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

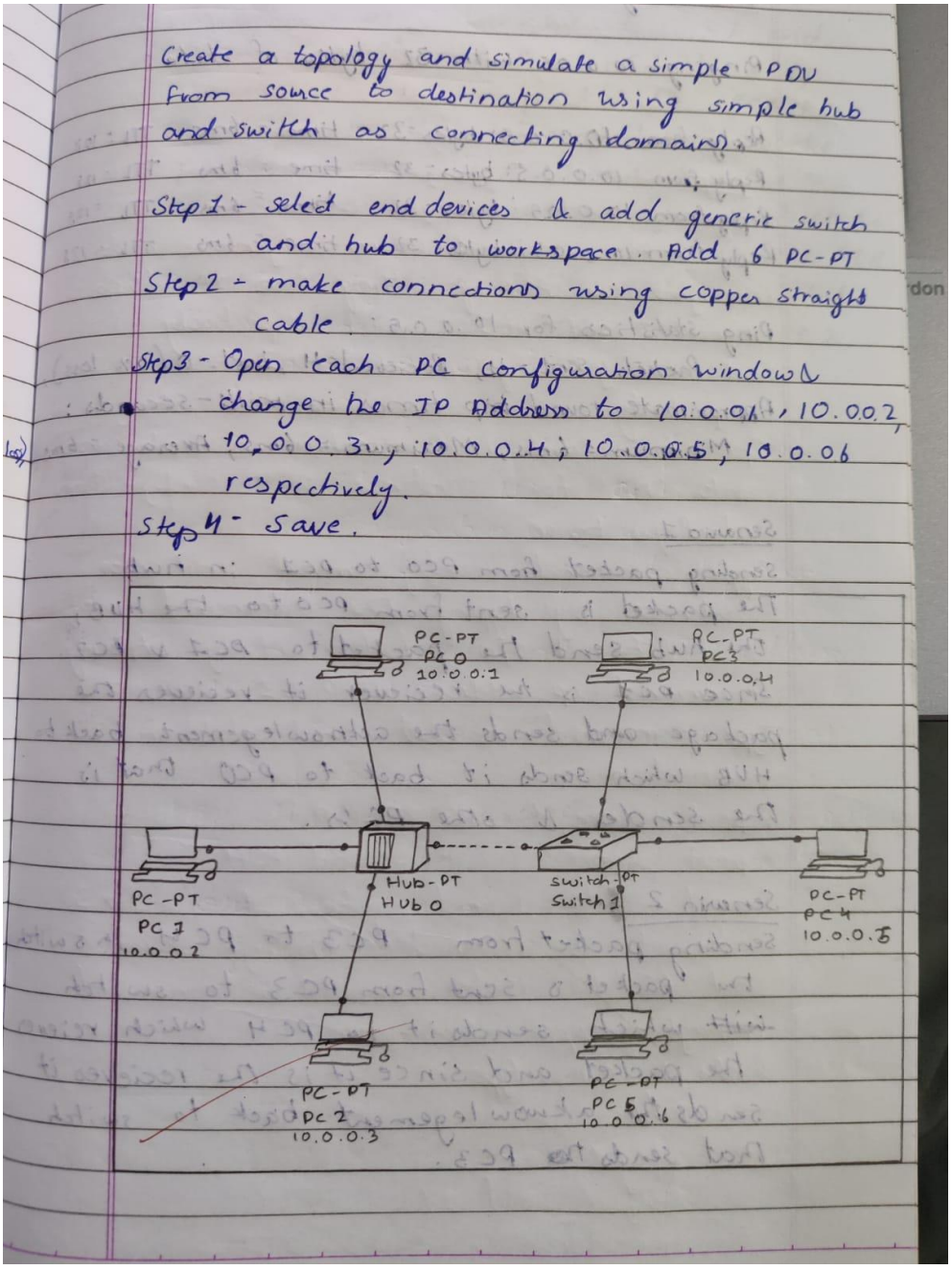
`

Index

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, Recieved = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 6ms, Maximum = 6ms, Average = 6ms

Senario 1

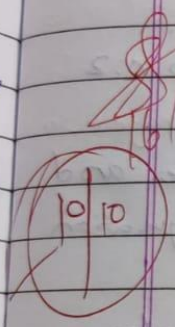
Sending packet from PC0 to PC1 in hub.
The packet is sent from PC0 to the HUB, the hub send the packet to PC1 & PC2. Since PC1 is the reciever it recieves the package and sends the acknowlegement back to HUB which sends it back to PC0 that is the sender & the PC's.

Senario 2

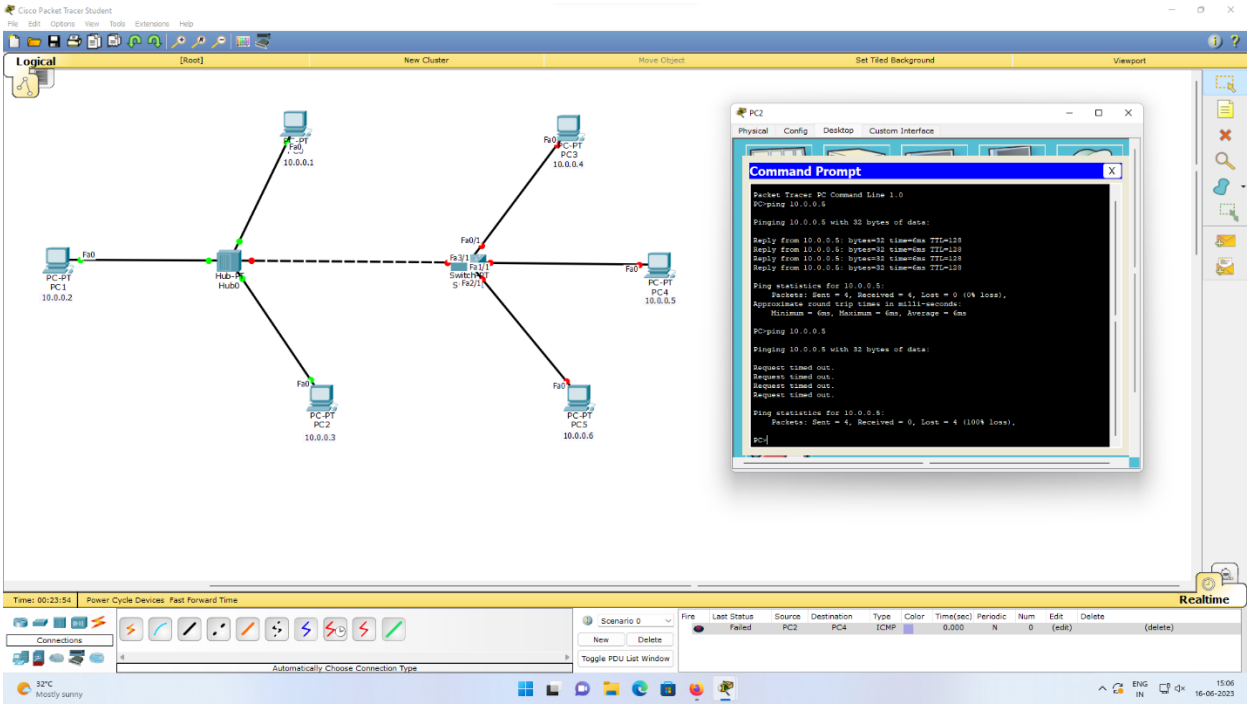
Sending packet from PC3 to PC4 in switch
The packet is send from PC3 to switch ~~with~~ which sends it to PC4 which recieves the packet and since it is the recieves it sends the aknowlegement back to switch that sends the PC3.

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 & from PC4 which is the reciever 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



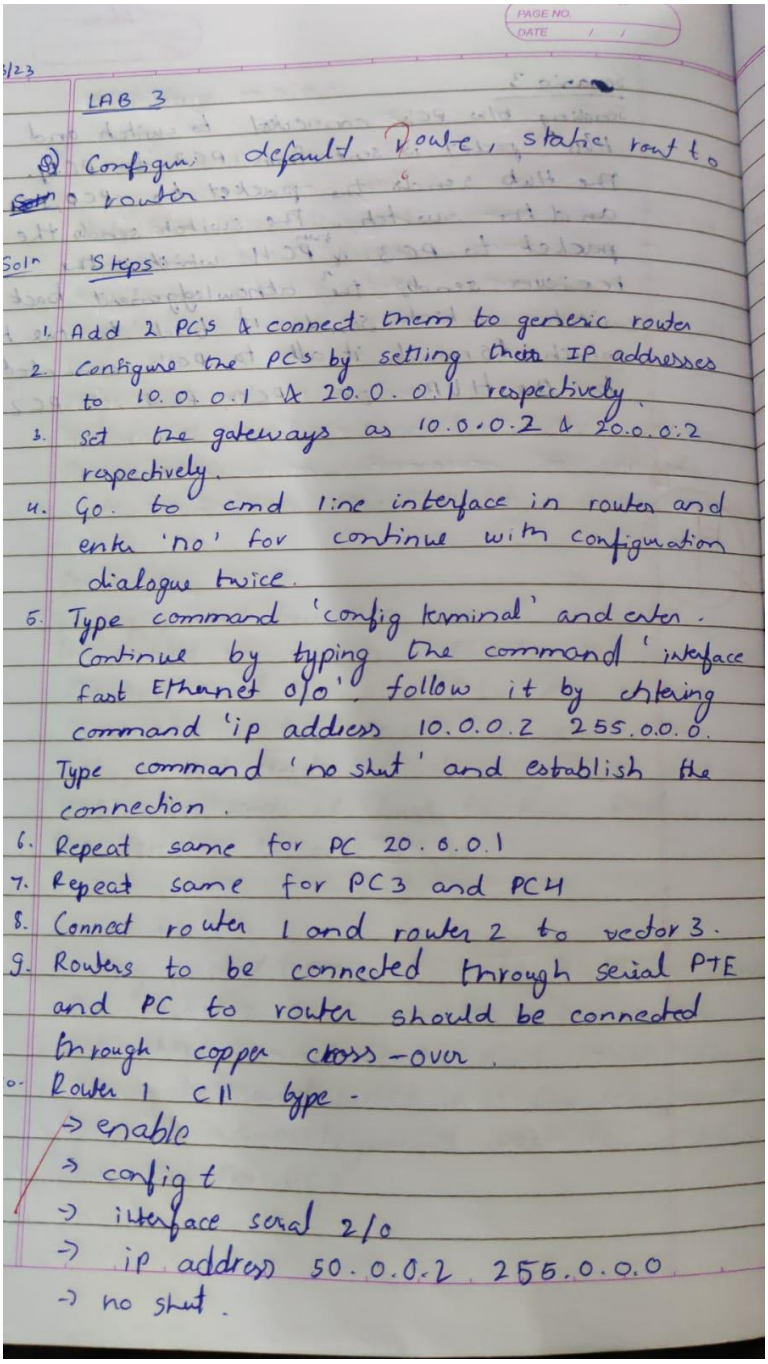
TOPOLOGY & OUTPUT:



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 default route, static route to router.

Soln Steps:

1. Add 2 PC's & connect them to generic router
2. Configure the PCs by setting their IP addresses to 10.0.0.1 & 20.0.0.1 respectively.
3. Set the gateways as 10.0.0.2 & 20.0.0.2 respectively.
4. Go to cmd line interface in router and enter 'no' for continue with configuration dialogue twice.
5. Type command 'config terminal' and enter. Continue by typing the command 'interface fast Ethernet 0/0' follow it by entering command 'ip address 10.0.0.2 255.0.0.0. Type command 'no shut' and establish the connection.
6. Repeat same for PC 20.0.0.1
7. Repeat same for PC3 and PC4
8. Connect router 1 and router 2 to vector 3.
9. Routers to be connected through serial PTE and PC to router should be connected through copper cross-over.
10. Router 1 CLI type -
    -> enable
    -> config t
    -> interface serial 2/0
    -> ip address 50.0.0.2 255.0.0.0
    -> no shut.

Adora
PAGE NO.
DATE / /

111's for 50.0.0.1 255.0.0.0.
The ~~comb~~ 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 10.0.0.0 255.00.0 50.0.0.1
ip route 20.0.0.0 2550.0.0 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, Recieved = 0, lost = 4 (100% loss)

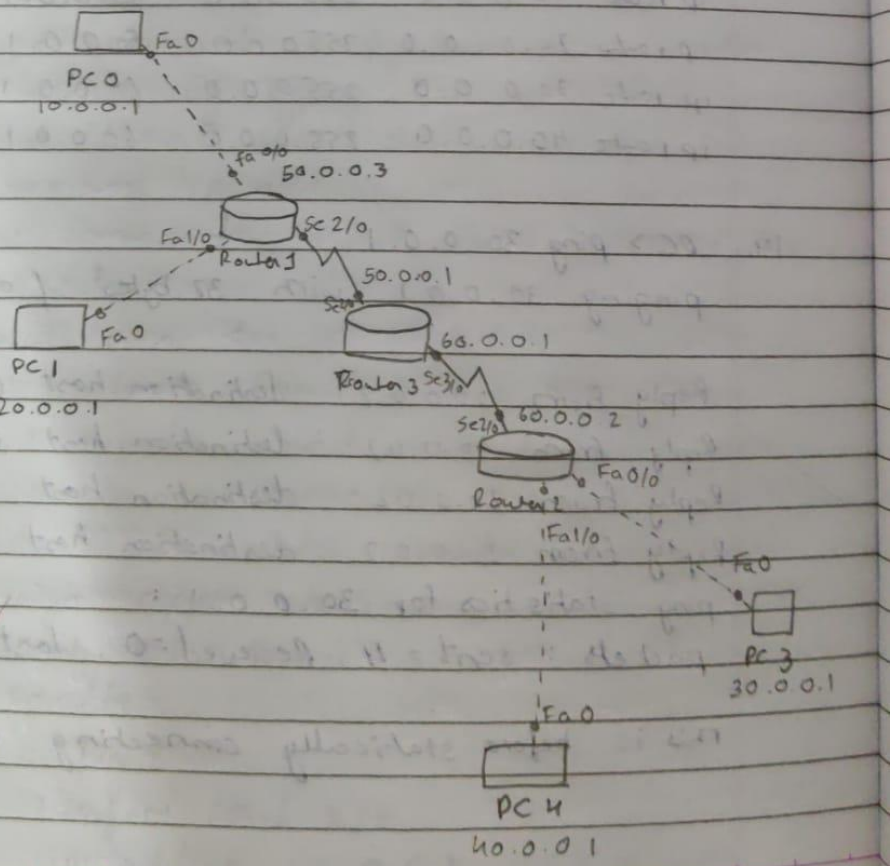
This is before statically connecting PC's.

PC > ping 40.0.0.1

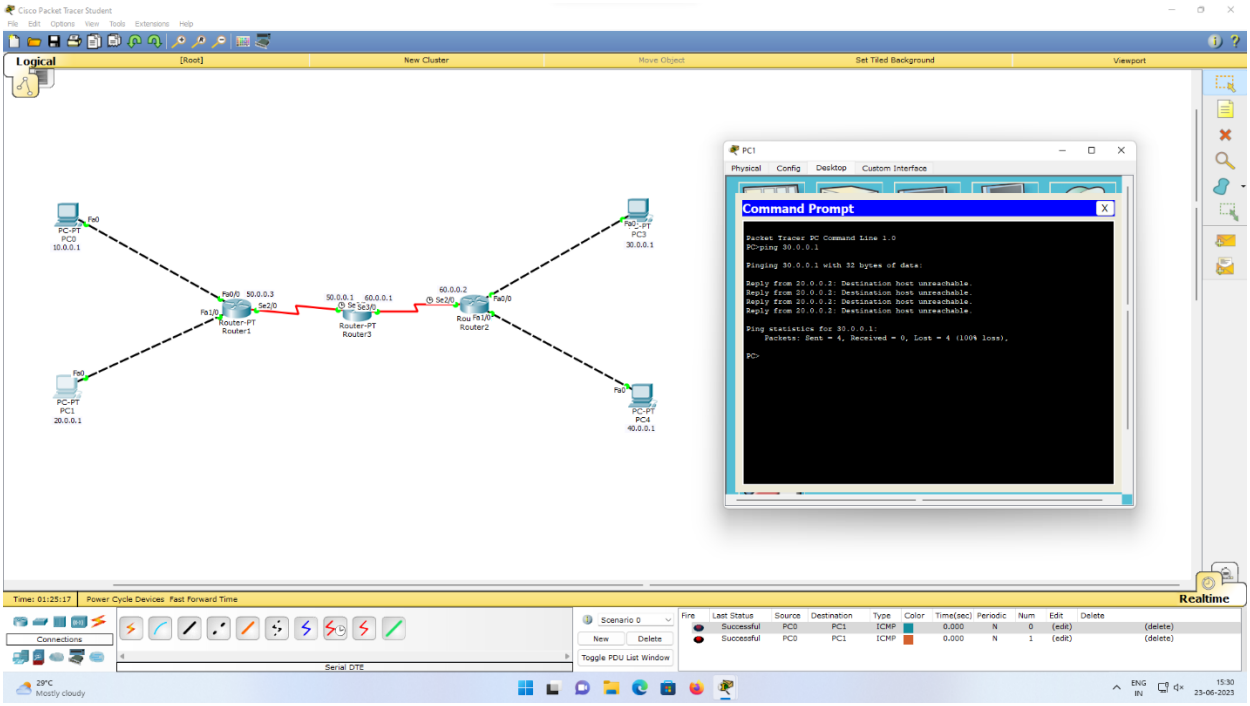
Pinging 40.0.0.1 with 32 bytes of data:

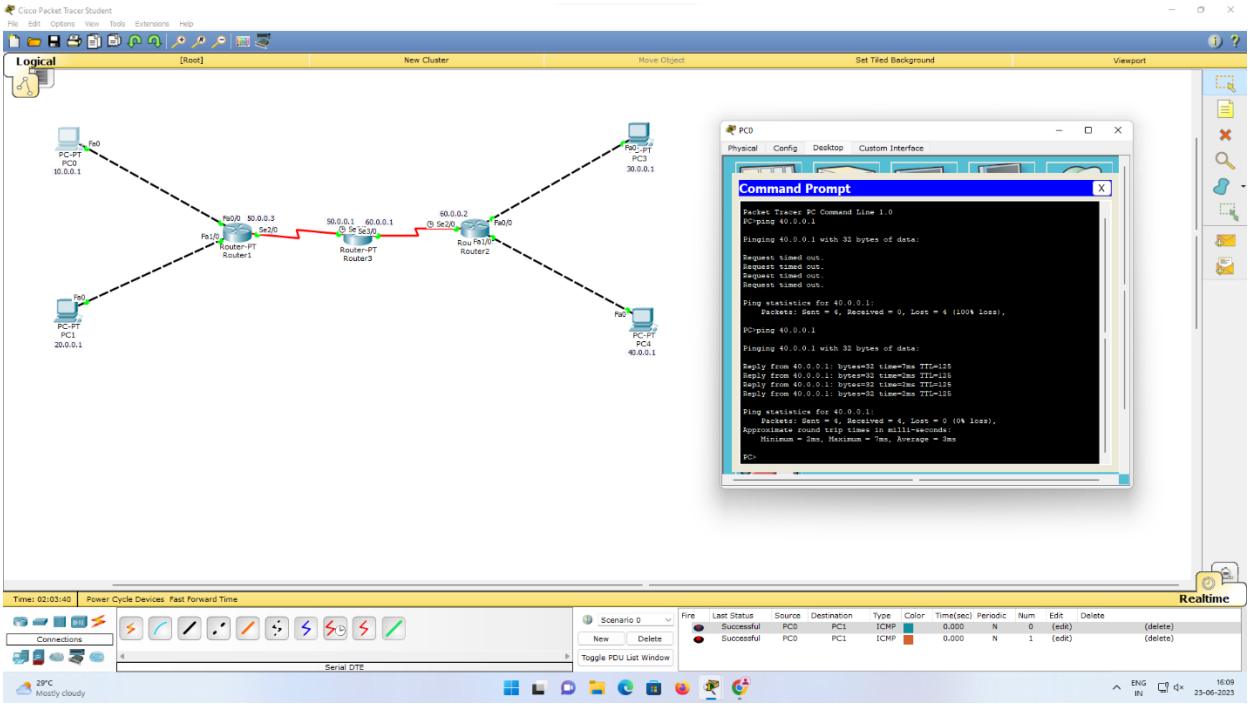
Reply from 40.0.0.1 : bytes=32 time = 7ms TTL = 125
Reply from 40.0.0.1 : bytes=32 time = 2ms TTL = 125
Reply from 40.0.0.1 : bytes=32 time = 2ms TTL = 125
Reply from 40.0.0.1 : bytes=32 time = 2ms TTL = 125

Ping statistics for 40.0.0.1:
Packets : sent = 4, Recieved = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 7ms, Average = 3ms



TOPOLOGY & OUTPUT:

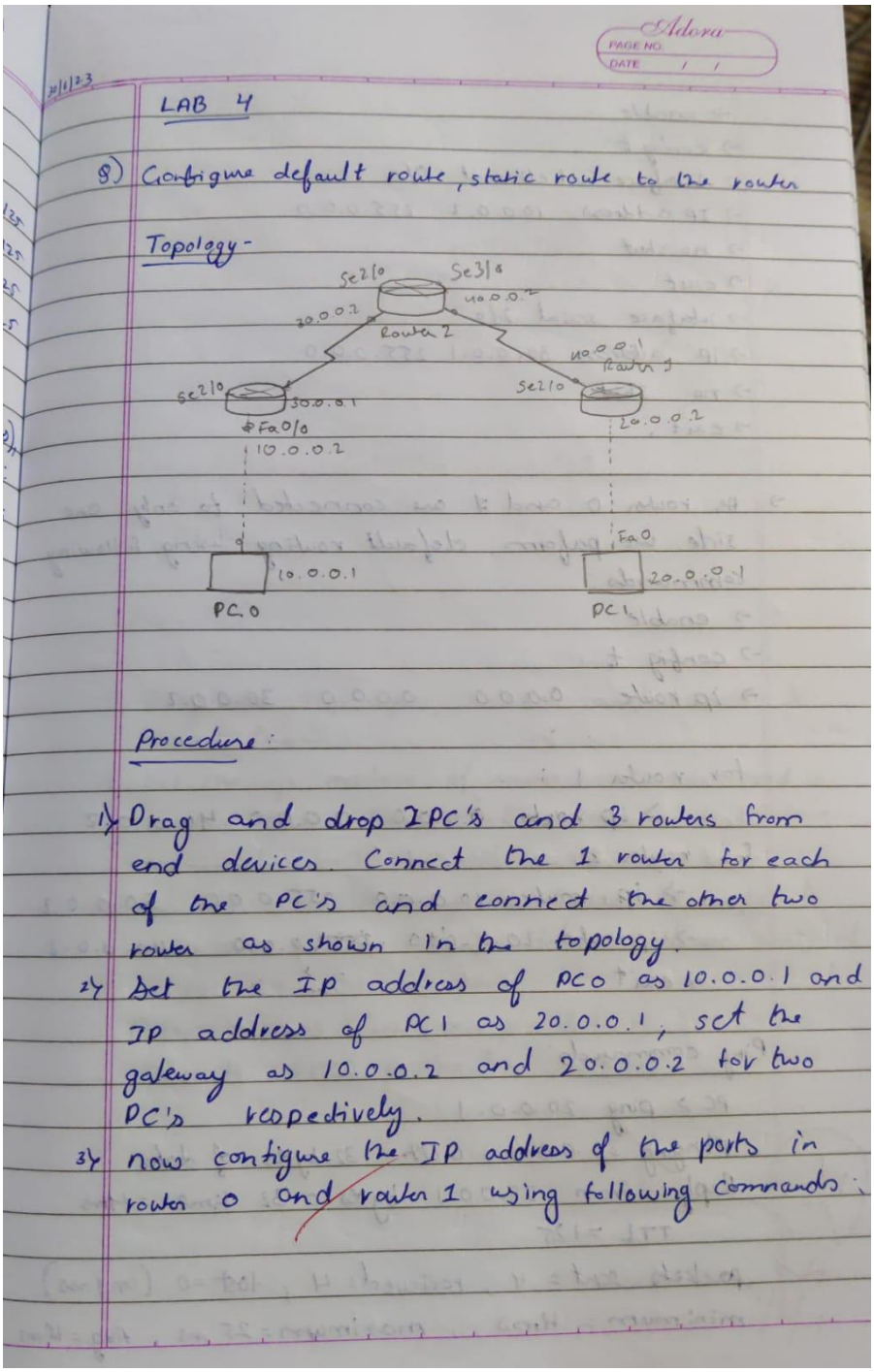




WEEK 3

Configure default route, static route to the Router.

OBSERVATION:



PAGE NO.
DATE / /

→ enable
→ config t
→ interface fastethernet 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.

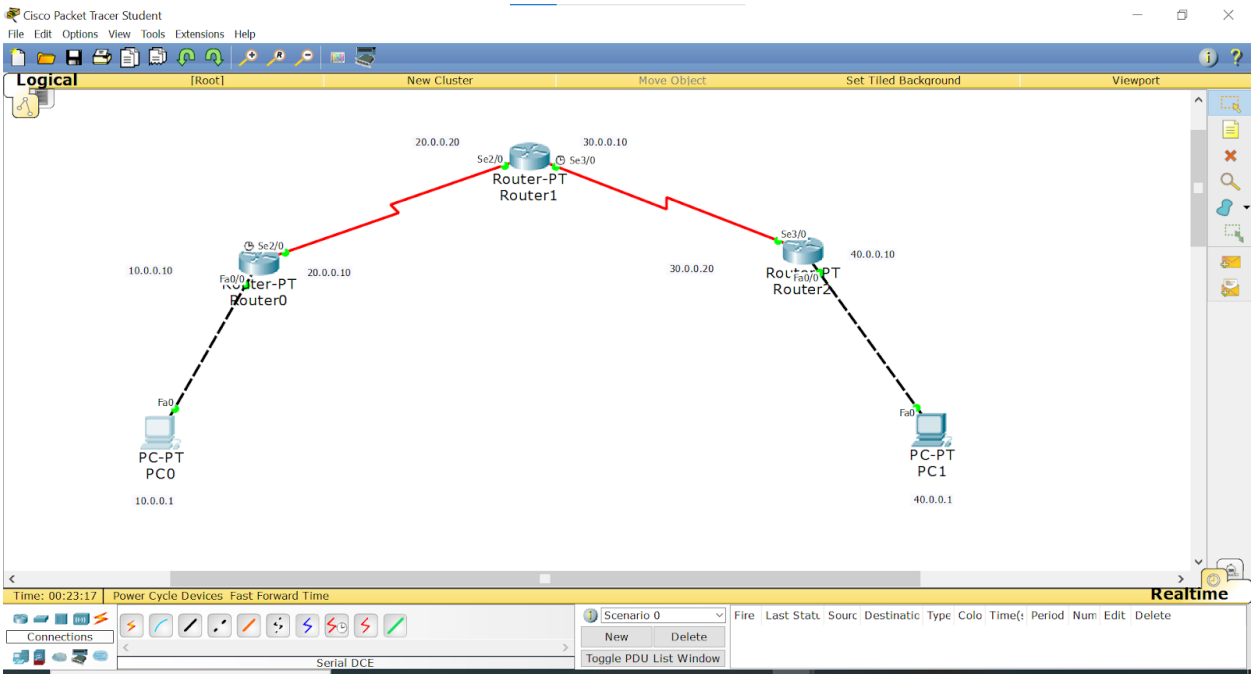
→ As router 0 and 1 are connected to only one side we paform default routing using following commands:
→ enable
→ config t
→ ip route 0.0.0.0 0.0.0.0 30.0.0.2

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

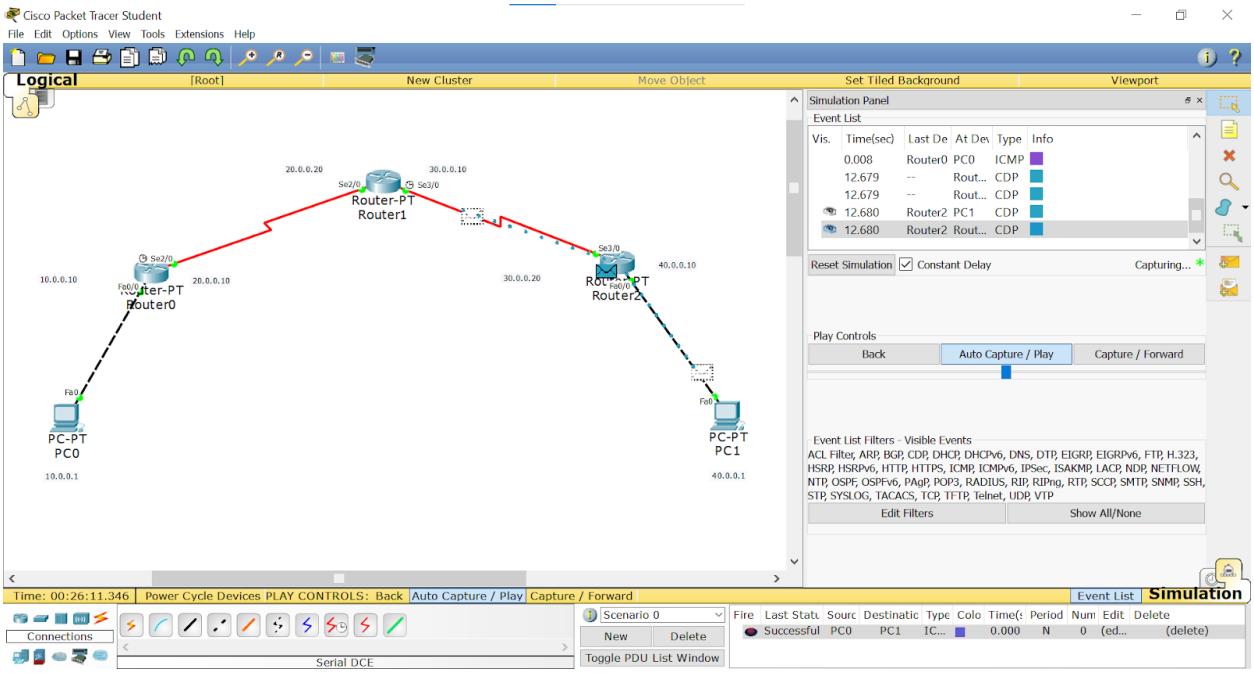
Ping commands

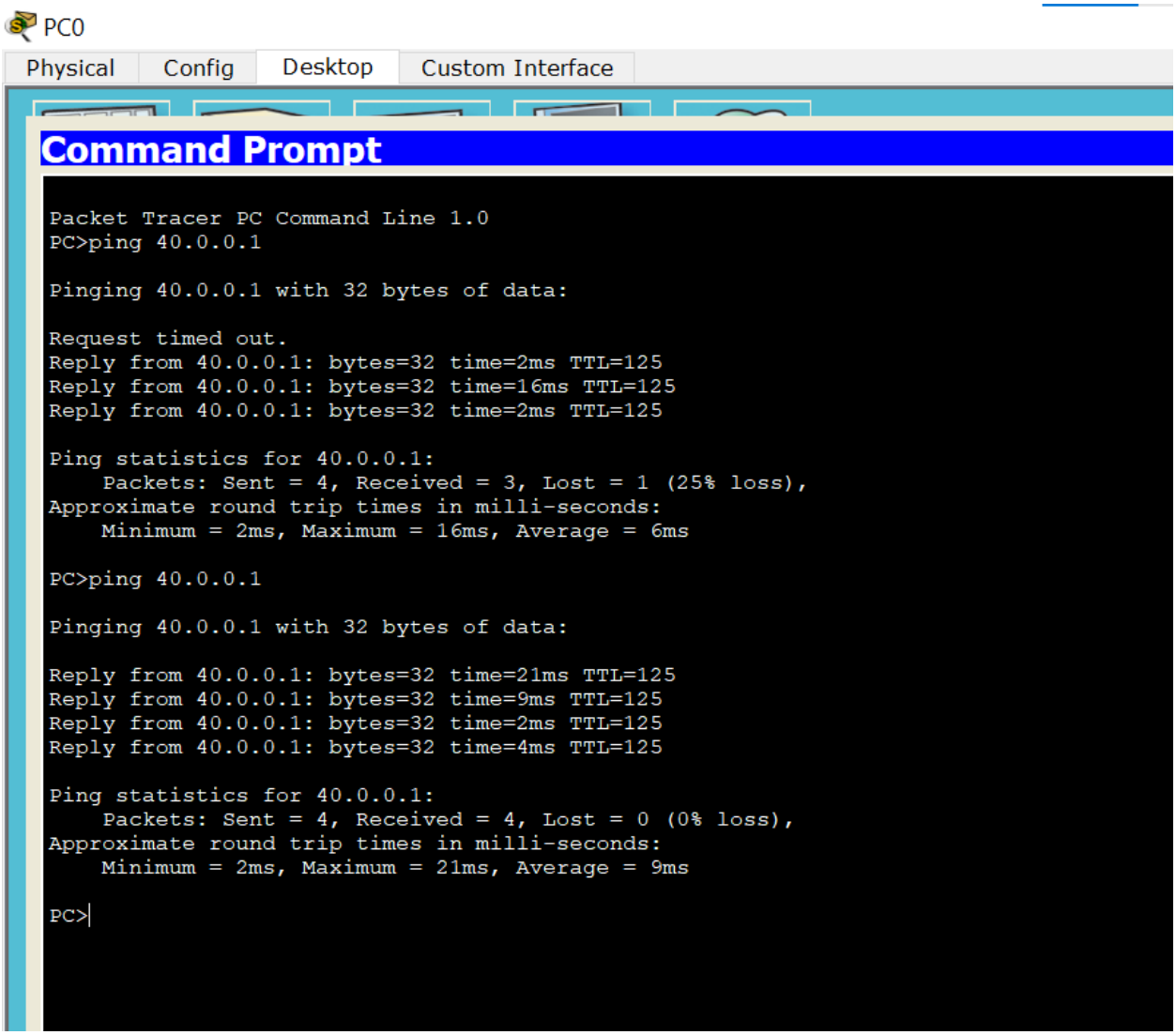
PC > ping 20.0.0.1
Pinging 20.0.0.1 with 32 bytes of data
Reply from 20.0.0.1 bytes = 32 time = 4ms
TTL = 125
packets sent = 4, recieved = 4, lost = 0 (0% loss)
minimum = 4ms, maximum = 25 ms, Avg = 16 ms

TOPOLOGY:



OUTPUT:

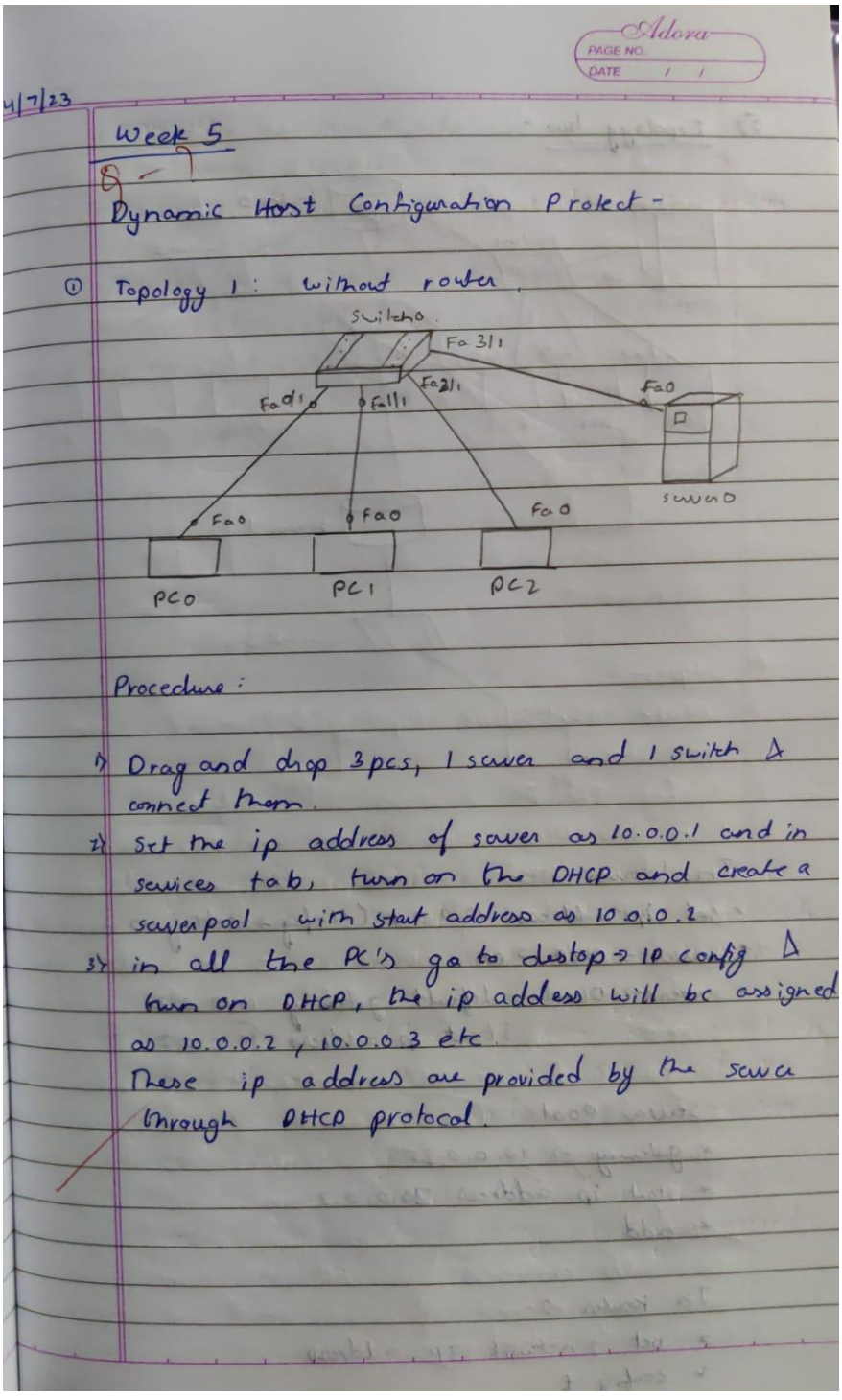



PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 40.0.0.1
Pinging 40.0.0.1 with 32 bytes of data:
Request timed out.
Reply from 40.0.0.1: bytes=32 time=2ms TTL=125
Reply from 40.0.0.1: bytes=32 time=16ms TTL=125
Reply from 40.0.0.1: bytes=32 time=2ms 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>

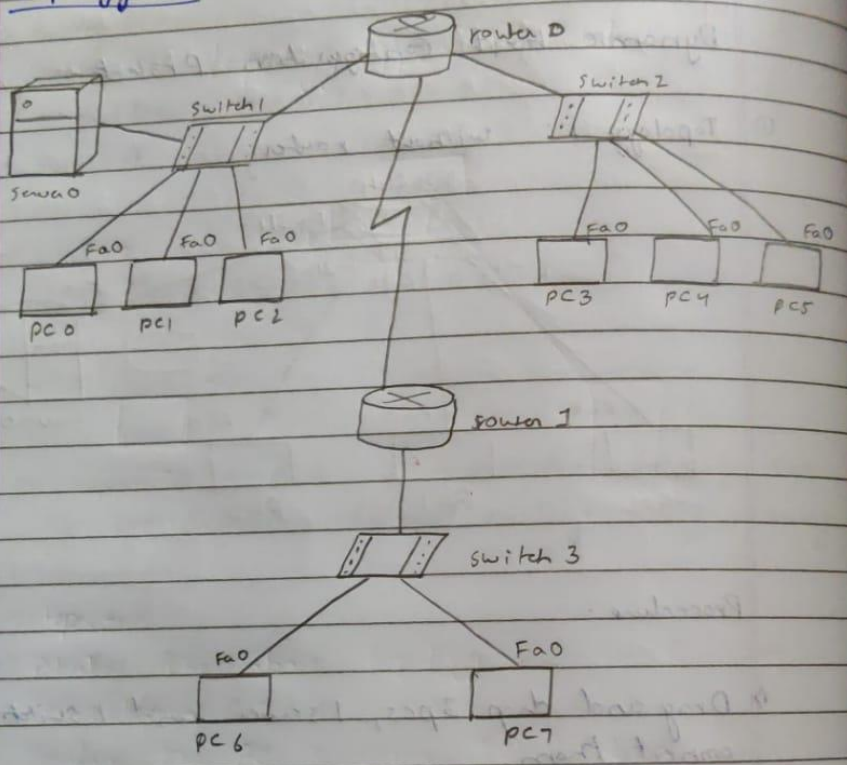
WEEK 4

Configure DHCP within a LAN and outside LAN.

OBSERVATION:



(2) 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 10.0.0.2
* add

In router 0 -

* set 2 network IP address
* config t

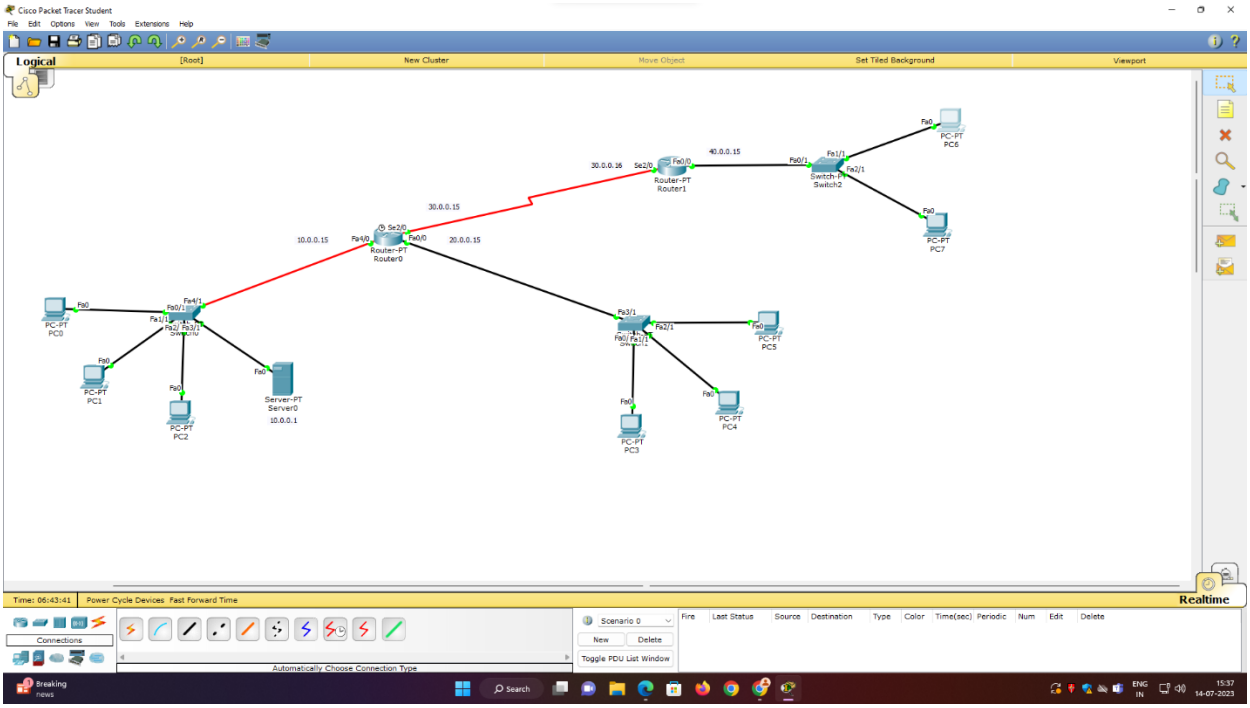
* interface fastethernet 4/0
* ip address 10.0.0.25 255.0.0.0
* no shut
(Ully 20.0.0.25)
* config t * interface fastethernet 0/0
* ip helper-address 10.0.0.1
* no shut
→ static route (for 40 ip)
config t
ip route 40.0.0.0 255.0.0.0 30.0.0.0

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

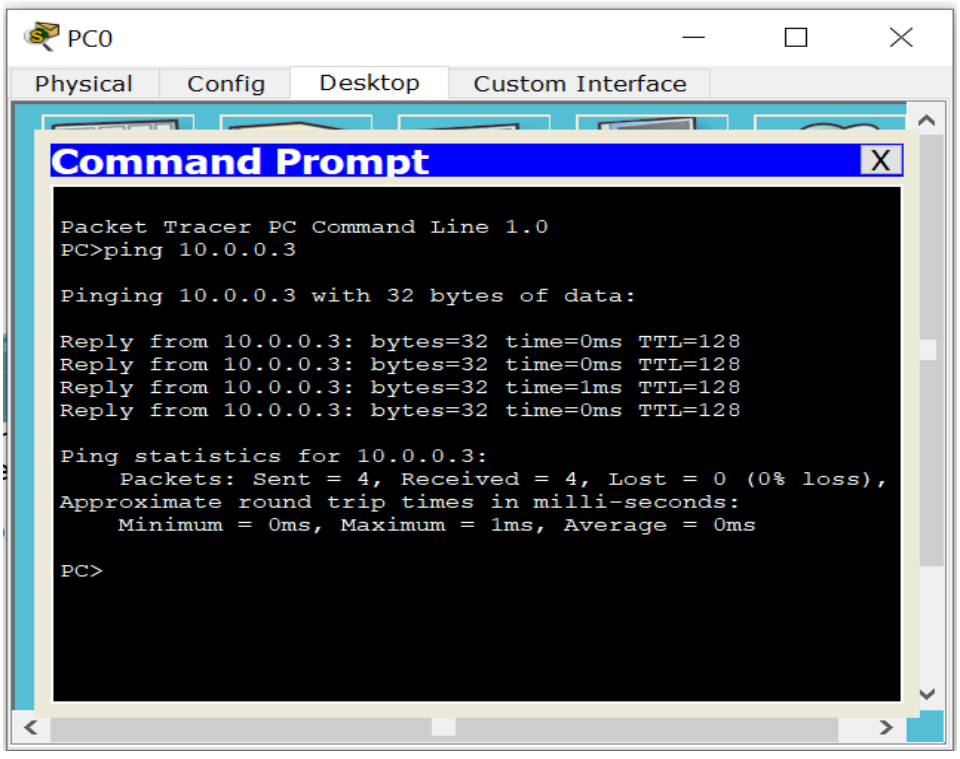
→ static routing for 10 & 20 network
config t
ip route 10.0.0.0 255.0.0.0 30.0.0.25
ip route 20.0.0.0 255.0.0.0 30.0.0.25
no shut

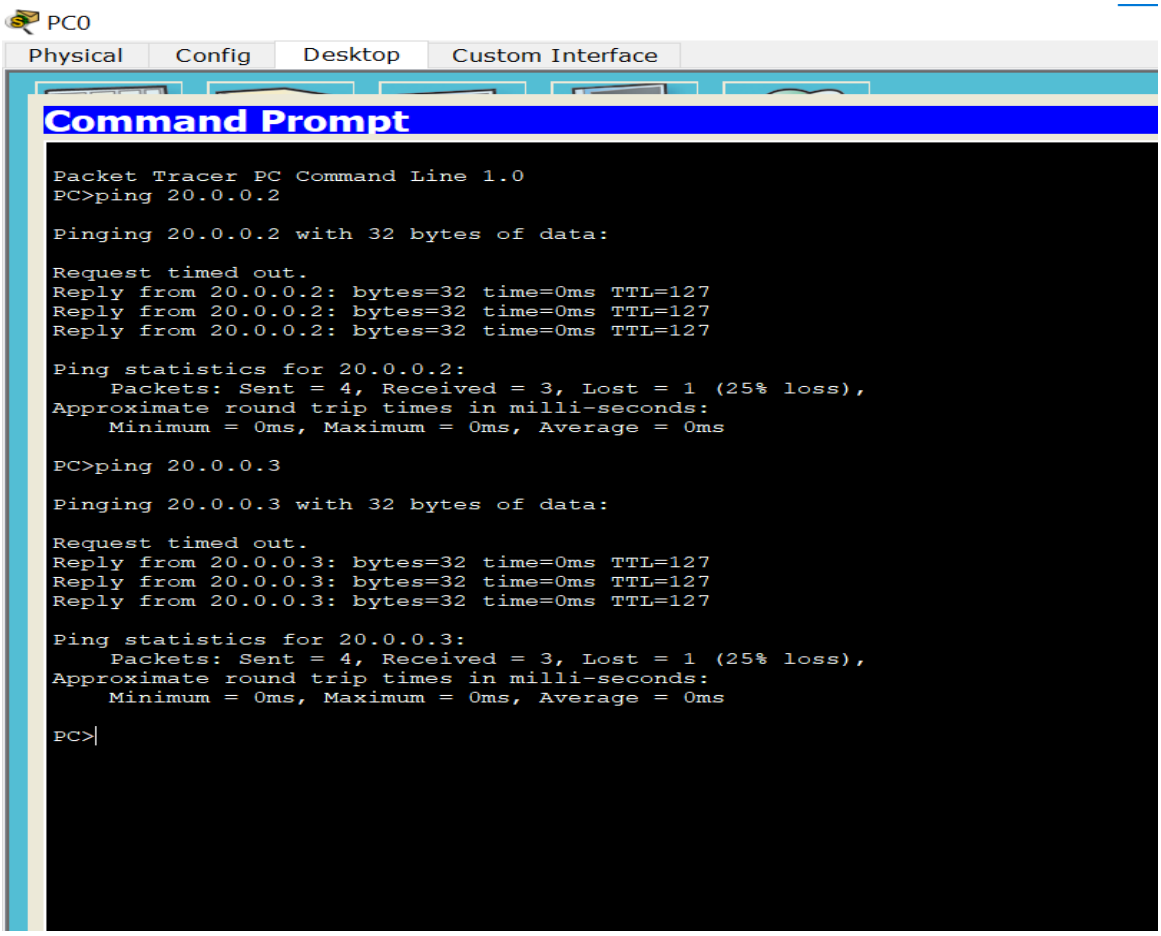
→ setting helper address
config t
interface fastethernet 0/0
ip-helper address (0.0.0.0)
no shut.

TOPOLOGY:



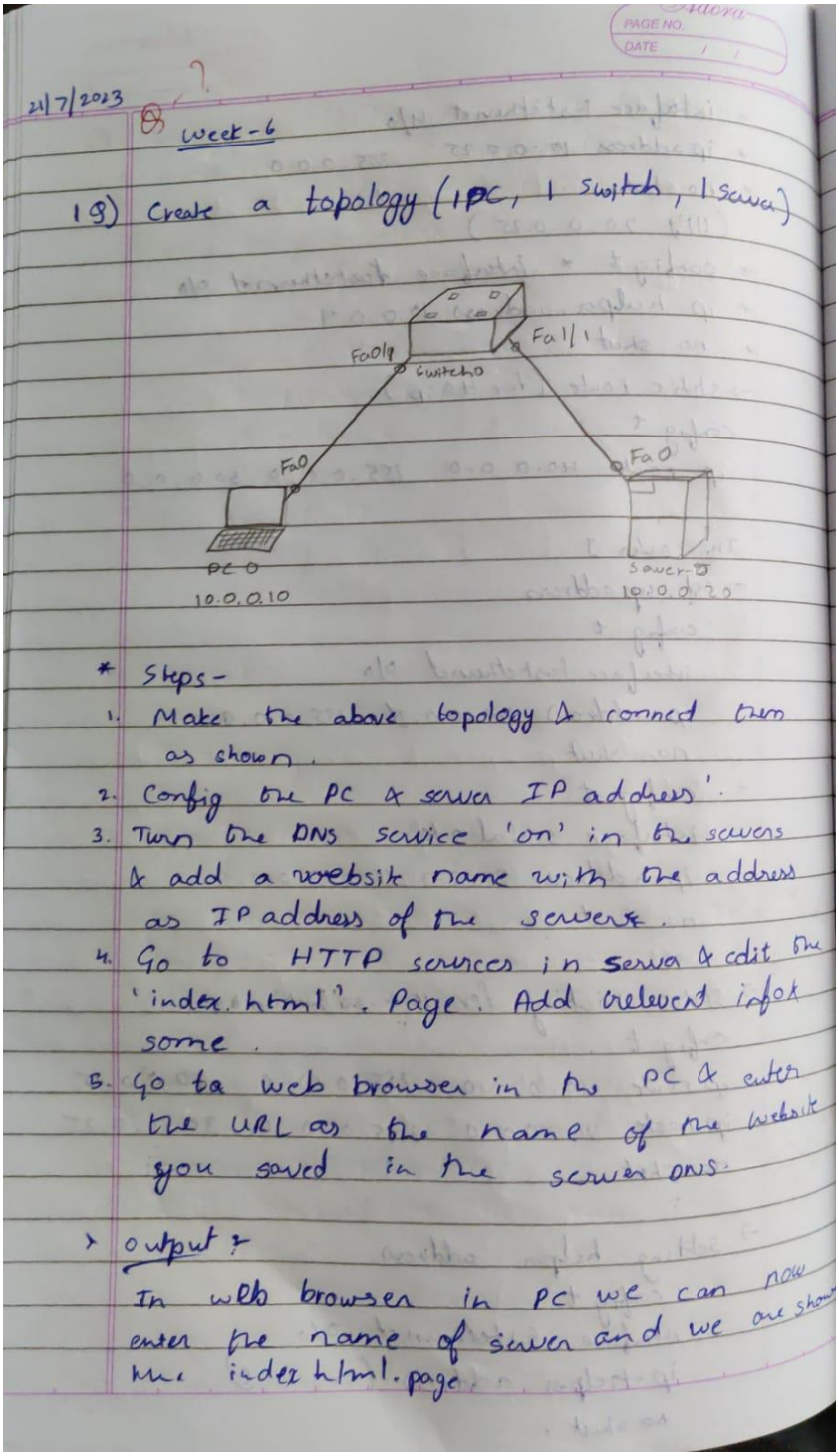
OUTPUT:




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



Adora
PAGE NO.
DATE / /

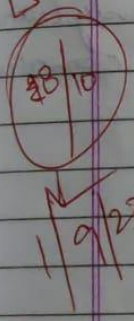
Rahul Shirur
1BM21CS157

- Observation

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

28). Output website

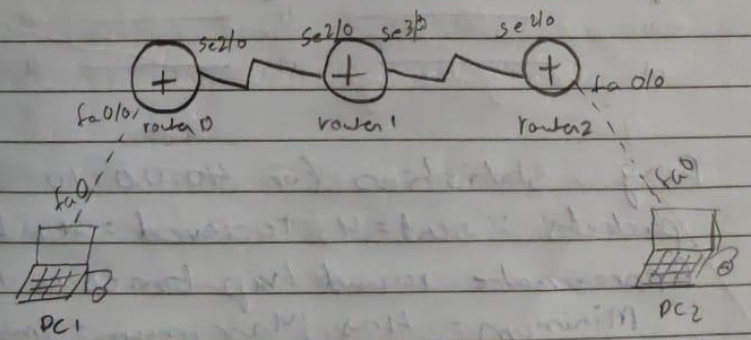
LS



Web browser
url | http://example.com
cisco packet tracer
Rahul Shirur
1BM21CS157

29) Configure RIP routing protocol in Routers.

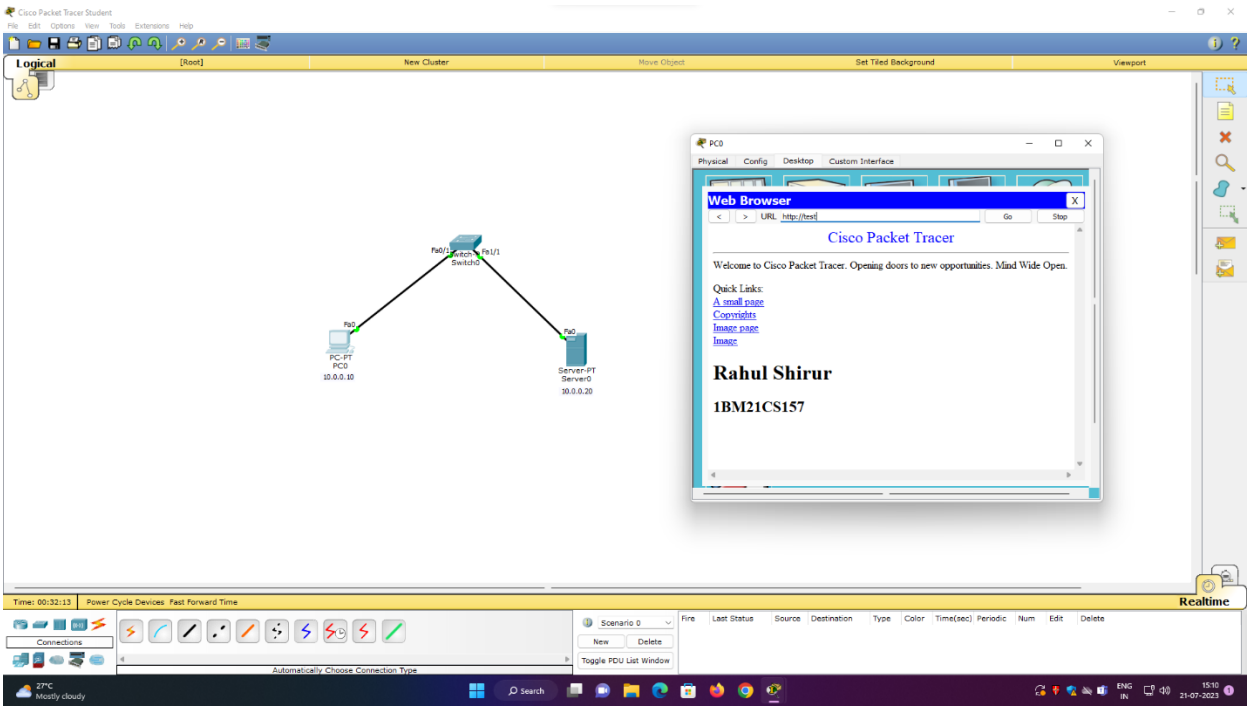
- Create topology -



Steps

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

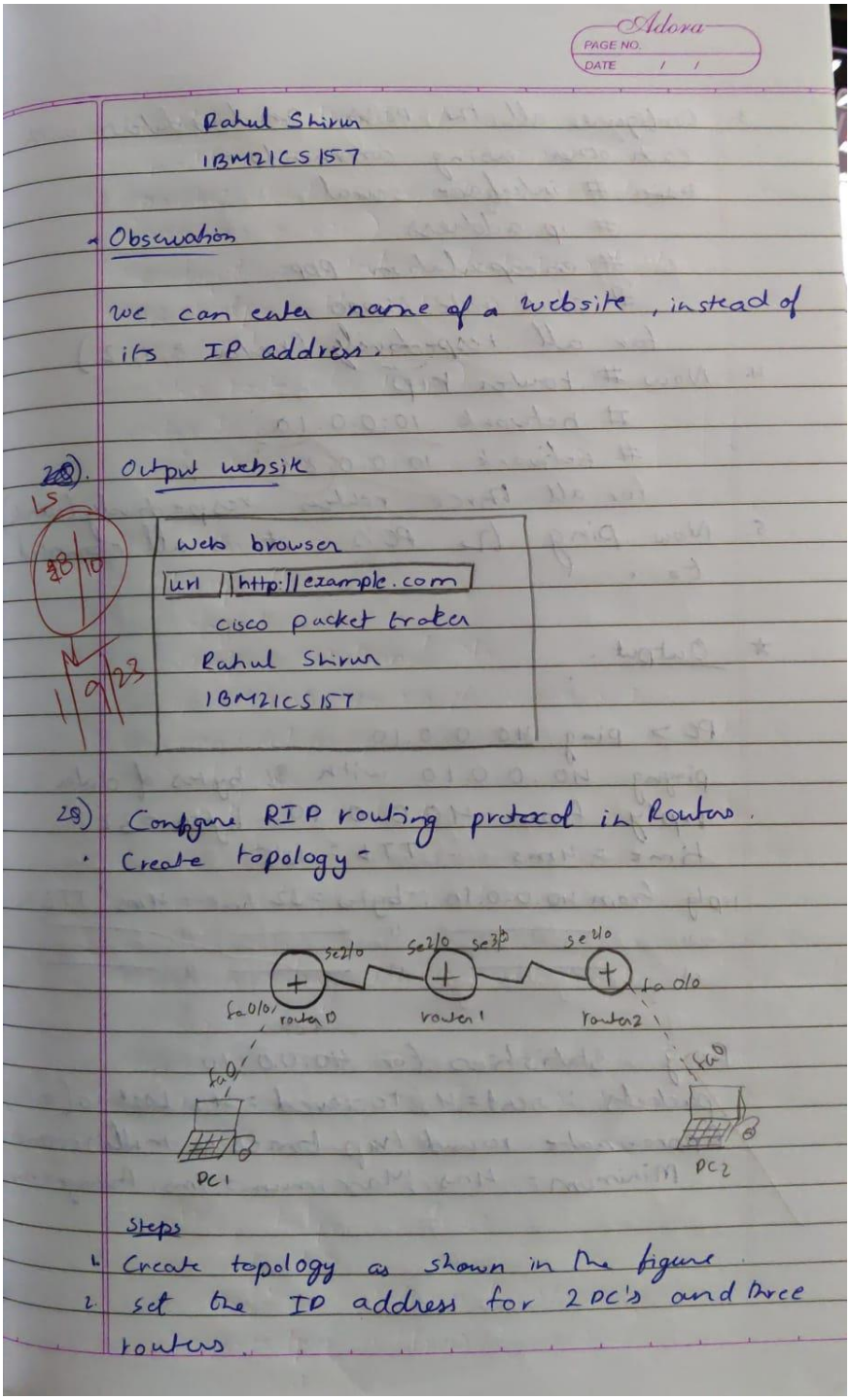
TOPOLOGY & OUTPUT:



WEEK 6

Configure RIP routing Protocol in Routers.

OBSERVATION:



DATE / /

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 (1 and 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, recieved = 4, Lost = 0 (0% loss)
Approximate round trip times in milliseconds
Minimum = 4ms Maximum = 7ms, Average = 4ms

Adora
PAGE NO.
DATE / /

* Commands:

① configure router (plan ips) -

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.

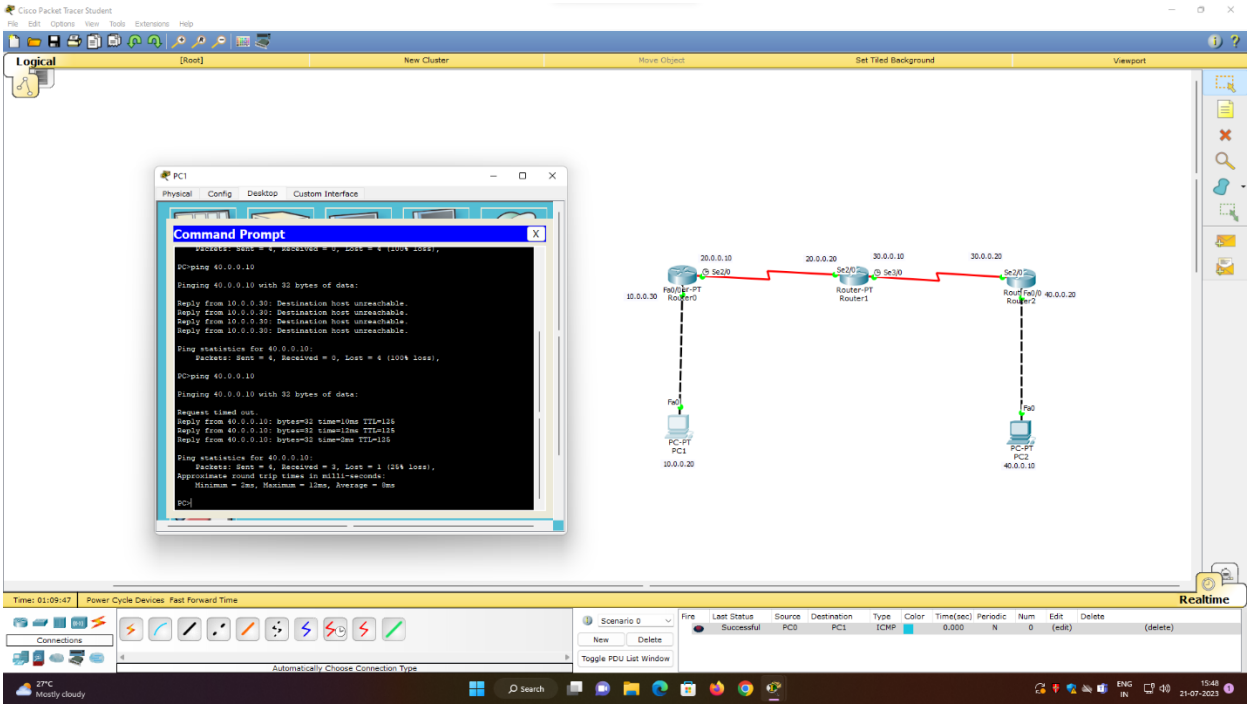
② encapsulation:
interface serial 2/0
encapsulation ppp.
clock rate. 64000
no shut.

③ for rip:
router rip.
network 10.00.0
network 20.0.0.0

8/10

1/9/23

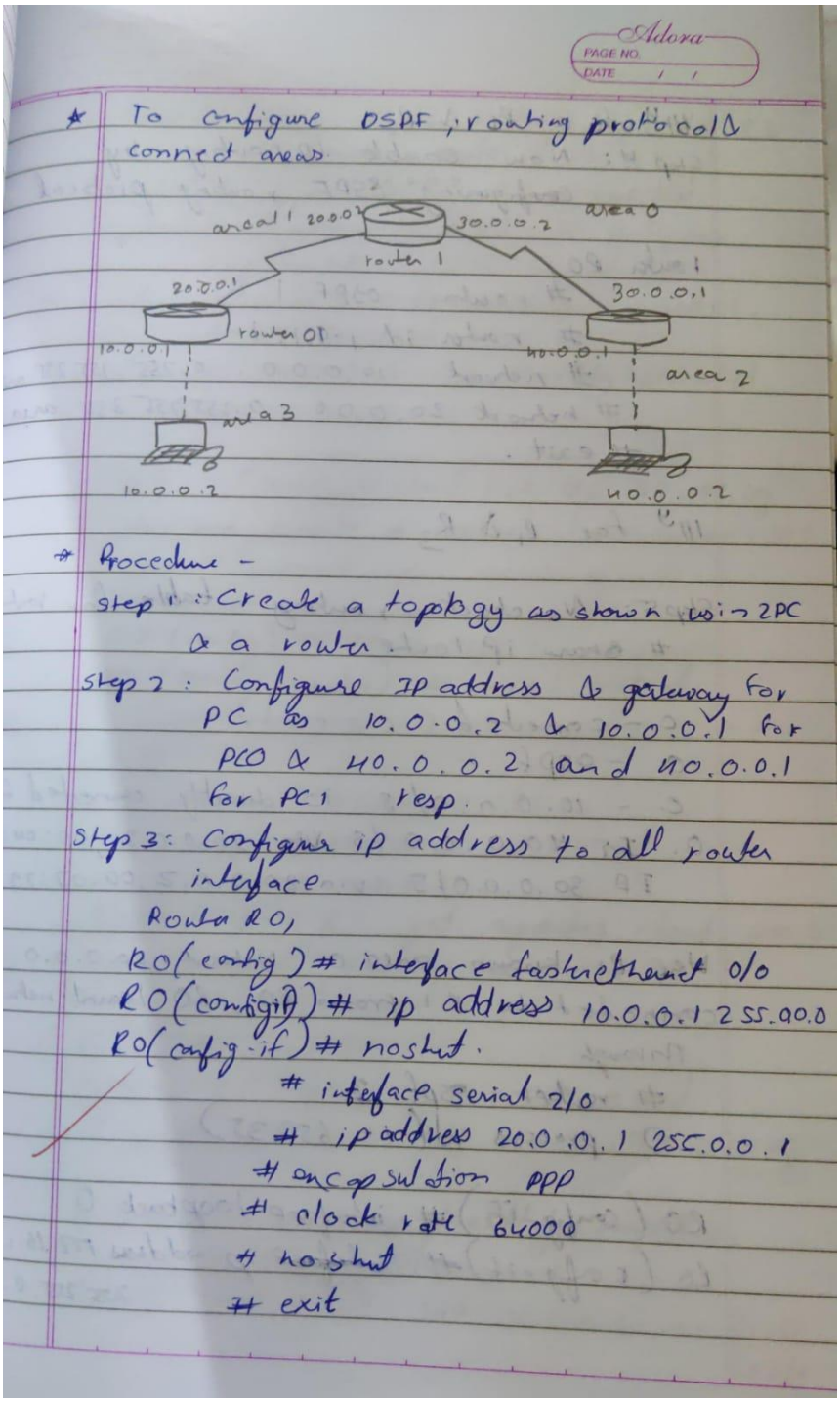
TOPOLOGY & OUTPUT:



WEEK 7

Configure OSPF routing protocol.

OBSERVATION:



Adora
PAGE NO.
DATE / /

III^U for R1 & R2

Step 4: Now enable ip routing by configuring OSPF routing protocol

router R0,
router, ospf 1
router id 1.0.1.1
network 10.0.0.0 . 0.255.255.255 area 0
network 20.0.0.0 0.255.255.255 area 1
exit.

III^U for R1 & R3

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

C - connected
O - ospf
C - 10.0.0.0/8 is directly connected fa0/0
O. Fa 40.0.0.0/8 via 20.0.0.2, 00:04:23 fa
FA 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) # 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 I A 20.0.0.0/8 via 30.0.0.2, 00:18:58, se 2/0

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

C 30.0.0.0/8 is directly connected, se 2/0.

Step 7 Create virtual link b/w R0, R1, by this we create a virtual link in area 340

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 IA 20.0.0.0/8 via 30.0.0.2 00:01:05 se 2/0

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

O IA 10.0.0.0/8 via 30.0.0.2 00:01:56 se 2/0

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, lost = 0

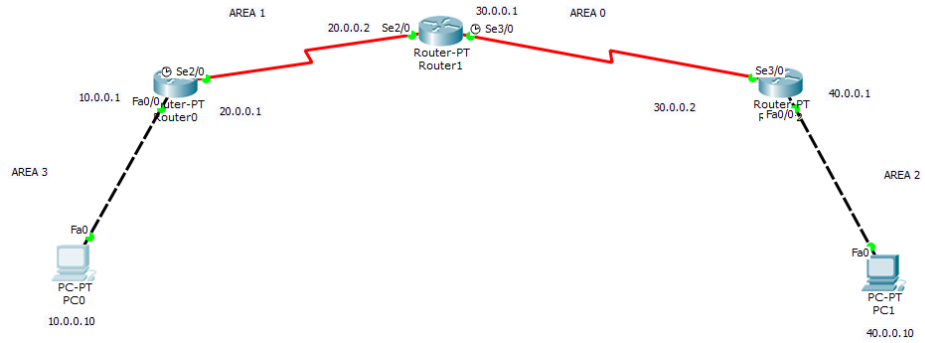
approx round trip times in ms

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

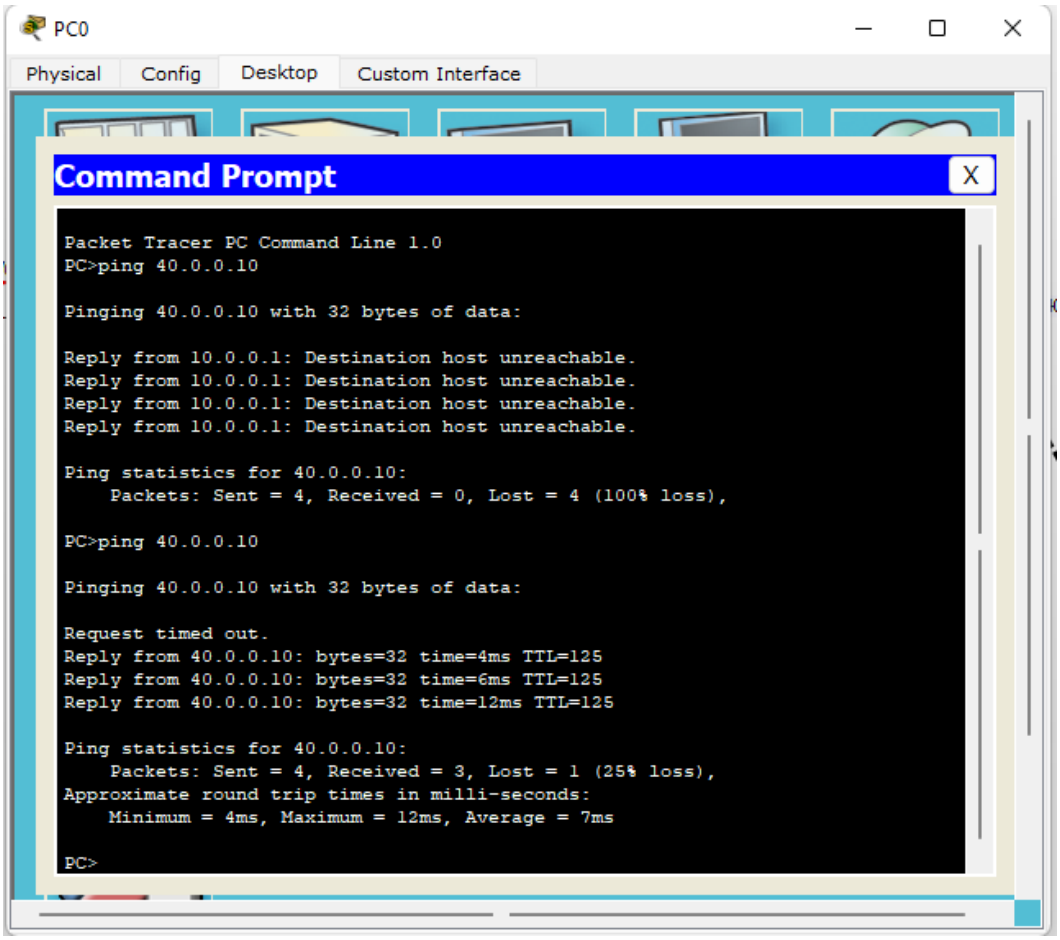
8/10 11/9/23

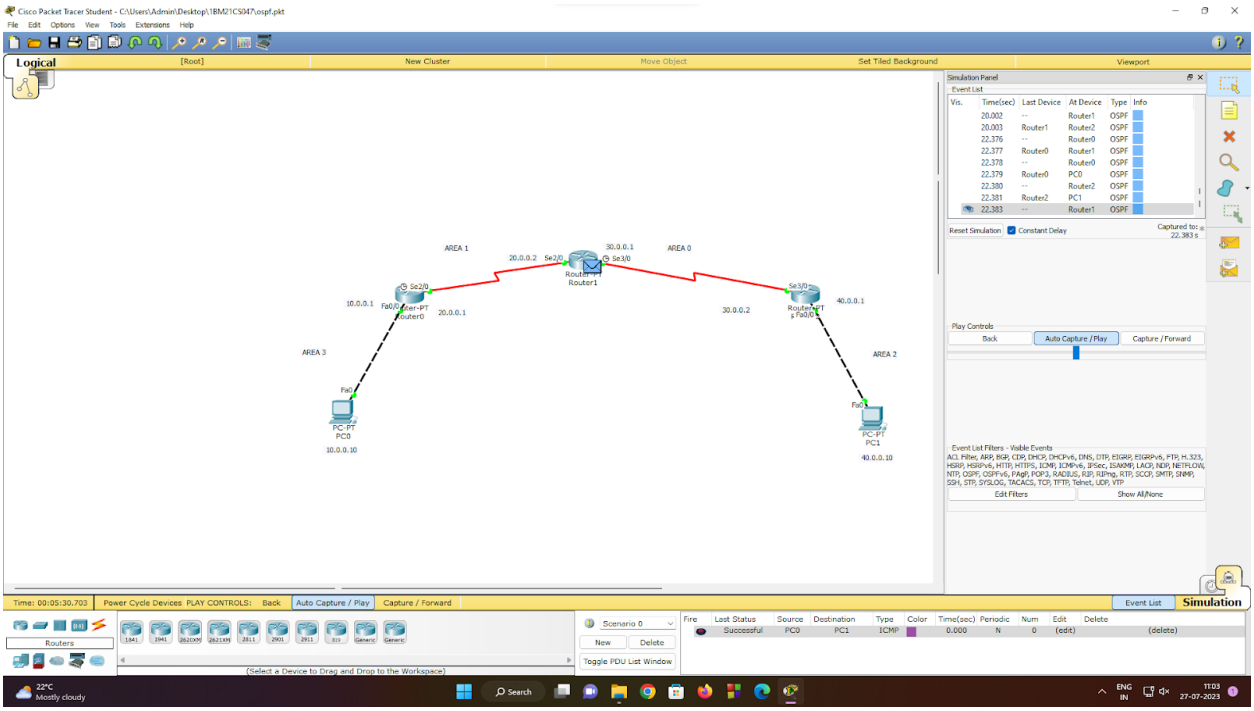
LS

TOPOLOGY:



OUTPUT:

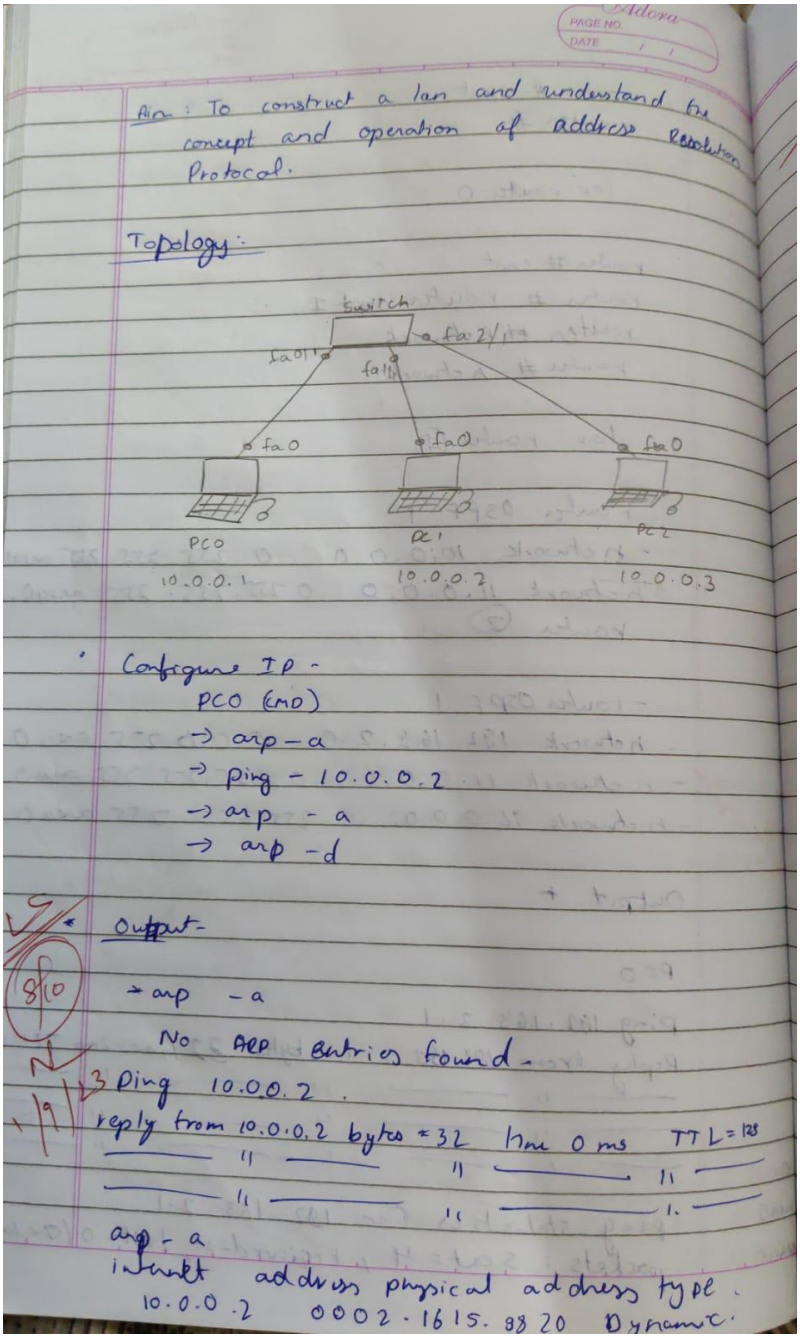



Cisco Packet Tracer Student - C:\Users\Admin\Desktop\18M21CS047\ospf.pkt
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
AREA 1
AREA 0
AREA 3
AREA 2
20.0.0.2
Se2/0
30.0.0.1
Se3/0
Router-PT
Router1
Router0
10.0.0.1
20.0.0.1
30.0.0.2
40.0.0.1
PC-PT
PC0
10.0.0.10
PC1
40.0.0.10
Simulation Panel
Event List
Reset Simulation
Constant Delay
Play Controls
Back
Auto Capture / Play
Capture / Forward
Edit Filters
Show All/None
Time: 00:05:30.703
Power Cycle Devices PLAY CONTROLS: Back Auto Capture / Play Capture / Forward
Event List
Simulation
Routers
Scenario 0
New
Delete
Toggle PDU List Window
(Select a Device to Drag and Drop to the Workspace)
Fire
Last Status
Source
Destination
Type
Color
Time(sec)
Periodic
Num
Edit
Delete
Successful
PC0
PC1
ICMP
0.000
N
0
(edit)
(delete)
22°C
Mostly cloudy
Search
ENG
IN
11:03
27-07-2023

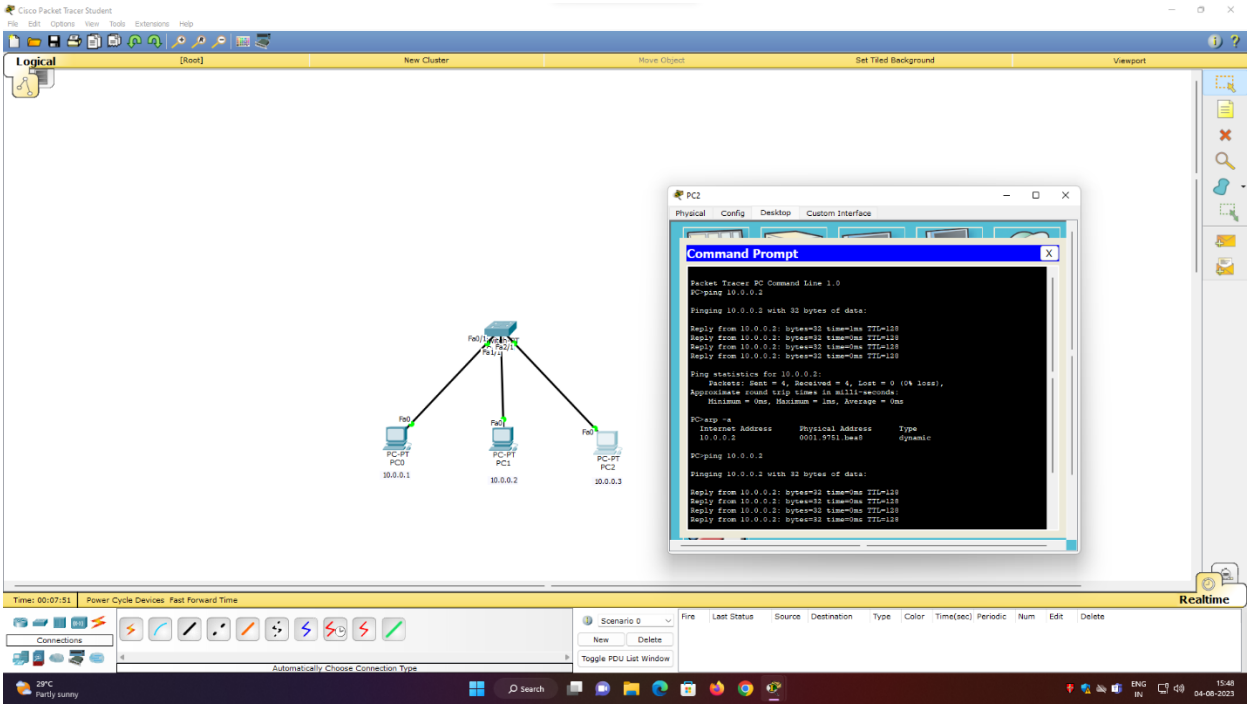
WEEK 8

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

OBSERVATION:



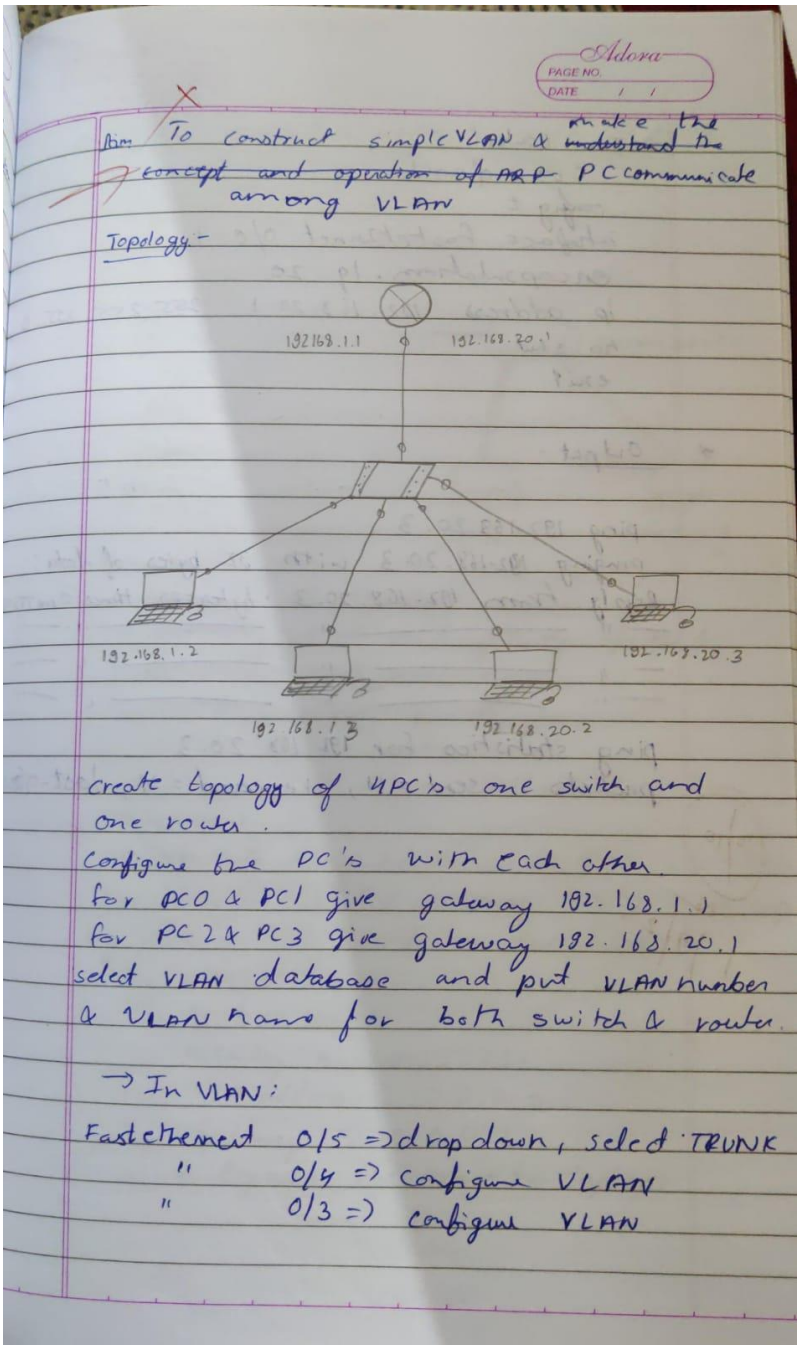
TOPOLOGY & OUTPUT:



WEEK 9

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

OBSERVATION:



→ In Router
CLI of router
config t
interface fastethernet 0/0.1
encapsulation dq 20
ip address 192.168.20.1 255.255.255.0
no shut
exit

* Output -

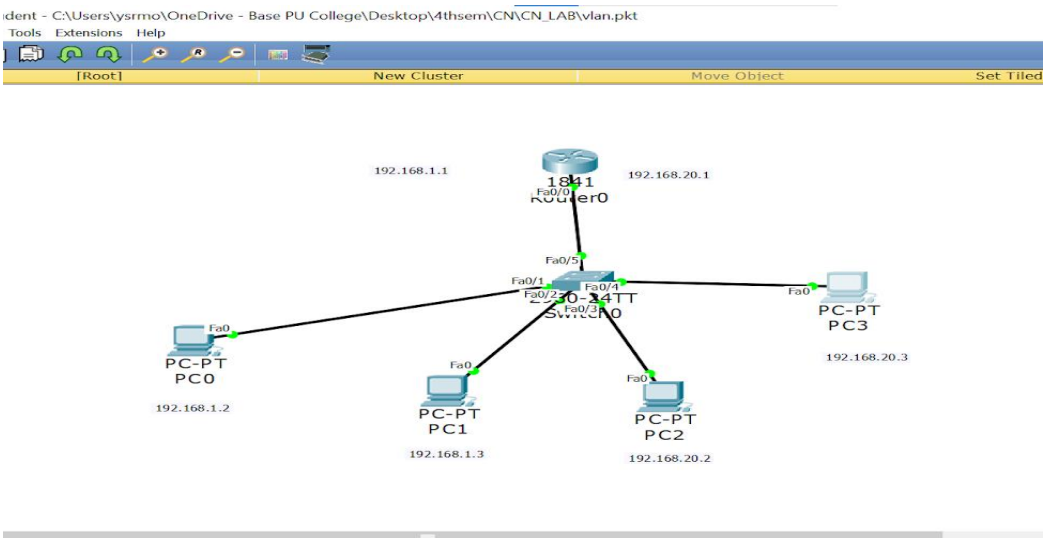
ping 192.168.20.3
pinging 192.168.20.3 with 32 bytes of data:
Reply from 192.168.20.3 : bytes=32 time=0ms TTL=
" " " "
" " " "
" " " "

ping statistics for 192.168.20.3
packets : sent = 4, recieved = 4, lost = 0(0

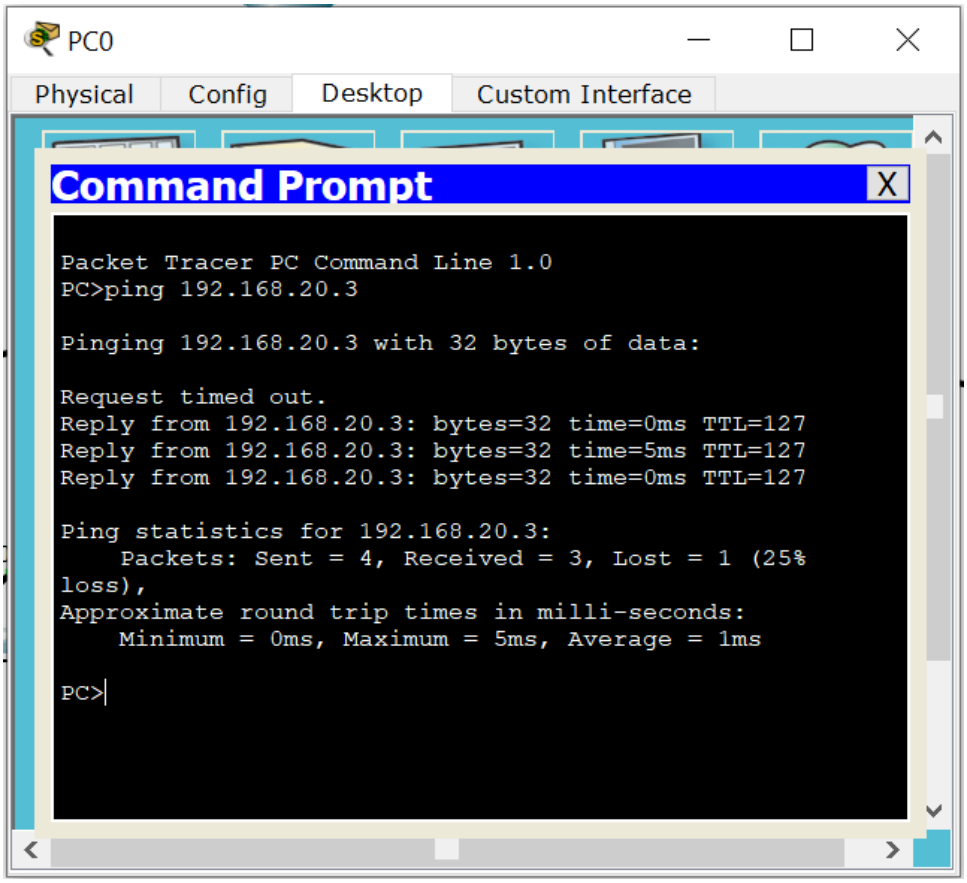
10/10

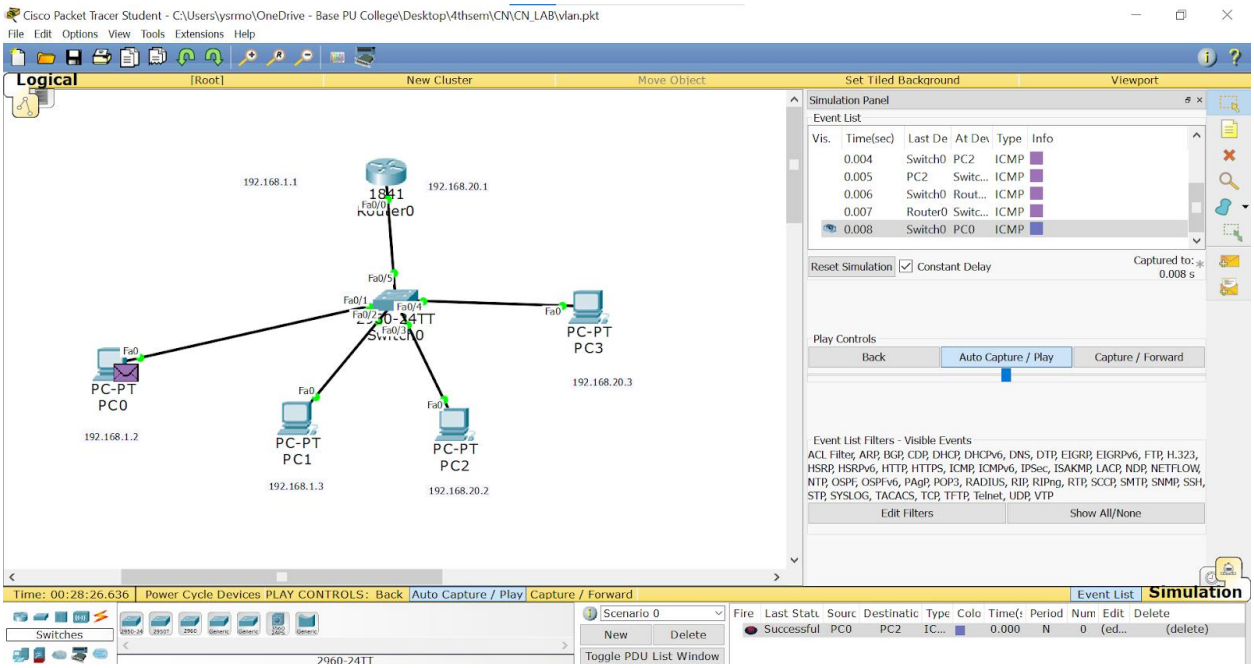
19/23

TOPOLOGY:



OUTPUT:

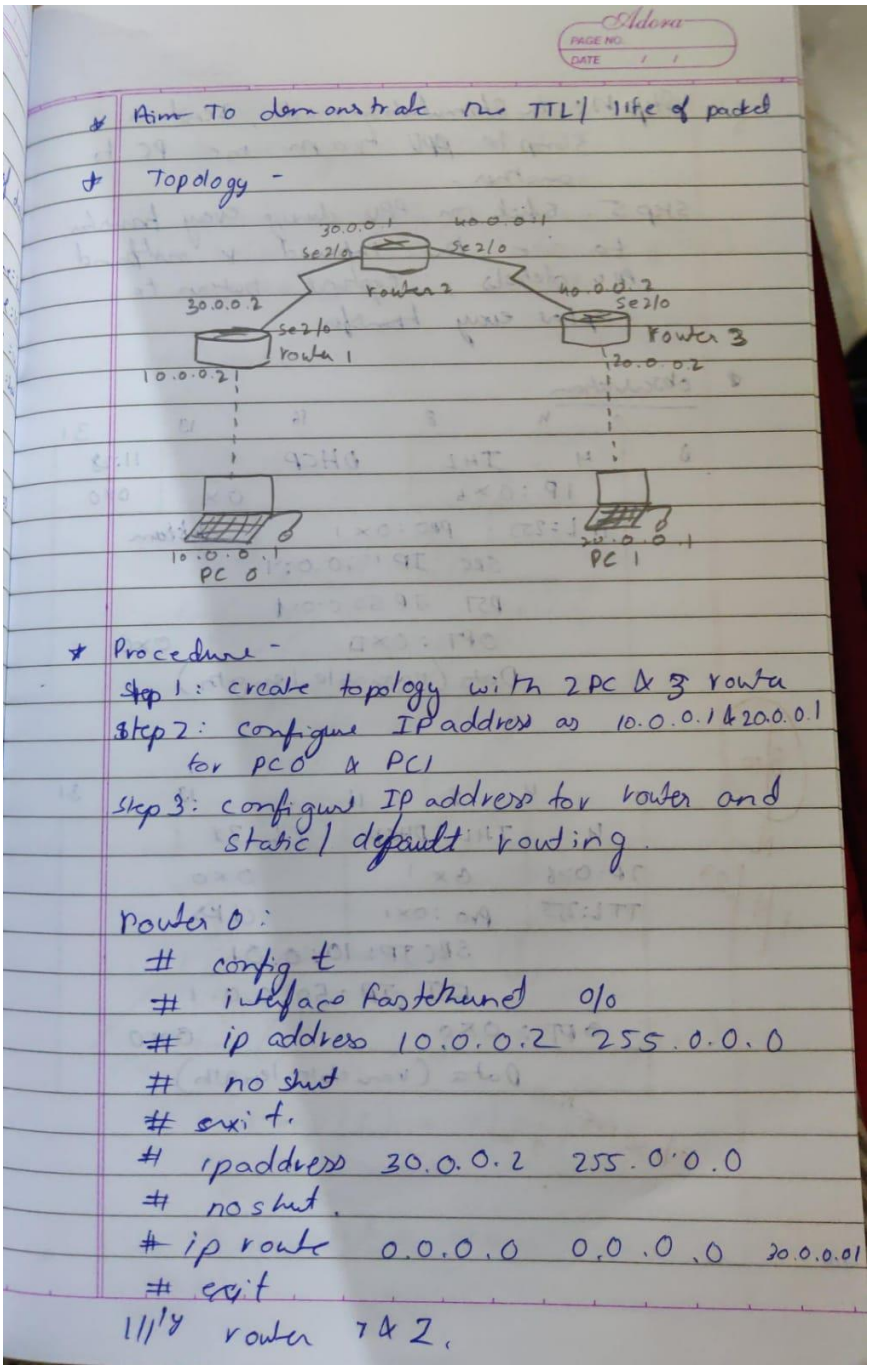



Cisco Packet Tracer Student - C:\Users\ysrmo\OneDrive - Base PU College\Desktop\4thsem\CN\CN_LAB\vlan.pkt
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
192.168.1.1
192.168.20.1
Router0
Fa0/5
Fa0/1
Fa0/4
Fa0/2
Fa0/3
Fa0
PC-PT PC0
192.168.1.2
PC-PT PC1
192.168.1.3
PC-PT PC2
192.168.20.2
PC-PT PC3
192.168.20.3
Simulation Panel
Event List
Vis. Time(sec) Last De At De Type Info
0.004 Switch0 PC2 ICMP
0.005 PC2 Switc... ICMP
0.006 Switch0 Rout... ICMP
0.007 Router0 Switc... ICMP
0.008 Switch0 PC0 ICMP
Reset Simulation
Constant Delay
Captured to: 0.008 s
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ACL Filter, ARP, BGP, CDP, DHCP, DHCPv6, DNS, DTP, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, RADIUS, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters
Show All/None
Time: 00:28:26.636
Power Cycle Devices PLAY CONTROLS: Back
Auto Capture / Play
Capture / Forward
Event List
Simulation
Switches
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Statu Sourc Destinatio Type Colo Time(s Period Num Edit Delete
Successful PC0 PC2 IC... 0.000 N 0 (ed... (delete)

WEEK 10

Demonstrate the TTL/ Life of a Packet.

OBSERVATION:



Step 4: In simulation 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.

* Observation:

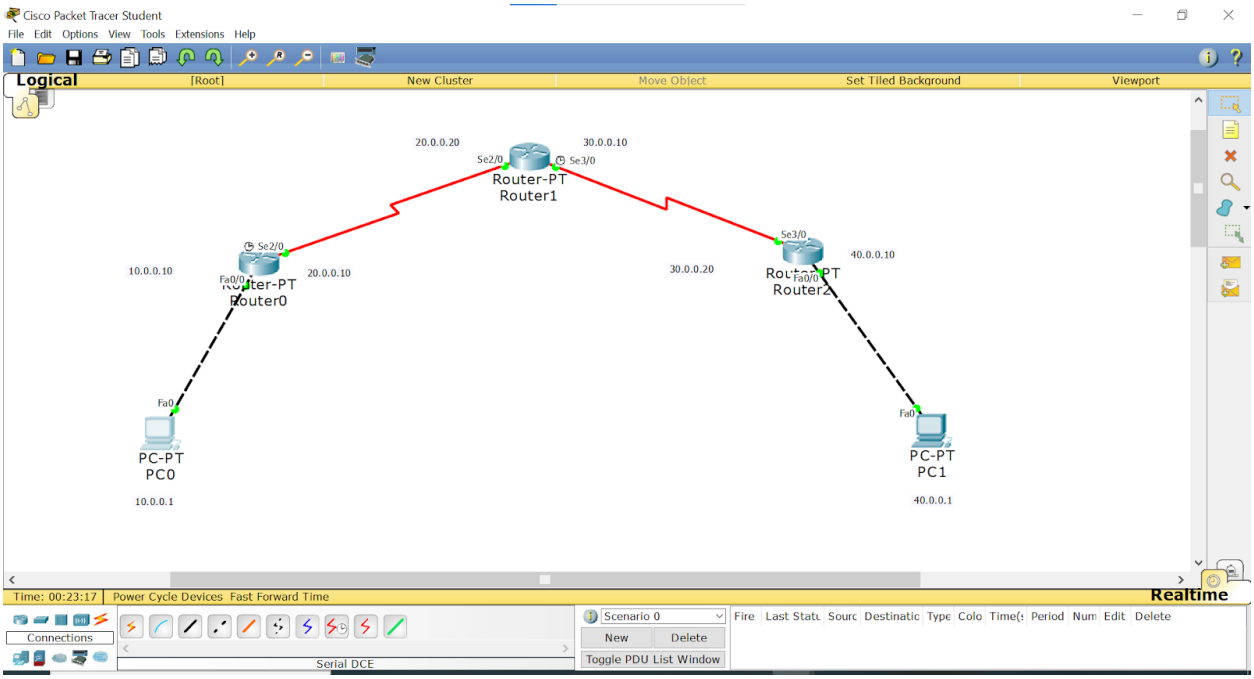
0	4	8	16	19	31
4	IHL	DHCP			11:28
IP: 0x6			0x		0x0
TTL: 255	PRO: 0x1		Checksum		
SRC IP 10:0:0::1					
DST IP 50:0:0::1					
OPT: 0xD					0x0
Data (variable length)					

8/10

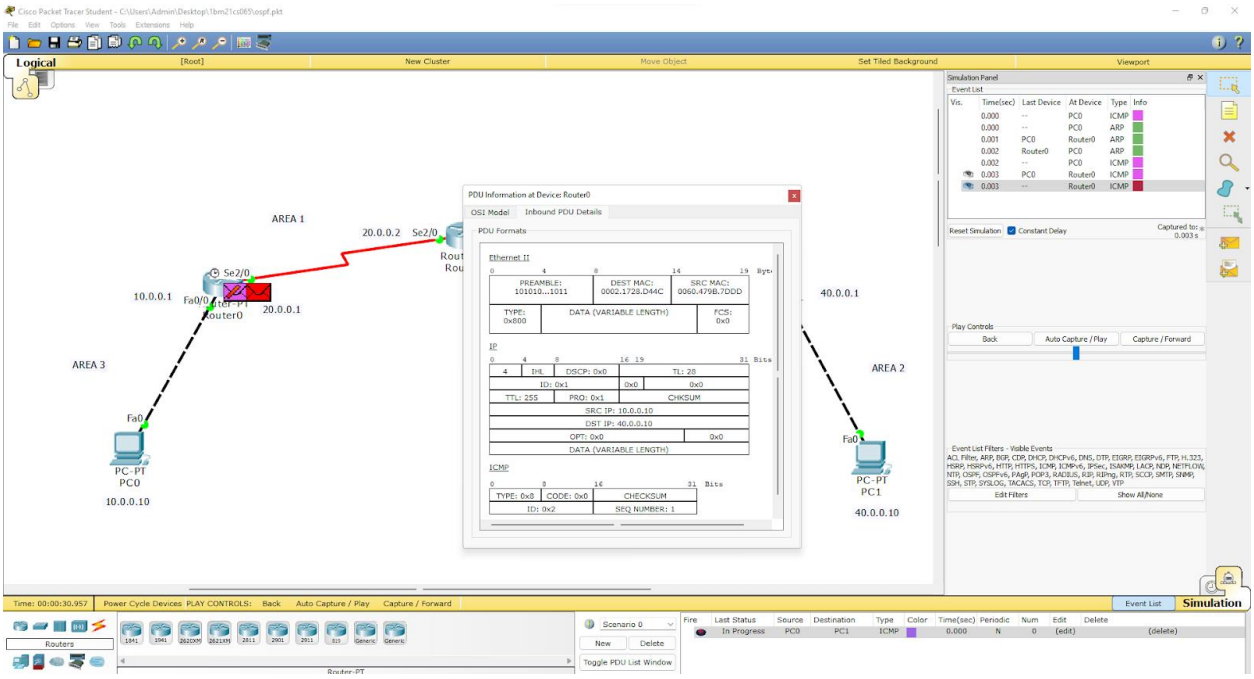
0	4	8	16	19	31
4	IHL	DHCP	TL: 28		
IP: 0x6	0x1		0x0		
TTL: 255	Pro: 0x1		chks		
SRC IP: 10.0.0.1					
DST IP: 50.0.0.1					
OPT: 0x0					0x0
Data (variable length)					

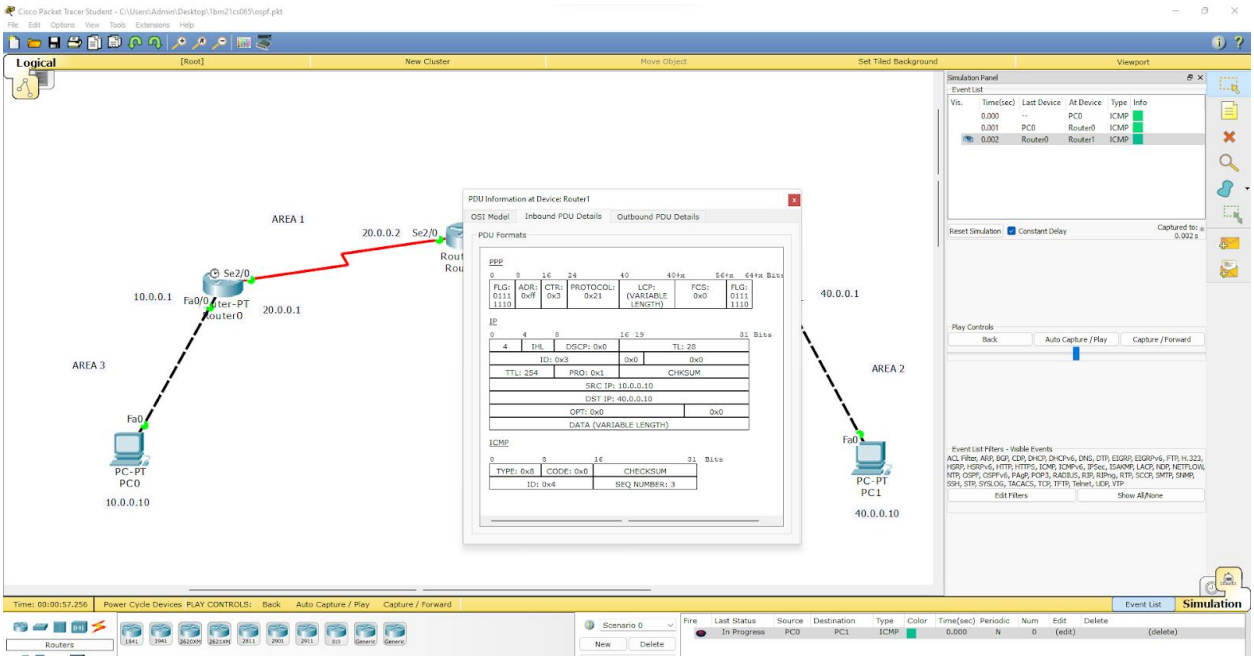
Observation:

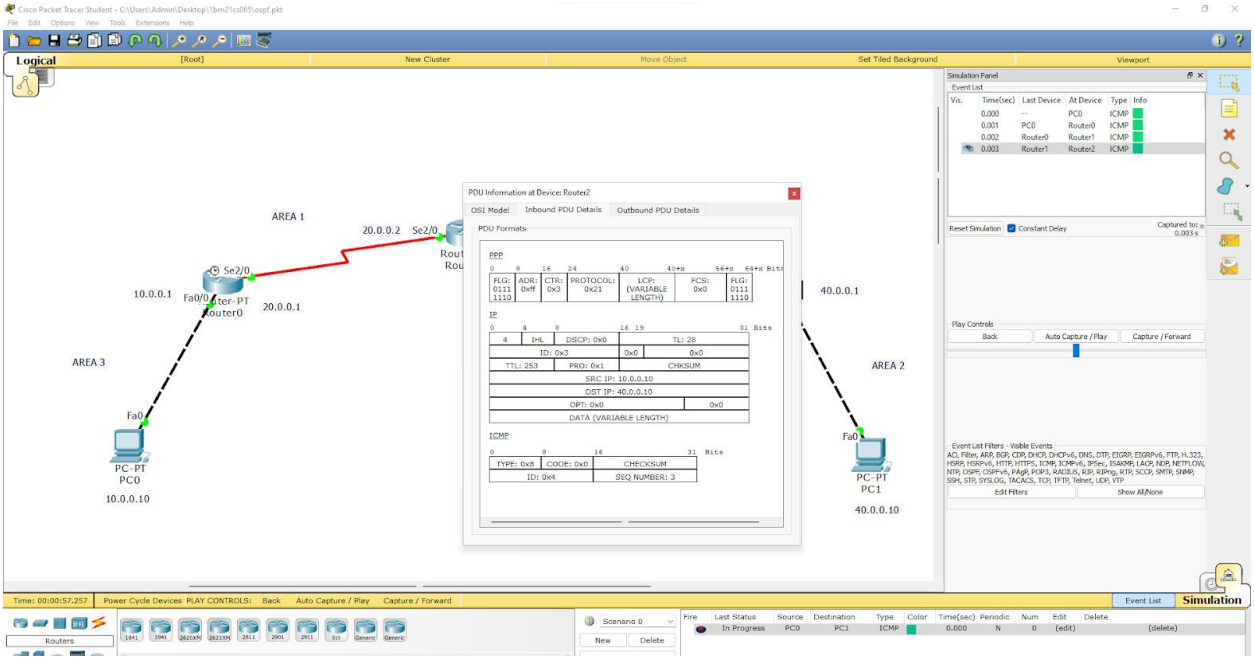
TOPOLOGY:

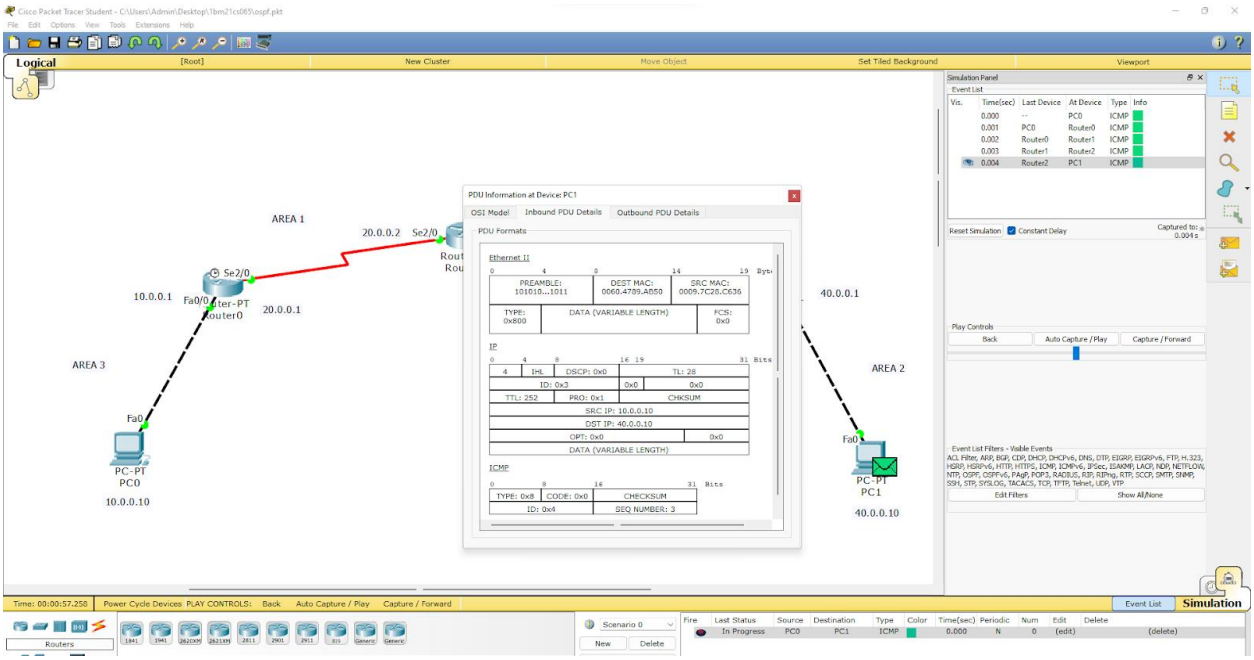


OUTPUT:




AREA 1
AREA 2
AREA 3
20.0.0.2
Se2/0
10.0.0.1
Fa0/0
Router-PT
Router0
20.0.0.1
40.0.0.1
Fa0
PC-PT
PC0
10.0.0.10
PC-PT
PC1
40.0.0.10
PDU Information at Device: Router1
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
PPP
IP
ICMP
SRC IP: 10.0.0.10
DST IP: 40.0.0.10
DATA (VARIABLE LENGTH)
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Simulation Panel
Event List
Play Controls
Back
Auto Capture / Play
Capture / Forward
Power Cycle Devices
PLAY CONTROLS:
Event List
Simulation

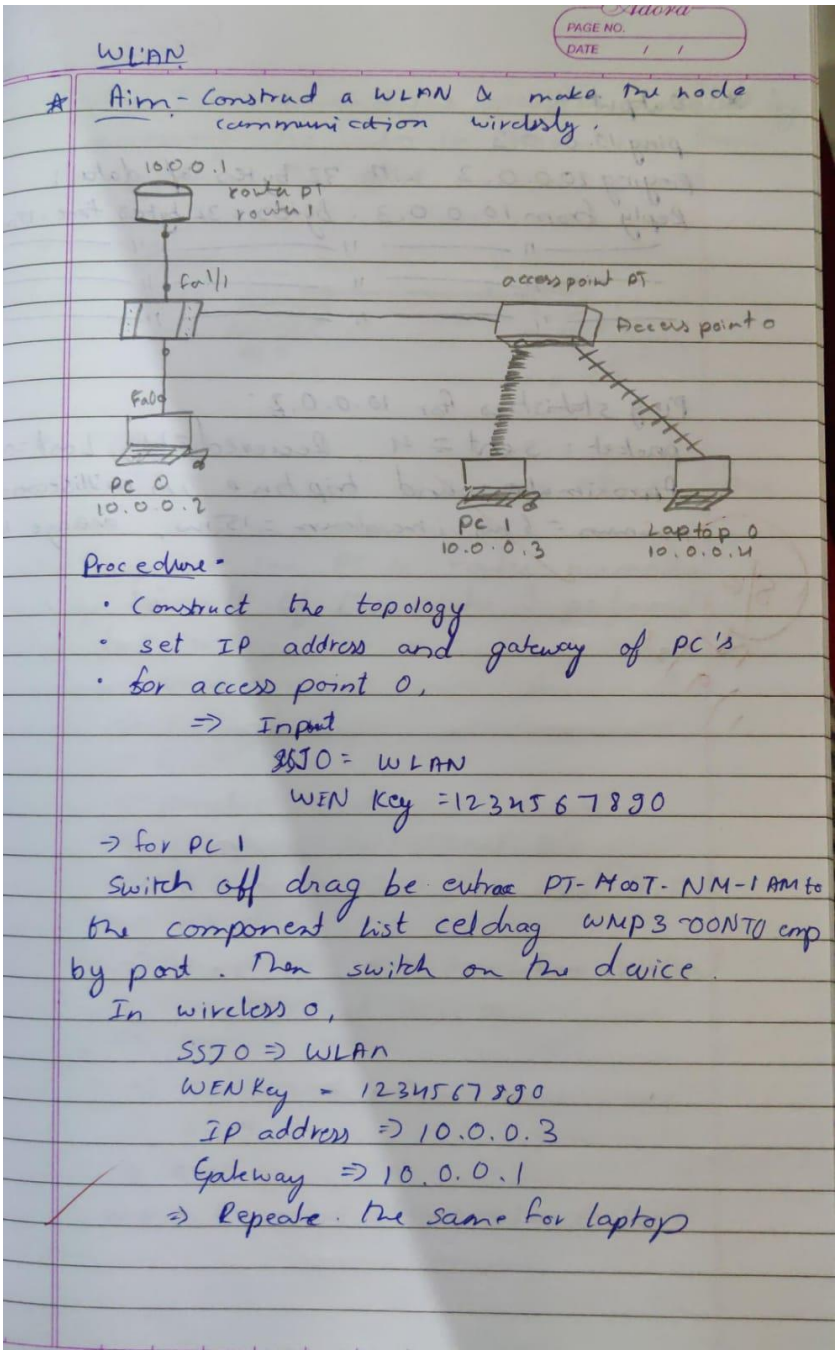

AREA 1
AREA 2
AREA 3
20.0.0.2
Se2/0
10.0.0.1
Fa0/0
Router-PT
Router0
20.0.0.1
40.0.0.1
Fa0
PC-PT
PC0
10.0.0.10
PC-PT
PC1
40.0.0.10
PDU Information at Device: Router2
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
PPP
IP
ICMP
SRC IP: 10.0.0.10
DST IP: 40.0.0.10
DATA (VARIABLE LENGTH)
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Simulation Panel
Event List
Play Controls
Back
Auto Capture / Play
Capture / Forward
Power Cycle Devices
PLAY CONTROLS:
Event List
Simulation


Logical
AREA 1
AREA 2
AREA 3
20.0.0.2
Se2/0
10.0.0.1
Fa0/0
20.0.0.1
40.0.0.1
Fa0
PC-PT
PC0
10.0.0.10
PC-PT
PC1
40.0.0.10
PDU Information at Device: PC1
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
Ethernet II
PREAMBLE: 101010...1011
DEST MAC: 0060.4789.AB50
SRC MAC: 0009.7C28.C636
TYPE: 0x800
DATA (VARIABLE LENGTH)
FCS: 0x0
IP
SRC IP: 10.0.0.10
DST IP: 40.0.0.10
OPT: 0x0
ICMP
TYPE: 0x8
CODE: 0x0
CHECKSUM
ID: 0x4
SEQ NUMBER: 3
Simulation Panel
Event List
Play Controls
Back
Auto Capture / Play
Capture / Forward
Edit Filters
Show All/None
Time: 00:00:57.258
Power Cycle Devices
PLAY CONTROLS:
Event List
Simulation
Routers
Scenario 0
New
Delete
In Progress
ICMP

WEEK 11

To construct a WLAN and make the nodes communicate wirelessly

OBSERVATION:



* 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 TTL

" " " "

" " " "

" " " "

Ping statistics for 10.0.0.3 :

packet : sent = 4 , Recieved = 4, Lost = 0

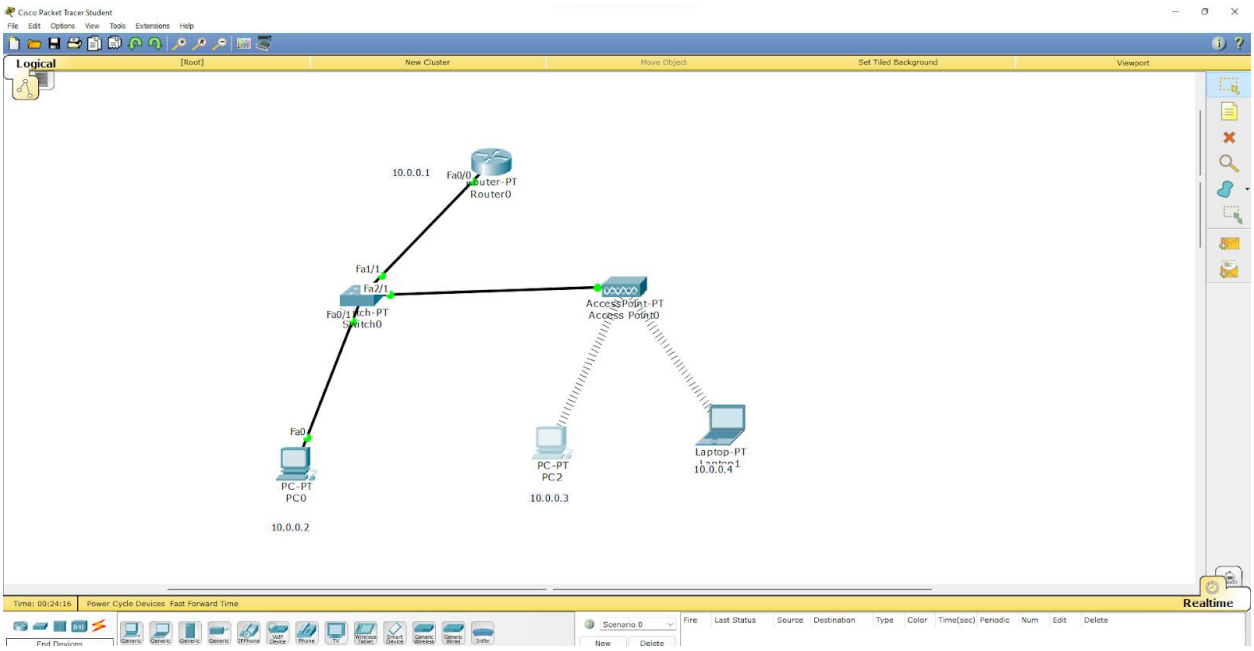
Approximate round trip time in milliseconds :

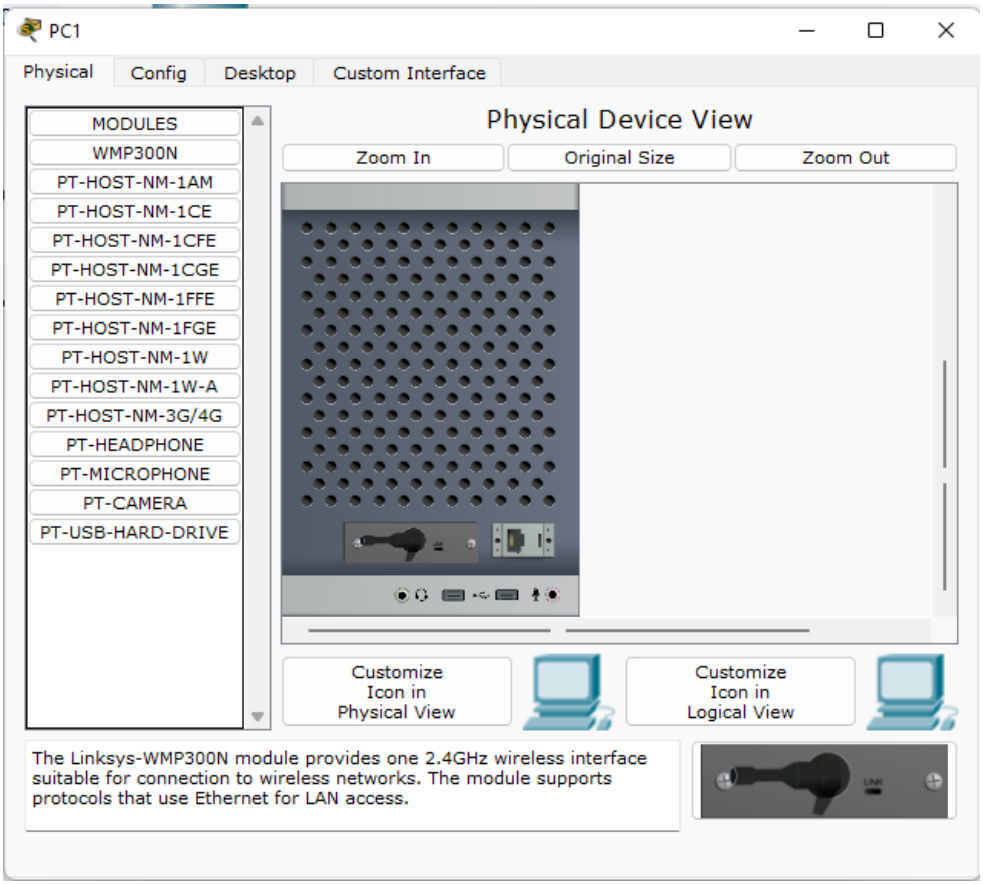
minimum = 6 ms , maximum = 15 ms, average 10ms

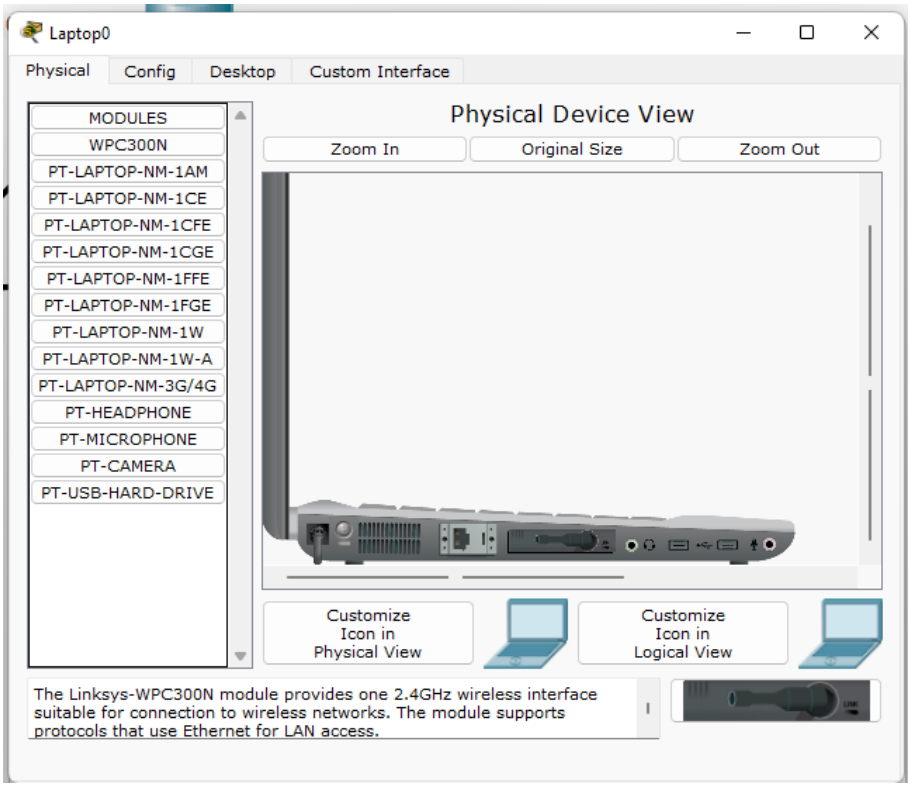
8/10

11/9/23

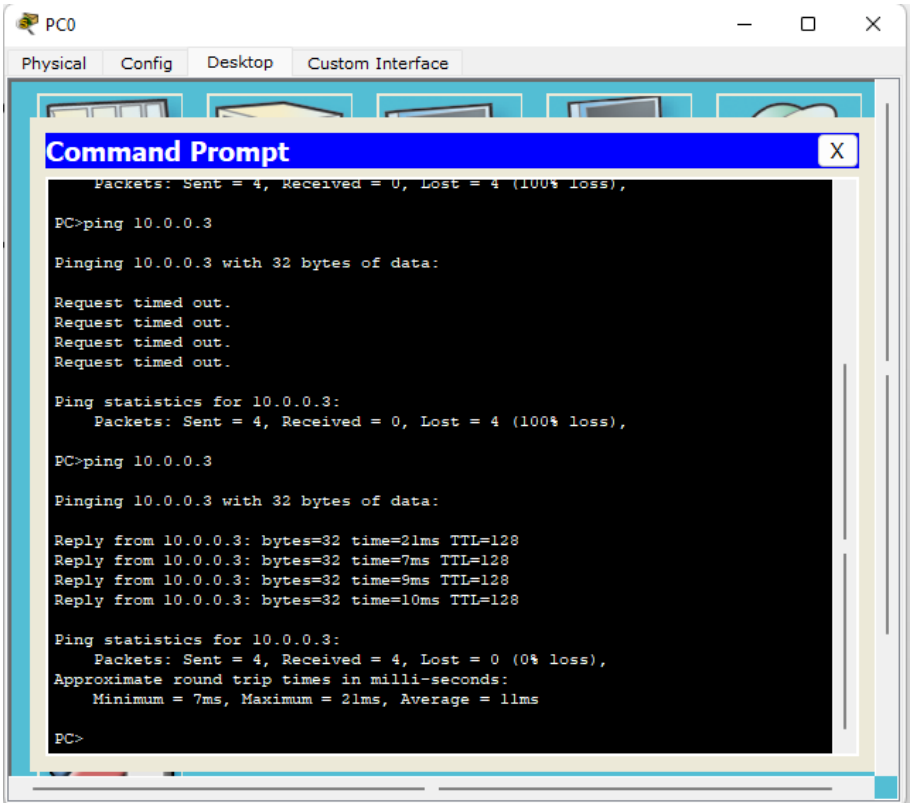
TOPOLOGY:


Logical
10.0.0.1
Router-PT
Router0
Fa1/1
Fa2/1
Switch-PT
Switch0
Access Point-PT
Access Point0
PC-PT
PC0
10.0.0.2
PC-PT
PC2
10.0.0.3
Laptop-PT
10.0.0.4
Realtime


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.

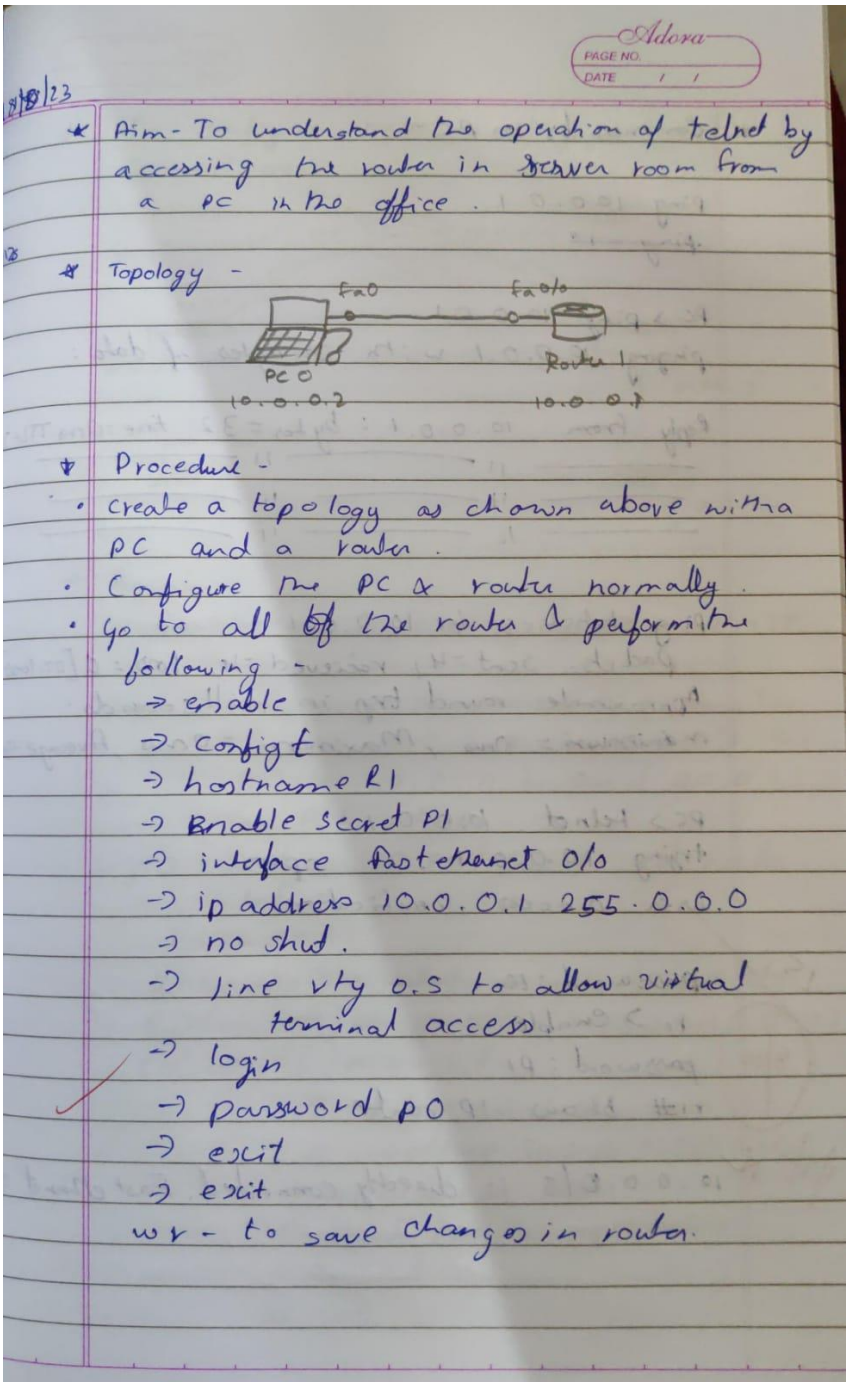
OUTPUT:

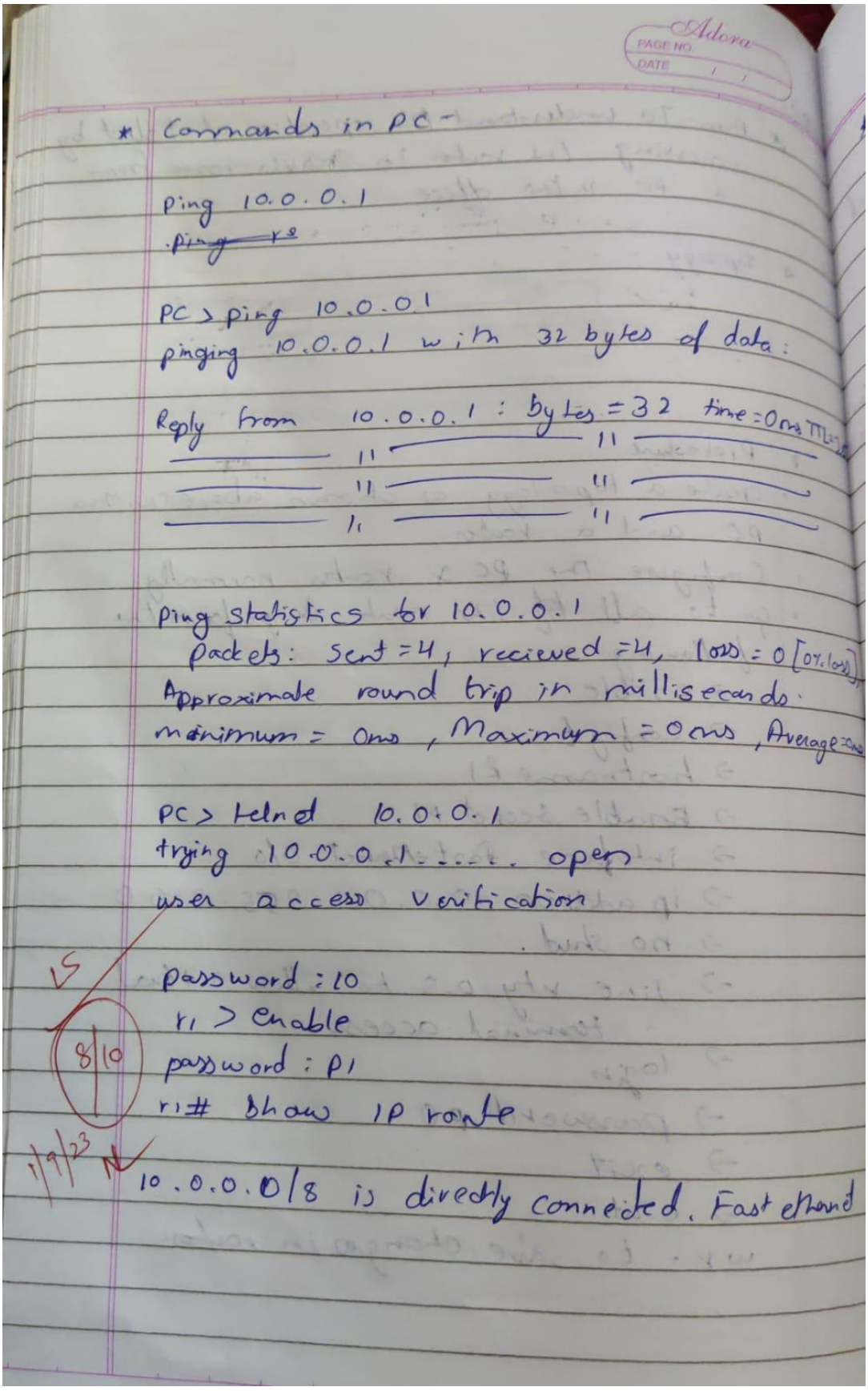

PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>ping 10.0.0.3
Pinging 10.0.0.3 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 10.0.0.3:
Packets: Sent = 4, Received = 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

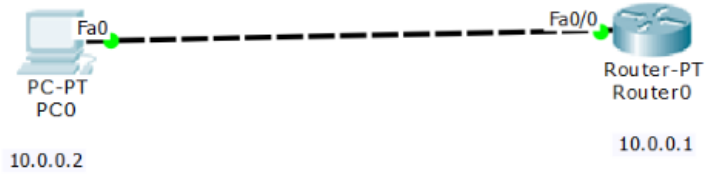
To understand the operation of TELNET by accessing the router in server room from a PC in IT office.

OBSERVATION:

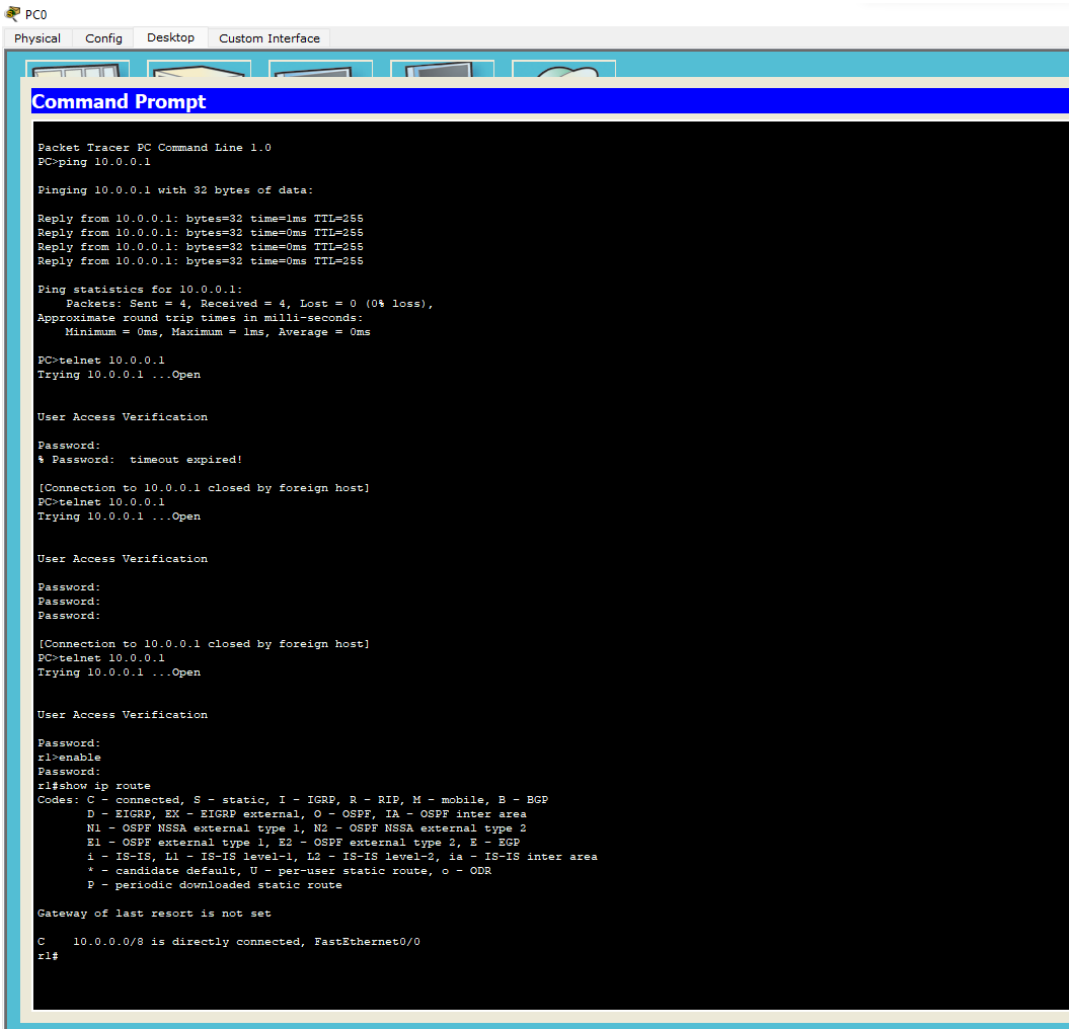



Adora
PAGE NO.
DATE
* Commands in PC -
ping 10.0.0.1
ping -s
PC > ping 10.0.0.1
pinging 10.0.0.1 with 32 bytes of data:
Reply from 10.0.0.1 : bytes = 32 time = 0ms TTL=
Ping statistics for 10.0.0.1
Packets: Sent = 4, recieved = 4, loss = 0 [0% loss]
Approximate round trip in milliseconds:
minimum = 0ms, Maximum = 0ms, Average = 0ms
PC > telnet 10.0.0.1
trying 10.0.0.1 ..... open
user access verification
password : 10
r1 > enable
password : p1
r1# show IP route
10.0.0.0/8 is directly connected, Fast ethernet
8/10
1/9/23

TOPOLOGY:



OUTPUT:



WEEK 13

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

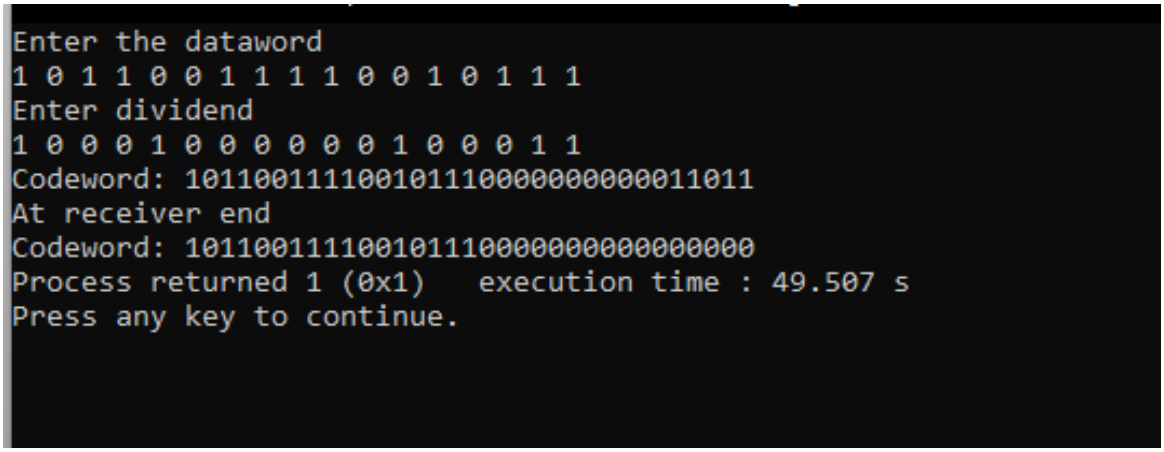
CODE:

```
#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");
```

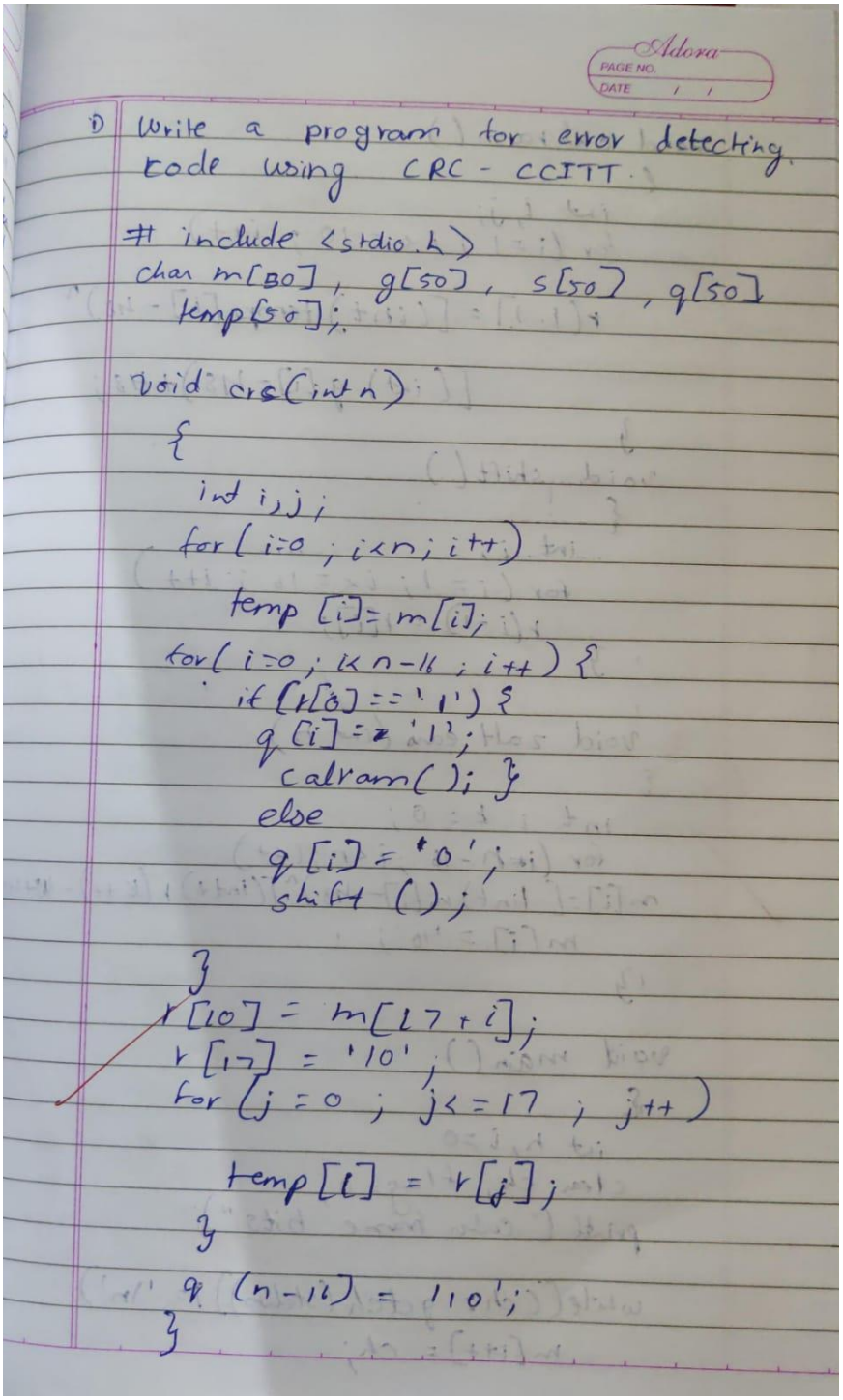
```
for(i=0;i<17;i++)
   scanf("%d",&dd[i]);

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:

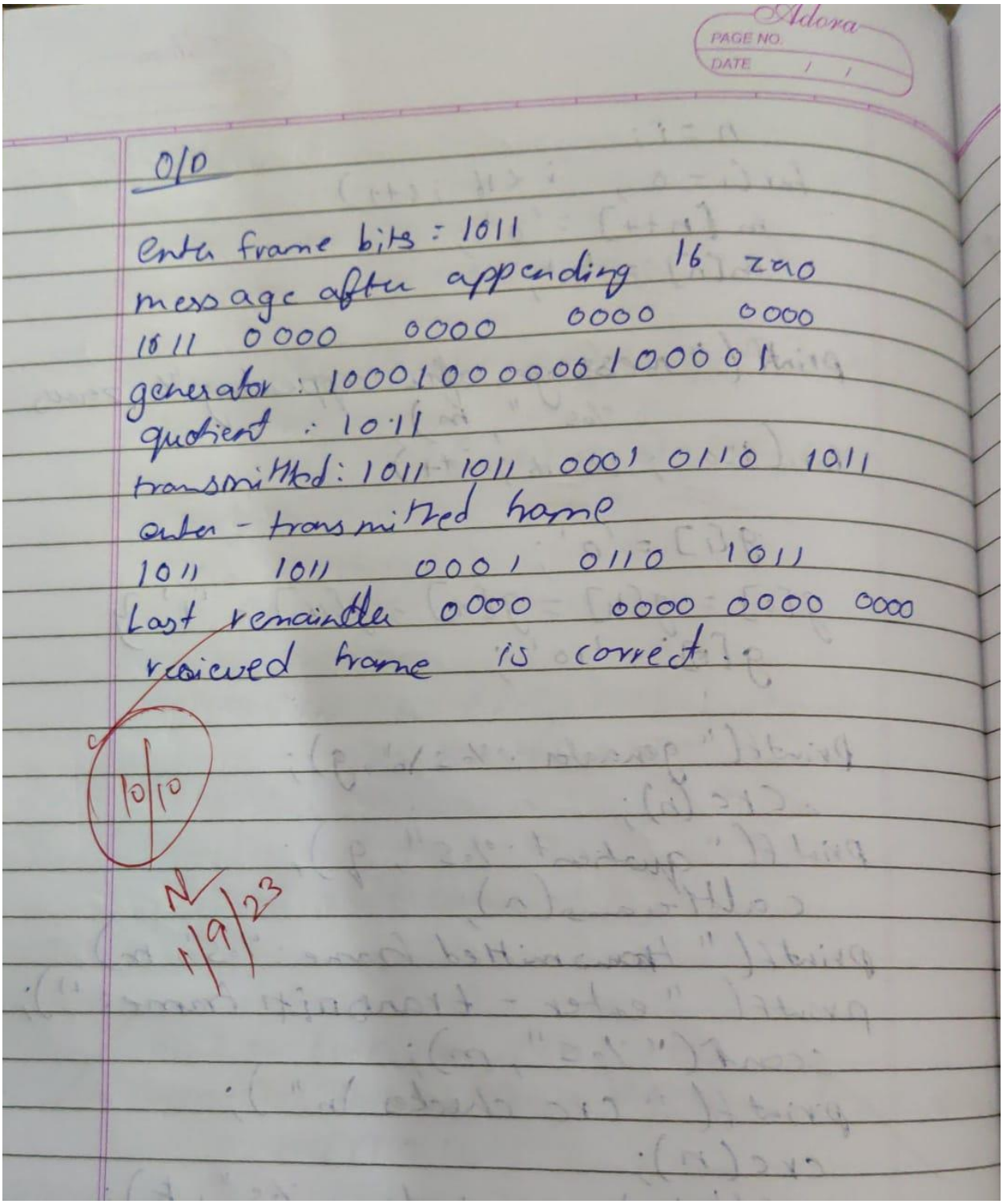


1) Write a program for error detecting code using CRC - CCITT.

```
# include <stdio.h>
char m[80], g[50], s[50], q[50]
  temp[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 [r[0] == '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[i] = r[j];
   }

   q (n-16) = '10';
  }
```

```
void calram()
 {
   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 xalt_cans(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';
 }

void main()
 {
   int n,i=0
  char ch, flag=0;
printf("enter frame bits");

while((ch=getch(stdin))!='\n')
   m[i++]=ch;
```

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

printf("message after appending 16 zeroes
          %s", m);
for(i=0 ; i<=16 ; i++)
    g[i] = '0';
g[0] = g[4] = g[11] = g[16] = '1' }
   g[k] = '10';

printf("generator : %s \n", g);
   crc(n);
printf("quotient : %s", q);
   calltrans(n);
printf("Transmitted frame : %s", m)
printf("enter - transmit frame : ");
   scanf("%s", m);
   printf("crc checkes \n");
   crc(n);
printf("last remainder : %s", k);
   for(i=0 ; i<16 ; i++)
if(r[i] != '0')
   flag = '1';
   else
   continue;
if(flag == '1')
   printf("error ");
   else
Printf("frames all correct");
}
```


Adora
PAGE NO.
DATE / /
0/0
Enter frame bits : 1011
message after appending 16 zero
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
recieved frame is correct.
10/10
1/9/23

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);
```

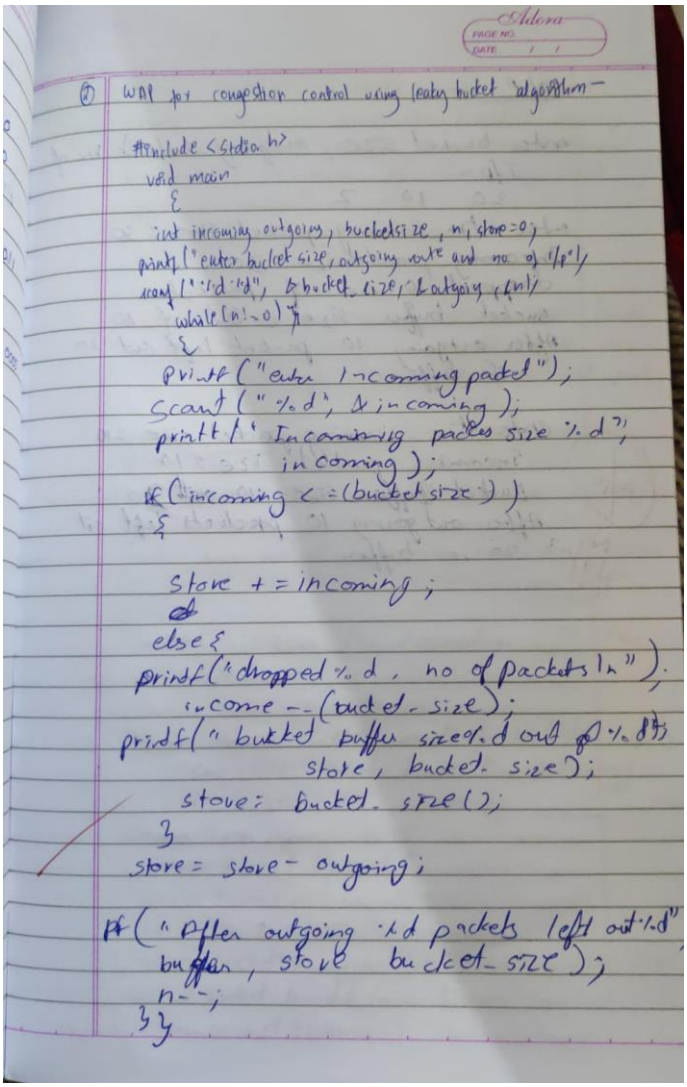
```
    }
    while (remaining < buckets) // Fixed the condition
    {
        if (buckets - remaining > outlets)
        {
            remaining += outlets; // Fixed the calculation
        }
        else
            remaining = buckets;
        printf("Remaining bytes: %d \n", remaining);
    }
  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

OUTPUT:

```
PS D:\VS Code> cd "d:\VS Code\OS\" ; 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:



① WAP for congestion control using leaky bucket algorithm –

```
#include <stdio.h>
void main
{
int incoming, outgoing, bucketsize, n, store=0;
printf("enter bucket size, outgoing rate and no. of i/p");
scanf("%d %d %d", &bucket_size, &outgoing, &n);
while(n!=0)
{
printf("enter incoming packet");
scanf("%d", &incoming);
printf("Incoming packet size %d",
        incoming);
if(incoming <=(bucketsize))
{

store += incoming;
}
else{
printf("dropped %d, no of packets\n"),
    income -=(bucket_size);
printf("bucket buffer size%d out of %d\n",
        store, bucket_size);
store= bucket_size();
}
store= store- outgoing;

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

o/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 of 20
After out going 10 packets left out 20 in buffer.

10/10

11/9/23

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

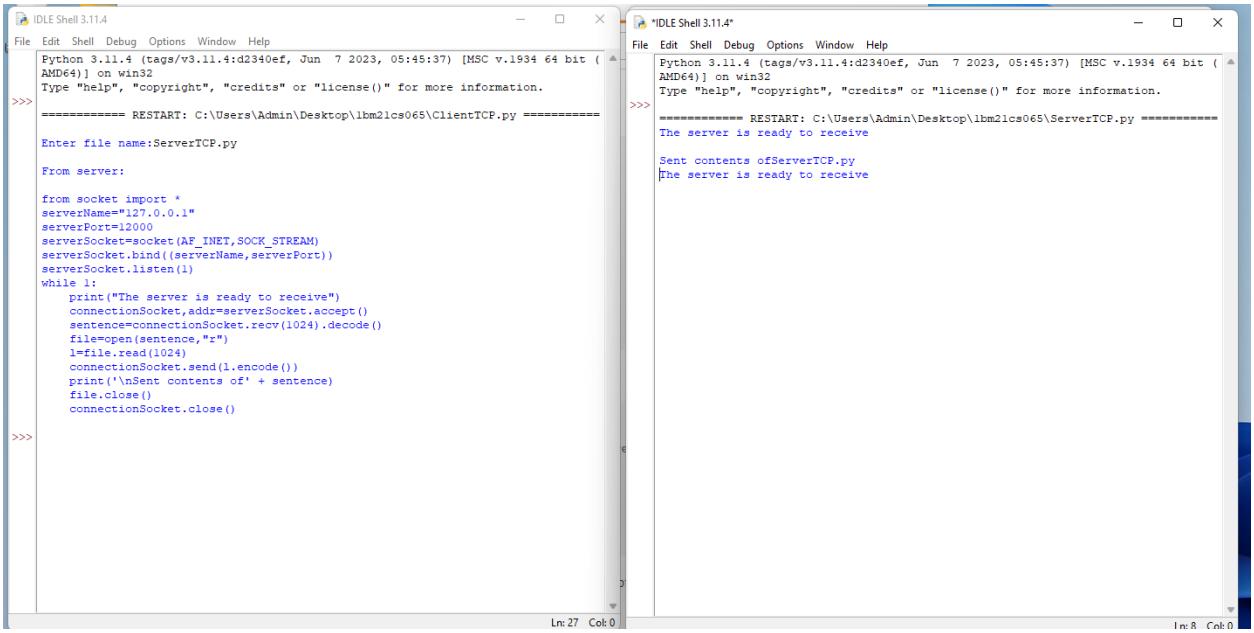
```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = input("\nEnter file name: ")
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print ("\nFrom Server:\n")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

ServerTCP.py

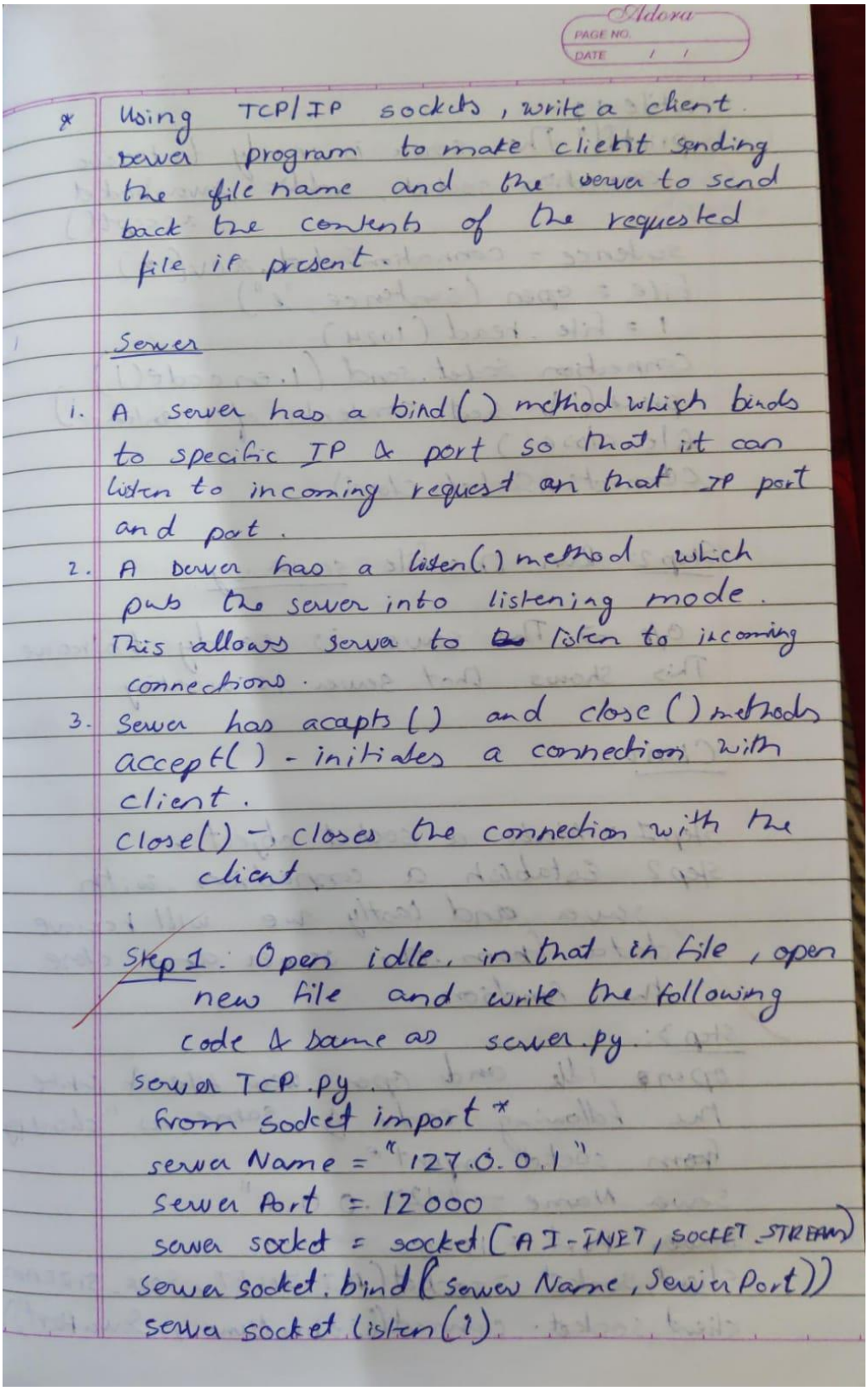
```
from socket import *
serverName="127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
print ("The server is ready to receive")
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024).decode()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

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

OUTPUT:



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 & same as server.py.

server TCP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AI-INET, SOCKET_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
```

```
while 1:
print(" The server is ready to recieve
  connection socket, addr = server Socket
                                .accept()
sentence = connectionSocket.recv(1024)
file = open (sentence, "r")
l = file.read (1024)
connectionSocket.send (l.encode(1))
print ("\n Sent contents of " + sentence)
file.close()
connectionSocket.close()
```

Step 2: Run the file server.py.

O/p => The server is ready to recieve
This shows that server is working.

Client

Step 1: Make a socket object.
Step 2: Establish a connection with server and lastly we will recieve data from the server and close the function

Step 3:
open idle and open new file & write the following code & same as "client.py"

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

Adora
PAGE NO.
DATE / /

sentence = input("\n Enter file name :")
clientSocket.send(sentence.encode())
file contents = clientSocket.recv(1024).decode()

print("\n From server :\n")
print(file contents)
clientSocket.close()

Run the file client.py

Client O/P

O/P ⇒ enter file name : server.py

Server O/P

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

The server is ready to recieve

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

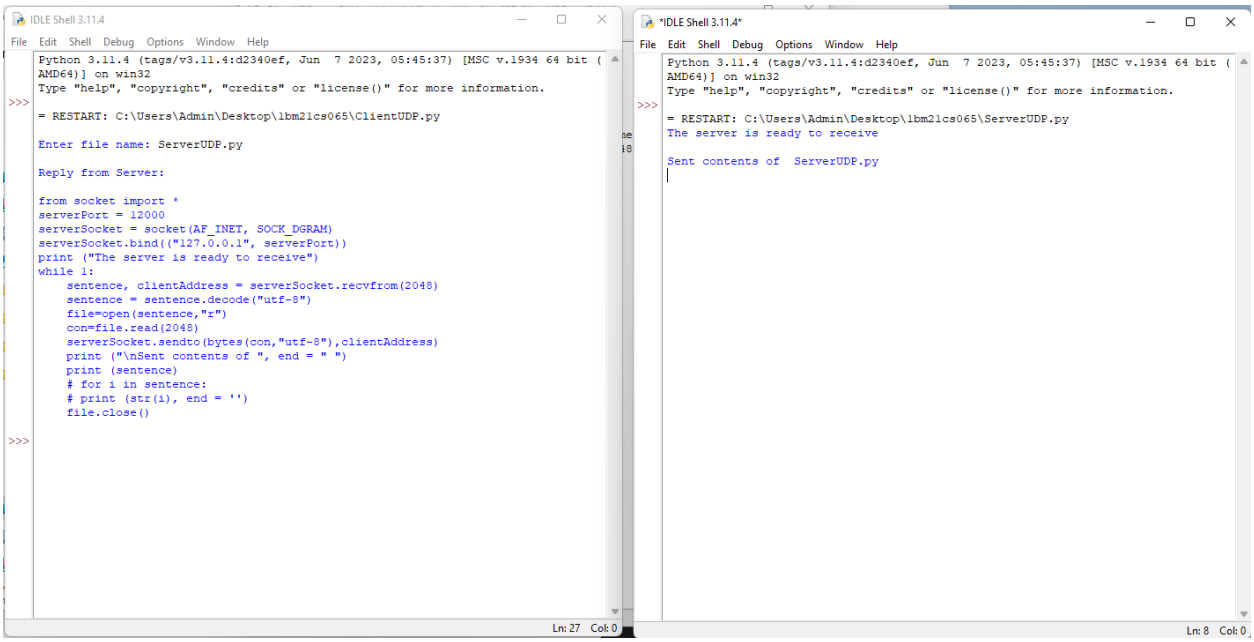
```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("\nEnter file name: ")
clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))
filecontents,serverAddress = clientSocket.recvfrom(2048)
print ("\nReply from Server:\n")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

ServerUDP.py

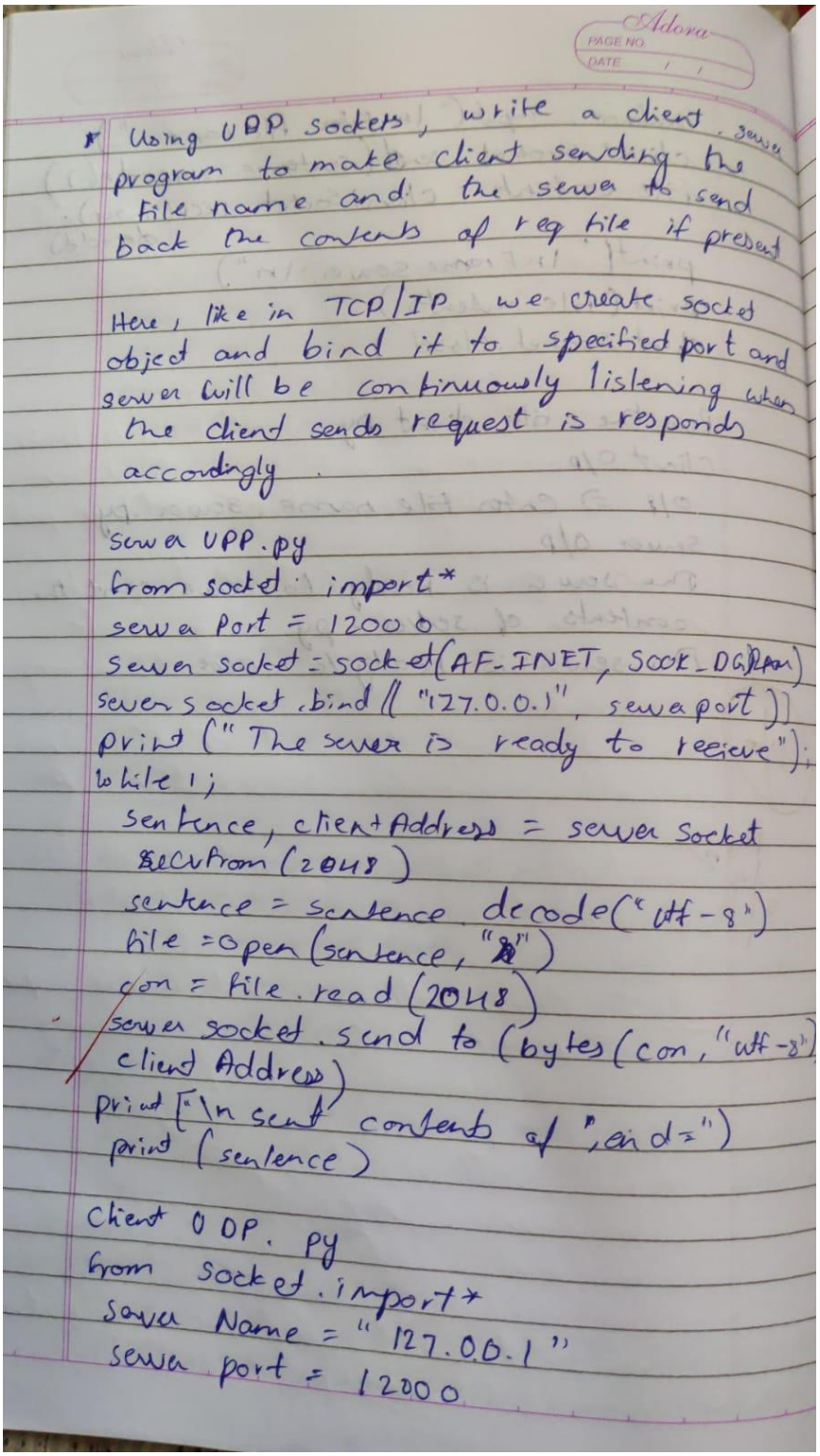
```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print ("The server is ready to receive")
while 1:
sentence, clientAddress = serverSocket.recvfrom(2048)
sentence = sentence.decode("utf-8")
file=open(sentence,"r")
```

```
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:



OBSERVATION:



Adora
PAGE NO.
DATE / /

* 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 UPP.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 ", end=")
    print(sentence)
```

Client UDP.py

```
from Socket.import*
serverName = "127.0.0.1"
serverPort = 12000
```

Adora
PAGE NO.
DATE / /

```
client socket = socket (AF_INET, SOCK_DGRAM)
sentence = input ["\n Enter file name: ");
client socket. send to (bytes [sentence,
    "utf-8" ), [server Name, server port))
file contents, server Address = client socket.
                recvfrom (2048)
print ("\n Reply from server: \n")
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

2/9/23