VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

COMPUTER NETWORKS

Submitted by

Pradyumna H (1BM21CS130)

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 **Pradyumna H (1BM21CS130),** 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. Swathi Sridharan
Associate Professor
Department of CSE
BMSCE, Bengaluru

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

`

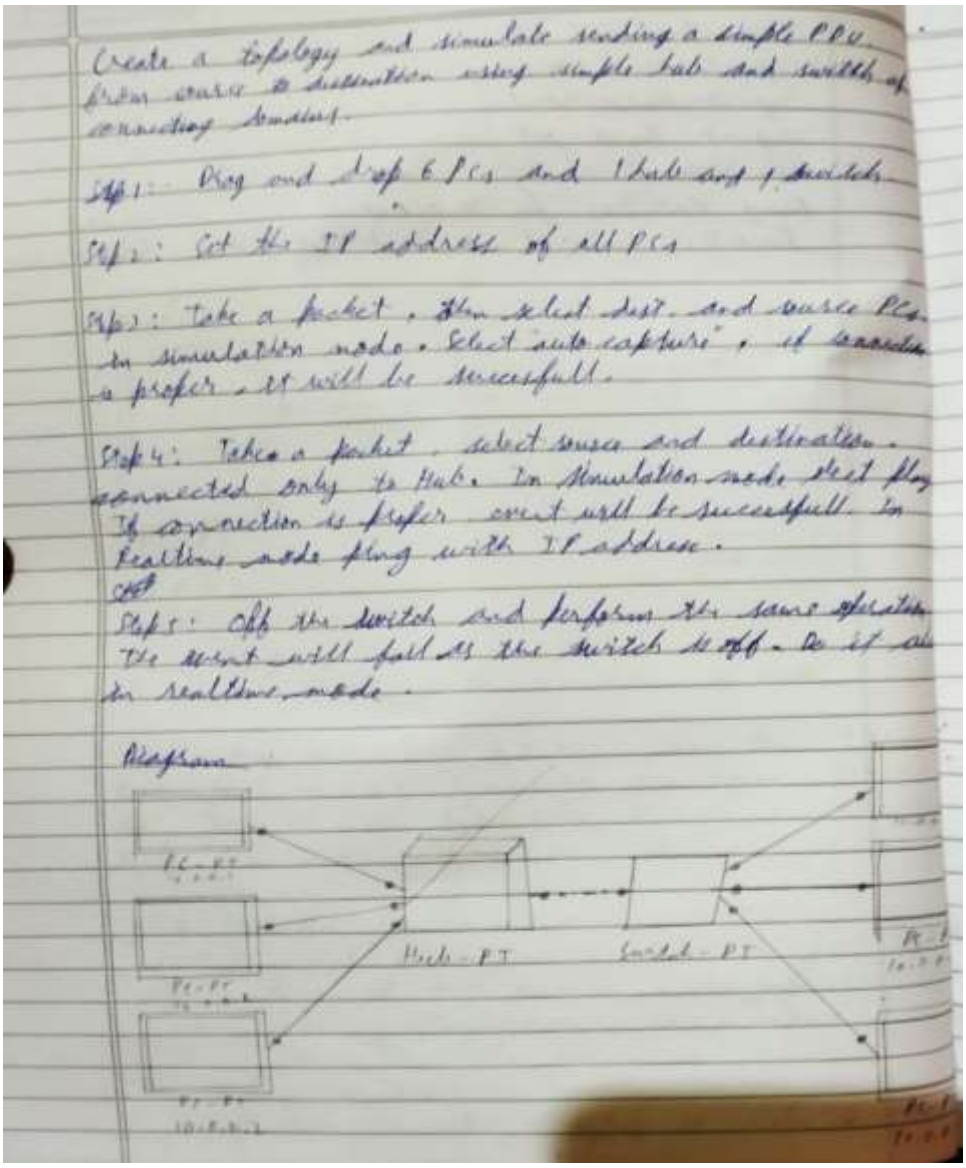
Index

Sl. No.	Date	Experiment Title	Page No.
		CYCLE 1	
1	15/6/23	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	22/6/23	Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.	9
3	13/7/23	Configure default route, static route to the Router.	18
4	13/7/23	Configure DHCP within a LAN and outside LAN.	23
5	20/7/23	Configure Web Server, DNS within a LAN.	32
6	20/7/23	Configure RIP routing Protocol in Routers.	35
7	27/7/23	Configure OSPF routing protocol.	40
8	3/8/23	To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP).	45
9	10/8/23	To construct a VLAN and make a pc communicate among VLAN.	49
10	10/8/23	Demonstrate the TTL/ Life of a Packet.	53
11	10/8/23	To construct a WLAN and make the nodes communicate wirelessly.	58
12	10/8/23	To understand the operation of TELNET by accessing the router in server room from a PC in IT office.	62
		CYCLE 2	
13	17/8/23	Write a program for error detecting code using CRC CCITT (16-bits).	66
14	17/8/23	Write a program for congestion control using Leaky bucket algorithm.	72
15	24/8/23	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.	76
16	24/8/23	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.	80

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:



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

Step 1:- Drag and drop 6 PCs and 1 hub and 1 switch.

Step 2: Set the IP address of all PCs

Step 3: Take a packet, then select dest. and source PCs. In simulation mode. Select "auto capture", if connection is proper, it will be successfull.

Step 4: Take a packet, select source and destination. connected only to Hub. In Simulation mode, select ping. If connection is proper, event will be successfull. In Realtime mode ping with IP address.

Step 5: Off the switch and perform the same operation. The event will fail as the switch is off. Do it also in realtime mode.

Diagram

Confirm ping:
when switch is off
Ping 10.0.0.1 in (10.0.0.4)
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)

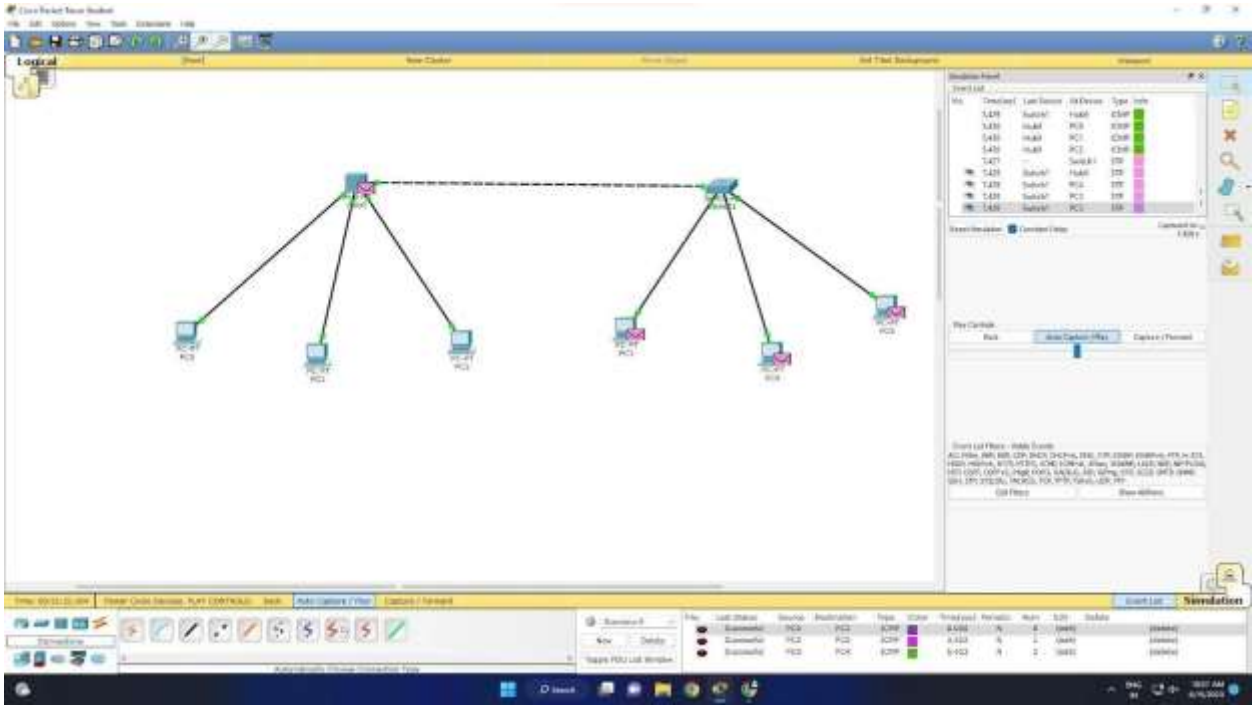
Transfer via Hub and switch: -
Ping 10.0.0.5 in (10.0.0.2)

Pinging 10.0.0.5 with 32 bytes of data:
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
Reply from 10.0.0.5 bytes = 32 time = 0ms TTL = 128

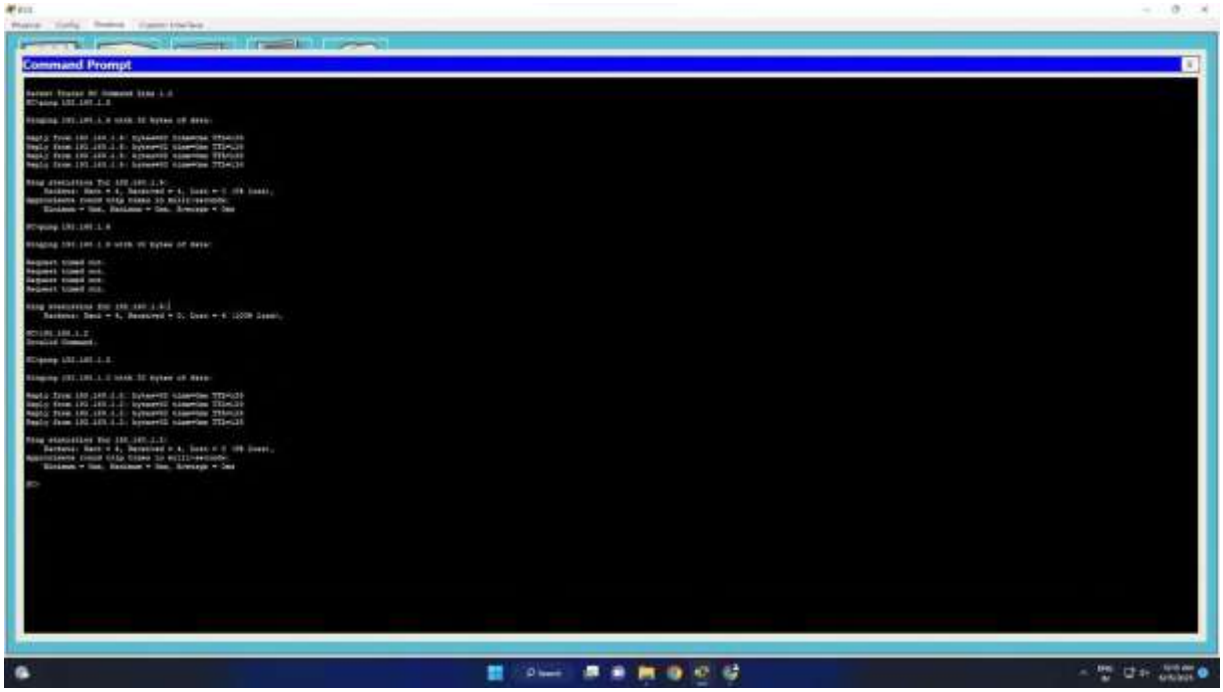
Ping statistics for 10.0.0.5:
Packets: sent = 4 Received = 4 lost = 0 (0% lost)

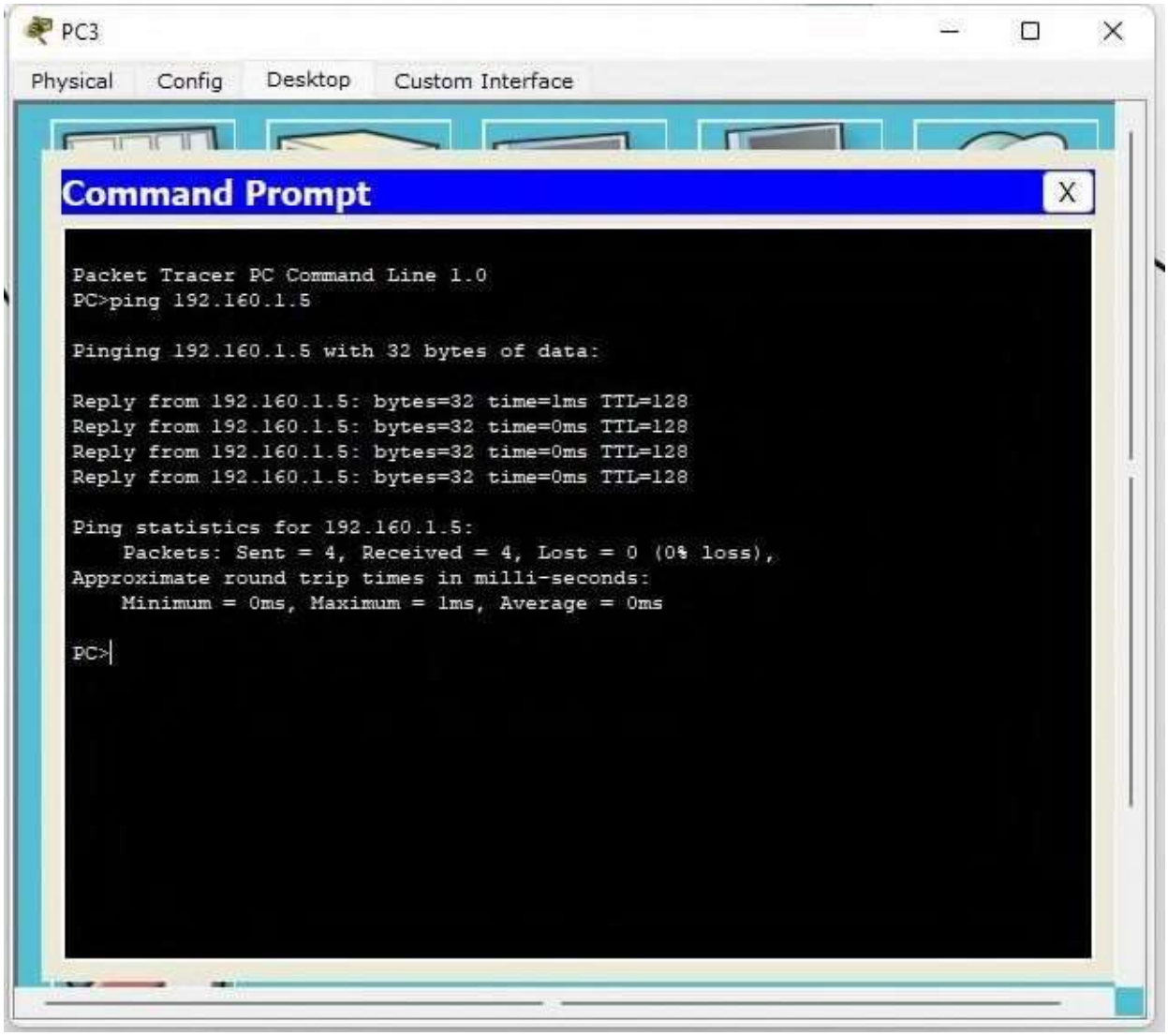
Approximate round trip times in milli seconds
minimum = 0ms. Maximum = 0ms, Average = 0ms.

TOPOLOGY:



OUTPUT:

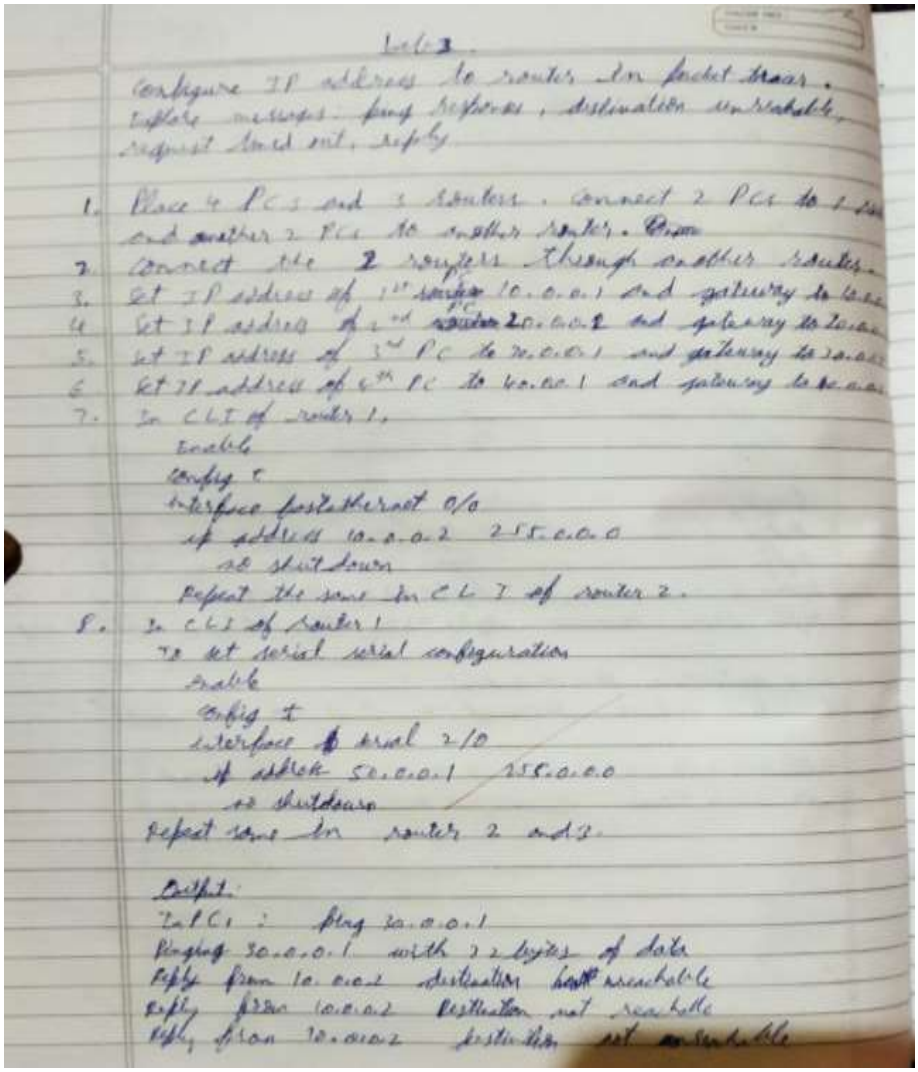



PC3
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 192.160.1.5
Pinging 192.160.1.5 with 32 bytes of data:
Reply from 192.160.1.5: bytes=32 time=1ms TTL=128
Reply from 192.160.1.5: bytes=32 time=0ms TTL=128
Reply from 192.160.1.5: bytes=32 time=0ms TTL=128
Reply from 192.160.1.5: bytes=32 time=0ms TTL=128
Ping statistics for 192.160.1.5:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 1ms, Average = 0ms
PC>

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:



Lab 3.

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

1. Place 4 PCs and 3 routers. Connect 2 PCs to 1 router and another 2 PCs to another router. Then
2. Connect the 2 routers through another router.
3. Set IP address of 1st PC 10.0.0.1 and gateway to 10.0.0.2
4. Set IP address of 2nd PC 20.0.0.2 and gateway to 20.0.0.1
5. Set IP address of 3rd PC to 30.0.0.1 and gateway to 30.0.0.2
6. Set IP address of 4th PC to 40.0.0.1 and gateway to 40.0.0.2
7. In CLI of router 1,

```
enable
config t
interface fastethernet 0/0
 ip address 10.0.0.2 255.0.0.0
 no shutdown
```

Repeat the same in CLI of router 2.

8. In CLI of Router 1
To set serial configuration

```
enable
config t
interface serial 2/0
 ip address 50.0.0.1 255.0.0.0
 no shutdown
```

Repeat same in router 2 and 3.

Output:

In PC1 : ping 30.0.0.1
Pinging 30.0.0.1 with 32 bytes of data
Reply from 10.0.0.2 destination host unreachable
Reply from 10.0.0.2 Destination net unreachable
Reply from 10.0.0.2 destination net unreachable

Ping statistics for 30.0.0.1
Packets sent = 4 received = 0 lost = 4 (100% loss)
Since IP address 30.0.0.1 is not directly connected to router 1.

In CLI of router 1.
To set static configuration
enable
config t
ip route 30.0.0.0 255.0.0.0 50.0.0.3
ip route 20.0.0.0 255.0.0.0 50.0.0.2
ip route 60.0.0.0 255.0.0.0 50.0.0.1
Repeat same to router 2 and 3.

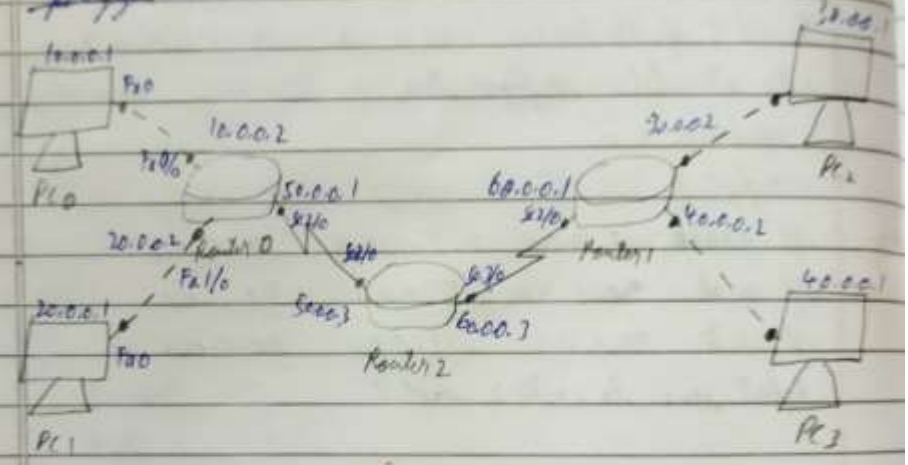
In PC1 : Ping 30.0.0.1
Pinging 30.0.0.1 with 32 bytes of data
Reply from 30.0.0.1 bytes = 32 time = 5ms TTL = 125
Reply from 30.0.0.1 bytes = 32 time = 2ms TTL = 125
Reply from 30.0.0.1 bytes = 32 time = 1ms TTL = 125

Ping statistics for 30.0.0.1
Packet: Sent = 4 Received = 3 lost = 1 (25% loss)
Approximate round trip times in milliseconds:
minimum = 2ms Maximum = 6ms Average = 5ms
First is lost as it takes time to identify the path.

Ping 3.0.0.1
Pinging 30.0.0.1 with 32 bytes of data
Reply from 30.0.0.1: bytes = 32 time = 3ms TTL = 125
Reply from 30.0.0.1: bytes = 32 time = 11ms TTL = 125
Reply from 30.0.0.1: bytes = 32 time = 4ms TTL = 125
Reply from 30.0.0.1 bytes = 32 time = 6ms TTL = 125
Ping statistics for 30.0.0.1:

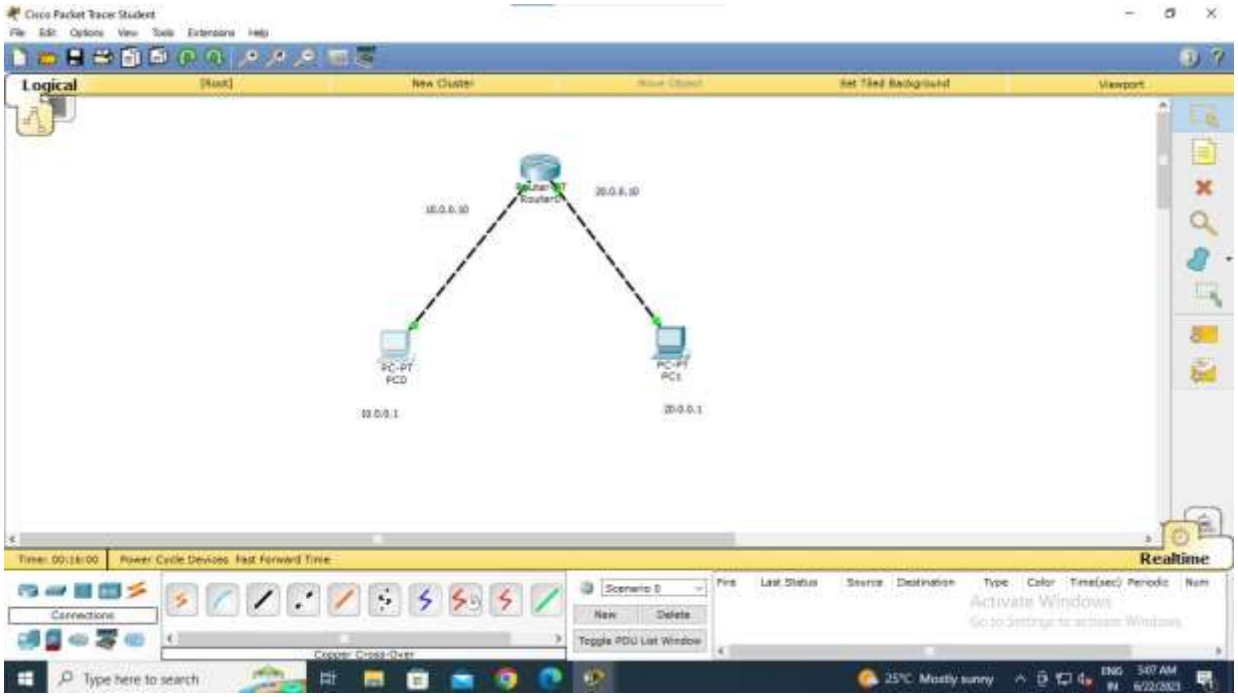
Packet sent = 4 received = 4 lost = 0 (0% loss)
Approximate round trip times in milli seconds
Minimum = 3ms, Maximum = 11ms Average = 5ms

Topology :-

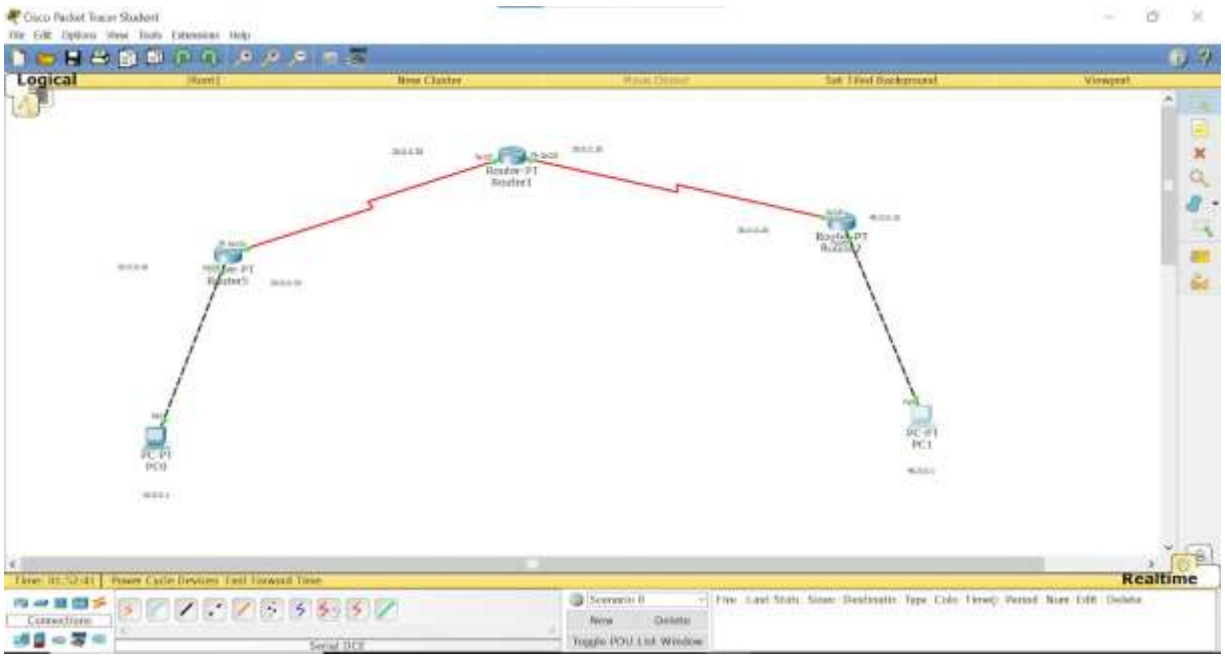


TOPOLOGY:

PROGRAM 2.1

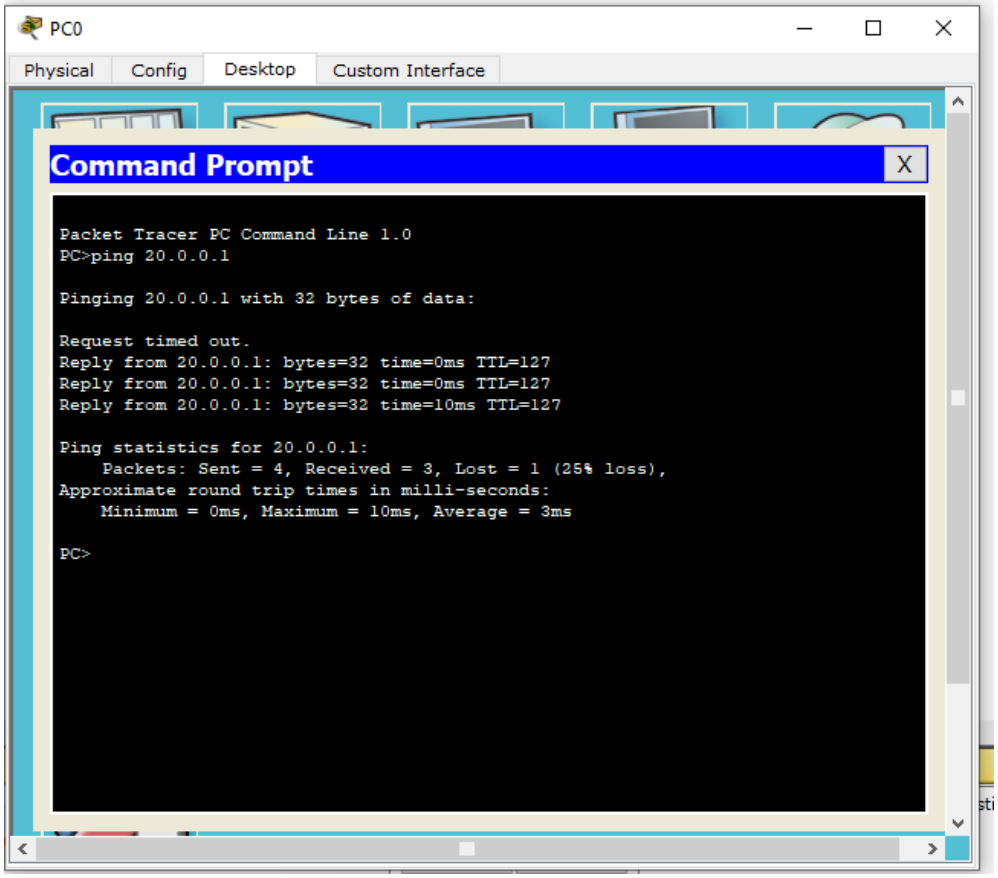


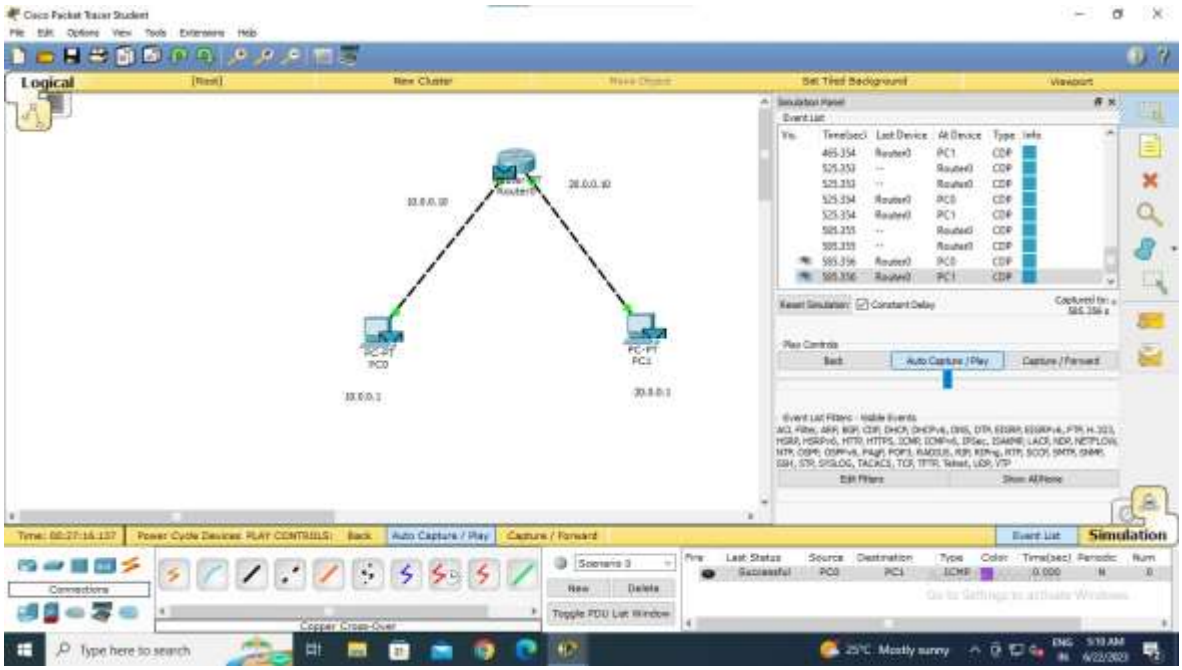
PROGRAM 2.2



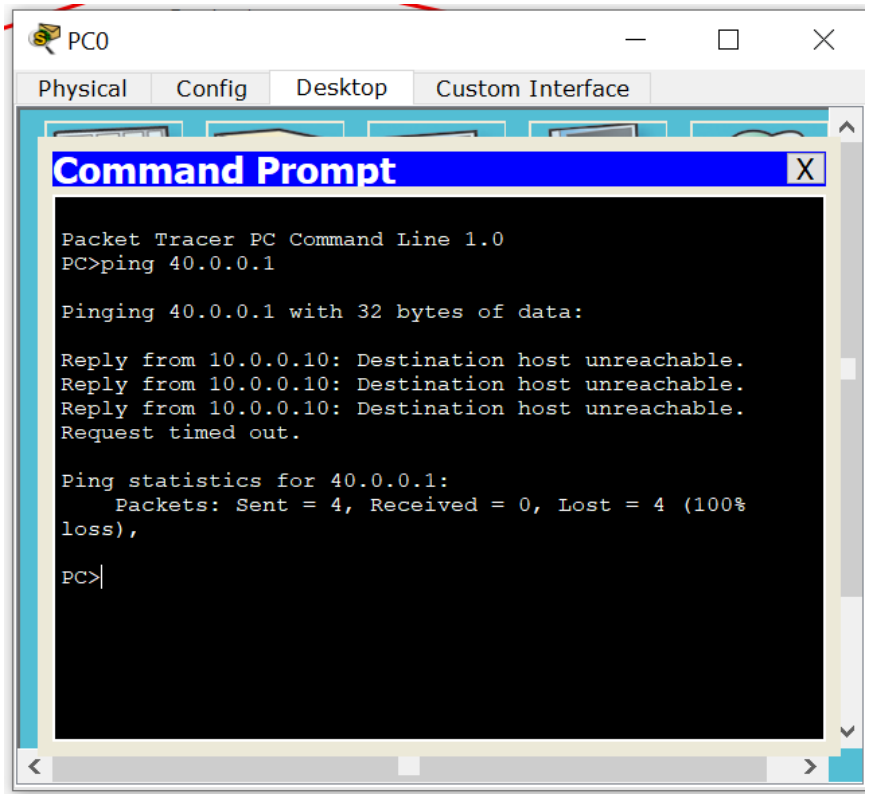
OUTPUT:

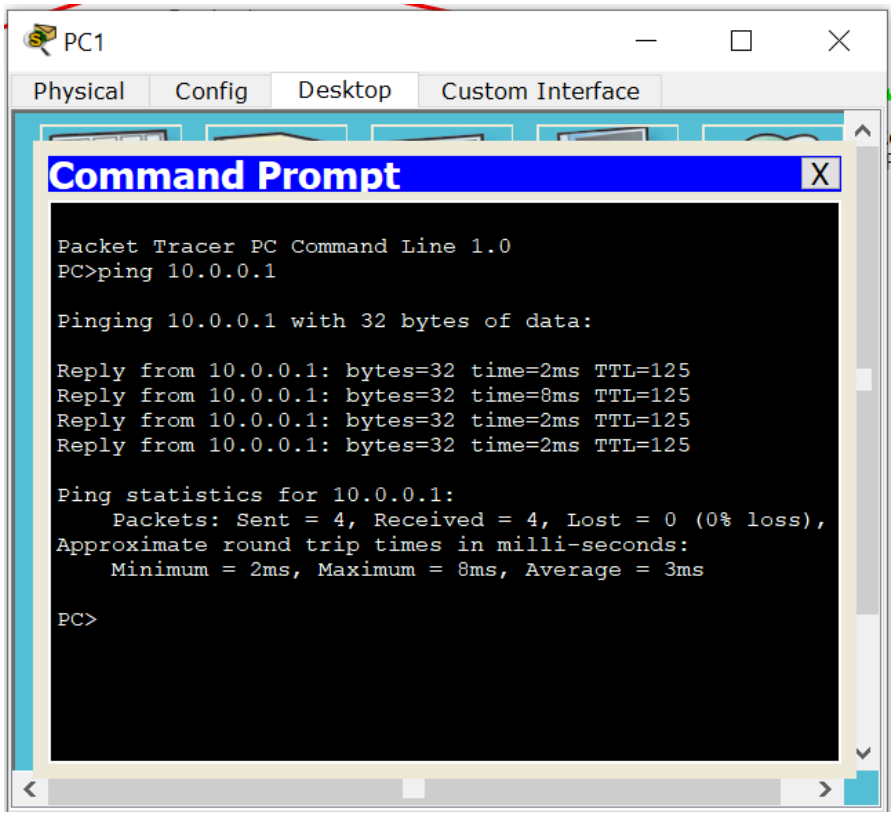
PROGRAM 2.1

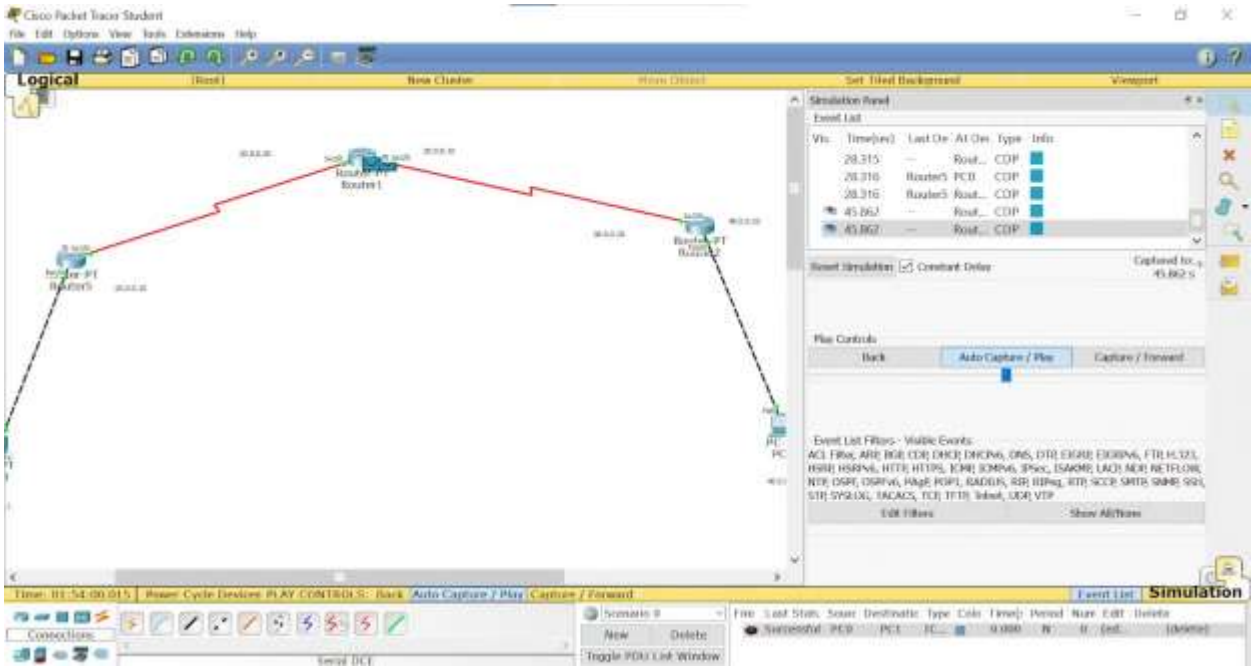




PROGRAM 2.2


PC0
Physical
Config
Desktop
Custom Interface
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:
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.
Request timed out.
Ping statistics for 40.0.0.1:
Packets: Sent = 4, Received = 0, Lost = 4 (100%
loss),
PC>

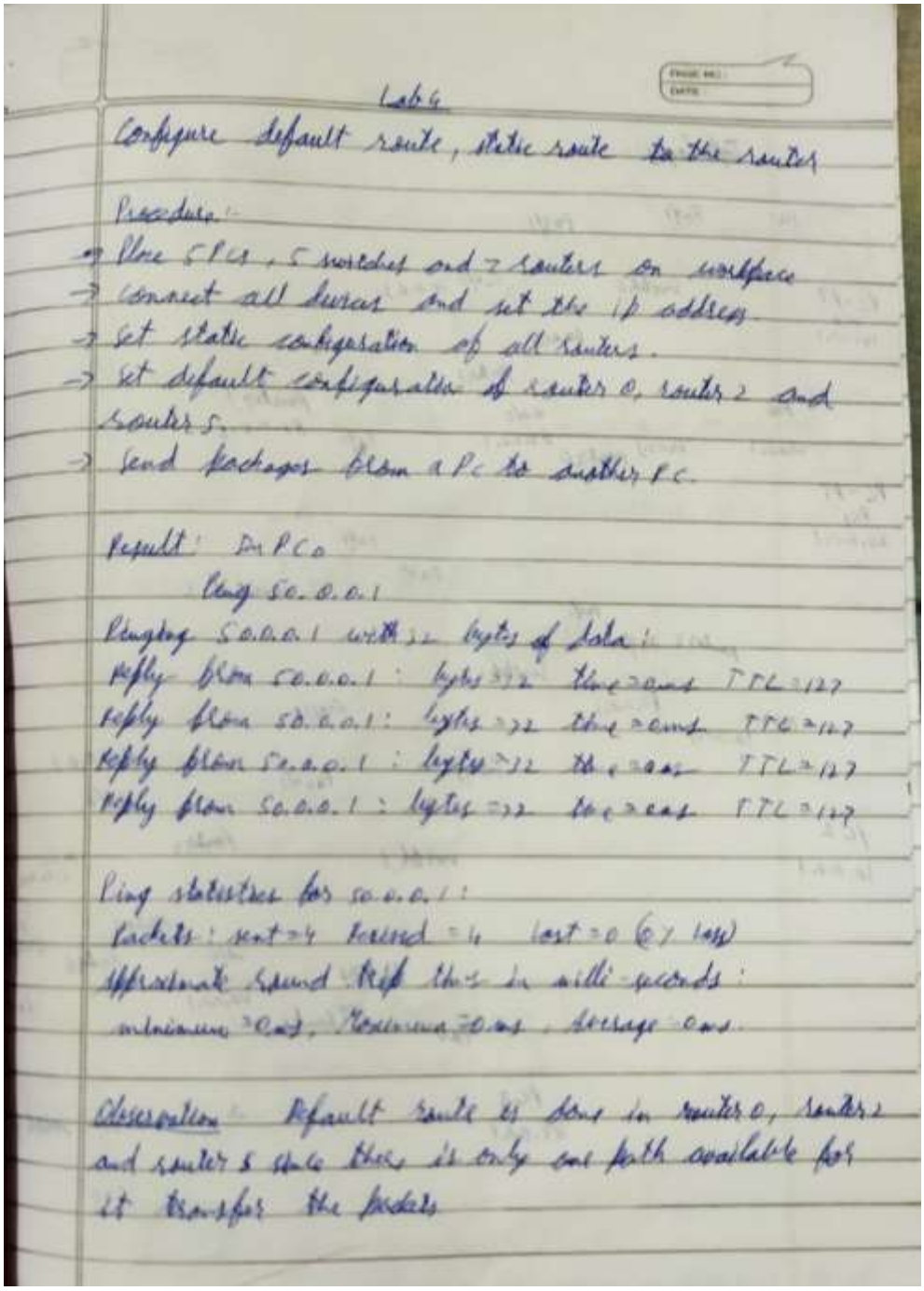

PC1
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.1
Pinging 10.0.0.1 with 32 bytes of data:
Reply from 10.0.0.1: bytes=32 time=2ms TTL=125
Reply from 10.0.0.1: bytes=32 time=8ms TTL=125
Reply from 10.0.0.1: bytes=32 time=2ms TTL=125
Reply from 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 = 8ms, Average = 3ms
PC>


Cisco Packet Tracer Student
Logical
Simulation Panel
Event List
Simulation

WEEK 3

Configure default route, static route to the Router.

OBSERVATION:



Lab 4

Configure default route, static route to the router

Procedure:

→ Place 5 PCs, 5 switches and 2 routers on workspace
→ Connect all devices and set the IP address.
→ Set static configuration of all routers.
→ Set default configuration of router 0, router 2 and router 5.
→ Send packages from a PC to another PC.

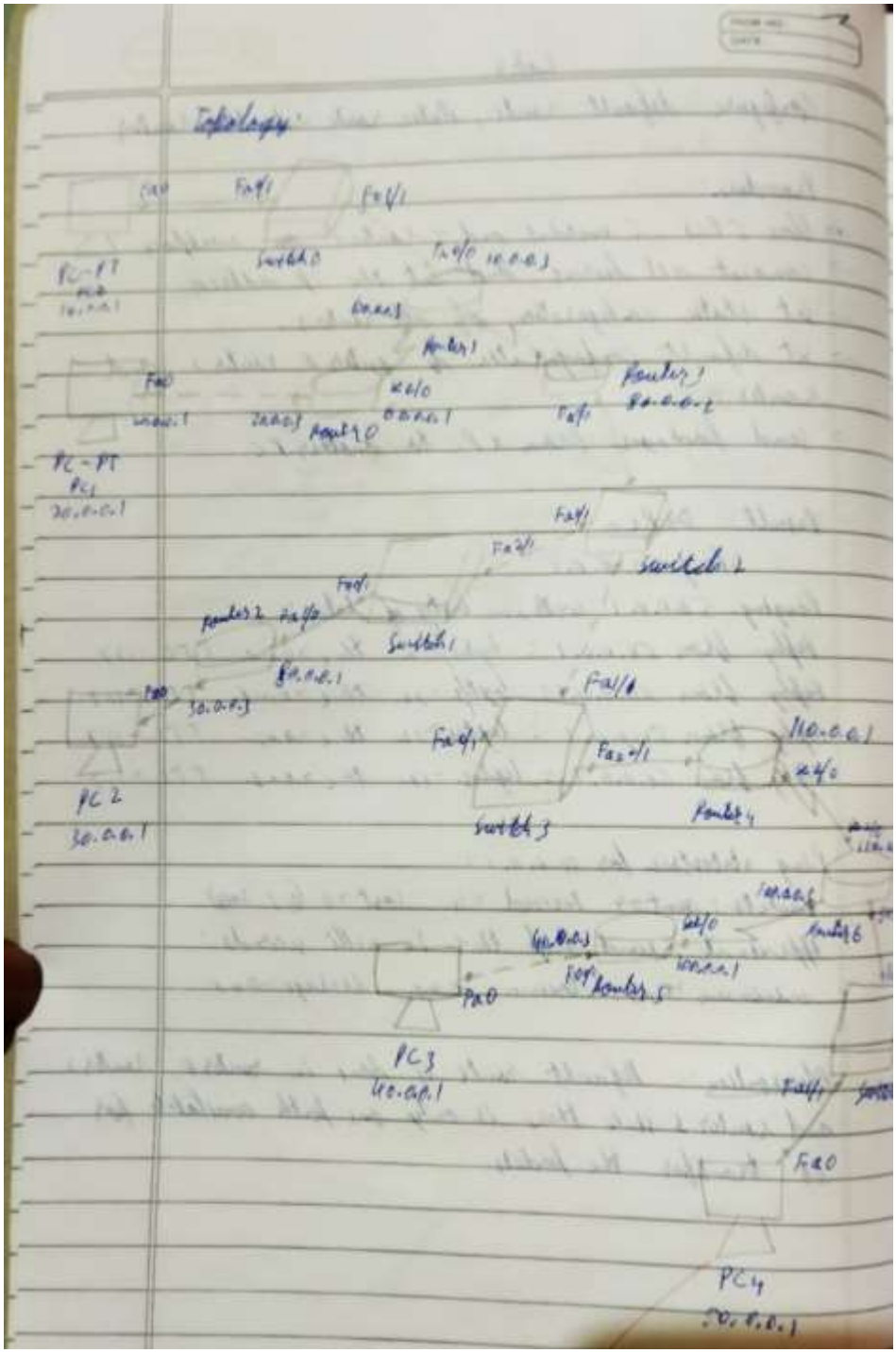
Result: In PC0

ping 50.0.0.1

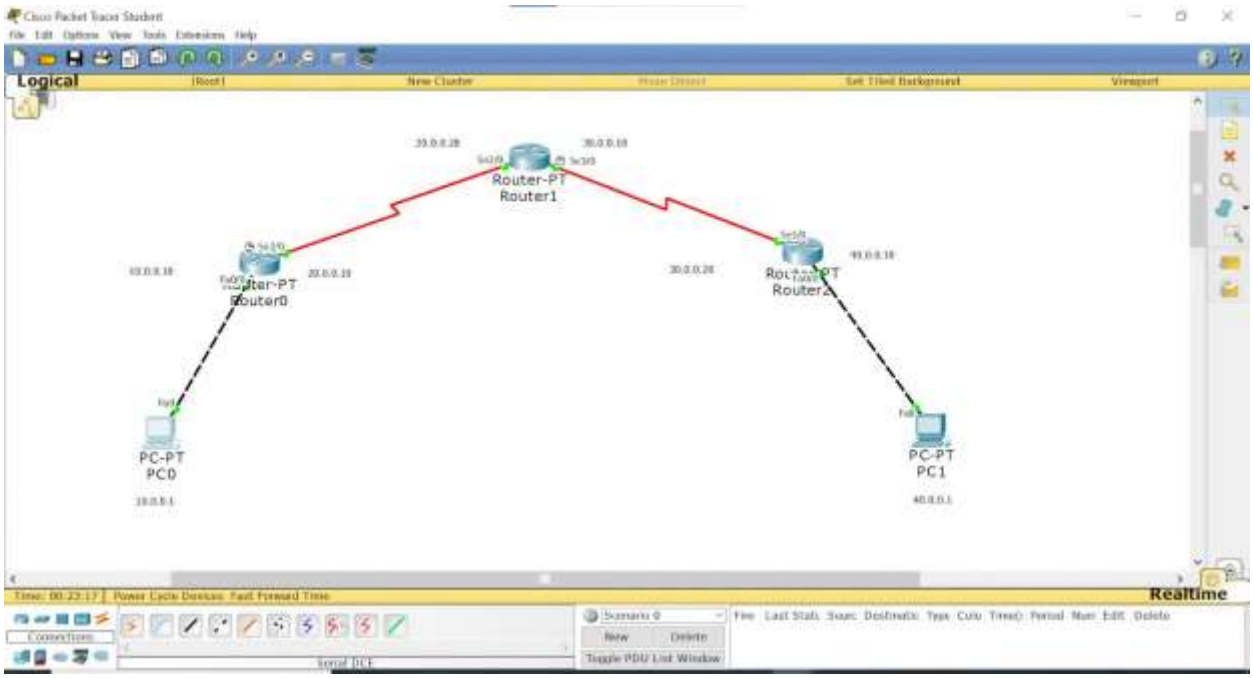
Pinging 50.0.0.1 with 32 bytes of data:
Reply from 50.0.0.1: bytes=32 time=0ms TTL=127
Reply from 50.0.0.1: bytes=32 time=0ms TTL=127
Reply from 50.0.0.1: bytes=32 time=0ms TTL=127
Reply from 50.0.0.1: bytes=32 time=0ms TTL=127

Ping statistics for 50.0.0.1:
Packets: sent=4 received=4 lost=0 (0% loss)
Approximate round trip times in milli-seconds:
minimum=0ms, Maximum=0ms, Average=0ms.

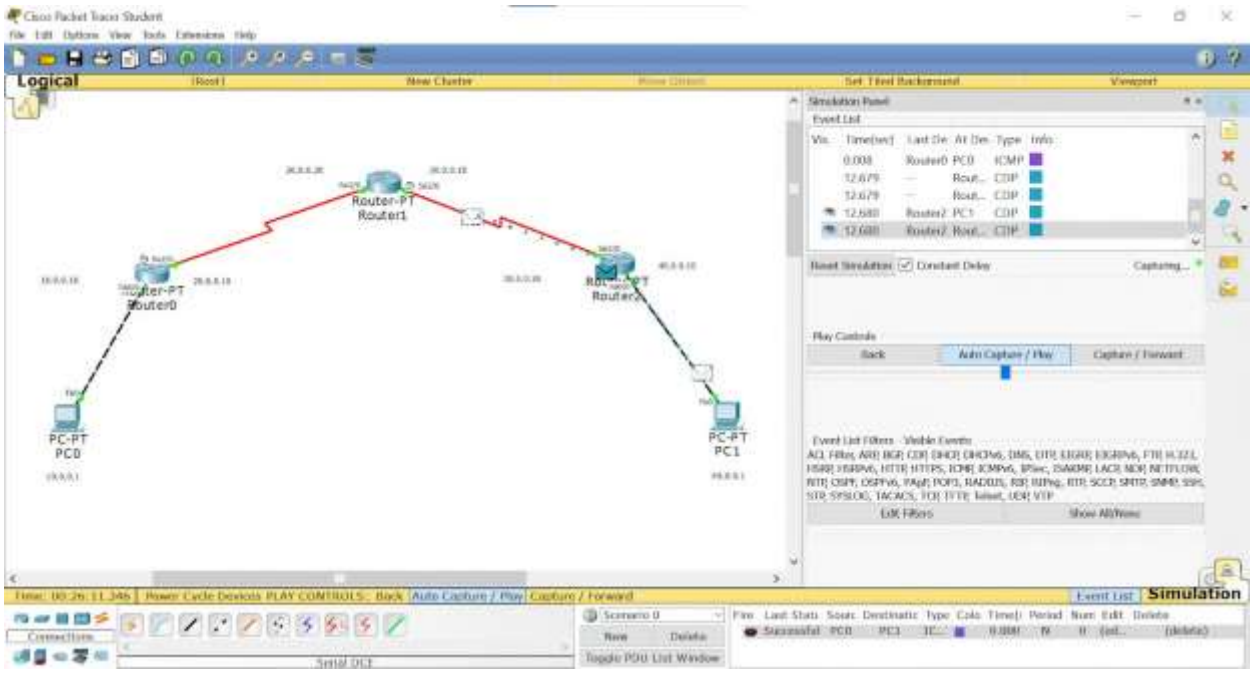
Observation: Default route is done in router 0, router 2 and router 5 since there is only one path available for it transfer the packets


Topology
PC-PT
PC-PT
PC 2
PC3
PC4

TOPOLOGY:



OUTPUT:



PC0

Physical | Config | Desktop | Custom Interface

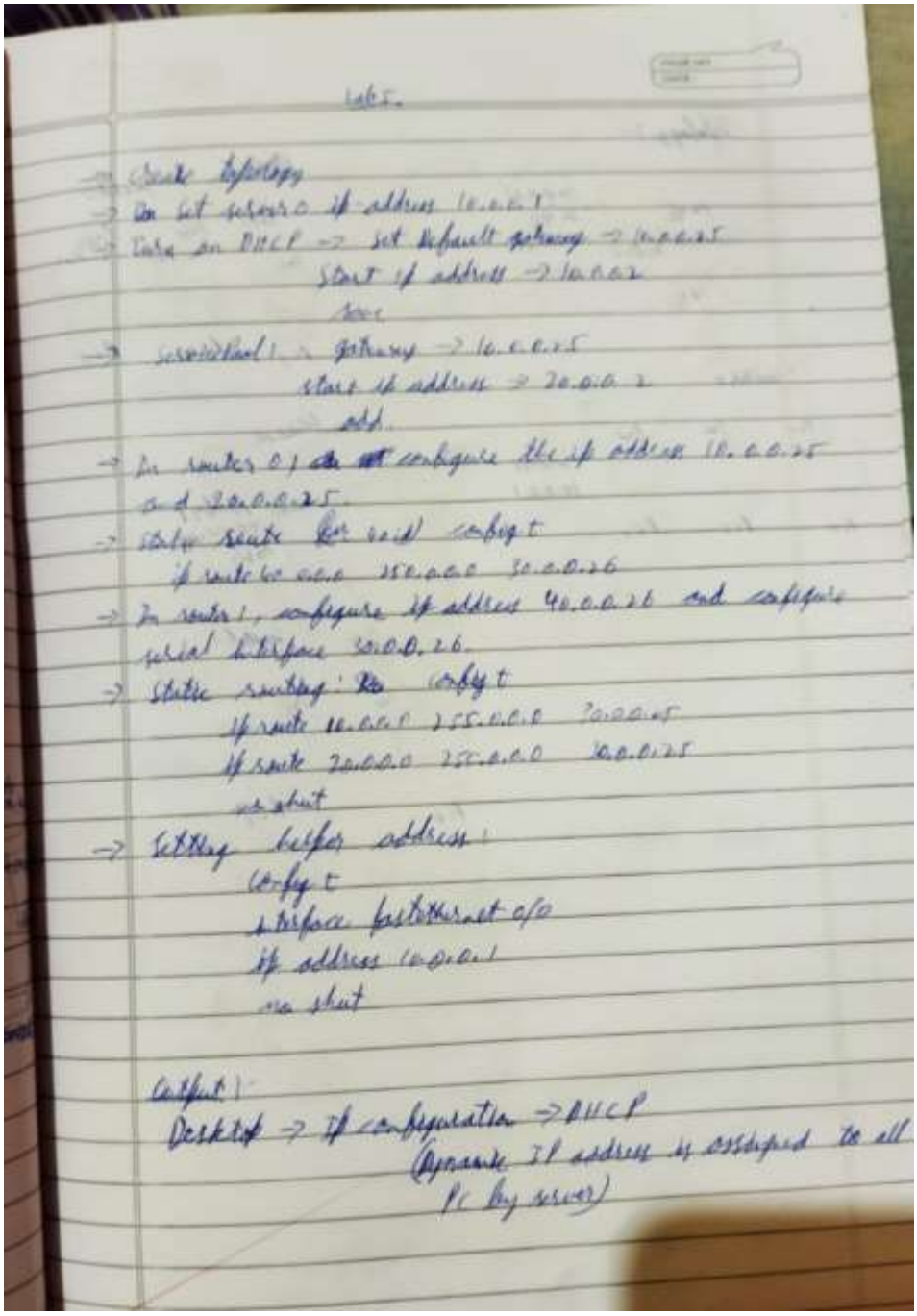
Command Prompt

```
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.

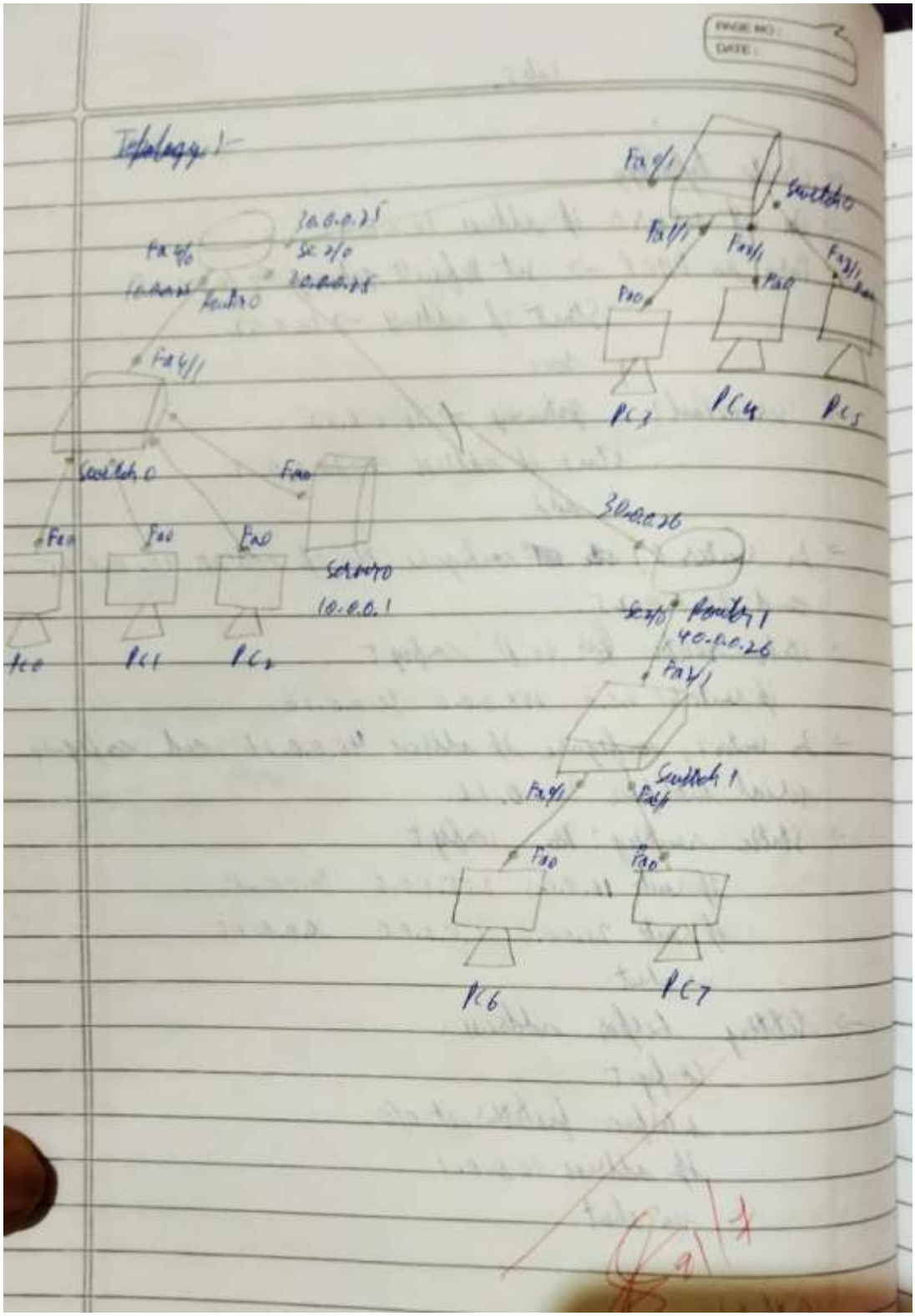
OBSERVATION:



Lab5.

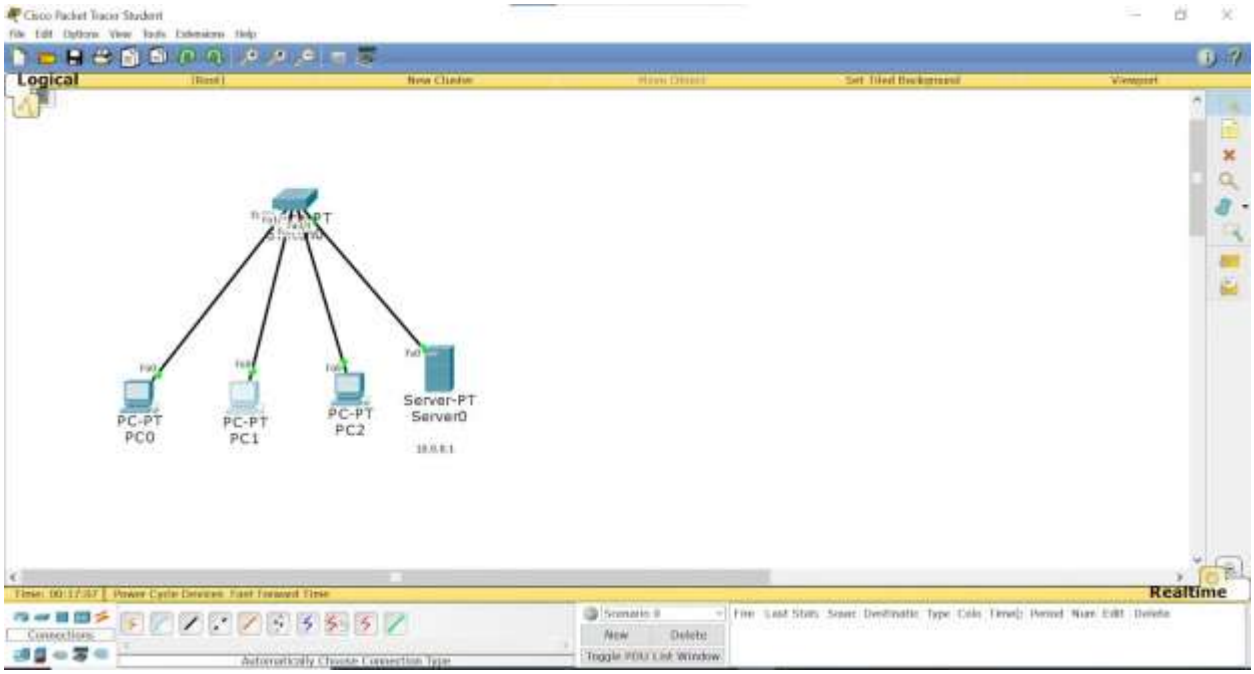
→ Create topology
→ On set server0 ip address 10.0.0.1
→ Take on DHCP → set default gateway → 10.0.0.25
Start ip address → 10.0.0.2
Save
→ server1 → gateway → 10.0.0.25
start ip address → 20.0.0.2
add.
→ In router 0, configure the ip address 10.0.0.25 and 30.0.0.25.
→ Static route for 0 and 1 config t
ip route 60.0.0.0 255.0.0.0 30.0.0.26
→ In router 1, configure ip address 40.0.0.26 and configure serial interface 30.0.0.26.
→ Static routing: 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 address 10.0.0.1
no shut

Output:
Desktop → IP configuration → DHCP
(Dynamic IP address is assigned to all PC by server)

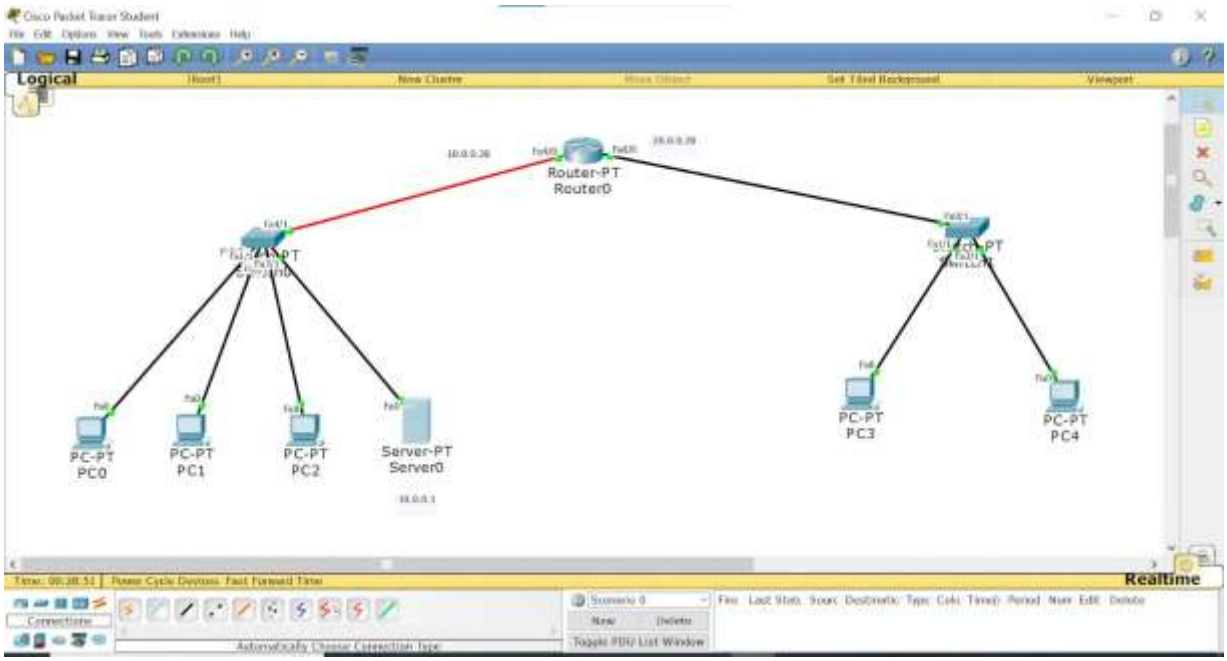

Topology:-
Router0
Fa4/1
Switch0
Fa0
PC0
PC1
PC2
Fa4/1
Switch0
Fa0
PC3
PC4
PC5
Router1
Fa4/1
Switch1
Fa0
Fa0
PC6
PC7

TOPOLOGY:

PROGRAM 4.1:

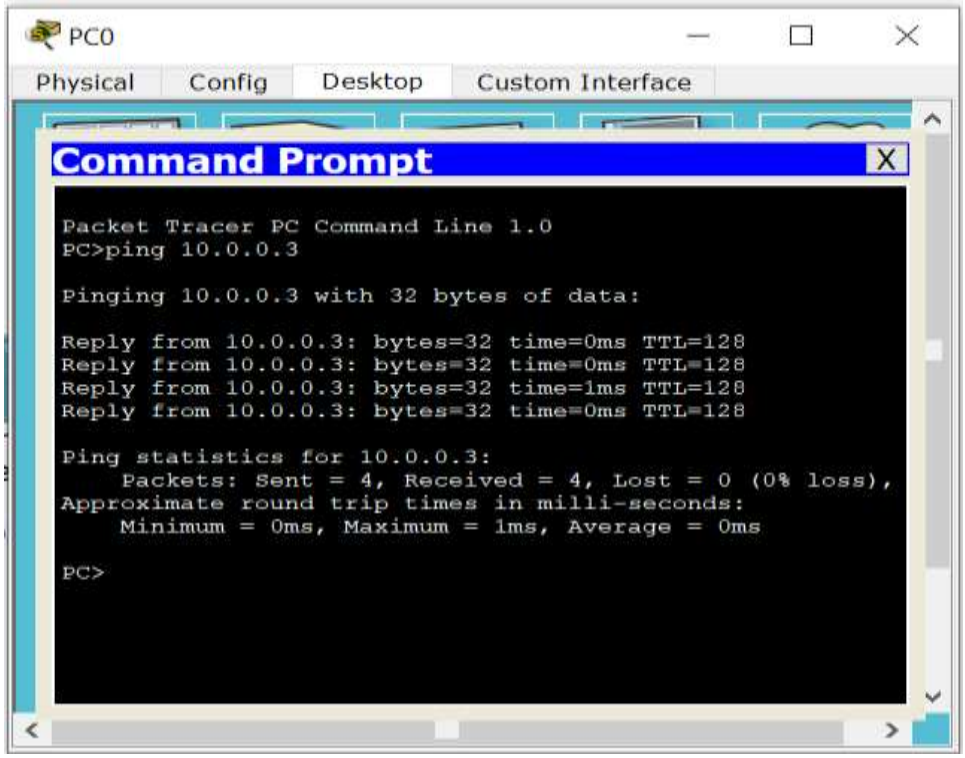


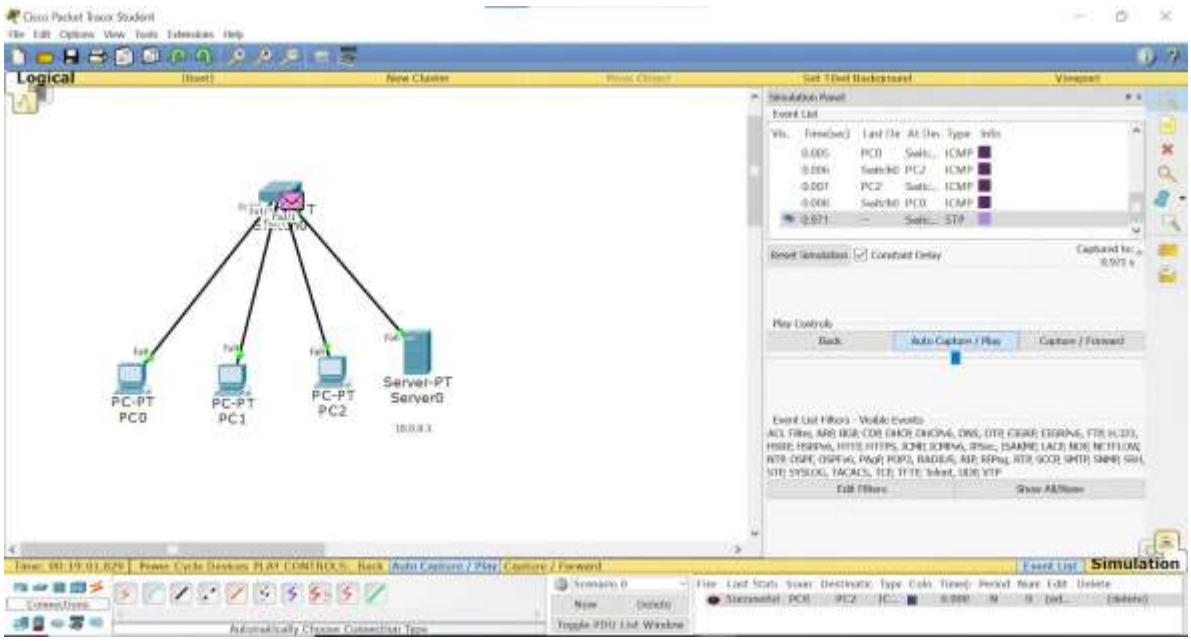
PROGRAM 4.2:



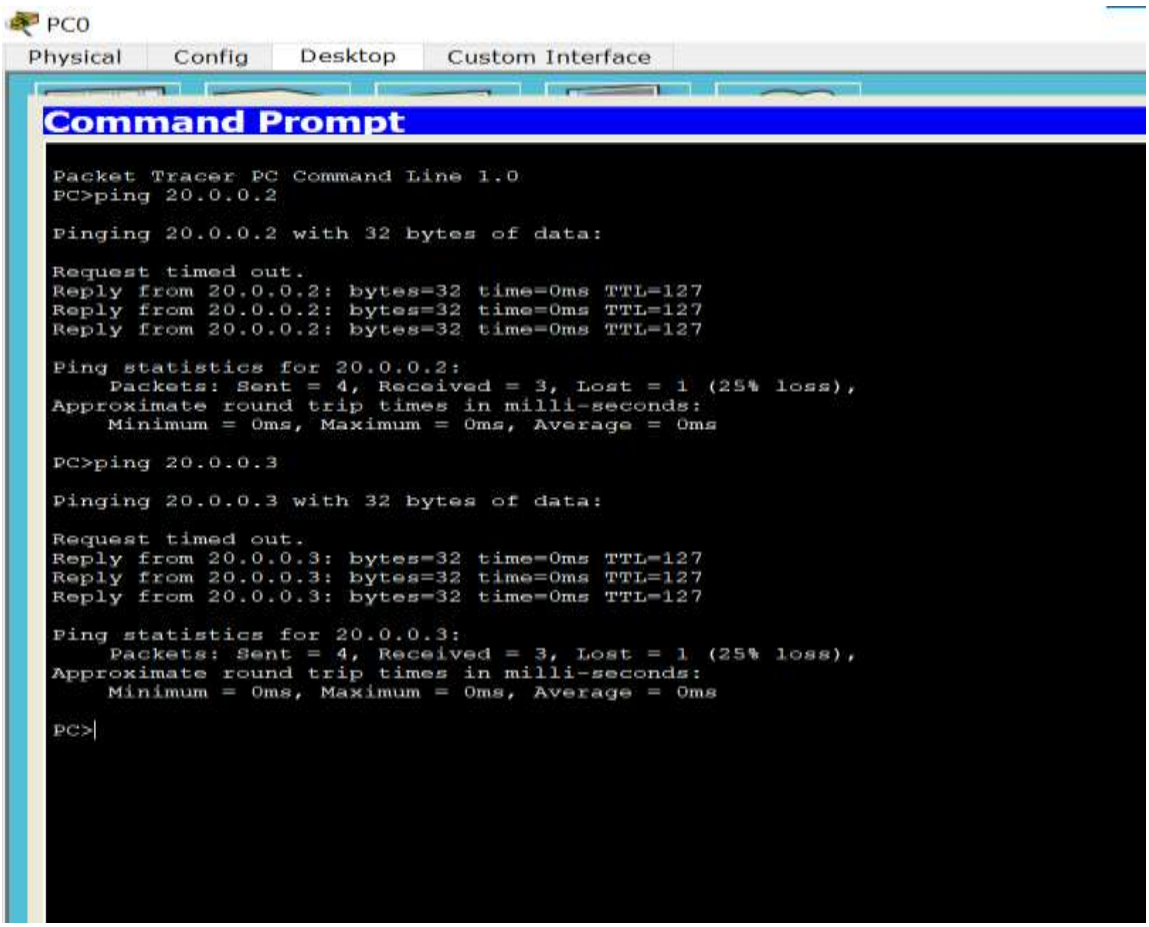
OUTPUT:

PROGRAM 4.1:





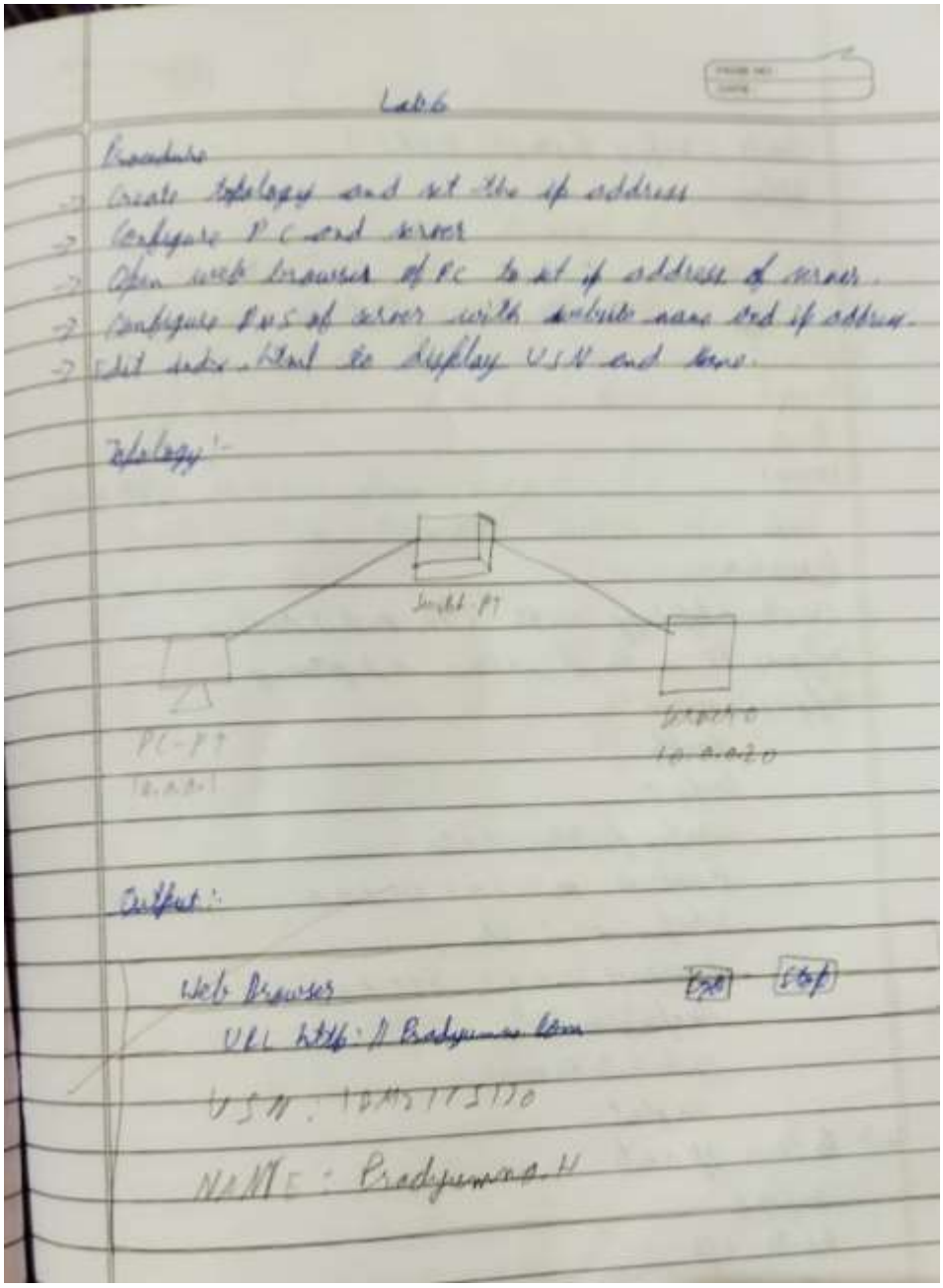
PROGRAM 4.2:


PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 20.0.0.2
Pinging 20.0.0.2 with 32 bytes of data:
Request timed out.
Reply from 20.0.0.2: bytes=32 time=0ms TTL=127
Reply from 20.0.0.2: bytes=32 time=0ms TTL=127
Reply from 20.0.0.2: bytes=32 time=0ms TTL=127
Ping statistics for 20.0.0.2:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
PC>ping 20.0.0.3
Pinging 20.0.0.3 with 32 bytes of data:
Request timed out.
Reply from 20.0.0.3: bytes=32 time=0ms TTL=127
Reply from 20.0.0.3: bytes=32 time=0ms TTL=127
Reply from 20.0.0.3: bytes=32 time=0ms TTL=127
Ping statistics for 20.0.0.3:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
PC>

WEEK 5

Configure Web Server, DNS within a LAN.

OBSERVATION:



Lab 6

Procedure

-> Create topology and set the ip address
-> Configure PC and server
-> Open web browser of PC to set ip address of server.
-> Configure DNS of server with domain name and ip address.
-> Edit index.html to display USN and name.

Topology:-

Switch-PT

PC-PT
10.0.0.1

Server-PT
10.0.0.20

Output:

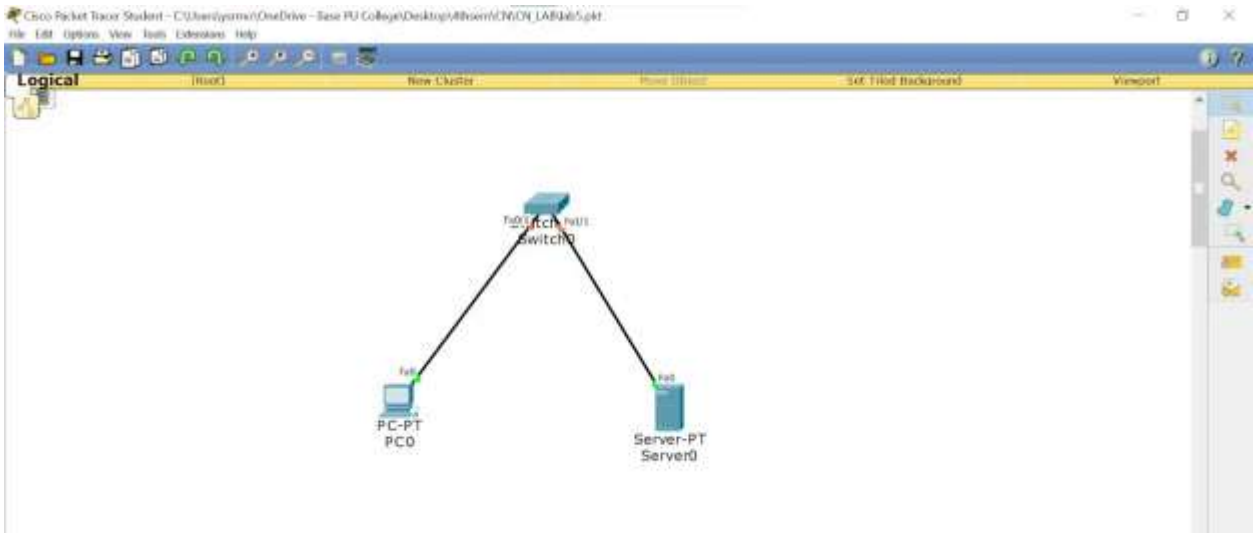
Web Browser

URL http://Prodyumna.com

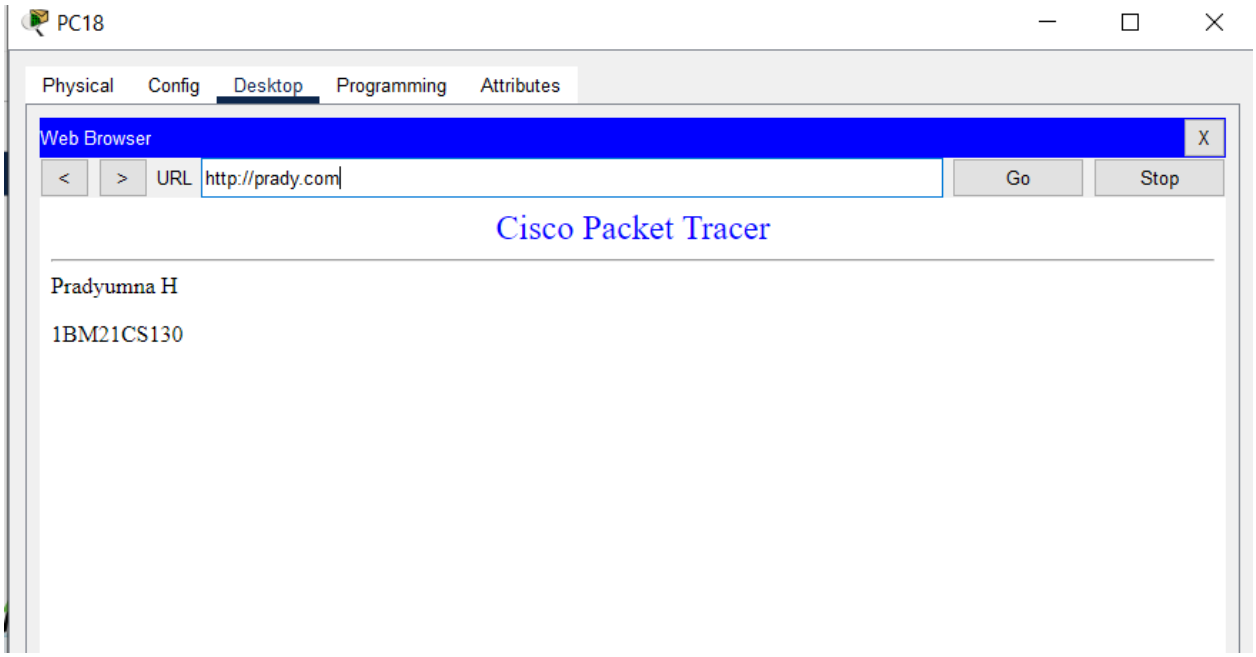
USN: 1DS17IS120

NAME: Prodyumna.H

TOPOLOGY:



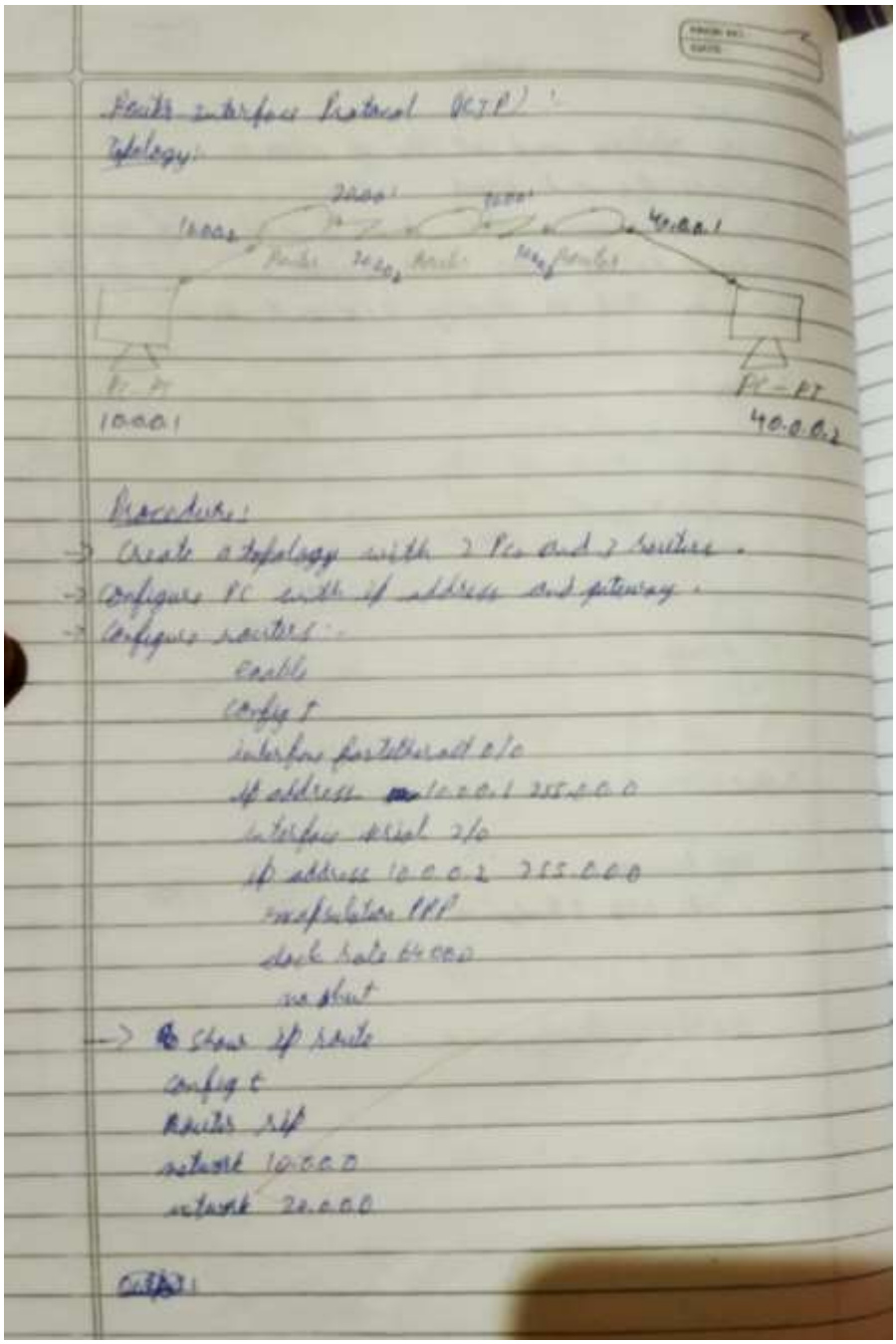
OUTPUT:



WEEK 6

Configure RIP routing Protocol in Routers.

OBSERVATION:



Routing Information Protocol (RIP):

Topology:

Procedure:

-> Create a topology with 2 PCs and 2 routers.
-> Configure PC with IP address and gateway.
-> Configure routers:

```
enable
config t
interface fastethernet 0/0
ip address 10.0.0.1 255.0.0.0
interface serial 2/0
ip address 10.0.0.2 255.0.0.0
encapsulation PPP
clock rate 64000
no shut
```

-> show ip route

```
config t
router rip
network 10.0.0.0
network 20.0.0.0
```

Output:

Output:

Ping 20.0.0.2

Pinging 20.0.0.2 with 32 bytes of data

Request timed out

Reply from 20.0.0.2 bytes=32 time=1ms TTL=128

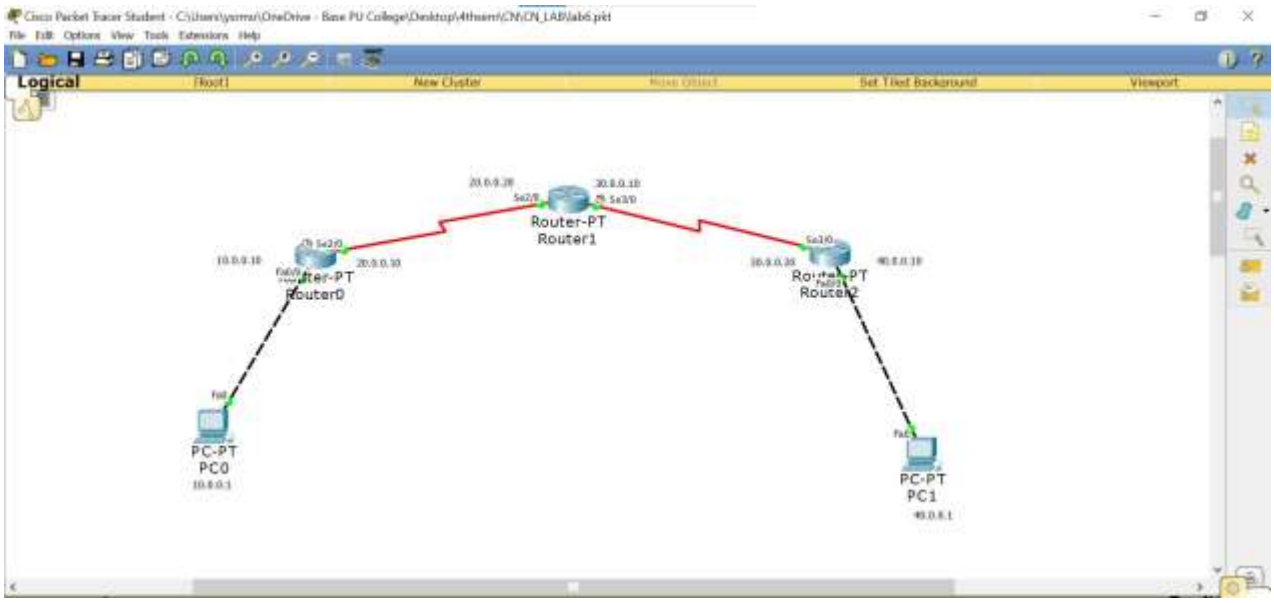
Reply from 20.0.0.2 bytes=32 time=7ms TTL=128

Reply from 20.0.0.2 bytes=32 time=7ms TTL=128

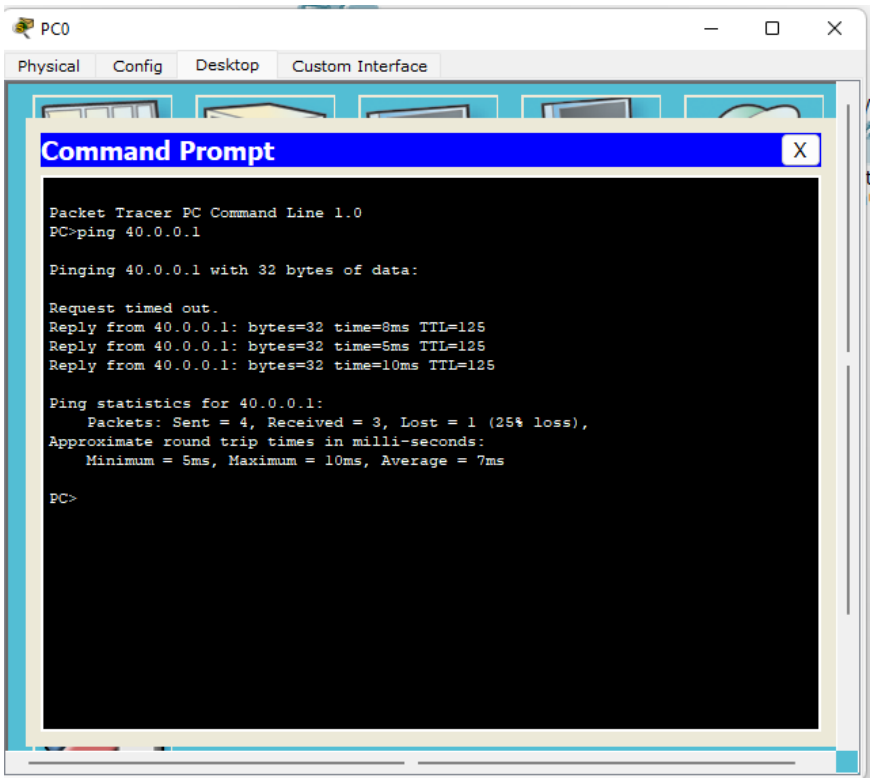
Ping statistics for 20.0.0.2

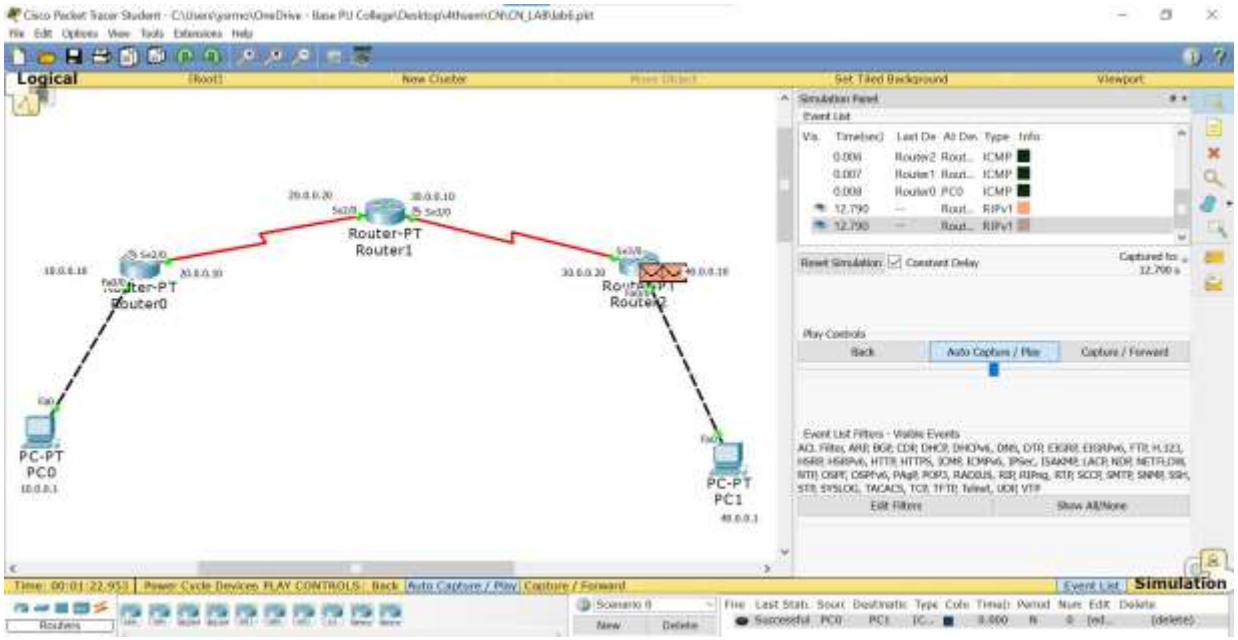
Packets: Sent=4 Received=3 Lost=1 (25% loss)

TOPOLOGY:



OUTPUT:

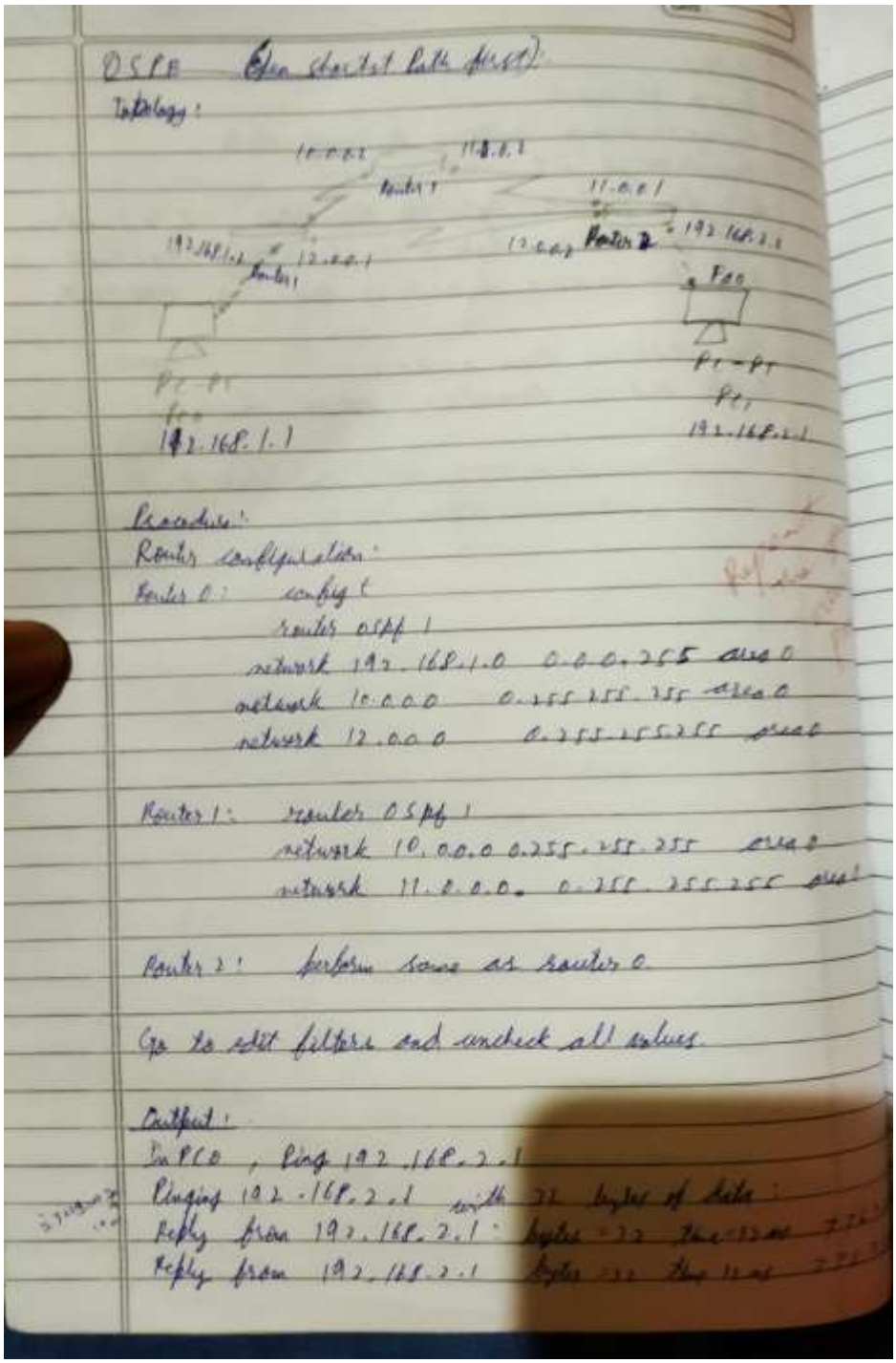



Logical
Router-PT
Router1
Router-PT
Router0
Router-PT
Router2
PC-PT
PC0
PC-PT
PC1
Simulation Panel
Event List
Play Controls
Simulation

WEEK 7

Configure OSPF routing protocol.

OBSERVATION:



OSPF (Open Shortest Path first)

Topology:

Procedure:

Router configuration:

Router 0: config t
router OSPF 1
network 192.168.1.0 0.0.0.255 area 0
network 10.0.0.0 0.255.255.255 area 0
network 12.0.0.0 0.255.255.255 area 0

Router 1: router OSPF 1
network 10.0.0.0 0.255.255.255 area 0
network 11.0.0.0 0.255.255.255 area 0

Router 2: perform same as router 0

Go to edit filters and uncheck all values

Output:

In PC0, Ping 192.168.2.1
Pinging 192.168.2.1 with 32 bytes of data:
Reply from 192.168.2.1: bytes=32 time=12ms TTL=
Reply from 192.168.2.1 bytes=32 time=12ms TTL=

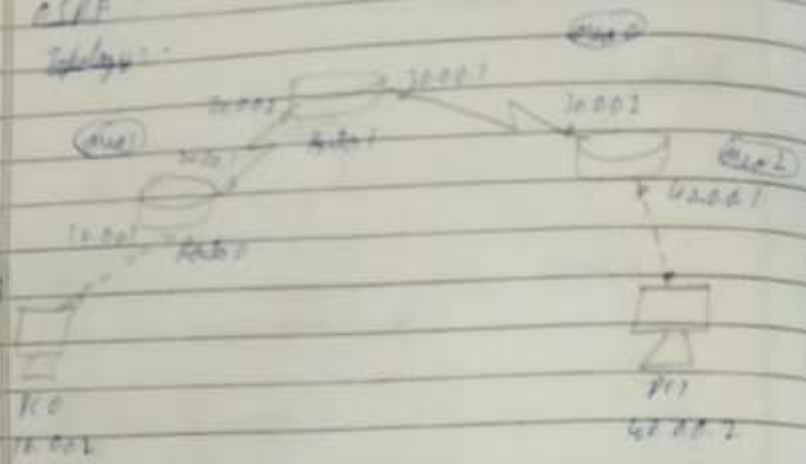
PAGE NO:
DATE:

Reply from 192.168.2.1 bytes=32 time=12ms TTL=126
Reply from 192.168.2.1 bytes=32 time=12ms TTL=126
Ping statistics for 192.168.2.1
Packets sent=4 received=4 lost=0 (0% loss)

Approximate round trip times in milliseconds.
Minimum=4ms maximum=12ms avg=8ms

OSPF

Topology:



Procedure:

-> Router configuration

interface fastethernet 1/0

ip address 10.0.0.1 255.0.0.0

no shut

exit

interface serial 1/0

ip address 20.0.0.1 255.0.0.0

no shut.

Do same for R_2 and R_3

-> Enabling ip routing by configuring OSPF protocol

In R1 -> 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 0

exit

-> configure loopback address to routers

R_1 interface loopback 0

ip add 172.16.1.252 255.255.0.0

no shut

to set for R_1 and R_3

check virtual link

to link 1

router OSPF 1

area 1 virtual link 2.2.2.2

check routing table for R_3

show ip route

R_1 and R_3 get updated about area 2

check routing table for R_2

show ip route

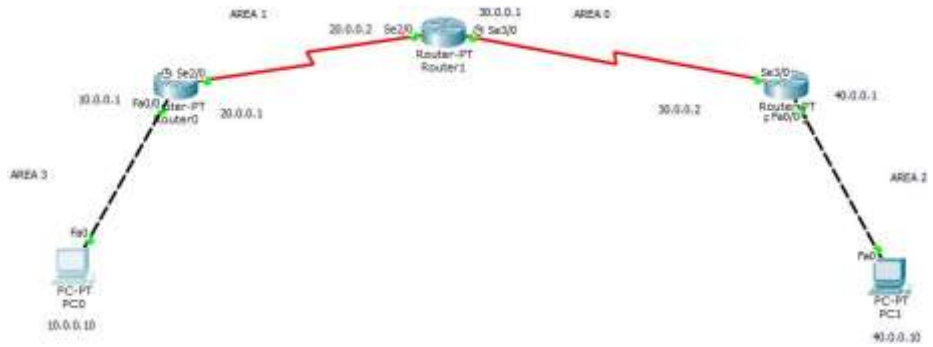
O IA 20.0.0.0/8 via 30.0.0.1

40.0.0.0/8 is directly connected

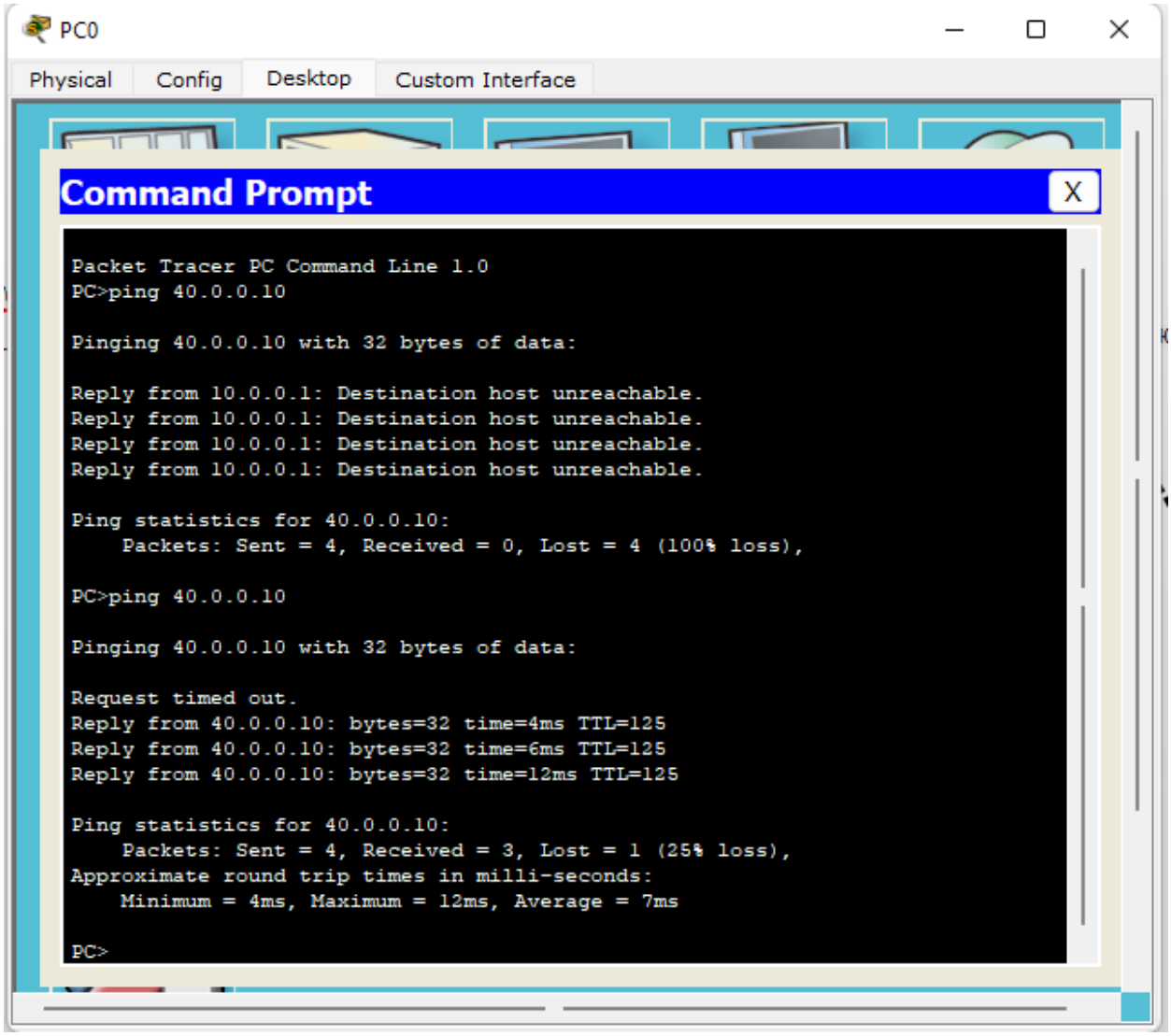
fastEthernet 0/0

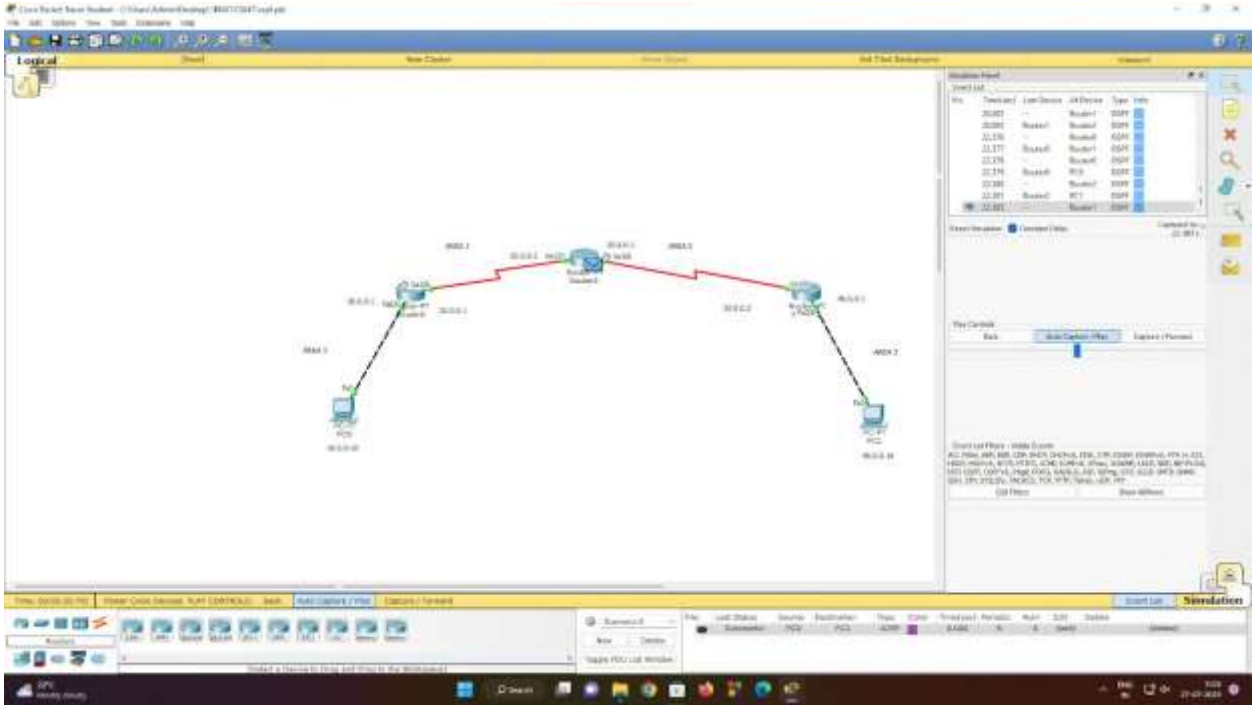
ping 10.0.0.2 to 40.0.0.2

TOPOLOGY:



OUTPUT:

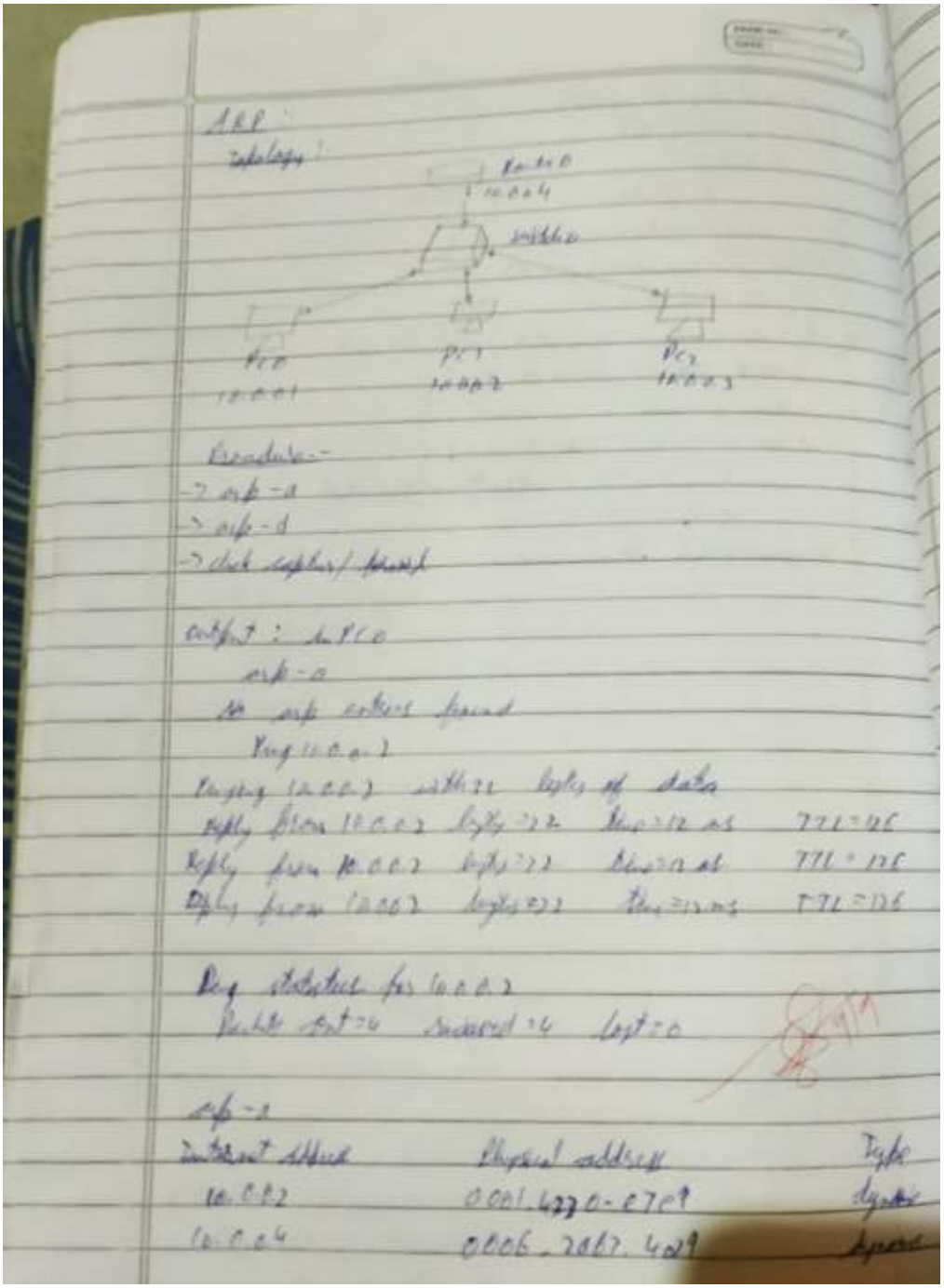



Logical
Simulation

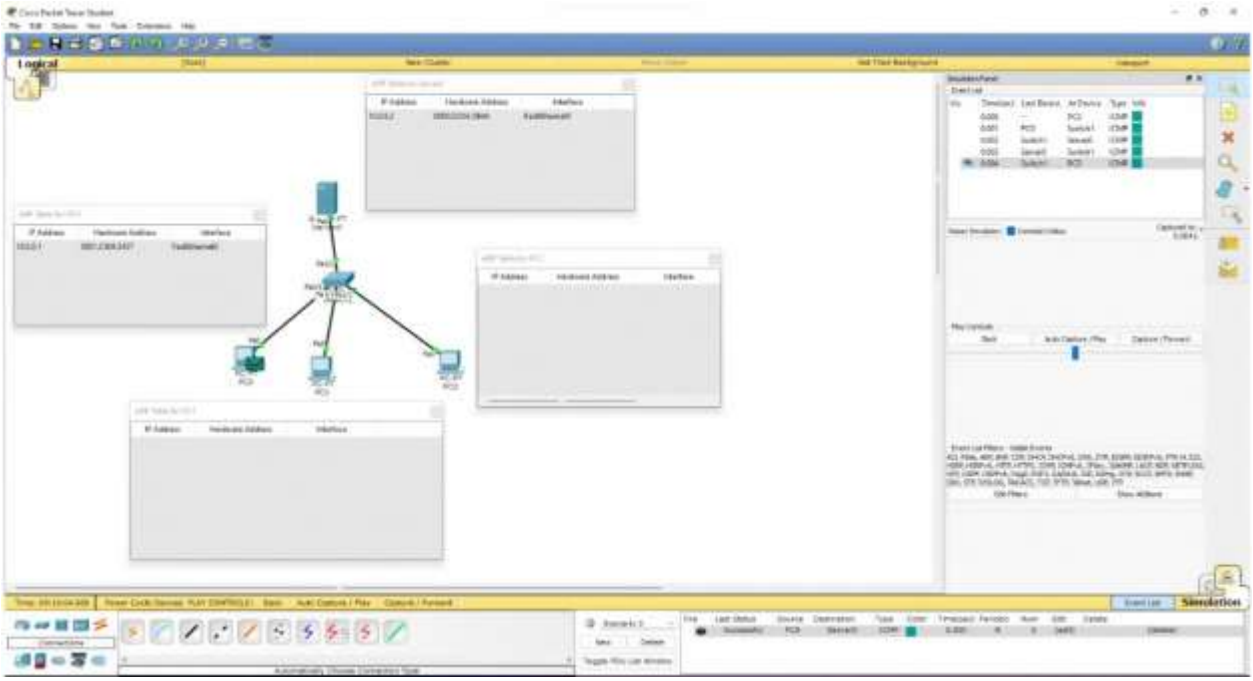
WEEK 8

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

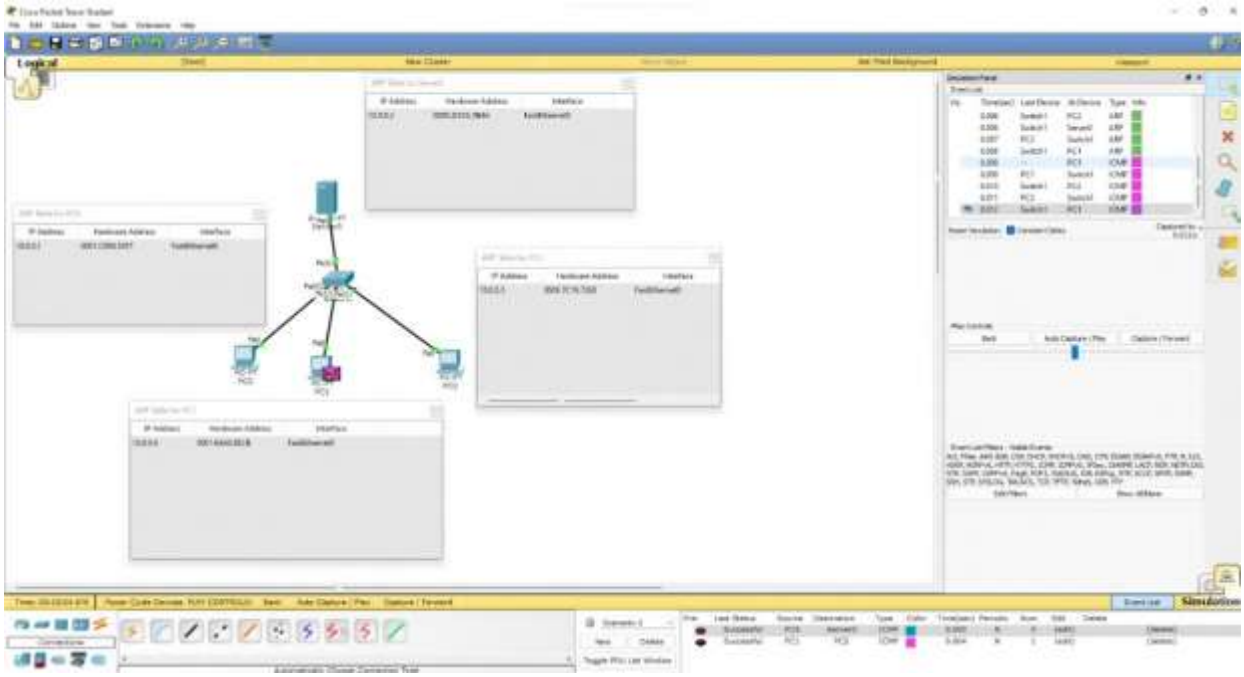
OBSERVATION:

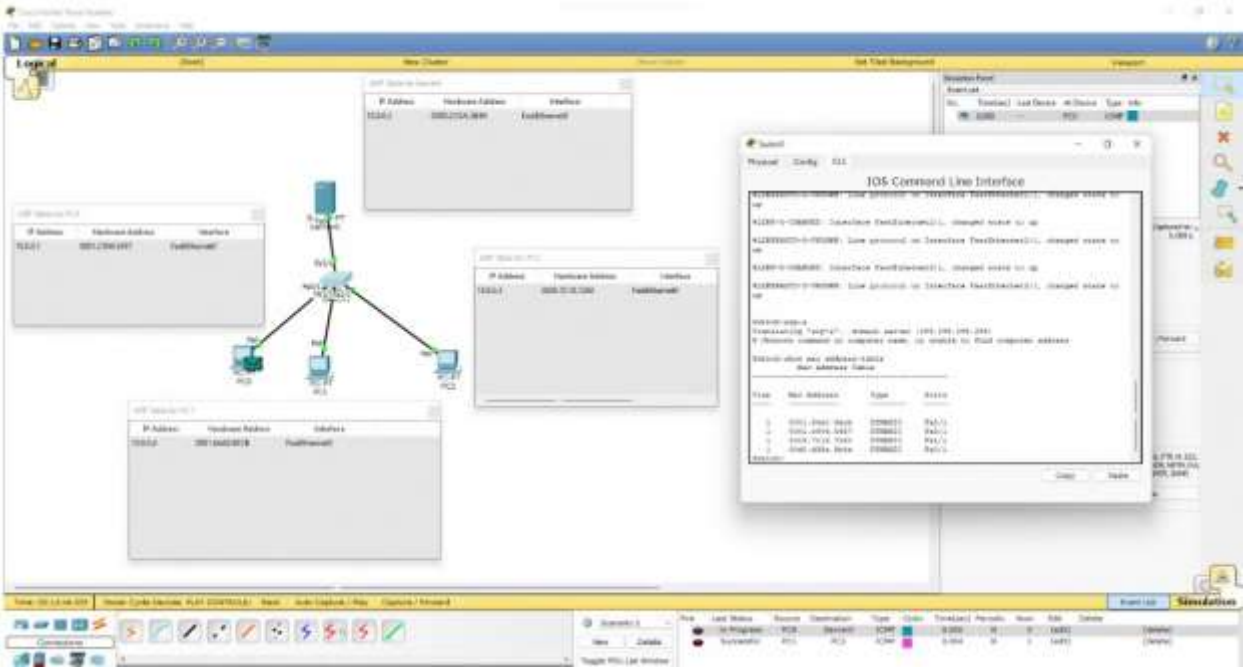


TOPOLOGY:



OUTPUT:

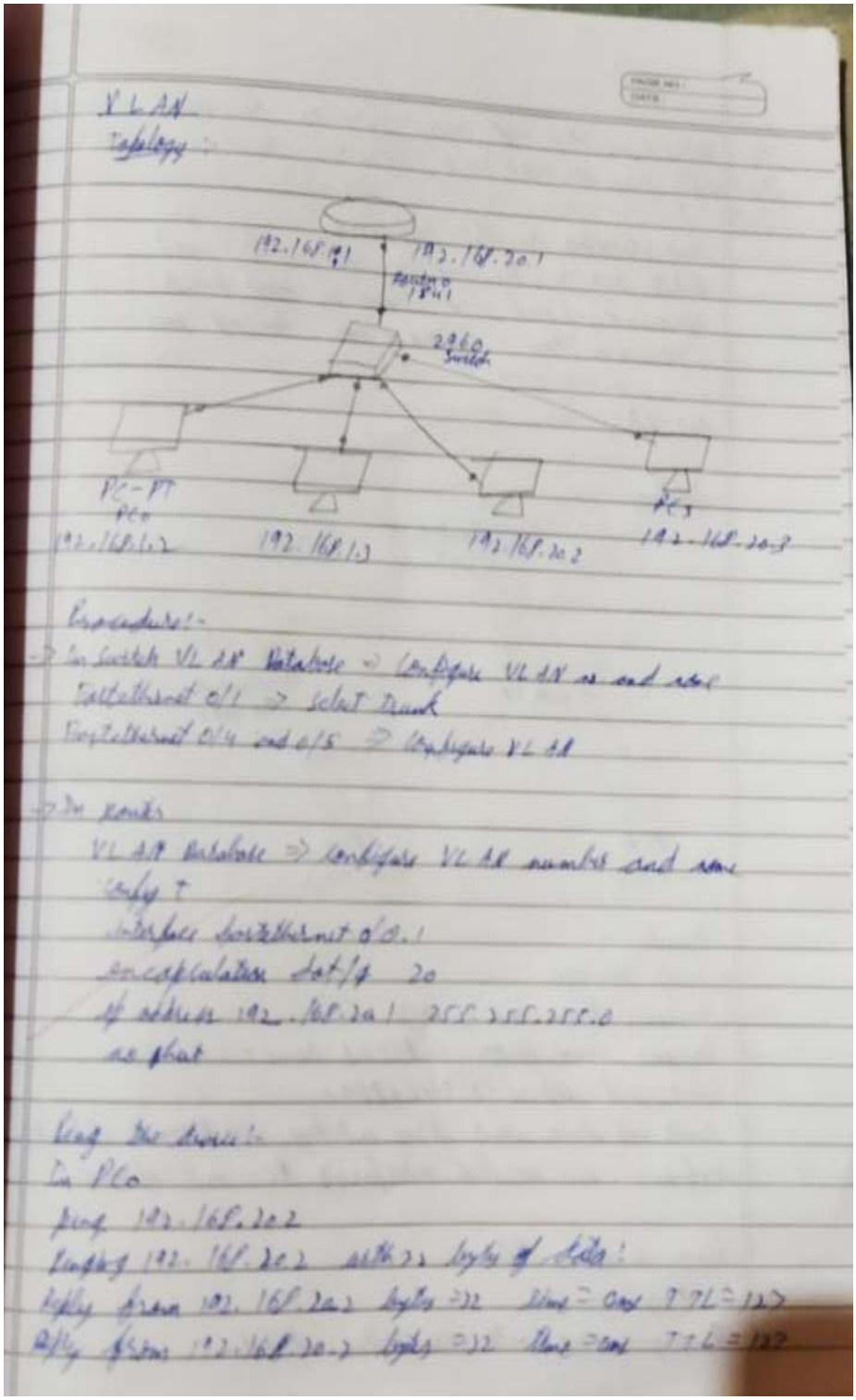


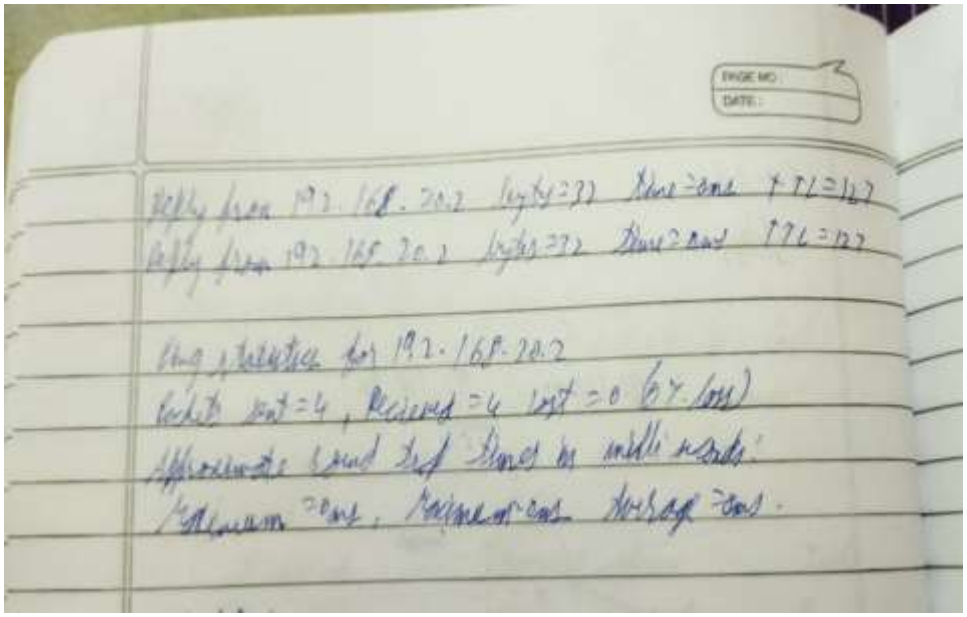

Logical
IOS Command Line Interface

WEEK 9

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

OBSERVATION:

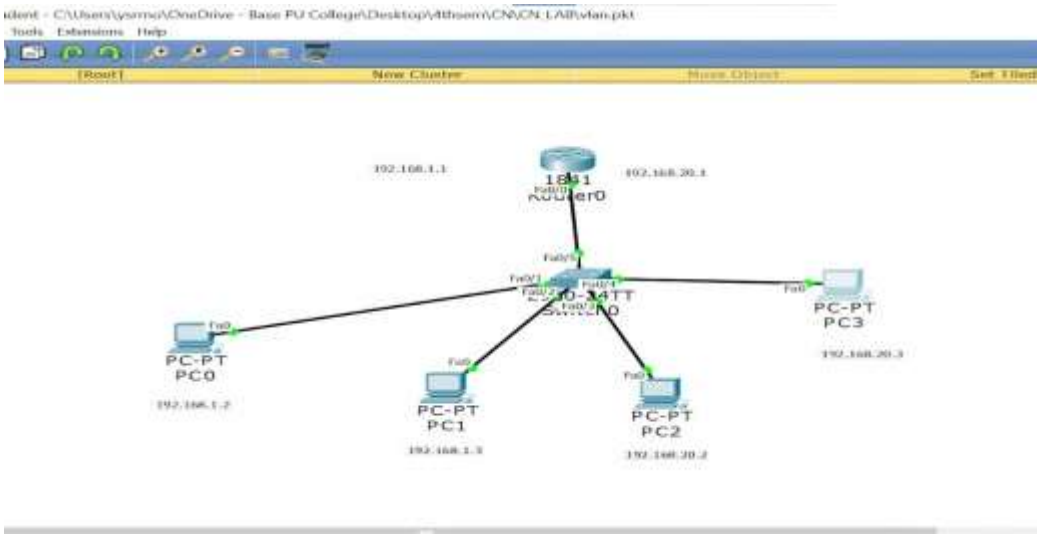




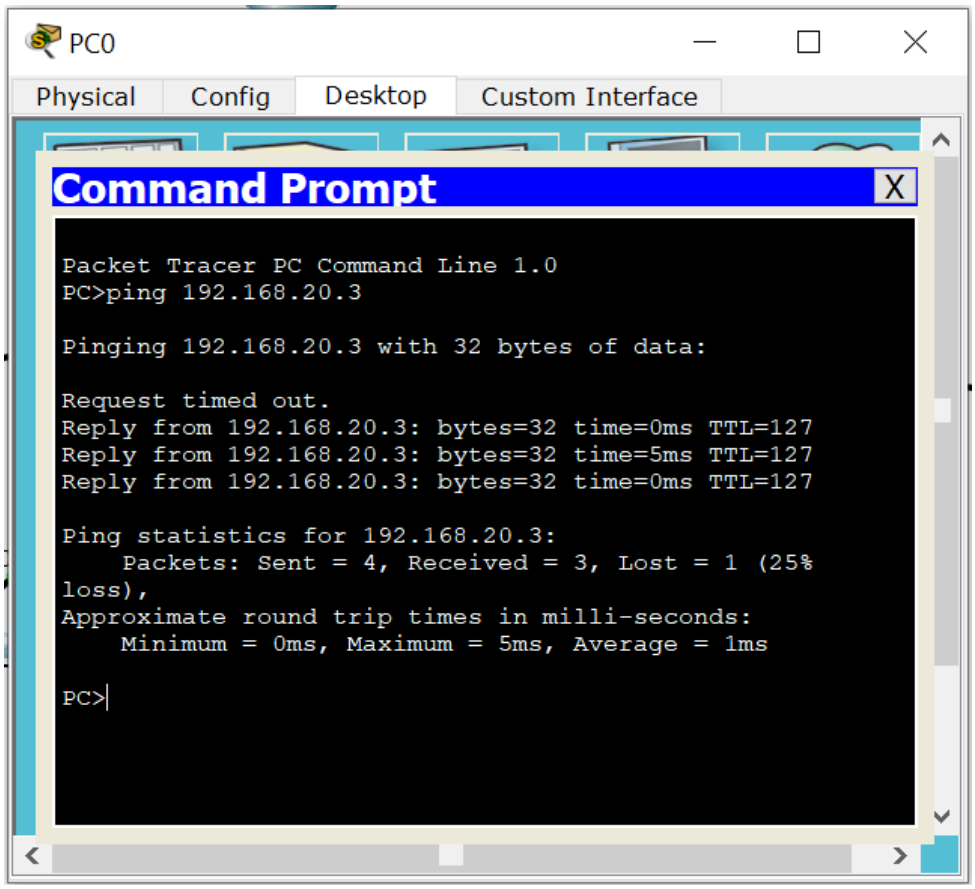
Reply from 192.168.20.2 bytes=32 time=2ms TTL=127
Reply from 192.168.20.2 bytes=32 time=2ms TTL=127

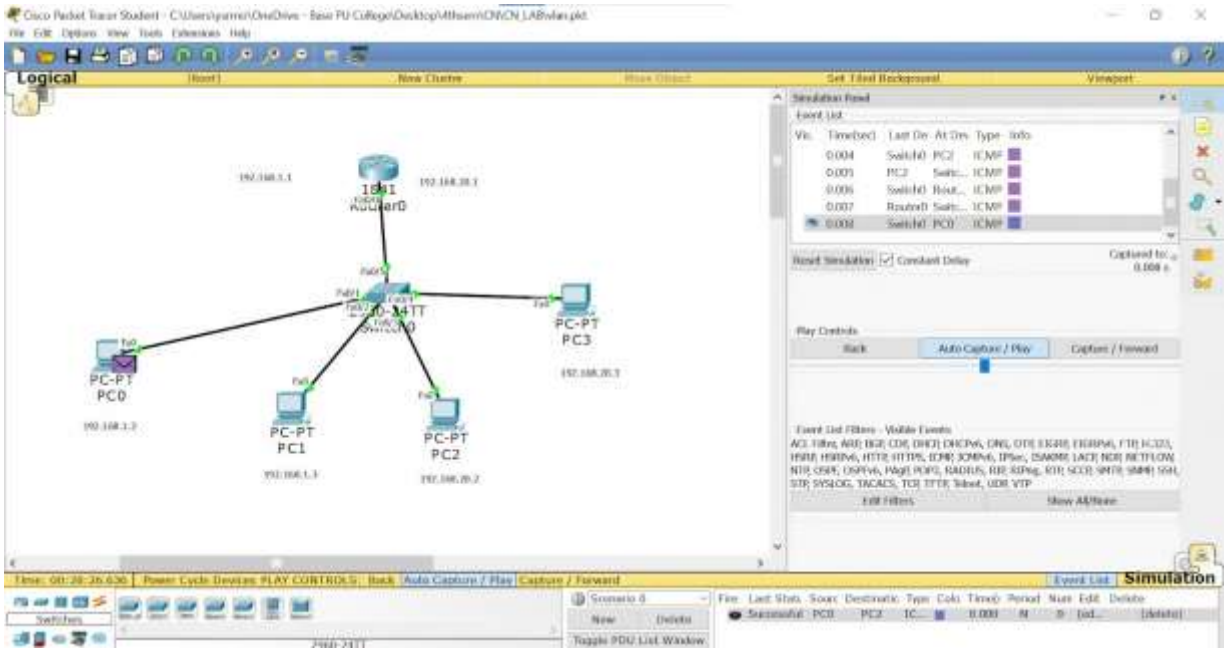
Ping statistics for 192.168.20.2
Packets: Sent=4, Received=4, Lost=0 (0% loss)
Approximate round trip times in milli-seconds:
Minimum=2ms, Maximum=2ms, Average=2ms.

TOPOLOGY:


Tools Extensions Help
[Root]
New Cluster
192.168.1.1
1841
Router0
192.168.20.1
2960-24TT
Switch0
PC-PT
PC0
192.168.1.2
PC-PT
PC1
192.168.1.3
PC-PT
PC2
192.168.20.2
PC-PT
PC3
192.168.20.3

OUTPUT:

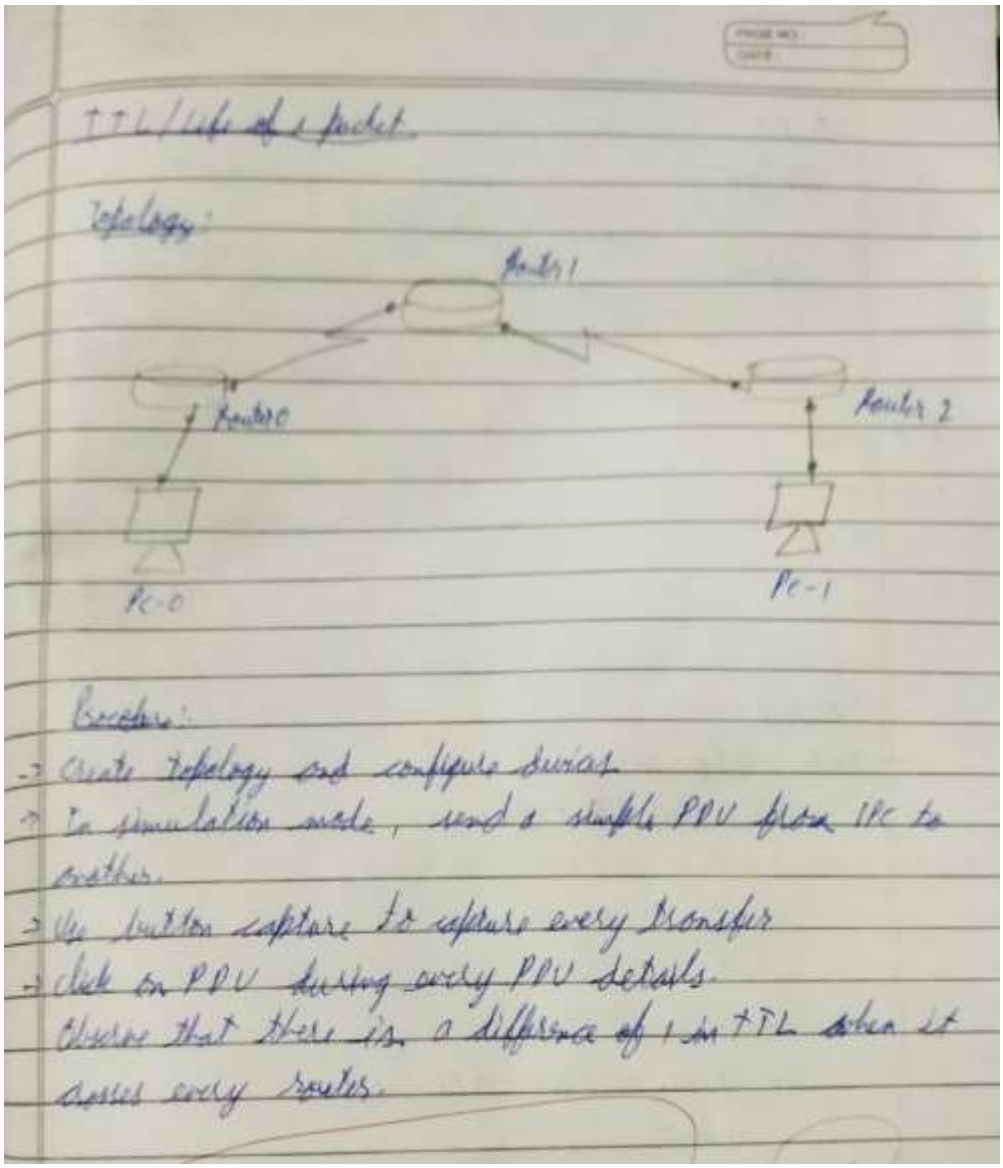

PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 192.168.20.3
Pinging 192.168.20.3 with 32 bytes of data:
Request timed out.
Reply from 192.168.20.3: bytes=32 time=0ms TTL=127
Reply from 192.168.20.3: bytes=32 time=5ms TTL=127
Reply from 192.168.20.3: 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>


Logical
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC3
Simulation

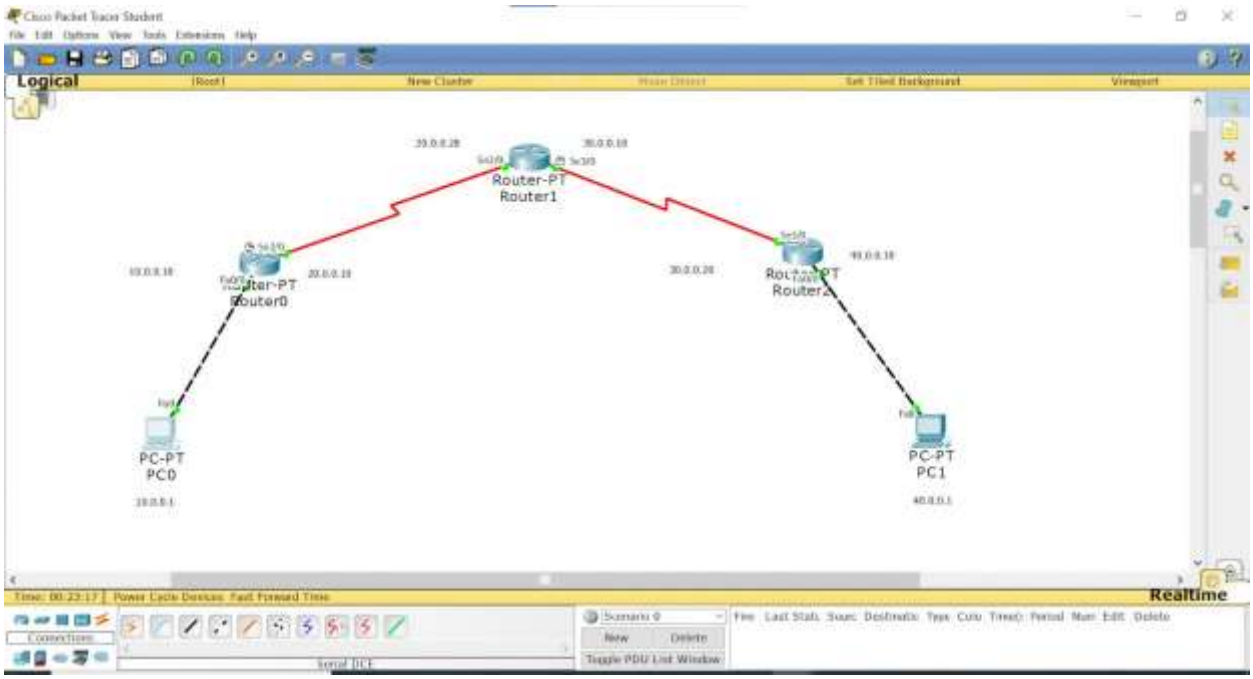
WEEK 10

Demonstrate the TTL/ Life of a Packet.

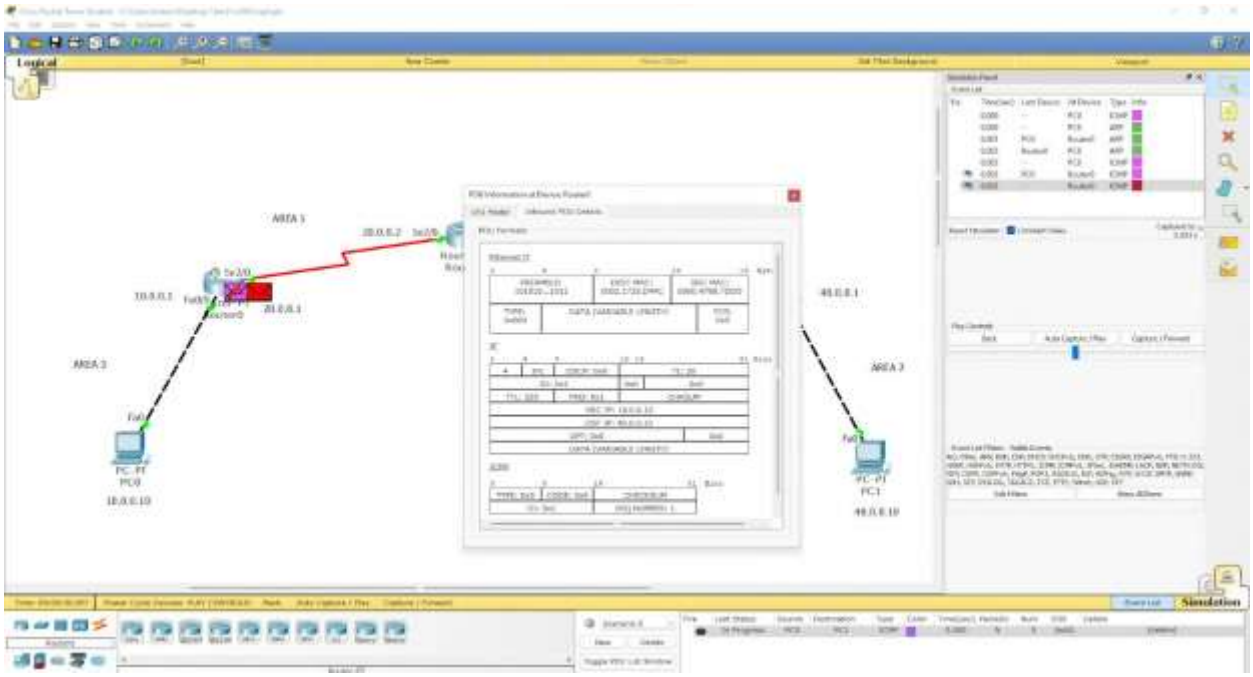
OBSERVATION:

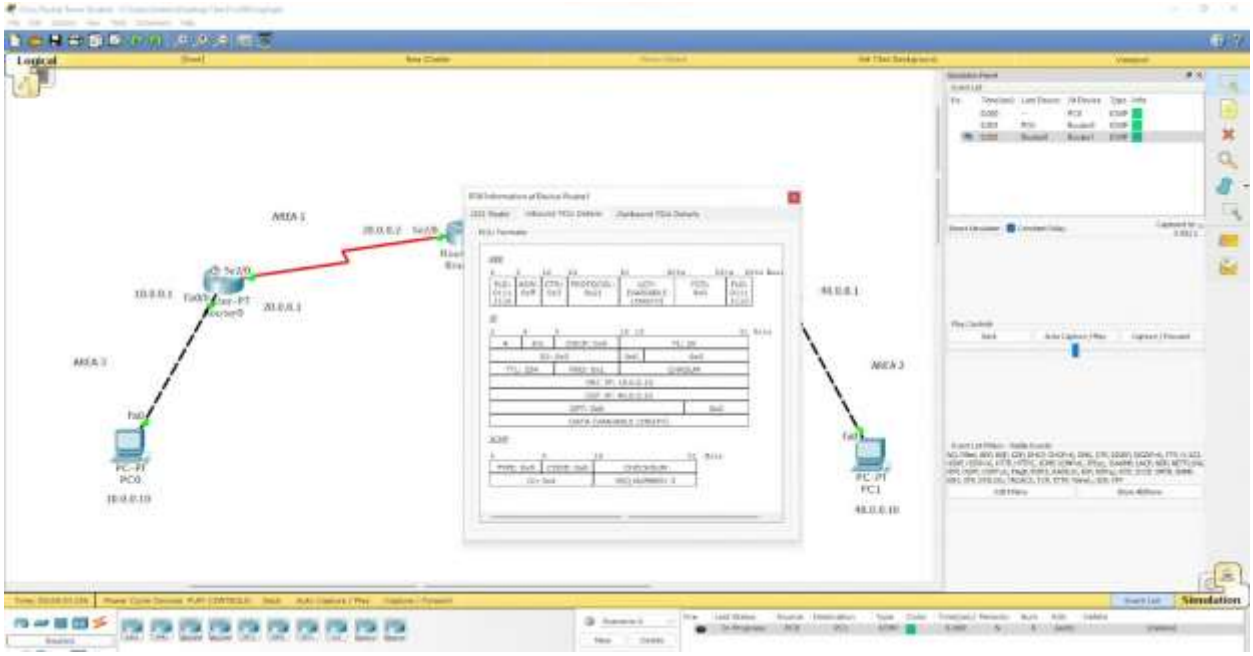


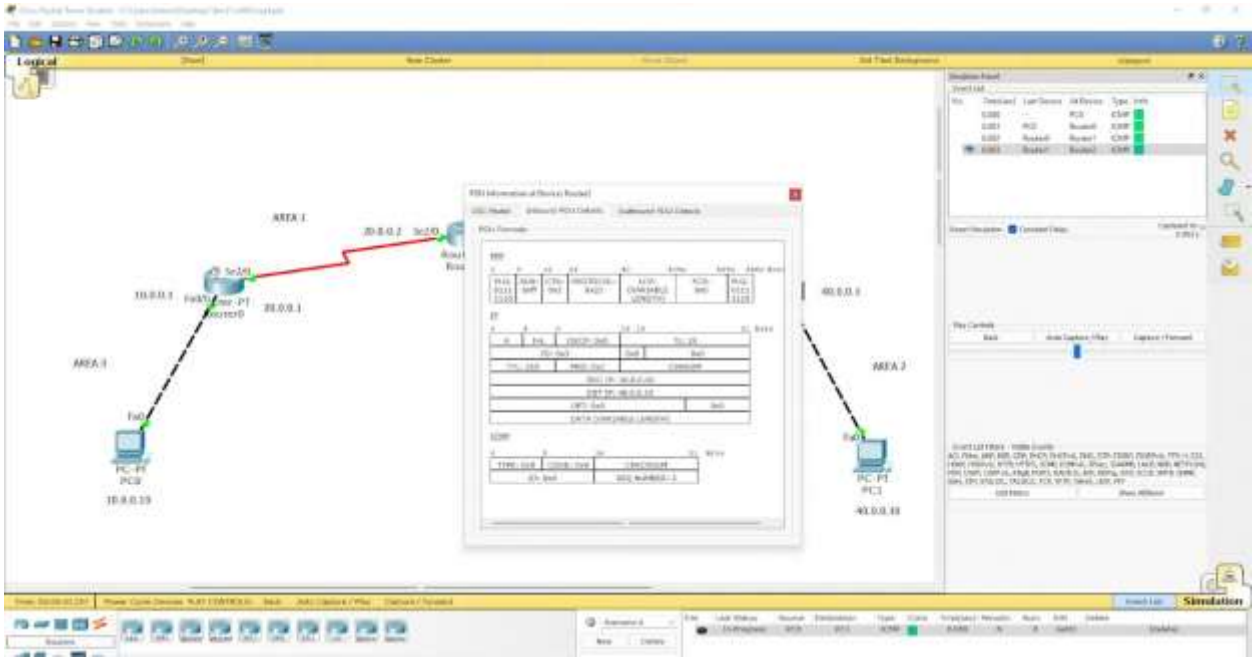
TOPOLOGY:

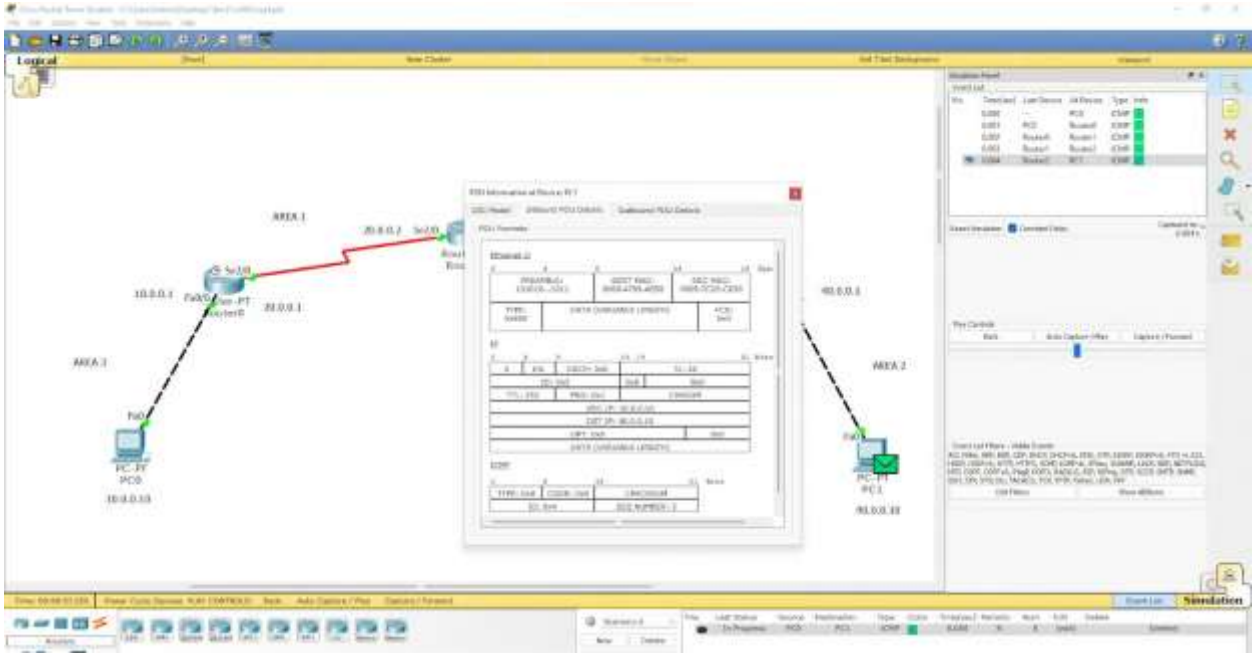


OUTPUT:




Logical
AREA 1
AREA 2
AREA 3
PC-PT
PC0
PC-PT
PC1
Simulation

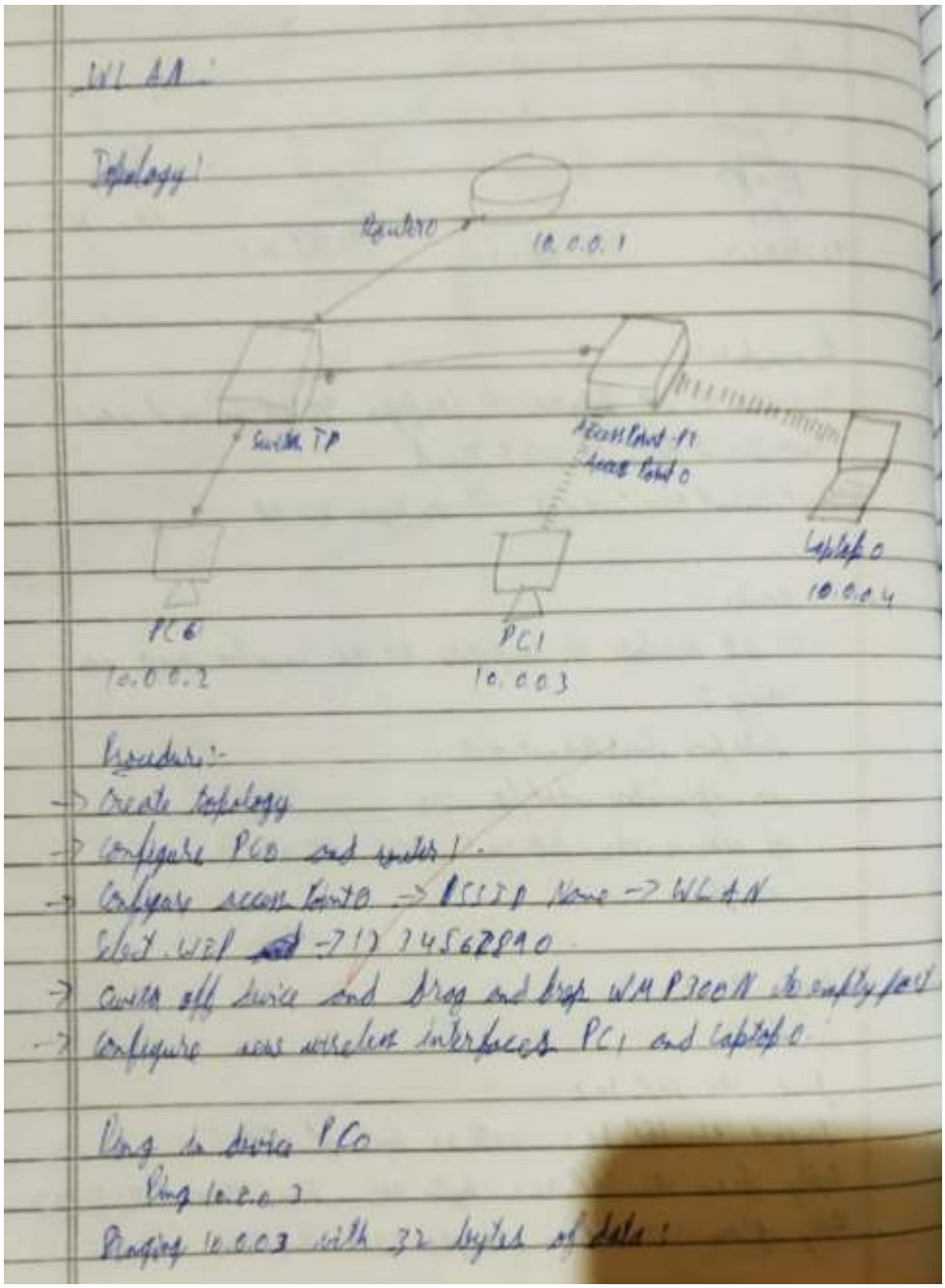

Logical
AREA 1
AREA 2
AREA 3
PC-PT
PC0
PC-PT
PC1
Simulation


Logical
AREA 1
AREA 2
Simulation

WEEK 11

To construct a WLAN and make the nodes communicate wirelessly

OBSERVATION:

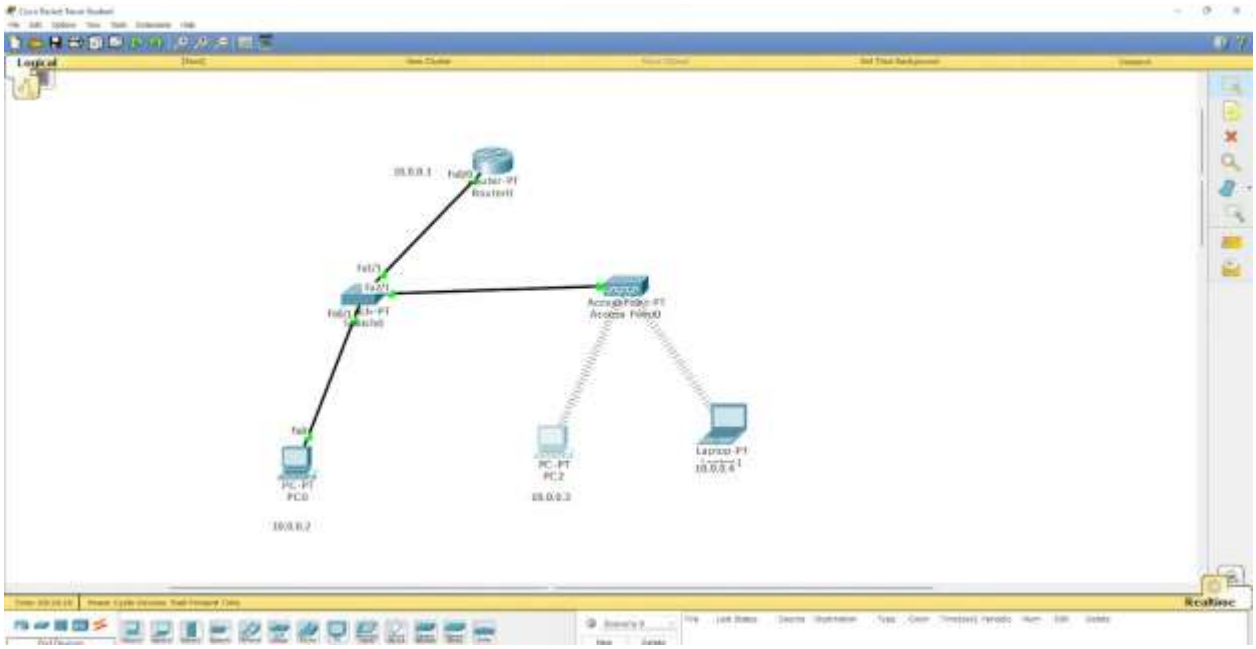


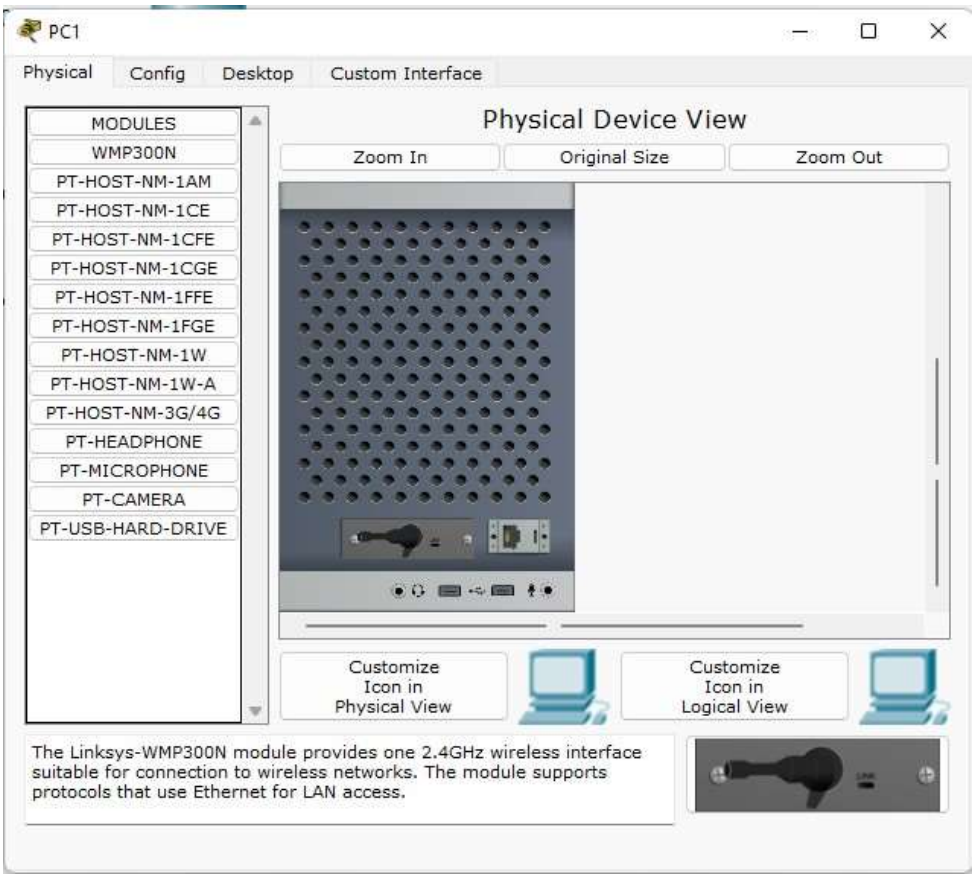
Reply from 10.0.0.2: bytes=32 time=21ms TTL=128
Reply from 10.0.0.2: bytes=32 time=1ms TTL=128
Reply from 10.0.0.2: bytes=32 time=17ms TTL=128
Reply from 10.0.0.2: bytes=32 time=1ms TTL=128

Ping statistics for 10.0.0.2:
Packets: sent=4 received=4 lost=0 (0% loss)
Approximate round trip time in milli-seconds:
minimum=1ms maximum=21ms Average=17ms

18

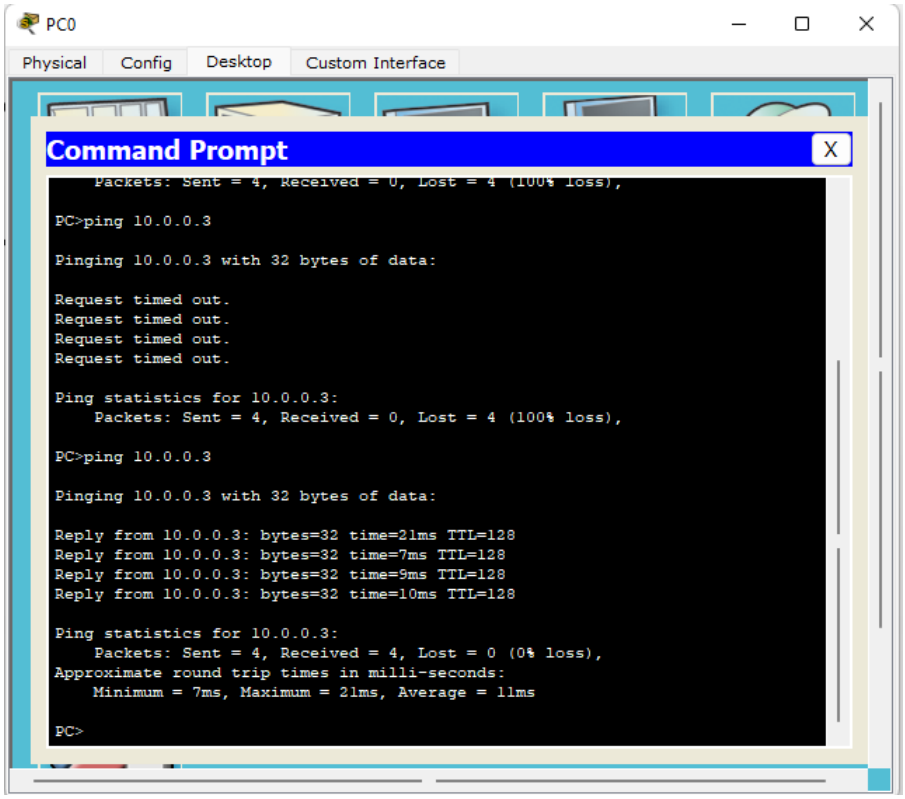
TOPOLOGY:




PC1
Physical
Config
Desktop
Custom Interface
MODULES
WMP300N
PT-HOST-NM-1AM
PT-HOST-NM-1CE
PT-HOST-NM-1CFE
PT-HOST-NM-1CGE
PT-HOST-NM-1FFE
PT-HOST-NM-1FGE
PT-HOST-NM-1W
PT-HOST-NM-1W-A
PT-HOST-NM-3G/4G
PT-HEADPHONE
PT-MICROPHONE
PT-CAMERA
PT-USB-HARD-DRIVE
Physical Device View
Zoom In
Original Size
Zoom Out
Customize Icon in Physical View
Customize Icon in Logical View
The Linksys-WMP300N module provides one 2.4GHz wireless interface suitable for connection to wireless networks. The module supports protocols that use Ethernet for LAN access.


Laptop0
Physical
Config
Desktop
Custom Interface
MODULES
WPC300N
PT-LAPTOP-NM-1AM
PT-LAPTOP-NM-1CE
PT-LAPTOP-NM-1CFE
PT-LAPTOP-NM-1CGE
PT-LAPTOP-NM-1FFE
PT-LAPTOP-NM-1FGE
PT-LAPTOP-NM-1W
PT-LAPTOP-NM-1W-A
PT-LAPTOP-NM-3G/4G
PT-HEADPHONE
PT-MICROPHONE
PT-CAMERA
PT-USB-HARD-DRIVE
Physical Device View
Zoom In
Original Size
Zoom Out
Customize Icon in Physical View
Customize Icon in Logical View
The Linksys-WPC300N module provides one 2.4GHz wireless interface suitable for connection to wireless networks. The module supports protocols that use Ethernet for LAN access.

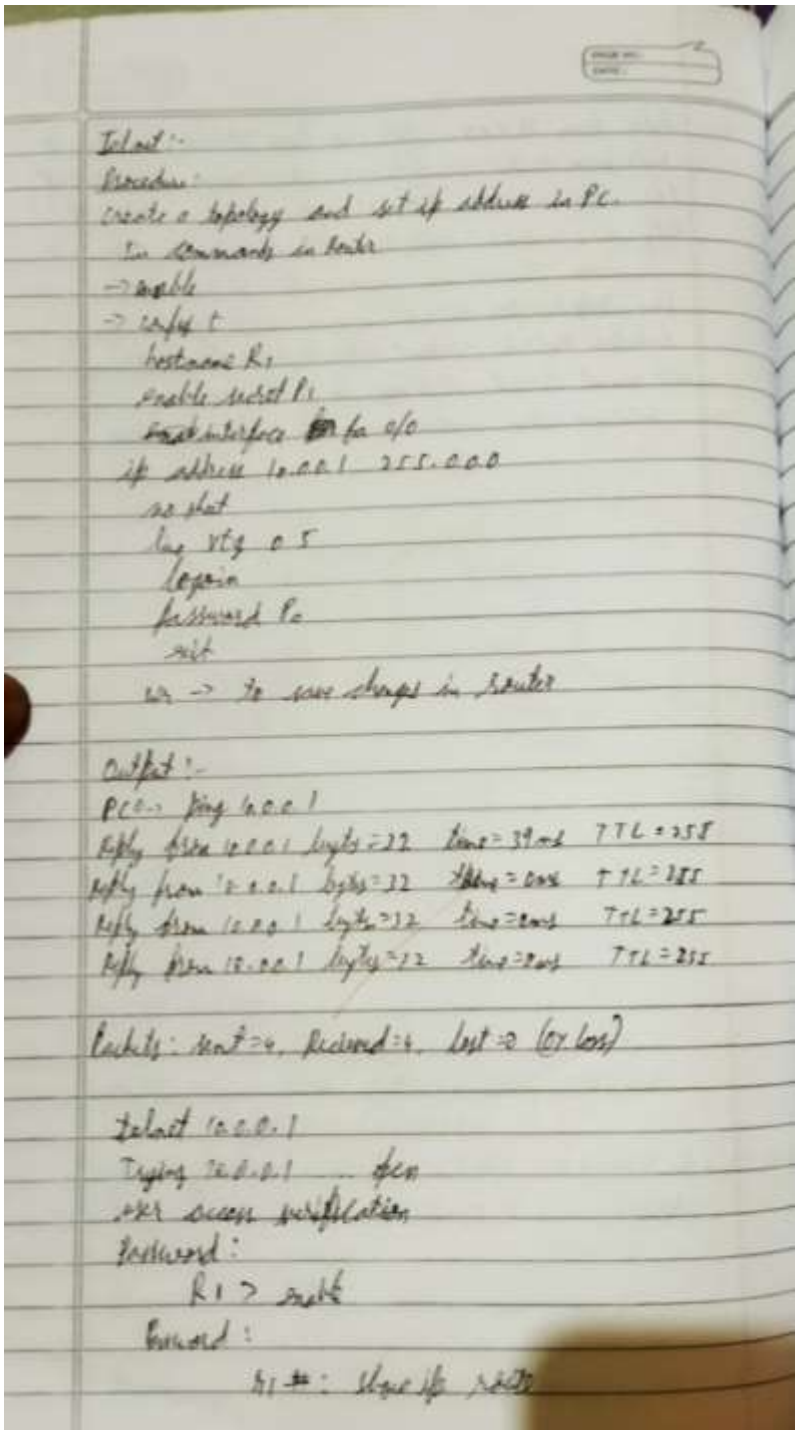
OUTPUT:


PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>ping 10.0.0.3
Pinging 10.0.0.3 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 10.0.0.3:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
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=21ms TTL=128
Reply from 10.0.0.3: bytes=32 time=7ms TTL=128
Reply from 10.0.0.3: bytes=32 time=9ms TTL=128
Reply from 10.0.0.3: bytes=32 time=10ms 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 = 7ms, Maximum = 21ms, Average = 11ms
PC>

WEEK 12

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

OBSERVATION:



Telnet:-

Procedure:

Create a topology and set up address in PC.

In commands in Router

-> enable

-> config t

hostname R1

enable secret P1

~~Fastethernet~~ interface fa 0/0

ip address 10.0.0.1 255.0.0.0

no shut

line vty 0 5

login

password P0

exit

wr -> to save changes in Router

Output:-

PC0:- ping 10.0.0.1

Reply from 10.0.0.1 bytes=32 time=39ms 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

Packets: sent=4, Received=4, lost=0 (0% loss)

telnet 10.0.0.1

Trying 10.0.0.1 ... Open

User access verification

Password:

R1> enable

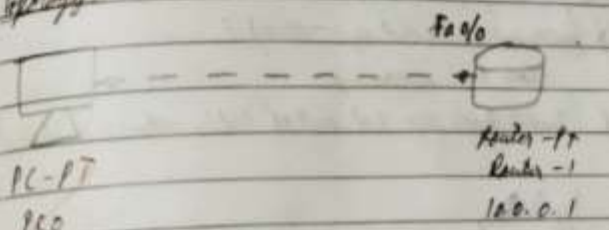
Password:

R1#: show ip route

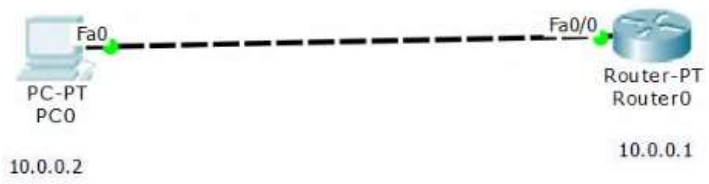
Codes: C - connected, S - static, I - IGRP, R - RIP,
M - mobile, B - BGP, D - EIGRP, EX - EIGRP, O - OSPF,
IA - OSPF inter area
N1 - OSPF NSSA external type.
area

* - candidate default
P - periodic download static route
Gateway of last resort is not set
C: 10.0.0.0/8 is directly connected Fa 0/0

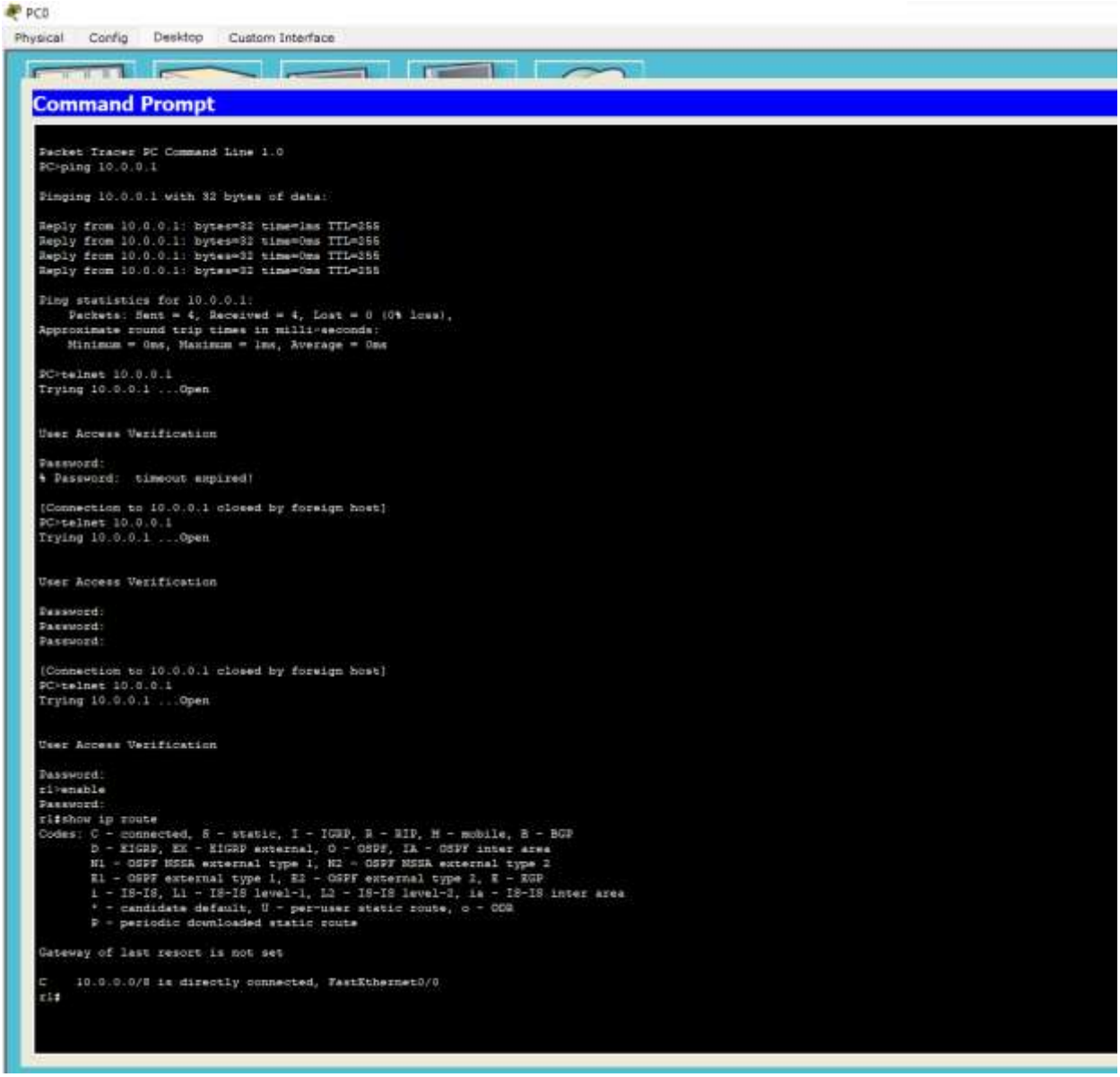
Topology:



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, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
P - periodic downloaded static route
Gateway of last resort is not set
C 10.0.0.0/8 is directly connected, FastEthernet0/0
r1#

WEEK 13

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

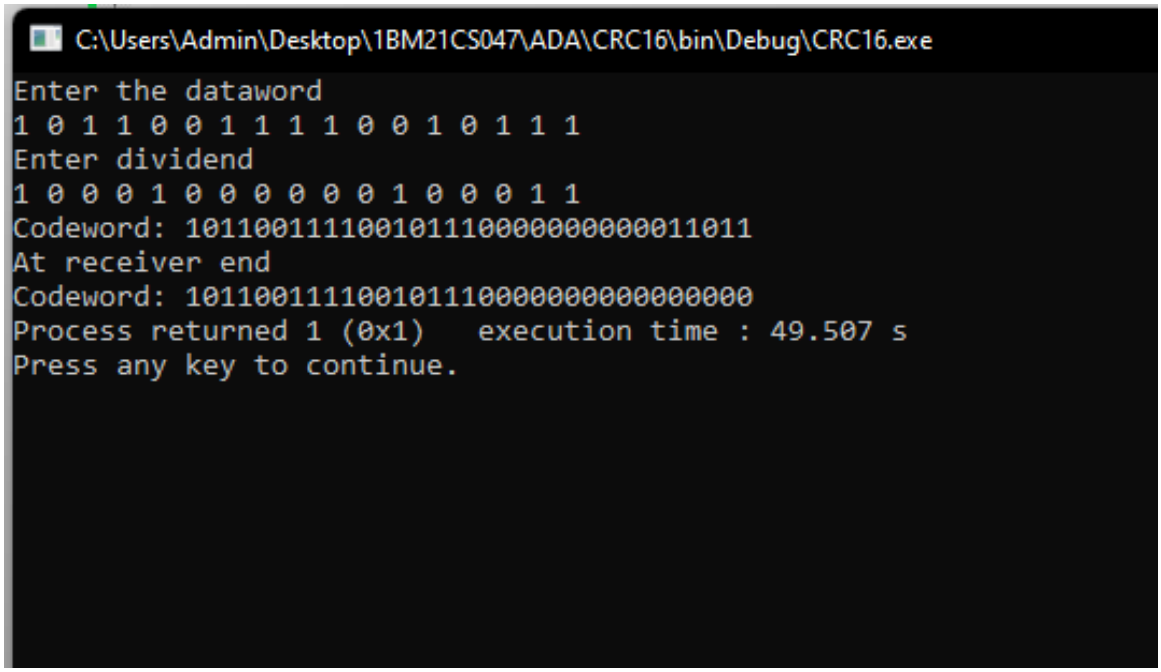
CODE:

```
#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");
```

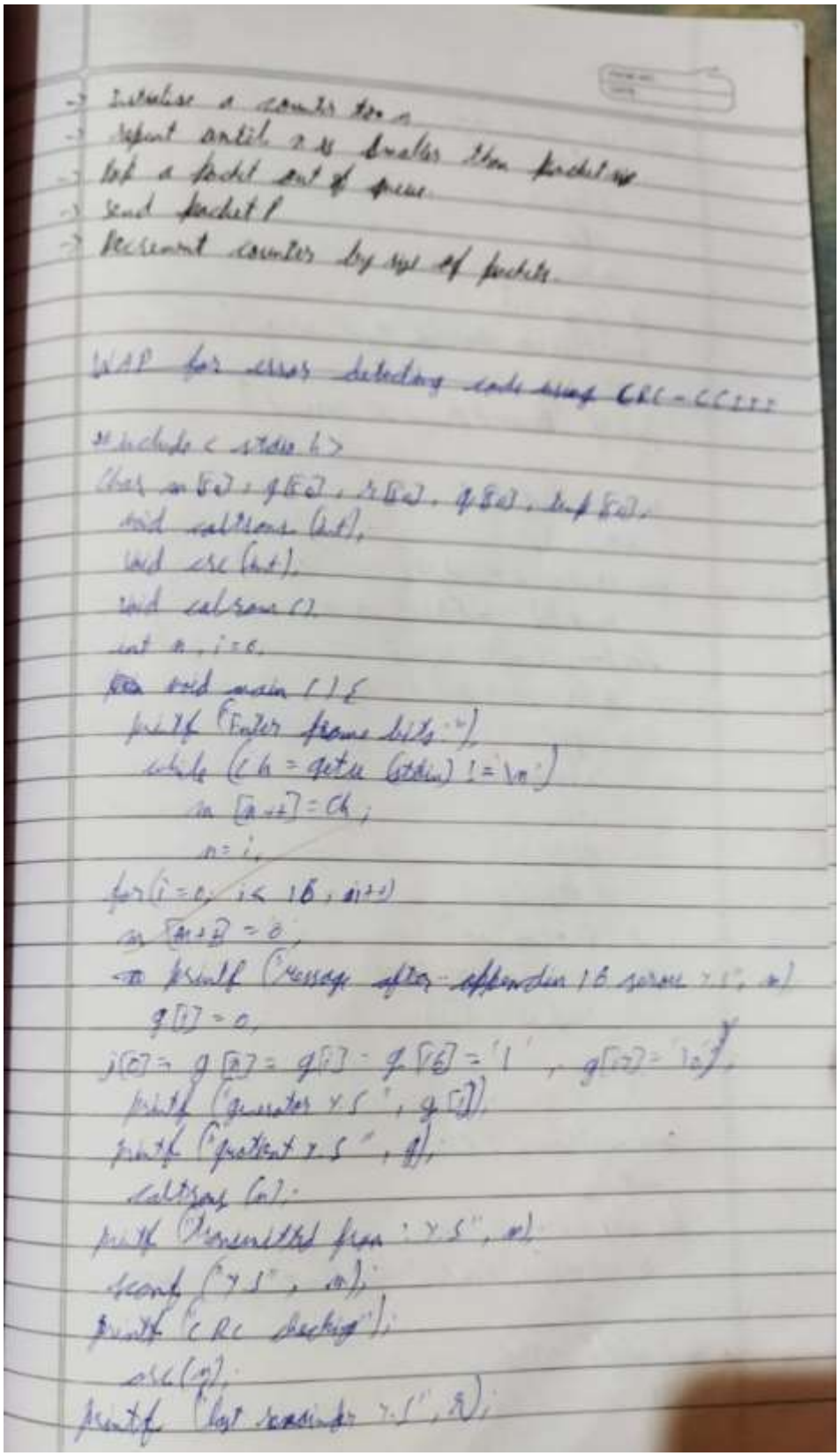
```
for(i=0;i<17;i++)
    scanf("%d",&dd[i]);

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:



OBSERVATION:



-> Initialize a counter to 0
-> Repeat until x is greater than packet size
-> Take a packet out of queue.
-> Send packet P
-> Decrement counter by size of packets.

WAP for error detecting code using CRC - CCITT

```
#include <stdio.h>
char m[5], g[5], r[5], q[5], buf[5];
void calcrem(int);
void crc(int);
void calsum();
int n, i=0;
void main ()
{
  printf("Enter frame bits:");
  while (ch = getch(stdin) != '\n')
      m[n++] = ch;
      n = i;
for(i=0; i< 16; i++)
m[n+i] = 0;
printf("message after appending 16 zeroes %s", m)
  q[1] = 0;
i[0] = q[0] = q[1] = q[5] = '1', q[2] = '0';
  printf("generator %s", q[1]);
printf("protocol %s", q);
  calcrem(n);
printf("transmitted frame: %s", m);
scanf("%s", m);
printf("CRC checking");
  crc(q);
printf("last remainder %s", r);
```

```
for(i=0, i<16, i++)
    if b[i] != b'
     flag=1,
     else
     continue,
    if (flag==1)
    printf("error during transmission"),
     else
    printf("Received frame is correct"),
    }
void crc(int a) {
    int i, j,
    for (i=0, i<n, i++)
        rem[i] = m[i],
    for (i=0, i<m-16, i++) {
        if (rem[i] == '1') {
            g[i] = '1',
            calcrem(i),
            else
            g[i] = '0',
            shift(i) }
        r[i] = m[m+i],
    for (i=0, i<r[i], i++)
        rem[i] = r[i],
        }
    }

void calcrem() {
    int i, j,
    for (i=1, i<=16, i++)
        r[i-1] = (int) rem[i]-48) ^ (int) g[i] = 48),
        }
void shift() {
    int i;
```

```
for(i=1, i<=16; i++)
    r[i-1] = r[i];
}
void caltrans (int n) {
    int i, k=0;
for(i=n-16; i<n; i++)
    m[i] = ((int) m[i]-48) ^ ((int) r[k++] - 48) + 48;
    m[i] = '0';
}
```

output:

Enter frame bits : 1011

message after appending 16 zeros: 1011000000 000000000

Generator : 10001000000100001

quotient : 1011

transmitted frame : 1011101100010110101011

enter transmitted frame : 1011101100010110101011

Last remainder 0000000000000000

Recieved frame is correct.

WEEK 14

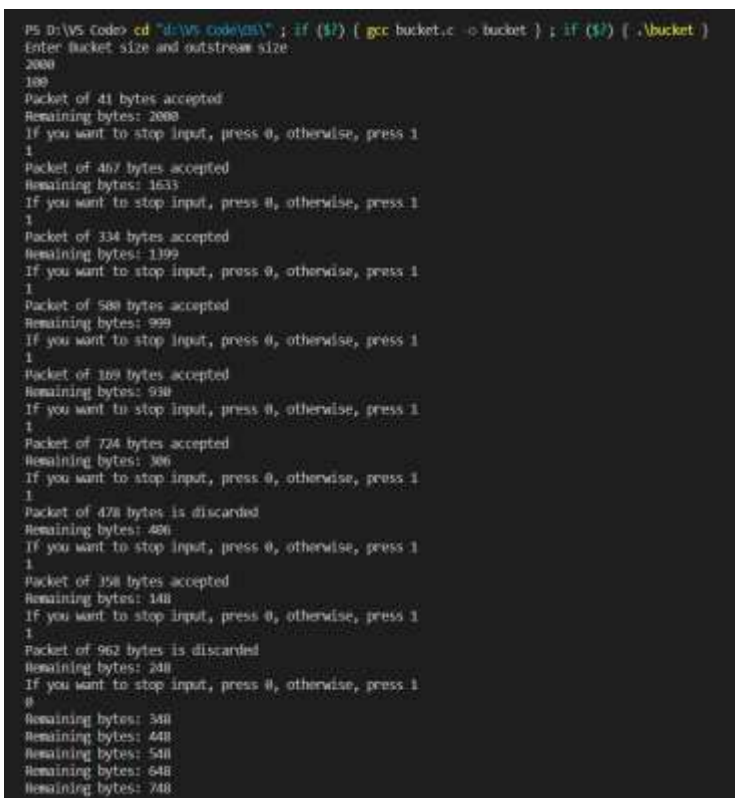
Write a program for congestion control using Leaky bucket algorithm.

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\CN\" ; 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 470 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:

Congestion - Management - Leaky bucket algorithm:

```
#include <stdio.h>
int main () {
int incoming, outgoing, store=0, n;
printf ("Enter bucket size, outgoing rate, no of inputs");
scanf ("%d %d %d", &bucket, &outgoing, &n);
while (n!=0) {
printf ("Enter incoming packet size");
scanf ("%d", &incoming);
printf ("Incoming packet size %d", incoming);
if (incoming <= bucket size - store) {
        store += incoming;
printf ("Bucket buffer size %d out of %d", store, bucket size);
}
else {
printf ("Bucket buffer size %d out of %d", store, bucket size);
store = bucket-size;
}
store = store - outgoing;
}
}
```

Output:

Enter bucket size, outgoing rate and no of inputs: 8 6 4
Enter incoming packet size: 2
Bucket buffer size 2 out of 8
After outgoing - 0 packets left out of 8 in buffer.

Explanation:

Leaky bucket algorithm can be implemented using FIFO queue. It holds the packet traffic consist of fixed size packets. The process removes fixed no of packets from queue at each tick of clock

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

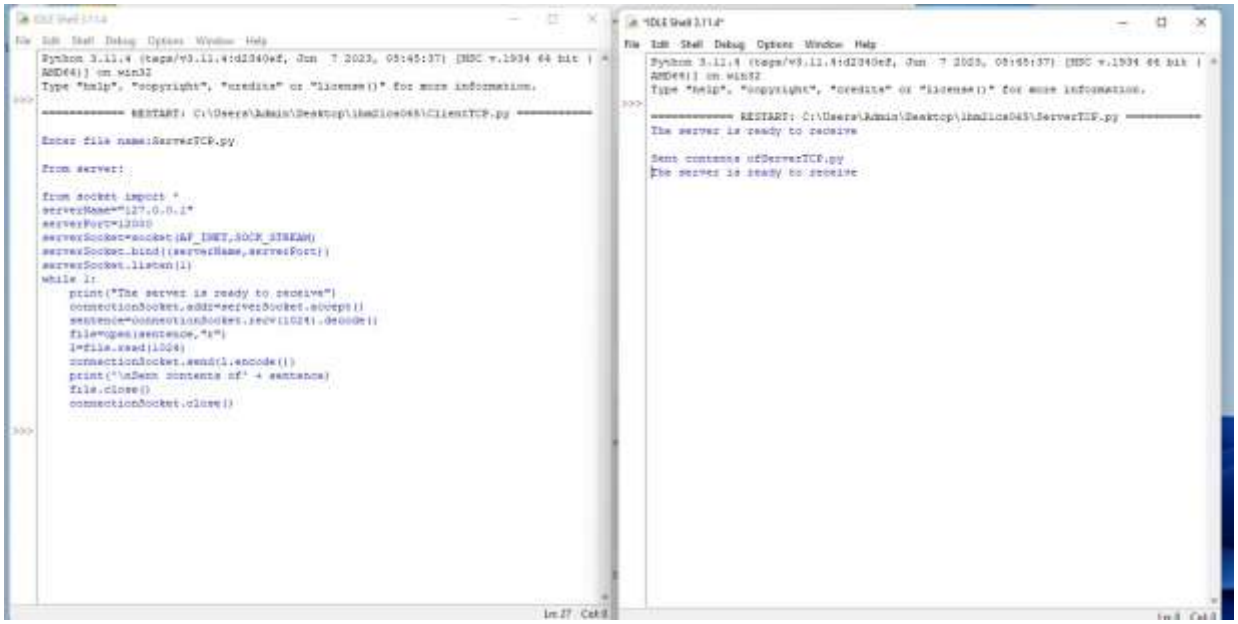
```
from socket import *
serverName = “127.0.0.1”
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = input(“\nEnter file name: “)
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print (“\nFrom Server:\n”)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

ServerTCP.py

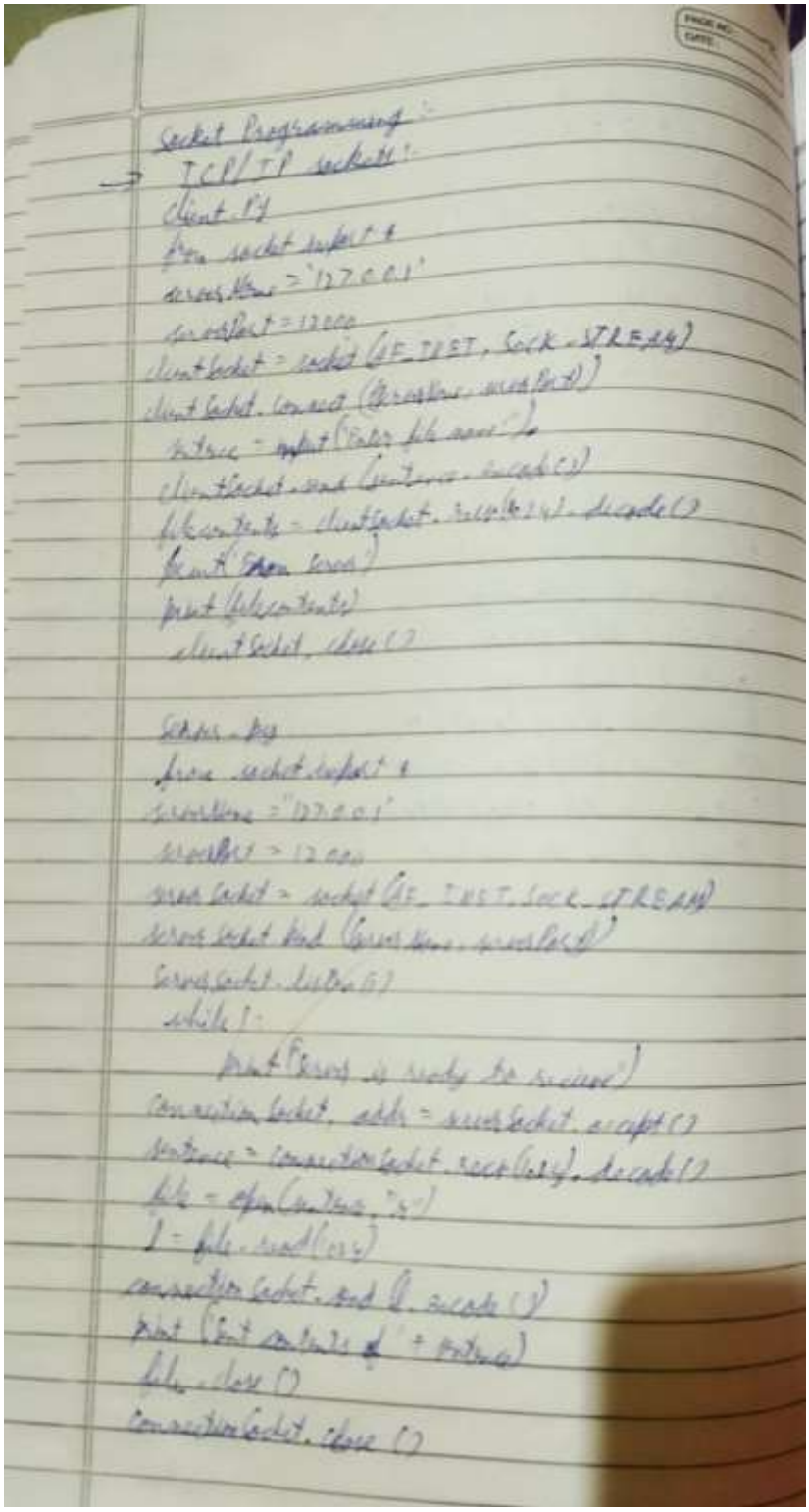
```
from socket import *
serverName=”127.0.0.1”
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
print (“The server is ready to receive”)
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024).decode()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

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

OUTPUT:



OBSERVATION:



Socket Programming:

→ TCP/IP sockets:

Client.py

from socket import *

serverHost = '127.0.0.1'

serverPort = 12000

clientSocket = socket(AF_INET, SOCK_STREAM)

clientSocket.connect((serverHost, serverPort))

inStr = input("Enter file name")

clientSocket.send(inStr.encode())

fileContents = clientSocket.recv(2048).decode()

print("Server sent")

print(fileContents)

clientSocket.close()

Server.py

from socket import *

serverHost = "127.0.0.1"

serverPort = 12000

serverSocket = socket(AF_INET, SOCK_STREAM)

serverSocket.bind((serverHost, serverPort))

serverSocket.listen(1)

while 1:

print("Server is ready to receive")

connectionSocket, addr = serverSocket.accept()

inStr = connectionSocket.recv(1024).decode()

f = open(inStr, "r")

l = f.read(1024)

connectionSocket.send(l.encode())

print("Sent contents of" + inStr)

f.close()

connectionSocket.close()

Output:

Enter filename : server.py

Error error : (Error code is received)

Output : the server is ready to receive

Sent contents of server.py

The server is ready to receive.

→ Using UDP sockets, write client server program

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()
```

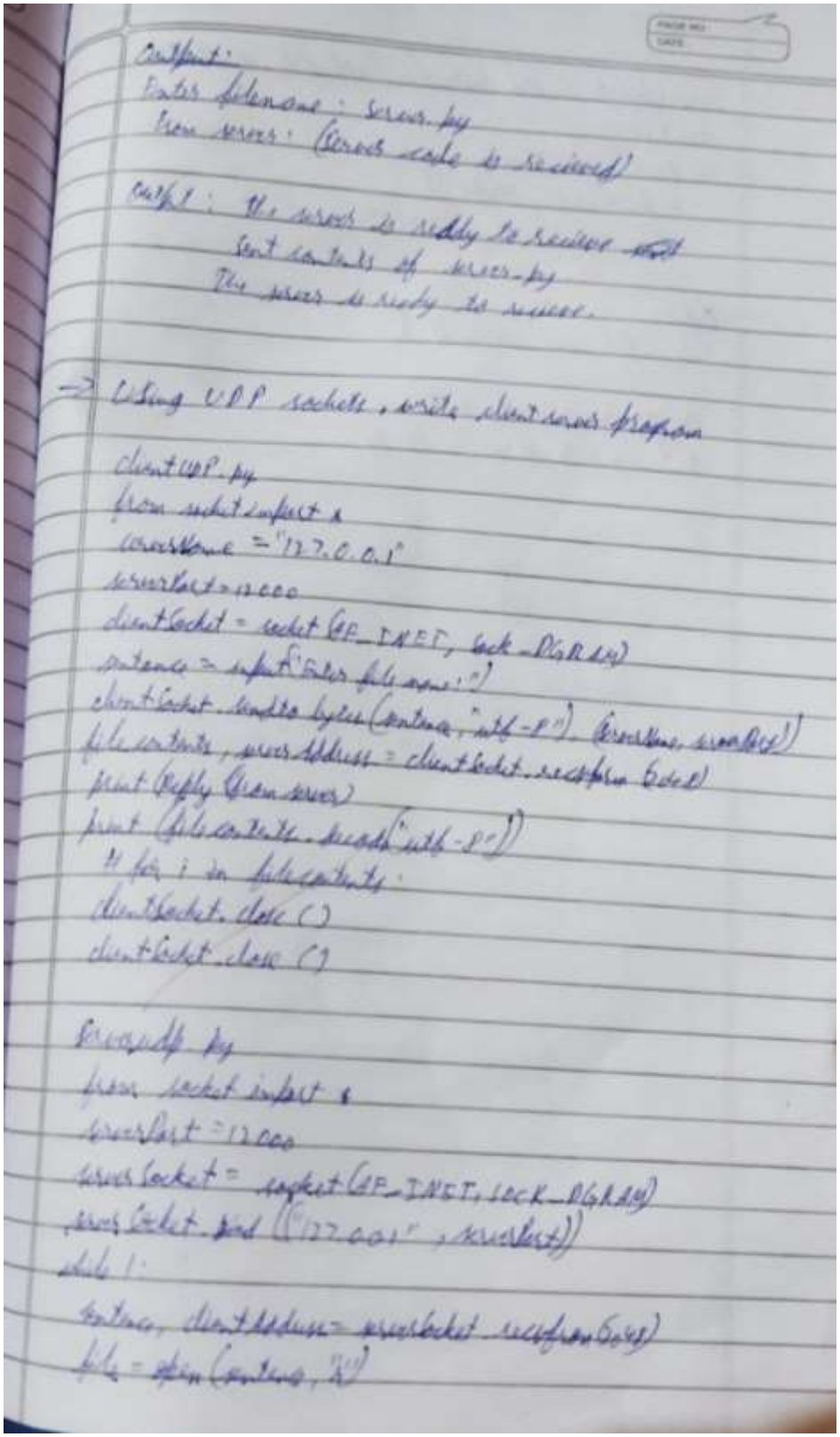
ServerUDP.py

```
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:

OBSERVATION:



Output:
Enter filename: server.py
from server: (Errors code is received)

output: The server is ready to receive
Sent contents of server.py
The server is ready to receive.

=> Using UDP sockets, write client server program

clientUDP.py
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("Enter file name:")
clientSocket.sendto bytes(sentence, "utf-8"), (serverName, serverPort))
fileContents, serverAddress = clientSocket.recvfrom(2048)
print("Reply from server)
print (fileContents.decode("utf-8"))
for i in fileContents:
clientSocket.close ()
clientSocket.close ()

serverUDP.py
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind (("127.0.0.1", serverPort))
while 1:
sentence, clientAddress = serverSocket.recvfrom(2048)
file = open(sentence, "r")

```
serversocket.sendto(bytes(Con.Calf.r"), clientAddress)
print(sentence)
# for i in sentence:
    # print(str(i), end="")
    file.close()
```

Output:

Enter file name: serverudp.py

The server is ready to receive

Sent contents of serverudp.py