

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT

on

COMPUTER NETWORKS

Submitted by

PRASANNA KUMAR R (1BM21CS138)

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

June-2023 to September-2023

B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering



CERTIFICATE

This is to certify that the Lab work entitled “COMPUTER NETWORKS” carried out by **PRASANNA KUMAR R(1BM21CS138)**, who is bonafide student of **B.M.S. College of Engineering**. It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the academic semester June-2023 to September-2023. The Lab report has been approved as it satisfies the academic requirements in respect of a **COMPUTER NETWORKS (22CS4PCCON)** work prescribed for the said degree.

Swathi Sridharan

Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak

Professor and Head
Department of CSE
BMSCE, Bengaluru

INDEX SHEET

Experiment No.	Title	Page number
	CYCLE 1	
1	Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.	6-8
2	Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply	9-13
3	Configure default route, static route to the Router	14-17
4	Configure DHCP within a LAN and outside LAN.	18-21
5	Configure RIP routing Protocol in Routers	22-24
6	Configure OSPF routing protocol	25-27
7	Demonstrate the TTL/ Life of a Packet	28-29
8	Configure Web Server, DNS within a LAN.	30-31
9	To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)	32-33
10	To understand the operation of TELNET by accessing the router in server room from a PC in IT office	34-36
11	To construct a VLAN and make the PC's communicate among a VLAN	37-39
12	To construct a WLAN and make the nodes communicate wirelessly	40-42
	CYCLE 2	
13	Write a program for error detecting code using CRCCCITT (16-bits).	43-47

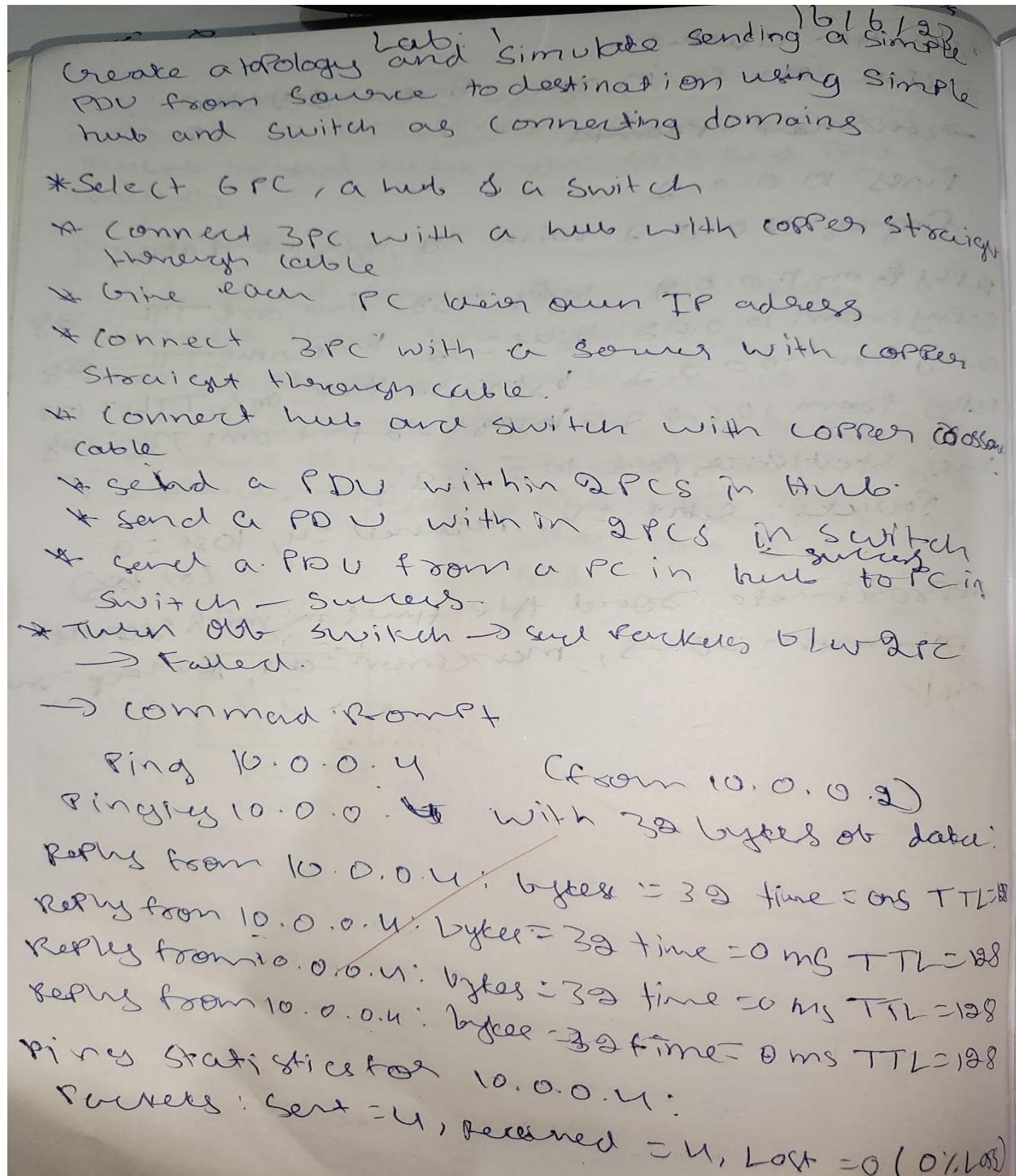
14	Write a program for congestion control using Leaky bucket algorithm.	48-50
15	Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present	51-53
16	Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.	54-56
17	Tool Exploration -Wireshark	57-58

Course Outcomes

CO1	Apply the fundamental concepts of communication in networking.
CO2	Analyze the various protocols, techniques in TCP/IP network architecture
CO3	Develop programs that demonstrate the functionalities of physical, Data Link, Network, Transport or Application layer

Experiment 1

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



Token Ring Switch

Prec 10.0.0.4

From 10.0.0.2

Ring 10.0.0.4 with 32 bytes of data:

Request time out

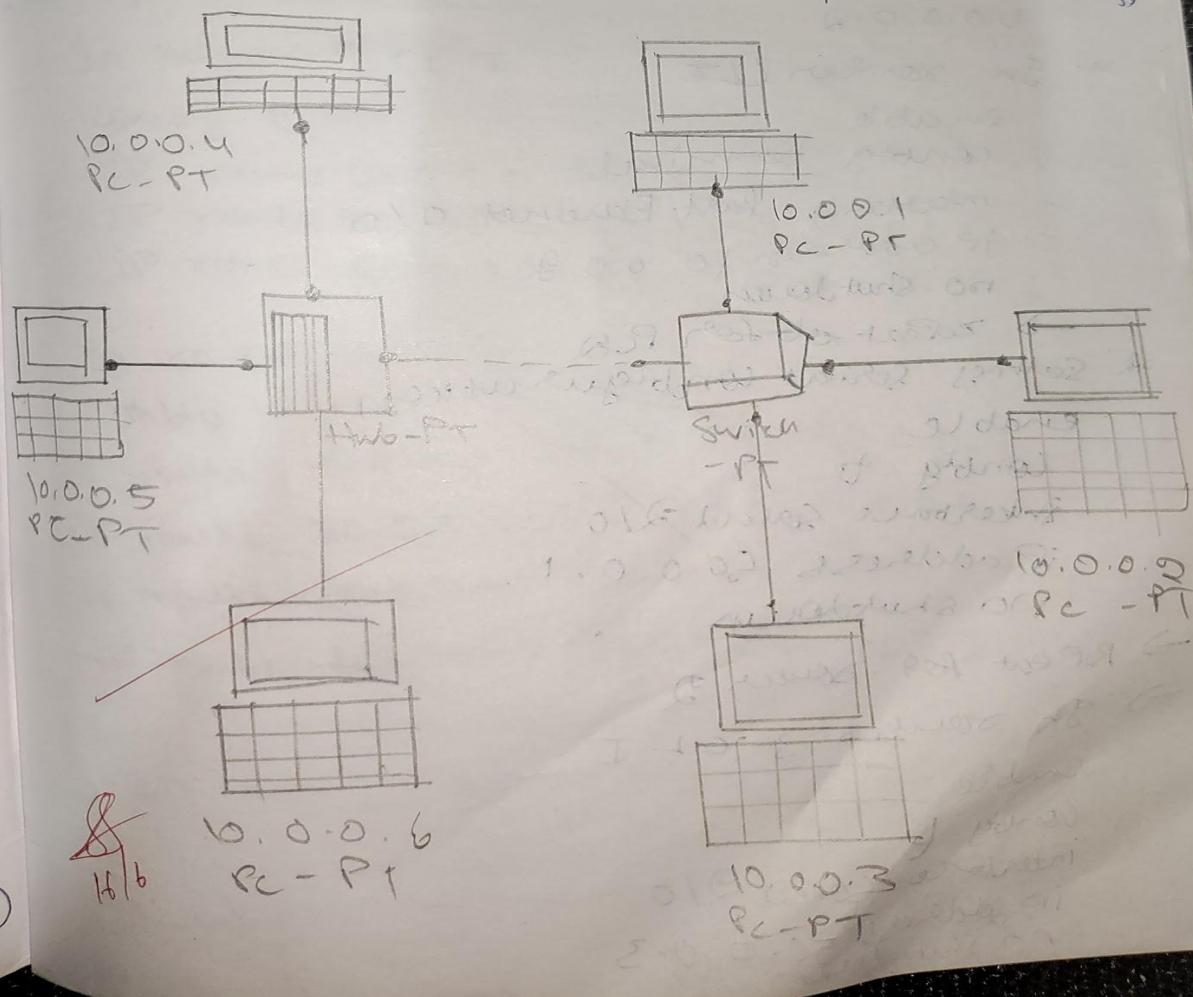
Received - Time out

Request - Time out

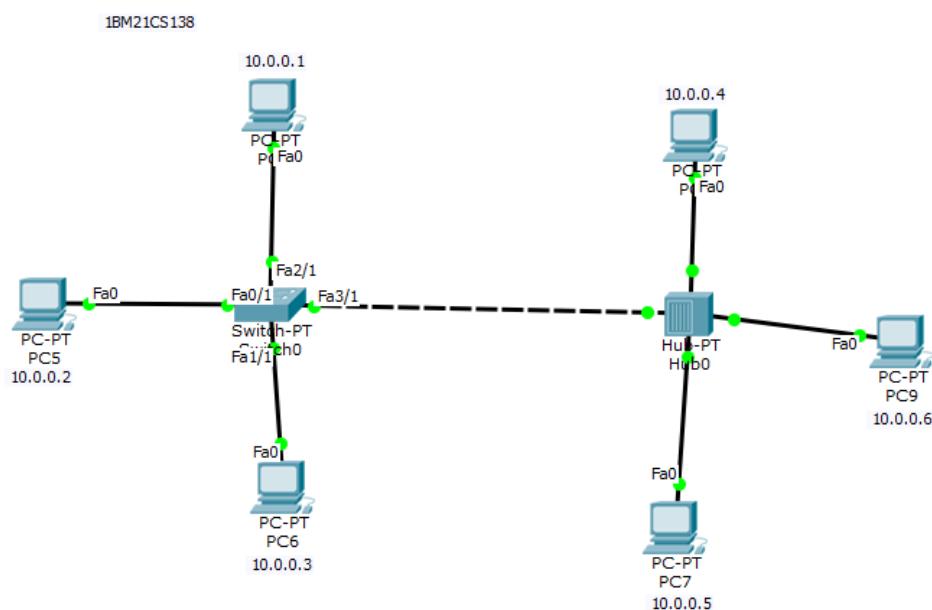
Request - Time out

Rx Statistics for 10.0.0.4:

Packets sent = 4, Received = 0, Lost = 4 (100% loss)



Topology:



Output:

PC2

Physical Config Desktop Programming Attributes

Command Prompt

```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 10.0.0.5

Pinging 10.0.0.5 with 32 bytes of data:

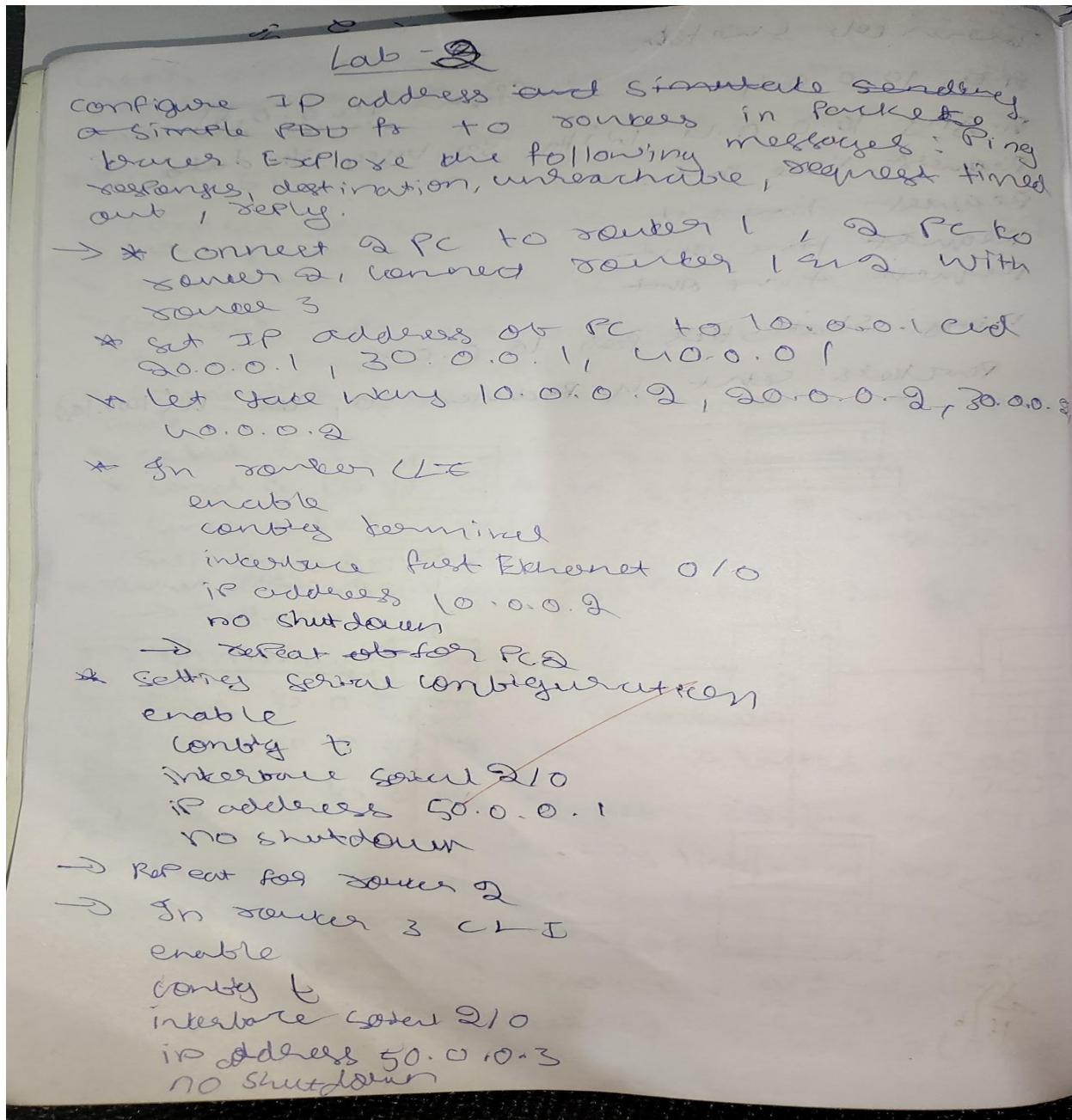
Reply from 10.0.0.5: bytes=32 time<1ms TTL=128

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 = 0ms, Average = 0ms

C:\>
```

Experiment 2

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



→ Interbase server 310
IP address 60.0.0.3
no shutdown

→ static routing

In router 1 CLT

Config t

IP route 30.0.0.0 255.0.0.0 50.0.0.3

IP route 40.0.0.0 255.0.0.0 50.0.0.3

IP route 60.0.0.0 255.0.0.0 50.0.0.3

Show IP route

In router 2 CLT

Config t

IP route 10.0.0.0 255.0.0.0 60.0.0.3

IP route 20.0.0.0 255.0.0.0 60.0.0.3

IP route 50.0.0.0 255.0.0.0 60.0.0.3

In router 3 CLT

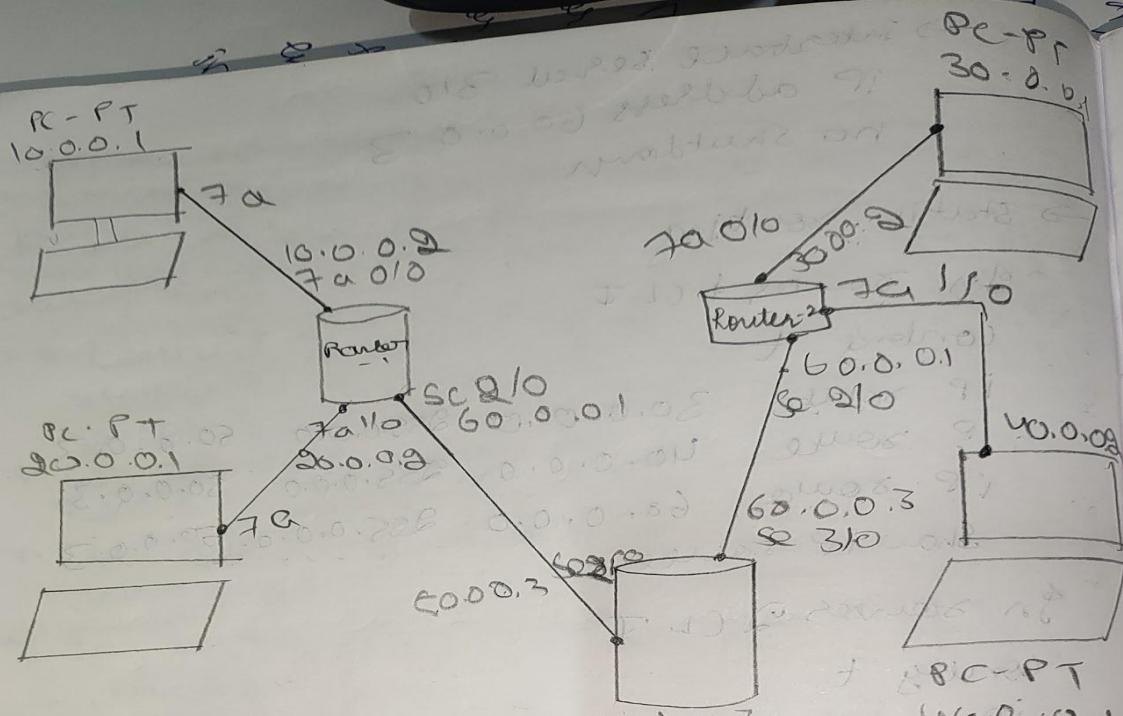
Config t

IP route 10.0.0.0 255.0.0.0 50.0.0.1

IP route 20.0.0.0 255.0.0.0 50.0.0.1

IP route 30.0.0.0 255.0.0.0 60.0.0.1

IP route 40.0.0.0 255.0.0.0 60.0.0.1



Outflow 0.000000.000 Date 31 Dec 1960 0.0.0.1

Aug 2014 08:00-08:20 0.00-0.00 30000 81

Piney \$30.00. 1

Pinging 30.0.0.1 with 30 bytes of data:

Replies from 10.0.0.2 : Destination host unreachable

Replies from 10.0.0.2: Destination host unreachable

seeing from 10.0.0.2) Default gateway is unreachable

Reply from 10.0.0.2: ~~host reached max unreachable~~

Pires Statistic $\frac{1}{2} \sum_{i=1}^n \text{dist}(x_i, \text{mean})$ where mean is unreachable

Pocket: Sent: 4, Recd: 10 for 30.0.019

Since $\Sigma \text{O}_2 = 1$, $\text{Levmed} = 0$, $\text{West} = 4$ (100% West)

30.0.0.1 is not connected to Yo 1
directly it should be statically registered

To view if you
in PC 1

Show IP route

- C 10.0.0.0/3 is directly connected, Fast Ethernet 0/0
- C 20.0.0.0/3 is directly connected, Fast Ethernet 0/0
- S 30.0.0.0/3 [1/0] via 50.0.0.3
- S 40.0.0.0/8 [1/0] via 50.0.0.3
- C 50.0.0.0/8 is directly connected, serial 0/0
- S 60.0.0.0/8 [1/0] via 50.0.0.3

In PC 1

Ping 30.0.0.1 and not usual should

Pinging 30.0.0.1 with 32 bytes or drake
request fired out

Reply from 30.0.0.1 : bytes=32 time=2ms TTL=125

Reply from 30.0.0.1 : bytes=32 time=2ms TTL=125

Reply from 30.0.0.1 : bytes=32 time=2ms TTL=125

Ping statistics for 30.0.0.1

Packets: sent = 4 received = 3 lost = 1 (25%)

Approximate round trip times in milliseconds

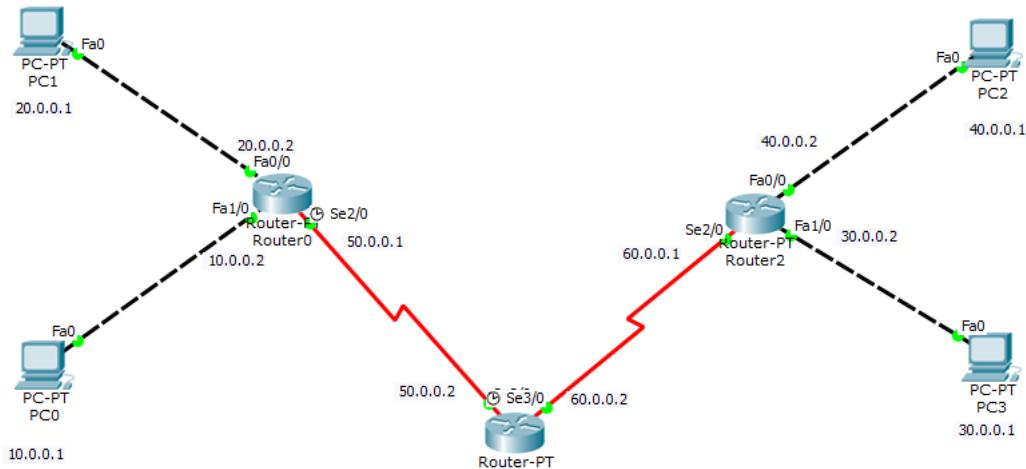
Minimum = 2ms, Maximum = 10ms, Average = 5ms

TTL is 125 since it takes time to identify

Packets

Topology:

BM21CS138



Output:

PC1

Physical	Config	<u>Desktop</u>	Programming	Attributes
----------	--------	----------------	-------------	------------

Command Prompt

```

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 = 0ms, Average = 0ms

C:\>ping 30.0.0.1

Pinging 30.0.0.1 with 32 bytes of data:

Reply from 20.0.0.2: Destination host unreachable.

Ping statistics for 30.0.0.1:
  Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
  
```

C:\>ping 50.0.0.1

```

Pinging 50.0.0.1 with 32 bytes of data:

Reply from 50.0.0.1: bytes=32 time<1ms TTL=255

Ping statistics for 50.0.0.1:
  Packets: Sent = 4, Received = 0, Lost = 0 (0% loss),
  Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping 50.0.0.2

Pinging 50.0.0.2 with 32 bytes of data:

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

Ping statistics for 50.0.0.2:
  Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
  
```

Experiment 3

Configure default route, static route to the Router

301612

Lab : 03
Configure default route, static routes
to the router
* connect devices as shown in topology
* set IP addresses to all PC and router.

Router enable config + interface fastethernet 0/0 or serial 0/0
IP address 10.0.0.1 255.0.0.0
no shut

* Default route from the Router 1, Router 5,
Router 3
→ ① ip route 0.0.0.0 0.0.0.0 10.0.0.1
enable config + ip source 0.0.0.0 0.0.0.0 60.0.0.3
→ ② ip route 0.0.0.0 0.0.0.0 100.0.0.1
→ ③ ip route 0.0.0.0 0.0.0.0 100.0.0.1
* static routing from Router 0, Router 6, Router 4, Router 2.
→ ①
ip route 20.0.0.0 255.0.0.0 60.0.0.1
ip route 30.0.0.0 255.0.0.0 70.0.0.1
ip route 40.0.0.0 255.0.0.0 70.0.0.1
ip route 50.0.0.0 255.0.0.0 70.0.0.1
ip route 80.0.0.0 255.0.0.0 70.0.0.1
ip route 100.0.0.0 255.0.0.0 70.0.0.1

if route 110.0.0.0 0.0.0.0 70.0.0.1
Similarly for rest routers

→ output :-

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=7ms TTL=126

Reply from 20.0.0.1 : bytes=32 time=6ms TTL=126

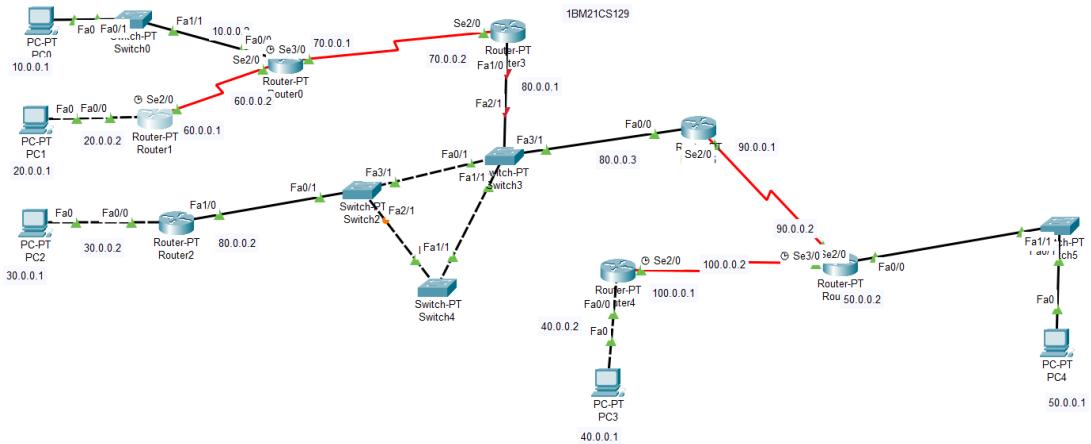
Reply from 20.0.0.1 : bytes=32 time=7ms TTL=126

ping statistics for 20.0.0.1 :

packets : sent=4, received=3, lost=1 (25%),
minimum=6ms, maximum=7ms, average=6ms

9/10
15/17/23

Topology:



Output:

```

Physical Config Desktop Programming Attributes
Command Prompt
Received: 0 bytes by received 0 bytes to 0 (000-1000)
C:\>ping 90.0.0.2
Pinging 90.0.0.2 with 32 bytes of data:
Reply from 90.0.0.2: bytes=32 time=12ms TTL=252
Reply from 90.0.0.2: bytes=32 time=2ms TTL=252
Reply from 90.0.0.2: bytes=32 time=10ms TTL=252
Reply from 90.0.0.2: bytes=32 time=2ms TTL=252

Ping statistics for 90.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 12ms, Average = 6ms

C:\>ping 40.0.0.1
Pinging 40.0.0.1 with 32 bytes of data:
Request timed out.
Reply from 40.0.0.1: bytes=32 time=3ms TTL=123
Reply from 40.0.0.1: bytes=32 time=3ms TTL=123
Reply from 40.0.0.1: bytes=32 time=3ms TTL=123

Ping statistics for 40.0.0.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 3ms, Maximum = 3ms, Average = 3ms

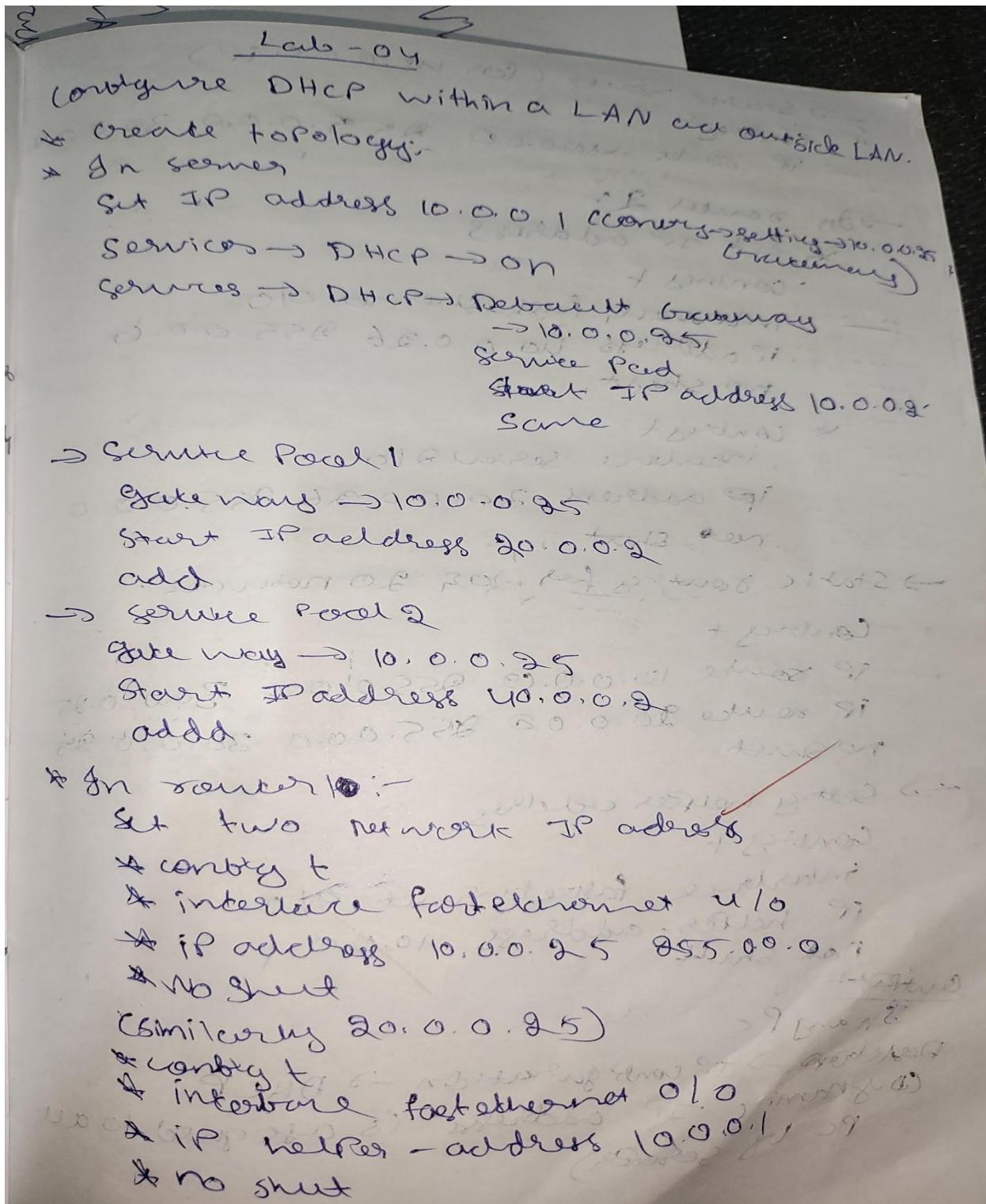
C:\>ping 50.0.0.1
Pinging 50.0.0.1 with 32 bytes of data:
Request timed out.
Reply from 50.0.0.1: bytes=32 time=3ms TTL=124
Reply from 50.0.0.1: bytes=32 time=6ms TTL=124
Reply from 50.0.0.1: bytes=32 time=2ms TTL=124

Ping statistics for 50.0.0.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 6ms, Average = 3ms
C:\>

```

Experiment 4

Configure DHCP within a LAN and outside LAN



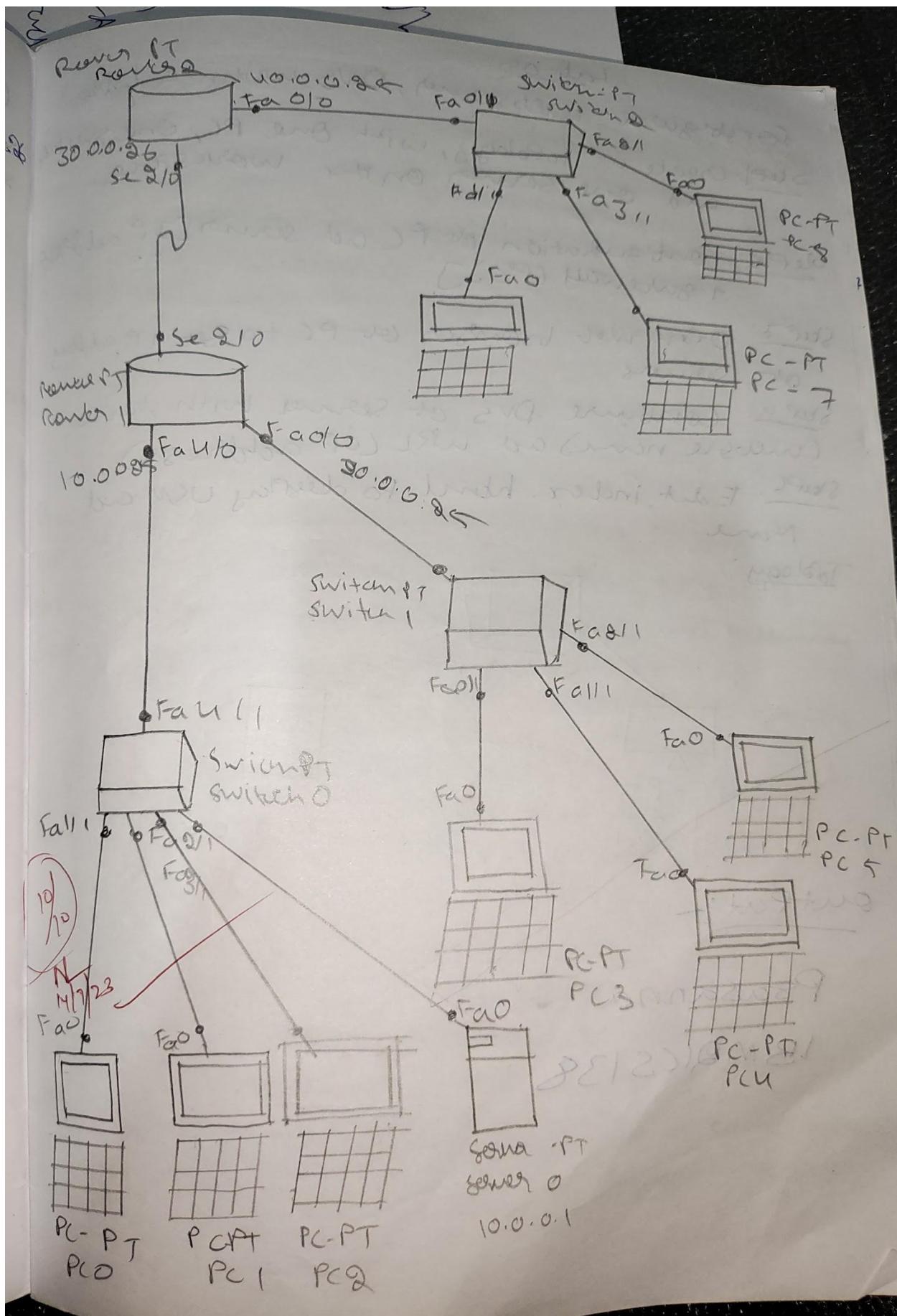
→ Static source (for 10.0.0.0 IP)
Config +
IP source 10.0.0.0 255.0.0.0
no shut

→ In router 2
* Set IP address
Config +
interface fast ethernet 0/0
IP address 10.0.0.26 255.0.0.0
no shut
* Config +
interface serial 0/0
IP address 200.0.0.26 255.0.0.0
no shut

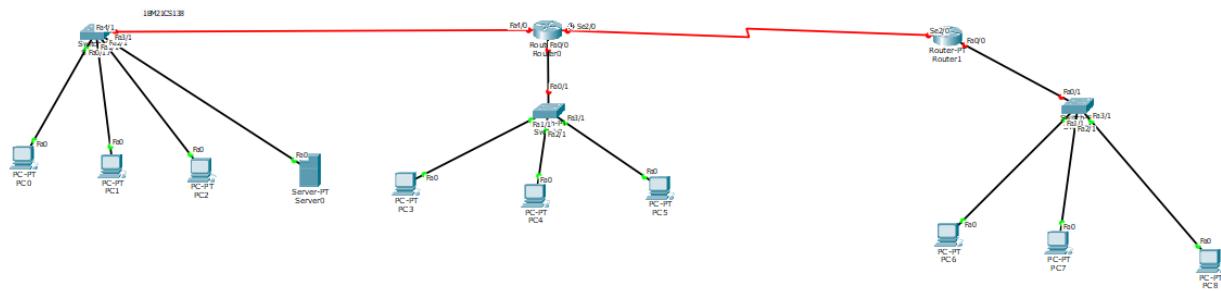
→ Static routes for 10.0.20 network
Config +
IP source 10.0.0.0 255.0.0.0
IP source 20.0.0.0 255.0.0.0
no shut

→ Setting helper address
Config +
interface fast ethernet 0/0
IP helper-address 10.0.0.1
no shut

Output
In any PC
Desktop → IP configuration → DHCP
Dynamic IP address is assigned to all
PC by server



Topology:



Output:

Screenshot of a network configuration interface, likely from a Cisco device's CLI or a management tool like WinBox. The interface is titled 'IP Configuration' and shows settings for 'FastEthernet0'.

IP Configuration

Interface: FastEthernet0

IP Configuration

DHCP Static

IPv4 Address: 10.0.0.2

Subnet Mask: 255.0.0.0

Default Gateway: 0.0.0.0

DNS Server: 0.0.0.0

IPv6 Configuration

Automatic Static

IPv6 Address: /

Link Local Address: FE80::260:2FFF:FE75:6CE1

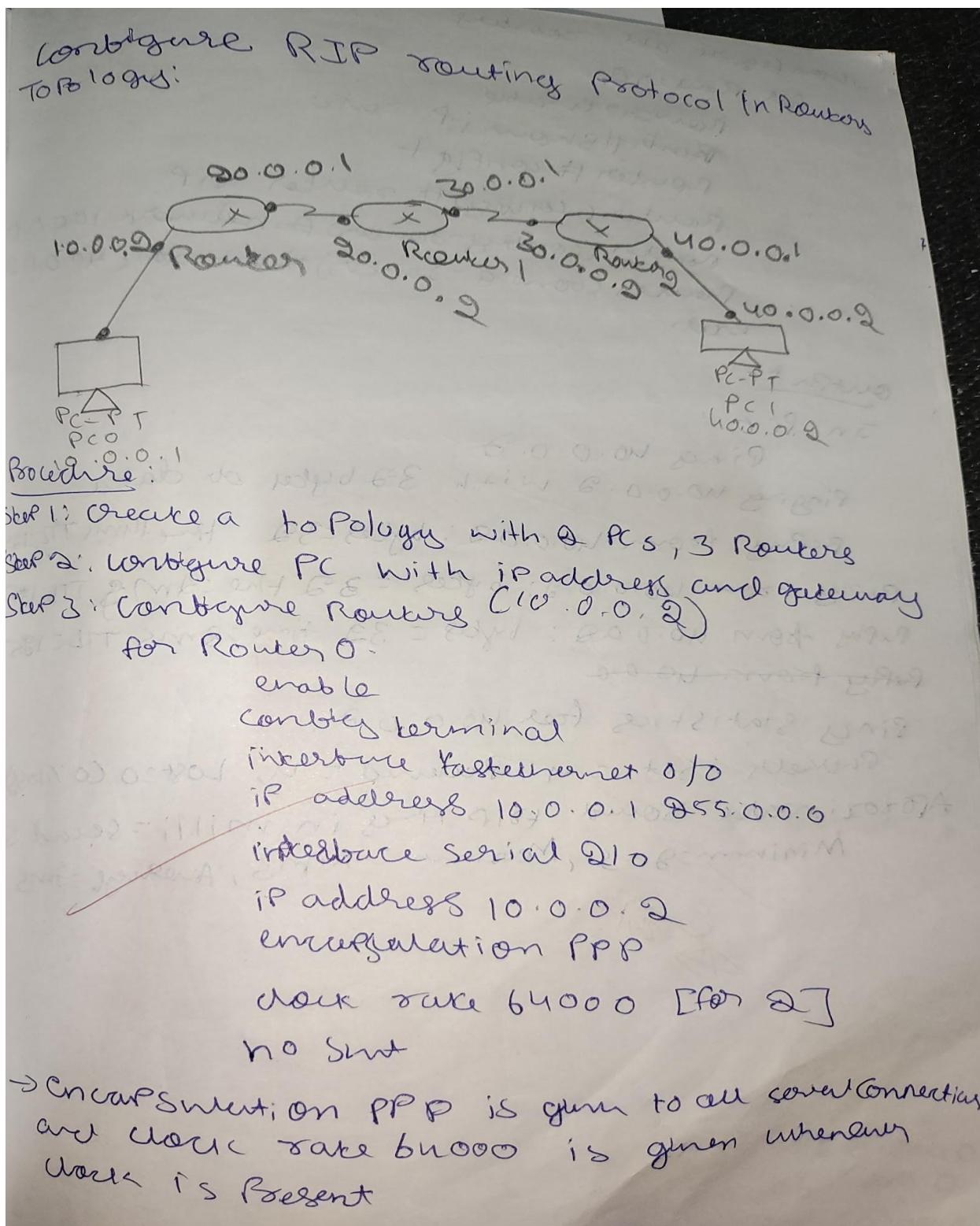
Default Gateway:

DNS Server:

802.1X

Experiment 5

Configure RIP routing Protocol in Routers



configure all routers
for Router 0,
Router> enable
Router# show ip route
Router# config +
Router(config)# router rip
router (config-router)# network 10.0.0.0
Router (config-router)# network 20.0.0.0
exit

Output:

In PC0

Ping 10.0.0.2
Pinging 10.0.0.2 with 32 bytes of data.
Reply from 10.0.0.2: bytes=32 time=2ms TTL=151
Reply from 10.0.0.2: bytes=32 time=2ms TTL=151
Reply from 10.0.0.2: bytes=32 time=2ms TTL=151
Reply from 10.0.0.2: bytes=32 time=2ms TTL=151

Ping Statistics from 10.0.0.2 lines

Packets: sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milliseconds

Min=2ms, Max=2ms, Average=2ms

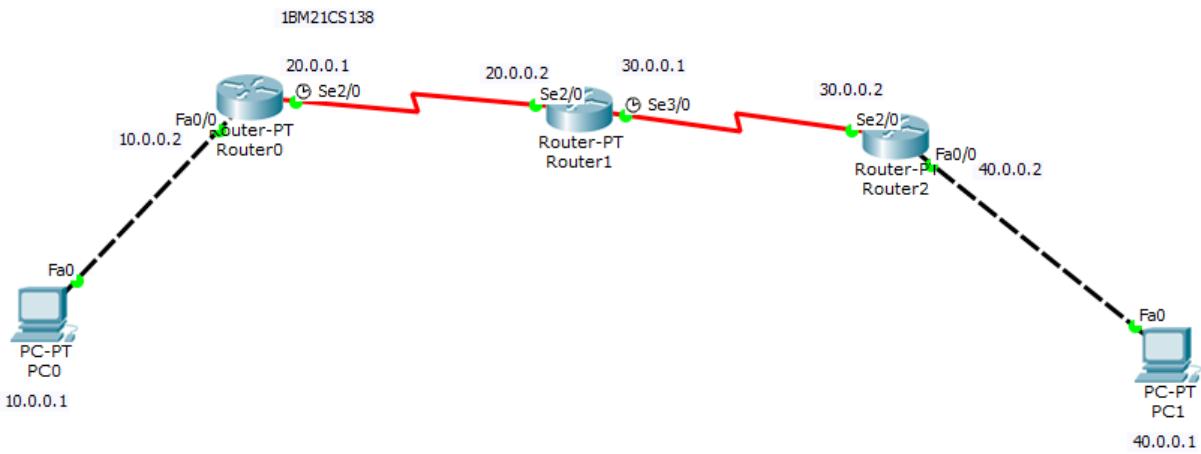
Latency 0.01 seconds
997 no packets lost

For next second also was

time on

minimum time of ping is 0.78 maximum time of ping is 1.14
average time of ping is 0.99 second also small was

Topology:



Output:

```
Physical Config Desktop Programming Attributes
Command Prompt
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data:

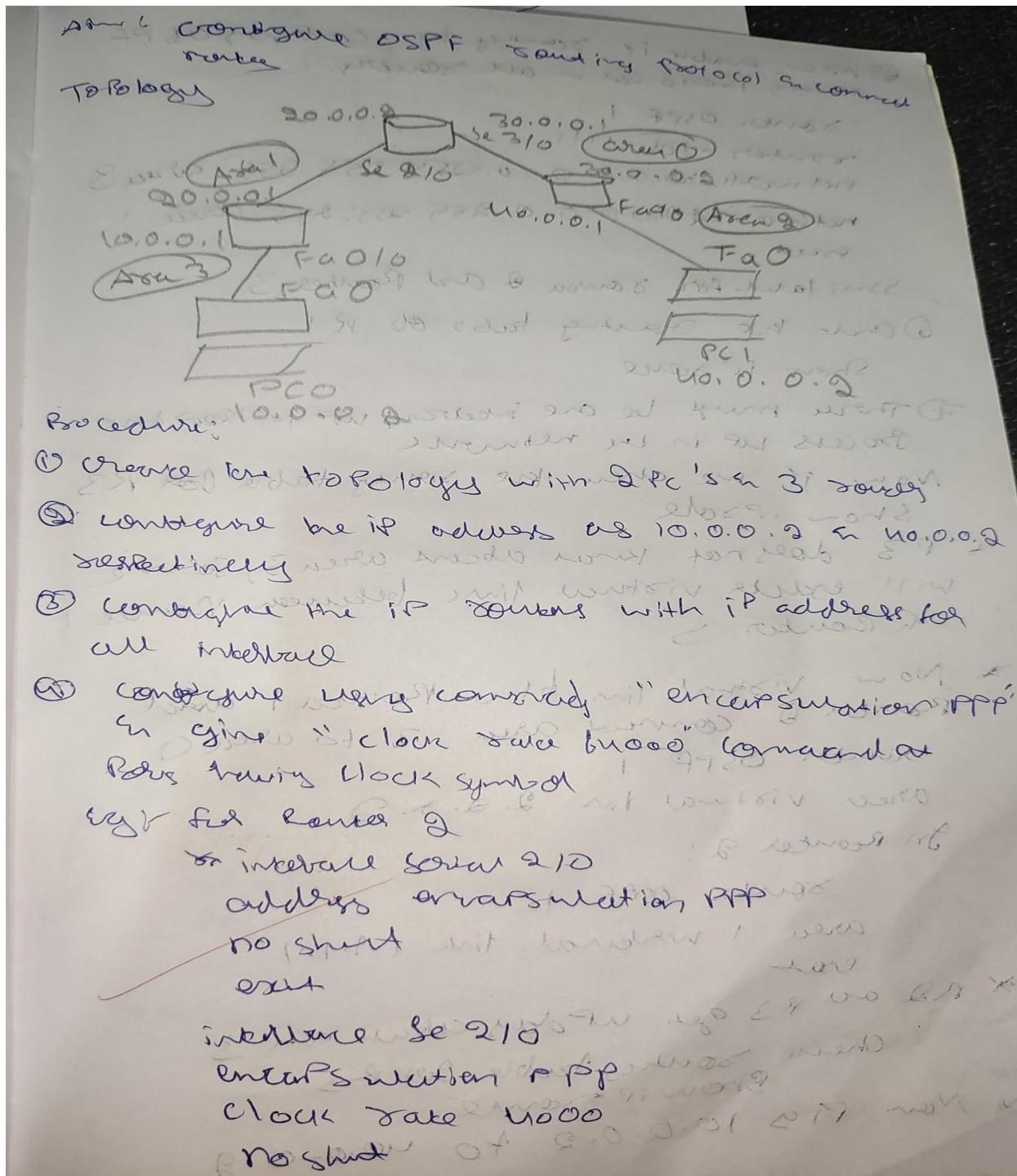
Request timed out.
Reply from 20.0.0.1: bytes=32 time=24ms TTL=125
Reply from 20.0.0.1: bytes=32 time=25ms TTL=125
Reply from 20.0.0.1: bytes=32 time=25ms TTL=125

Ping statistics for 20.0.0.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 24ms, Maximum = 25ms, Average = 24ms

C:\>
```

Experiment 6

Configure OSPF routing protocol



Q Now, enable is running by 20 ~~Configuring~~ OSPF
on all routers.

Router OSPF 1

Router-id 1.1.1.1

Network 10.0.0.0 0.255.255.255 area 0

Network 10.0.0.0 0.255.255.255 area 1

exit

Similar for Router 2 and Router 3

Q And the routing table of R1

Show IP route

There must be one process up to keep the process up in the network

Now if we check the routing table for R3

Show IP route

R3 does not know about area 0 so we will extract virtual link between Router 2 and Router 3

* Now virtual link between Router 1 and Router 2 connect area 0 to area 0

Router OSPF 1

Area virtual link 2.2.0.2

In Router 2:

Router OSPF 2

Area 1 virtual link 1.1.0.1

exit

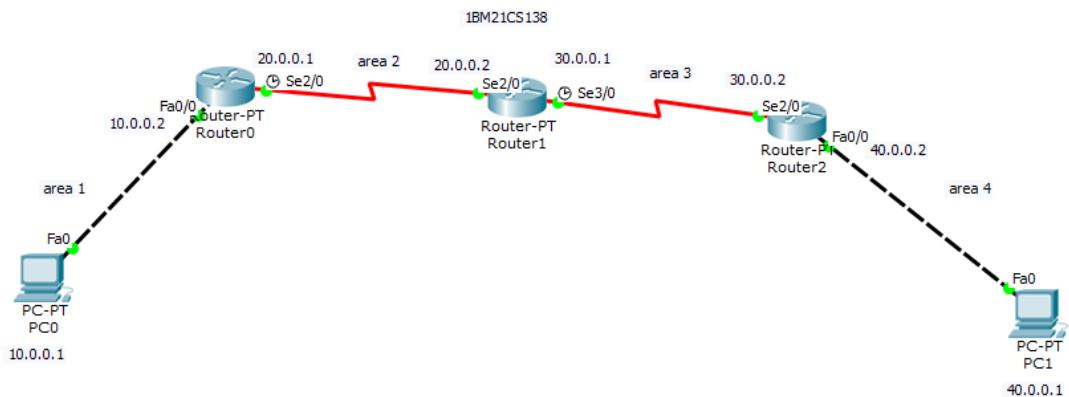
* R2 and R3 get update about area 0

Check routers table for R3

Show IP route

Now R3 10.0.0.2 to 40.0.0.72

Topology:



Output:

```
Physical Config Desktop Programming Attributes
Command Prompt
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 40.0.0.2

Pinging 40.0.0.2 with 32 bytes of data:

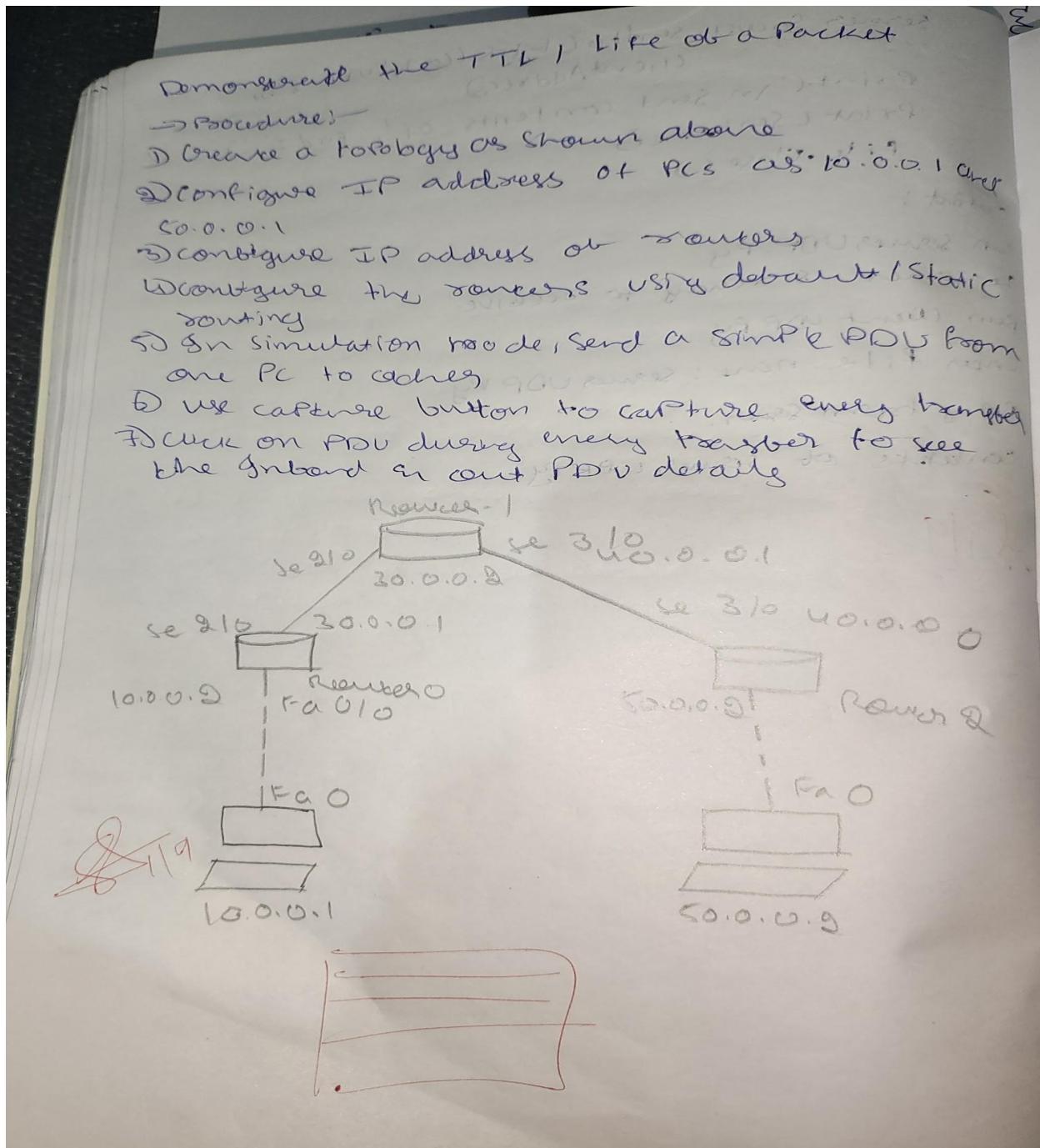
Request timed out.
Reply from 40.0.0.2: bytes=32 time=2ms TTL=125
Reply from 40.0.0.2: bytes=32 time=21ms TTL=125
Reply from 40.0.0.2: bytes=32 time=24ms TTL=125

Ping statistics for 40.0.0.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 24ms, Average = 15ms

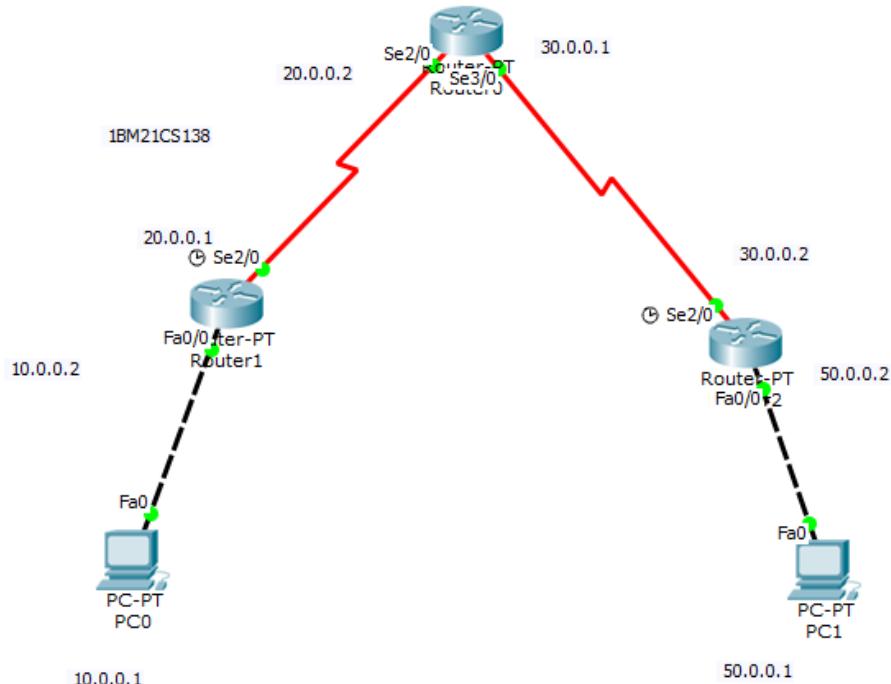
C:\>
```

Experiment 7

Demonstrate the TTL/ Life of a Packet

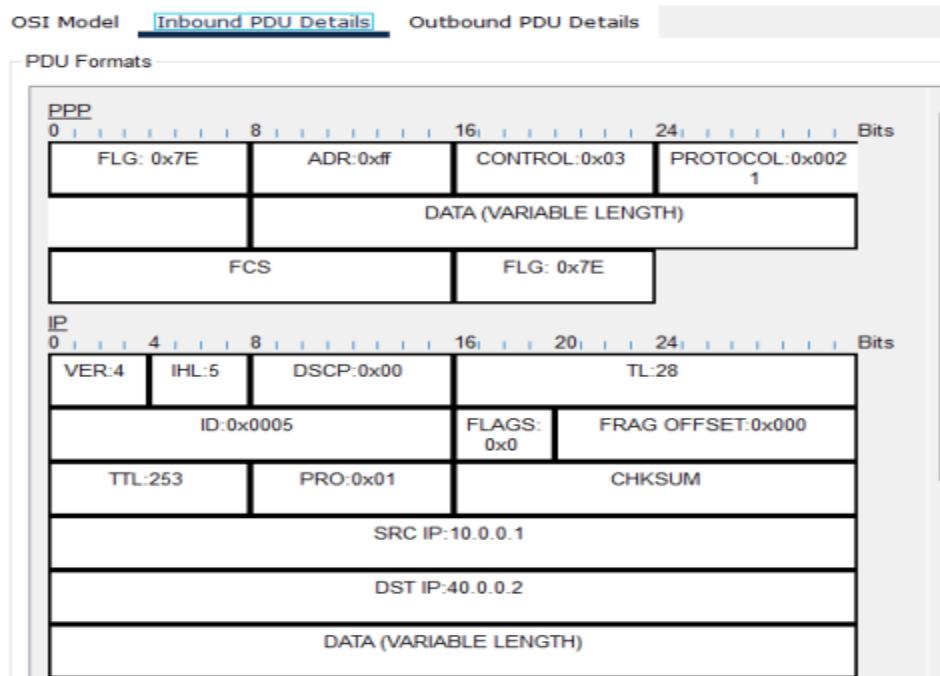


Topology:



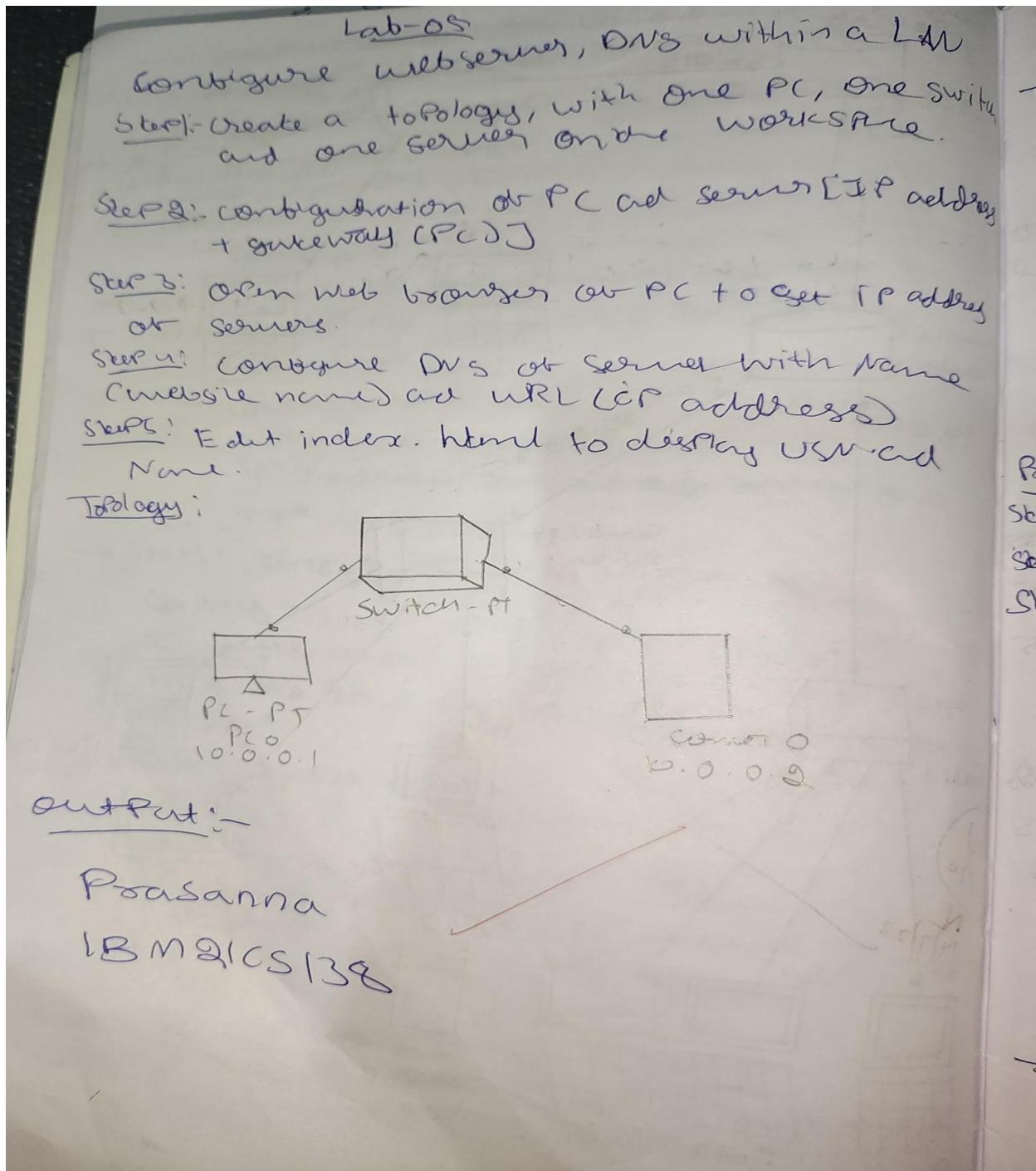
Output:

PDU Information at Device: Router2

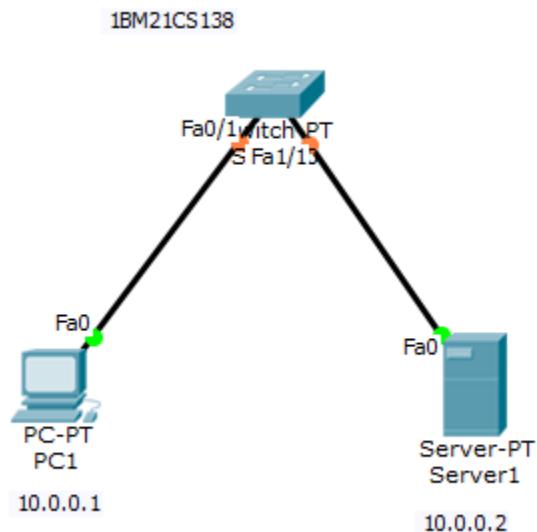


Experiment 8

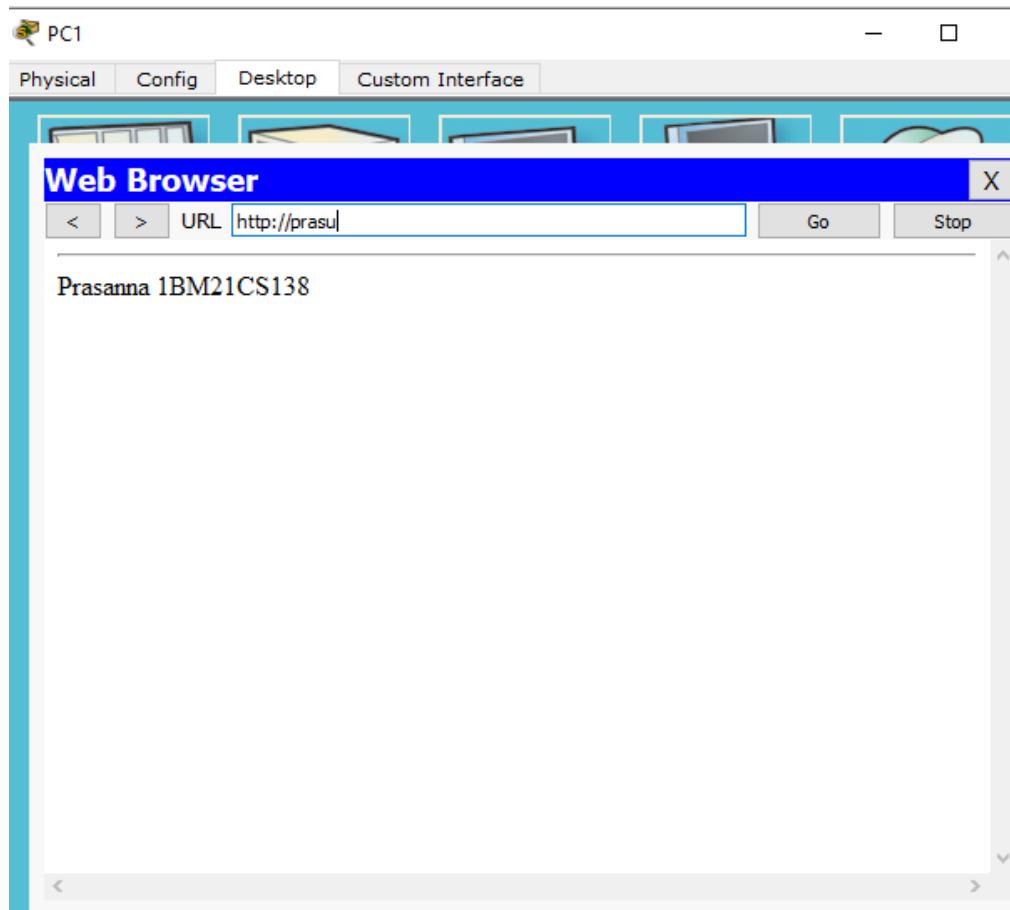
Configure Web Server, DNS within a LAN.



Topology:

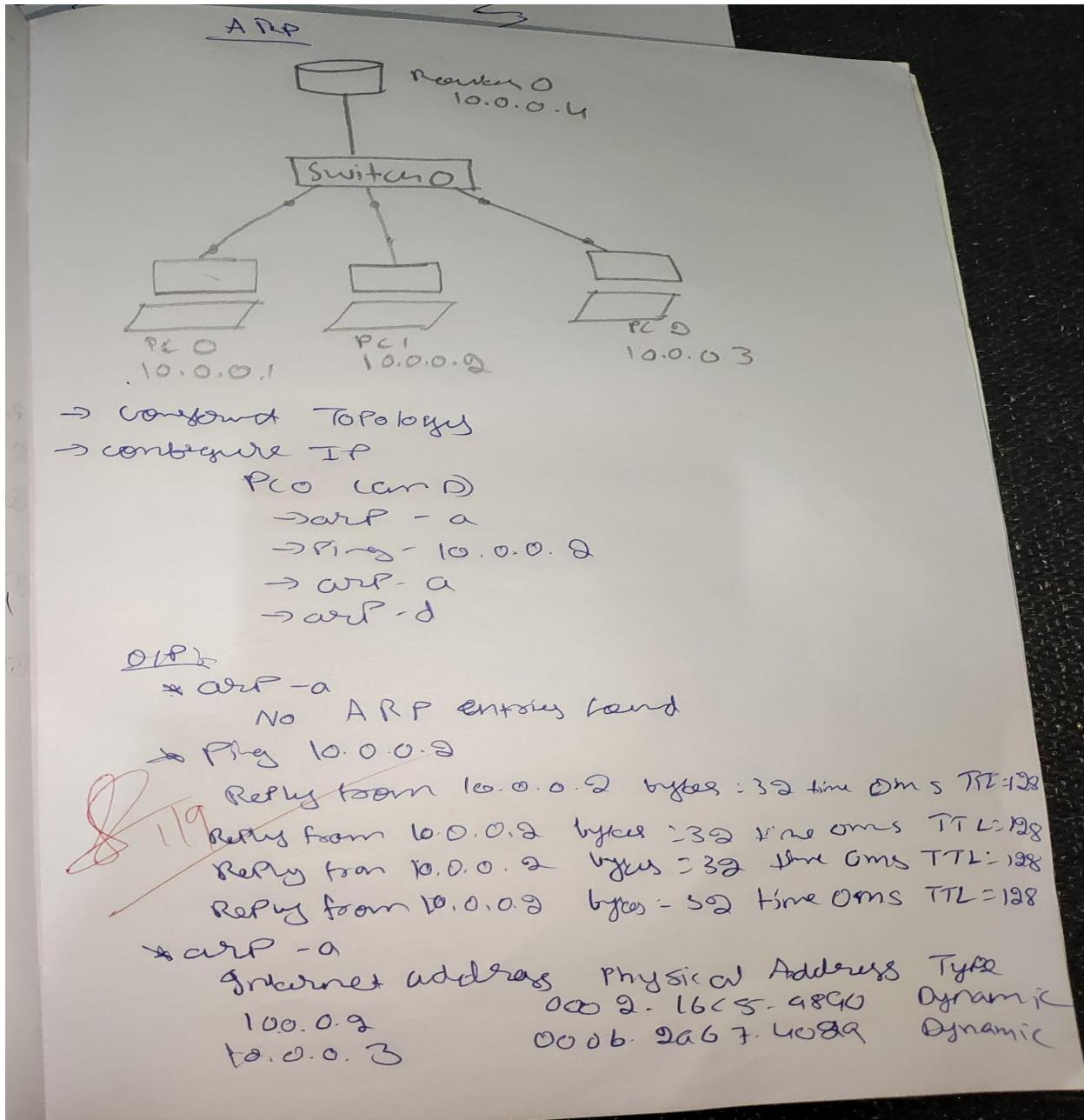


Output:

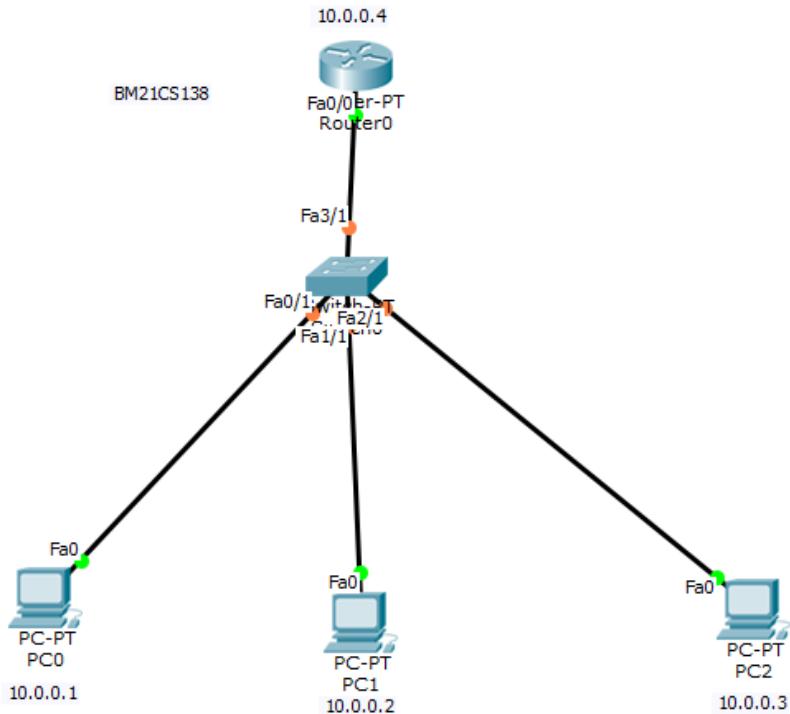


Experiment 9

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



Topology:



Output:

```
Physical | Config | Desktop | Custom Interface | X
Command Prompt
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 = 8ms, Average = 2ms
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=0ms TTL=128

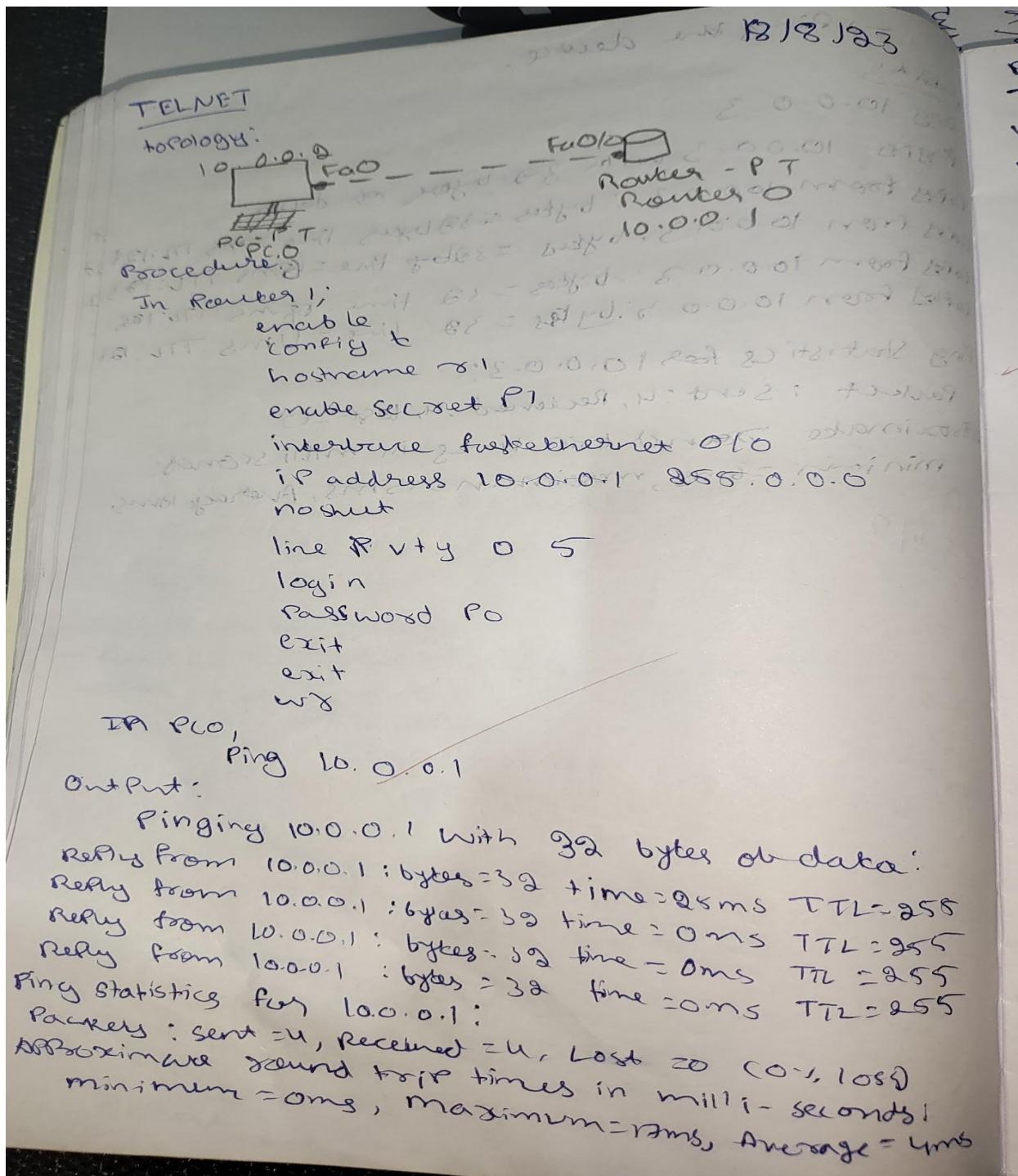
Ping statistics for 10.0.0.3:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
  Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
PC>arp -p
Invalid Command.

PC>arp -a
  Internet Address        Physical Address          Type
  10.0.0.2                  0001.962d.9094      dynamic
  10.0.0.3                  00e0.b0a9.ce08      dynamic

PC>
```

Experiment 10

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



PC \rightarrow telnet 10.0.0.1

Trying 10.0.0.1 [eth0] - Open

User Access Verification

Password: P0

R1 > enable

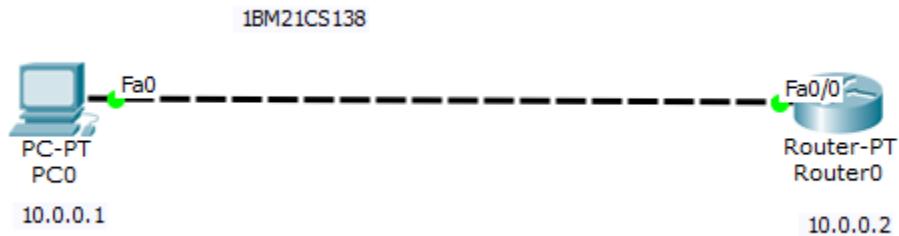
Password: P1

R1# show ip route

10.0.0.0/8 is directly connected, FastEthernet 0/0

8/8

Topology:



Output:

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=96ms 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 = 96ms, Average = 24ms

PC>telnet 10.0.0.2
Trying 10.0.0.2 ... Open

User Access Verification

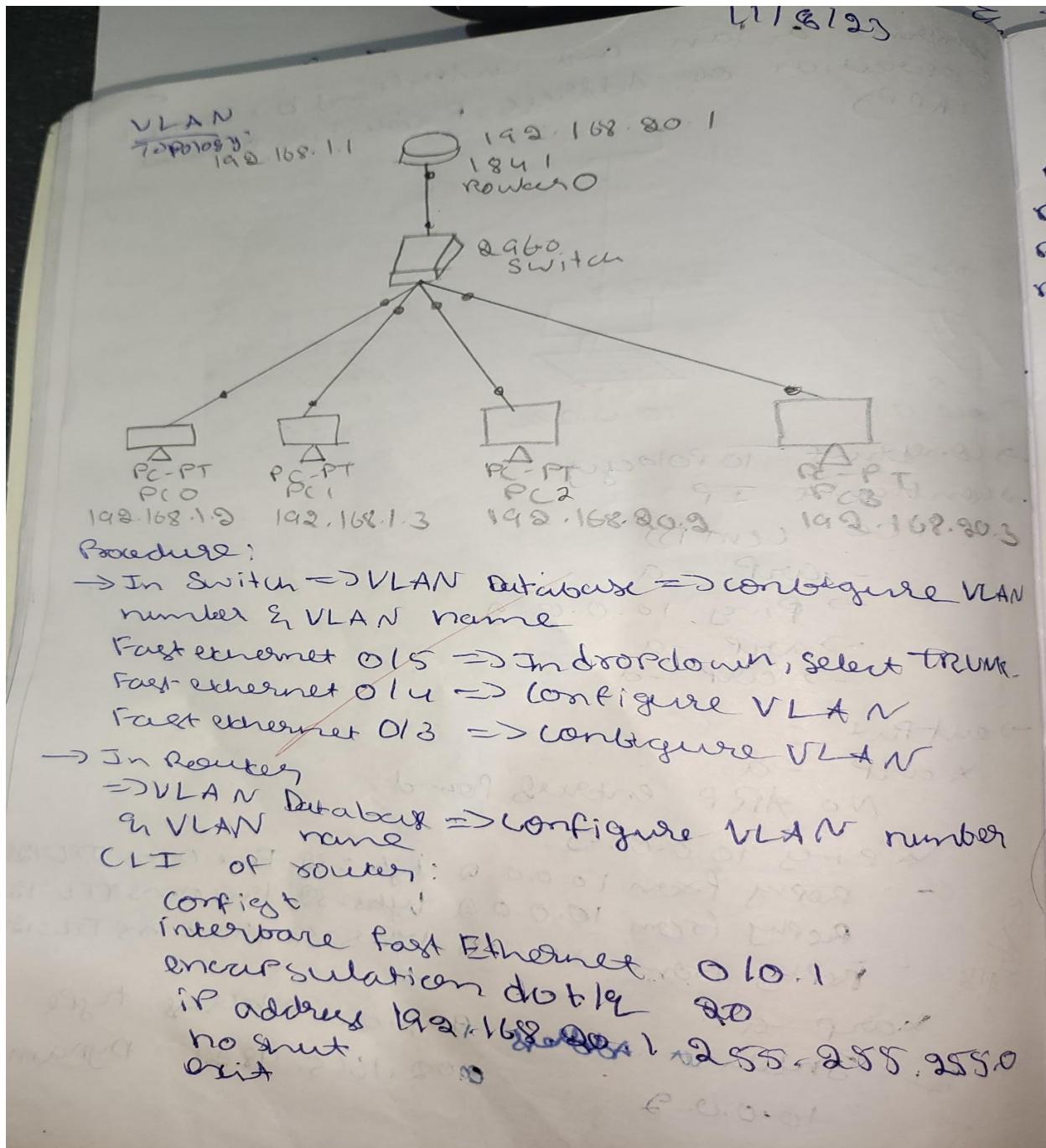
Password:
prasu>enable
Password:
prasu#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - 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
prasu#
```

Experiment 11

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



Output

Ping 192.168.20.3

Pinging 192.168.20.3

Reply from 192.168.20.3: bytes = 32 time=0ms TTL=127

Ping statistics for 192.168.20.3:

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

centrosoy

Updated and shielding

259 to provide better details 97-13-12-5

10-2102000 1076-

1000000 =

MANC = 0123

OPBODANSEIGER WITH

109 00

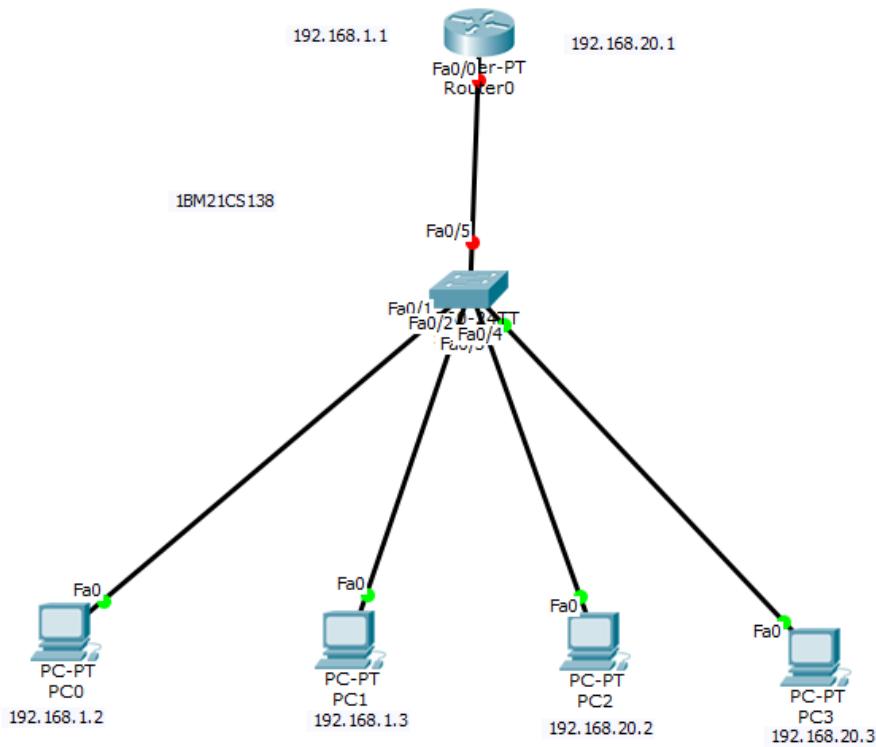
WHL-WM-T60H-T9 printed out for the meeting
at 1000 89MW part was taken away out of
service and no service most. 100% up and
running

0 failures 0%

MANC = 0123

OPBODANSEIGER WITH
MANC

Topology:



Output:

```
C:\>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

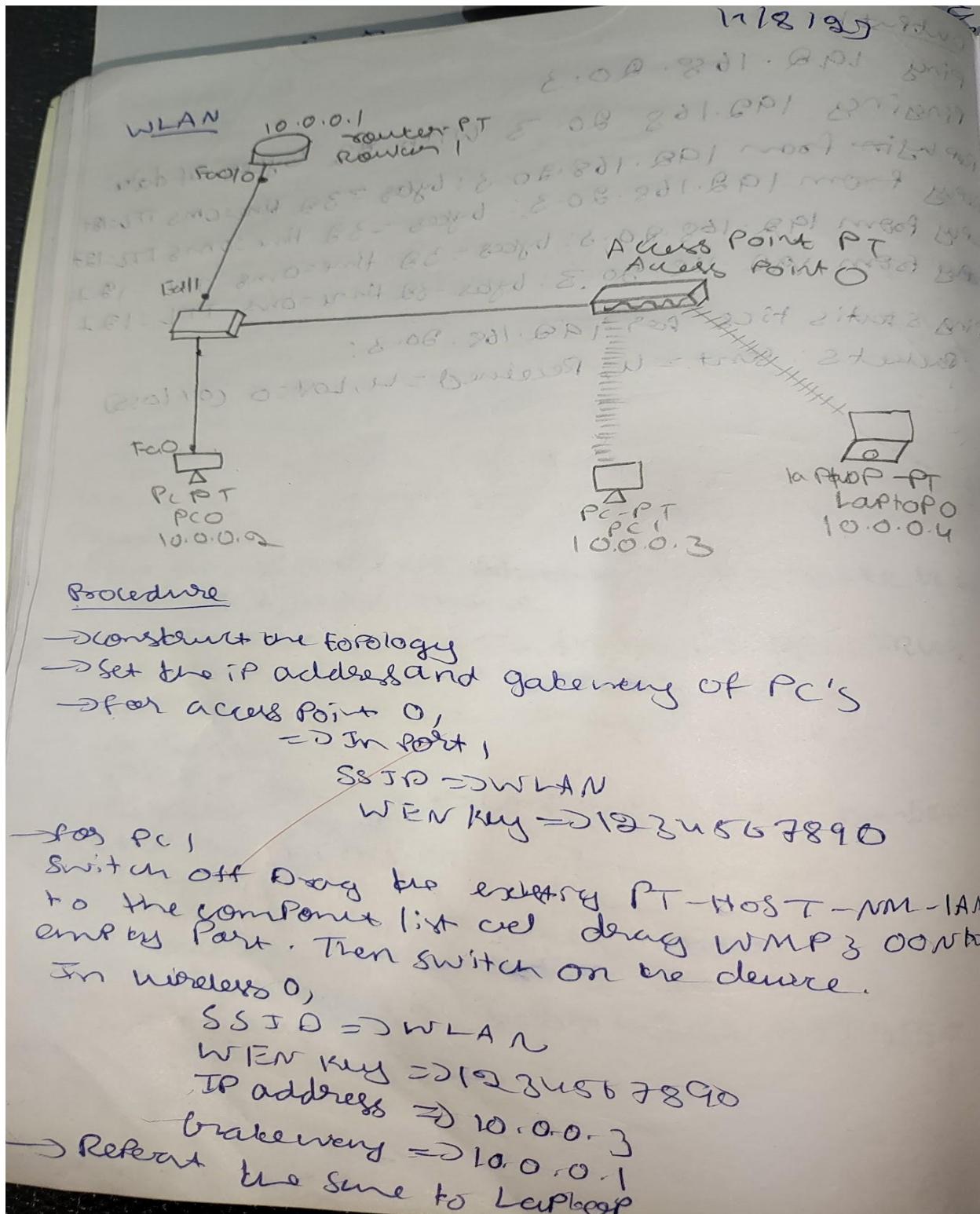
Request timed out.
Reply from 192.168.2.2: bytes=32 time<1ms TTL=127
Reply from 192.168.2.2: bytes=32 time<1ms TTL=127
Reply from 192.168.2.2: bytes=32 time<1ms TTL=127

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 = 0ms, Average = 0ms

C:\>
```

Experiment 12

To construct a WLAN and make the nodes communicate wirelessly



→ Pives the device.

Dent Dent 13

Ring 10-0-0-3

Pingers 10.0.0-3 with 32 bytes of data.
Reply from 10.0.0.3

Replies from 10.0.0.3: bytes = 32 bytes time = 15ms TTL = 128
Reply from 10.0.0.3: bytes = 32 bytes time = 15ms TTL = 128

Delly form 10.0.0.3 byees = 32 time = 16ms TTL: 198

reply from 10.0.0.3 by tics = 32 time = 10ms TTL: 128

Ping Statistics for 192.0.0.3 - connection

8 days infestation 20% mortality

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

Approximate round trip times in milliseconds-

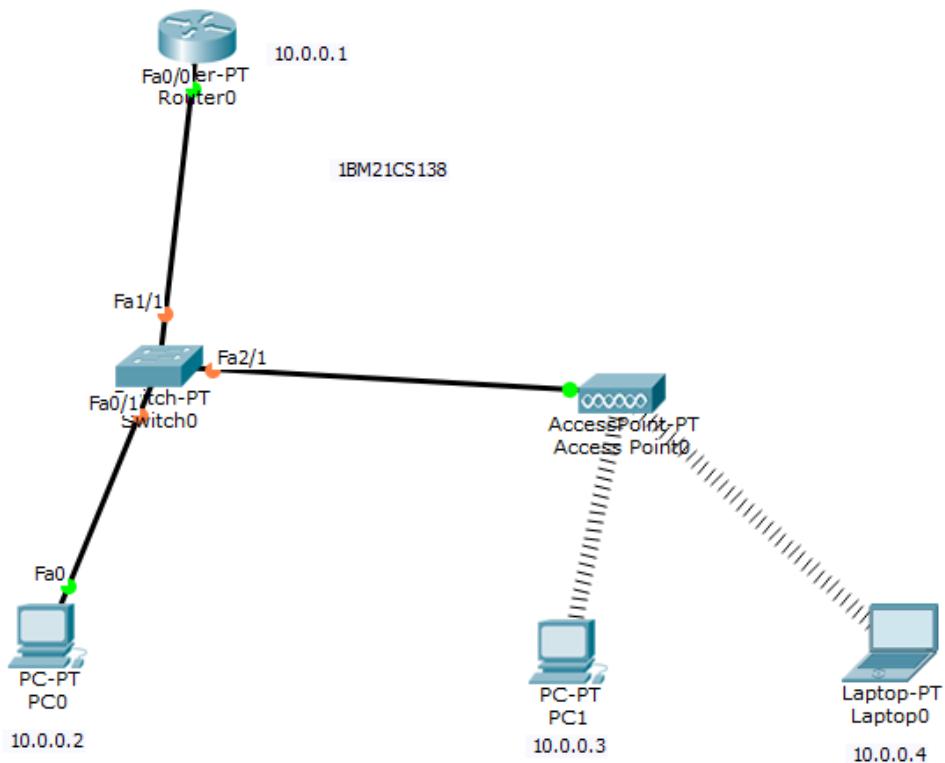
minimum = 6ms, maximum = 15ms, Average 10ms.

8

100% MET

292

Topology:



Output:

```
Command Prompt

Cisco Packet Tracer PC Command Line 1.0
C:\>PING 10.0.0.3

Pinging 10.0.0.3 with 32 bytes of data:

Request timed out.
Reply from 10.0.0.3: bytes=32 time=48ms TTL=128
Reply from 10.0.0.3: bytes=32 time=40ms TTL=128
Reply from 10.0.0.3: bytes=32 time=27ms TTL=128

Ping statistics for 10.0.0.3:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 27ms, Maximum = 48ms, Average = 38ms

C:\>|
```

Experiment 13

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

WAP for error detecting code using
CRC-CCITT

```
#include <stdio.h>
char m[50], g[50], & [50], q[50], temp[50];
void caltrans();
void crc();
void calc();
void calrem();
void shift();
int main()
{
    int n, i = 0;
    char ch, flag = 0;
    printf("Enter the frame like: ");
    while ((ch = getc(stdin)) != '\n')
        m[i + flag] = ch;
    n = i;
    for (i = 0; i < 16; i++)
        m[n + i] = '0';
    m[n] = '0';
    printf("Message after appending 16 zeros: %s", m);
    for (i = 0; i <= 16; i++)
        g[i] = '0';
    g[0] = g[1] = g[2] = g[3] = '1';
    g[17] = '0';
    printf("Generator: %s", g);
    printf("\n\n quotient: %s", q);
    caltrans();
    printf("\n transmitted frame: %s", q);
    printf("Enter transmitted frame: %s", q);
    scanf("%s", q);
```

```

printf("CRC checking\n");
crc_end;
printf("in last remainder is %s", &s);
for(i=0; i<16; i++)
    if(crc[i] == '0'):
        flag = 1;
    else
        continue;
    if(flag == 0)
        printf("Error during transmission");
    else
        printf("The Received frame is correct");
}

void correct()
{
    int i, j;
    for(i=0; i<n; i++)
        temp[i] = m[i];
    for(i=0; i<16; i++)
        temp[i] = m[i];
    for(i=0; i<n-16; i++)
    {
        if(crc[i] == '1')
        {
            q[i] = '1';
            (else);
        }
        else
        {
            q[i] = '0';
            shift(i);
        }
    }
}

```

```

 $\gamma[16] = m[i+17];$ 
 $\gamma[17] = '0';$ 
 $\text{for } (j=0; j < i+17; j++) \quad \text{(d1) } d1 = 0 - 17n$ 
 $\quad \text{temp}[j] = \gamma[j]; \quad \text{(d2) } d2 = 0 - 17n$ 
 $\}$ 
 $\gamma[n-16] = '0';$ 
 $}$ 
void catSame()
{
    int i;
    for (i=1; i < 16; i++)
         $\gamma[i-1] = ((\text{int}) \text{temp}[i] - 48) \wedge ((\text{int}) \gamma[i-1] + 48)$ 
}
void shiftL()
{
    int i;
    for (i=1; i < 16; i++)
         $\gamma[i-1] = \gamma[i];$ 
}
void catSame(int n)
{
    int i, k=0;
    for (i=n-16; i < n; i++)
         $m[i] = ((\text{int}) m[i] - 48) \wedge ((\text{int}) \gamma[k] + 48);$ 
     $m[i] = '0';$ 
}

```

OKP

Enter frame bits : 1011
message after appending 16 zeros : 1011000000000000

10110000000000000000

Generace: 160010000000100001

Quotient: 1011

transmitted frame: 1011011000101011

ever transmitted frame: 101110110001011010

~~Retained Name is correct.~~

Widened as on S.V. bank on 9/3/1918

WbT der W 63-3876 reihig schwach
verzweigt

46

Output:

```
Enter the frame bits:1011
Message after appending 16 zeros:1011000000000000000000
generator:1000100000100001
```

```
quotient:1011
transmitted frame:10111011000101101011
Enter transmitted freme:10111011000101101011
CRC checking
```

```
last remainder:0000000000000000
```

```
Received freme is correct|
```

Experiment 14

Write a program for congestion control using Leaky bucket algorithm.

```
MAP for congestion control using
Leaky Bucket algorithm is as follows:
#include <stdio.h> & <limits.h> <math.h>
int main()
{
    int incoming, outgoing, buck-size, n, store=0;
    printf("Enter bucket size, outgoing packets of
inputs: ");
    scanf("%d %d", &buck-size, &outgoing);
    while (n!=0)
    {
        printf("Enter the incoming Packet size: ");
        scanf("%d", &incoming);
        if (incoming <= (buck-size - store))
        {
            store += incoming;
            printf("Bucket buffer size %d out of %d",
                store, buck-size);
        }
        else
        {
            printf("Dropped %d no of Packets\n",
                incoming - (buck-size - store));
            printf("Bucket buffer size %d out of %d",
                store, buck-size);
            store = buck-size;
        }
        store -= outgoing;
    }
}
```

Start ("After" Outgoing 7. d. Packets left
 out at 7. d. in buffer size, store, buck-size;
 n--:
 }
 3 ~~1818~~
 0 1 P
 enter bucket size, outgoing rate and no of
 inputs: 20 10 2
 Enter the incoming packet size: 30
 Incoming Packet Size: 30
 Dropped 10, no of packets
 Bucket buffer size 0 out at 20
 After outgoing 10 packets left out at 20 in buffer
 enter the incoming packet size: 10
 Incoming Packet Size 10
 Bucket buffer size 20 out at 20
 After outgoing 10 packets left out at 20 in
 buffer

This is Leaky bucket algo used to regulate network traffic. It has input for burst size + outgoing rate, no. of input for each packet.

1. Init at the Packet Size.
 2. If there's enough space in the bubble, add the Packet size to it.
 3. If not, clear the excess packets and set the buffer to full.
 4. Simulate outgoing packets by subtracting outgoing take from bubble.
 5. Repeat for all packets.

Output:

```
Enter bucket size, outgoing rate and no of inputs: 10 10 2
Enter the incoming packet size : 30
Incoming packet size 30
Dropped 20 no of packets
Bucket buffer size 0 out of 10
After outgoing 0 packets left out of 10 in buffer
Enter the incoming packet size : 10
Incoming packet size 10
Bucket buffer size 10 out of 10
After outgoing 0 packets left out of 10 in buffer
```

Experiment 15

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

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.

ClientTCP.py

```
from socket import *
ServerName = '127.0.0.1'
ServerPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((ServerName, ServerPort))
sentence = input("Enter file name: ")
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print('From Server: ' + 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 True:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    file = open(sentence, "r")
    l = file.read(1024)
    connectionSocket.send(l)
    file.close()
    connectionSocket.close()
```

connectionSocket. send(1. encode(s))
Point ("in sent contents of p" + sentence)
file.close()

connectionSocket.close()

Output :-

Run the ServerTCP.py

The server is ready to receive

Run the ClientTCP.py

Enter file name: Server TCP.py

From Server:

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
```

The content of Server file is shown.

Output:

The screenshot displays three windows related to a socket application:

- servertop.py**: A Python script running in a terminal window. It binds to port 12000 and listens for connections. When a connection is made, it reads a file named 'sentence' from the client and prints its contents.
- clienttop.py**: Another Python script running in a terminal window. It connects to the server at port 12000 and sends the file 'sentence' to the server.
- DLL Shell 3.10.8**: A terminal window showing the command-line interface of the DLL Shell. It lists available modules and provides options for file operations like 'Send' and 'Recv'.

The terminal output shows the server receiving the file 'sentence' from the client and printing its contents.

```
servertop.py - C:/Users/Admin/AppData/Local/Programs/Python/Python310/servertop.py [1.1.1.1]
File Edit Format Run Options Window Help
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 ("Sent contents of " + sentence)
    file.close()
    connectionSocket.close()

DLL Shell 3.10.8*
File Edit Shell Debug Options Window Help
Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2023, 14:50:30) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python310/servertop.py
The server is ready to receive
Sent contents of servertop.py
The server is ready to receive

clienttop.py - C:/Users/Admin/AppData/Local/Programs/Python/Python310/clienttop.py [1.1.1.1]
File Edit Format Run Options Window Help
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()

DLL Shell 3.10.8
File Edit Shell Debug Options Window Help
Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2023, 14:50:30) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
Erase file name: servertop.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 ("Sent contents of " + sentence)
    file.close()
    connectionSocket.close()

>>>
```

Experiment 16

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

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

Client UDP. Py

```
from socket import *
ServerName = "127.0.0.1"
ServerPort = 12000
ClientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("Enter file name: ")
ClientSocket.sendto(sentence.encode("utf-8"), (ServerName, ServerPort))
filecontents, ServerAddress = ClientSocket.recvfrom(4096)
print("In Reply from Server: ", filecontents.decode("utf-8"))
for i in filecontents:
    print(str(i), end=" ")
ClientSocket.close()
ClientSocket.close()
```

Server UDP. 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(4096)
    sentence = sentence.decode("utf-8")
    file = open(sentence, "r")
    content = file.read(2048)
    ServerSocket.sendto(content.encode("utf-8"), ClientAddress)
```

```
serverSocket.sendto(4bytes,(com,"utf-8"),  
                    (clientAddress))  
print('in sent contents of',end='')  
print(sentence)  
file.close()  
output =
```

run Server UDP.py
The server is ready to receive
run Client UDP.py
Enter file name: Server UDP.py
Reply from server:
~~contents of Server UDP.py was as below:~~

8/19

Output:

The screenshot shows three windows related to a UDP server-client application:

- serverudp.py - C:/Users/Admin/AppData/Local/Programs/Python/Python310/serverudp.py ...**: The top-left window displays the Python source code for the server. It creates a socket on port 12000, binds it to '127.0.0.1', and enters a loop to receive messages from clients. It then reads the contents of a file named 'file' (which contains 'Hello World') and sends it back to the client.
- IDLE Shell 3.10.0**: The bottom-left window shows the Python shell output. It starts with the Python version and license information. Then, it shows the command to restart the server: `= RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python310/serverudp.py`. Finally, it prints the message "The server is ready to receive".
- IDLE Shell 3.10.0**: The rightmost window shows the Python shell output. It starts with the Python version and license information. Then, it shows the command to restart the client: `= RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python310/clientudp.py`. It then prompts for a file name: "Enter file name: serverudp.py". After pressing Enter, it prints "Reply from Server:" followed by the received message: "Hello World".

Experiment 17

Tool Exploration - Wireshark

Tool Exploration - Wireshark

Wireshark

Wireshark is an open source packet analyser which is used for educational analysis.

Software development, communication protocol development and network troubleshooting. It is used to view packets to that each one is filtered to meet user's better needs.

It is commonly called as a sniffer network protocol analyser, network analyzer.

It is also used by network security engineer to examine security problems.

Wireshark is a free application used to analyse data packets. It is also known as a free packet sniffer computer app. This programme can also analyse media file to extract all packets until it receives.

- It is used by network security engineers to trouble shoot network usage.

- It is also used to analyse undesired packets.

- It helps to troubleshoot many malfunctions on the network.

- It helps to monitor how all devices like laptop, mobile phones, desktops, servers, communicate and their behaviour or how they work.

- It helps to analyse the wireless communication and their behaviour or how they work.

Properties of Wireshark:

- It is similar to a TCP Dump in networking
- It is a graphic and fast and filtering further
- It also monitors the broadcast traffic with the Port
- Net sent to networks MAC address interface.
- Port monitoring is a method to monitor the broadcast traffic. When it is enabled switch sends copies of all network packets present at one port to another port.

Features of Wireshark:

- It is a multi-platform software i.e. it can run on Linux, Windows, OSX, True BSD, NetBSD etc.
- It is a standard three pane packet viewer.
- It performs deep inspection of many protocols.
- It has standard filtering option which enables user to search through the data.
- It can capture some USB traffic.
- It is useful in IP analysis.