

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT

on

COMPUTER NETWORKS

Submitted by

RAHUL C SHIRUR (1BM21CS157)

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

JUN-2023 to SEP-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 **RAHUL C SHIRUR (1BM21CS157)** , who is a bonafide student of **B. M. S. College of Engineering**. It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 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.

Dr. Nandini Vineeth
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

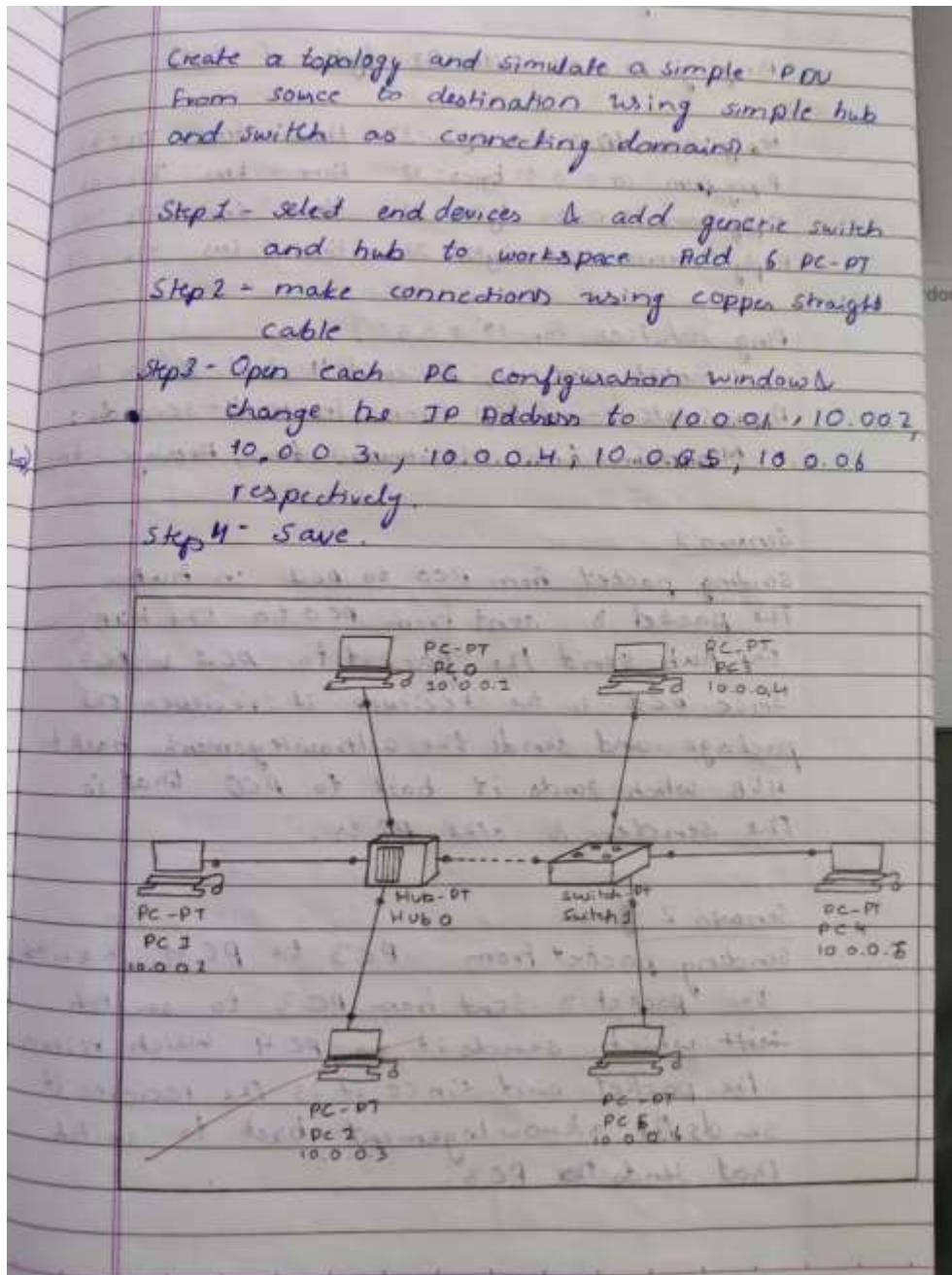
Index

Sl. No.	Date	Experiment Title	Page No.
		CYCLE 1	
1	16/6/23	Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrating ping messages.	4
2	26/6/23	Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.	8
3	17/7/23	Configure default route, static route to the Router.	12
4	17/7/23	Configure DHCP within a LAN and outside LAN.	16
5	25/7/23	Configure Web Server, DNS within a LAN.	21
6	25/7/23	Configure RIP routing Protocol in Routers.	24
7	1/8/23	Configure OSPF routing protocol.	28
8	1/8/23	To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP).	34
9	1/8/23	To construct a VLAN and make a pc communicate among VLAN.	36
10	18/8/23	Demonstrate the TTL/ Life of a Packet.	40
11	18/8/23	To construct a WLAN and make the nodes communicate wirelessly.	45
12	26/8/23	To understand the operation of TELNET by accessing the router in server room from a PC in IT office.	49
		CYCLE 2	
13	1/9/23	Write a program for error-detecting code using CRC CCITT (16-bits).	52
14	1/9/23	Write a program for congestion control using Leaky bucket algorithm.	59
15	1/9/23	Using TCP/IP sockets, write a client-server program to make the client sending the file name and the server to send back the contents of the requested file if present.	63
16	1/9/23	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.	68

WEEK 1

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

OBSERVATION:



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=6ms TTL=128

Reply from 10.0.0.5: bytes=32 time=6ms TTL=128

Reply from 10.0.0.5: bytes=32 time=6ms TTL=128

Reply from 10.0.0.5: bytes=32 time=6ms 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=6ms, Maximum=6ms, Average=6ms

Scenario 1

Sending packet from PC0 to PC1 in hub.

The packet is sent from PC0 to the HUB,

The hub send the packet to PC1 & PC2.

Since PC1 is the receiver it receives the package and sends the acknowledgement back to HUB which sends it back to PC0 that is the sender & other PC's.

Scenario 2

Sending packet from PC3 to PC4 in switch

The packet is sent from PC3 to switch

~~with~~ which sends it to PC4 which receives

the packet and since it is the receiver it sends the acknowledgement back to switch that sends to PC3.

Scenario 3

Sending b/w PC's connected to switch and HUB packet is sent from PC2 to PC4. The Hub sends the packet to PC0, PC1 and the switch. The switch sends the packet to PC3 & PC4 which is the receiver sends the acknowledgement back to switch which sends it back to the HUB which forwards it all to PC's connected to the HUB i.e. PC0, PC1, & PC2.

= 128

= 128

= 128

= 128

loss),

6ms

to

ch

es

to

ch

es

to

ch

es

to

ch

es

to

ch

es

to

ch

es

to

ch

es

to

ch

es

to

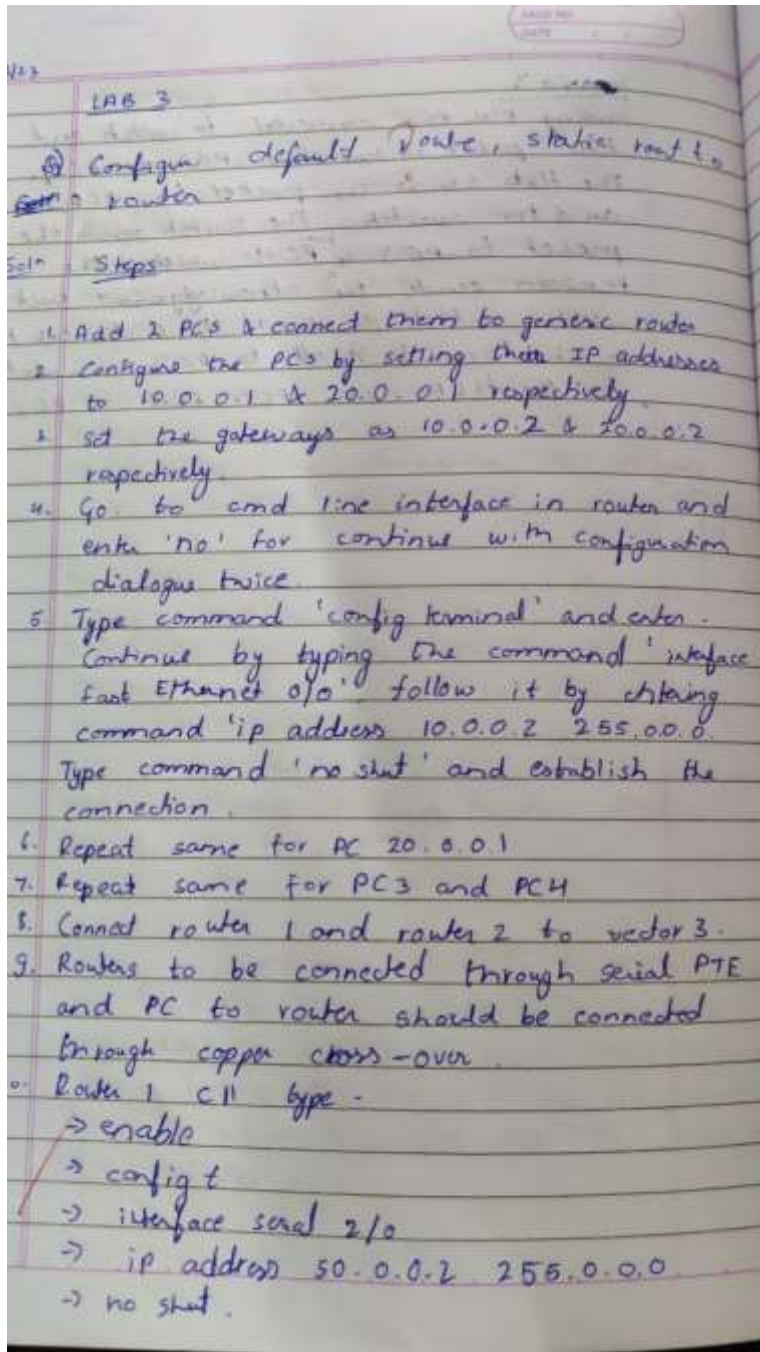
ch

[illegible]

WEEK 2

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

OBSERVATION:



1111 for 50.0.0.1 255.0.0.0

The ~~conn~~ connection b/w router 1 & 3 is now active.

11. Repeat step 10 for router 2 & 3
12. Go to router 1 CLI & type show ip route.
13. We now connect routers to each other by typing.

ip route 30.0.0.0 255.0.0.0 50.0.0.1

ip route 40.0.0.0 255.0.0.0 40.0.0.1 for router 1.

ip route 10.0.0.0 255.0.0.0 60.0.0.1

ip route 20.0.0.0 255.0.0.0 60.0.0.1 for router 2

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 for router 3

PC > ping 30.0.0.1

pinging 30.0.0.1 with 32 bytes of data:

Reply from 20.0.0.2 : destination host unreachable

Reply from 20.0.0.2 : destination host unreachable

Reply from 20.0.0.2 : destination host unreachable

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)

this is before statically connecting PC's.

PC > ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

Reply from 40.0.0.1: bytes=32 Time=7ms TTL=125

Reply from 40.0.0.1: bytes=32 Time=2ms TTL=125

Reply from 40.0.0.1: bytes=32 Time=2ms TTL=125

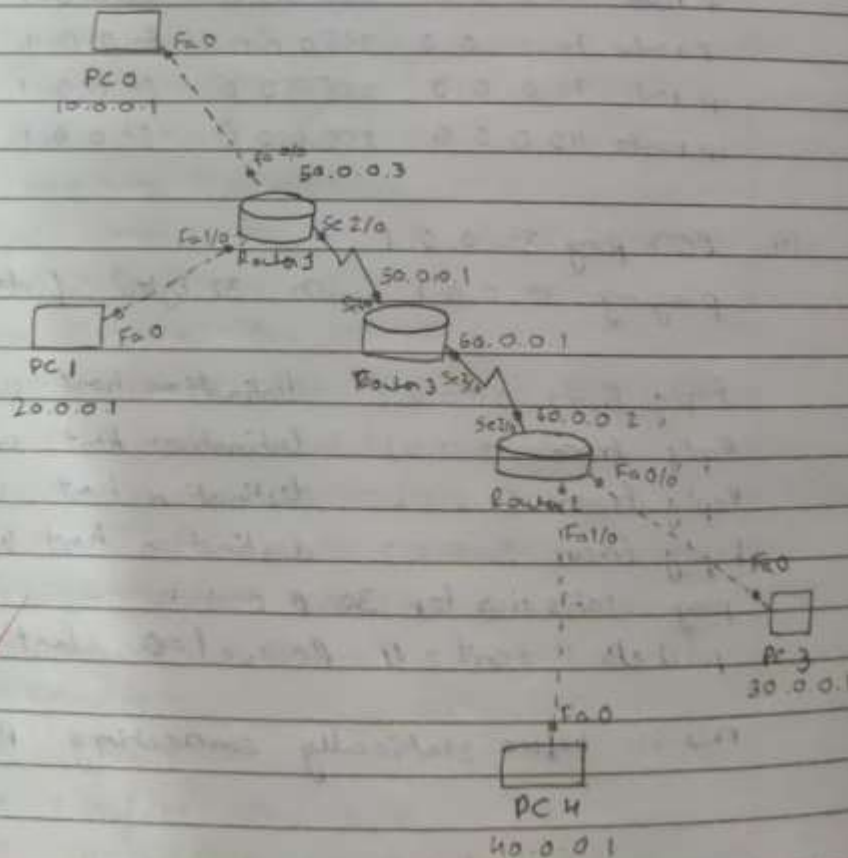
Reply from 40.0.0.1: bytes=32 Time=2ms TTL=125

Ping Statistics for 40.0.0.1:

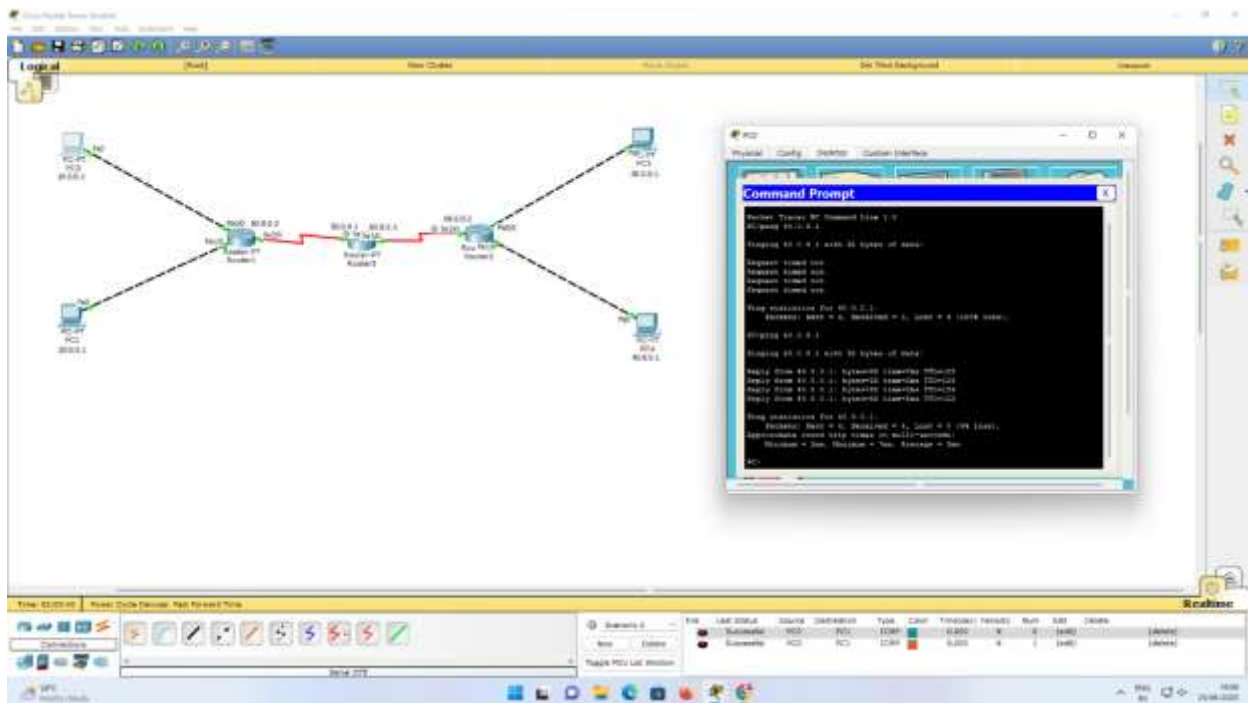
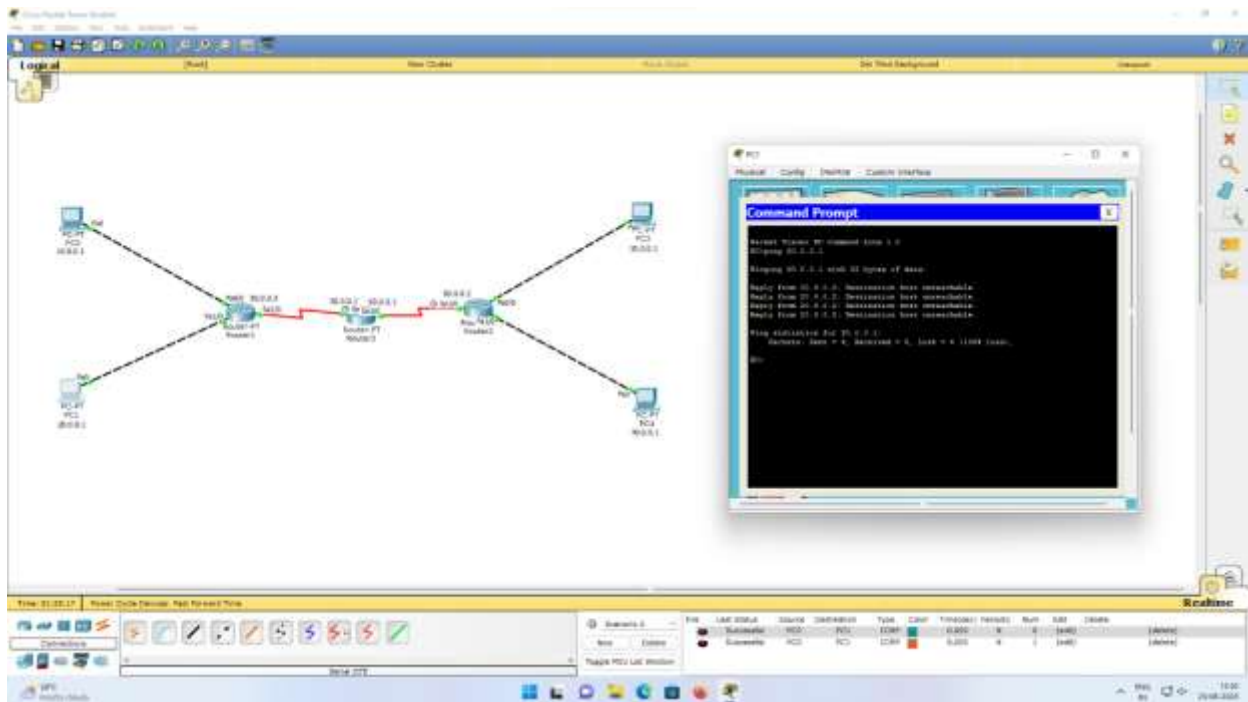
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)

Approximate round trip times in milli-seconds:

Minimum = 2ms, Maximum = 7ms, Average = 3ms



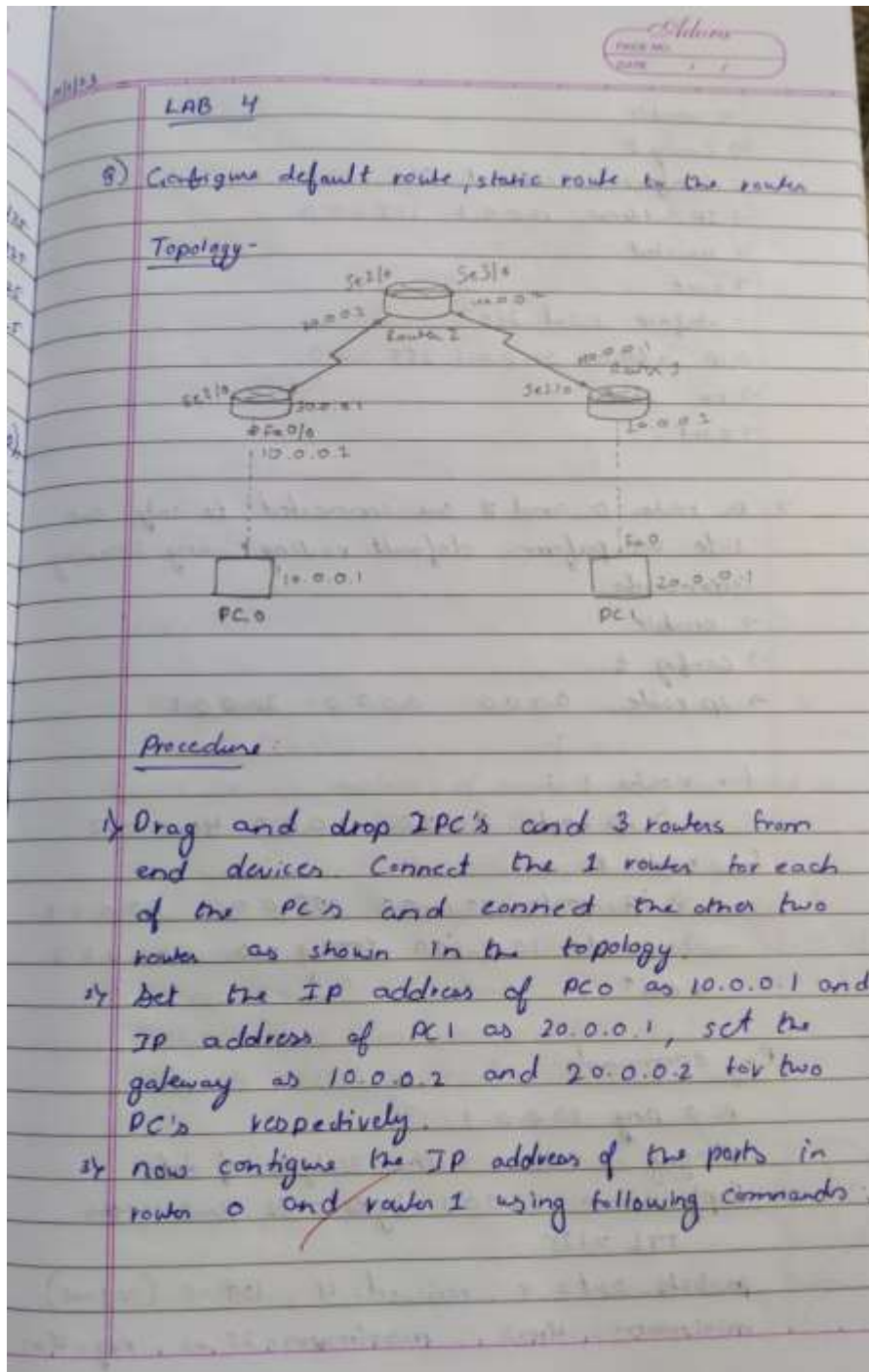
TOPOLOGY & OUTPUT:



WEEK 3

Configure default route, static route to the Router.

OBSERVATION:



→ enable
 → config t
 → interface fast ethernet 0/0
 → IP address 10.0.0.2 255.0.0.0
 → no shut
 → exit
 → interface serial 2/0
 → ip address 30.0.0.1 255.0.0.0
 → no shut
 → exit.

→ As router 0 and 1 are connected to only one side we perform default routing using following commands:

→ enable
 → config t
 → ip route 0.0.0.0 0.0.0.0 30.0.0.2

for router 1 :

> ip route 0.0.0.0 0.0.0.0 40.0.0.2

for router 2 :

→ ip route 10.0.0.0 255.0.0.0 50.0.0.2

→ ip route 20.0.0.0 255.0.0.0 40.0.0.2

→ exit

Ping commands

PC > ping 20.0.0.1

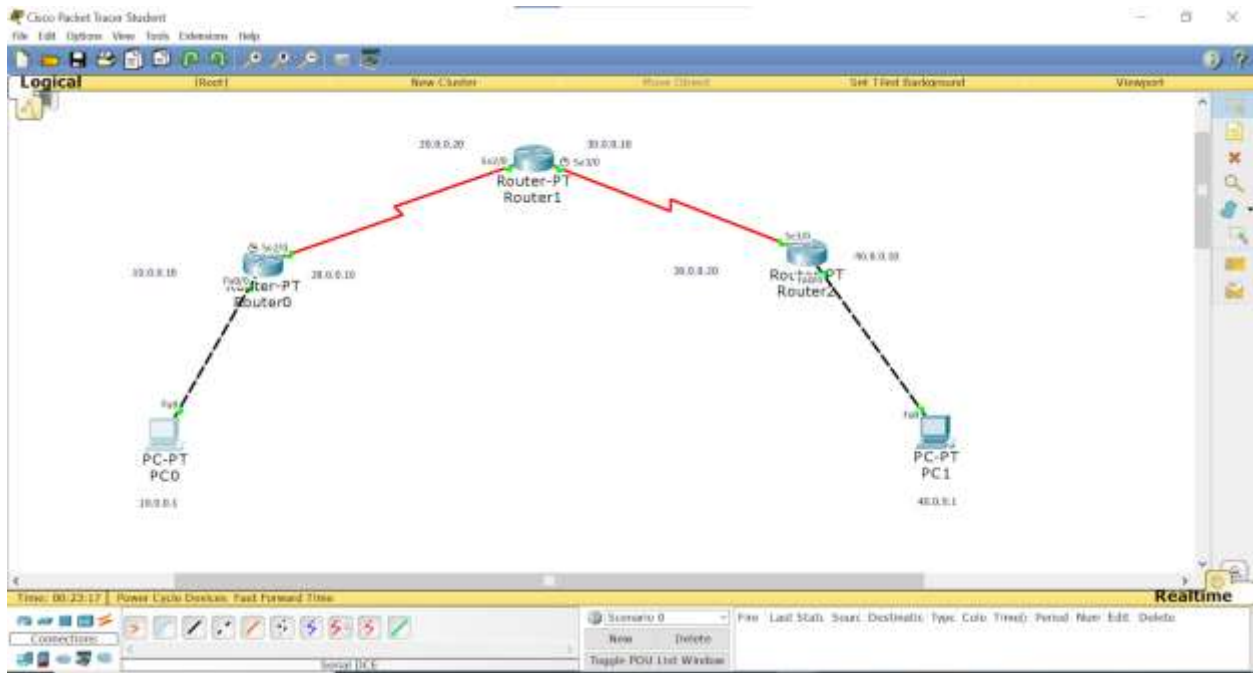
Pinging 20.0.0.1 with 32 bytes of data:

Reply from 20.0.0.1 bytes = 32 time = 4ms

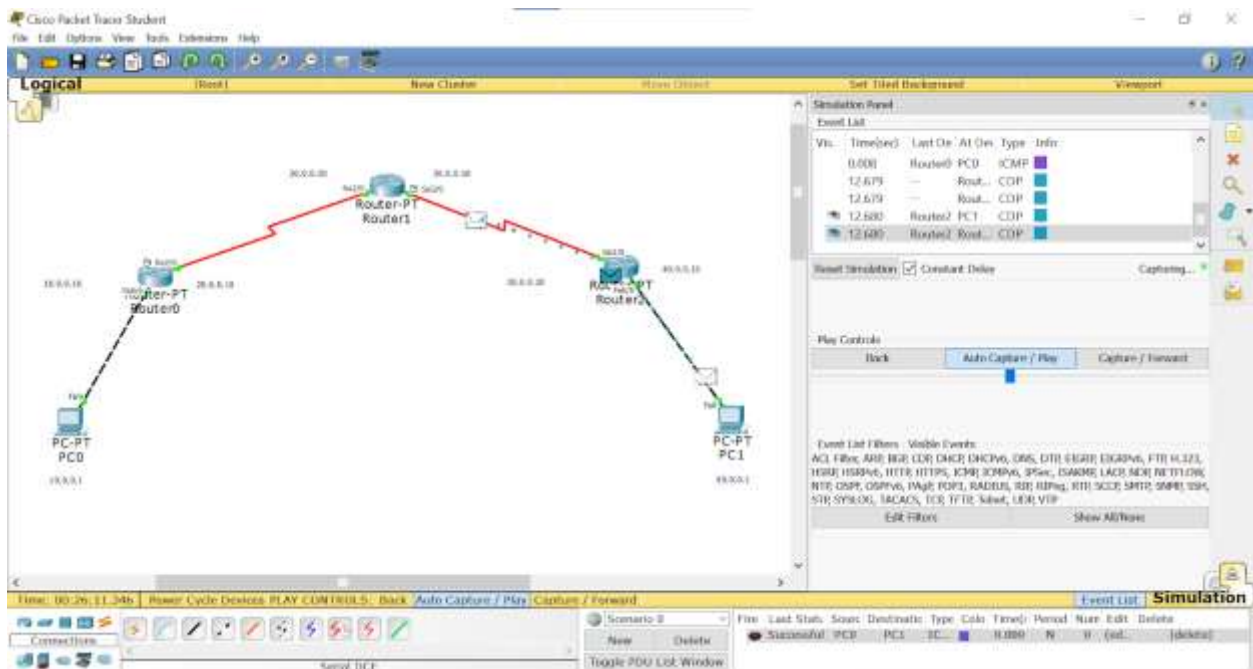
TTL = 125

packets sent = 4, received = 4, lost = 0 (0% loss)
 minimum = 4ms, maximum = 25 ms, Avg = 16ms

TOPOLOGY:



OUTPUT:



Command Prompt

```
Packet Tracer PC Command Line 1.0
PC>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=2ms TTL=125
Reply from 40.0.0.1: bytes=32 time=16ms TTL=125
Reply from 40.0.0.1: bytes=32 time=2ms TTL=125

Ping statistics for 40.0.0.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 16ms, Average = 6ms

PC>ping 40.0.0.1

Pinging 40.0.0.1 with 32 bytes of data:

Reply from 40.0.0.1: bytes=32 time=21ms TTL=125
Reply from 40.0.0.1: bytes=32 time=9ms TTL=125
Reply from 40.0.0.1: bytes=32 time=2ms TTL=125
Reply from 40.0.0.1: bytes=32 time=4ms TTL=125

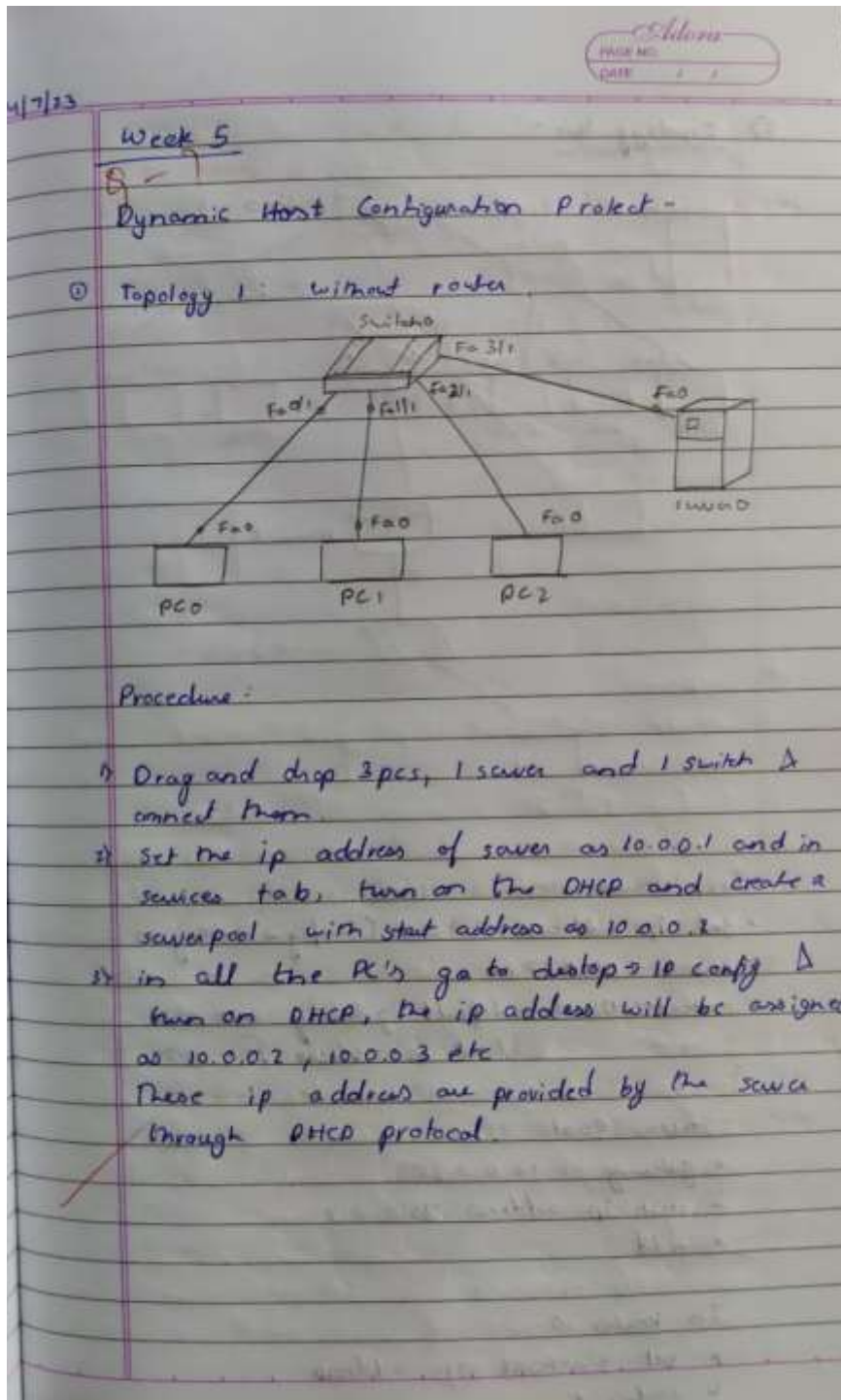
Ping statistics for 40.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 21ms, Average = 9ms

PC>|
```

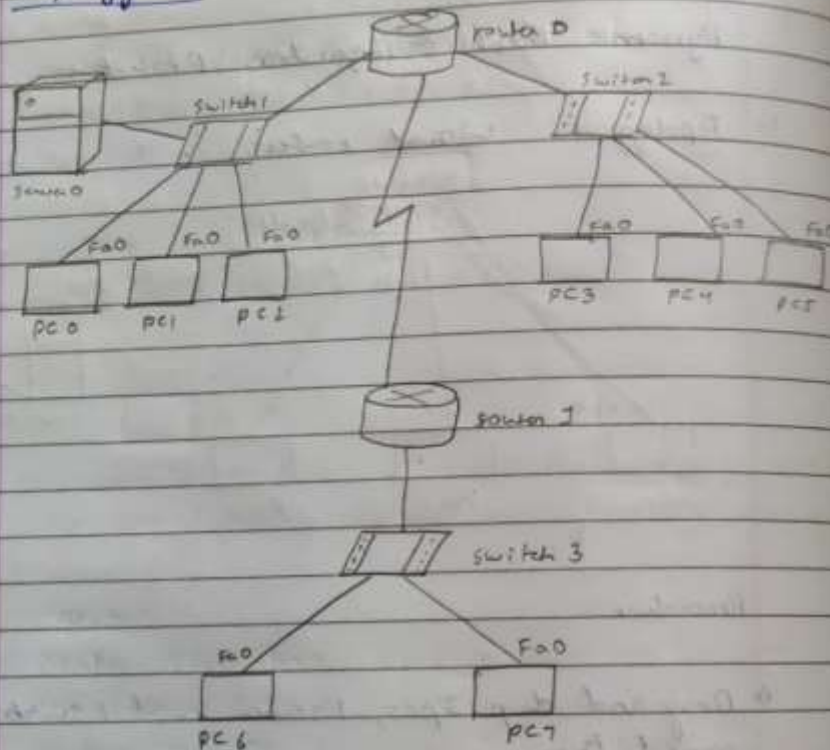
WEEK 4

Configure DHCP within a LAN and outside LAN.

OBSERVATION:



Topology two -



In server

- * set ip address 10.0.0.1 (config → setting → 10.0.0.25)
- * server → DHCP → on
- * server → DHCP → default gateway 10.0.0.25
start ip address 10.0.0.2

Server Pool

- * gateway → 10.0.0.25
- * start ip address 20.0.0.2
- * add

In router 0 -

- * set 2 network IP address
- * config t

* interface fastethernet 4/0
 * ip address 10.0.0.25 255.0.0.0
 * no shut
 (1111 20.0.0.25)
 * config t * interface fastethernet 0/0
 * ip helper-address 10.0.0.1
 * no shut
 → static route (for 40:0)
 config t
 ip route 40.0.0.0 255.0.0.0 30.0.0.0

In router 1

* set up address
 config t
 interface fastethernet 0/0
 ip address 40.0.0.26 255.0.0.0
 no shut
 * config t
 interface serial 2/0
 ip address 30.0.0.26 255.0.0.0
 no shut

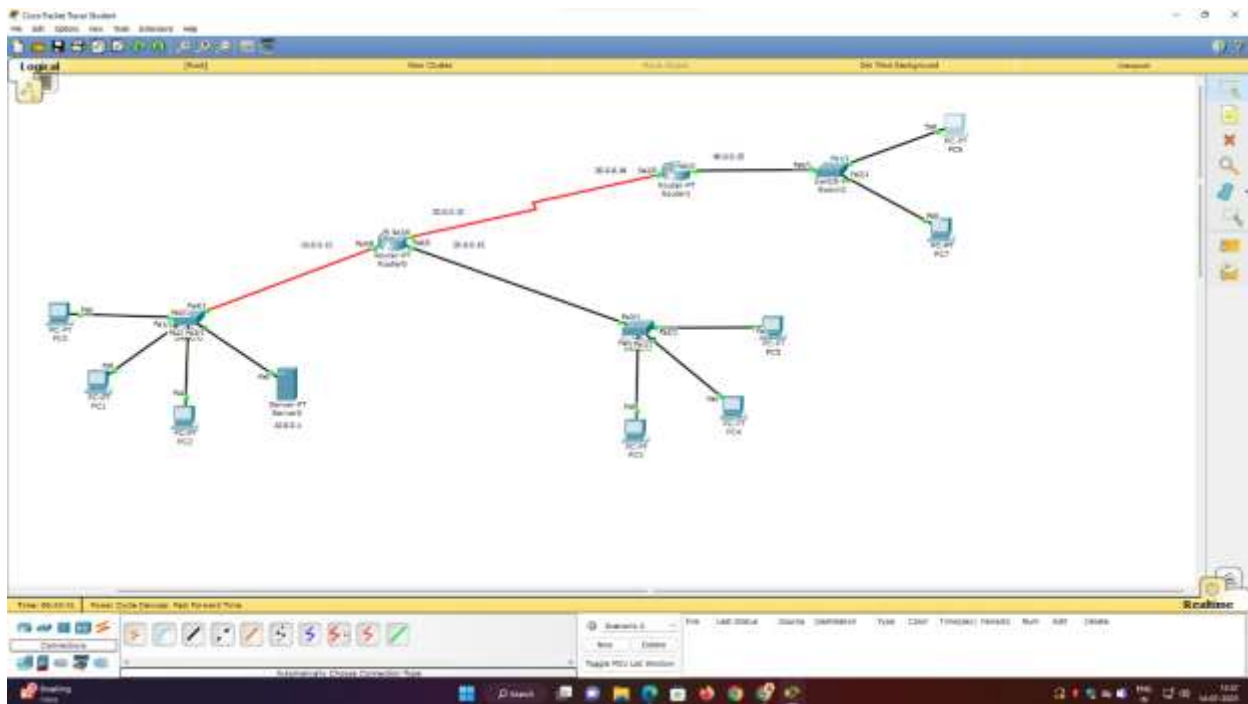
→ static routing for 10 & 20 network

config t
 ip route 10.0.0.0 255.0.0.0 30.0.0.25
 ip route 20.0.0.0 255.0.0.0 30.0.0.25
 no shut

→ setting helper address

config t
 interface fastethernet 0/0
 ip-helper-address (0.0.0.0)
 no shut.

TOPOLOGY:



OUTPUT:

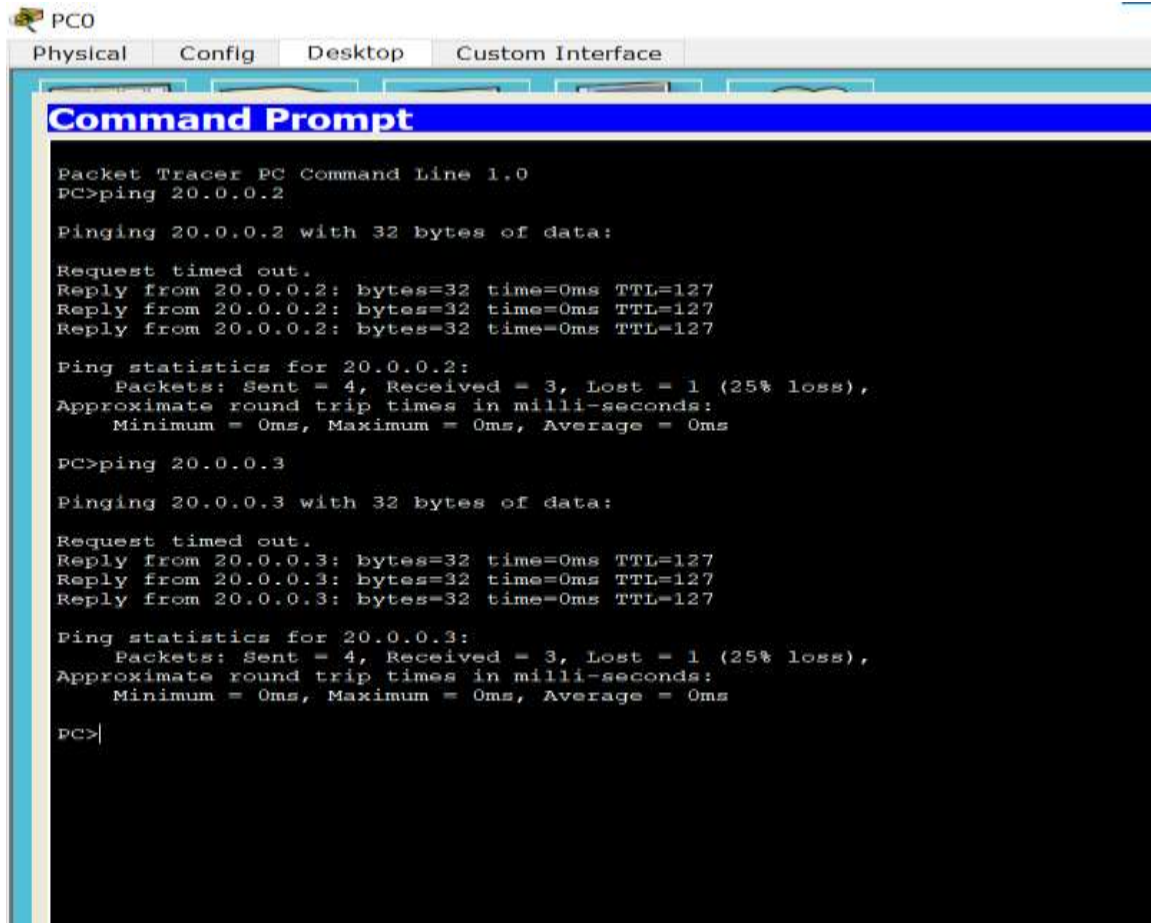
```
PC0
Physical Config Desktop Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.3

Pinging 10.0.0.3 with 32 bytes of data:

Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=0ms TTL=128
Reply from 10.0.0.3: bytes=32 time=1ms TTL=128
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 = 1ms, Average = 0ms

PC>
```



WEEK 5

Configure Web Server, DNS within a LAN.

OBSERVATION:

21/7/2023

Week-6

19) Create a topology (1 PC, 1 Switch, 1 Server)

```
graph TD
    PC0[PC-0  
10.0.0.10] --- Fa0_0[Fa0/0] --- Switch[Switch-0]
    Switch --- Fa0_1[Fa0/1] --- Server0[Server-0  
10.0.0.20]
```

* Steps-

1. Make the above topology & connect them as shown.
2. Config the PC & server IP addresses.
3. Turn the DNS service 'on' in the servers & add a website name with the address as IP address of the servers.
4. Go to HTTP services in server & edit the 'index.html' Page. Add relevant info some.
5. Go to web browser in the PC & enter the URL as the name of the website you saved in the server DNS.

> output :-

In web browser in PC we can now enter the name of server and we are shown the index.html page.

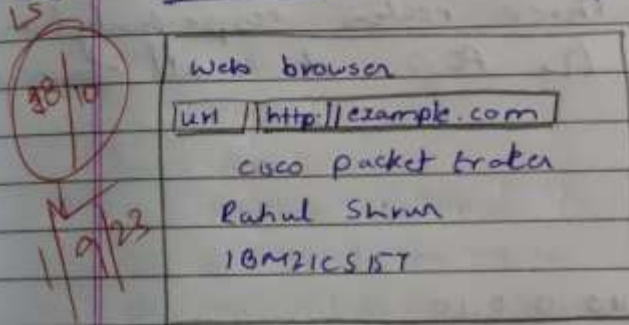
Rahul Shrivastava

IBM21CS157

Observation

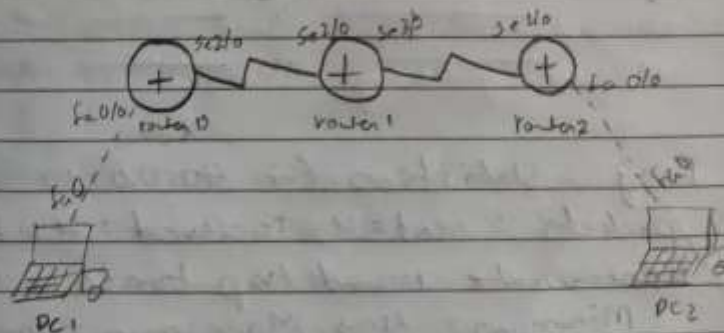
We can enter name of a website, instead of its IP address.

28) Output website



29) Configure RIP routing protocol in Routers

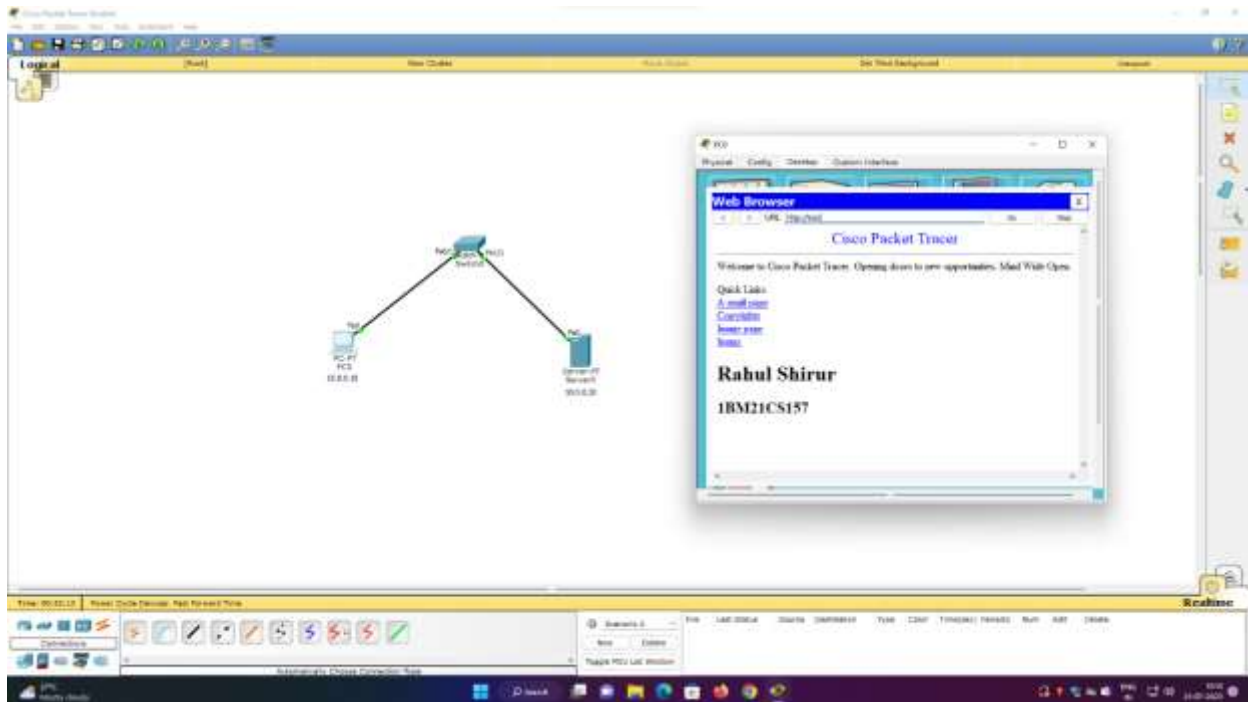
- Create topology -



Steps

1. Create topology as shown in the figure
2. Set the IP address for 2 PC's and three routers

TOPOLOGY & OUTPUT:



WEEK 6

Configure RIP routing Protocol in Routers.

OBSERVATION:

Adora
PAGE NO. _____
DATE / /

Rahul Shrivastava
IBM21CS157

Observation

we can enter name of a website, instead of its IP address.

28) Output website

LS
38/10
1/9/23

web browser
url //http://example.com
cisco packet tracer
Rahul Shrivastava
IBM21CS157

29) Configure RIP routing protocol in Routers.

- Create topology =

Steps

- Create topology as shown in the figure
- set the IP address for 2 PC's and three routers

3. Configure all the PC's and routers with each other using commands -
interface serial
ip address
encapsulation PPP
clock rate 64000
for all respectively (router 0, 1, 2)

4. Now # router rip
network 10.0.0.10
network 10.0.0.20
for all three routers respectively (router 0, 1, 2)
5. Now ping the PC's, it is all connected to.

* Output :

PC > ping 40.0.0.10

pinging 40.0.0.10 with 32 bytes of data:

reply from 40.0.0.10: bytes = 32

time = 4ms TTL = 125

reply from 40.0.0.10: bytes = 32 time = 4ms TTL = 125

" " " " " "

" " " " " "

Ping statistics for 40.0.0.10:

packets: sent = 4, received = 4, lost = 0 (0%)

Approximate round trip times in milliseconds:

Minimum = 4ms Maximum = 7ms Average = 5ms

* Commands:

① configure router (plan ip) -

```
Router > enable
Router# config t
# interface fastEthernet 0/0.
# ip address 10.0.0.3 255.0.0.0.
# no shut.
# ip interface serial 2/0
# ip address 20.0.0.1 255.0.0.0
# no shut.
```

② encapsulation

```
# interface serial 2/0
# encapsulation ppp.
# clock rate 64000
# no shut.
```

③ for ip

```
# router rip.
# network 10.0.0.0
# network 20.0.0.0
```

8/10

11/9/23

The screenshot displays a Cisco Packet Tracer simulation environment. On the left, a PC window is open, showing a Command Prompt with the following commands and outputs:

```

PC> ipconfig /all

Windows IP Configuration

Ethernet adapter Ethernet0:

   Media . . . . . : 10/100/1000 Mbps
   Link-local IPv6 Address . . . . . : fe80::5400:59ff:fe33:c0a9%1
   IPv4 Address. . . . . : 10.0.0.10
   Subnet Mask . . . . . : 255.255.255.0
   Default Gateway . . . . . : 10.0.0.1

Ethernet adapter Loopback0:

   Media . . . . . : 10/100/1000 Mbps
   Link-local IPv6 Address . . . . . : fe80::5400:59ff:fe33:c0a9%2
   IPv4 Address. . . . . : 127.0.0.1
   Subnet Mask . . . . . : 255.255.255.255
   Default Gateway . . . . . :

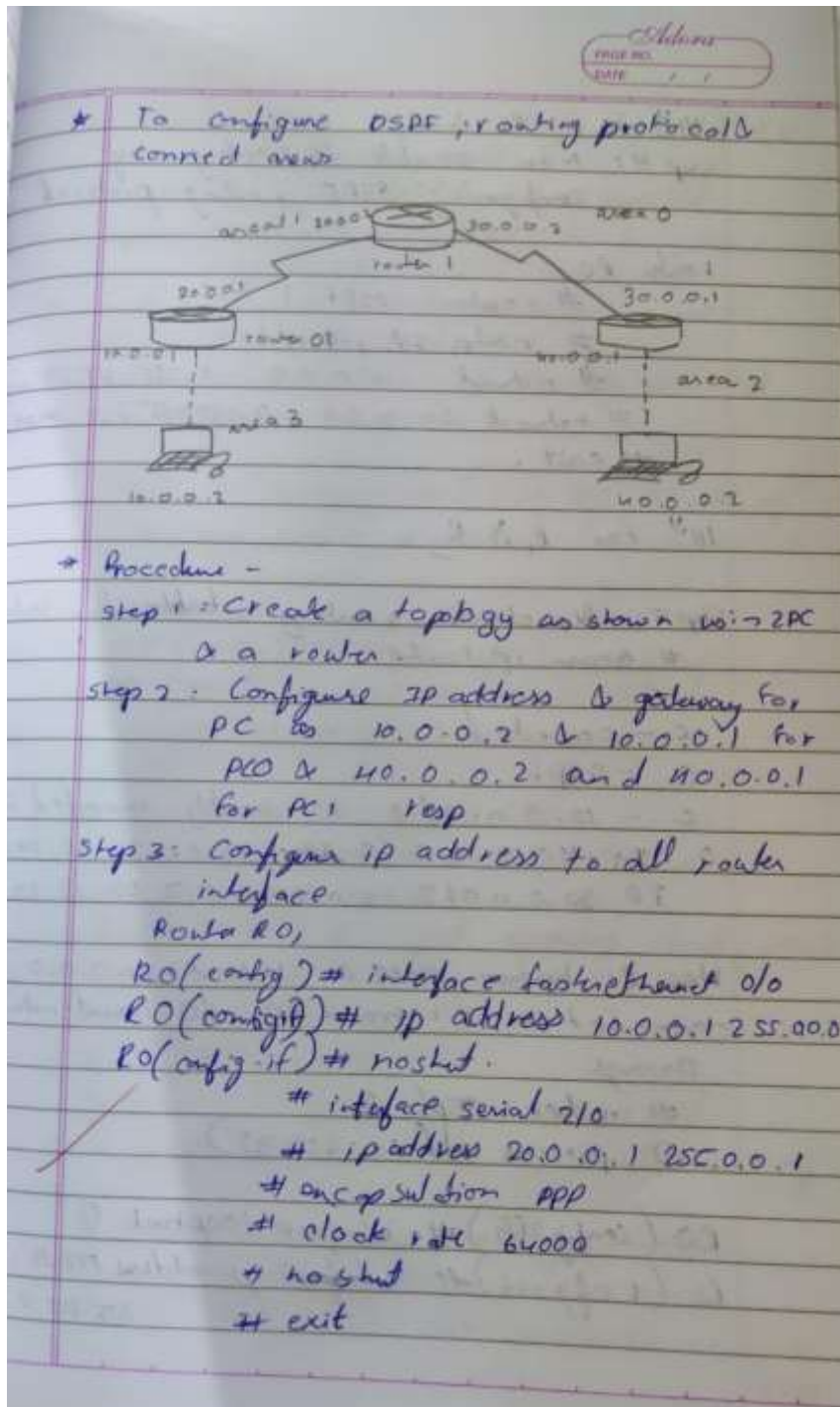
```

On the right, a network topology is visible. It consists of three routers (R1, R2, R3) connected in a line. R1 is connected to R2, and R2 is connected to R3. R1 is also connected to a PC (PC1) and a switch (S1). R3 is connected to a PC (PC2) and a switch (S2). The routers are labeled with their IP addresses and names. The switches are labeled with their IP addresses and names. The PCs are labeled with their IP addresses and names.

WEEK 7

Configure OSPF routing protocol.

OBSERVATION:



IIIrd for R₁ & R₂
Step 4: Now enable IP routing by
configuring OSPF routing protocol

Router R0,
router ospf 1
router id 1.1.1.1
network 10.0.0.0 0.255.255.255 area 0
network 20.0.0.0 0.255.255.255 area 1
exit.

IIIrd for R₁ & R₃

Step 5: Now check routing table for route
show ip route

C - connected
O - ospf
C - 10.0.0.0/8 is directly connected to R0
O - For 40.0.0.0/8 via 20.0.0.2, 00:04:23 h
IP 30.0.0.0/8 via 20.0.0.2, 00:07:29

Here R1 known area 0, Network 20.0.0.0
connected to R1 from R0, R0 learnt network
through

router ospf 1
=> process id (1 - 65535)

R0 (config-if) # interface loopback 0
R0 (config-if) # ~~interface~~ ip address 172.16.1.252
255.255.0.0

Step 6: Now check routing table for R3
R3 # show ip route

Codes: O - ospf C - connected
O 1.0 20.0.0.0/8 via 30.0.0.2, 00:18:58, Se2/0

C 40.0.0.0/8 directly connected, fast ethernet 0/0

C 30.0.0.0/8 is directly connected, Se2/0

Step 7 Create virtual link b/w R0, R1, by this we create a virtual link in area 3 to

In R0

R0 (config) # router ospf 1

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

In R1

R1 (config) # router ospf 1

R1 (config-router) # area 1 virtual link 1.1.1.1

Step 8: R1 & R2 get updates about area 3.
Now check routing table for R2.

R2 # show ip route

Codes: O - ospf C - connected

O 1.0 20.0.0.0/8 via 30.0.0.2 00:01:58 Se2/0

C 40.0.0.0/8 is directly connected, fast ethernet 0/0

O FA 10.0.0.0/8 via 30.0.0.2 00:01:58 Se2/0

Result-

In PC 0

PC > ping 40.0.0.2

pinging 40.0.0.2 with 32 bytes of data

Reply from 40.0.0.2: bytes=32 time=2ms

" " " " " " time=10ms

" " " " " " time=14ms

" " " " " " time=2ms

ping statistic for 40.0.0.2

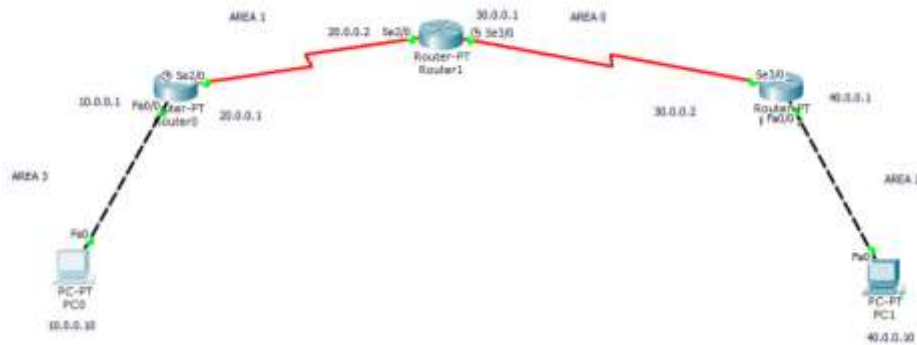
packets: sent: 4, received: 4, loss: 0

approx round trip times in ms

min = 2ms, max = 14ms, avg = 7ms

8/10
11/23
LS

TOPOLOGY:



OUTPUT:

```
PC0
Physical Config Desktop Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 40.0.0.10

Pinging 40.0.0.10 with 32 bytes of data:

Reply from 10.0.0.1: Destination host unreachable.
Reply from 10.0.0.1: Destination host unreachable.
Reply from 10.0.0.1: Destination host unreachable.
Reply from 10.0.0.1: Destination host unreachable.

Ping statistics for 40.0.0.10:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

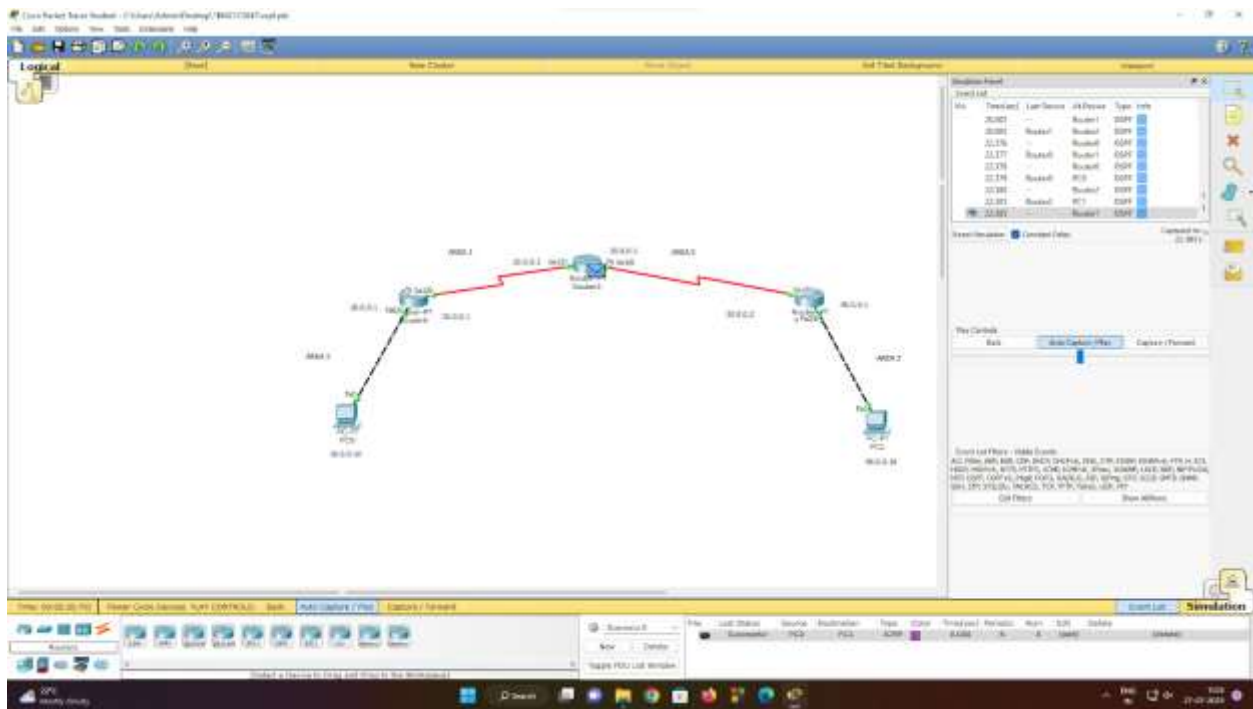
PC>ping 40.0.0.10

Pinging 40.0.0.10 with 32 bytes of data:

Request timed out.
Reply from 40.0.0.10: bytes=32 time=4ms TTL=125
Reply from 40.0.0.10: bytes=32 time=6ms TTL=125
Reply from 40.0.0.10: bytes=32 time=12ms TTL=125

Ping statistics for 40.0.0.10:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 12ms, Average = 7ms

PC>
```

WEEK 8

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

OBSERVATION:

An: To construct a lan and understand the concept and operation of address Resolution Protocol.

Topology:

```
graph TD
    Switch[Switch] --- PC0[PC0]
    Switch --- PC1[PC1]
    Switch --- PC2[PC2]
```

Configure IP -
PC0 (R0)
→ arp -a
→ ping - 10.0.0.2
→ arp -a
→ arp -d

Output:

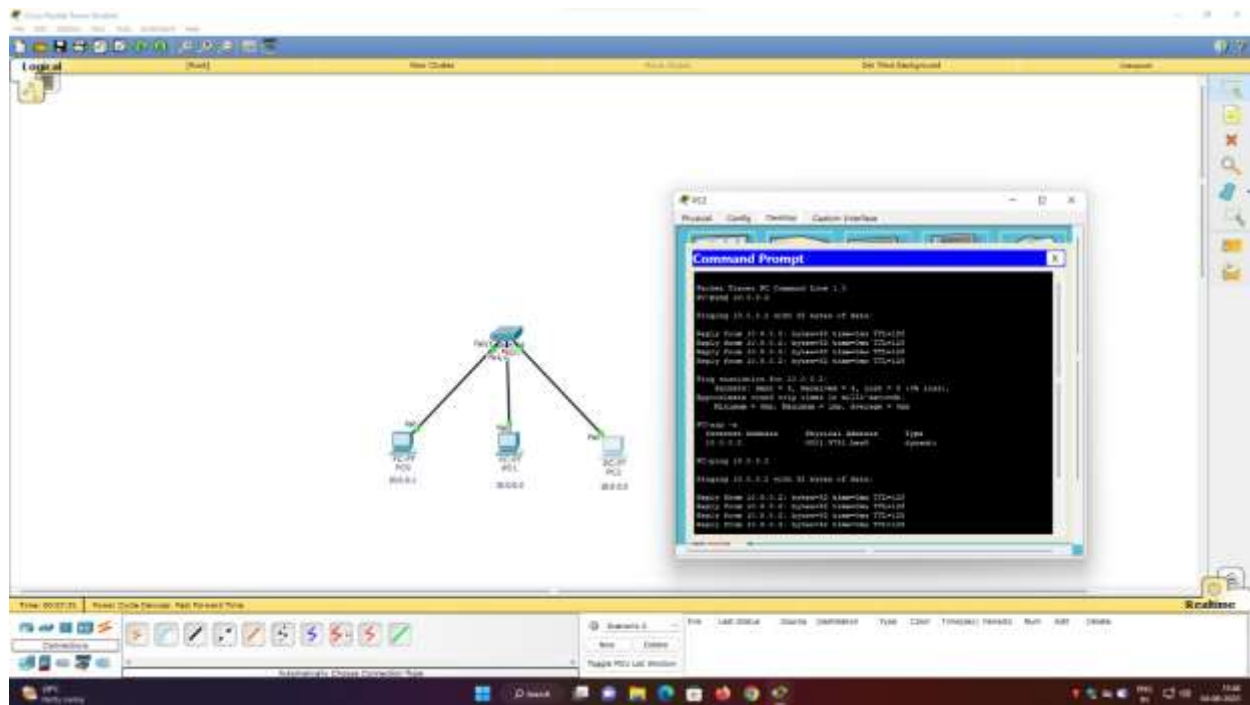
9/10 → arp -a
No ARP Entries found.

3 Ping 10.0.0.2
Reply from 10.0.0.2: bytes = 32 time = 0 ms TTL = 64

11	11	11
11	11	11

arp -a
Internet address physical address type
10.0.0.2 0002-1615-9820 Dynamic

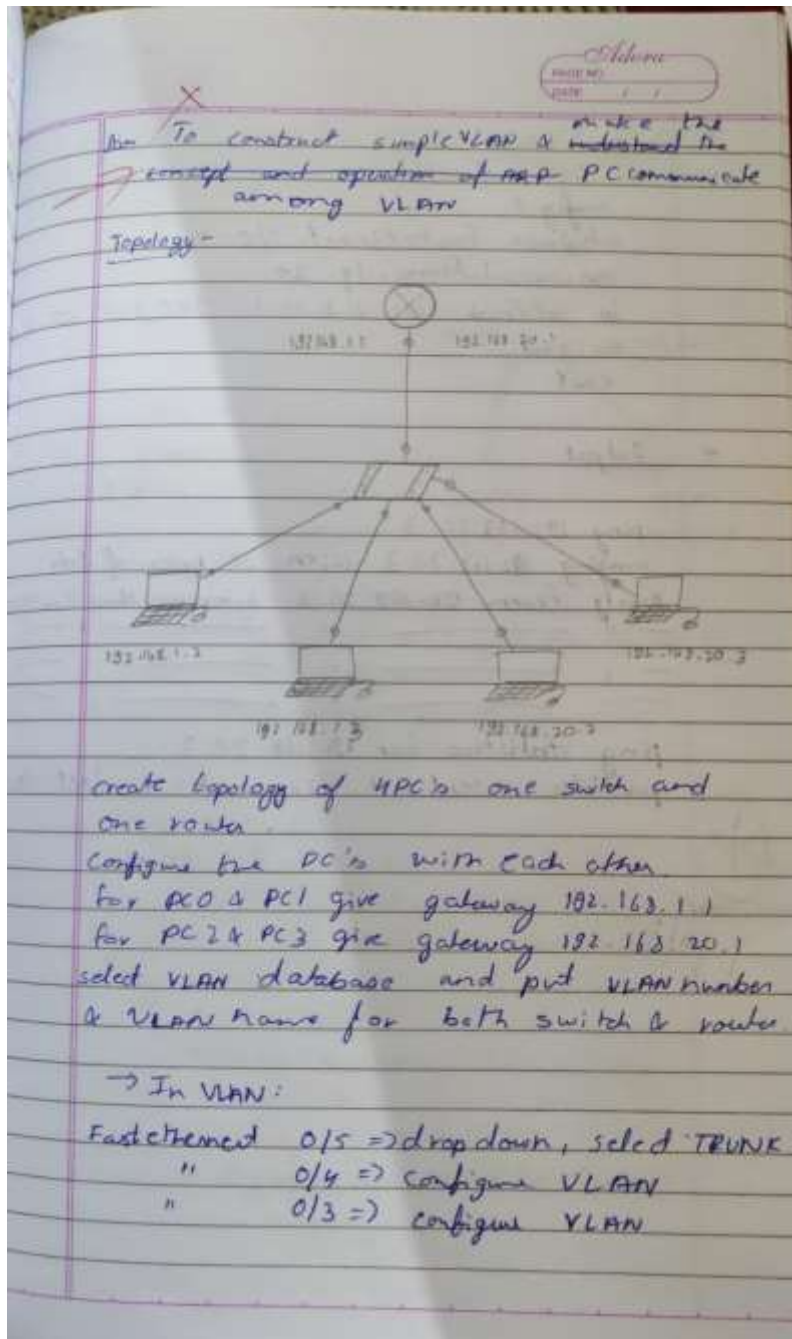
TOPOLOGY & OUTPUT:



WEEK 9

To construct a VLAN and make a pc communicate among VLAN.

OBSERVATION:



2. In Router

CLI of router

config t

interface fastethernet 0/0 1

encapsulation ip 20

ip address 192.168.20.1 255.255.255.0

no shut

exit

* Output

ping 192.168.20.3

pinging 192.168.20.3 with 32 bytes of data:

Reply from 192.168.20.3: bytes=32 time=0ms

" " " "

" " " "

" " " "

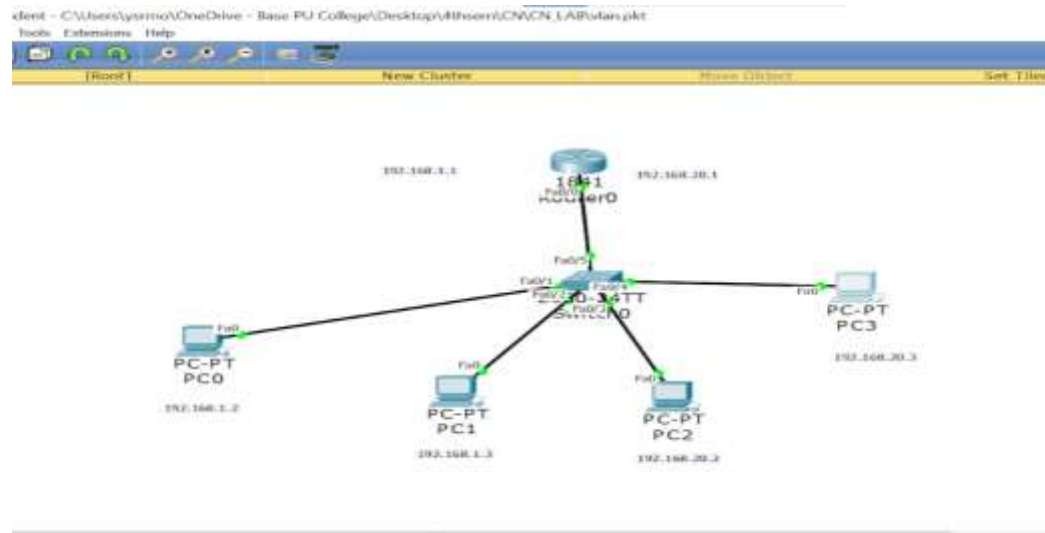
ping statistics for 192.168.20.3

packets: sent = 4, received = 4, lost = 0

10/10

1/9/23

TOPOLOGY:



OUTPUT:

PC0

Physical Config Desktop Custom Interface

Command Prompt X

```
Packet Tracer PC Command Line 1.0
PC>ping 192.168.20.3

Pinging 192.168.20.3 with 32 bytes of data:

Request timed out.
Reply from 192.168.20.3: bytes=32 time=0ms TTL=127
Reply from 192.168.20.3: bytes=32 time=5ms TTL=127
Reply from 192.168.20.3: bytes=32 time=0ms TTL=127

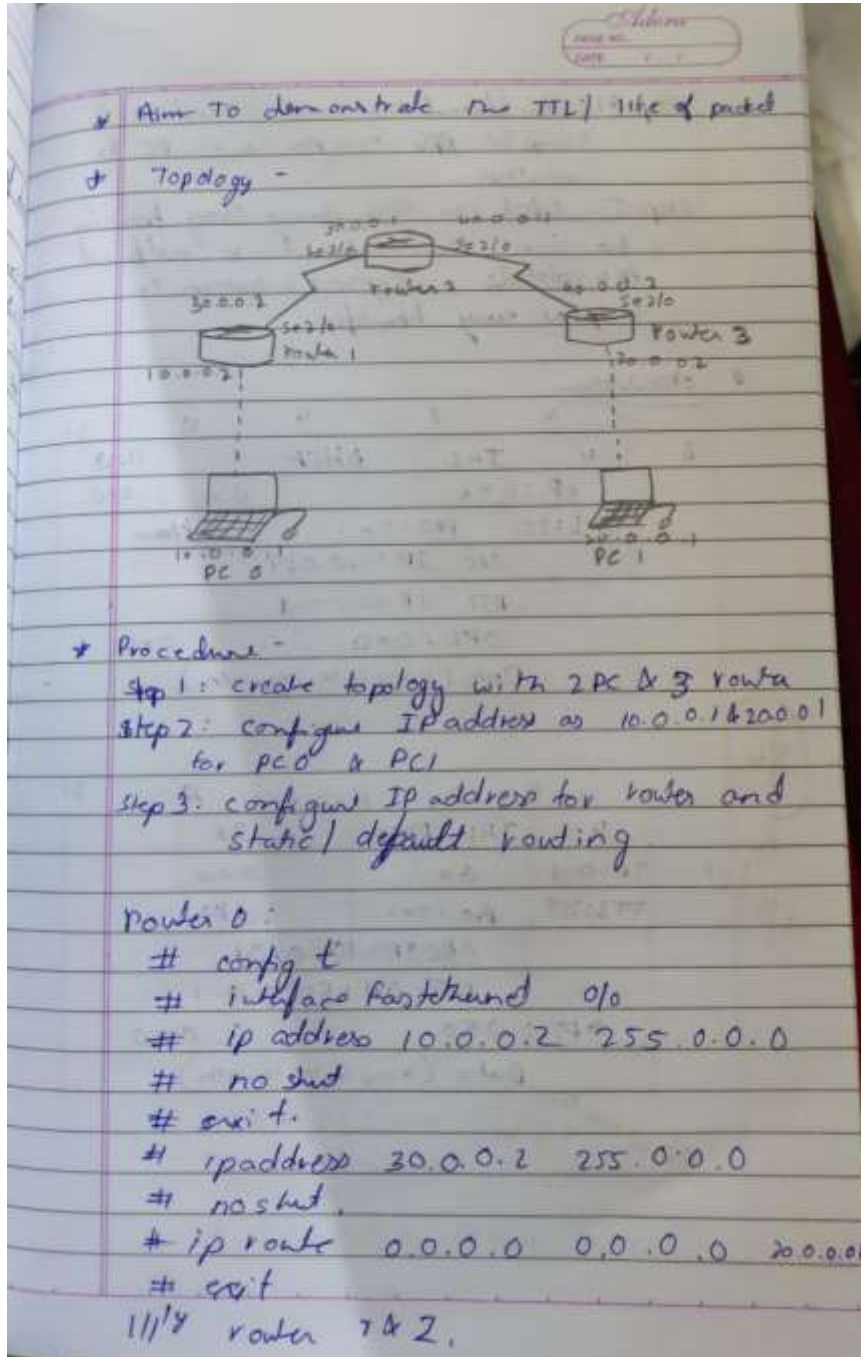
Ping statistics for 192.168.20.3:
    Packets: Sent = 4, Received = 3, Lost = 1 (25%
    loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 5ms, Average = 1ms

PC>
```


WEEK 10

Demonstrate the TTL/ Life of a Packet.

OBSERVATION:



Step 11: In stimulation mode, send a simple PDU from one PC to another.

Step 5: Click on PDU during every transfer to see the inbound & outbound PDU details, capture button to capture every transfer.

8 observation

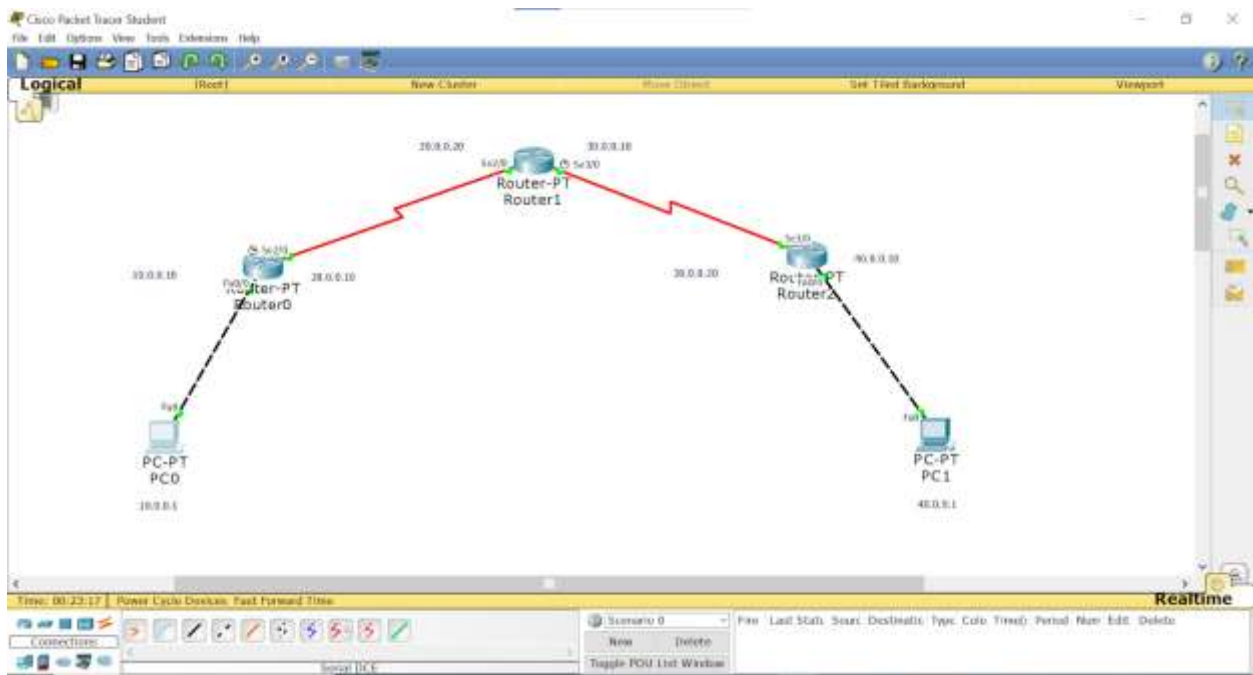
Observation					
0	4	8	16	19	31
0	4	IHL	DHCP		11:28
		IP: 0x6		0x	0x0
		TTL: 255	PRO: 0x1	checksum	
		SRC IP 10.0.0.1			
		DST IP 50.0.0.1			
		OPT: 0xD			0x0
		Data (variable length)			

8:00
N
1/9/23

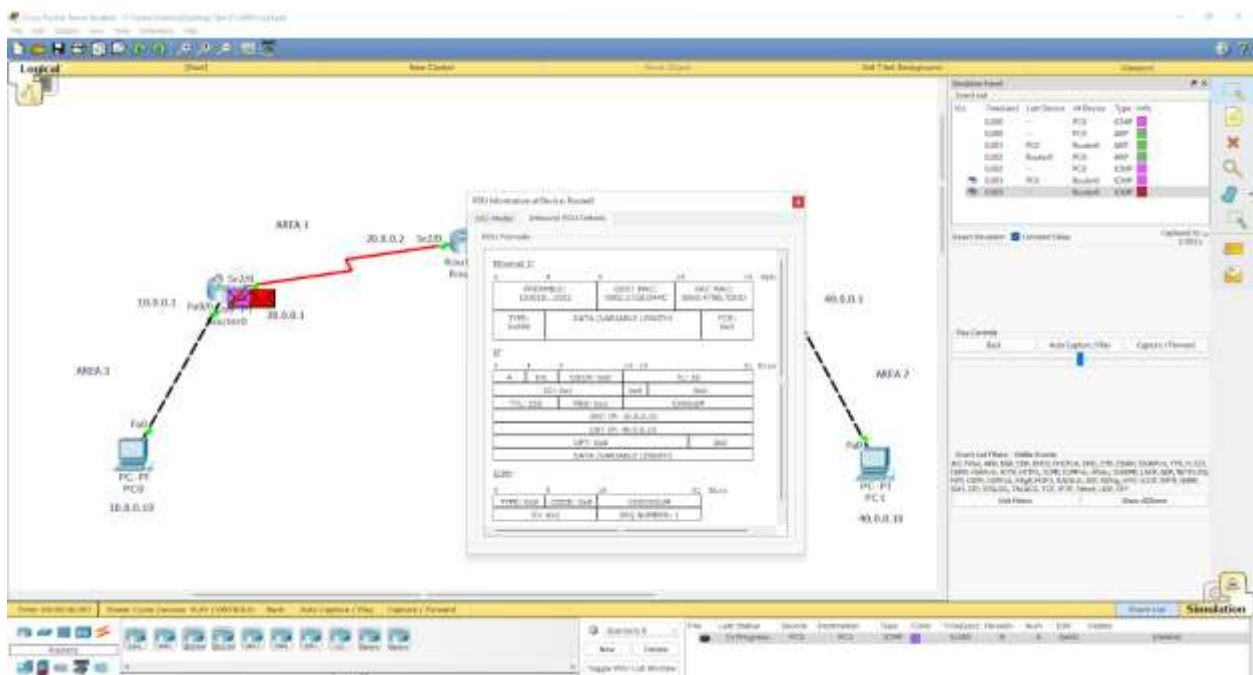
0	4	8	16	19	31
4	IHL	DHCP	TTL: 25		
IP: 0x6		0x1		0x0	
TTL: 255		PRO: 0x1		CHK	
		SRC IP: 10.0.0.1			
		DST IP: 50.0.0.1			
		OPT: 0x0			0x0
		Data (variable length)			

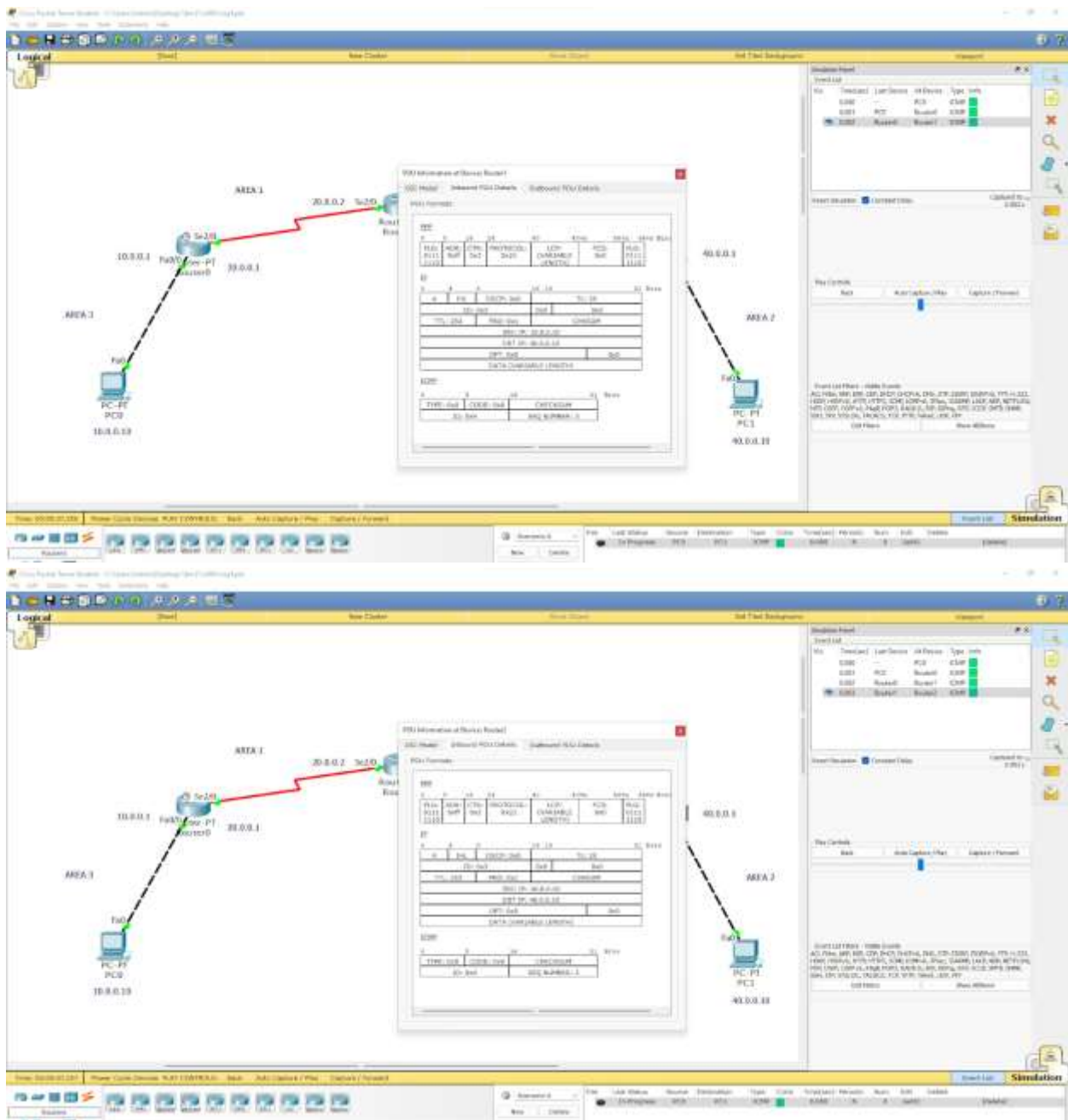
Observation ?

TOPOLOGY:



OUTPUT:





WEEK 11

To construct a WLAN and make the nodes communicate wirelessly

OBSERVATION:

WLAN

★ Aim - Construct a WLAN & make the nodes communicate wirelessly.

Procedure -

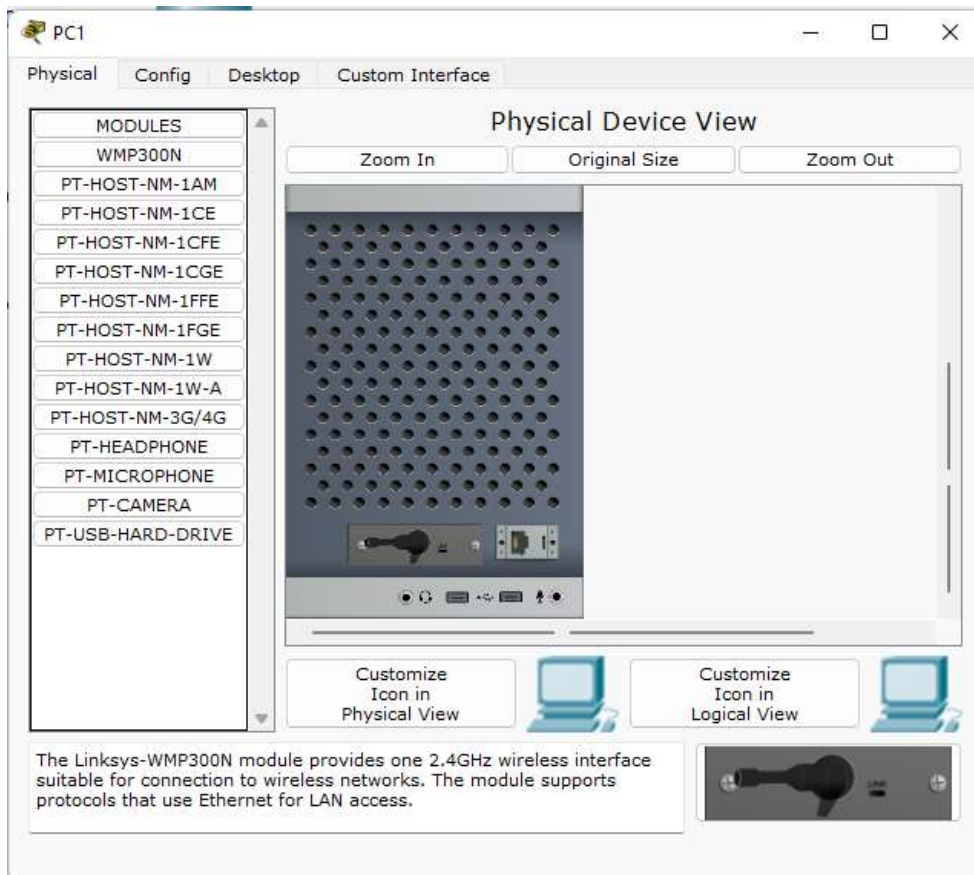
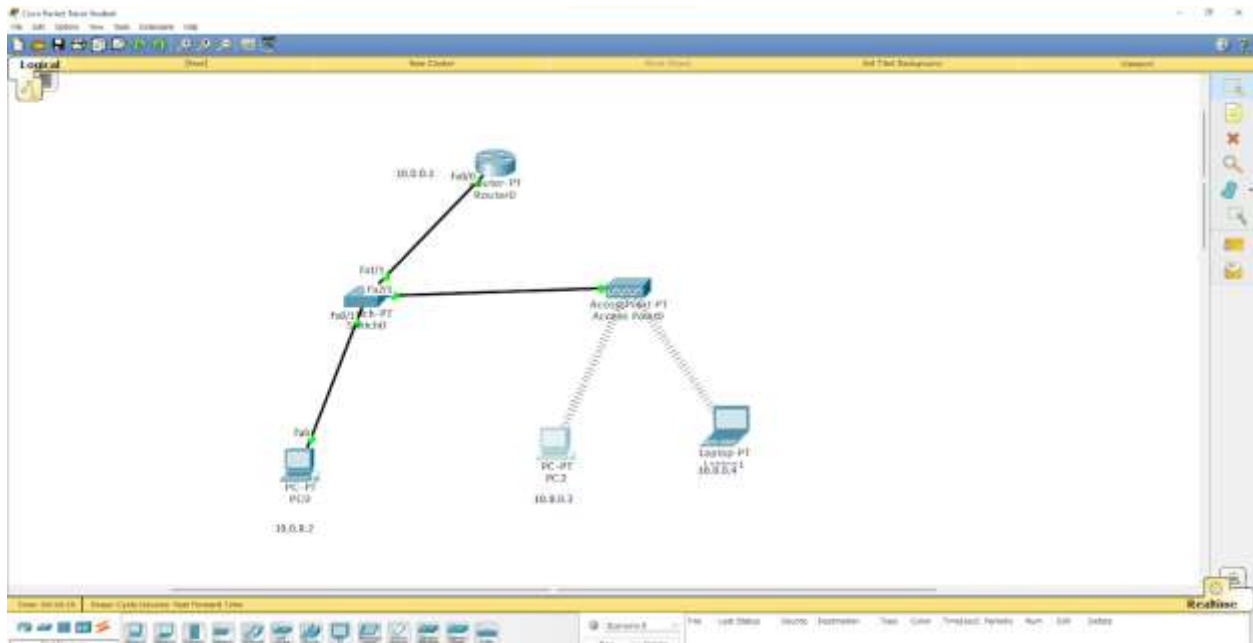
- Construct the topology
- set IP address and gateway of PC's
- for access point 0.
⇒ Input
SSID = WLAN
WLAN Key = 1234567890
- ⇒ for PC 1
Switch off drag the extra PT-MOAT-NM-1AM to the component list click drag WMP3 00N70 emp by port. Then switch on the device
In wireless 0.
SSID = WLAN
WLAN Key = 1234567890
IP address ⇒ 10.0.0.3
Gateway ⇒ 10.0.0.1
⇒ Repeat the same for laptop

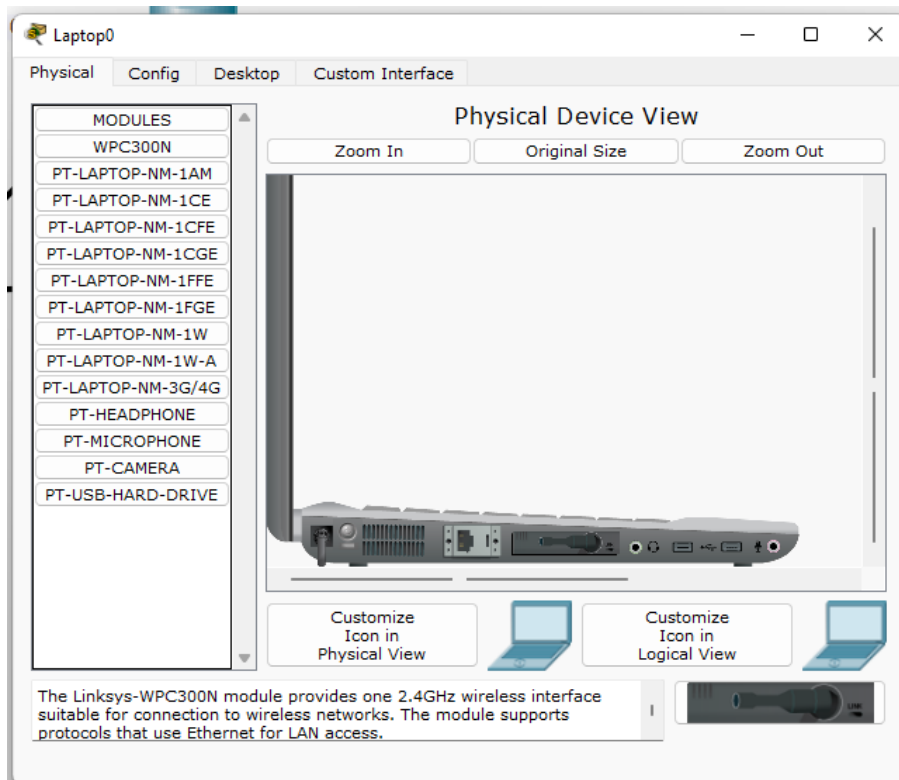
* Output -
 ping 10.0.0.3
 pinging 10.0.0.3 with 32 bytes of data :
 Reply from 10.0.0.3 : bytes = 32 bytes time = 15ms
 " " " "
 " " " "
 " " " "

Ping statistics for 10.0.0.3 :
 packet : sent = 4 , Received = 4 , Lost = 0
 Approximate round trip time in milliseconds:
 minimum = 6ms , maximum = 15ms , average 10ms

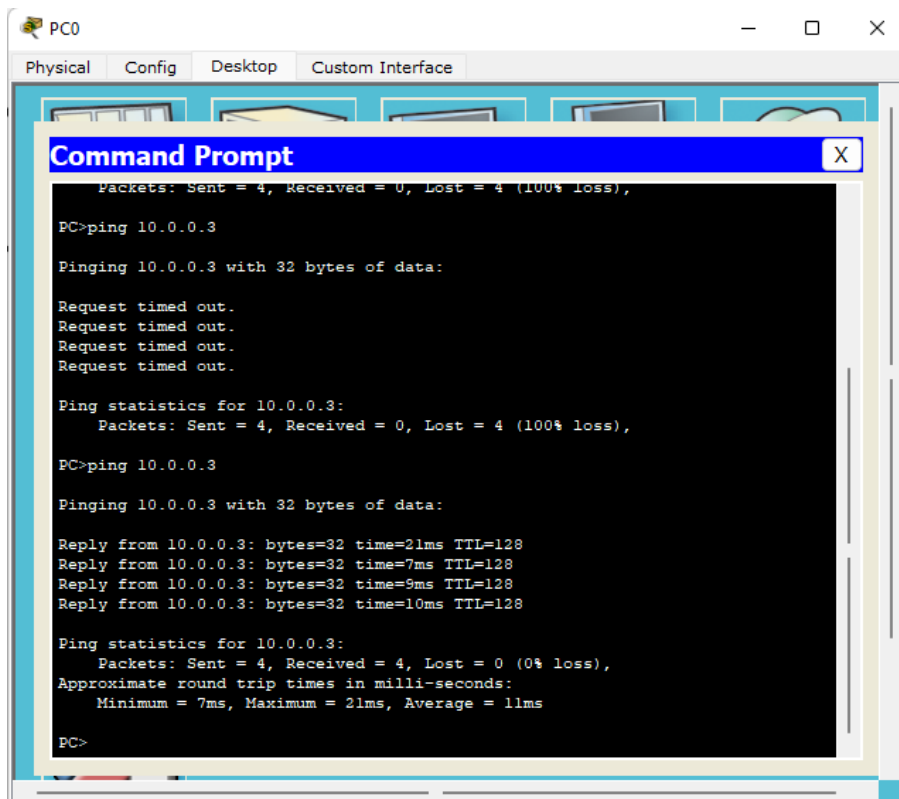
8/10
 1/9/23

TOPOLOGY:





OUTPUT:



WEEK 12

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

OBSERVATION:

12/10/23

Shilpa
PAGE NO. _____
DATE ____/____/____

* Aim - To understand the operation of telnet by accessing the router in server room from a PC in the office.

* Topology -

```
graph LR
    PC0[PC 0  
10.0.0.2] --- E00[E0/0] --- E00 --- Router1[Router 1  
10.0.0.1]
```

* Procedure -

- Create a topology as shown above with a PC and a router.
- Configure the PC & router normally.
- Go to all of the router & perform the following -
 - enable
 - config
 - hostname R1
 - Enable secret P1
 - interface fastethernet 0/0
 - ip address 10.0.0.1 255.0.0.0
 - no shut.
 - line vty 0,5 to allow virtual terminal access
 - login
 - password p0
 - exit
 - exit

wr - to save changes in router.

* Commands in PC -

ping 10.0.0.1

~~ping re~~

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=0ms TTL=64
 _____ " _____
 _____ " _____
 _____ " _____

ping statistics for 10.0.0.1

Packets: Sent = 4, received = 4, loss = 0 [0% loss]

Approximate round trip in milliseconds

minimum = 0ms, Maximum = 0ms, Average = 0ms

PC > telnet 10.0.0.1

trying 10.0.0.1... open

user access verification

15

password: 10

r1 > enable

8/10

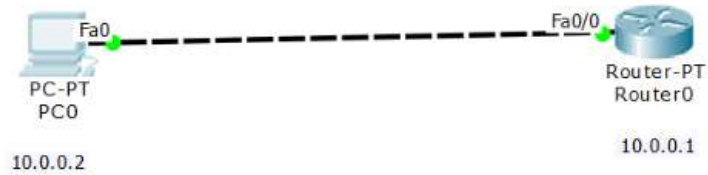
password: p1

r1# show ip route

1/9/23

10.0.0.0/8 is directly connected. FastEthernet

TOPOLOGY:



OUTPUT:

```
PC0
Physical Config Desktop Custom Interface

Command Prompt

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=1ms 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 = 1ms, Average = 0ms

PC>telnet 10.0.0.1
Trying 10.0.0.1 ...Open

User Access Verification

Password:
% Password: timeout expired!

[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:
Password:
Password:

[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>enable
Password:
r1#show ip route
Codes: C - connected, S - static, I - IGRP, E - EIGRP, H - OSPF, N - mobile, B - BGP
        D - RIPv2, 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, S - SGP
        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
#
```

WEEK 13

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

CODE:

```
#include<stdio.h>
int arr[17];

void xor(int x[], int y[])
{
    int k=0;
    for(int i=1;i<16;i++)
    {
        if(x[i]==y[i])
            arr[k++]=0;
        else
            arr[i]=1;
    }
}

void main()
{
    int dd[17],div[33],ze[17],i,k;

    printf("Enter the dataword \n");
    for(i=0;i<17;i++)
        scanf("%d",&div[i]);

    for(i=i;i<33;i++)
        div[i]=0;

    for(i=0;i<17;i++)
        ze[i]=0;
    printf("Enter dividend \n");
```

```

for(i=0;i<17;i++)
    scanf("%d",&dd[i]);

i=0;
k=0;
for(i=i;i<17;i++)
    arr[k++]=div[i];
while(i<33)
{
    if(arr[0]==0)
        xor(arr,ze);
    else
        xor(arr,dd);

    arr[16]=div[i++];

}
k=0;
for(i=17;i<33;i++)
    div[i]=arr[k++];
printf("Codeword: ");
for(i=0;i<33;i++)
    printf("%d",div[i]);

for(i=0;i<17;i++)
    arr[i]=0;

printf("\nAt receiver end \n");

k=0;
for(i=i;i<17;i++)
    arr[k++]=div[i];
while(i<33)
{
    if(arr[0]==0)

```



```

        xor(arr,ze);
    else
        xor(arr,dd);

    arr[16]=div[i++];

}
k=0;
for(i=17;i<33;i++)
    div[i]=arr[k++];

printf("Codeword: ");
for(i=0;i<33;i++)
    printf("%d",div[i]);
}

```

OUTPUT:

```

Enter the dataword
1 0 1 1 0 0 1 1 1 1 0 0 1 0 1 1 1
Enter dividend
1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 1
Codeword: 1011001111001011100000000000011011
At receiver end
Codeword: 1011001111001011100000000000000000
Process returned 1 (0x1)   execution time : 49.507 s
Press any key to continue.

```

OBSERVATION:

Q) Write a program for error detecting code using CRC - CCITT.

```
#include <stdio.h>
char m[80], g[50], s[50], q[50];
kmp[50];
```

```
void crc(int n)
```

```
{
```

```
    int i, j;
```

```
    for (i = 0; i < n; i++)
```

```
        temp[i] = m[i];
```

```
    for (i = 0; i < n - 16; i++) {
```

```
        if (m[i] == '1') {
```

```
            q[i] = '1';
```

```
            calram(); }
```

```
        else
```

```
            q[i] = '0';
```

```
            shift();
```

```
    }
```

```
    r[10] = m[17 + i];
```

```
    r[17] = '10';
```

```
    for (j = 0; j <= 17; j++)
```

```
        temp[j] = r[j];
```

```
    }
```

```
    q[n - 16] = '10';
```

```
}
```

```

void cal ram ()
{
    int i, j;
    for (i = 1; i <= 10; i++)
        r[i-1] = [(int) temp[i] - 48) ^
                [(int) g[i] - 48) + 48;
}

```

```

void shift ()
{
    int i;
    for (i = 1; i <= 16; i++)
        r[i-1] = r[i];
}

```

```

void zalt case (int n)
{
    int i, k = 0;
    for (i = n - 16; i < n; i++)
        m[i] = [(int) m[i] - 48) ^ [(int) i + (k++) - 48 + 48];
    m[i] = '0';
}

```

```

void main ()
{
    int n, i = 0;
    char ch, flag = 0;
    printf ("Enter frame bits");
    while ((ch = getch (stdin)) != '\n')
        m[i++] = ch;
}

```

```
n = 1;
for (i = 0; i < 16; i++)
    m[n++] = '0';
m[n] = '10';
```

```
printf("message after appending 16 zeros\n", m);
```

```
for (i = 0; i < 16; i++)
```

```
    g[i] = '0';
    g[0] = g[4] = g[11] = g[15] = '1';
    g[16] = '10';
```

```
printf("generator : %s\n", g);
```

```
crc(n);
```

```
printf("quotient : %s", q);
```

```
calttrans(n);
```

```
printf("Transmitted frame : %s", m);
```

```
printf("enter - transmitted frame: ");
```

```
scanf("%s", m);
```

```
printf("CRC checker\n");
```

```
crc(n);
```

```
printf("last remainder : %s", r);
```

```
for (i = 0; i < 16; i++)
```

```
    if (r[i] != '0')
```

```
        flag = '1';
```

```
    else
```

```
        continue;
```

```
    if (flag == '1')
```

```
        printf("error");
```

```
    else
```

```
        printf("frames all correct");
```

```
}
```

0/0

Enter frame bits : 1011

message after appending 16 zeros

1011 0000 0000 0000 0000

generator : 10001000000100001

quotient : 1011

transmitted : 1011 1011 0001 0110 1011

enter - transmitted frame

1011 1011 0001 0110 1011

Last remainder 0000 0000 0000 0000

received frame is correct

10/10

11/9/23

WEEK 14

Write a program for congestion control using Leaky bucket algorithm.

CODE:

```
#include <stdio.h>
#include <stdlib.h> // Include this for the rand() function
int main()
{
    int buckets, outlets, k = 1, num, remaining;
    printf("Enter Bucket size and outstream size\n");
    scanf("%d %d", &buckets, &outlets);
    remaining = buckets;
    while (k)
    {
        num = rand() % 1000; // Generate a random number between 0 and 999
        if (num < remaining)
        {
            remaining = remaining - num;
            printf("Packet of %d bytes accepted\n", num); // Added missing variable
        }
        else
        {
            printf("Packet of %d bytes is discarded\n", num);
        }
        if (buckets - remaining > outlets)
        {
            remaining += outlets; // Fixed the calculation
        }
        else
            remaining = buckets;
        printf("Remaining bytes: %d \n", remaining);
        printf("If you want to stop input, press 0, otherwise, press 1\n");
        scanf("%d", &k);
    }
}
```

```

    }
    while (remaining < buckets) // Fixed the condition
    {
        if (buckets - remaining > outlets)
        {
            remaining += outlets; // Fixed the calculation
        }
        else
            remaining = buckets;
        printf("Remaining bytes: %d \n", remaining);
    }
    return 0; // Added a return statement to indicate successful completion
}

```

OUTPUT:

```

PS D:\VS Code> cd "D:\VS Code\06\" ; if ($?) { gcc bucket.c -o bucket } ; if ($?) { .\bucket }
Enter Bucket size and outstream size
2000
100
Packet of 41 bytes accepted
Remaining bytes: 2000
If you want to stop input, press 0, otherwise, press 1
1
Packet of 467 bytes accepted
Remaining bytes: 1633
If you want to stop input, press 0, otherwise, press 1
1
Packet of 334 bytes accepted
Remaining bytes: 1399
If you want to stop input, press 0, otherwise, press 1
1
Packet of 500 bytes accepted
Remaining bytes: 999
If you want to stop input, press 0, otherwise, press 1
1
Packet of 169 bytes accepted
Remaining bytes: 930
If you want to stop input, press 0, otherwise, press 1
1
Packet of 724 bytes accepted
Remaining bytes: 306
If you want to stop input, press 0, otherwise, press 1
1
Packet of 478 bytes is discarded
Remaining bytes: 406
If you want to stop input, press 0, otherwise, press 1
1
Packet of 358 bytes accepted
Remaining bytes: 148
If you want to stop input, press 0, otherwise, press 1
1
Packet of 962 bytes is discarded
Remaining bytes: 248
If you want to stop input, press 0, otherwise, press 1
0
Remaining bytes: 348
Remaining bytes: 448
Remaining bytes: 548
Remaining bytes: 648
Remaining bytes: 748

```

```

Remaining bytes: 348
Remaining bytes: 448
Remaining bytes: 548
Remaining bytes: 648
Remaining bytes: 748
Remaining bytes: 848
Remaining bytes: 948
Remaining bytes: 1048
Remaining bytes: 1148
Remaining bytes: 1248
Remaining bytes: 1348
Remaining bytes: 1448
Remaining bytes: 1548
Remaining bytes: 1648
Remaining bytes: 1748
Remaining bytes: 1848
Remaining bytes: 1948
Remaining bytes: 2000
PS D:\VS Code\OS>

```

OBSERVATION:

② WAP for congestion control using leaky bucket algorithm -

```

#include <stdio.h>
int main()
{
    int incoming, outgoing, bucketSize, n, store = 0;
    printf("Enter bucket size, outgoing rate and no. of packets\n");
    scanf("%d %d %d", &bucketSize, &outgoing, &n);
    while(n-- > 0)
    {
        printf("Enter 1-incoming packet\n");
        scanf("%d", &incoming);
        printf("Incoming packet size %d\n", incoming);
        if(incoming <= (bucketSize - store))
        {
            store += incoming;
        }
        else
        {
            printf("dropped %d. no of packets left\n", incoming - (bucketSize - store));
            incoming = (bucketSize - store);
            printf("bucket buffer size %d out of %d\n", store, bucketSize);
            store = bucketSize;
        }
        n--;
        store = store - outgoing;
    }
    printf("After outgoing %d packets left out %d\n", bucketSize, store);
    return 0;
}

```

Q/P

enter bucket size, outgoing rate & no. of
I/P.

20 10 2

enter the incoming packet size = 30

Incoming packet size = 30

dropped 10 no. of packets

bucket buffer size 0 out of 20

After outgoing 10 packet left out 20
in buffer

enter the incoming packet size = 10

Incoming packet size = 10

Bucket buffer size 10 out 20

After outgoing 10 packets left out
20 in buffer.

10/10

1/9/23

WEEK 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.

CODE:

ClientTCP.py

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

ServerTCP.py

```
from socket import *
serverName="127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
    print ("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    file=open(sentence,"r")
    l=file.read(1024)
    connectionSocket.send(l.encode())
```



```
print ("\nSent contents of " + sentence)
file.close()
connectionSocket.close()
```

OUTPUT:



```
Python 3.11.4 (tags/v3.11.4:0d2340ef, Jun 7 2023, 04:45:37) [AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Admin\Desktop\lhm2lcs045\ClientTCP.py =====
Enter file name:ServerTCP.py.

From server:

from socket import *
serverName="127.0.0.1"
serverPort=12000
serverSocket=socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen()
while 1:
    print("The server is ready to receive")
    connectionSocket,addr=serverSocket.accept()
    sentence=connectionSocket.recv(1024).decode()
    file=open(sentence,"r")
    data=file.read(1024)
    connectionSocket.sendall(data)
    print("Sent contents of " + sentence)
    file.close()
    connectionSocket.close()
>>>
```

```
Python 3.11.4 (tags/v3.11.4:0d2340ef, Jun 7 2023, 04:45:37) [AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Admin\Desktop\lhm2lcs045\ServerTCP.py =====
The server is ready to receive

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

OBSERVATION:

- * 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.

Server

1. A server has a `bind()` method which binds to specific IP & port so that it can listen to incoming request on that IP port and port.
2. A server has a `listen()` method which puts the server into listening mode. This allows server to listen to incoming connections.
3. Server has `accept()` and `close()` methods.
`accept()` - initiates a connection with client.
`close()` - closes the connection with the client.

Step 1. Open idle, in that in file, open new file and write the following code & name as `server.py`.

```
server TCP.py
from socket import *
server Name = "127.0.0.1"
server Port = 12000
server socket = socket(AF_INET, SOCKET_STREAM)
server socket.bind(server Name, server Port)
server socket.listen(1)
```

Adom
PAGE NO.
DATE / /

```

while 1:
    print("The server is ready to receive  
connection socket, addr= serverSocket  
accept()")
    sentence = connectionSocket.recv(1024)
    file = open(sentence, "r")
    I = file.read(1024)
    connectionSocket.send(I.encode())
    print("In sent contents of + sentence")
    file.close()
    connectionSocket.close()

```

Step 2: Run the file server.py.

O/P → The server is ready to receive
This shows that server is working.

Client

Step 1: Make a socket object.

Step 2: Establish a connection with
server and lastly we will receive
data from the server and close
the function.

Step 3:

open idle and open new file & write
the following code & name as "client.py"

```

from socket import *
Server Name = "127.0.0.1"
Server Port = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((Server Name, Server Port))

```

```

sentence = input("Enter file name : ")
client_socket.send(sentence.encode())
file_contents = client_socket.recv(1024).decode()

print("In Frome server : \n")
print(file_contents)
client_socket.close()

```

Run the file client.py

Client o/p

o/p \Rightarrow Enter file name : server.py

Server o/p

The server is ready to receive sent the contents of server.py

The server is ready to receive

10/10

2/9/23

WEEK 16

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

CODE:

ClientUDP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("\nEnter file name: ")
clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))
filecontents,serverAddress = clientSocket.recvfrom(2048)
print ("\nReply from Server:\n")
print (filecontents.decode("utf-8"))
# for i in filecontents:
# print(str(i), end = " ")
clientSocket.close()
clientSocket.close()
```

ServerUDP.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print ("The server is ready to receive")
while 1:
sentence, clientAddress = serverSocket.recvfrom(2048)
sentence = sentence.decode("utf-8")
file=open(sentence,"r")
```



```

con=file.read(2048)
serverSocket.sendto(bytes(con,"utf-8"),clientAddress)
print ("\nSent contents of ", end = "")
print (sentence)
# for i in sentence:
# print (str(i), end = "")
file.close()

```

OUTPUT:

The image shows two side-by-side screenshots of a Python Shell window. The left window displays the client code, and the right window displays the server code. Both windows show the execution of the code, with the server window indicating that the server is ready to receive and has sent the contents of the file.

```

Python 3.11.4 (tags/v3.11.4:0d2404ef, Jun 7 2023, 05:45:37) [AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:\Users\Admin\Desktop\lmslms063\ClientUDP.py
Enter file name: ServerUDP.py
Reply from Server:
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print ("The server is ready to receive")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file=open(sentence,"r")
    con=file.read(2048)
    serverSocket.sendto(bytes(con,"utf-8"),clientAddress)
    print ("\nSent contents of ", end = "")
    print (sentence)
    # for i in sentence:
    # print (str(i), end = "")
    file.close()
>>>

```

```

Python 3.11.4 (tags/v3.11.4:0d2404ef, Jun 7 2023, 05:45:37) [AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:\Users\Admin\Desktop\lmslms063\ServerUDP.py
The server is ready to receive
Sent contents of ServerUDP.py
|
>>>

```

OBSERVATION:

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

Here, like in TCP/IP we create socket object and bind it to specified port and server will be continuously listening when the client sends request is responds accordingly.

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(2048)
    sentence = sentence.decode("utf-8")
    file = open(sentence, "r")
    con = file.read(2048)
    serverSocket.sendto(bytes(con, "utf-8"), clientAddress)
    print("\n sent contents of 'req.txt'")
    print(sentence)
```

client UDP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
```

```

client socket = socket (AF_INET, SOCK_STREAM)
sentence = input ("Enter file name: ")
client socket. send to (bytes (sentence,
"utf-8"), (server Name, server port))
file contents, server Address = client socket.
recvfrom (2048)
print ("In Reply from server: ")
print (file contents. decode ("utf-8"))

```

O/P (client)

Enter the file name: server UDP.py.

O/P (server)

The server is ready to receive sent contents of server UDP.py.

10/10
12
2/19/23