VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

COMPUTER NETWORKS

Submitted by

Hadia Fathima (1BM22CS106)

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

Sep 2024-Jan 2025

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 **Hadia Fathima (1BM22CS106),** who is bonafide student of **B.M.S. College of Engineering.** who is bonafide student of B. M. S. College of Engineering. It is in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belgaum during the year 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of Computer Networks Lab - (23CS5PCCON) work prescribed for the said degree.

Dr. Latha N.R.
Associate Professor,
Department of CSE
BMSCE, Bengaluru

Dr. Kavitha Sooda
Professor and Head,
Department of CSE
BMSCE, Bengaluru

Index
Cycle 1

Index

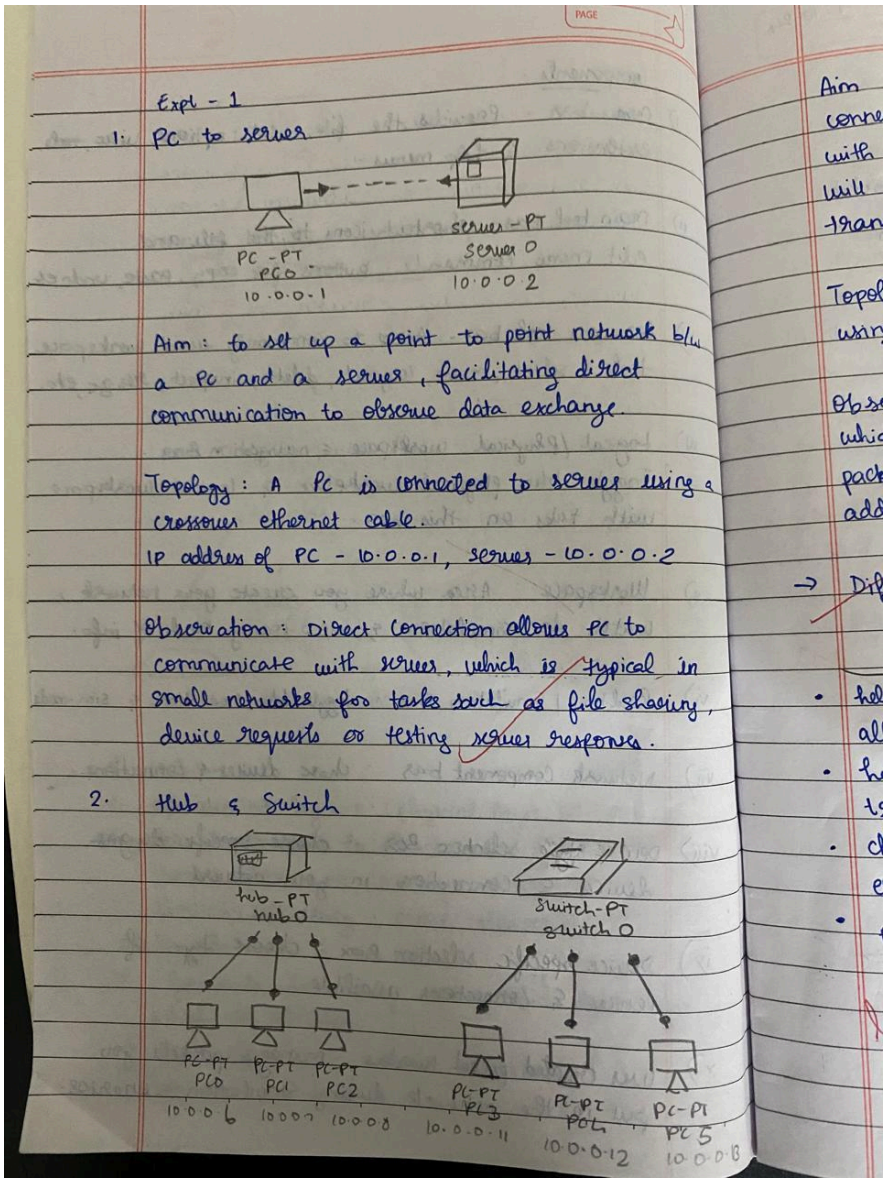
Cycle 2

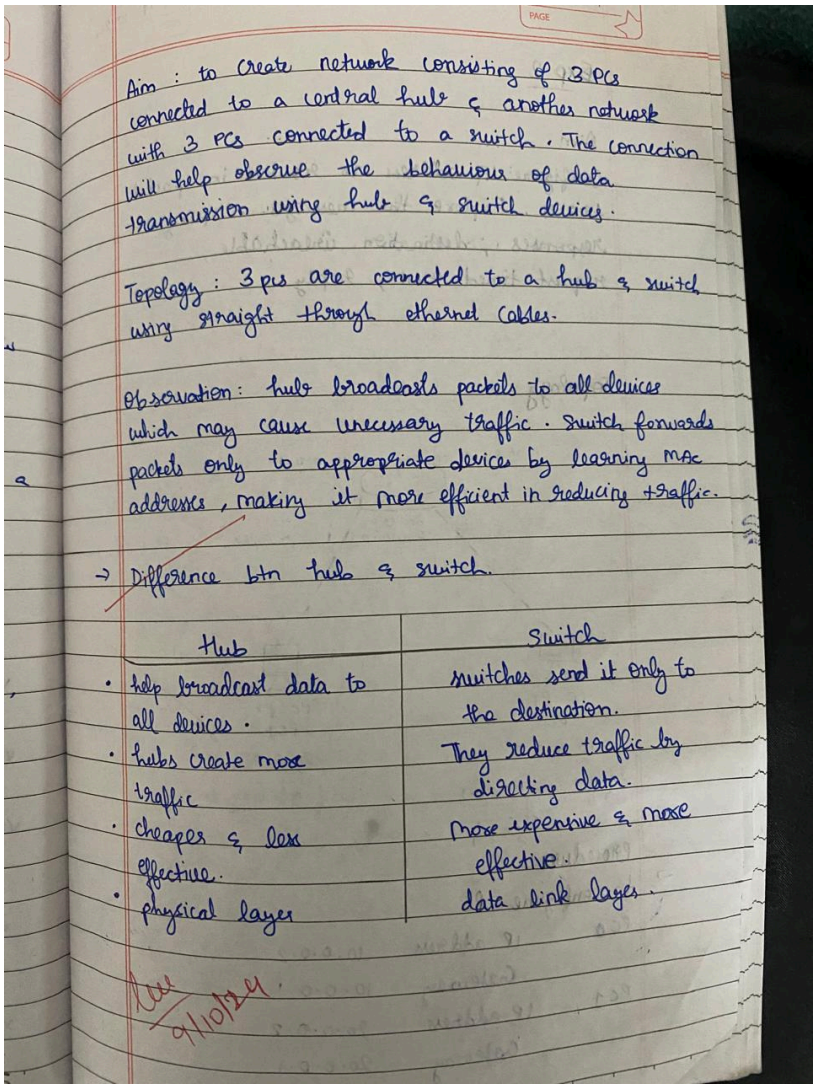
Cycle 1
Program 1

Aim of the program:

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

Procedure along with the topology:





Aim : to create network consisting of 3 PCs connected to a central hub & another network with 3 PCs connected to a switch. The connection will help observe the behaviour of data transmission using hub & switch devices.

Topology : 3 pcs are connected to a hub & switch using straight through ethernet cables.

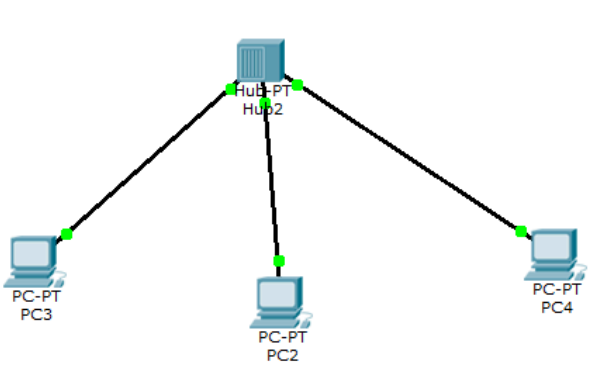
Observation : hub broadcasts packets to all devices which may cause unnecessary traffic. Switch forwards packets only to appropriate devices by learning MAC addresses, making it more efficient in reducing traffic.

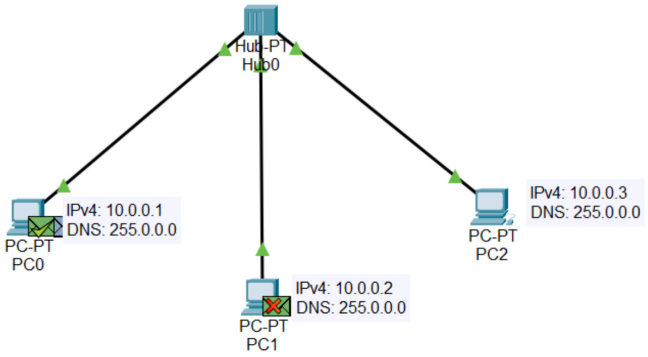
→ Difference btn hub & switch.

Hub	Switch
• help broadcast data to all devices.	switches send it only to the destination.
• hubs create more traffic	They reduce traffic by directing data.
• cheaper & less effective.	More expensive & more effective.
• physical layer	data link layer.

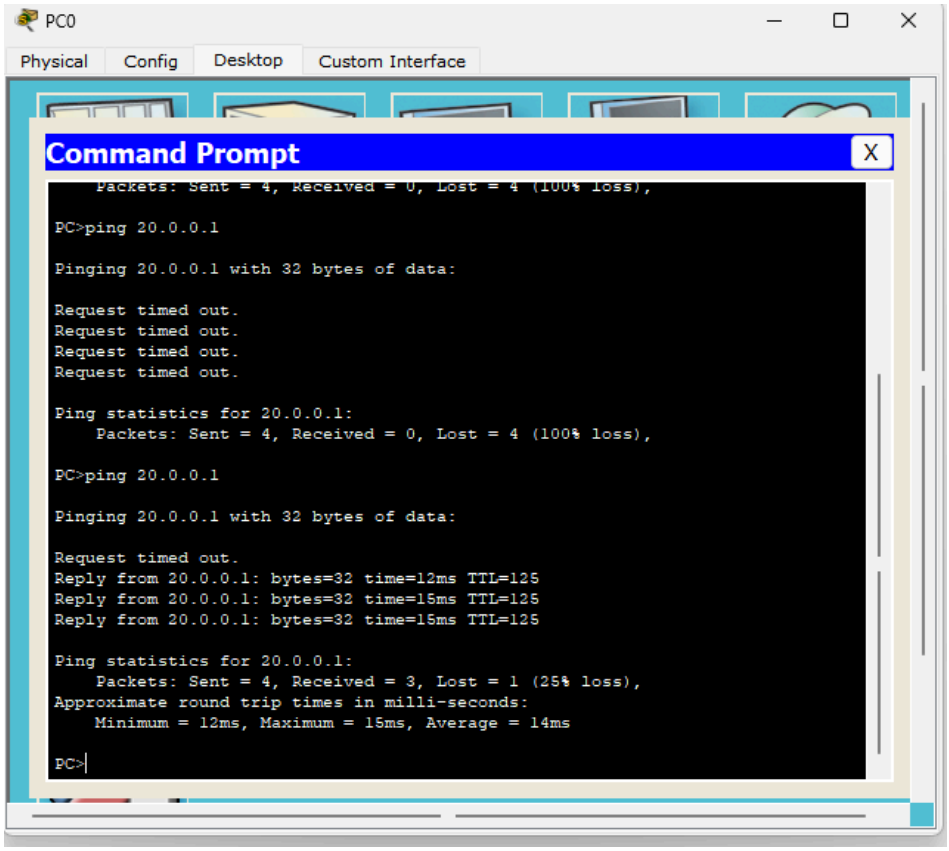
2/10/24

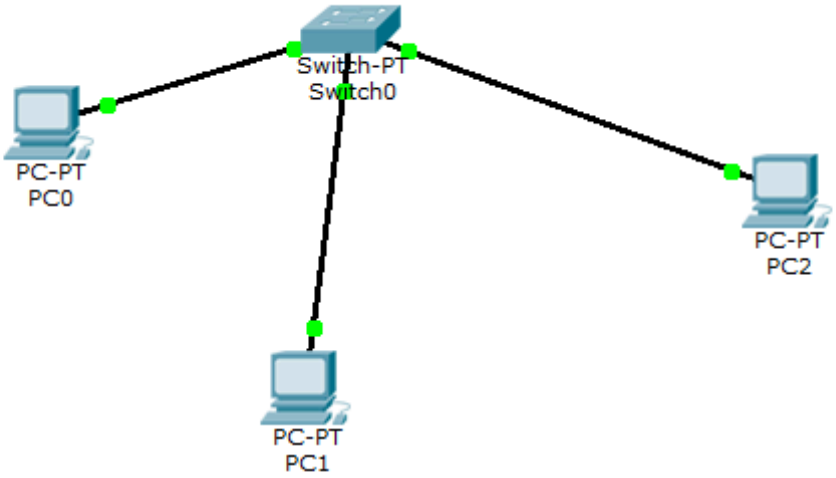
Screen shots/ output:





Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	PC2	ICMP		0.000	N	0	(edit)	





Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit	Delete
	Successful	PC0	PC1	ICMP		0.000	N	0	(edit)	(delete)

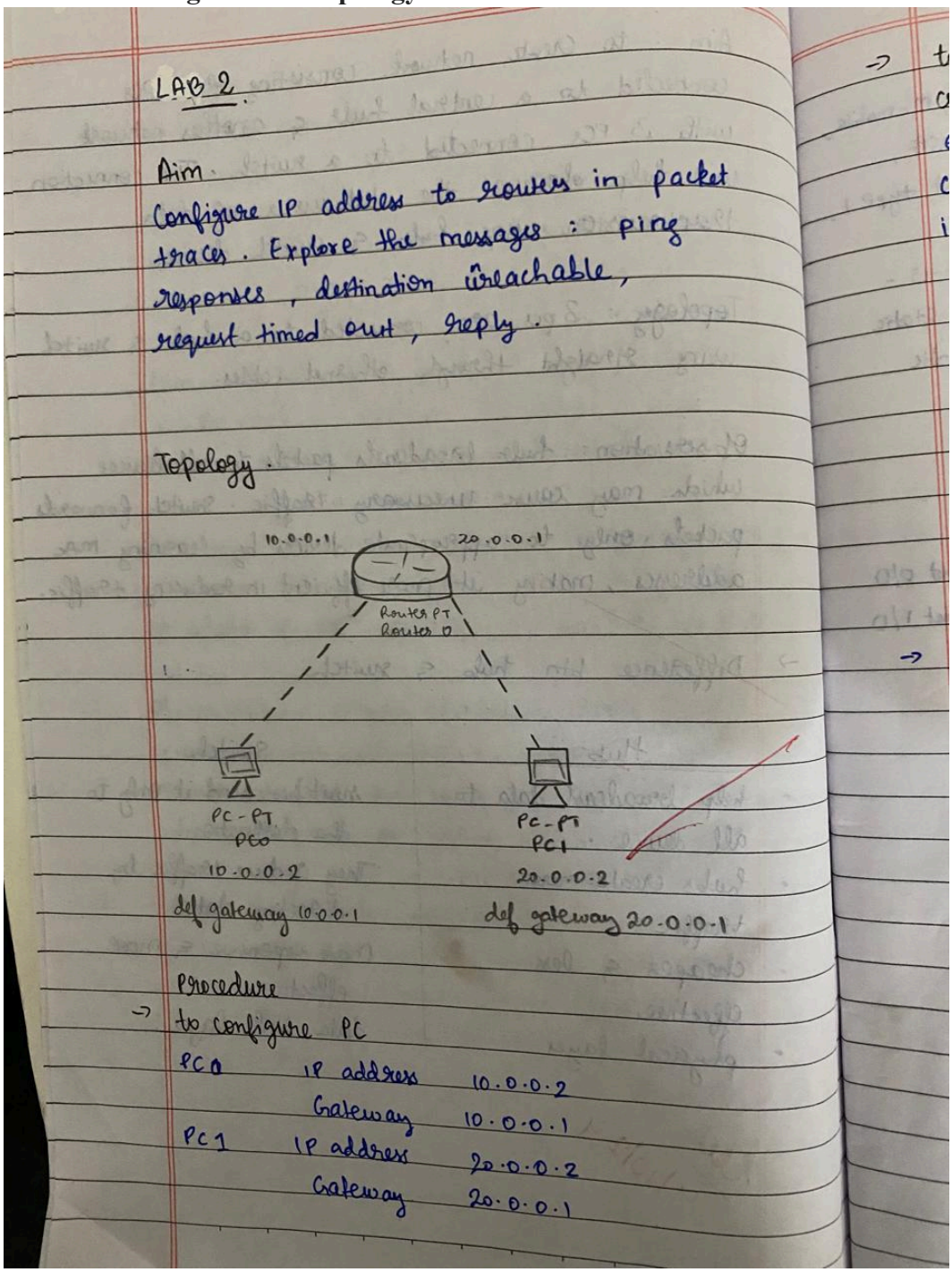
```
C:\>ping 10.0.0.3

Pinging 10.0.0.3 with 32 bytes of data:

Reply from 10.0.0.3: bytes=32 time=9ms TTL=128
Reply from 10.0.0.3: bytes=32 time<1ms TTL=128
Reply from 10.0.0.3: bytes=32 time=1ms TTL=128
Reply from 10.0.0.3: bytes=32 time<1ms TTL=128

Ping statistics for 10.0.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 9ms, Average = 2ms
```

Program 2

Aim of the program:

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

Procedure along with the topology:



→ to configure routers.
command line interface (CLI)
enable
config terminal
interface fastethernet 0/0
ip address 10.0.0.2 255.0.0.0
no shutdown
exit

interface fastethernet 0/1
ip address 20.0.0.2 255.0.0.0
no shutdown
exit

→ connect the router to each PC using copper cross over wire

Observation
PC> Ping 20.0.0.2
pinging 20.0.0.2 with 32 bytes of data:
Reply from 20.0.0.2 : bytes = 32 time = 3ms TTL = 128
Reply from 20.0.0.2 : bytes = 32 time = 2ms TTL = 128
Reply from 20.0.0.2 : bytes = 32 time = 0ms TTL = 128
Reply from 20.0.0.2 : bytes = 32 time = 6ms TTL = 128
Ping statistics for 20.0.0.2 :
Packets : Sent = 4, Received = 4, lost = 0 (0% loss),
Approximate round trip times in milli-seconds :
Minimum = 0ms, Maximum = 6 ms, Average = 2 ms
PC>

Router > 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 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, FastEthernet 0/0
C 20.0.0.0/8 is directly connected, FastEthernet 1/0
Router >

lll
9/10/24

LAB - 3

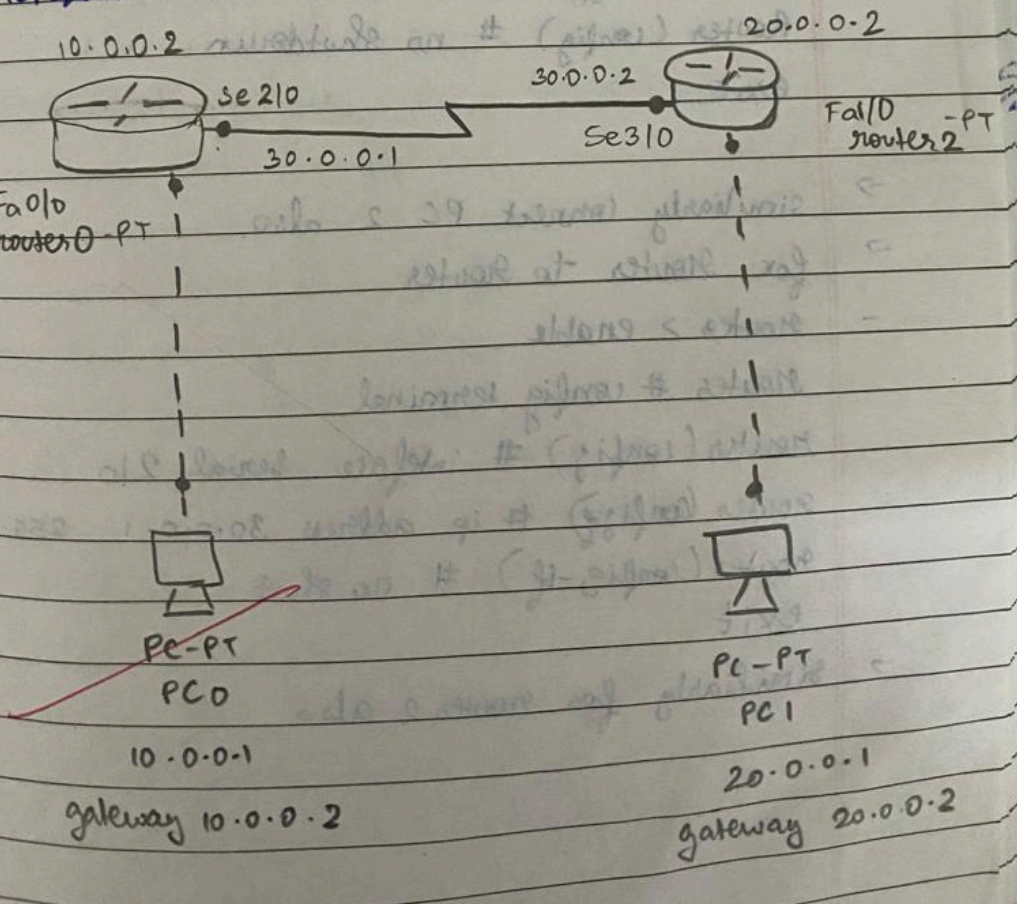
Experiment 2.

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

AIM :

To connect two routers serially & connect end devices to ~~rep~~ respective routers

TOPOLOGY :



PROCEDURE:

→ Add two generic PC's & two generic routers in the workspace
→ Give the pc's a default gateway & IP address.
→ each PC is connected with a copper wire to the respective router.
→ click on the router & go to CLI commands.

Router > enable
Router # config terminal
Router (config) # interface fastethernet 0/0
Router (config) # ip address 10.0.0.2 255.0.0.0
Router (config) # no shutdown
exit

→ Similiarly connect PC 2 also
→ for router to router
- router > enable
router # config terminal
router (config) # inteface serial 2/0
router (config-if) # ip address 30.0.0.1 255.0.0.0
router (config-if) # no shut
exit
→ similiarly for router 2 also

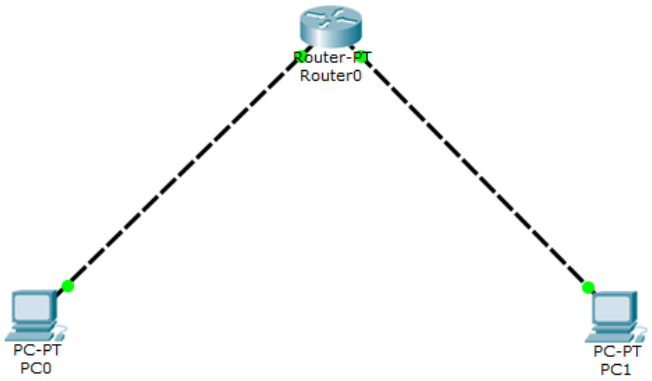
DATE / /
PAGE

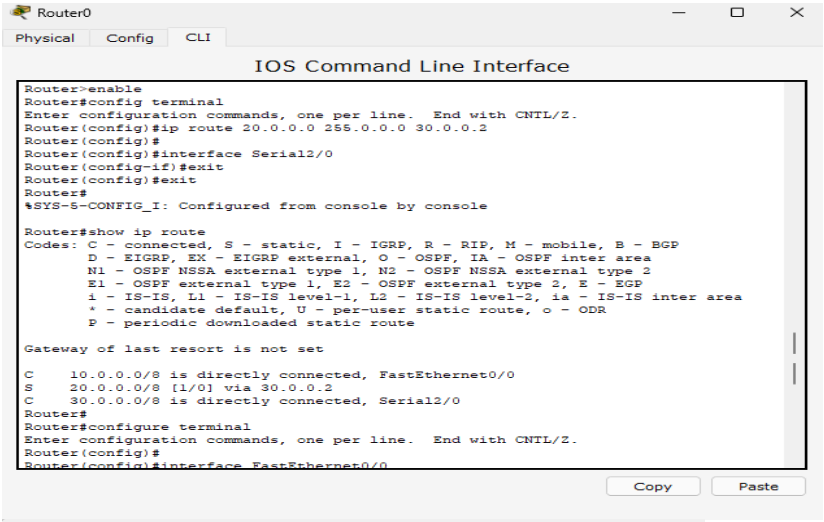
OBSERVATION :

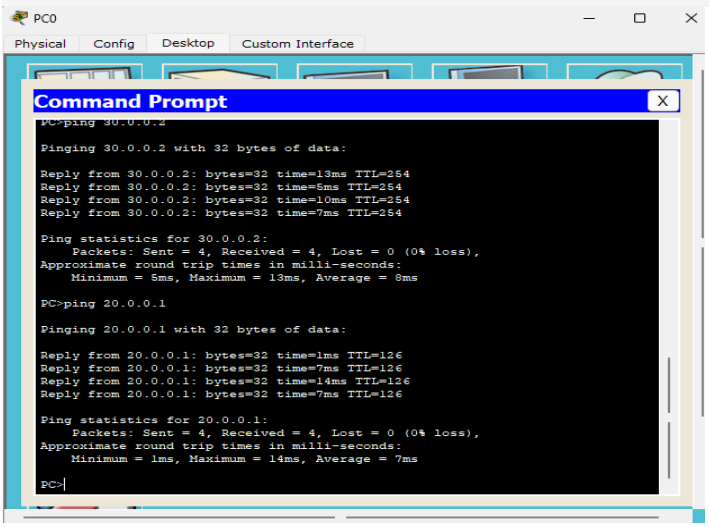
→ on pinging PC1 from PC0 we are unreachable.
→ on pinging router 0 from PC0 message packets are sent.
→ on pinging router 1 from PC0 it is unreachable.

∴ from the end devices we are not able to send or recieve messages packets through the routers.

Screen shots/ output:



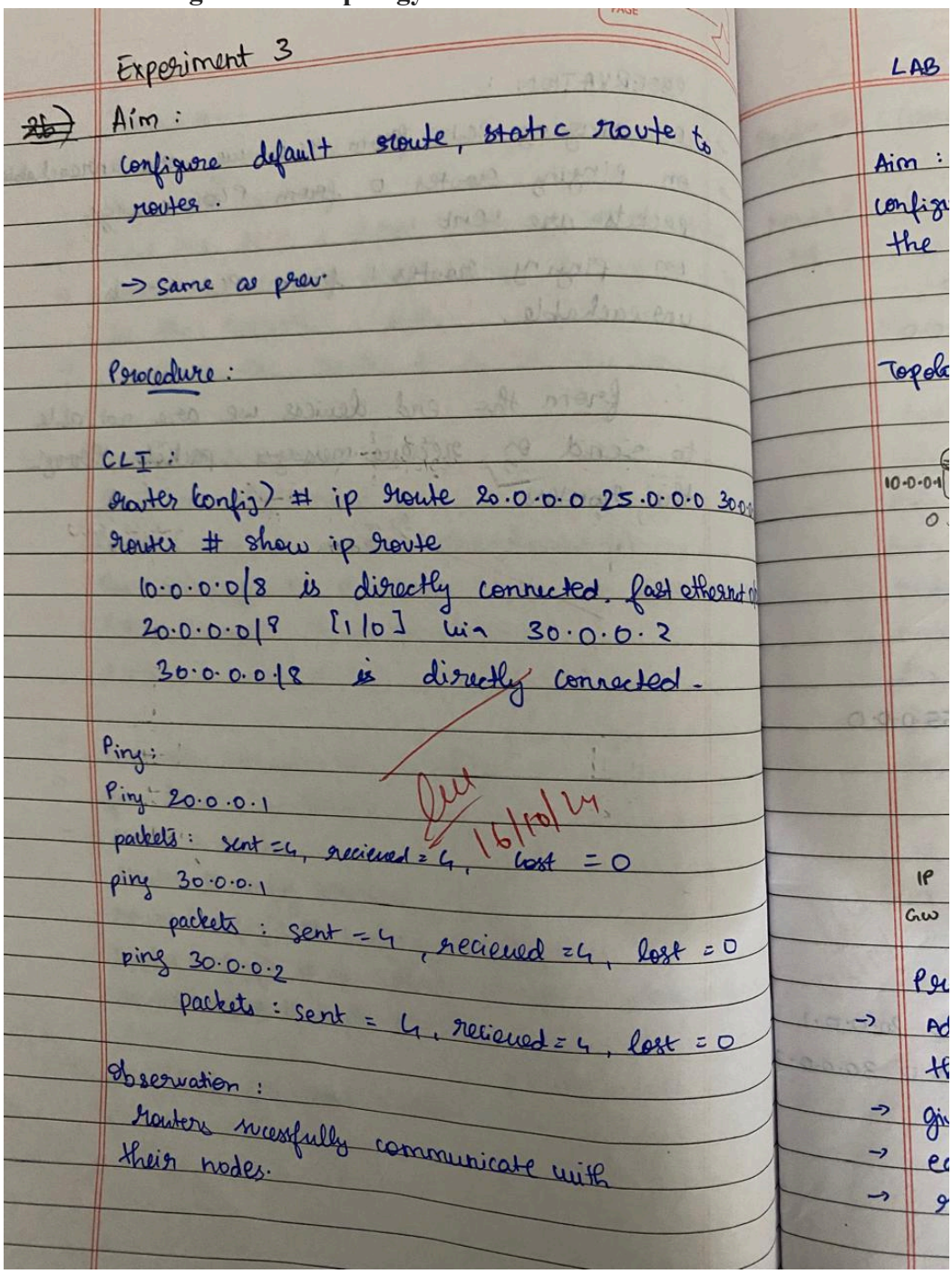




Program 3

Aim of the program:
Configure default route, static route to the Router

Procedure along with the topology:



Experiment 3

Aim :
Configure default route, static route to router.

→ same as prev.

Procedure :

CLI :
Router (config) # ip route 20.0.0.0 25.0.0.0 30.0.0.2
Router # show ip route
10.0.0.0/8 is directly connected, fast ethernet 0/0
20.0.0.0/8 [1/0] via 30.0.0.2
30.0.0.0/8 is directly connected.

Ping :
Ping 20.0.0.1
packets : sent = 4, recieved = 4, lost = 0
ping 30.0.0.1
packets : sent = 4, recieved = 4, lost = 0
ping 30.0.0.2
packets : sent = 4, recieved = 4, lost = 0

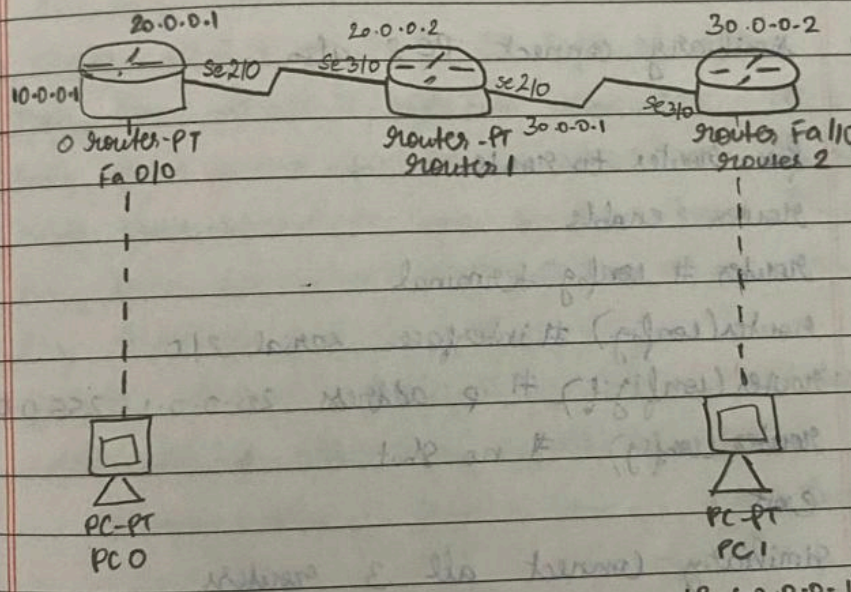
Observation :
Routers sucessfully communicate with their nodes.

DATE 23/10
PAGE

LAB – 4

Aim :
Configure default route, static route to the routers.

Topology :



Procedure.

→ Add two generic PC's and 3 generic routers in the workspace.
→ Give the PC's a default gateway & IP address.
→ Each PC is connected with copper cross-over to routers
→ routers are connected using serial DCE.

→ click on router and go to CLI command.

router > enable
router # config terminal
router (config) # interface fastethernet 0/0
router (config-if) # ip address 10.0.0.1 255.0.0.0
router (config) # no shut
exit

→ similiary connect PC 2 also

→ for routers to routers
router > enable
router # config terminal
router (config) # interface serial 2/0
router (config-if) # ip address 20.0.0.1 255.0.0.0
router (config) # no shut
exit

→ similiary connect all 3 routers

CLI :

router 1
SR

router # config terminal
router (config) # ip route 10.0.0.0 255.0.0.0 20.0.0.
ip route 40.0.0.0 255.0.0.0 30.0
exit

router 0
DR

router 2
DR

Router 0 R(config) # ip route 0.0.0.0 0.0.0.0 20.0.0.2
DR

Router 2 R(config) # ip route 0.0.0.0 0.0.0.0 30.0.0.1
DR

Observation:
In Command prompt of PC0

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=7ms TTL=125
reply from 40.0.0.10: bytes=32 time=6ms TTL=125
reply from 40.0.0.10: bytes=32 time=6ms TTL=125
reply from 40.0.0.10: bytes=32 time=8ms TTL=125

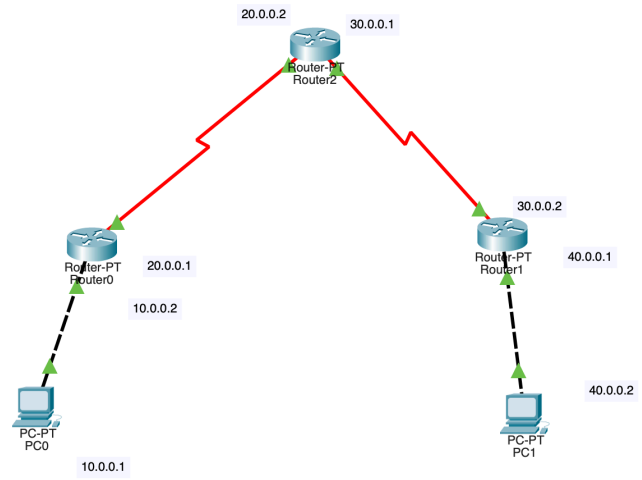
Ping statistics for 40.0.0.10:
packets: Sent=4, recieved=4, lost=0 (0% loss),
approx. round trip times in milli-seconds:
minimum = 6ms, max = 8ms, Avg = 6ms

In router 0 CLI
23/10/24
router> show ip route
C 10.0.0.0/8 is directly connected, fastethernet 0/0
C 20.0.0.0/8 is directly connected, serial 2/0
S* 0.0.0.0/0 [1/0] via 20.0.0.2

In router 2 CLI router> show IP route
C 30.0.0.0/8 is directly connected, serial 3/0
C 40.0.0.0/8 is directly connected, fastethrnt 1/0
S* 0.0.0.0/0 [1/0] via 30.0.0.1

Result: Each PC is now connected to all the 3 routers & other PC. So packet can be sent.

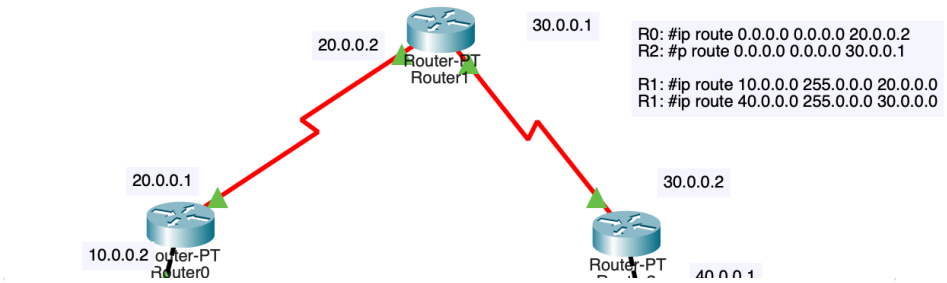
Screen shots/ output:

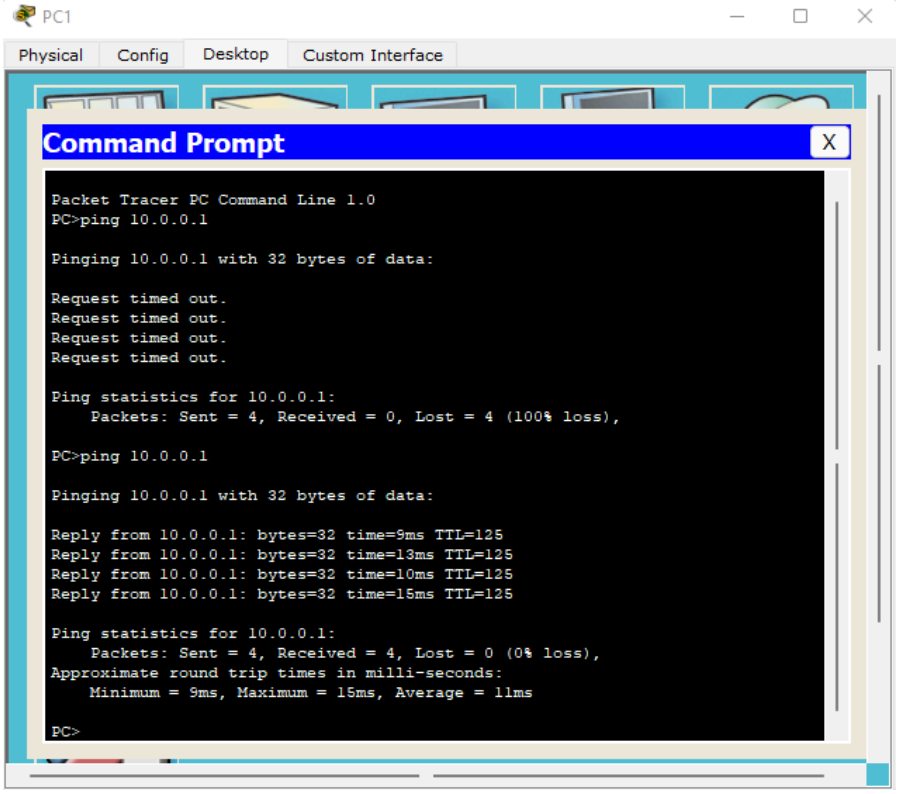


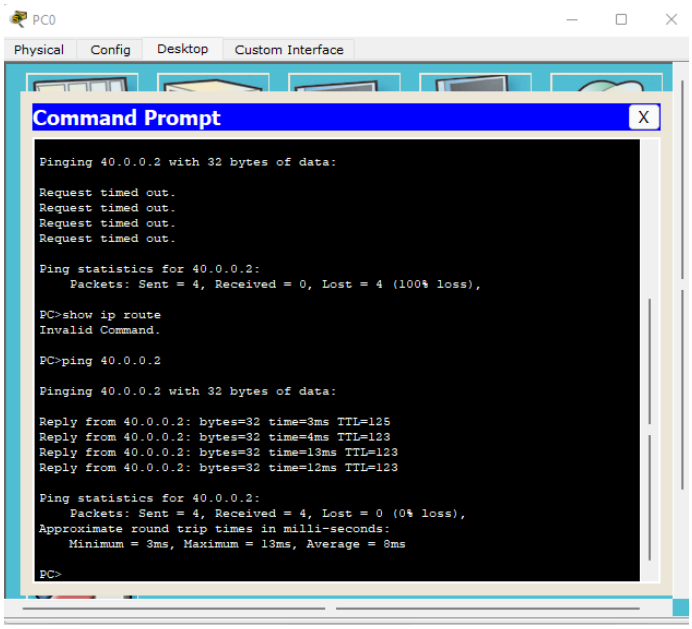
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Failed	PC0	PC1	IC...		0.000	N	0	(...	(delete)

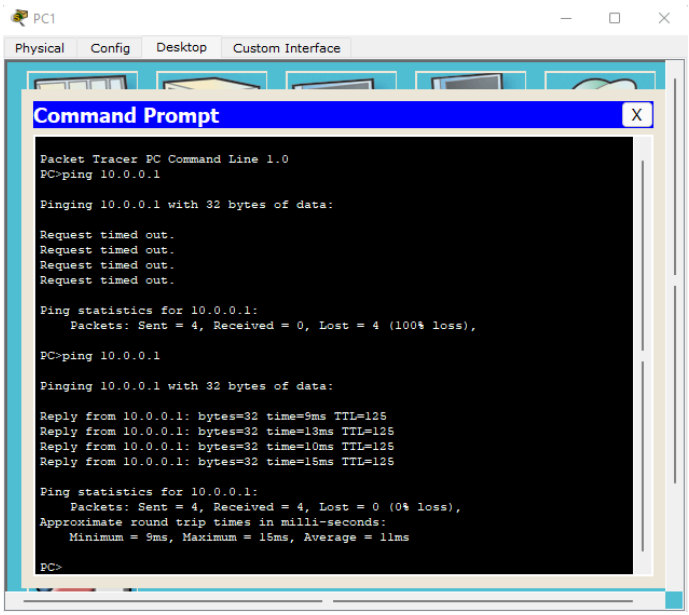
```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 40.0.0.2

Pinging 40.0.0.2 with 32 bytes of data:

Reply from 40.0.0.2: bytes=32 time=4ms TTL=125
Reply from 40.0.0.2: bytes=32 time=21ms TTL=125
Reply from 40.0.0.2: bytes=32 time=55ms TTL=125
Reply from 40.0.0.2: bytes=32 time=39ms TTL=125

Ping statistics for 40.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 55ms, Average = 29ms

C:\>
```

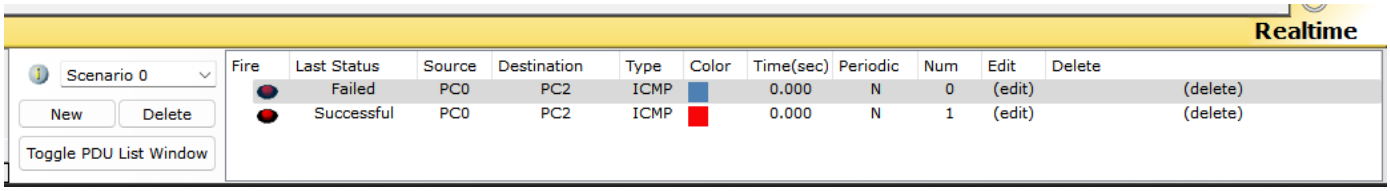
```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 10.0.0.1

Pinging 10.0.0.1 with 32 bytes of data:

Reply from 10.0.0.1: bytes=32 time=4ms TTL=125
Reply from 10.0.0.1: bytes=32 time=2ms TTL=125
Reply from 10.0.0.1: bytes=32 time=47ms TTL=125
Reply from 10.0.0.1: bytes=32 time=54ms 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 = 54ms, Average = 26ms

C:\>
```

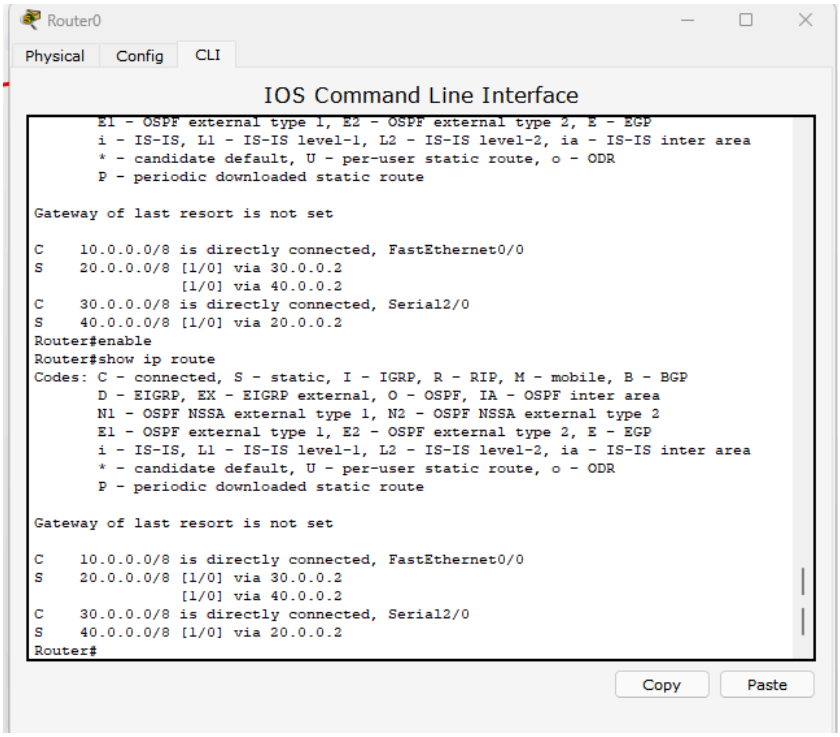





PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Pinging 40.0.0.2 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 40.0.0.2:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>show ip route
Invalid Command.
PC>ping 40.0.0.2
Pinging 40.0.0.2 with 32 bytes of data:
Reply from 40.0.0.2: bytes=32 time=3ms TTL=125
Reply from 40.0.0.2: bytes=32 time=4ms TTL=123
Reply from 40.0.0.2: bytes=32 time=13ms TTL=123
Reply from 40.0.0.2: bytes=32 time=12ms TTL=123
Ping statistics for 40.0.0.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 3ms, Maximum = 13ms, Average = 8ms
PC>


PC1
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.1
Pinging 10.0.0.1 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 10.0.0.1:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>ping 10.0.0.1
Pinging 10.0.0.1 with 32 bytes of data:
Reply from 10.0.0.1: bytes=32 time=9ms TTL=125
Reply from 10.0.0.1: bytes=32 time=13ms TTL=125
Reply from 10.0.0.1: bytes=32 time=10ms TTL=125
Reply from 10.0.0.1: bytes=32 time=15ms 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 = 9ms, Maximum = 15ms, Average = 11ms
PC>

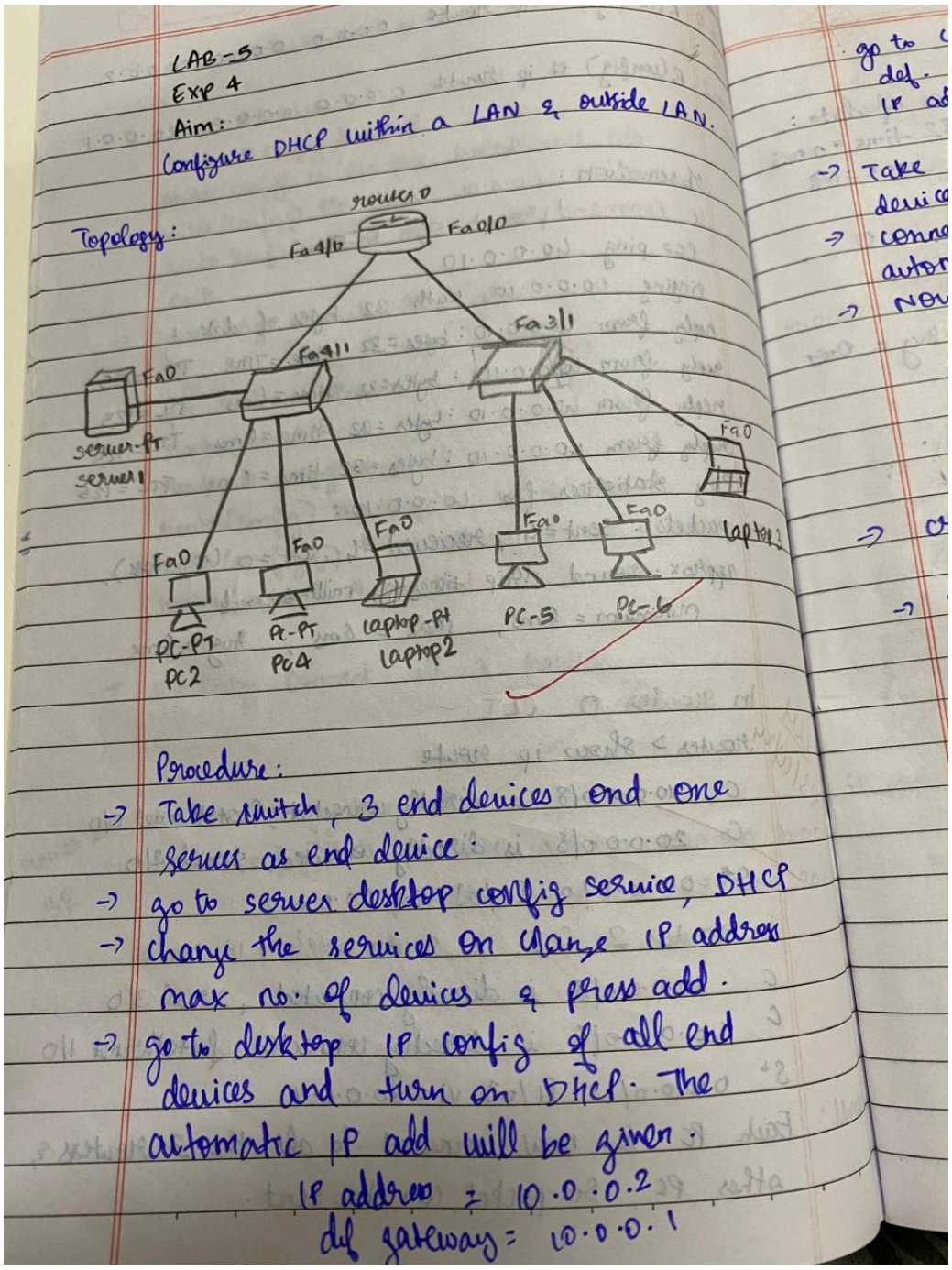

Realtime
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time(sec)
Periodic
Num
Edit
Delete
Failed PC0 PC2 ICMP 0.000 N 0 (edit) (delete)
Successful PC0 PC2 ICMP 0.000 N 1 (edit) (delete)


Router0
Physical
Config
CLI
IOS Command Line Interface
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
S 20.0.0.0/8 [1/0] via 30.0.0.2
[1/0] via 40.0.0.2
C 30.0.0.0/8 is directly connected, Serial2/0
S 40.0.0.0/8 [1/0] via 20.0.0.2
Router#enable
Router#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
S 20.0.0.0/8 [1/0] via 30.0.0.2
[1/0] via 40.0.0.2
C 30.0.0.0/8 is directly connected, Serial2/0
S 40.0.0.0/8 [1/0] via 20.0.0.2
Router#
Copy
Paste

Program 4

Aim of the program:
Configure DHCP within the same LAN and outside LAN.

Procedure along with the topology:



LAN.

· go to config → services → DHCP set poolname.
def. gateway 10.0.0.0 show
IP address 10.0.0.3 & mak number 100

→ Take another switch & connect 3 end devices.
→ connect 2 switch to router through automatic connection (fast ethernet).
→ Now goto server IP config.
IP 10.0.0.2
def gateway 10.0.0.1
change pool name switch
def. gateway 10.0.0.1
save

→ create another pool with name switch 2.
→ set default gateway 20.0.0.1
IP → 20.0.0.3
max → 100 Add.

Route terminal
CLI commands.

interface fastethernet 4/0
ip address 10.0.0.1 255.0.0.0
ip helper-address 10.0.0.2
no shut
exit
interface fastethernet 0/0
ip address 20.0.0.1 25.0.0.0
ip helper-addres 10.0.0.2
no shut
exit

observation.

PC> ping 10.0.0.4

pinging 10.0.0.4 with 32 bytes of data:

reply from 10.0.0.4: bytes = 32 time = 0ms TTL = 128

ping statistics for 10.0.0.4:

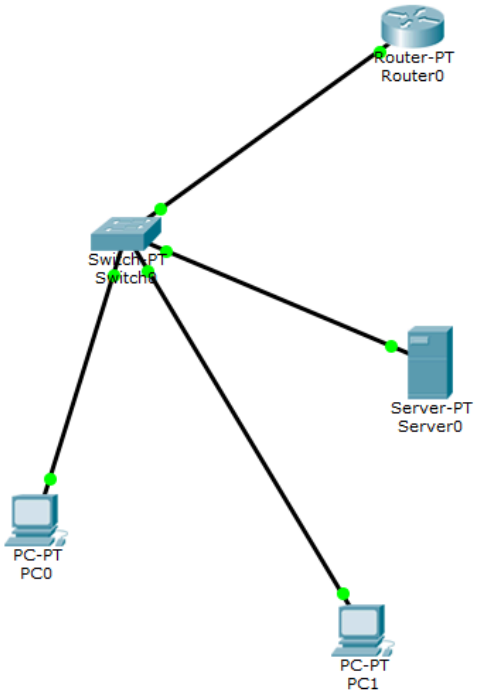
packets sent = 4, recieved = 4, lost = 0

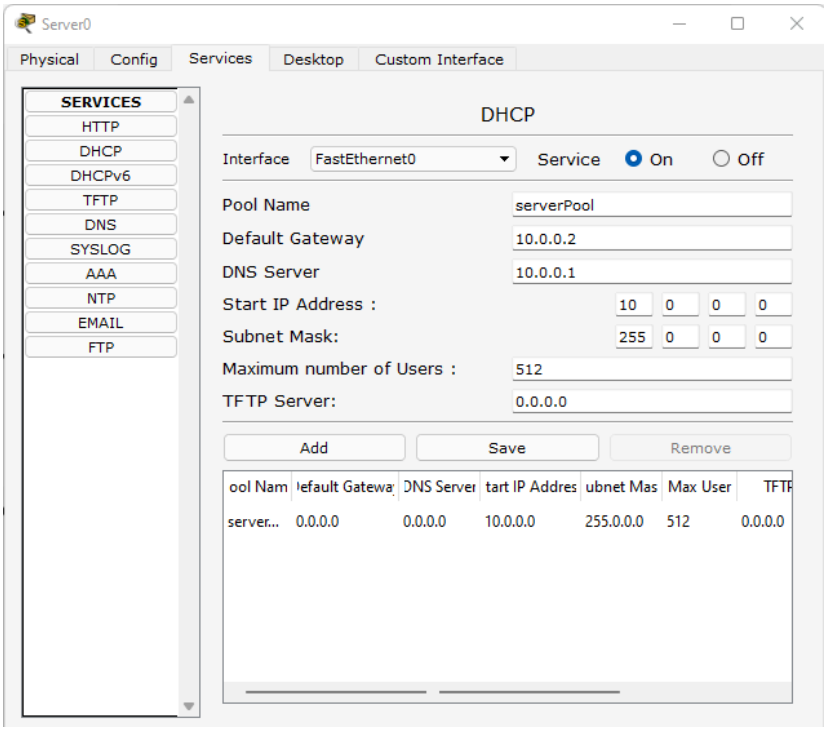
0% loss.

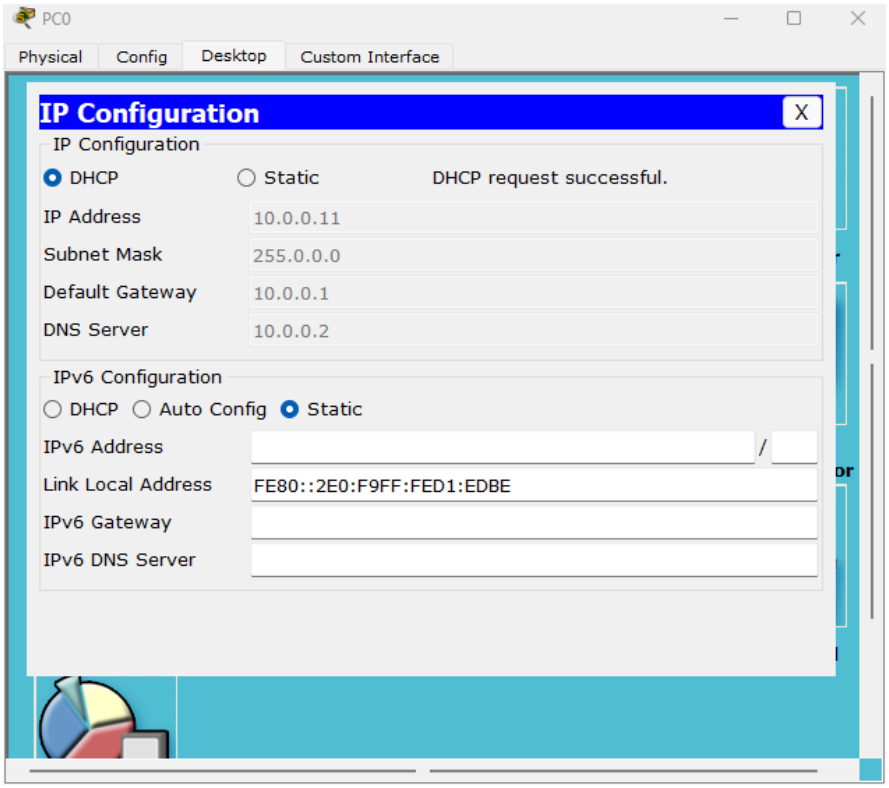
Approx round trip time in ms

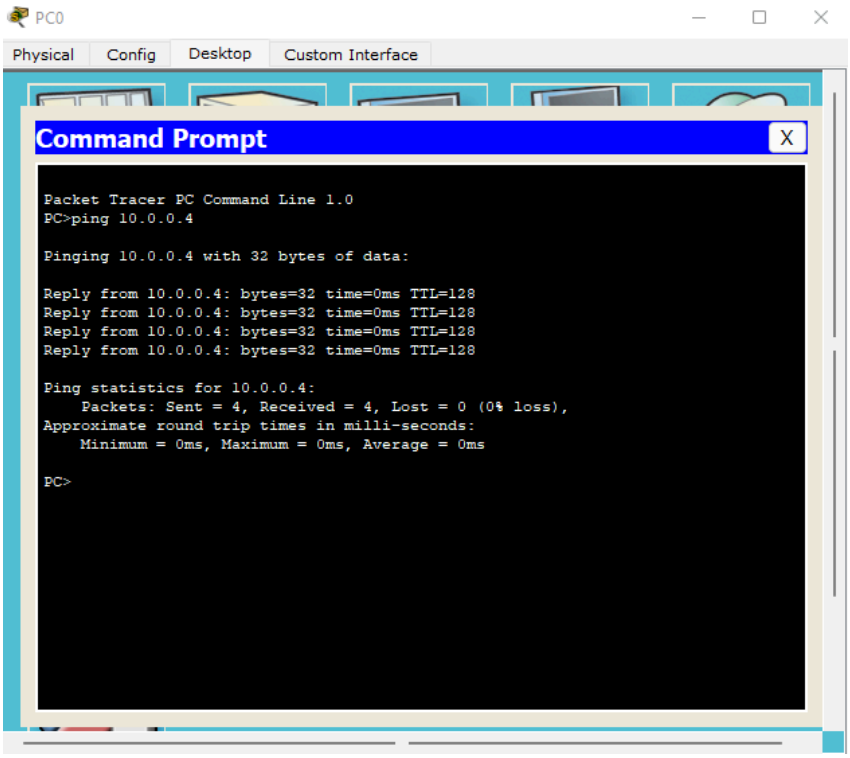
min = 0 ms, max = 0ms, Avg = 0ms

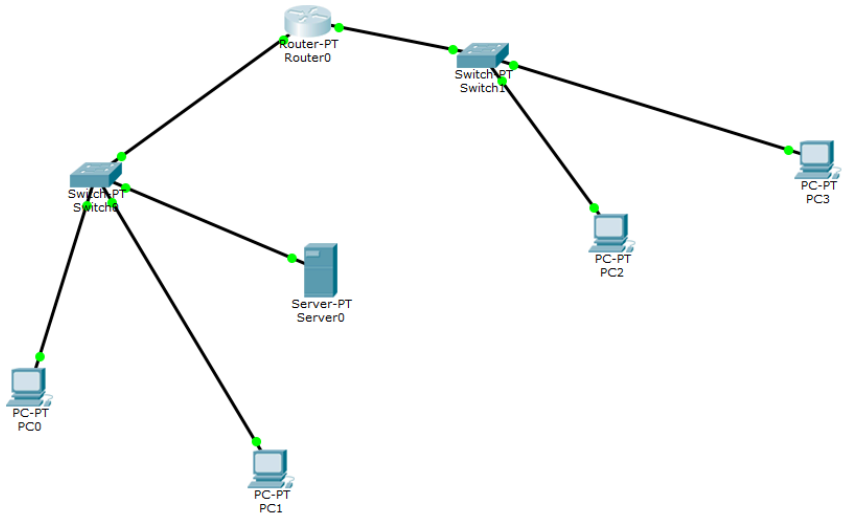
Screen shots/ output :

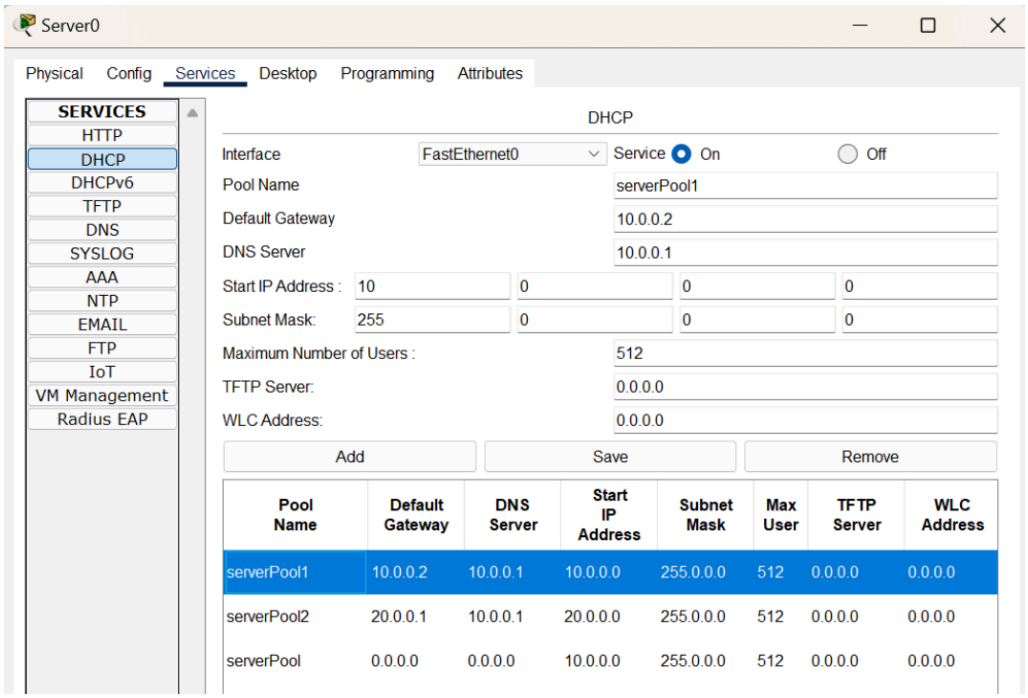




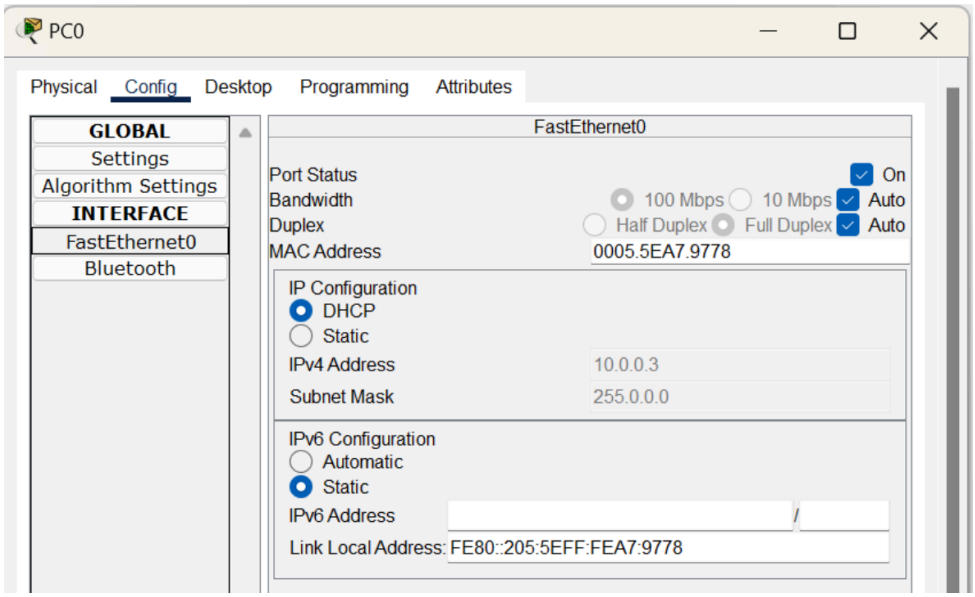



PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.4
Pinging 10.0.0.4 with 32 bytes of data:
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Reply from 10.0.0.4: bytes=32 time=0ms TTL=128
Ping statistics for 10.0.0.4:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
PC>


Router-PT
Router0
Switch-PT
Switch1
Switch-PT
Switch0
PC-PT
PC3
PC-PT
PC2
Server-PT
Server0
PC-PT
PC0
PC-PT
PC1



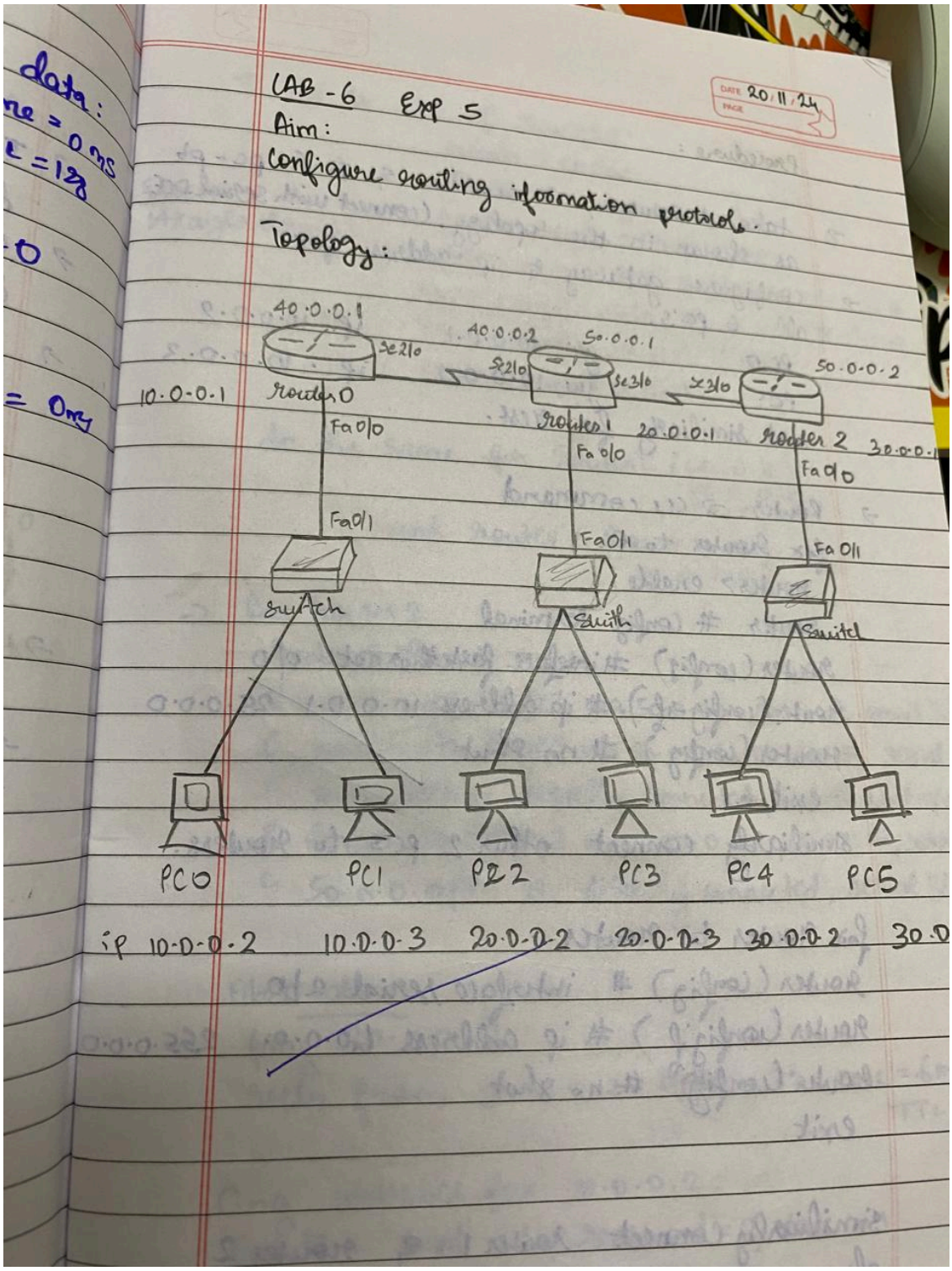
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop0	ICMP		0.000	N	0	(edit)	
	Successful	PC1	Laptop1	ICMP		0.004	N	1	(edit)	



Program 5

Aim of the program:
Configure RIP routing Protocol in Routers

Procedure along with the topology:



Procedure :

→ take 3 routers, 3 switches & 6 pc-pt as shown in the topology. (connect with serial DCE & copper straight)

→ configure gateway & ip address for all 6 pc's.

PC 0 gw 10.0.0.1 ip. 10.0.0.2

PC 1 gw 10.0.0.1 ip . 10.0.0.3

and similiary for rest.

→ Router → cli command

for router to PC.

router> enable

router # config terminal

router(config) #inteface fastethernet 0/0

router(config-if) # ip address 10.0.0.1 255.0.0.0

router(config) # no shut

exit

similiarly connect other 2 pc's to routers.

for router to router.

router(config) # interface serial 2/0

router(config-if) # ip address 40.0.0.1 255.0.0.0

router(config) # no shut

exit

similiarly connect router 1 & router 2 also.

6 PC - PC
with serial DCE

& copper straight through)

:0.0.2
.0.0.3

:0.0

:0.0

→ Go to CLI of router0 -
router > enable
router # config terminal
router(config) # router rip
router(config-router) # network 40.0.0.0
router(config-router) # network 10.0.0.0
router(config-router) # exit
router(config) # exit

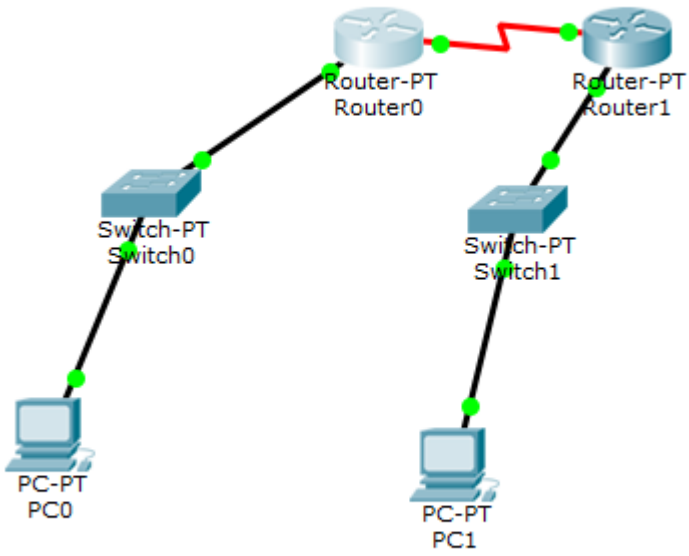
do the same for router1 (40.0.0.0
50.0.0.0
20.0.0.0)
and router 2 (50.0.0.0
30.0.0.0)

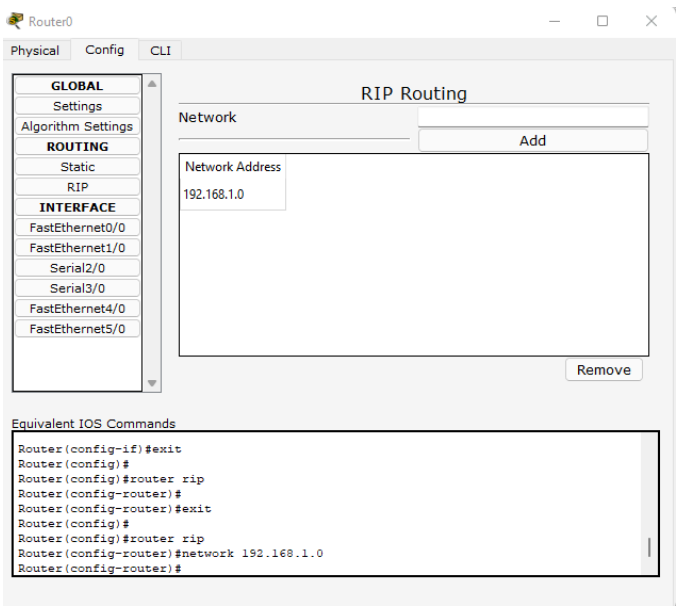
→ for router 2
router # show ip route
R 10.0.0.0/8 [120/2] via 50.0.0.1, 00:00:05, serial 3/0
R 20.0.0.0/8 [120/1] via 50.0.0.1, 00:00:05, serial 3/0
C 30.0.0.0/8 is directly connected, Fast ethernet 0/0
R 40.0.0.0/8 [120/1] via 50.0.0.1, 00:00:05, serial 3/0
C 50.0.0.0/8 is directly connected, serial 3/0

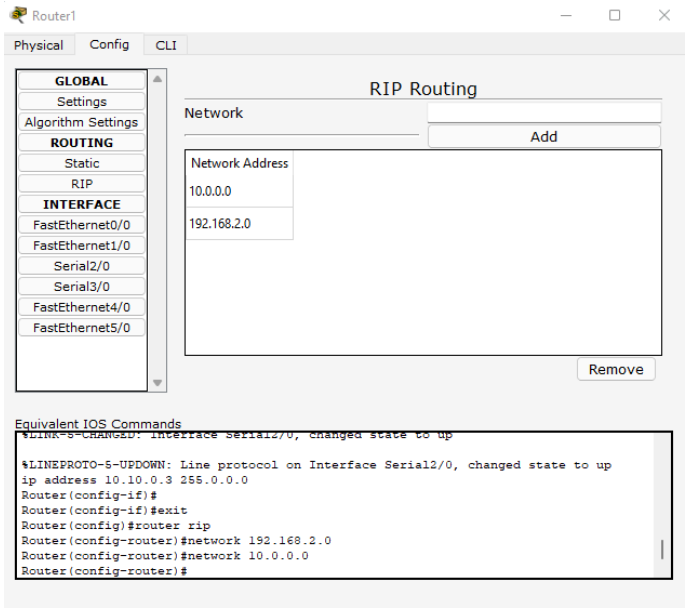
Observation:
ping 30.0.0.2
reply from 30.0.0.2 bytes=32 time=6ms
TTL=125

Ping statistics for 30.0.0.2:
Packets: sent = 4, received = 3, loss = 1
Approx. round trip time in ms:
Min = 6ms, max = 7ms, Avg = 6 ms.

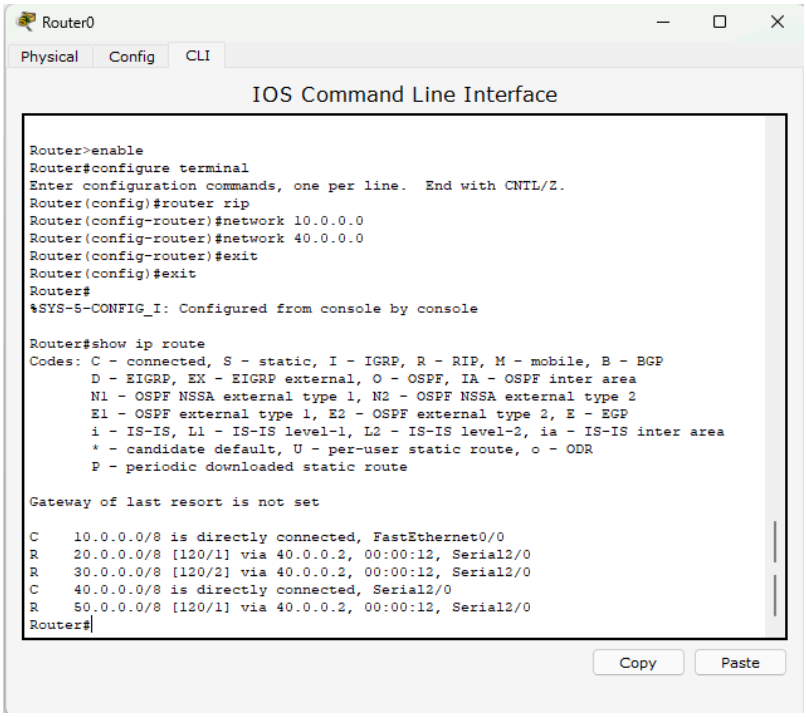
Screen shots/ output:

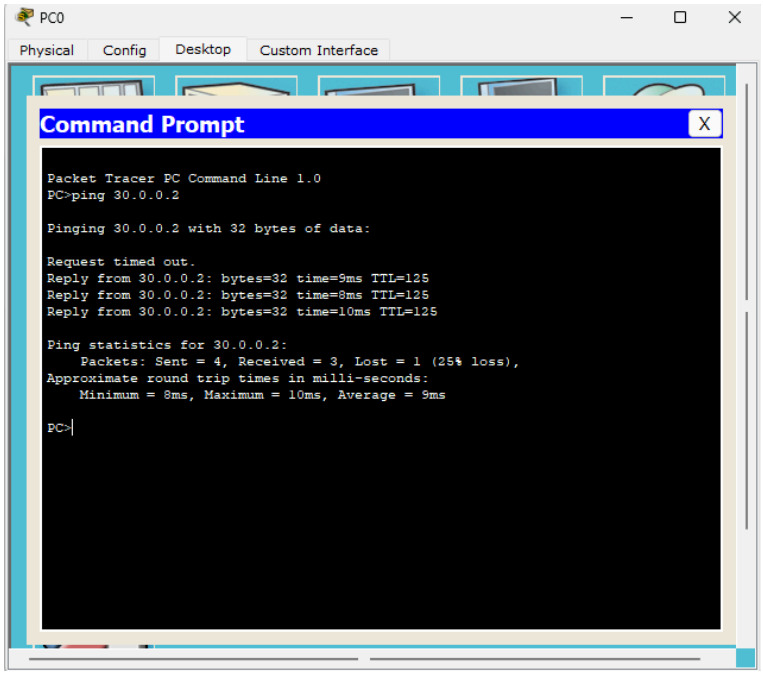






Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Failed	PC0	PC1	ICMP		0.000	N	0	(edit)	(delete)
	Successful	Router0	Router1	ICMP		0.000	N	1	(edit)	(delete)
	Successful	PC0	PC1	ICMP		0.000	N	2	(edit)	(delete)

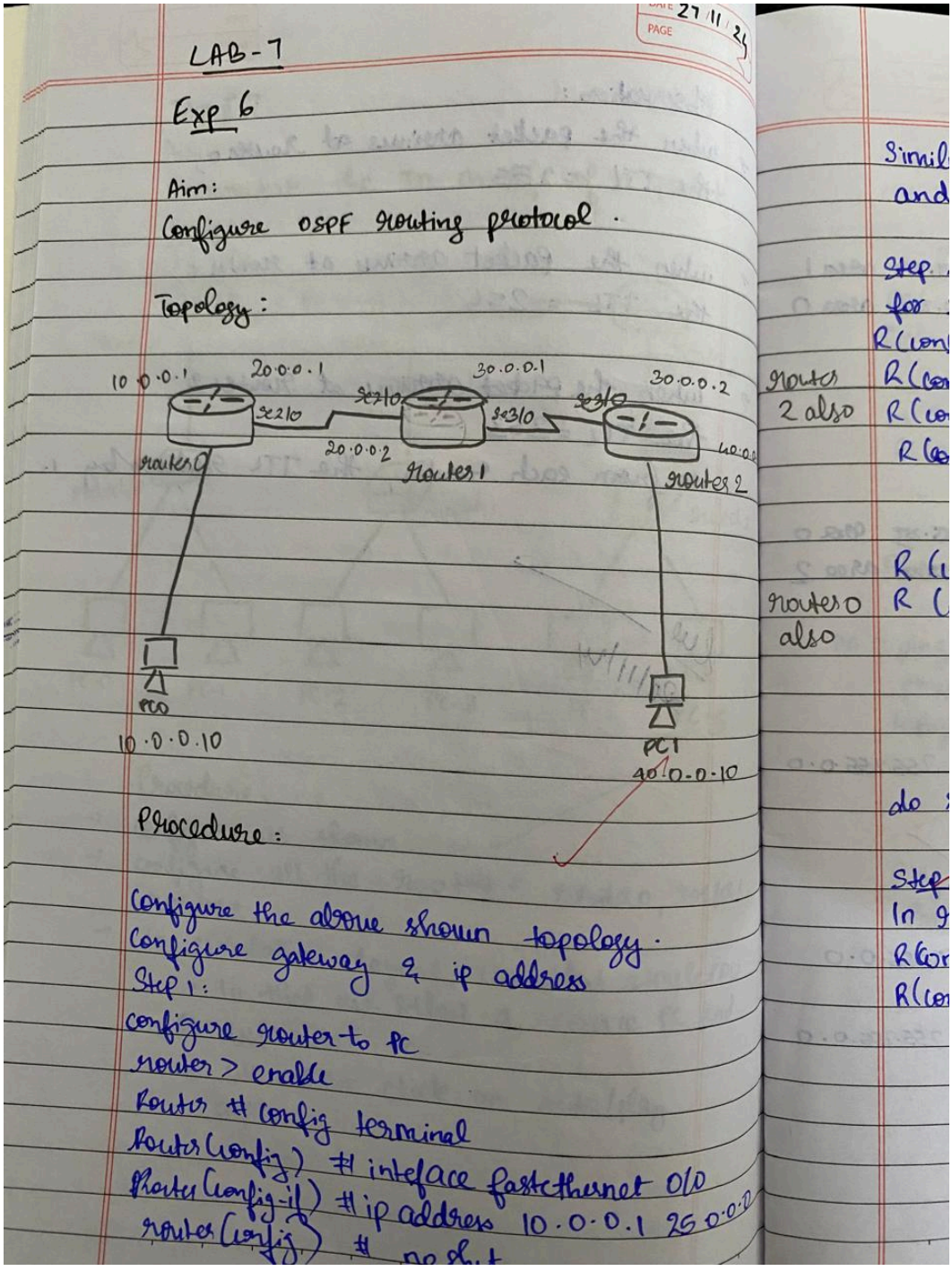

Router0
Physical
Config
CLI
IOS Command Line Interface
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#router rip
Router(config-router)#network 10.0.0.0
Router(config-router)#network 40.0.0.0
Router(config-router)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
Router#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
R 20.0.0.0/8 [120/1] via 40.0.0.2, 00:00:12, Serial2/0
R 30.0.0.0/8 [120/2] via 40.0.0.2, 00:00:12, Serial2/0
C 40.0.0.0/8 is directly connected, Serial2/0
R 50.0.0.0/8 [120/1] via 40.0.0.2, 00:00:12, Serial2/0
Router#
Copy
Paste


PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 30.0.0.2
Pinging 30.0.0.2 with 32 bytes of data:
Request timed out.
Reply from 30.0.0.2: bytes=32 time=9ms TTL=125
Reply from 30.0.0.2: bytes=32 time=8ms TTL=125
Reply from 30.0.0.2: bytes=32 time=10ms TTL=125
Ping statistics for 30.0.0.2:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 8ms, Maximum = 10ms, Average = 9ms
PC>

Program 6

Aim of the program:
Configure OSPF routing protocol

Procedure along with the topology:



Similiarly configure Router 2 to PC1 and configure router to routes connections

step 2:

for router 1

R(config) # interface serial 2/0
R(config-if) # ip address 20.0.0.2 25.0.0.0
R(config-if) # encapsulation ppp
R(config-if) # no shut
exit

router 2 also

R(config) # interface serial 3/0
R(config-if) # ip address 30.0.0.1 255.0.0.0
encapsulation ppp
clock rate 64000
no shut
exit

router 0 also

do similiary for routes 0 & router 2

Step 3:

In router 0
R(config) # routes ospf 1
R(config-router) # router-id 1.1.1.1
network 10.0.0.0 0.255.255.255 area 3
network 20.0.0.0 0.255.255.255 area 1
exit

In router 1

R(config)# router ospf 1
router-id 2.2.2.2
network 20.0.0.0 0.255.255.255 area 1
network 30.0.0.0 0.255.255.255 area 0
exit

In router 2

R(config)# router ospf 1
router-id 3.3.3.3
network 30.0.0.0 0.255.255.255 area 0
network 40.0.0.0 0.255.255.255 area 2
exit

Step 4:

R0(config-if)# interface loopback 0
R0(config-if)# ip add 172.16.1.252 255.255.0.0
no shut

Similarly:

R1(config-if)# ip add 172.16.1.253 255.255.0.0

R2(config-if)# ip add 172.16.1.254 255.255.0.0

Step 5:
Create virtual link between R0 & R2

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

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

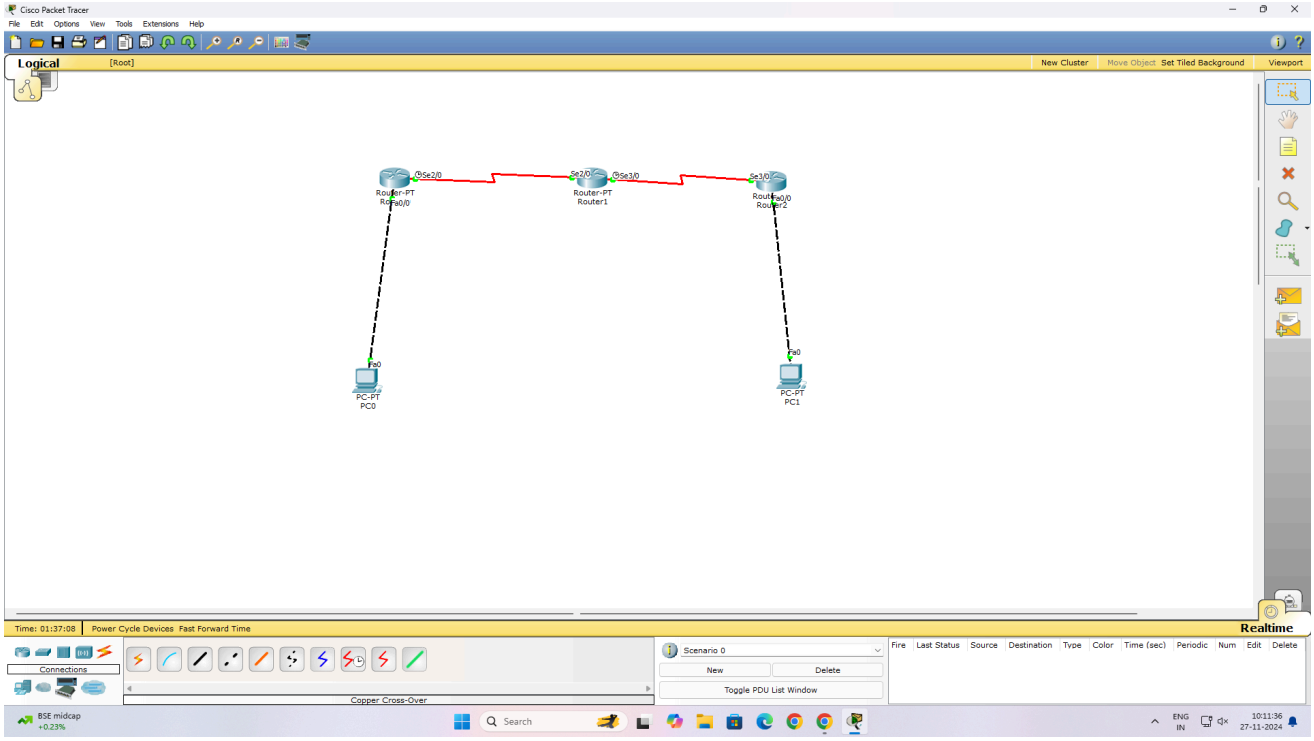
Observation

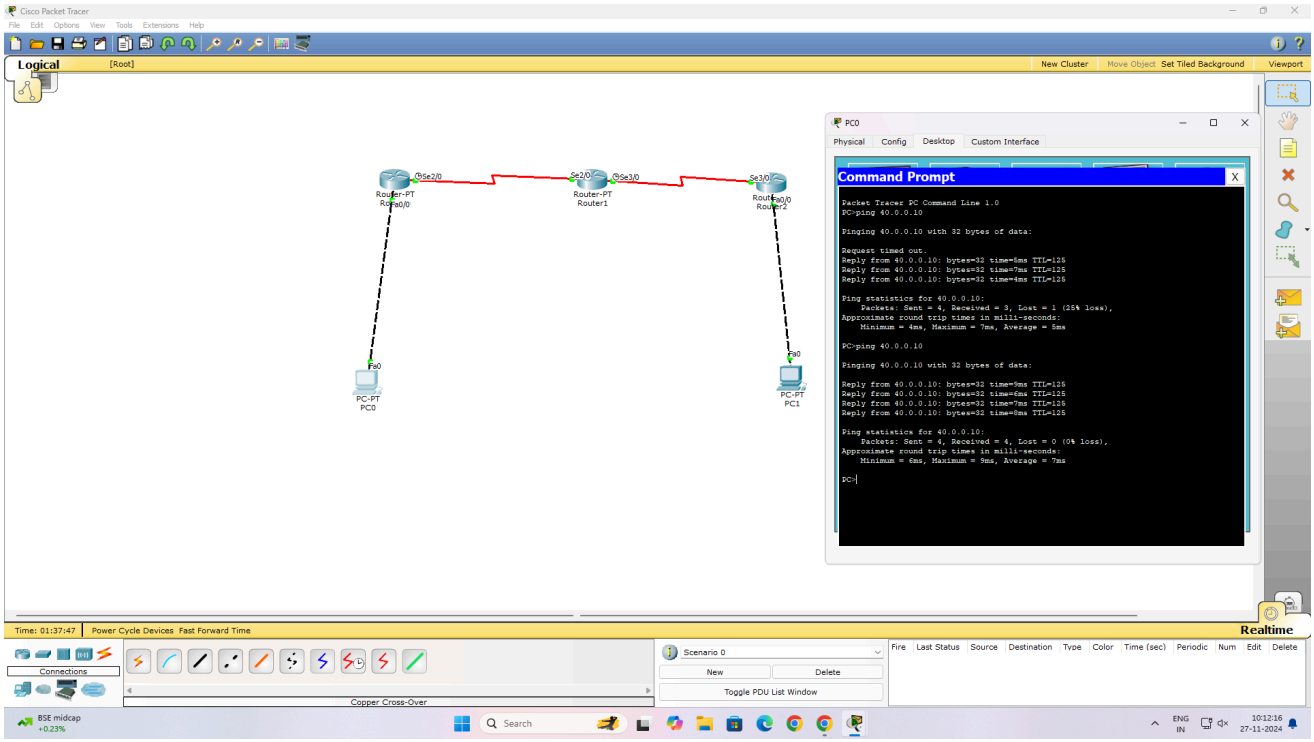
PC > ping 40.0.0.10
Pinging 40.0.0.10 with 32 bytes of data:
Reply from 40.0.0.10: bytes = 32 time = 9ms TTL = 125
" time = 6ms "
" time = 7ms "
" time = 8ms "

Ping statistics for 40.0.0.10:
Packets: sent = 4, received = 4, lost = 0 (0% loss),
approx. round trip times in ms:
Min = 6ms, Max = 9ms, Avg = 7ms

lll 27/11/24

Screen shots/ output:

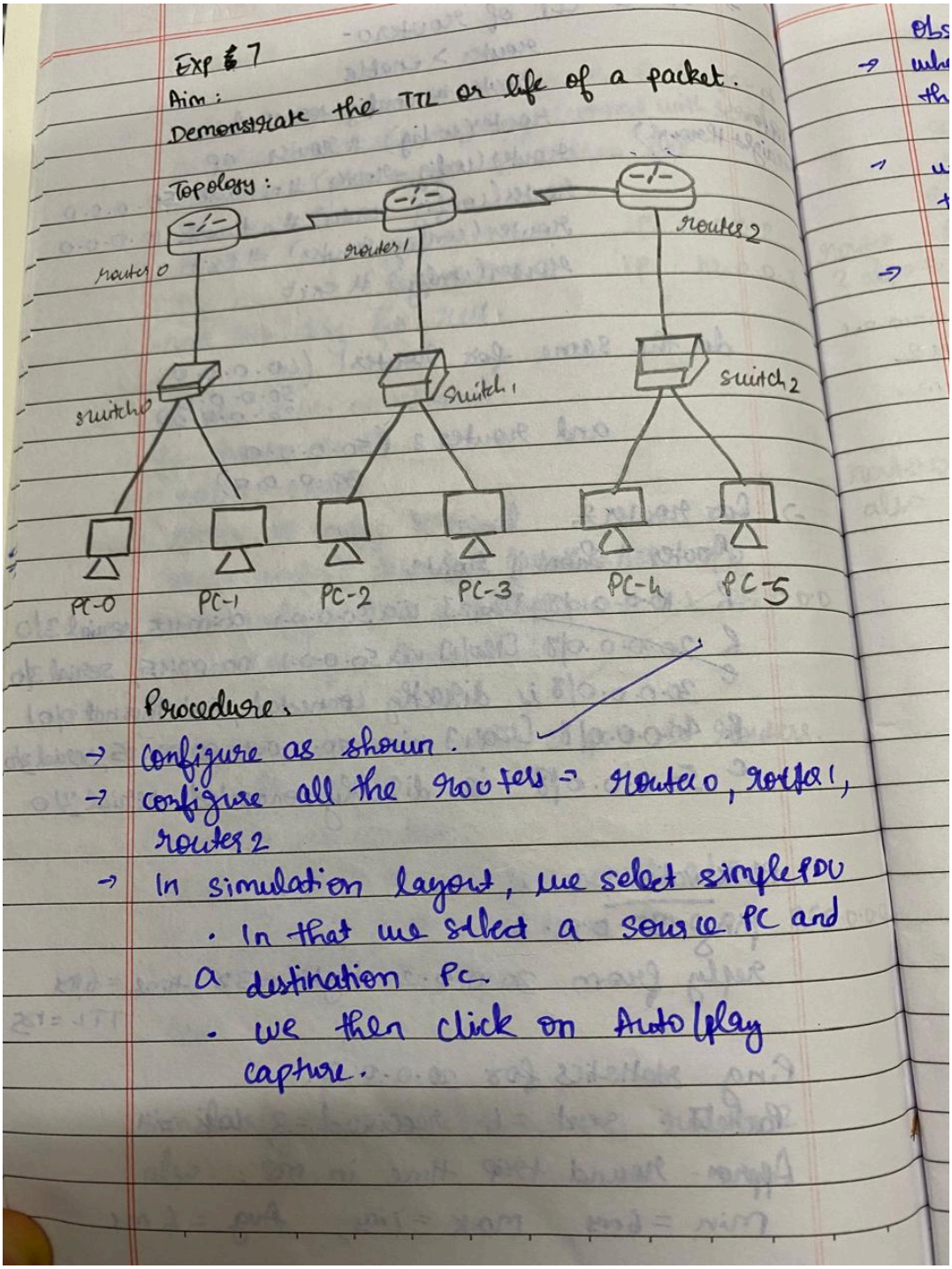




Program 7

Aim of the program:
Demonstrate the TTL/ Life of a Packet

Procedure along with the topology:



Observation:

→ when the packet arrives at router 0, the TTL = 255

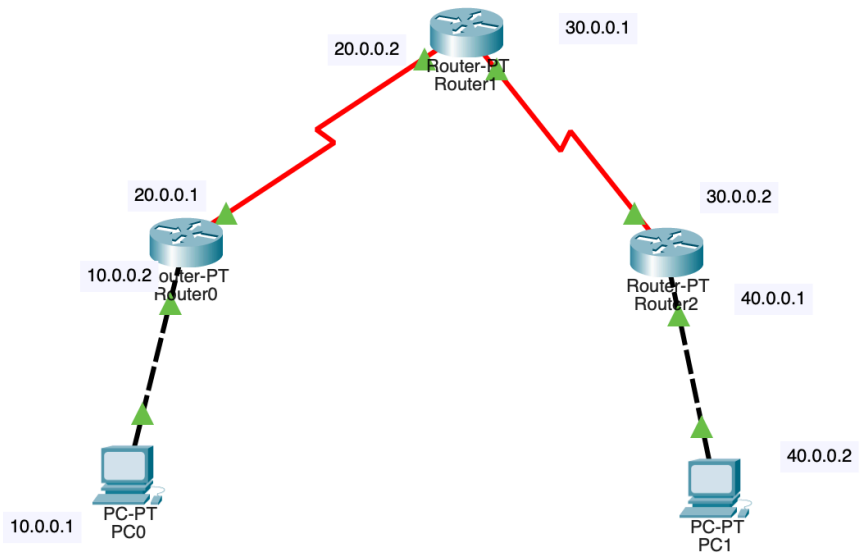
→ when the packet arrives at router 1, the TTL = 254

→ when the packet arrives at router 2, the TTL = 253.

so from each router, the TTL reduces by 1.

20/11/24

Screen shots/ output:



PDU Information at Device: Router0

OSI Model | Inbound PDU Details | Outbound PDU Details

PDU Formats

Ethernet II

0	4	8	14	19 Byt
PREAMBLE: 101010...1011		DEST MAC: 0001.C780.D8C7		SRC MAC: 00E0.F711.E727
TYPE: 0x800	DATA (VARIABLE LENGTH)			FCS: 0x0

IP

0	4	8	16 19	31 Bits
4	IHL	DSCP: 0x0	TL: 28	
ID: 0x4		0x0	0x0	
TTL: 255	PRO: 0x1	CHKSUM		
SRC IP: 10.0.0.1				
DST IP: 40.0.0.2				
OPT: 0x0		0x0		
DATA (VARIABLE LENGTH)				

ICMP

0	8	16	31 Bits
TYPE: 0x8	CODE: 0x0	CHECKSUM	
ID: 0x5		SEQ NUMBER: 4	

PDU Information at Device: Router0

OSI Model | Inbound PDU Details | Outbound PDU Details

PDU Formats

HDLC

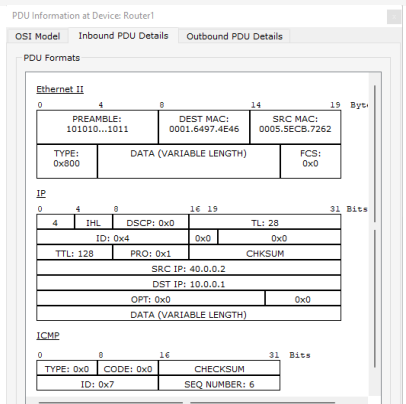
0	8	16	32	32+x	48+x	56+x
FLG: 0111 1110	ADR: 0x8f	CONTROL: 0x0	DATA: (VARIABLE LENGTH)		FCS: 0x0	FLG: 0111 1110

IP

0	4	8	16 19	31 Bits
4	IHL	DSCP: 0x0	TL: 28	
ID: 0x6		0x0	0x0	
TTL: 253	PRO: 0x1	CHKSUM		
SRC IP: 10.0.0.1				
DST IP: 40.0.0.2				
OPT: 0x0		0x0		
DATA (VARIABLE LENGTH)				

ICMP

0	8	16	31 Bits
TYPE: 0x8	CODE: 0x0	CHECKSUM	
ID: 0x7		SEQ NUMBER: 6	



PDU Information at Device: Router0

OSI Model | Inbound PDU Details | Outbound PDU Details

PDU Formats

HDLC

0	8	16	32	32+x	48+x	56+x
FLG: 0111 1110	ADR: 0x8f	CONTROL: 0x0	DATA: (VARIABLE LENGTH)		FCS: 0x0	FLG: 0111 1110

IP

0	4	8	16 19	31 Bits
4	IHL	DSCP: 0x0	TL: 28	
ID: 0x4		0x0	0x0	
TTL: 124	PRO: 0x1	CHKSUM		
SRC IP: 40.0.0.2				
DST IP: 10.0.0.1				
OPT: 0x0		0x0		
DATA (VARIABLE LENGTH)				

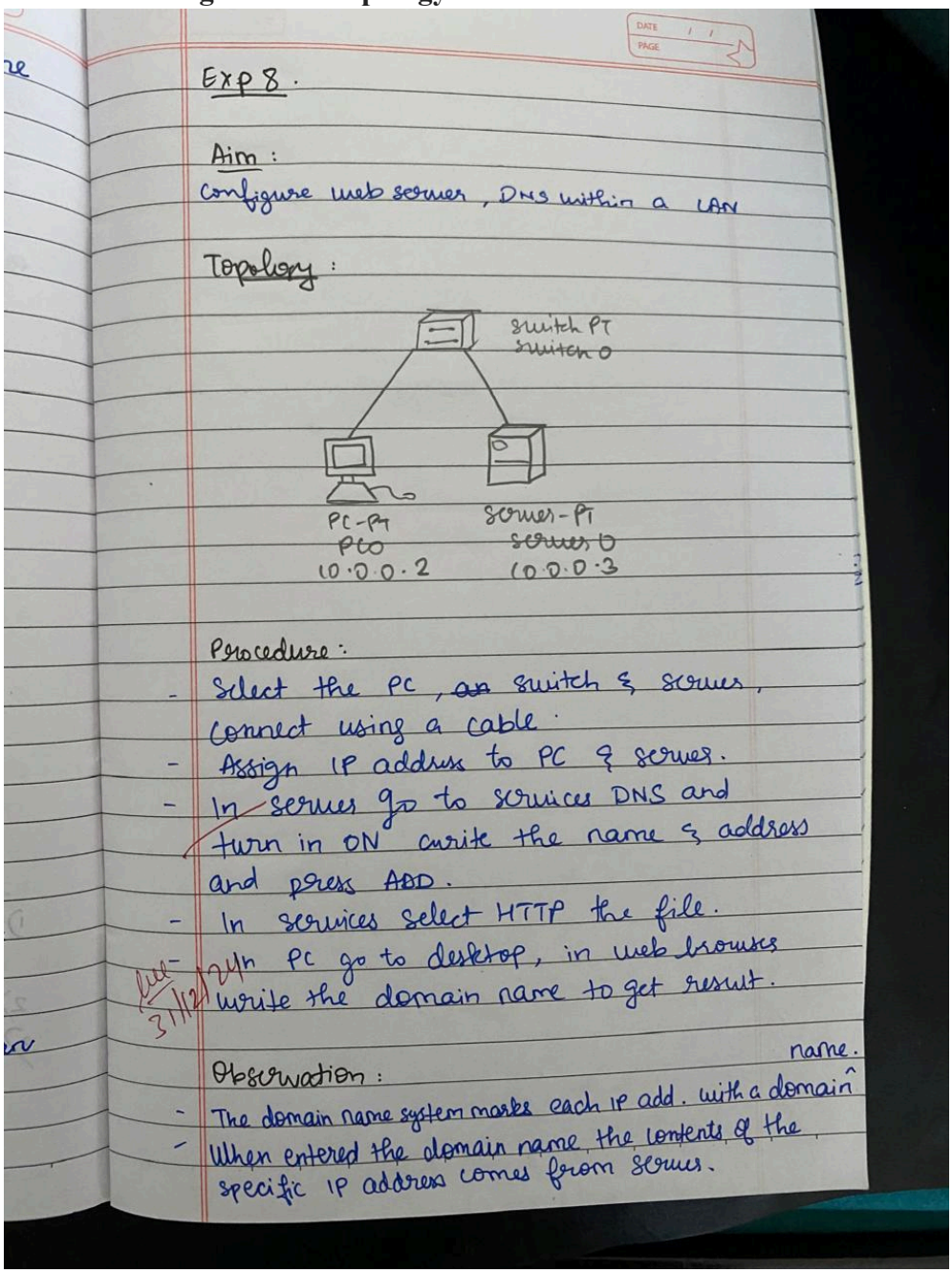
ICMP

0	8	16	31 Bits
TYPE: 0x0	CODE: 0x0	CHECKSUM	
ID: 0x7		SEQ NUMBER: 6	

Program 8

Aim of the program:
Configure Web Server, DNS within a LAN.

Procedure along with the topology:



Exp 8.

Aim :
Configure web server, DNS within a LAN

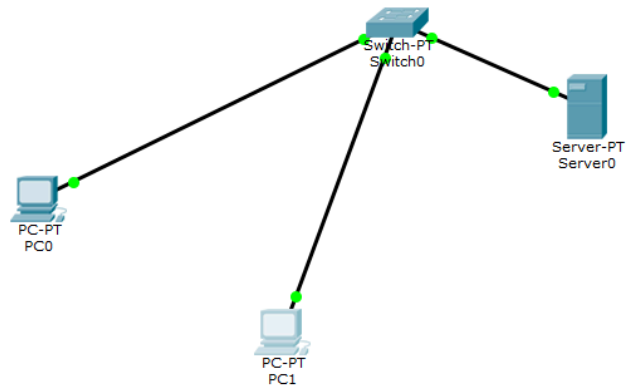
Topology :

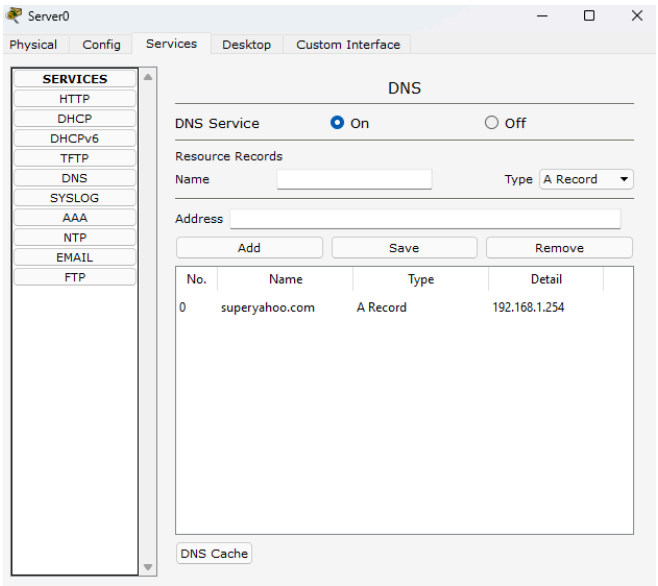
Procedure :
- Select the PC, switch & server, connect using a cable.
- Assign IP address to PC & server.
- In server go to services DNS and turn in ON write the name & address and press ADD.
- In services select HTTP the file.
- In PC go to desktop, in web browser write the domain name to get result.

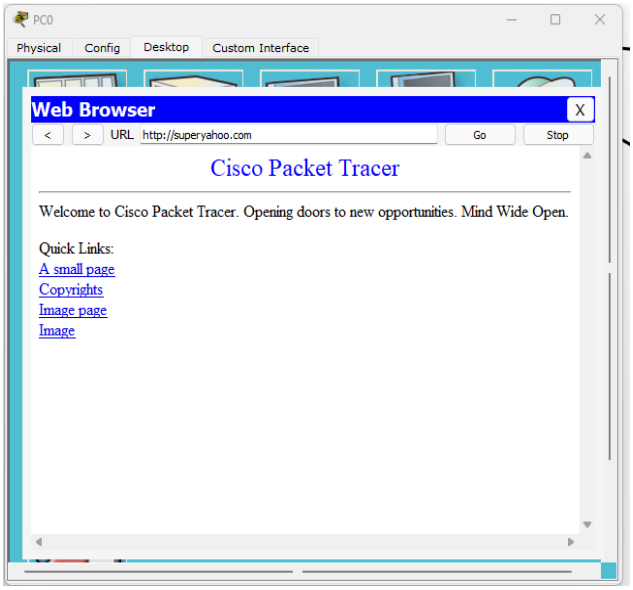
31/12/24

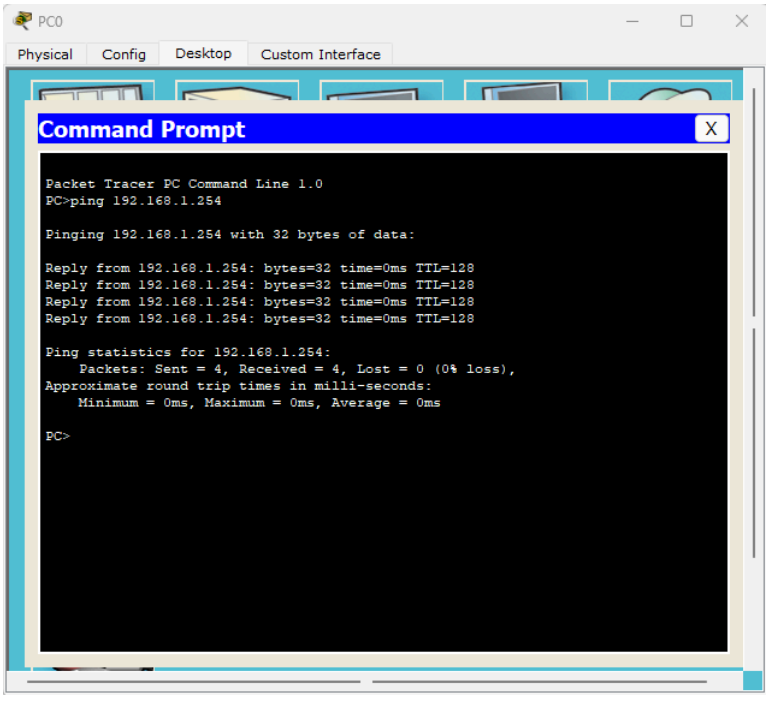
Observation :
- The domain name system marks each IP add. with a domain name.
- When entered the domain name the contents of the specific IP address comes from server.

Screen shots/ output:





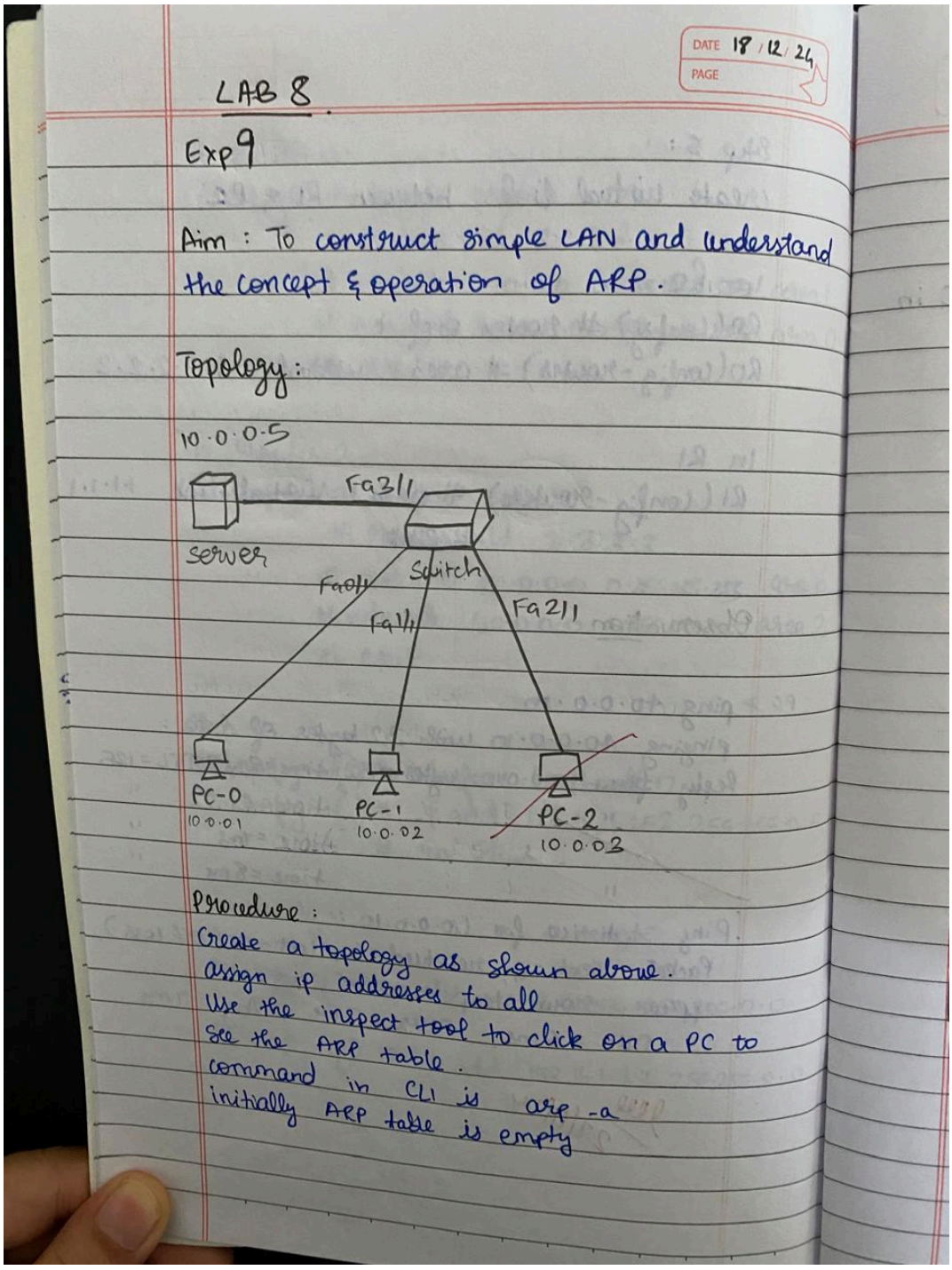




Program 9

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

Procedure along with the topology:



In CLI of switch, the command - show mac address-table on every transaction to see how switch learns from transactions & builds address-table.

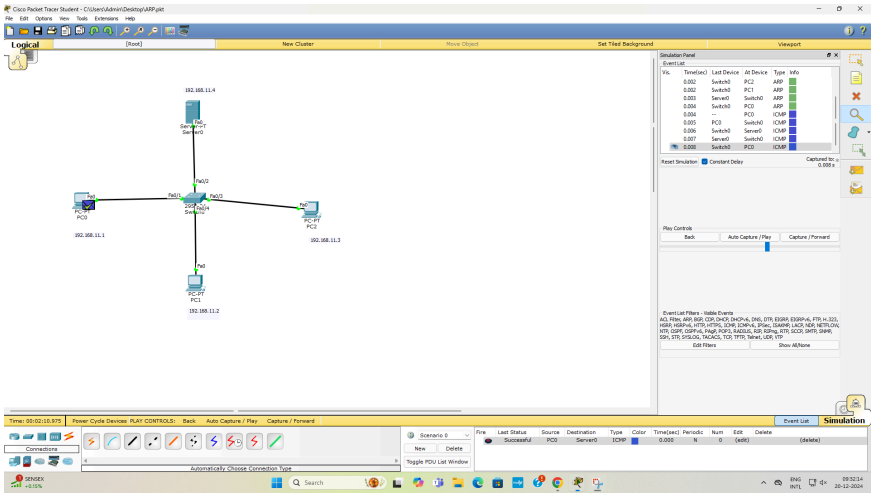
Go to simulation & use capture button so ARP changes are seen step by step.

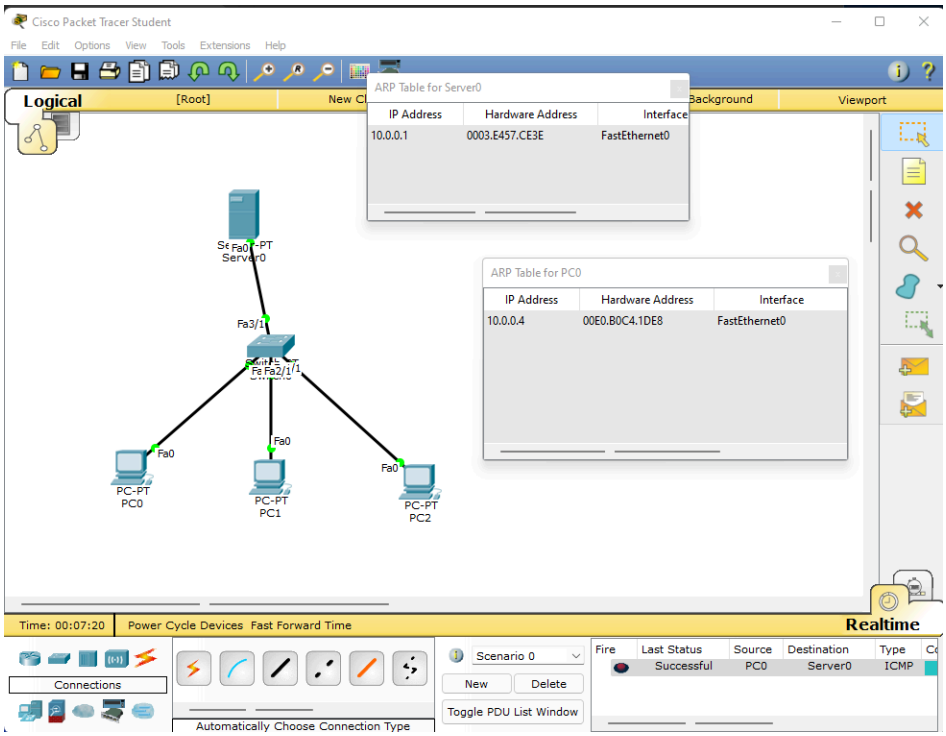
Add simple PDU select between 2 pcs and switch & PC. [capture/forward we can see ip add. of each table.]

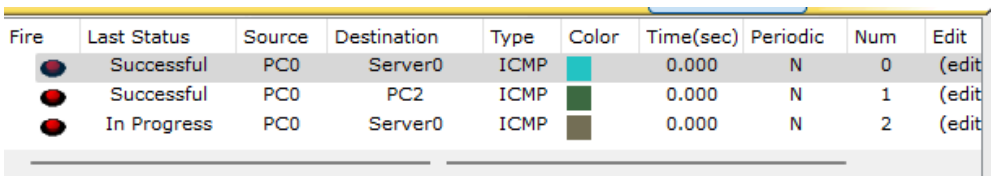
Observation:
We can see that through ARP protocol server has resolved the address of all PC's and send next message to an from having MAC table.

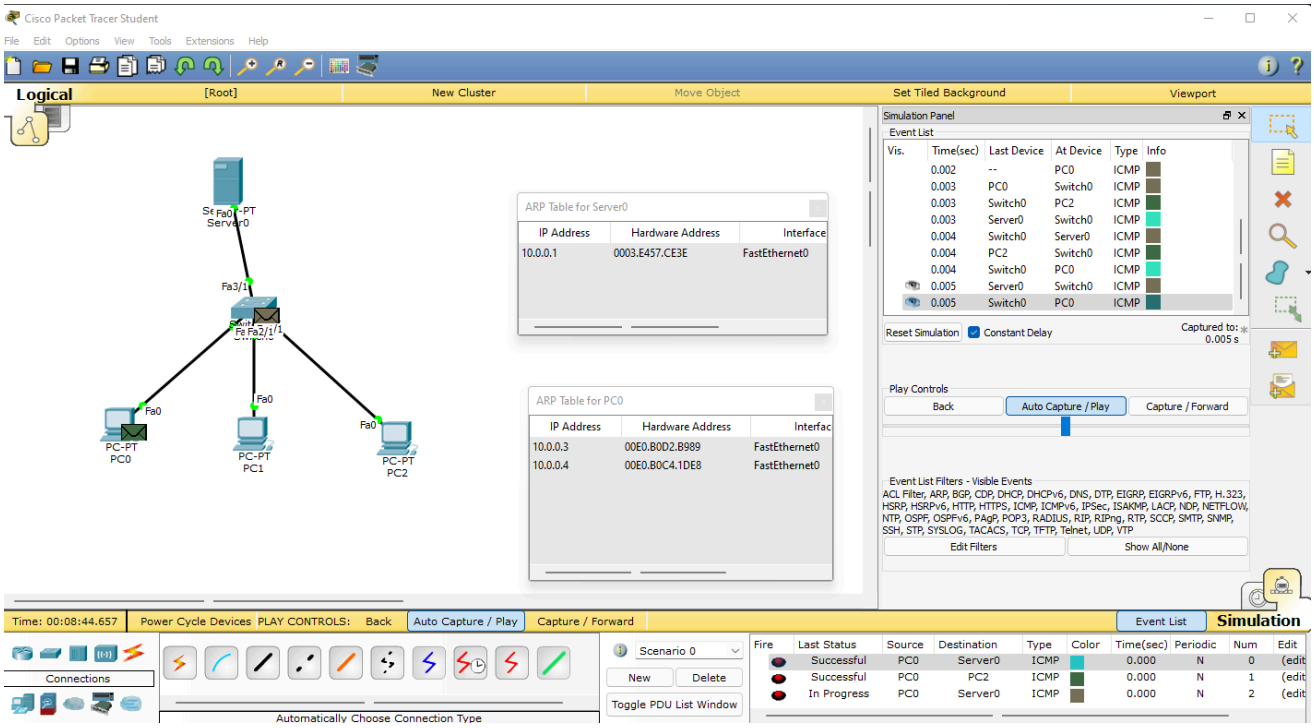
VLAN	MacAddress	Port
1	0001.6420.CC0A	Fast ethernet 1/1
1	0021.9738.C6E0	Fast ethernet 2/1
1	0002.4A1C.393B.A	Fast ethernet 3/1
1	000C.CF04.C5B2	Fast ethernet 0/1

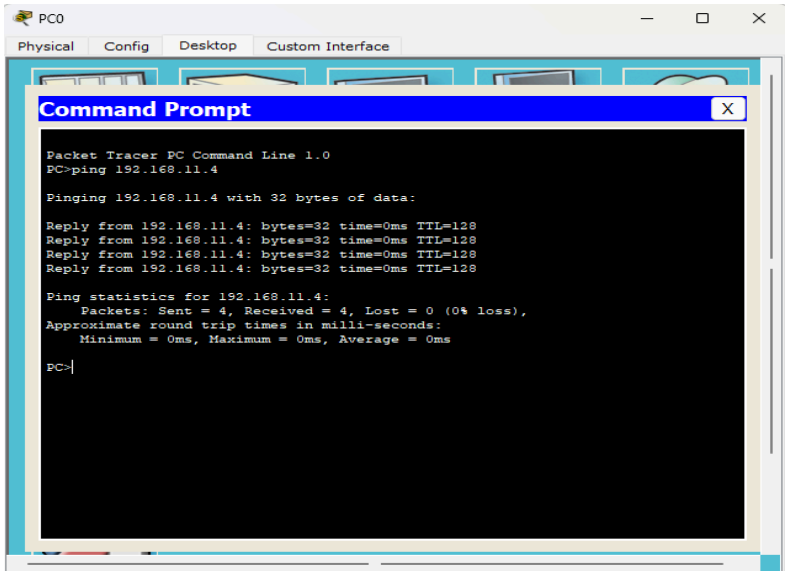
Screen shots/ output:

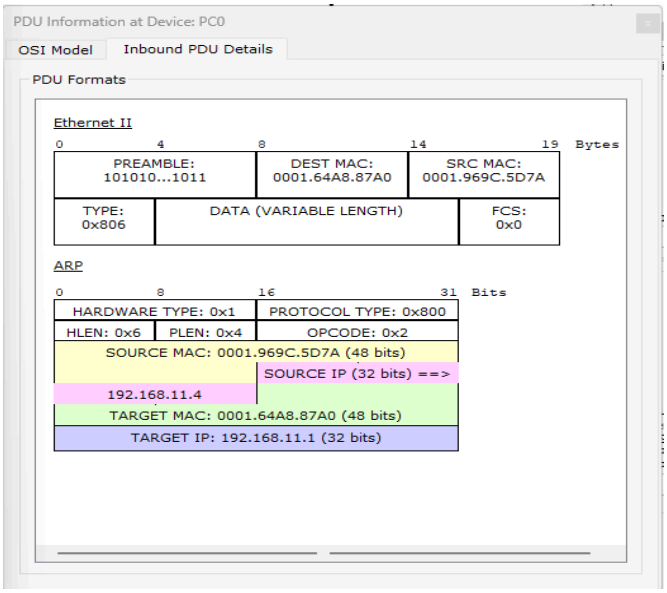







Cisco Packet Tracer Student
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Server0
Fa0
Fa3/1
Fa2/1/1
PC-PT PC0
PC-PT PC1
PC-PT PC2
ARP Table for Server0
IP Address
Hardware Address
Interface
10.0.0.1
0003.E457.CE3E
FastEthernet0
ARP Table for PC0
10.0.0.3
00E0.B0D2.B989
10.0.0.4
00E0.B0C4.1DE8
Simulation Panel
Event List
Vis.
Time(sec)
Last Device
At Device
Type
Info
Reset Simulation
Constant Delay
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
Edit Filters
Show All/None
Time: 00:08:44.657
Power Cycle Devices
PLAY CONTROLS:
Simulation
Connections
Automatically Choose Connection Type
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Color
Periodic
Num
Edit
Successful
In Progress

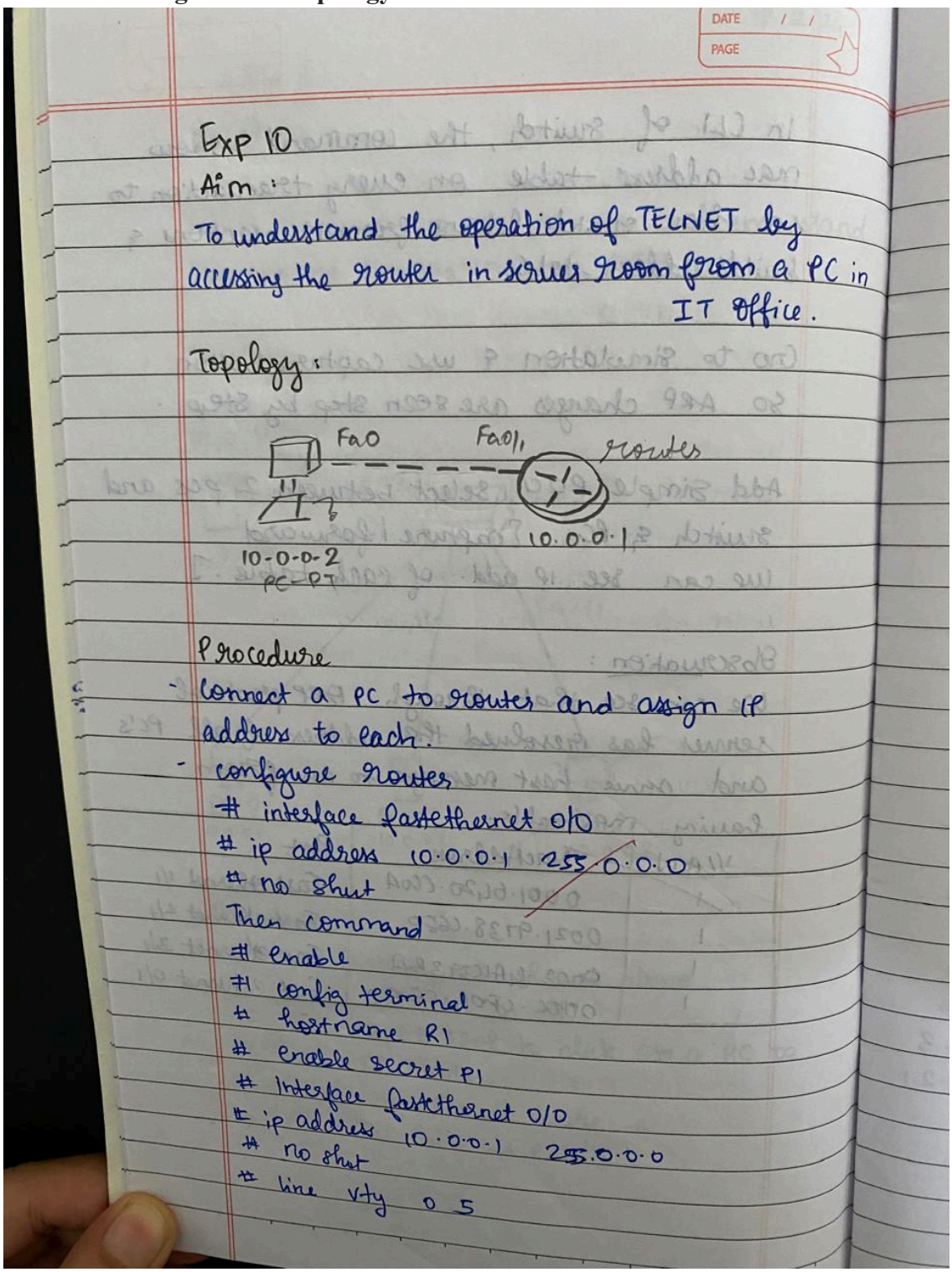

PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 192.168.11.4
Pinging 192.168.11.4 with 32 bytes of data:
Reply from 192.168.11.4: bytes=32 time=0ms TTL=128
Reply from 192.168.11.4: bytes=32 time=0ms TTL=128
Reply from 192.168.11.4: bytes=32 time=0ms TTL=128
Reply from 192.168.11.4: bytes=32 time=0ms TTL=128
Ping statistics for 192.168.11.4:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
PC>


PDU Information at Device: PC0
OSI Model
Inbound PDU Details
PDU Formats
Ethernet II
0
4
8
14
19
Bytes
PREAMBLE: 101010...1011
DEST MAC: 0001.64A8.87A0
SRC MAC: 0001.969C.5D7A
TYPE: 0x806
DATA (VARIABLE LENGTH)
FCS: 0x0
ARP
0
8
16
31
Bits
HARDWARE TYPE: 0x1
PROTOCOL TYPE: 0x800
HLEN: 0x6
PLEN: 0x4
OPCODE: 0x2
SOURCE MAC: 0001.969C.5D7A (48 bits)
SOURCE IP (32 bits) ==>
192.168.11.4
TARGET MAC: 0001.64A8.87A0 (48 bits)
TARGET IP: 192.168.11.1 (32 bits)

Program 10

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

Procedure along with the topology:

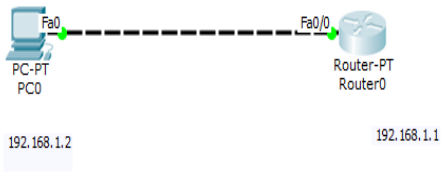


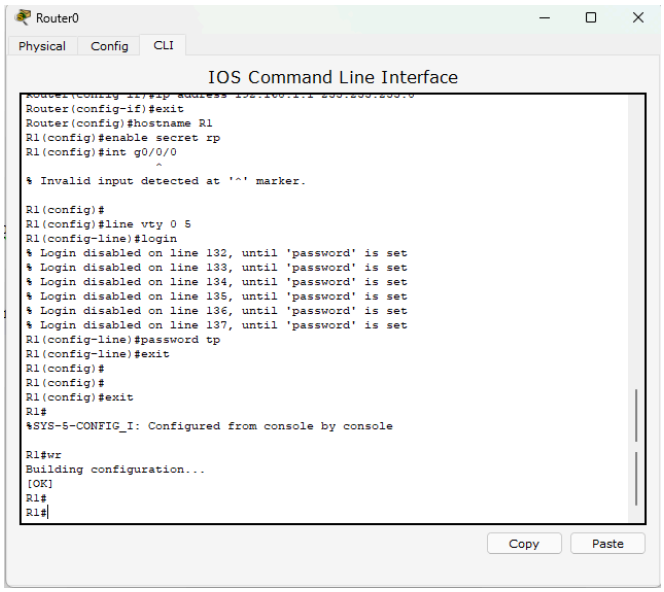
login
password p0
exit
exit
R1 # wr

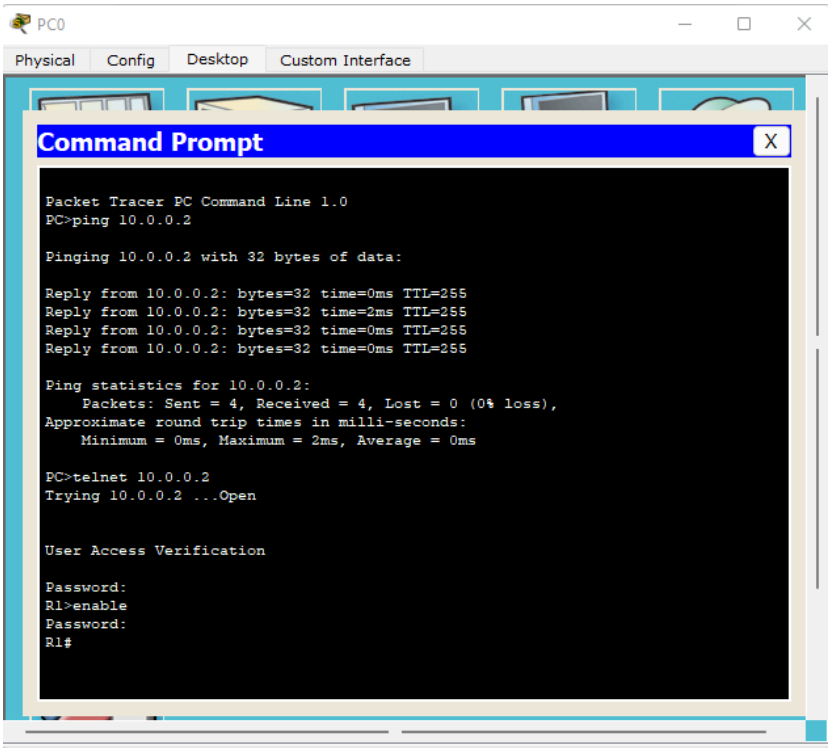
Observations:
It is observed that through the telnet the hostname & password is given to cli & router in any other device.
- In pc type ping to know its connected.
telnet 10.0.0.1 command access to R1 host in PC 0

User access verification
Password: p0
R1 > enable
Password: p1
R1 #

Screen shots/ output:





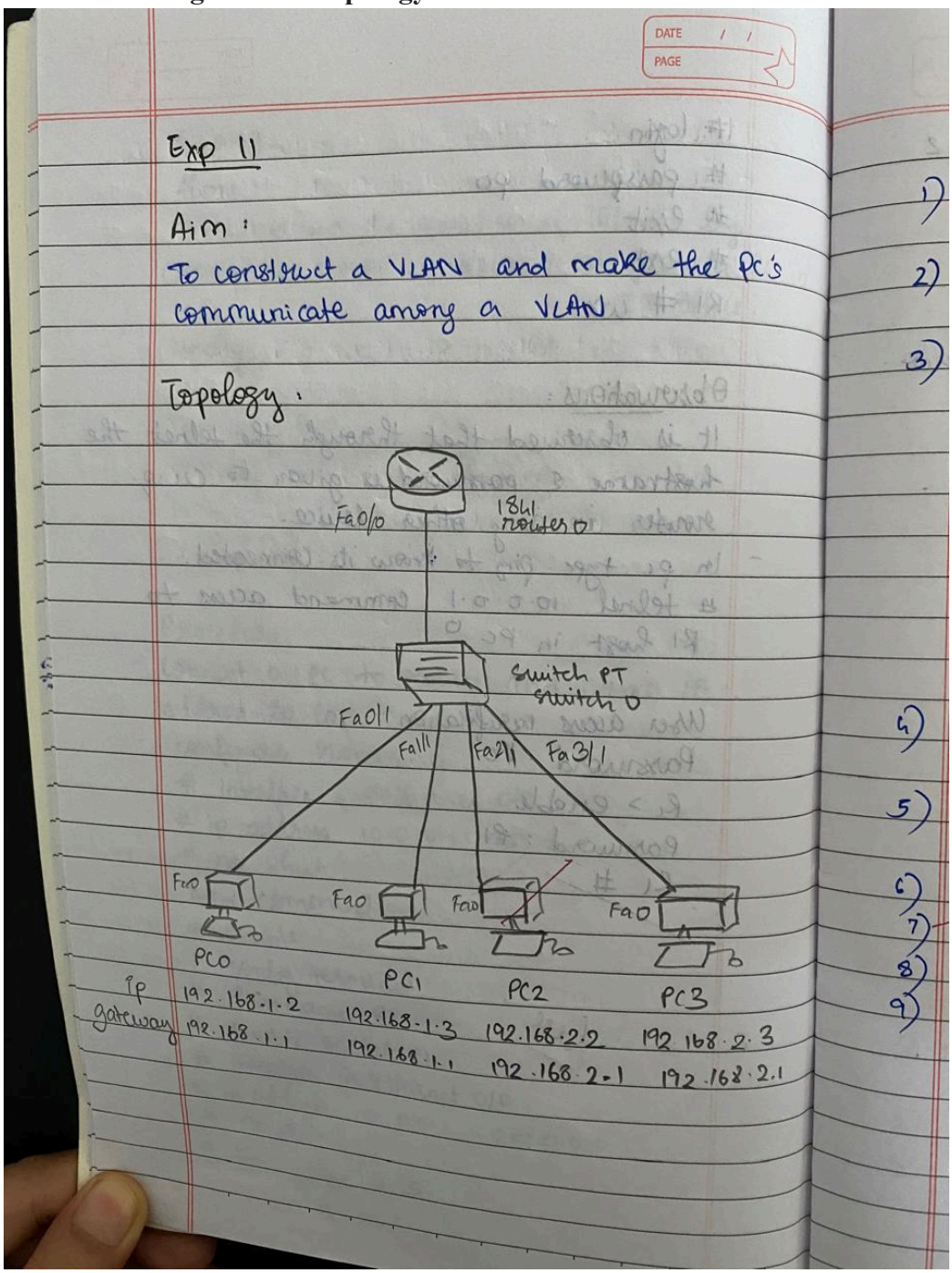


Program 11

Aim of the program:

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

Procedure along with the topology:



Procedure:

1) Take a router connect it to a switch & connect 4 PCs to switch - PT 0.
2) Configure ip address & gateways as shown in topology.
3) Go to router configure or fix 2 pcs in the router command.
   # enable
   # config terminal
   # interface fastethernet 0/0
   # ip address 192.168.1.1 255.255.255.0
   # no shut
   # exit
4) Then select switch, go to config, select VLAN database.
5) Set VLAN number to 2 and name, press add.
6) Do VLAN trunking
7) Go to fastethernet 0/1 select Trunck
8) Go to fastethernet 2/1 select the VLAN trunk
9) In ~~trunk~~ select VLAN database and enter the number & name of VLAN created.

Go to CLI - commands.
Router (VLAN) # exit
Router # config terminal
interface fastethernet 0/0.1

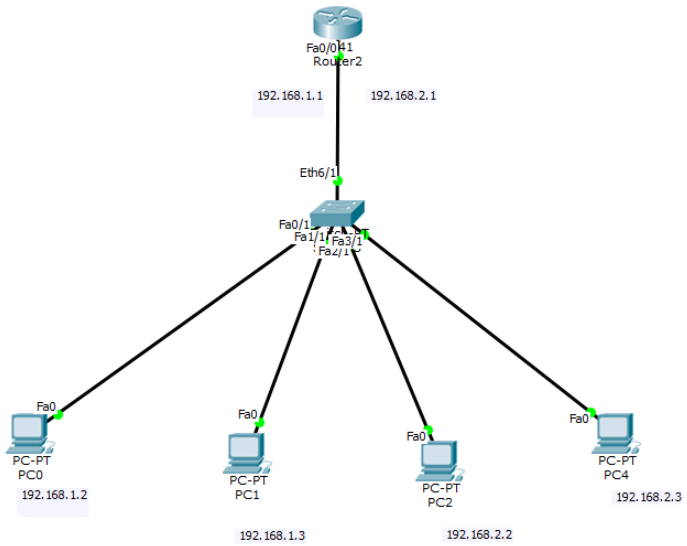
Router(config-sub)# encapsulation dot1q 2
ip address 192.168.2.1 255.255.255.0
no shut
exit
exit

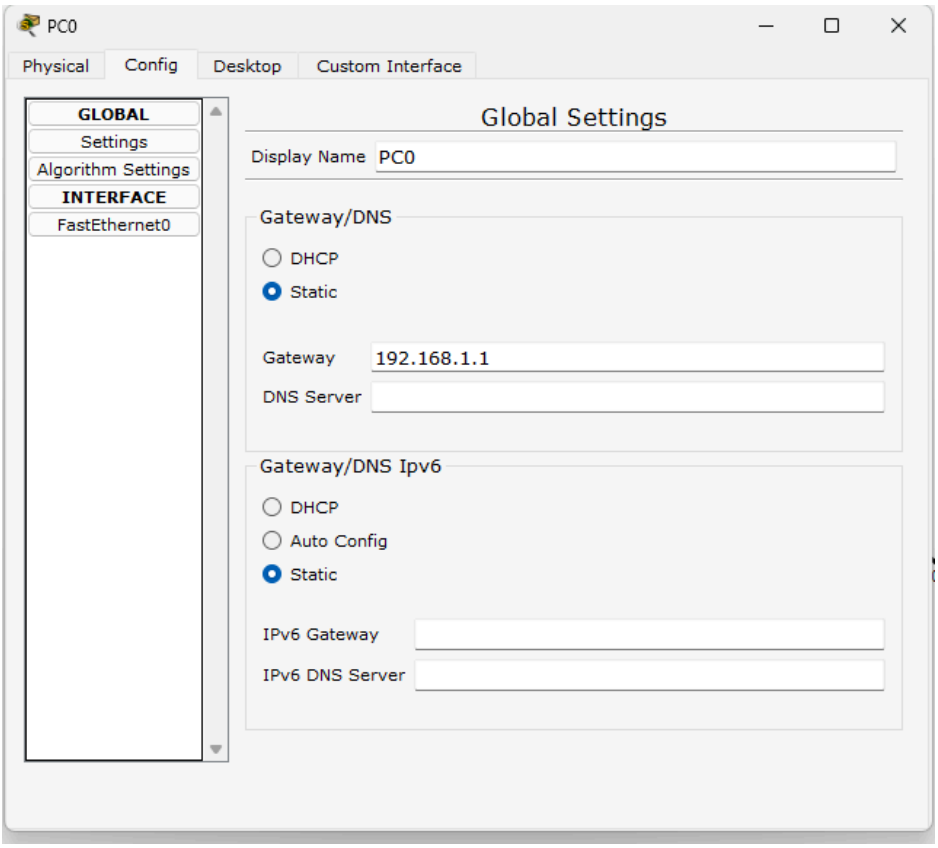
10) Ping from two routers to other two.

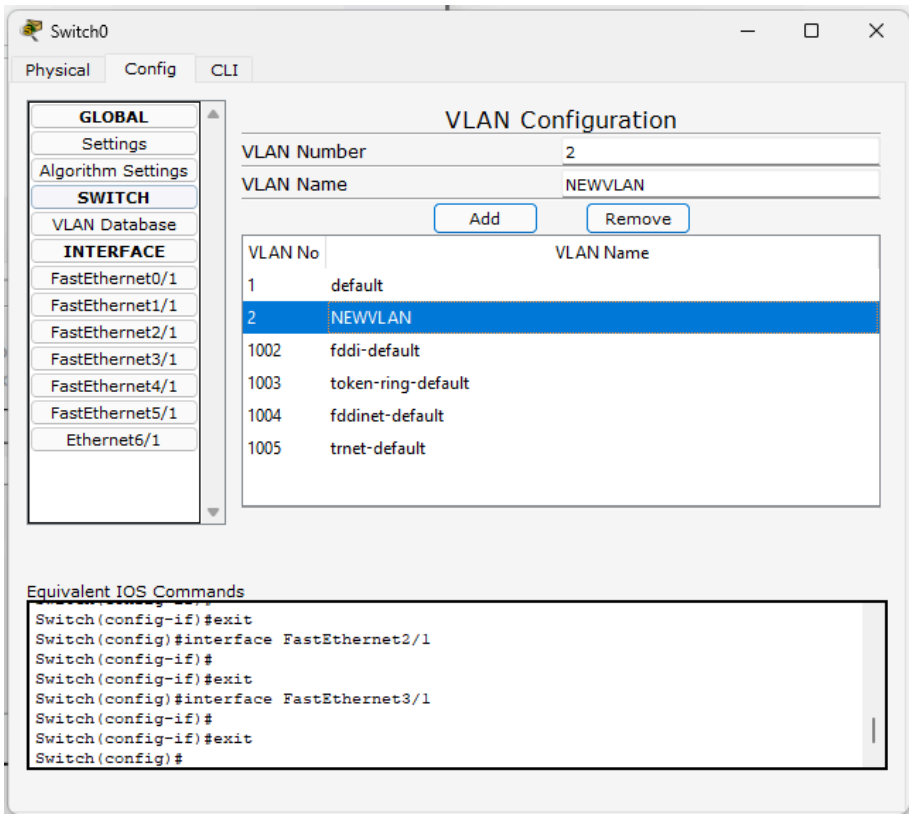
Observation:

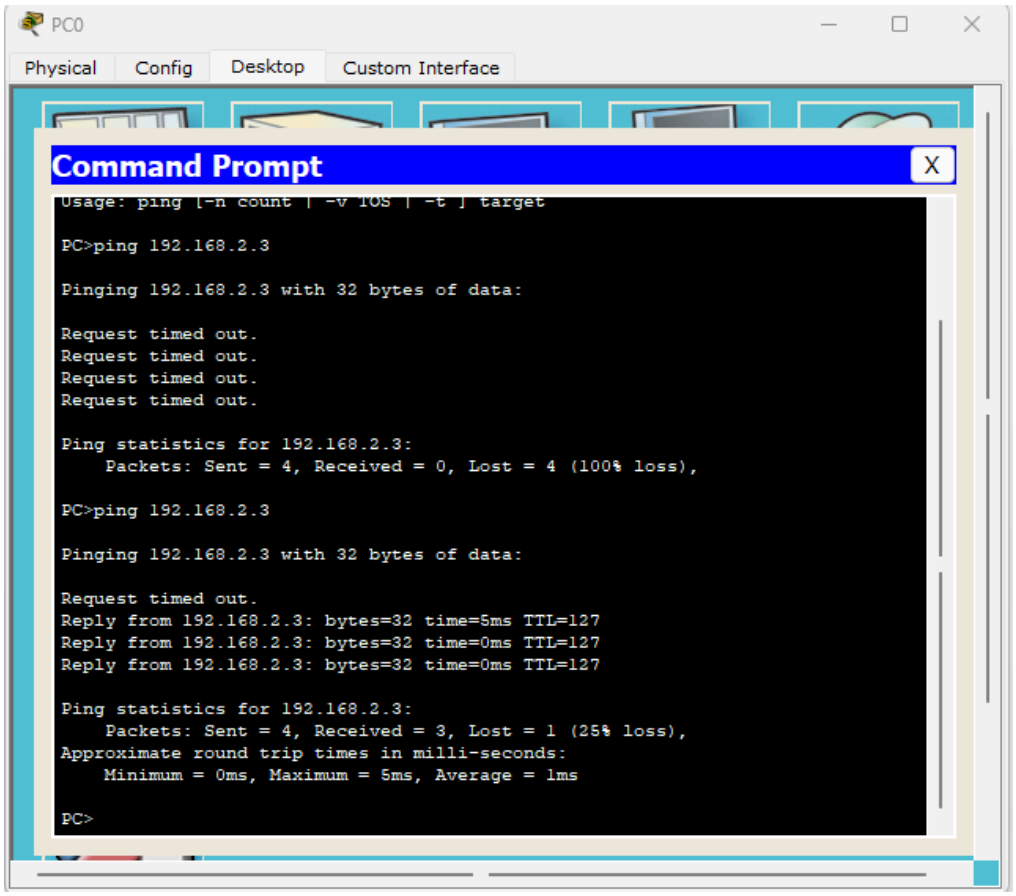
VLAN trunking allows switches to forward from different VLAN, over single linked cable trunk.
This is done by adding an additional header information called tag to the ethernet frame. The process of adding this is is called VLAN tagging.
It makes switch understand NEW VLAN.

Screen shots/ output:





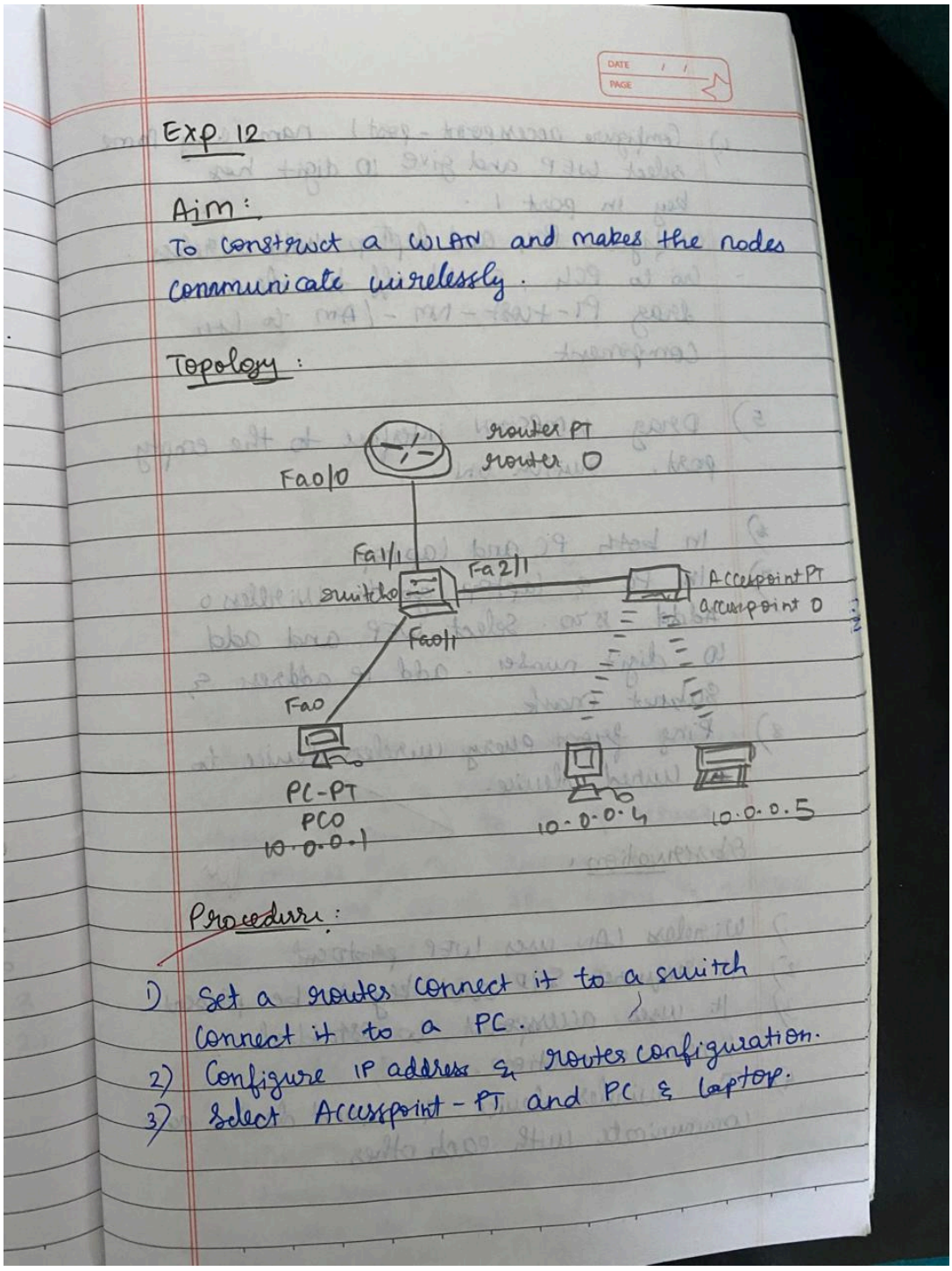

Switch0
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
SWITCH
VLAN Database
INTERFACE
FastEthernet0/1
FastEthernet1/1
FastEthernet2/1
FastEthernet3/1
FastEthernet4/1
FastEthernet5/1
Ethernet6/1
VLAN Configuration
VLAN Number
2
VLAN Name
NEWVLAN
Add
Remove
VLAN No
VLAN Name
1 default
2 NEWVLAN
1002 fddi-default
1003 token-ring-default
1004 fddinet-default
1005 trnet-default
Equivalent IOS Commands
Switch(config-if)#exit
Switch(config)#interface FastEthernet2/1
Switch(config-if)#
Switch(config-if)#exit
Switch(config)#interface FastEthernet3/1
Switch(config-if)#
Switch(config-if)#exit
Switch(config)#


PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Usage: ping [-n count | -v TOS | -t] target
PC>ping 192.168.2.3
Pinging 192.168.2.3 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 192.168.2.3:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>ping 192.168.2.3
Pinging 192.168.2.3 with 32 bytes of data:
Request timed out.
Reply from 192.168.2.3: bytes=32 time=5ms TTL=127
Reply from 192.168.2.3: bytes=32 time=0ms TTL=127
Reply from 192.168.2.3: bytes=32 time=0ms TTL=127
Ping statistics for 192.168.2.3:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 5ms, Average = 1ms
PC>

Program 12

Aim of the program:
To construct a WLAN and make the nodes communicate wirelessly

Procedure along with the topology:



4) Configure accesspoint - port 1 name any name select WEP and give 10 digit hex key in port 1.
Configure PCs and laptop with wireless.
- Go to PC, switch off the device, drag PT-Host-NM-1AM to LH, component

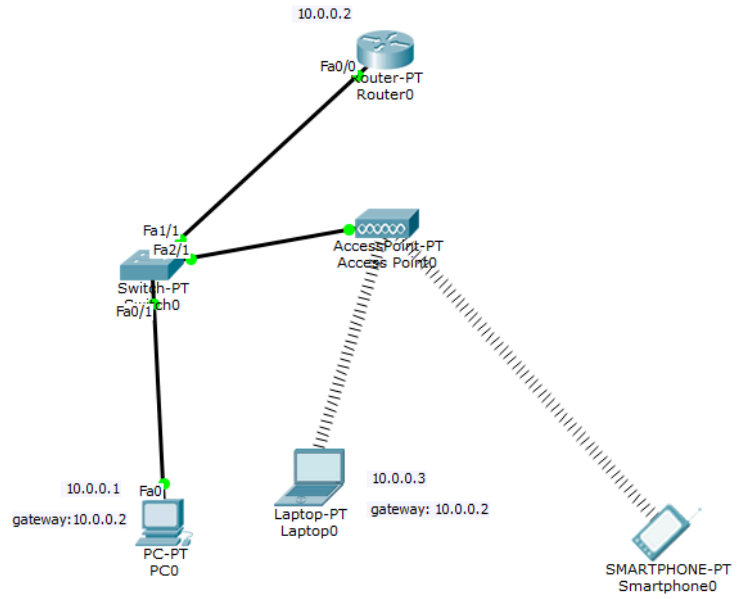
5) Drag WMP300N interface to the empty port, switch ON.

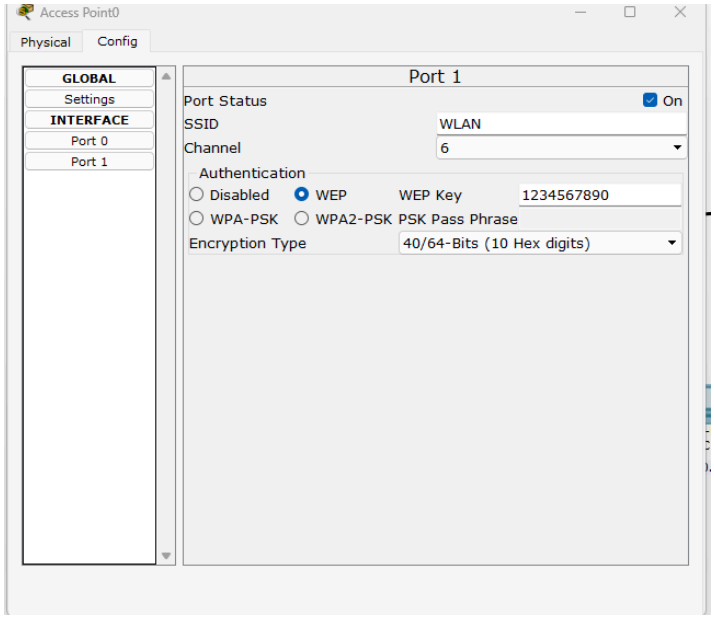
6) In both PC and laptop
7) In PC & laptop go to Wireless 0 Addd ssid. Select WEP and add 10 digit number. add IP address & subnet mask.
8) Ping from every wireless device to a wired device.

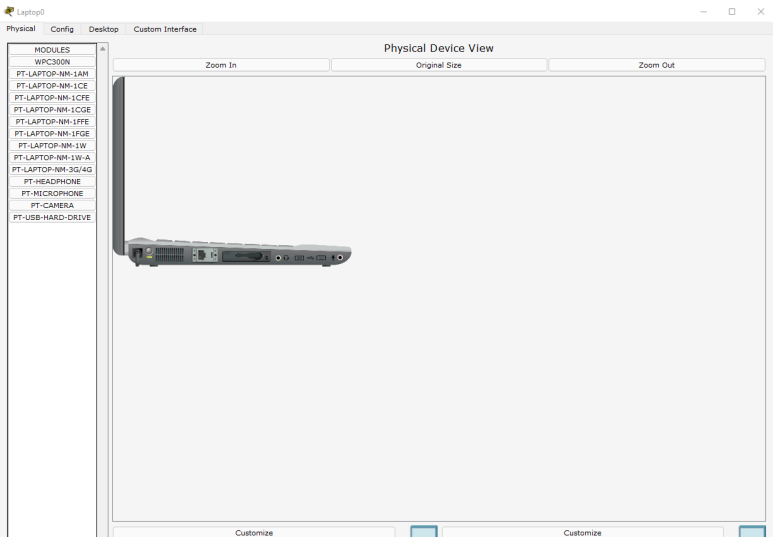
Observation:

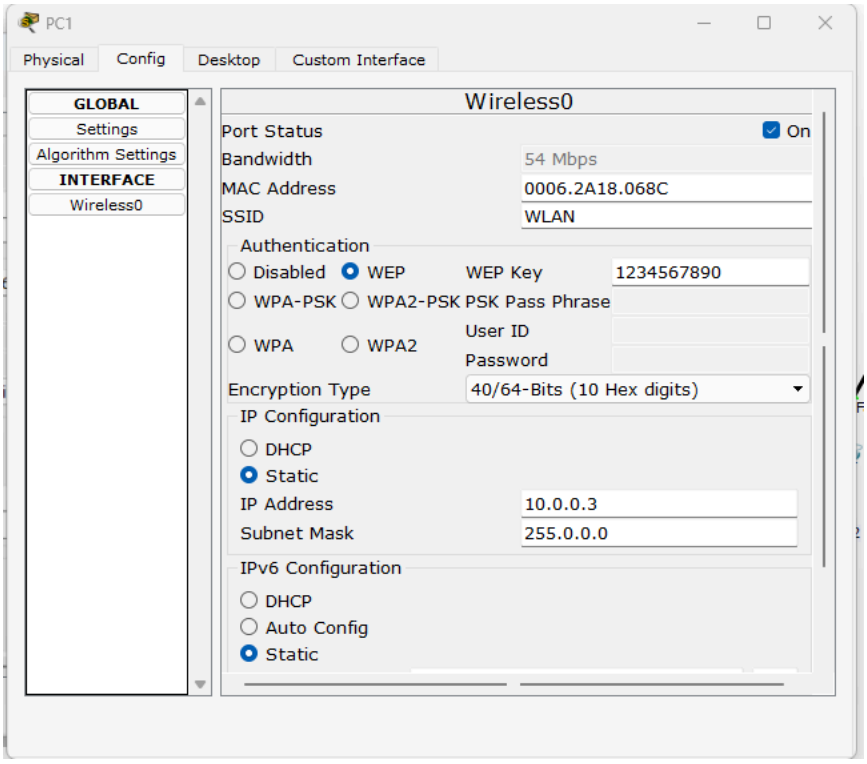
1) Wireless LAN uses WEP protocol
2) It requires SSID and key to be present
3) It uses accesspoint to establish wireless connection.
4) The wireless device & wired device can communicate with each other.

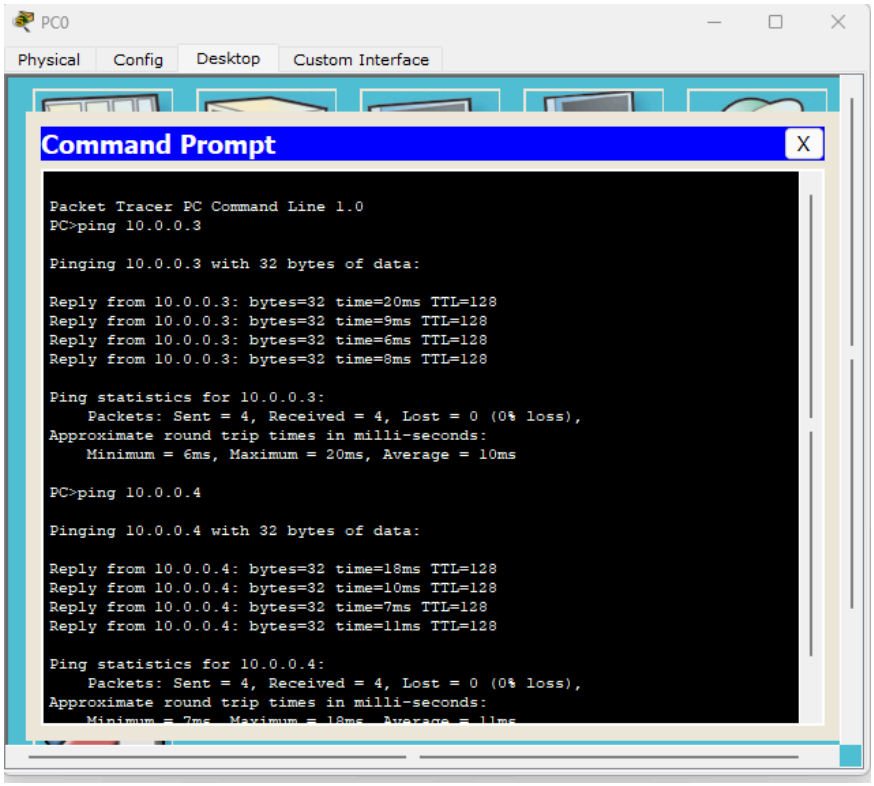
Screen shots/ output :








PC1
Physical
Config
Desktop
Custom Interface
GLOBAL
Settings
Algorithm Settings
INTERFACE
Wireless0
Wireless0
Port Status
On
Bandwidth
54 Mbps
MAC Address
0006.2A18.068C
SSID
WLAN
Authentication
Disabled
WEP
WEP Key
1234567890
WPA-PSK
WPA2-PSK
PSK Pass Phrase
WPA
WPA2
User ID
Password
Encryption Type
40/64-Bits (10 Hex digits)
IP Configuration
DHCP
Static
IP Address
10.0.0.3
Subnet Mask
255.0.0.0
IPv6 Configuration
DHCP
Auto Config
Static


PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.3
Pinging 10.0.0.3 with 32 bytes of data:
Reply from 10.0.0.3: bytes=32 time=20ms TTL=128
Reply from 10.0.0.3: bytes=32 time=9ms TTL=128
Reply from 10.0.0.3: bytes=32 time=6ms TTL=128
Reply from 10.0.0.3: bytes=32 time=8ms 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 = 6ms, Maximum = 20ms, Average = 10ms
PC>ping 10.0.0.4
Pinging 10.0.0.4 with 32 bytes of data:
Reply from 10.0.0.4: bytes=32 time=18ms TTL=128
Reply from 10.0.0.4: bytes=32 time=10ms TTL=128
Reply from 10.0.0.4: bytes=32 time=7ms TTL=128
Reply from 10.0.0.4: bytes=32 time=11ms TTL=128
Ping statistics for 10.0.0.4:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:

CYCLE-2

Program 1

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

Code:

```
#include<stdio.h>
#include<string.h>
#define N strlen(gen_poly)

char data[28], check_value[28], gen_poly[10];
int data_length,i,j;

void XOR(){
   for(j = 1;j < N; j++)
   check_value[j] = (( check_value[j] == gen_poly[j])?'0':'1');
}

void receiver(){

   printf("Enter the received data: ");
   scanf("%s", data);
   printf("\n-----------------------------\n");
   printf("Data received: %s", data);

   crc();

   for(i=0;(i<N-1) && (check_value[i]!='1');i++);
      if(i<N-1)
         printf("\nError detected\n\n");
      else
         printf("\nNo error detected\n\n");
}


void crc(){
   for(i=0;i<N;i++)
      check_value[i]=data[i];
   do{
      if(check_value[0]=='1')
         XOR();
      for(j=0;j<N-1;j++)
         check_value[j]=check_value[j+1];
      check_value[j]=data[i++];
   }while(i<=data_length+N-1);
}

int main()
```

```
{
   printf("\nEnter data to be transmitted: ");
   scanf("%s",data);
   printf("\n Enter the Generating polynomial: ");
   scanf("%s",gen_poly);
   data_length=strlen(data);
   for(i=data_length;i<data_length+N-1;i++)
      data[i]='0';
   printf("\n----------------------------------------");
   printf("\n Data padded with n-1 zeros : %s",data);
   printf("\n----------------------------------------");
   crc();
   printf("\nCRC or Check value is : %s",check_value);
   for(i=data_length;i<data_length+N-1;i++)
      data[i]=check_value[i-data_length];
   printf("\n----------------------------------------");
   printf("\n Final data to be sent : %s",data);
   printf("\n----------------------------------------\n");
   receiver();
      return 0;
}
```

OUTPUT:

```
Enter data to be transmitted: 1001100
Enter the Generating polynomial: 100001011

----------------------------------------
Data padded with n-1 zeros: 100110000000000
CRC or Check value is: 0100010
Final data to be sent: 10011000100010
----------------------------------------

Enter the received data: 10011000100011

-----------------------------
Data received: 10011000100011
Error detected
```

Cycle 2 -
Experiment 13

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

CODE :

```
def crc(ip, poly, mode):
    op = list(ip)
if mode:
    op.extend('0' * (len(poly)-1))

for i in range(len(ip)):
    if op[i] == '1':
        for j in range(len(poly)):
            if i+j < len(op):
                op[i+j] = '0' if op[i+j] == poly[j]
                    else '1'

if mode:
    return ip + ''.join(op[len(ip):])
return all(bit == '0' for bit in op[len(ip):])

if __name__ == '__main__':
    poly = "10001000000100001"
ip = input("Enter the input message in binary:")
transmitted_msg = crc(ip, poly, 1)
```

```
print(f"The transmitted message in binary: ")
if recv == ip:
    print("No error in data")
else:
    print("error in transmision has occured")
```

Output:

Enter the input message in binary : 1101
The transmitted message is : 1101110100011010
Enter the received message in binary : 1101
no error in data.

Program 2

Aim of the program:
Write a program for congestion control using Leaky bucket algorithm.

Code:

```
# initial packets in the bucket
storage = 0

# total no. of times bucket content is checked
no_of_queries = 4

# total no. of packets that can
# be accommodated in the bucket
bucket_size = 10

# no. of packets that enters the bucket at a time
input_pkt_size = 4

# no. of packets that exits the bucket at a time
output_pkt_size = 1
for i in range(0, no_of_queries):  # space left

    size_left = bucket_size - storage
    if input_pkt_size <= size_left:
      # update storage
        storage += input_pkt_size
    else:
        print("Packet loss = ", input_pkt_size)

    print(f"Buffer size= {storage} out of bucket size = {bucket_size}")

    # as packets are sent out into the network, the size of the storage decreases
    storage -= output_pkt_size
```

OUTPUT:

```
Enter initial packets in the bucket: 0
Enter total no. of times bucket content is checked: 4
Enter total no. of packets that can be accommodated in the bucket: 10
Enter no. of packets that enters the bucket at a time: 4
Enter no. of packets that exits the bucket at a time: 1
Buffer size = 4 out of bucket size = 10
Buffer size = 7 out of bucket size = 10
Buffer size = 10 out of bucket size = 10
Packet loss = 4
Buffer size = 9 out of bucket size = 10
```

Experiment 14

Aim:
Write a program for congestion control using leaky bucket algorithm.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define NOF_packets 5
/*
   int rand (int a)
   {
      int rn = (random() % 10) % a;
      return rn == 0 ? 1 : rn;
   }
*/

int main()
{ int packet_sz [NOF_packets], i, clk, b,
o_rate, p_sz, rm = 0, p_sz, p_time, op;
for (i = 0; i < NOF_packets; ++i)
    packet_sz[i] = random() % 100;
for (i = 0; i < NOF_packets; ++i)
 printf ("packet [%d] : %d bytes", i, packet_sz[i]);
```

```
printf("Enter the output rate ");
scanf("%d", &o_rate);
printf("Enter the bucket size :");
scanf("%d", &b_size);
for(i=0; i<NOF_packets ; ++i)
{
 if(packet_sz[i] + p_sz_rm) > b_size)
if (packet_sz[i] > b_size)
 printf("Incoming packet size (%d bytes)
is greater that bucket capacity (%d bytes).
PACKET REJECTED", packet_sz[i], b_size);
 else
printf("Bucket capacity exceeded -
        PACKETS REJECTED");

else
{ p_sz_rm += packet_sz[i];
printf("Incoming packet size : %d",
          packet_sz[i]);
printf("Bytes remaining to transmit : %d",
              p_sz_rm);

while (p_sz_rm > 0)

{ sleep(1);
    if (p_sz_rm)
    {
```

```
if (p_sz_rm <= o_rate)
  op = p_sz_rm, p_sz_rm = 0;
else
  op = o_rate, p_sz_rm = o_rate;
printf("Packet of size %d transmitted", op);
printf("\n Bytes remaining to transmit: %d
      p_sz_rm);
    }
  else
  {
   printf("No packets to transmit !! ");
    }
  }
 }
}
```

Program 3

Aim of the program:
Using TCP/IP sockets, write a client-server program to make the client sending the file name and the server to send back the contents of the requested file if present.

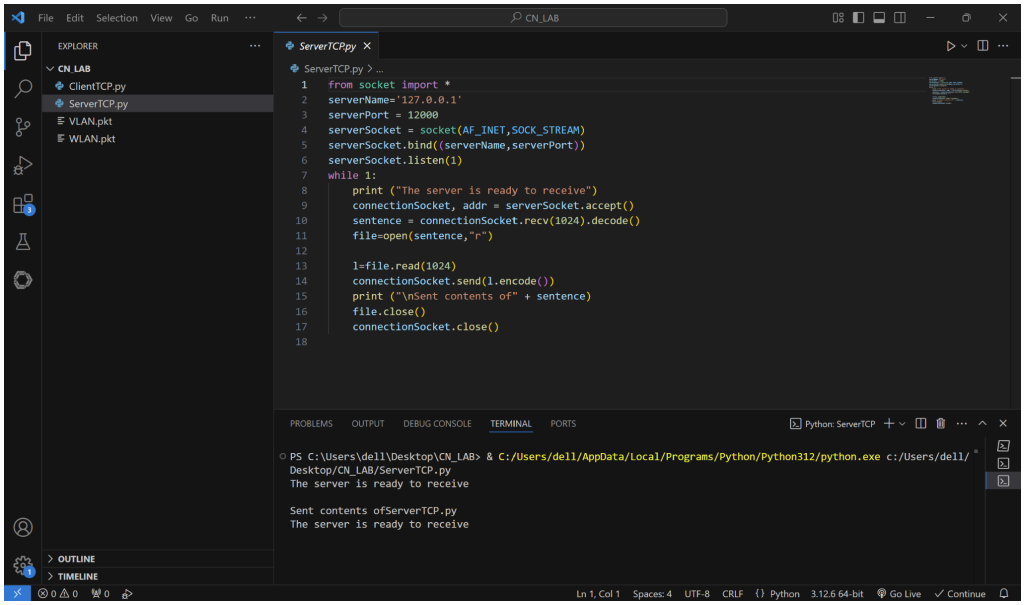
Code:
ClientTCP.py

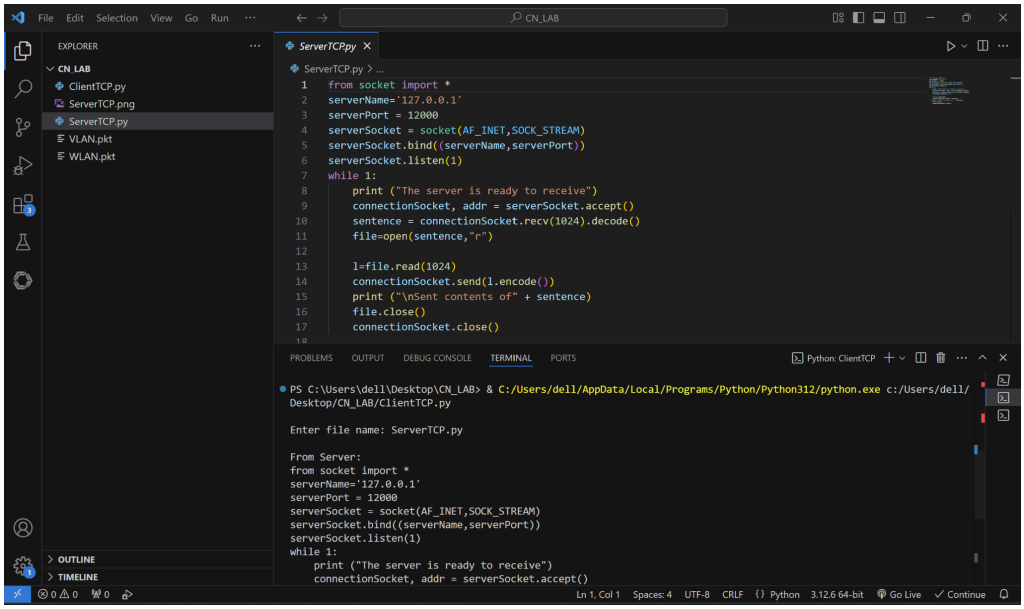
```
from socket import *

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:


File Edit Selection View Go Run
CN_LAB
EXPLORER
CN_LAB
ClientTCP.py
ServerTCP.py
VLAN.pkt
WLAN.pkt
ServerTCP.py
ServerTCP.py > ...
1 from socket import *
2 serverName='127.0.0.1'
3 serverPort = 12000
4 serverSocket = socket(AF_INET,SOCK_STREAM)
5 serverSocket.bind((serverName,serverPort))
6 serverSocket.listen(1)
7 while 1:
8 print ("The server is ready to receive")
9 connectionSocket, addr = serverSocket.accept()
10 sentence = connectionSocket.recv(1024).decode()
11 file=open(sentence,"r")
12
13 l=file.read(1024)
14 connectionSocket.send(1.encode())
15 print ("\nSent contents of" + sentence)
16 file.close()
17 connectionSocket.close()
18
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python: ServerTCP
PS C:\Users\dell\Desktop\CN_LAB> & C:/Users/dell/AppData/Local/Programs/Python/Python312/python.exe c:/Users/dell/Desktop/CN_LAB/ServerTCP.py
The server is ready to receive
Sent contents ofServerTCP.py
The server is ready to receive
OUTLINE
TIMELINE
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.6 64-bit Go Live Continue


File Edit Selection View Go Run
CN_LAB
EXPLORER
CN_LAB
ClientTCP.py
ServerTCP.png
ServerTCP.py
VLAN.pkt
WLAN.pkt
ServerTCP.py
ServerTCP.py > ...
1 from socket import *
2 serverName='127.0.0.1'
3 serverPort = 12000
4 serverSocket = socket(AF_INET,SOCK_STREAM)
5 serverSocket.bind((serverName,serverPort))
6 serverSocket.listen(1)
7 while 1:
8 print ("The server is ready to receive")
9 connectionSocket, addr = serverSocket.accept()
10 sentence = connectionSocket.recv(1024).decode()
11 file=open(sentence,"r")
12
13 l=file.read(1024)
14 connectionSocket.send(1.encode())
15 print ("\nSent contents of" + sentence)
16 file.close()
17 connectionSocket.close()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python: ClientTCP
PS C:\Users\dell\Desktop\CN_LAB> & C:/Users/dell/AppData/Local/Programs/Python/Python312/python.exe c:/Users/dell/Desktop/CN_LAB/ClientTCP.py
Enter file name: ServerTCP.py
From Server:
from socket import *
serverName='127.0.0.1'
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
print ("The server is ready to receive")
connectionSocket, addr = serverSocket.accept()
OUTLINE
TIMELINE
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.6 64-bit Go Live Continue

Experiment 15

Aim:
Using TCP/IP Sockets, write a client server program to make client sending the file name and the server to send back the contents of the requested file if present

Solution:
Client TCP.py

```
from socket import *
servername = '127.0.0.1'
serverport = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((servername, serverport))
sentence = input("Enter file name")

clientSocket.send(sentence.encode())
file contents = clientSocket.recv(1024).decode()
print('From server')
print(file contents)
clientSocket.close()
```

serverTCP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_NET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
while(1):
    print("The server is ready to recieve")
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024).decode()

file = open(sentence, "r")
file = open(sentence, "r")
    l = file.read(1024)
connectionSocket.send(l.encode())
    print('Sent contents of' + sentence)
    file.close()
connectionSocket.close()
```

Output:

The server is ready to recieve
Enter file name : serverTCP.py
Sent contents of serverTCP.py

Program 4

Aim of the program:
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.

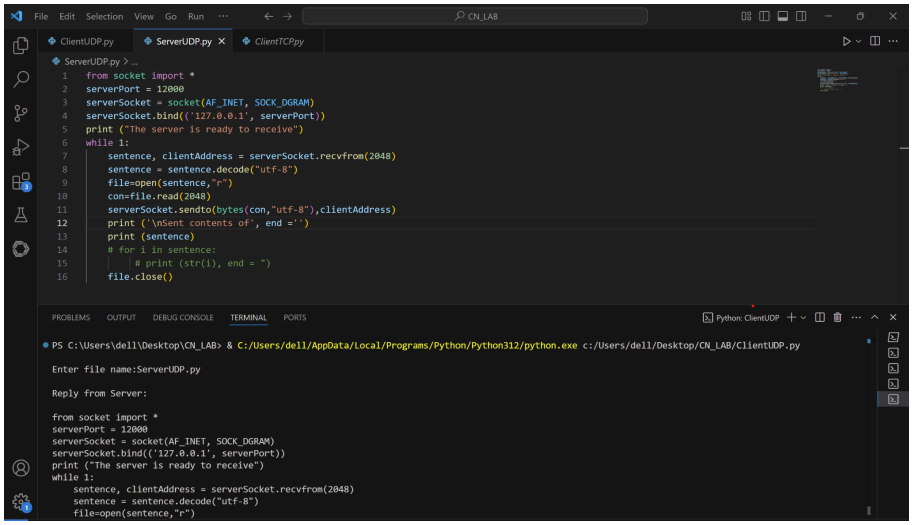
Code:
ClientUDP.py

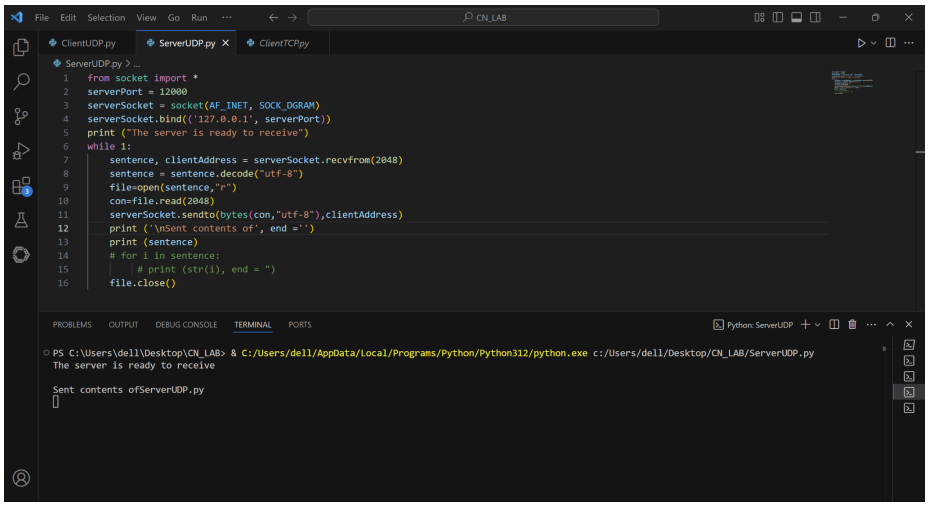
```
from socket import *

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"))
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)

    file.close()
```

OUTPUT:


File Edit Selection View Go Run
CN_LAB
ClientUDP.py
ServerUDP.py
ClientTCP.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()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python: ClientUDP
PS C:\Users\dell\Desktop\CN_LAB> & C:/Users/dell/AppData/Local/Programs/Python/Python312/python.exe c:/Users/dell/Desktop/CN_LAB/ClientUDP.py
Enter file name:ServerUDP.py
Reply from Server:
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('127.0.0.1', serverPort))
print ("The server is ready to receive")
while 1:
sentence, clientAddress = serverSocket.recvfrom(2048)
sentence = sentence.decode("utf-8")
file=open(sentence,"r")


File Edit Selection View Go Run
CN_LAB
ClientUDP.py
ServerUDP.py
ClientTCP.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()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python: ServerUDP
PS C:\Users\dell\Desktop\CN_LAB> & C:/Users/dell/AppData/Local/Programs/Python/Python312/python.exe c:/Users/dell/Desktop/CN_LAB/ServerUDP.py
The server is ready to receive
Sent contents ofServerUDP.py

Experiment 16

Aim :
Using UDP sockets, write a client server program to make client sending the file name and the server to send back the contents of the required file if present.

Solution :
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:
        # print(str(i), end ='')
clientSocket.close()
clientSocket.close()
```

ServerUDP.py

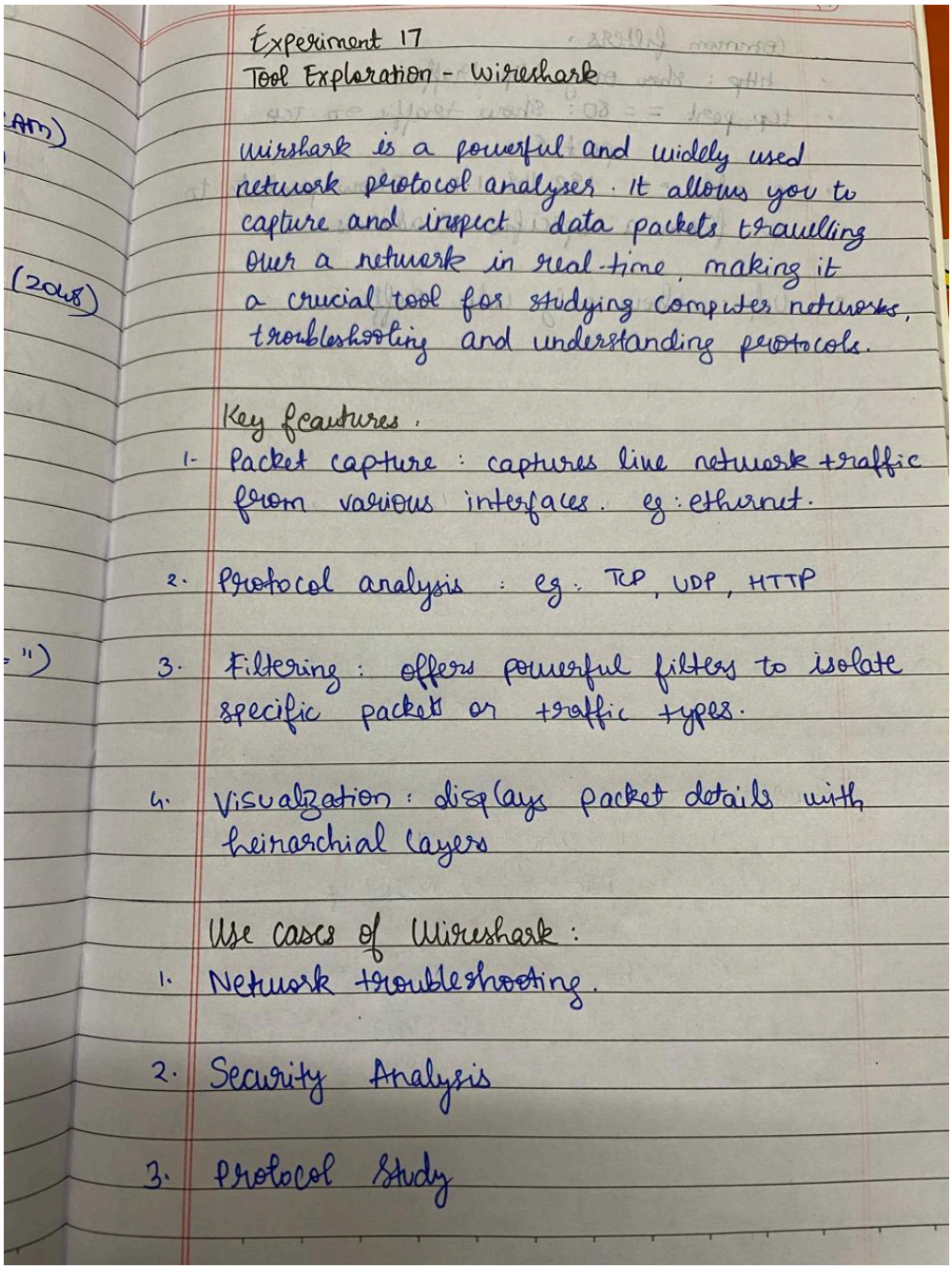
```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print("The server is ready to recieve")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file = open(sentence,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

serverSocket.sendto(bytes(con, "utf-8"),
                    clientAddress)
print('Sent contents of', end = ' ')
print(sentence)
# for i in sentence:  # print(str(i), end="")
    file.close()
```

Output:

```
The server is ready to recieve
Enter file name : ServerUDP.py

Sent contents of ServerUDP.py
The server is ready to recieve.
```

Program 5

Aim of the program:
Tool exploration- Wireshark



Experiment 17
Tool Exploration - Wireshark

Wireshark is a powerful and widely used network protocol analyser. It allows you to capture and inspect data packets travelling over a network in real-time, making it a crucial tool for studying computer networks, troubleshooting and understanding protocols.

Key features.

1- Packet capture : captures live network traffic from various interfaces. eg: ethernet.

2. Protocol analysis : eg: TCP, UDP, HTTP

3. Filtering : offers powerful filters to isolate specific packets or traffic types.

4. Visualization : displays packet details with heirarchial layers

Use cases of Wireshark :

1. Network troubleshooting.

2. Security Analysis

3. Protocol Study

Common filters:

- http : show only http traffic
- tcp.port == 80 : show traffic on TCP port 80.
- ip.addr == 192.168.1.1 : show packets to or from a specific IP address.
- udp : show only udp traffic.