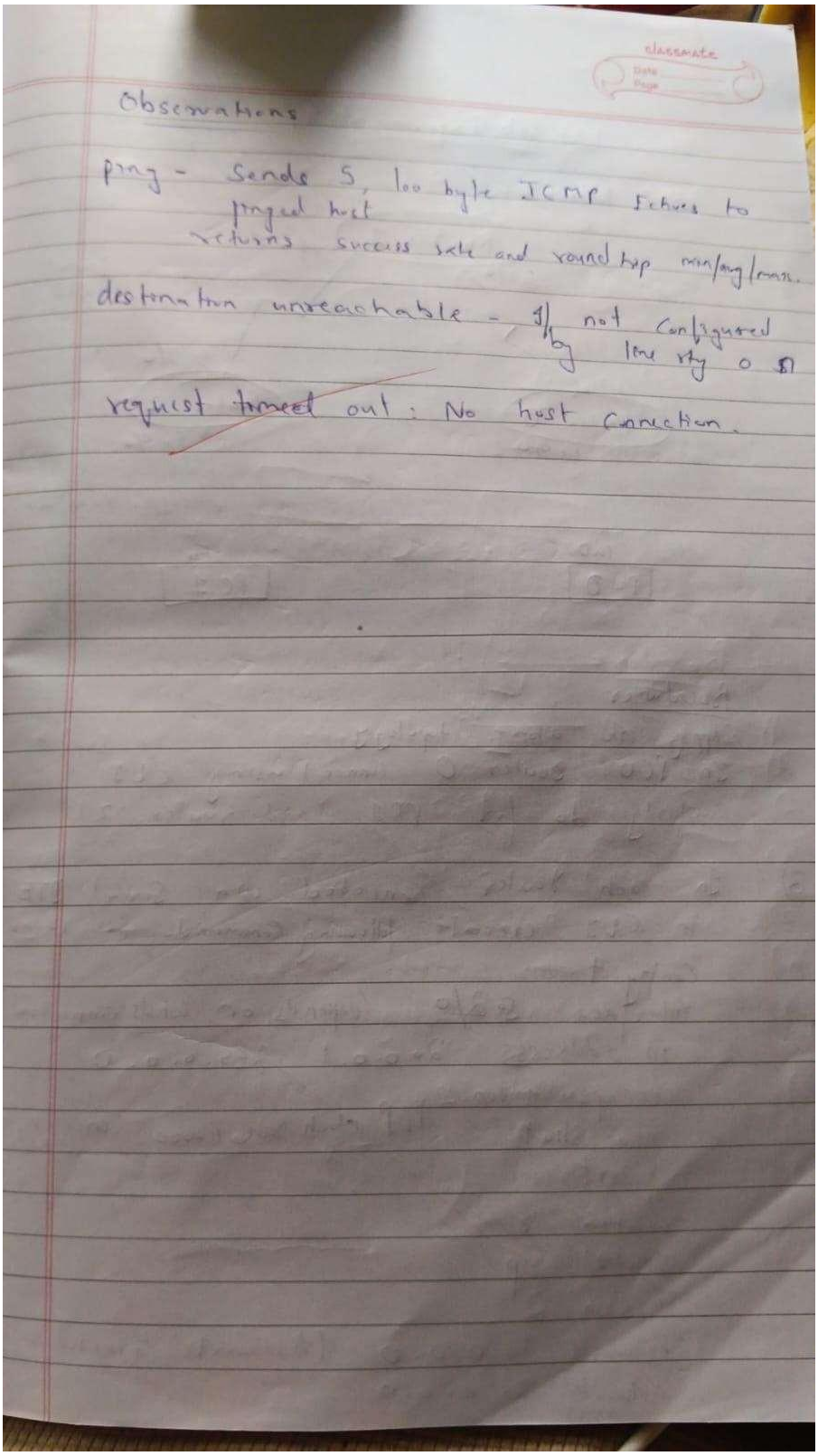
10.0.0.2 — 10.0.0.1

Procedure :

1) Configure topology as above. Configure IP address and gateway and the router generally.

2) In Router CLI

Router > enable
Router # config t
Router (config) # hostname <name> (here r1)
r1 (config) enable secret <password> (take p2)
r1 (config) # interface <name>
r1 (config) # ip address 10.0.0.1 255.0.0.0
no shut
exit
line vty 0 5
r1 (config-line) # password p2
login
exit



Observations

ping - Sends 5, 100 byte ICMP Echos to pinged host
returns success rate and round trip min/avg/max.

destination unreachable - if not configured by long rtg o n

request timed out : No host connection.

Output:

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

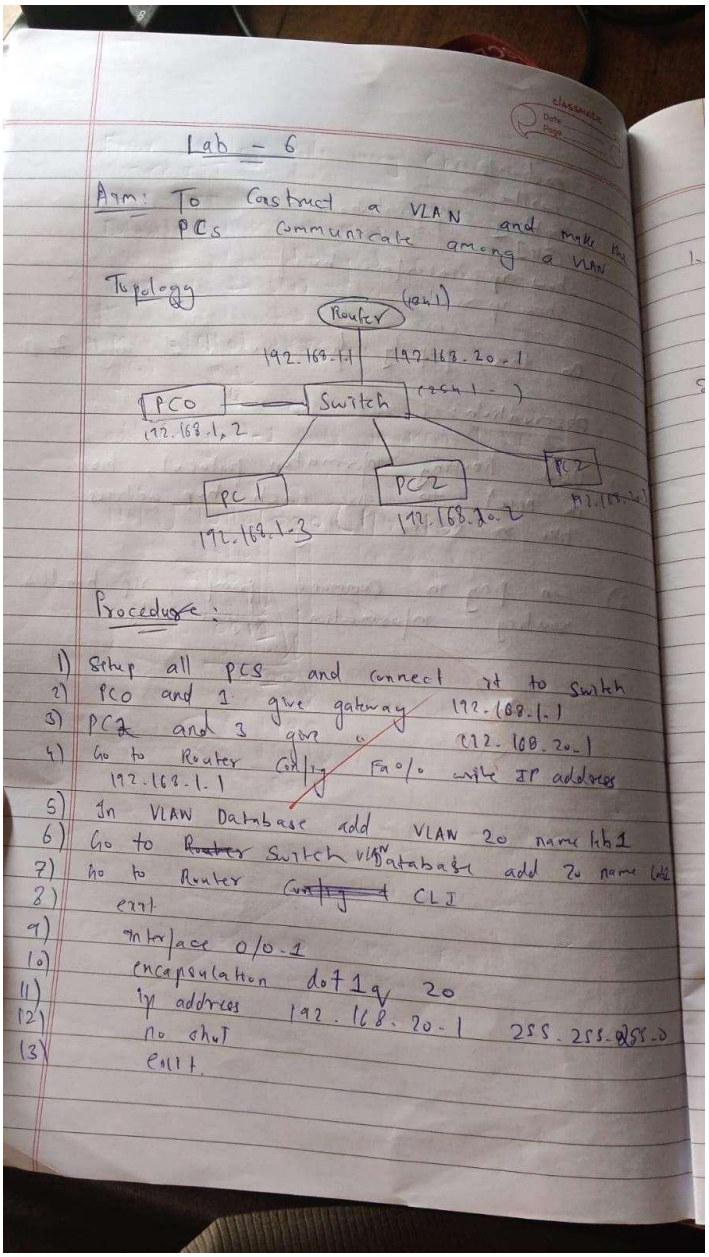
```
PC>telnet 10.0.0.1
Trying 10.0.0.1 ...Open

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

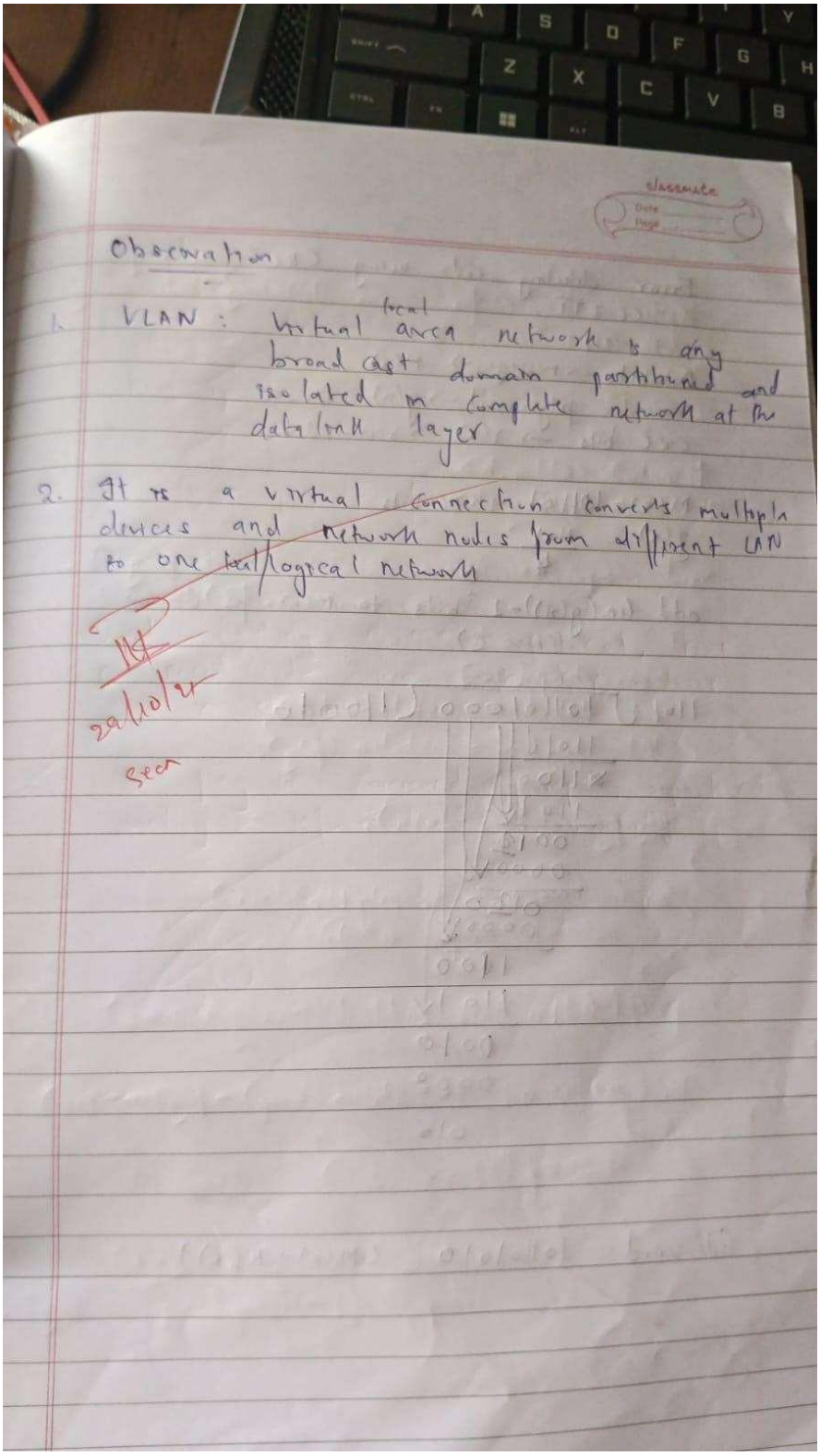
```
[Connection to 10.0.0.1 closed by foreign host]
PC>telnet 10.0.0.1
Trying 10.0.0.1 ...Open


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:





Observation

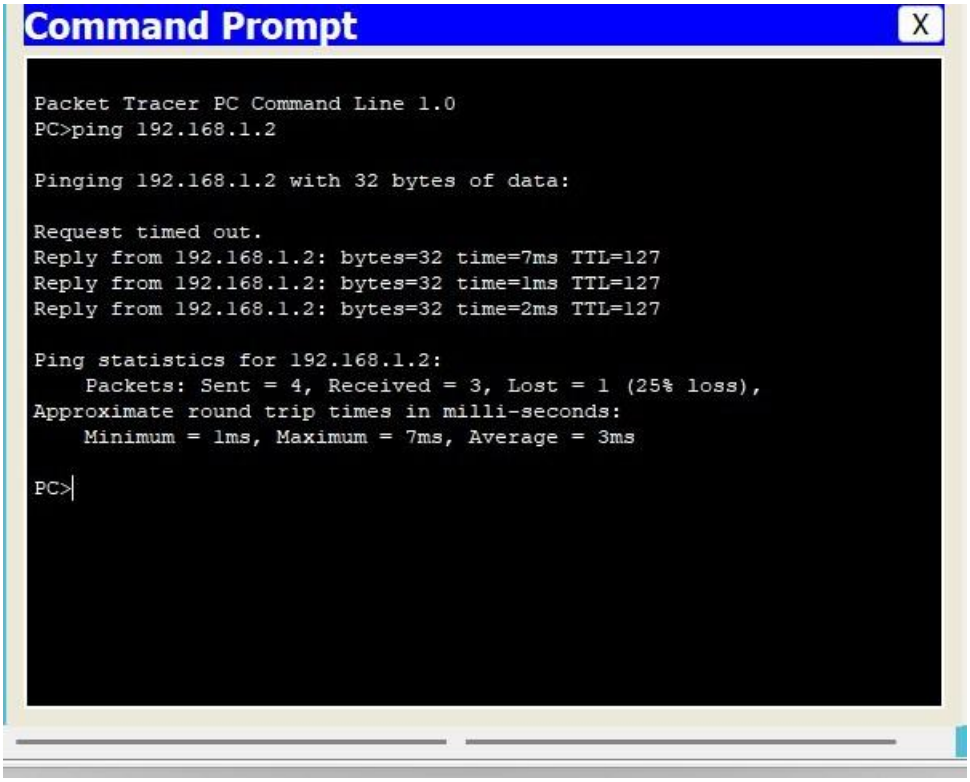
1. VLAN : Virtual local area network is any broad cast domain partitioned and isolated in Complete network at the data link layer

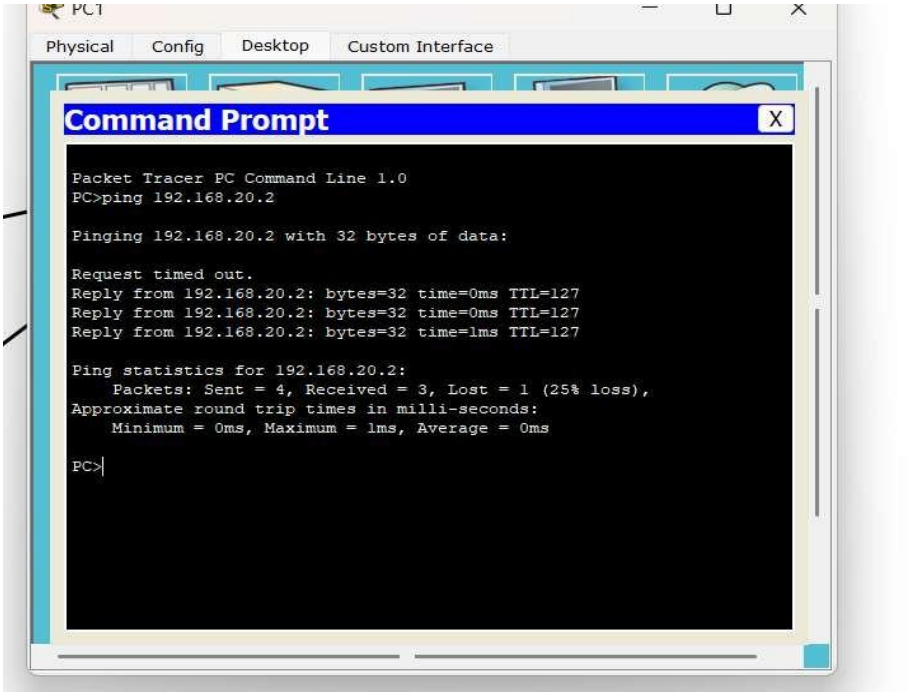
2. It is a virtual connection converts multiple devices and network nodes from different LAN to one logical network

29/10/24

Seen

Output:

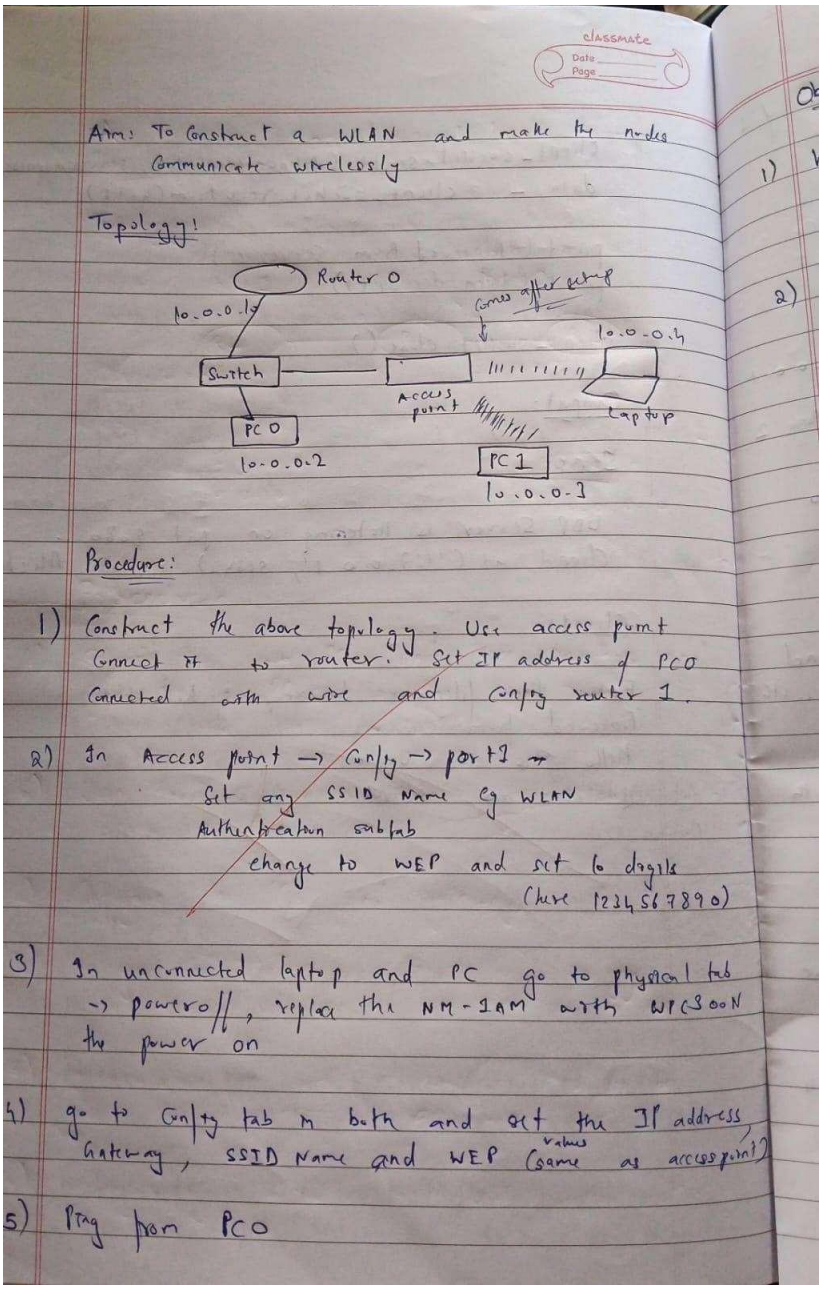

Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 192.168.1.2
Pinging 192.168.1.2 with 32 bytes of data:
Request timed out.
Reply from 192.168.1.2: bytes=32 time=7ms TTL=127
Reply from 192.168.1.2: bytes=32 time=1ms TTL=127
Reply from 192.168.1.2: bytes=32 time=2ms TTL=127
Ping statistics for 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>

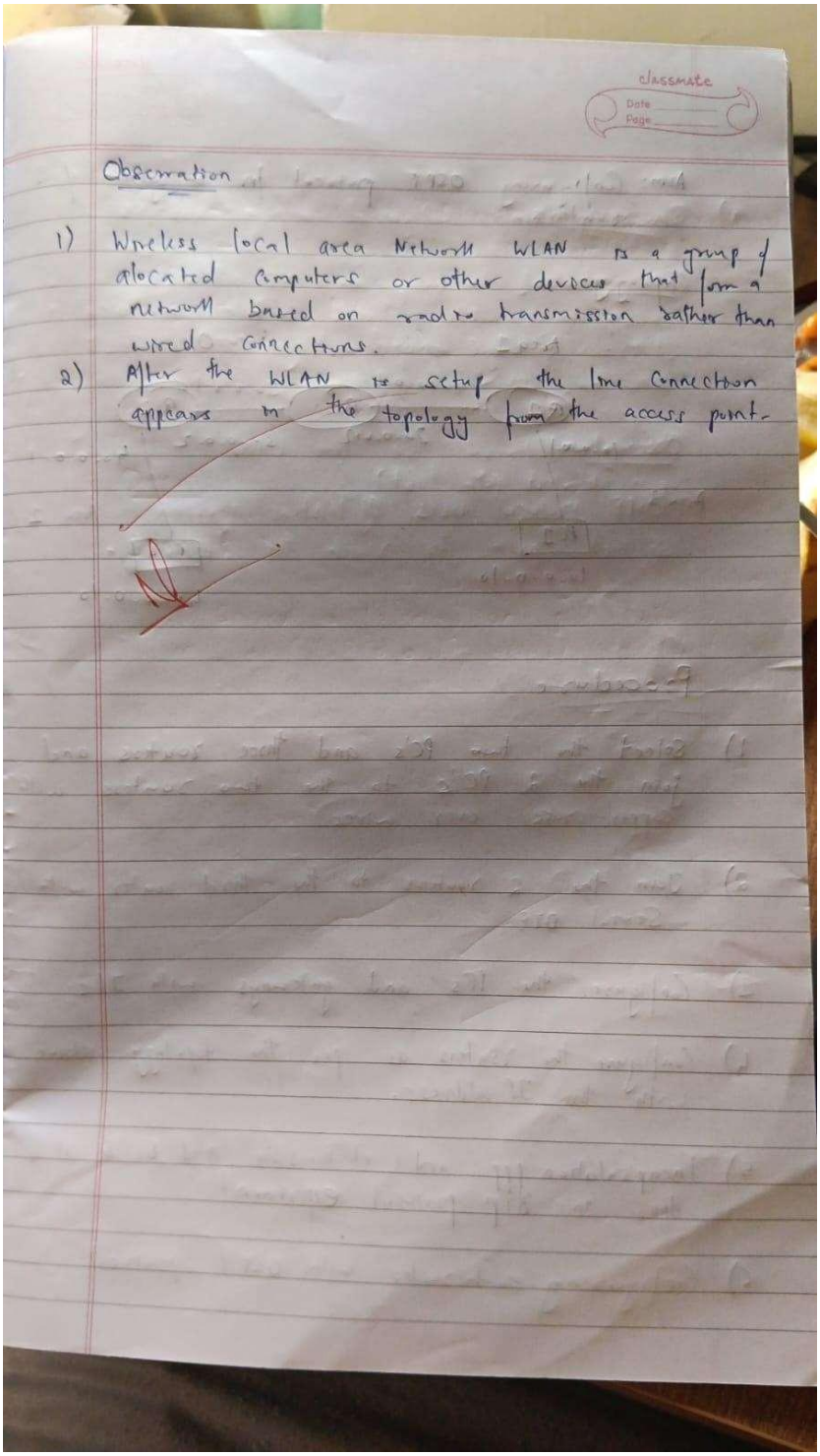

PC1
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 192.168.20.2
Pinging 192.168.20.2 with 32 bytes of data:
Request timed out.
Reply from 192.168.20.2: bytes=32 time=0ms TTL=127
Reply from 192.168.20.2: bytes=32 time=0ms TTL=127
Reply from 192.168.20.2: 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:



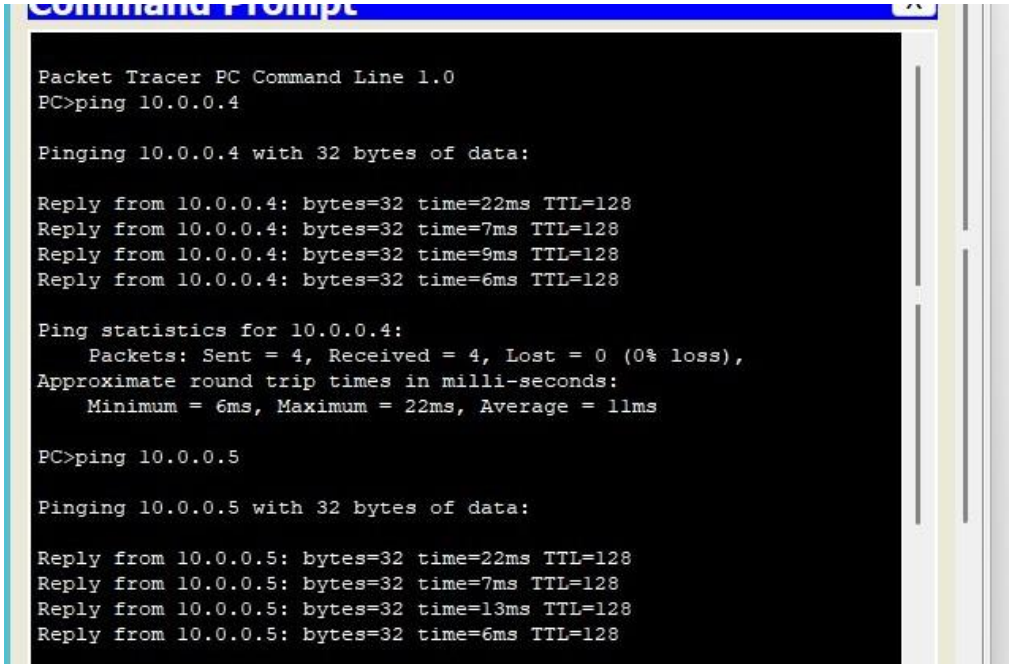


Observation

1) Wireless local area Network WLAN is a group of allocated computers or other devices that form a network based on radio transmission rather than wired connections.

2) After the WLAN is setup the line connection appears in the topology from the access point.

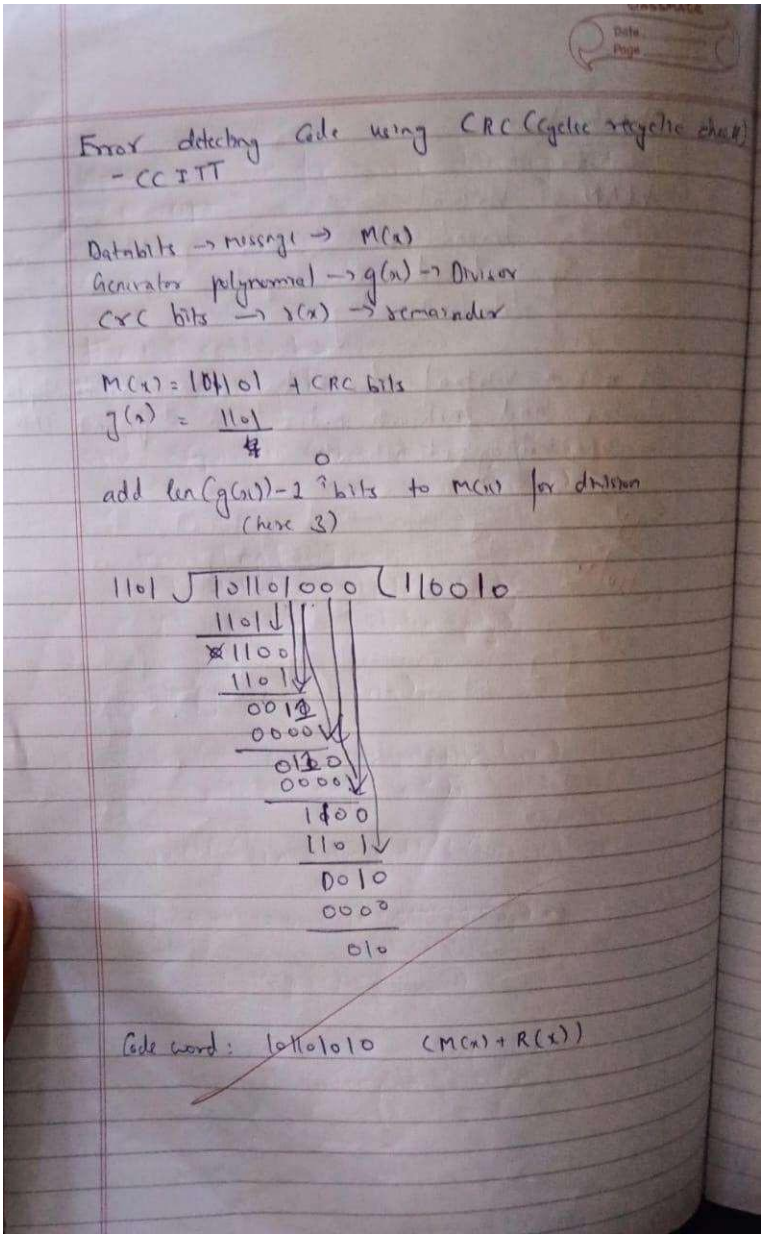
Output:

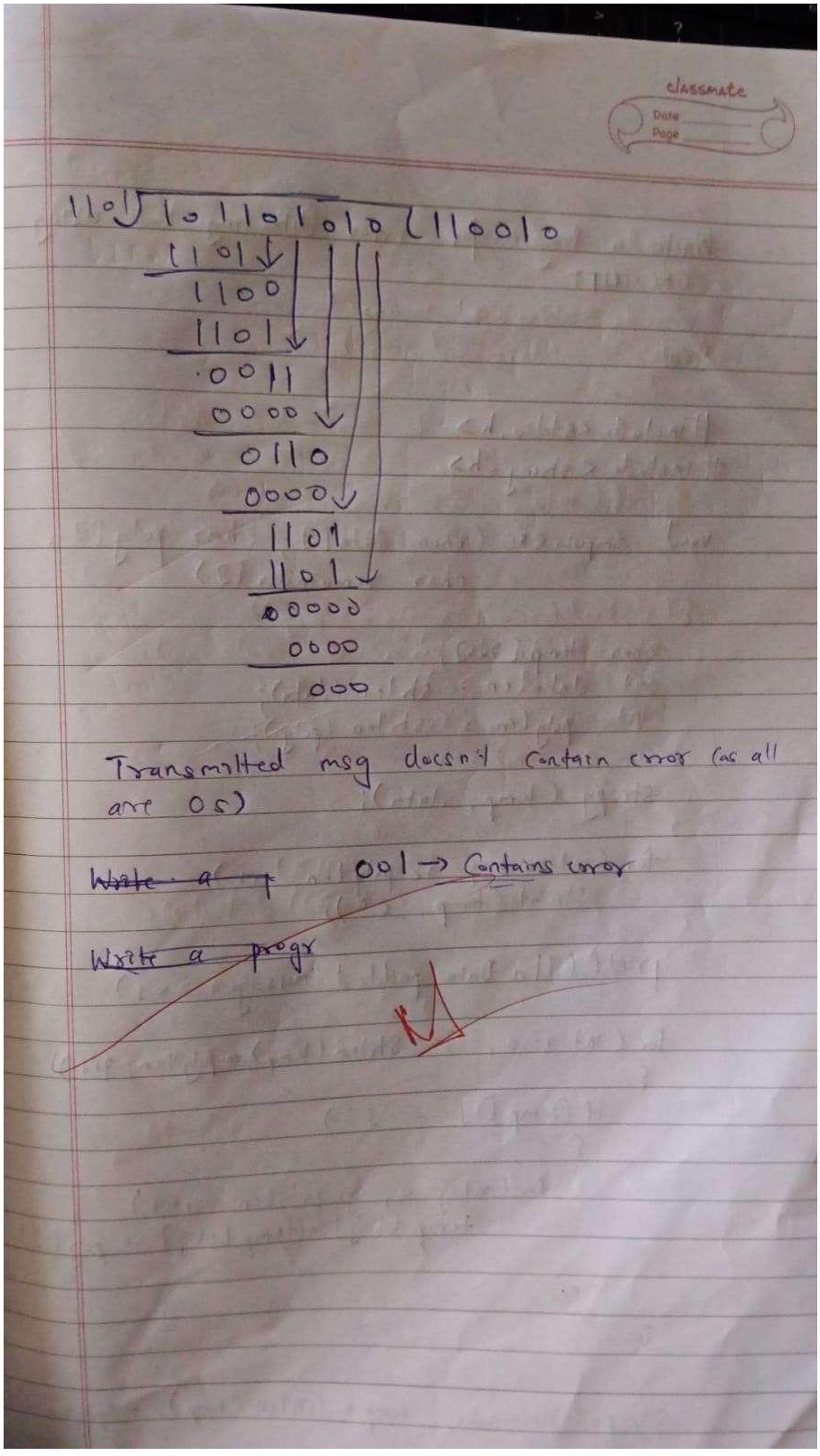

Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.4
Pinging 10.0.0.4 with 32 bytes of data:
Reply from 10.0.0.4: bytes=32 time=22ms TTL=128
Reply from 10.0.0.4: bytes=32 time=7ms TTL=128
Reply from 10.0.0.4: bytes=32 time=9ms TTL=128
Reply from 10.0.0.4: bytes=32 time=6ms TTL=128
Ping statistics 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:




classmate
Date
Page
1101) 10111010 (11001 0
1101
1100
1101
0011
0000
0110
0000
1101
1101
0000
0000
000
Transmitted msg doesn't contain error (as all are 0s)
001 → Contains error
Write a progr

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';
      }
   }

   // CRC = last (polyLen-1) bits
   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';
        }
    }

    // Check any '1' in remainder portion
    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;
}
```

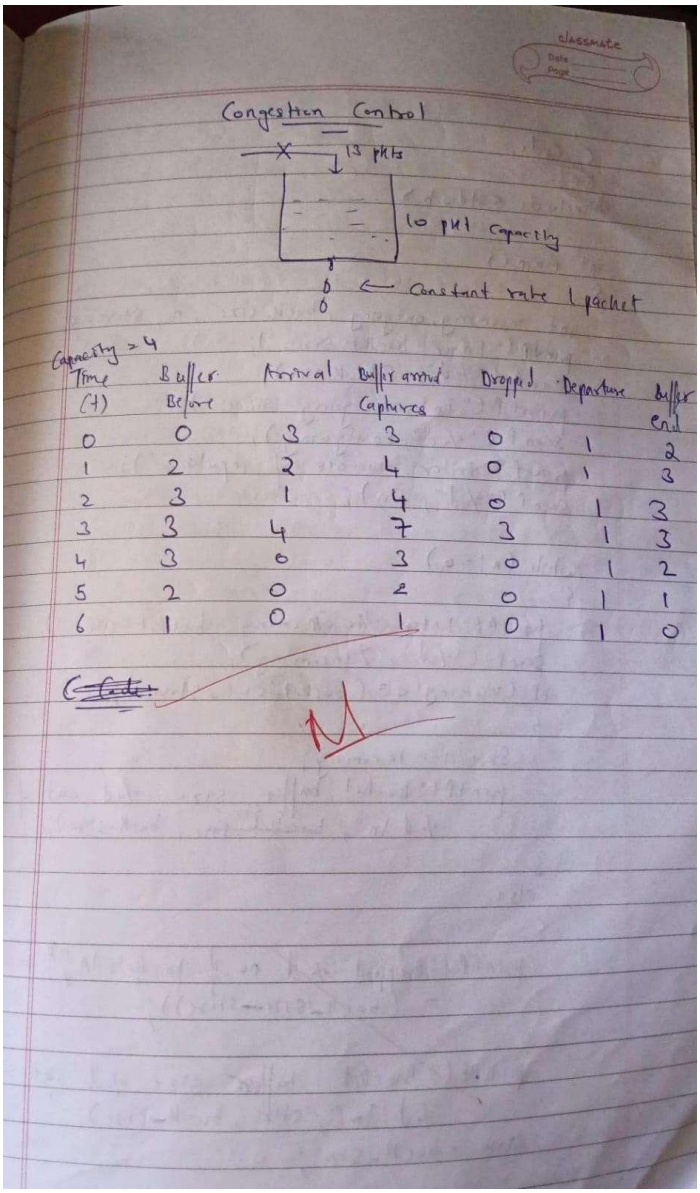
Output:

```
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:



Congestion Control

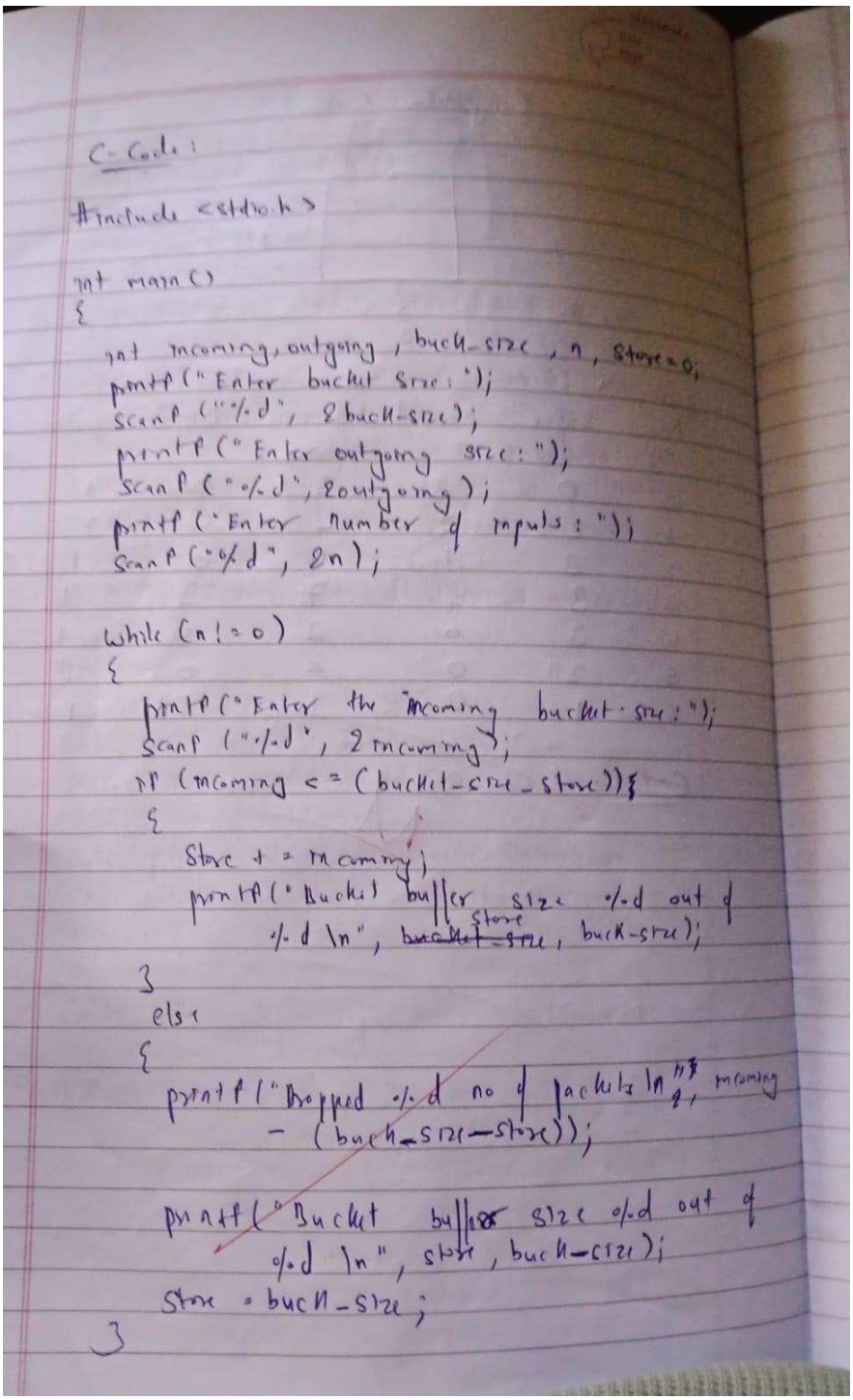
13 pkts

10 pkt capacity

0 ← Constant rate 1 packet

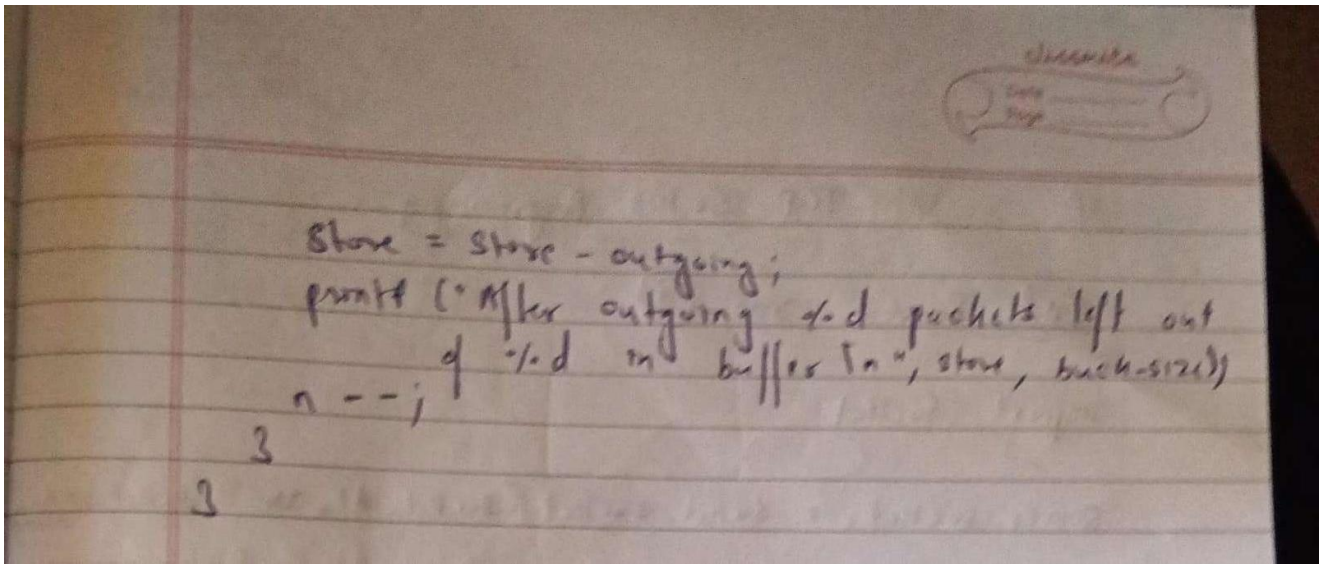
Capacity = 4

Time (t)	Buffer Before	Arrival	Buffer arrival Captures	Dropped	Departure	Buffer end
0	0	3	3	0	1	2
1	2	2	4	0	1	3
2	3	1	4	0	1	3
3	3	4	7	3	1	3
4	3	0	3	0	1	2
5	2	0	2	0	1	1
6	1	0	1	0	1	0



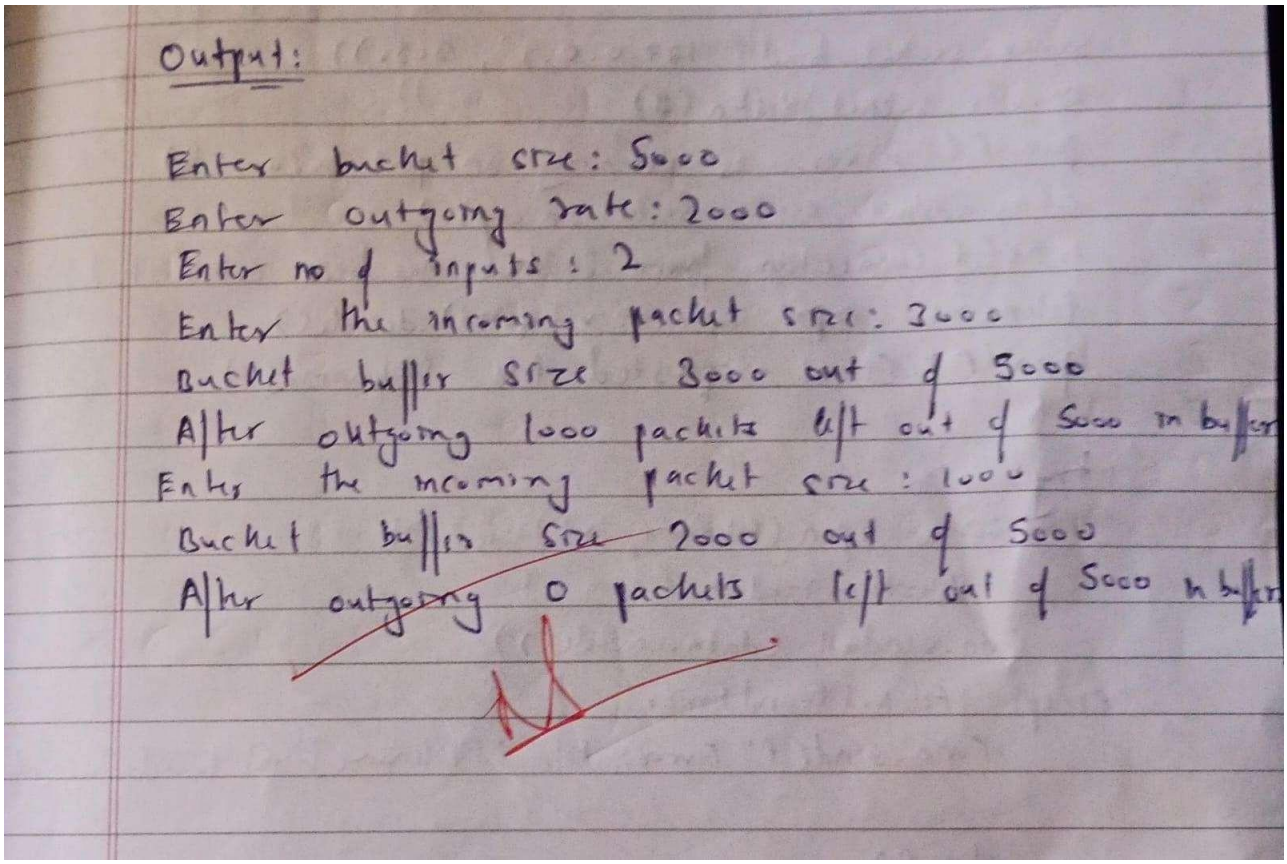
C - Code :

```
#include <stdio.h>

int main()
{
  int incoming, outgoing, buck_size, n, store=0;
  printf("Enter bucket size: ");
  scanf("%d", &buck_size);
  printf("Enter outgoing size: ");
  scanf("%d", &outgoing);
  printf("Enter number of inputs: ");
  scanf("%d", &n);

  while(n!=0)
  {
    printf("Enter the incoming bucket size: ");
    scanf("%d", &incoming);
    if(incoming <= (bucket_size - store))
    {
      store += incoming;
      printf("Bucket buffer size %d out of
          %d \n", store, buck_size);
    }
    else
    {
      printf("Dropped %d no of packets \n", incoming
          - (buck_size - store));

      printf("Bucket buffer size %d out of
          %d \n", store, buck_size);
      store = buck_size;
    }
```



```
store = store - outgoing;
printf("After outgoing %d packets left out
        of %d in buffer\n", store, bucket_size);
n--;
}
}
```

Output:



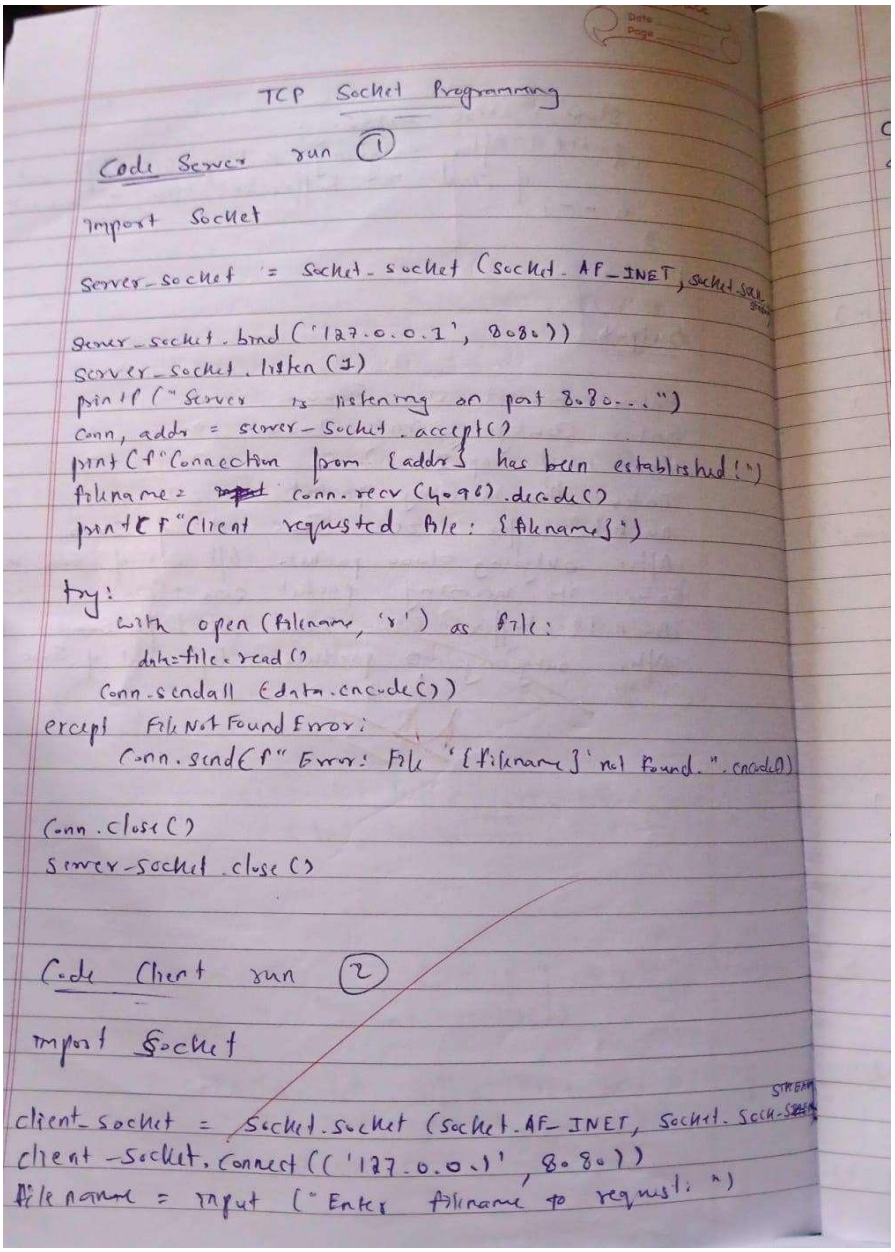
Output:

```
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:



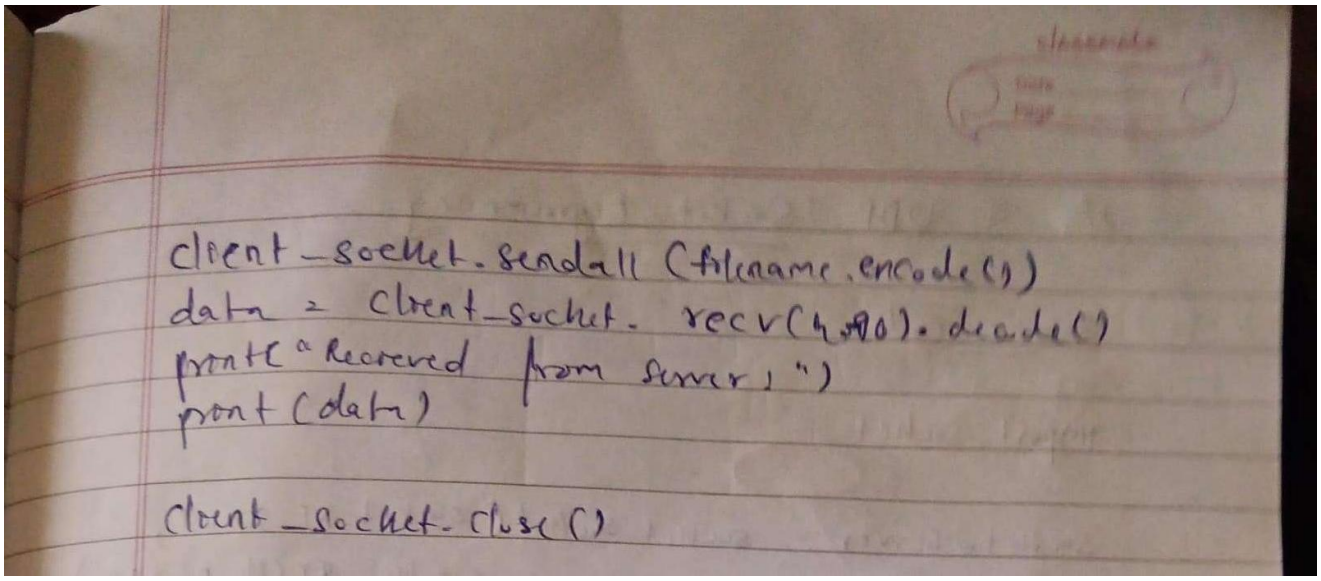
TCP Socket Programming

Code Server run (1)

```
Import Socket

Server_socket = Socket.socket (socket.AF_INET, socket.SOCK_STREAM)

Server_socket.bind ('127.0.0.1', 8080))
Server_socket.listen (1)
print ("Server is listening on port 8080...")
Conn, addr = server-socket.accept()
print(f"Connection from {addr} has been established!")
filename = conn.recv (4096).decode()
print(f"Client requested file: {filename}")

try:
    with open (filename, 'r') as file:
        data=file.read()
    Conn.sendall (data.encode())
except FileNotFoundError:
    Conn.send(f" Error: File '{filename}' not found.".encode())

Conn.close()
server-socket.close()
```

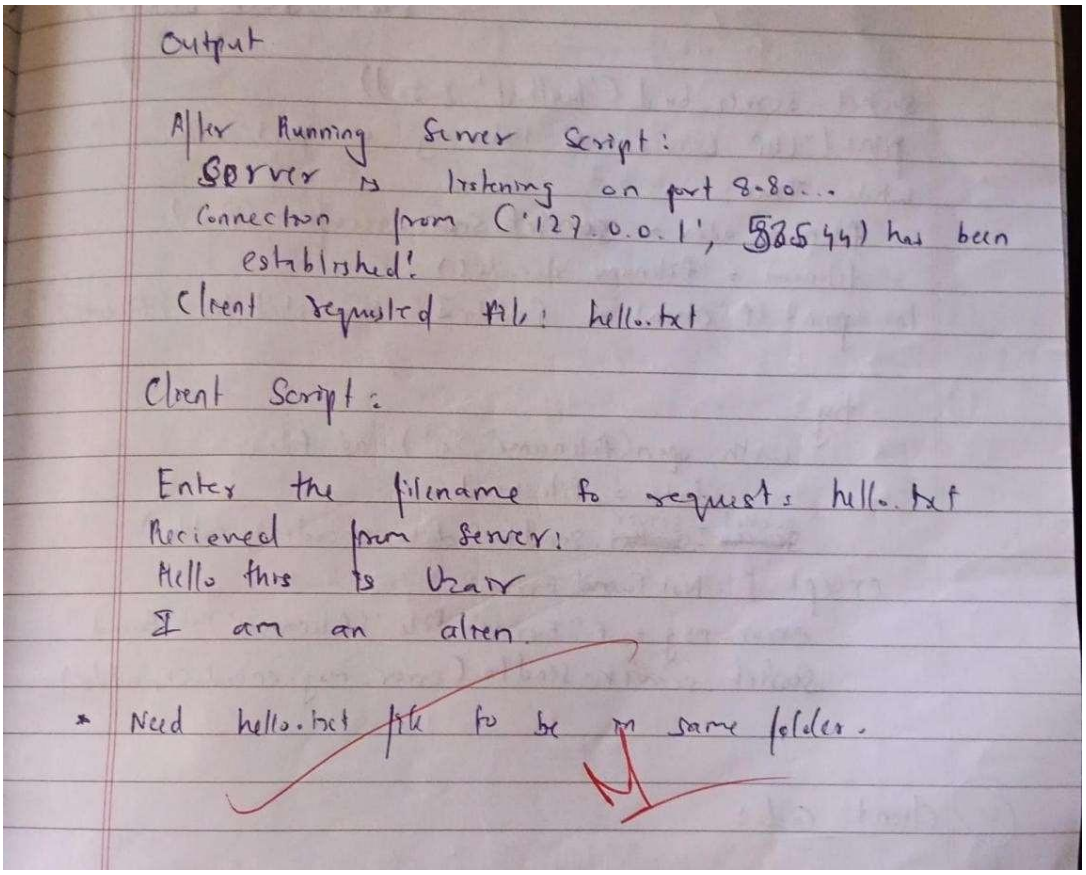
Code Client run (2)

```
import Socket

client_socket = Socket.socket (socket.AF_INET, Socket.SOCK_STREAM)
client-socket.Connect (('127.0.0.1', 8080))
filename = input ("Enter filename to request: ")
```



```
client_socket.sendall(filename.encode())
data = client_socket.recv(4096).decode()
print("Received from server:")
print(data)

client_socket.close()
```

Output:



Output

After Running Server Script:

Server is listening on port 8080...
Connection from ('127.0.0.1', 58544) has been established!
Client requested file: hello.txt

Client Script:

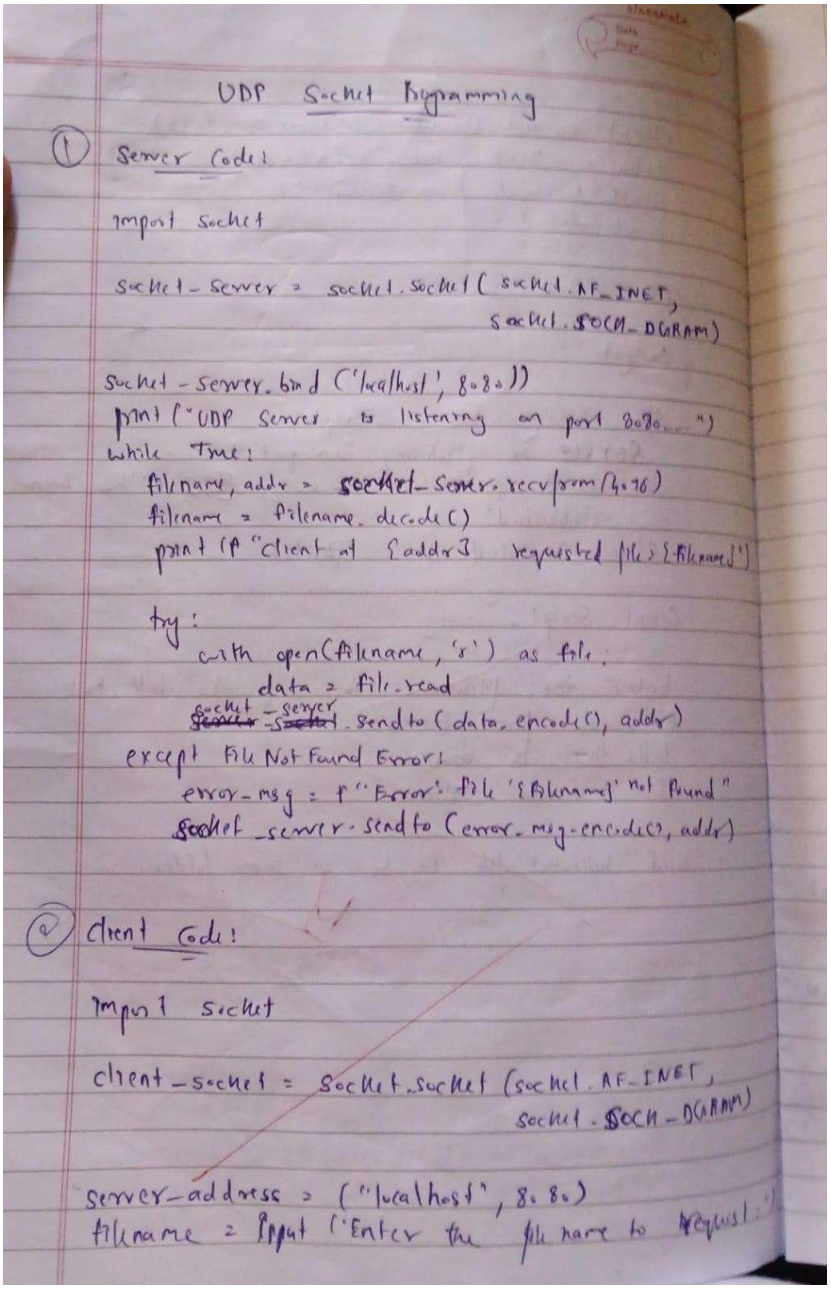
Enter the filename to request: hello.txt
Recieved from server:
Hello this is Uzair
I am an alien.

* Need hello.txt file to be in same folder.

Program 15

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 Programming

① Server Code:

```
import socket

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

socket_server.bind(('localhost', 8080))
print("UDP Server is listening on port 8080...")
while True:
    filename, addr = socket_server.recvfrom(4096)
    filename = filename.decode()
    print(f"Client at {addr} requested file: {filename}")

    try:
        with open(filename, 'r') as file:
            data = file.read
        socket_server.sendto(data.encode(), addr)
    except FileNotFoundError:
        error_msg = f"Error: File '{filename}' not found"
        socket_server.sendto(error_msg.encode(), addr)
```

② Client Code:

```
import socket

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

server_address = ("localhost", 8080)
filename = input("Enter the file name to request:")
```

```
client_socket.sendto(filename.encode(), serveraddress)
data, _ = client_socket.recvfrom(4096)

print("Received from Server:")
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 client