VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

Sanath S Shetty(1BM23CS297)

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

Academic Year 2024-25 (odd)

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 Network (23CS5PCCON)" carried out by **Sanath S Shetty(1BM23CS297),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements of the above-mentioned subject and the work prescribed for the said degree.

Leelavathi.B Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index

		CYCLE-2	
1	29/10/25	Write a program for error detecting code using CRC-CCITT (8-bits).	82-91
2	29/10/25	Write a program for congestion control using Leaky bucket algorithm.	92-100
3	12/11/25	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.	101-112
4	12/11/25	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	113-124

Github Link:
https://github.com/Sanath-S-Shetty/Sanath_CN_LAB

CYCLE – I

Program 1:

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

Aim:

To demonstrate the transmission of a simple PDU between 2 devices connected using a hub and a switch.

Observation

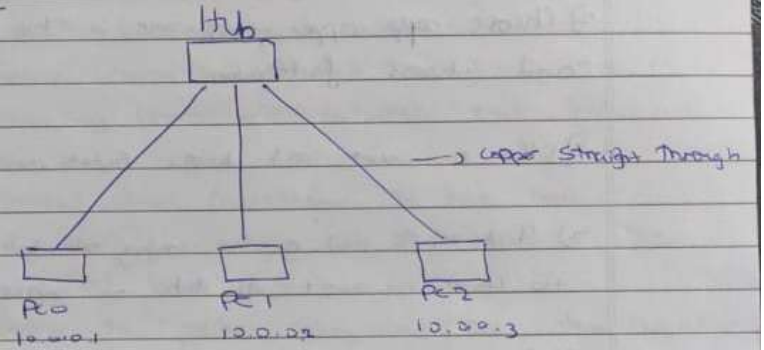
PAGE NO :
DATE :

Week - 1

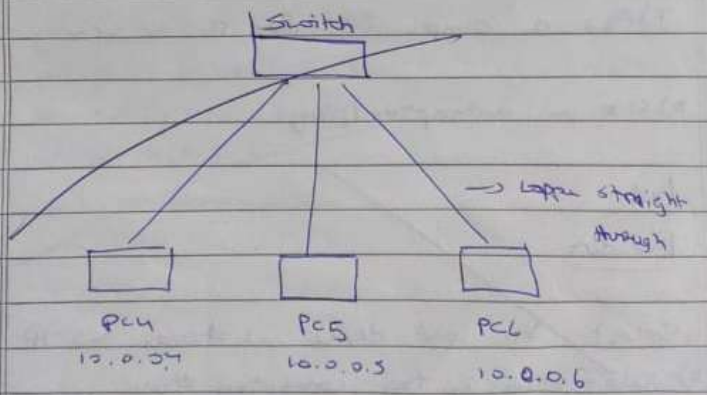
Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting device and demonstrate ping message.

Topology

Hub



Switch



PAGE NO :
DATE :

Aim : To understand simple PC-PDU configuration

Procedure

1) Select PC from end device, select generic PC and drop it in workspace, similarly select generic server

2) Choose copper-cross-over in the connections and choose fastethernet

3) Click on server and choose FastEthernet

4) Click on PC and go to config tab. Set IP address to 10.0.0.1 and click on subnet Mask

5) Repeat same step and set IP address for server

6) In simulation mode, to edit filters click only IPV

7) Add a simple PDU from PC to server

8) Click on autocapture/play.

Hub

Procedure

1) Select the end device and change the IP address
2) Select hub as the connecting device
3) Select copper-straight-through as the connecting wire between end device and hub

PAGE NO :
DATE :

4) Connect the fastethernet to the hub port.
5) Select the message and first click on source device and destination device
6) Observe the packet transmission and acknowledgement receiving procedure

Switch

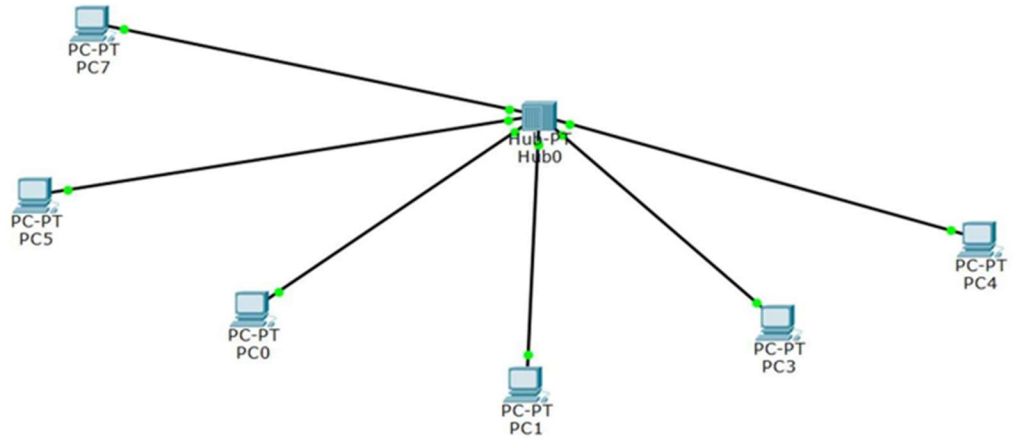
1) Select the end device and change their IP
2) Select Switch as the connecting device
3) Select copper straight through as the connecting device wire between end device and hub
4) Connect the fastethernet to hub port.
5) Select the message and first click on source device and then destination device
6) Observe the packet transmission and acknowledgement receiving procedure

Commonly used

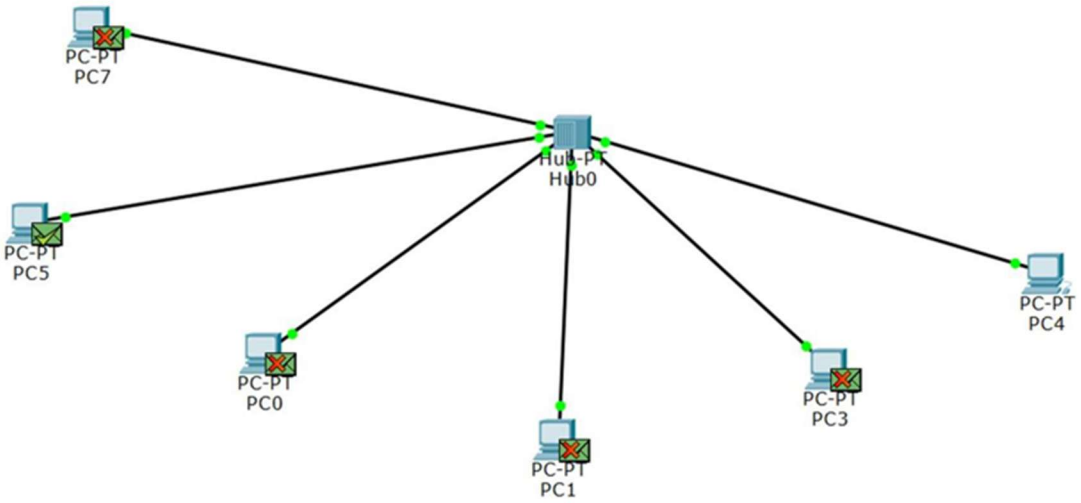
enable
config terminal
interface gig 0/0
ip address 10.0.0.254 255.0.0.0
no shutdown
exit

Hub

Topology

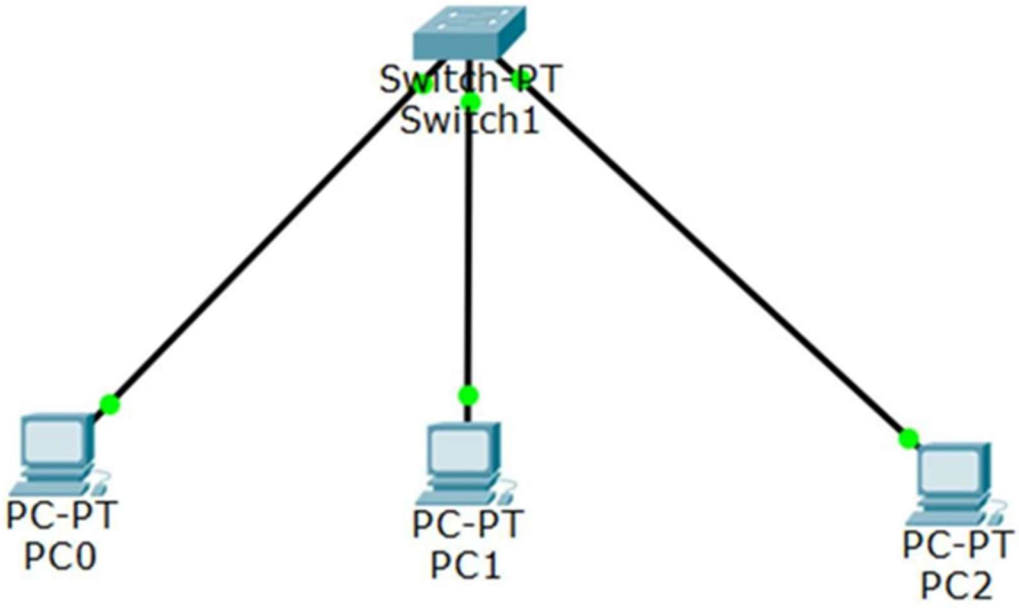


Output

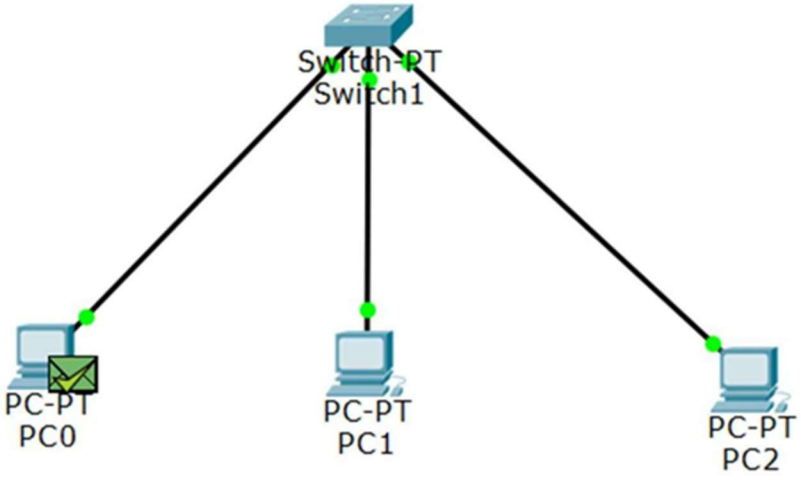


Switch

Topology



Output



Program 2:
Configure default route, static route to the Router

Aim: To configure default and static routers to a connection of routers.

Observation

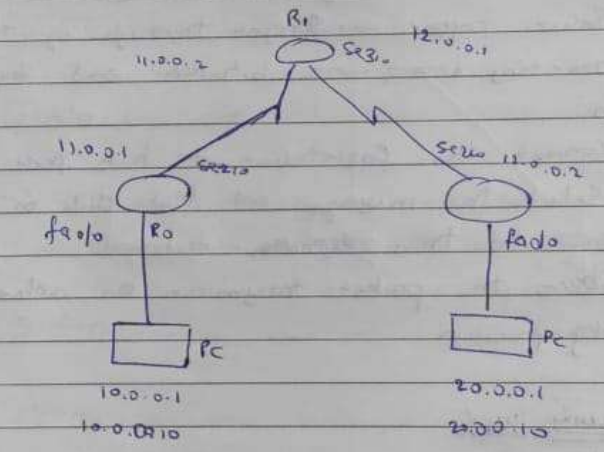
PAGE NO :
DATE :

Week 2

Packet Routing using Routers in the connection data

Aim :-

Configure default Route from a static route Through the Router.



step 1 : First we we take 2 Pc's and config the IP address of both the Pc's

Pc-0 IP addres - 10.0.0.1

default gateway - 10.0.0.10

Config IP for Pc1

Pc-1 IP address - 20.0.0.1

default gateway - 20.0.0.10

Now we use the cross cover cable and and fast ethernet

PAGE NO :
DATE :

to connect to routers from both the PC's
So in router 0, The ~~Cut~~ CLI commands are:

Go to CLI
enable
Config terminal
interface fa0/0
ip address 10.0.0.10 255.0.0.0
no shutdown → important
exit

interface s2/0
ip address 21.0.0.1 255.0.0.0
no shutdown
exit

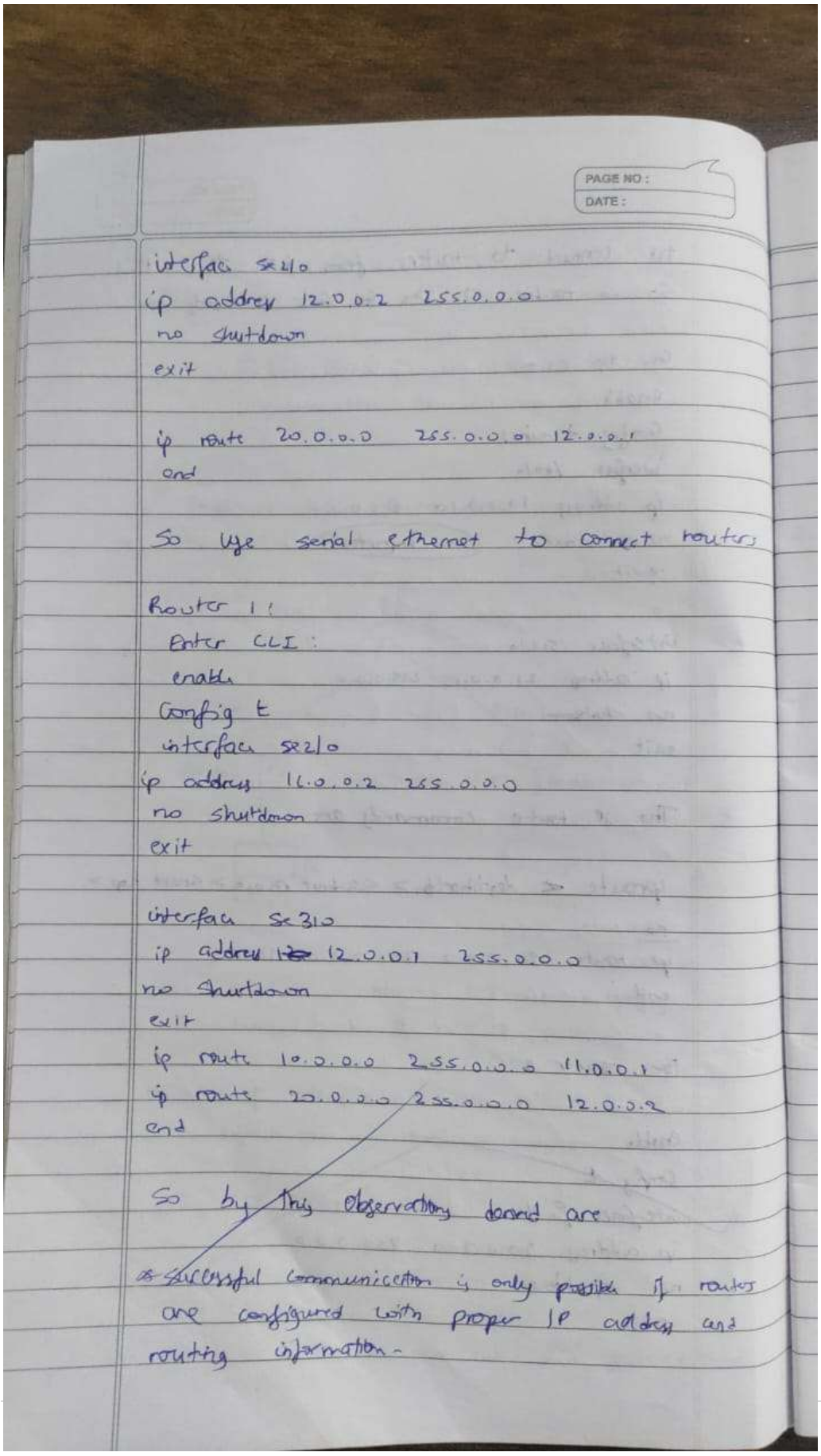
The ip route commands are

ip route <destination> <subnet mask> <next hop>

Ex:
ip route 20.0.0.0 255.0.0.0 11.0D.2
end

For router 2:

Enable
Config t
interface fa 0/0
ip address 20.0.0.10 255.0.0.0
no shutdown
exit



PAGE NO :
DATE :

interface se2/0
ip address 12.0.0.2 255.0.0.0
no shutdown
exit

ip route 20.0.0.0 255.0.0.0 12.0.0.1
end

So use serial ethernet to connect routers

Router 1 :
Enter CLI :
enable
Config t
interface se2/0
ip address 11.0.0.2 255.0.0.0
no shutdown
exit

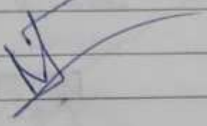
interface se3/0
ip address ~~12~~ 12.0.0.1 255.0.0.0
no shutdown
exit
ip route 10.0.0.0 255.0.0.0 11.0.0.1
ip route 20.0.0.0 255.0.0.0 12.0.0.2
end

So by this observation denied are

successful communication is only possible if routers are configured with proper IP address and routing information.

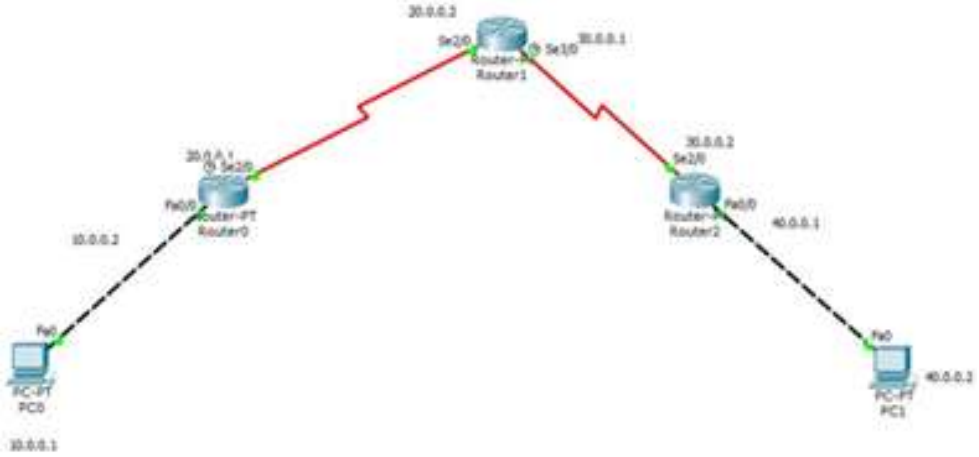
PAGE NO :
DATE :

* without routing, packets to other networks fail.
* static and default routes enable successful communication across networks.

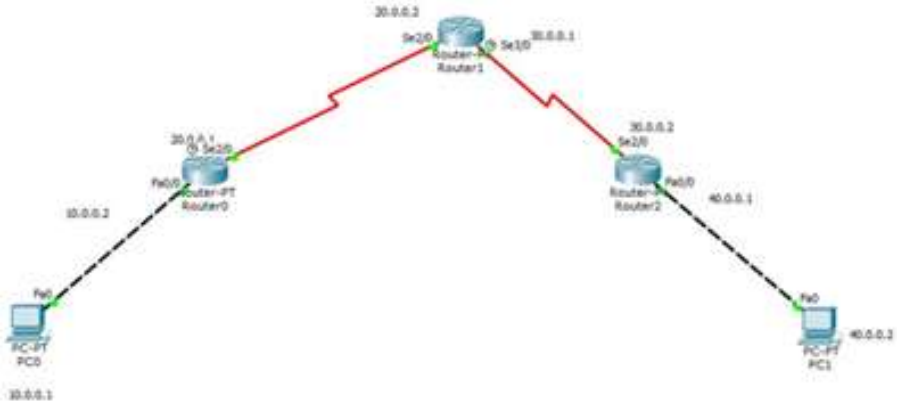


Topology

Default route:



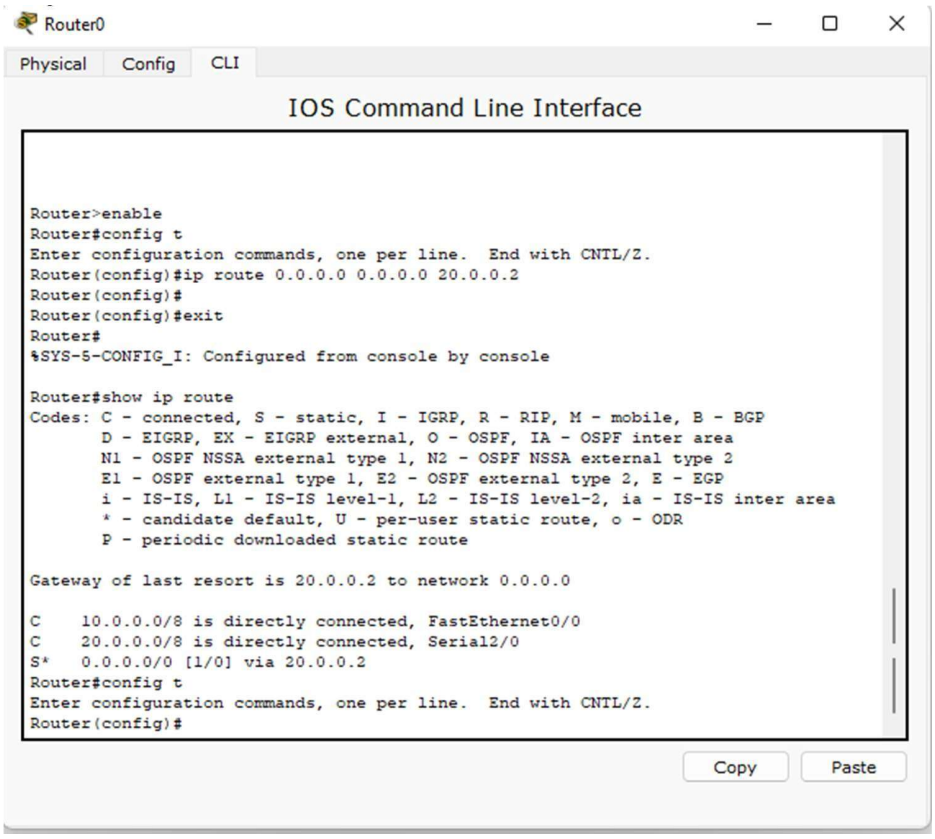
Static route:



Output

```
Router(config)#ip route 10.0.0.0 255.0.0.0 20.0.0.1
Router(config)#ip route 40.0.0.0 250.0.0.0 30.0.0.2
%Inconsistent address and mask
Router(config)#ip route 40.0.0.0 255.0.0.0 30.0.0.2
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
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

S    10.0.0.0/8 [1/0] via 20.0.0.1
C    20.0.0.0/8 is directly connected, Serial2/0
C    30.0.0.0/8 is directly connected, Serial3/0
S    40.0.0.0/8 [1/0] via 30.0.0.2
Router(config)#ip route 30.0.0.0 255.0.0.0 20.0.0.2
Router(config)#ip route 40.0.0.0 255.0.0.0 20.0.0.2
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
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
C    20.0.0.0/8 is directly connected, Serial2/0
S    30.0.0.0/8 [1/0] via 20.0.0.2
S    40.0.0.0/8 [1/0] via 20.0.0.2
```

```
Router(config)#ip route 20.0.0.0 255.0.0.0 30.0.0.1
Router(config)#ip route 10.0.0.0 255.0.0.0 30.0.0.1
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
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

S    10.0.0.0/8 [1/0] via 30.0.0.1
S    20.0.0.0/8 [1/0] via 30.0.0.1
C    30.0.0.0/8 is directly connected, Serial2/0
C    40.0.0.0/8 is directly connected, FastEthernet0/0
```



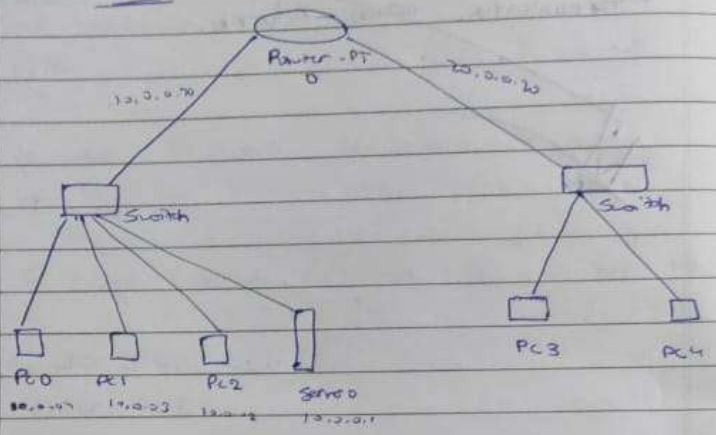

```
PC>ping 40.0.0.2

Pinging 40.0.0.2 with 32 bytes of data:

Reply from 40.0.0.2: bytes=32 time=7ms TTL=125
Reply from 40.0.0.2: bytes=32 time=7ms TTL=125
Reply from 40.0.0.2: bytes=32 time=11ms TTL=125
Reply from 40.0.0.2: bytes=32 time=9ms 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 = 7ms, Maximum = 11ms, Average = 8ms
```

Program 3:
Configure DHCP within a LAN and outside LAN.

Aim:
To configure DHCP within a LAN and outside LAN.

PAGE NO :
DATE :

Week 3 - Configure DHCP within a LAN and outside LAN



Procedure:

1) Select two or more PC and a server connecting to switch and another network with only end device and switch

2) Connect both switches a to router.

3) Set IP address of server as 10.0.0.1

4) Now, go to service <select DHCP> save the current IP address 20.0.0.2

5) Now, check the IP address of other device in the network in the IP configuration is dynamic

6) Now in the CLI of router enable follow steps
enable
config t
interface f0/0

PAGE NO :
DATE :

ip address 10.0.0.10 255.0.0.0
no shutdown
exit
interface fa 0/0
ip address 20.0.0.20 255.0.0.0
no shut
exit

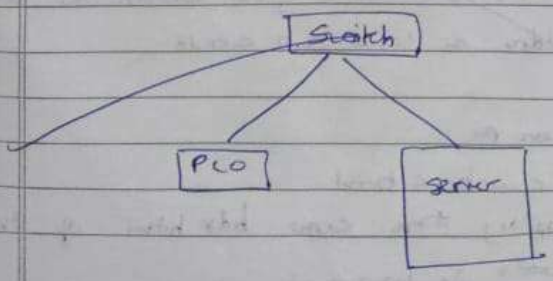
7) Go to server config < gateway 10.0.0.20

8) Now in router, we need to set IP address of server.
config t
fast ethernet 0/0
ip helper-address 10.0.0.1
no shut
exit

9) Now go to server < services < DHCP < add new IP address 20.0.0.2

10) To check the connection go to the IP config window of PC outside the network and click on DHCP and the IP gateway will be visible.

Topology



PAGE NO :
DATE :

Procedure

1) The topology consists of a PC, server and switch

2) Connect them using copper-straight through wire

3) Set IP address of PC as 10.0.0.1 and that of server as 10.0.0.2

4) Click on pc
Desktop < web browser < 10.0.0.2
It displays the index.html page of the server

5) Click on server
Service < index.html < edit < change the subject to BMS < save

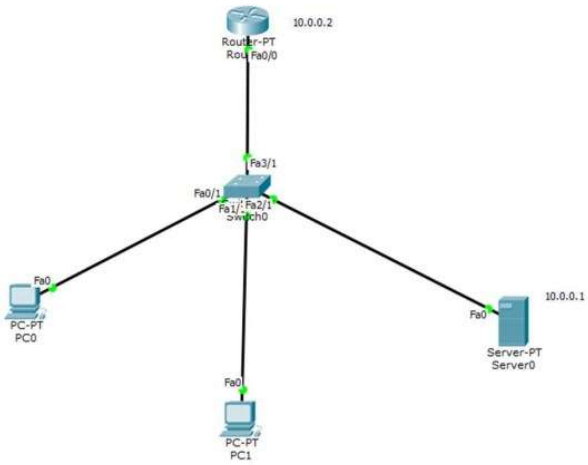
6) Click on PC
Desktop < web < 10.0.0.2
It changes to content of index.html page as set by us

7) Click on Server
Service < DNS < ON < give domain as bms < set address as 10.0.0.2 < add

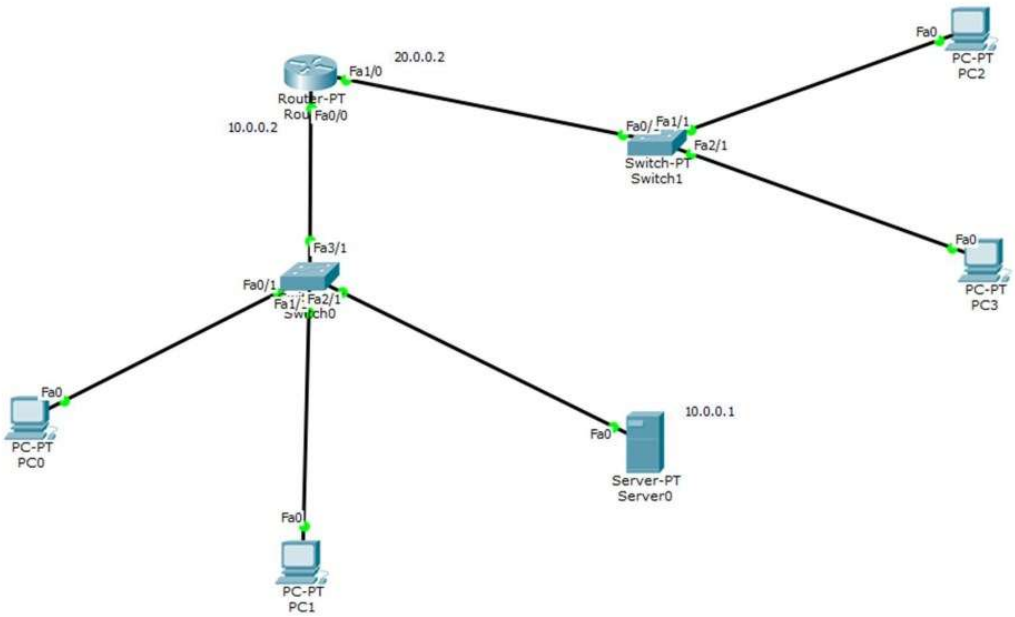
Click on PC
Desktop < web < bms
It display the same hdx.html of the server (with ip 10.0.0.2)

Topology

within a LAN

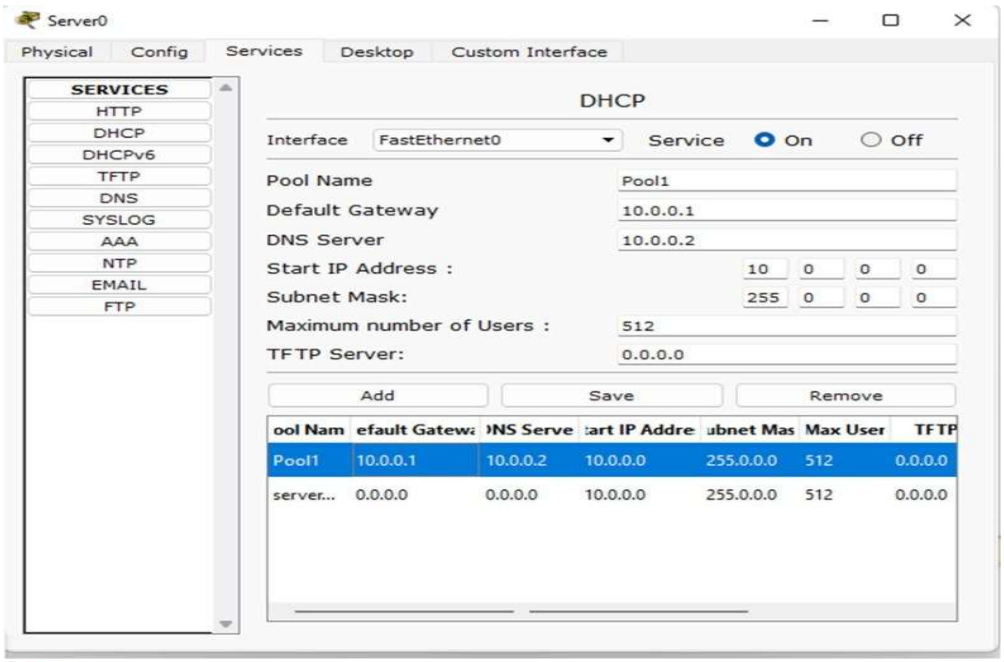


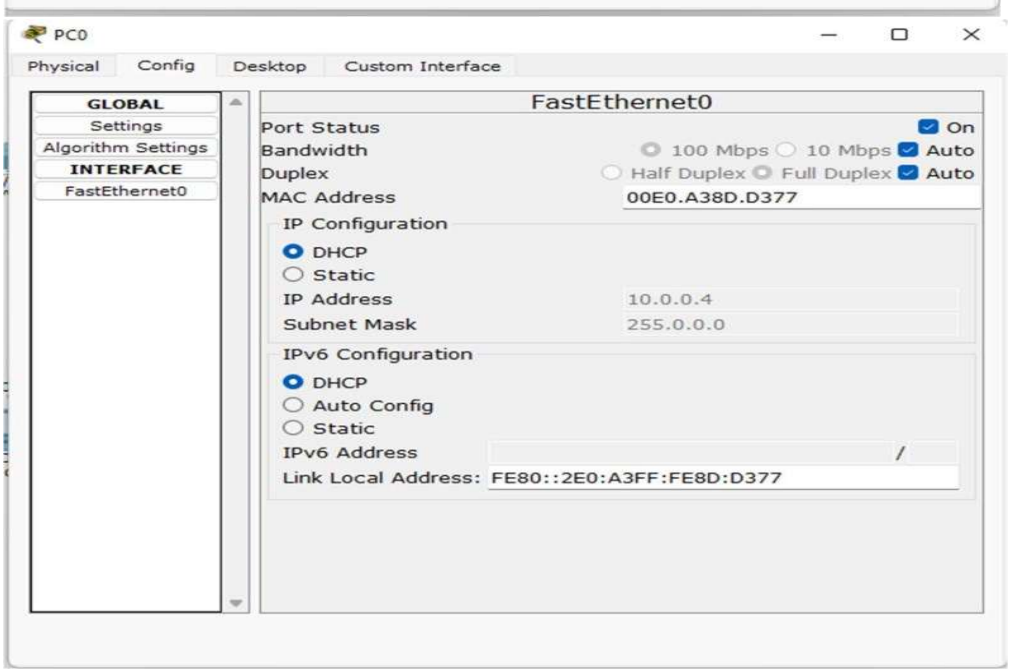
Outside LAN

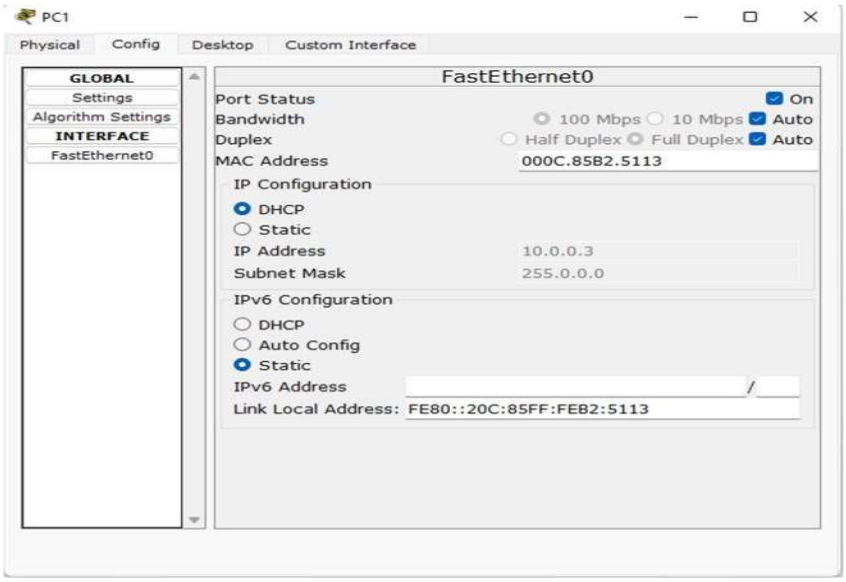


Output:

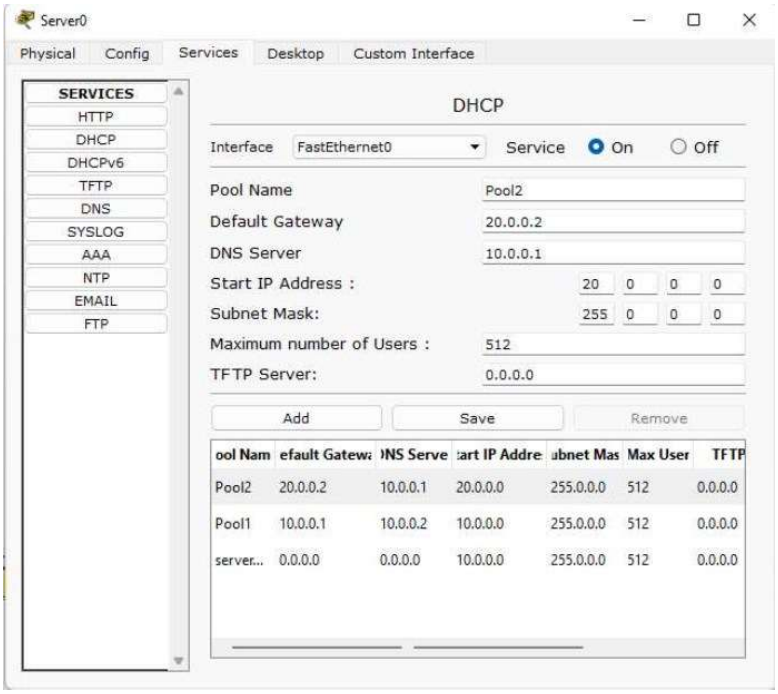
Within LAN

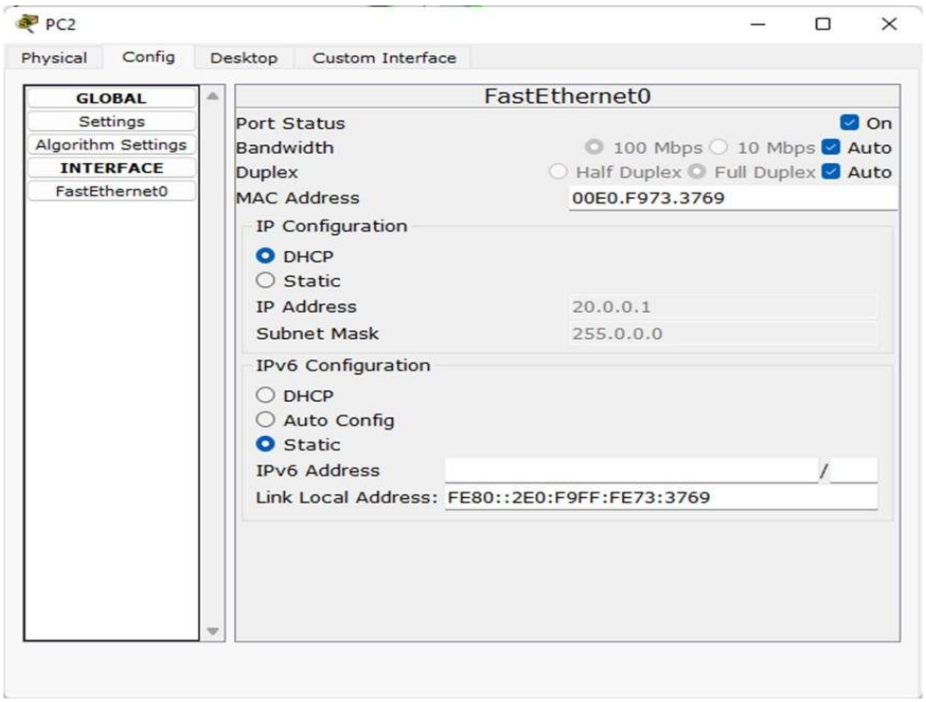


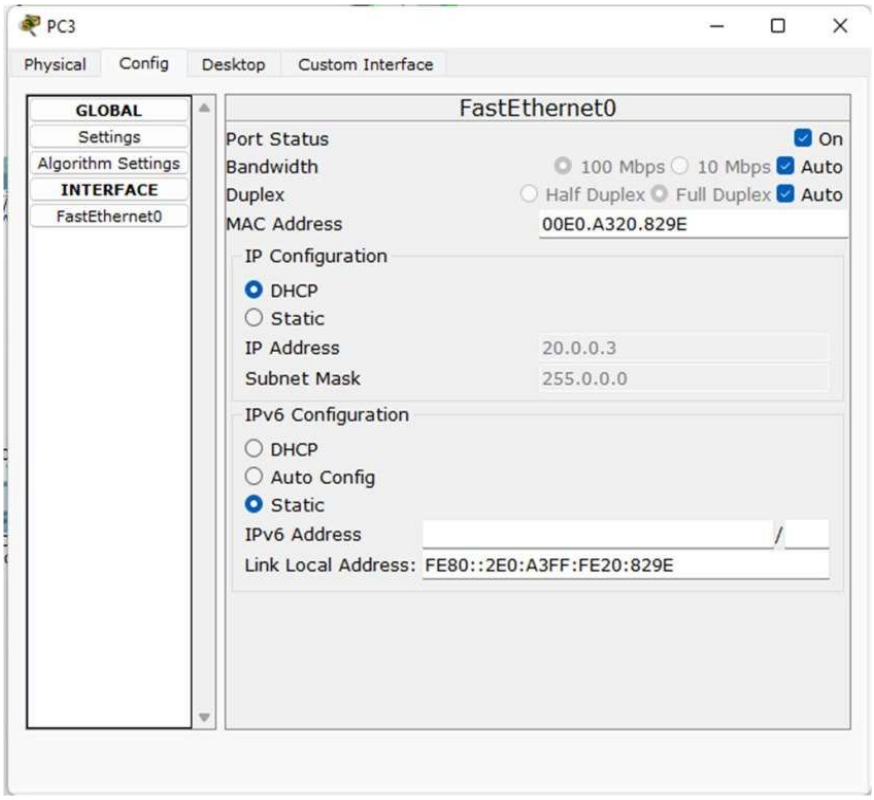



PC1
Physical
Config
Desktop
Custom Interface
GLOBAL
Settings
Algorithm Settings
INTERFACE
FastEthernet0
FastEthernet0
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
MAC Address
000C.85B2.5113
IP Configuration
DHCP
Static
IP Address
10.0.0.3
Subnet Mask
255.0.0.0
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
Link Local Address:
FE80::20C:85FF:FEB2:5113

Outside LAN


Server0
Physical
Config
Services
Desktop
Custom Interface
SERVICES
HTTP
DHCP
DHCPv6
TFTP
DNS
SYSLOG
AAA
NTP
EMAIL
FTP
DHCP
Interface
FastEthernet0
Service
On
Off
Pool Name
Pool2
Default Gateway
20.0.0.2
DNS Server
10.0.0.1
Start IP Address :
20 0 0 0
Subnet Mask:
255 0 0 0
Maximum number of Users :
512
TFTP Server:
0.0.0.0
Add
Save
Remove
ool Nam
efault Gatewa
NS Serve
art IP Addre
ubnet Mas
Max User
TFTP
Pool2 20.0.0.2 10.0.0.1 20.0.0.0 255.0.0.0 512 0.0.0.0
Pool1 10.0.0.1 10.0.0.2 10.0.0.0 255.0.0.0 512 0.0.0.0
server... 0.0.0.0 0.0.0.0 10.0.0.0 255.0.0.0 512 0.0.0.0


PC2
Physical
Config
Desktop
Custom Interface
GLOBAL
Settings
Algorithm Settings
INTERFACE
FastEthernet0
FastEthernet0
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
MAC Address
00E0.F973.3769
IP Configuration
DHCP
Static
IP Address
20.0.0.1
Subnet Mask
255.0.0.0
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
Link Local Address:
FE80::2E0:F9FF:FE73:3769


PC3
Physical
Config
Desktop
Custom Interface
GLOBAL
Settings
Algorithm Settings
INTERFACE
FastEthernet0
FastEthernet0
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
MAC Address
00E0.A320.829E
IP Configuration
DHCP
Static
IP Address
20.0.0.3
Subnet Mask
255.0.0.0
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
Link Local Address:
FE80::2E0:A3FF:FE20:829E

Program 4:
Configure Web Server, DNS within a LAN.

Aim: To configure DNS server to demonstrate the mapping of IP addresses and domain names.

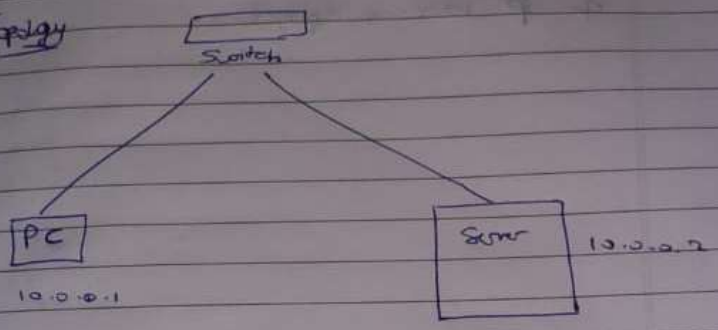
Observation:

PAGE NO :
DATE :

Program 4

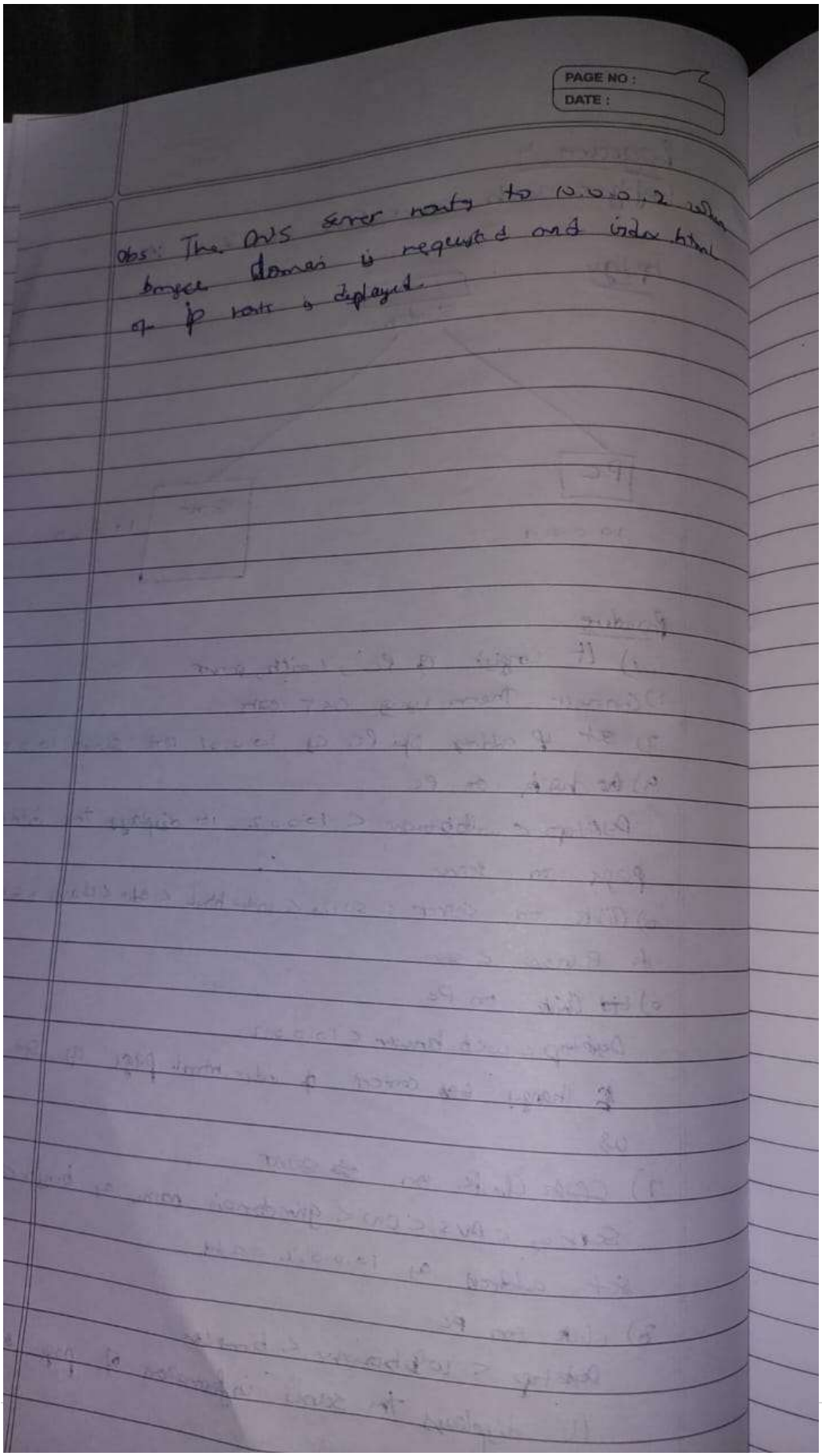
Configure web server, DNS within a LAN

Topology



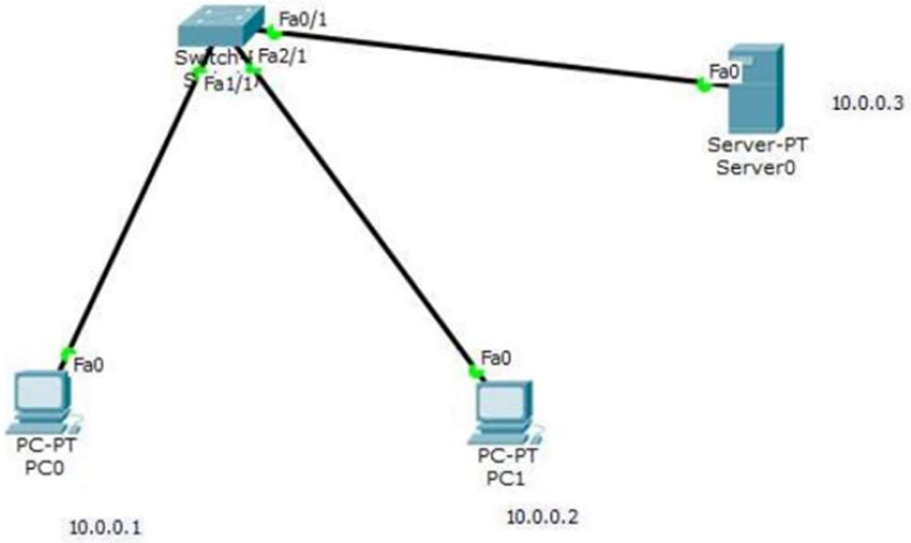
Procedure

1) It consist of PC, switch, server.
2) Connect them using CST cable
3) Set ip address of PC as 10.0.0.1 and Server 10.0.0.2
4) Go to check on PC
   Desktop < web browser < 10.0.0.2, it displays the index.html page on server
5) Click on server : Services < index.html < edit < change < submit to Browse < save
6) Click on Pc
   Desktop < web browser < 10.0.0.2
   It changes content of index.html page as set by us
7) Click on server.
   Services < DNS < ON < give domain name as browser URL set address as 10.0.0.2 < add
8) click on Pc
   Desktop < web browser < browse
   It displays the same information of page of the server

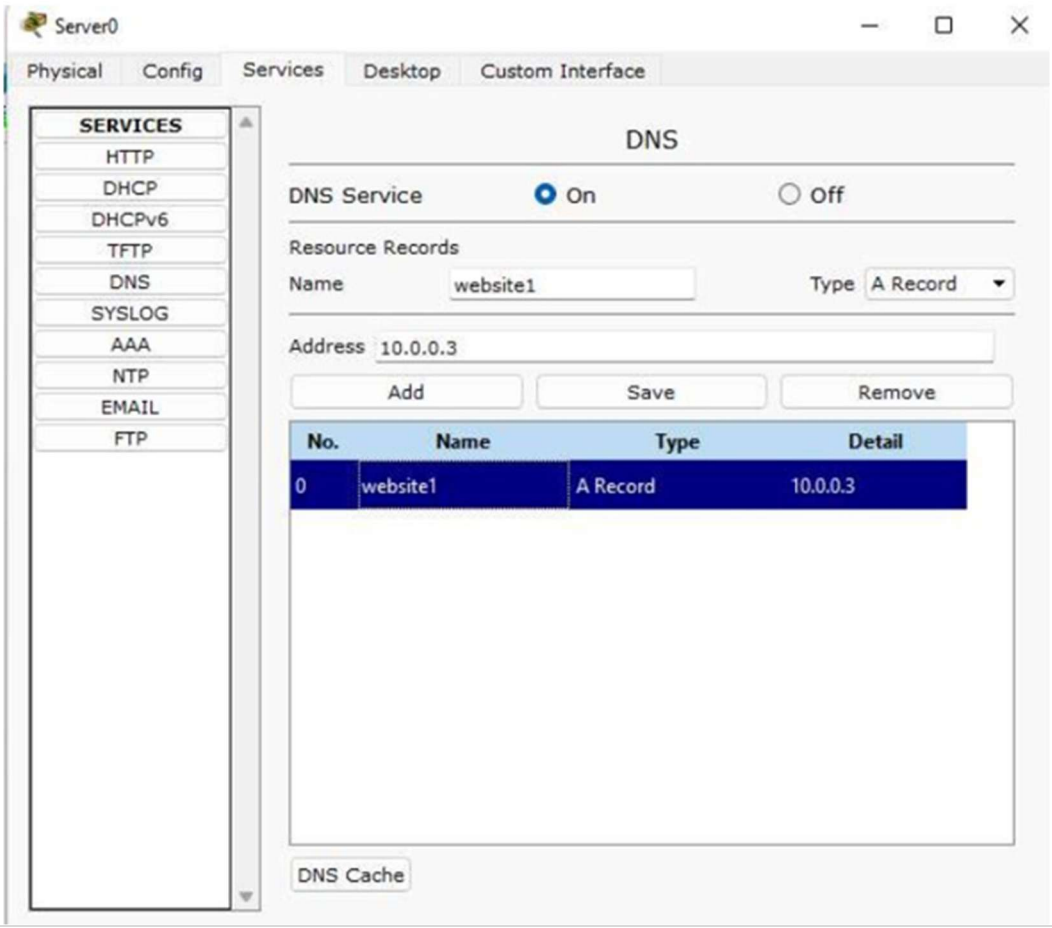

PAGE NO :
DATE :

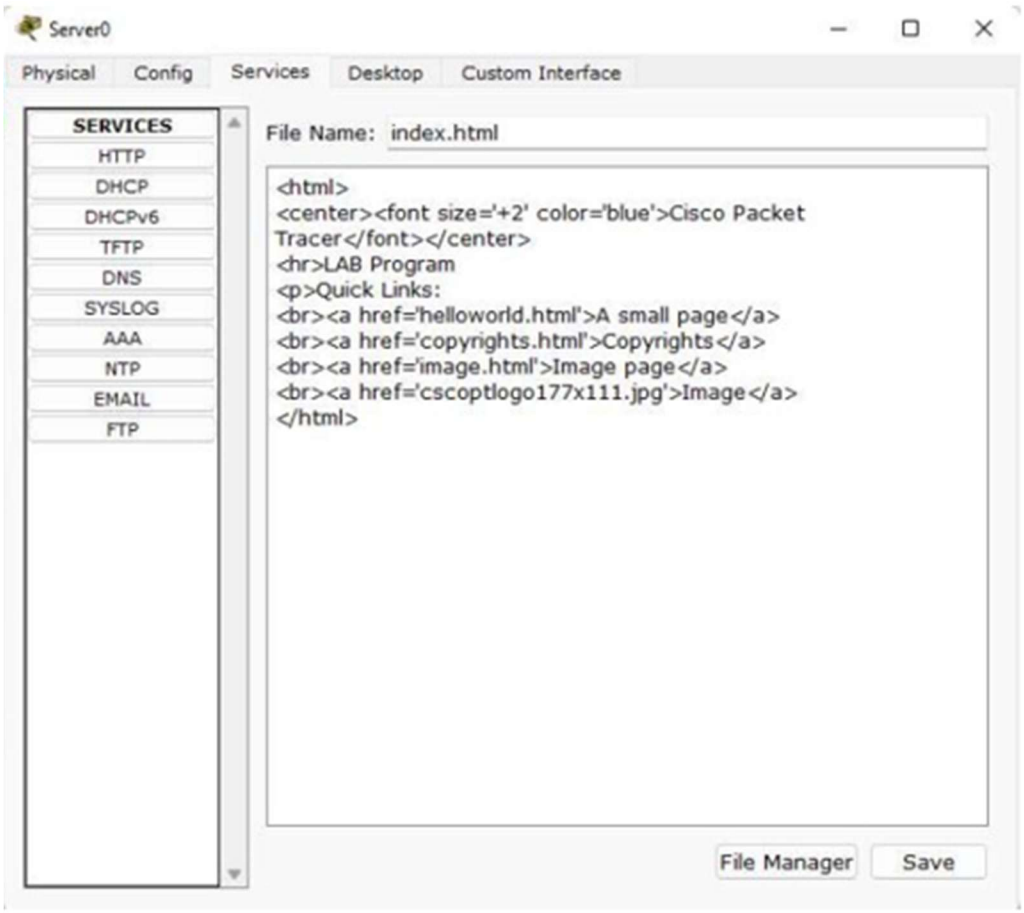
Obs: The DNS server routes to 10.0.0.2 when browse domain is requested and index.html of ip route is displayed.

Topology



Output



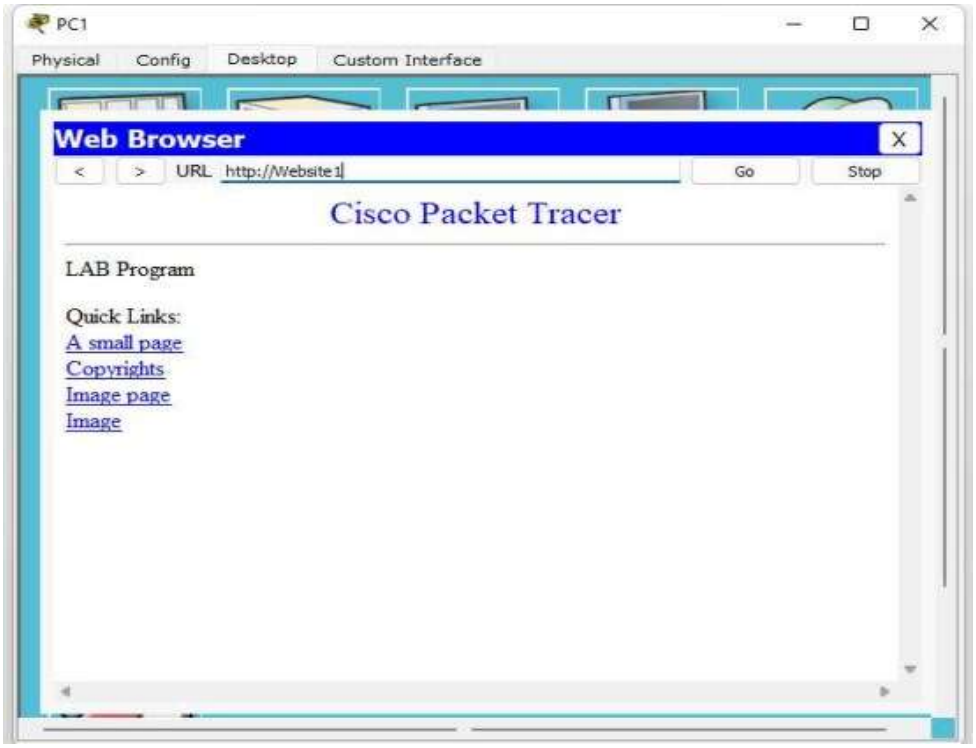

Server0
Physical
Config
Services
Desktop
Custom Interface
SERVICES
HTTP
DHCP
DHCPv6
TFTP
DNS
SYSLOG
AAA
NTP
EMAIL
FTP
File Name: index.html
<html>
Cisco Packet Tracer
<hr>LAB Program
<p>Quick Links:

A small page

Copyrights

Image page

Image
</html>
File Manager
Save


PC1
Physical
Config
Desktop
Custom Interface
Web Browser
X
<
>
URL
http://Website1
Go
Stop
Cisco Packet Tracer
LAB Program
Quick Links:
A small page
Copyrights
Image page
Image

Program 5:

To understand the operation of TELNET by accessing the router in server room from a PC in IT office.Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply

Aim:

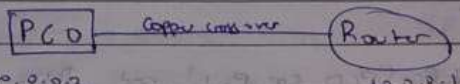
To use TELNET to remotely access a router from a PC and observe ping messages like reply, request timed out, and destination unreachable in Packet Tracer.

Observation

PAGE NO :
DATE :

Week - 4

Topology



Procedure

1. Configure topology as above, use copper cross over wire to connect both configure IP address and gateway and the router generally

2. In Router CLI

Router > enable

Router (config) # hostname r1

r1 (config) # enable secret p1

r1 (config) # interface fastethernet 0/0

r1 (config) # ip address 10.0.0.1 255.0.0.0

r1 (config-if) # line vty 0 4

r1 (config-line) # login password p2

r1 (config-line) # login p2

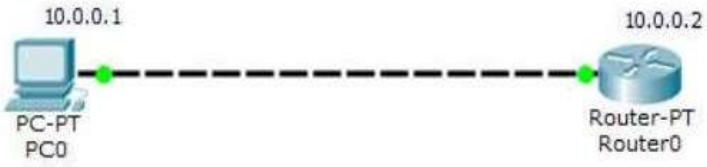
r1 (config-line) # exit

r1 # wr

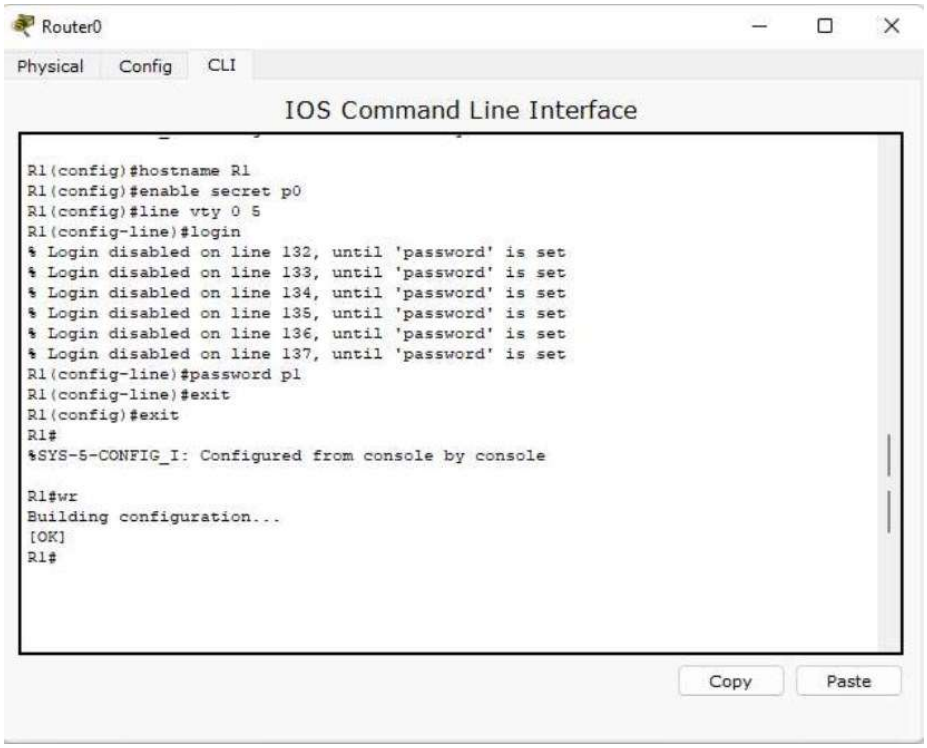
Building configuration

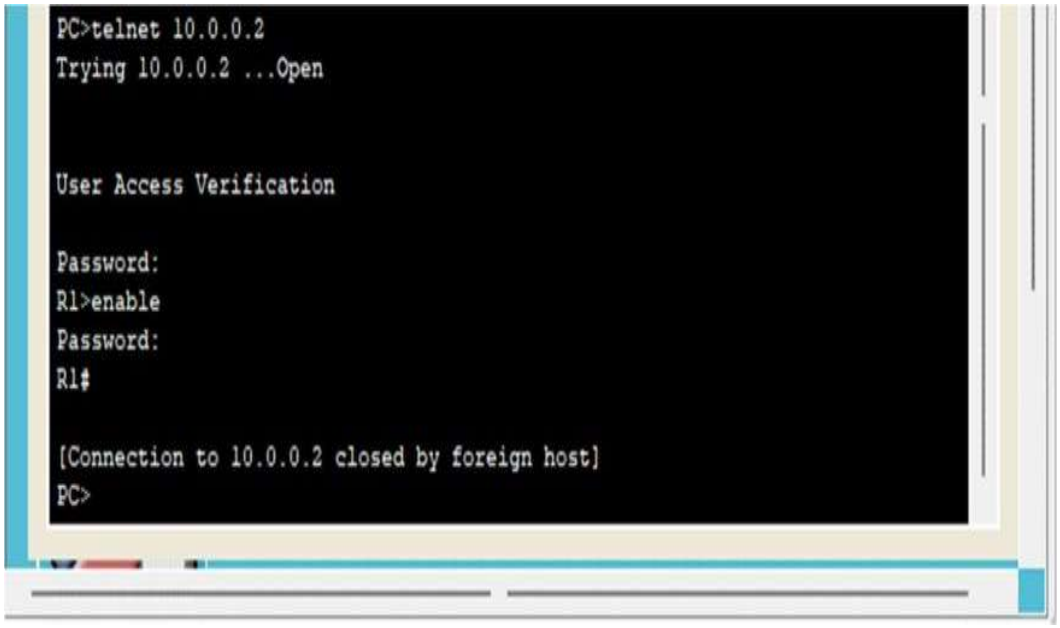
8/10

Topology



Output




PC>telnet 10.0.0.2
Trying 10.0.0.2 ...Open
User Access Verification
Password:
R1>enable
Password:
R1#
[Connection to 10.0.0.2 closed by foreign host]
PC>

Program 6:

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

Aim:

To create VLANs in a network and enable communication between PCs within the same VLAN**.**

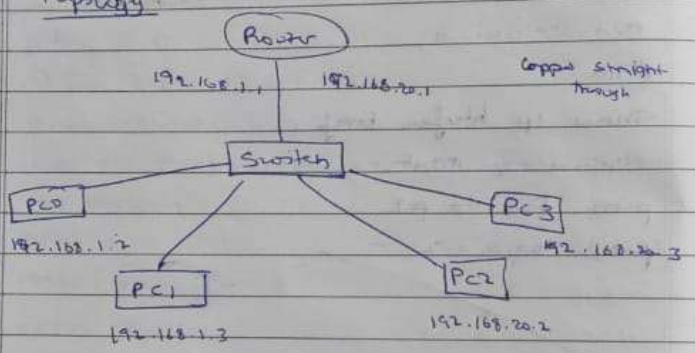
Observation

PAGE NO :
DATE :

Week 6

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

Topology :



Procedure

1) Set up the topology as shown above, use 1891 router
2) Add an extra router port to the switch as it needed
3) Use copper straight through wire set the IP address and gateway
4) In switch → config → VLAN Database, give any VLAN number, here 20, and VLAN name here → VLAN
5) Select add → select the interface (here - fast ethernet 4/1), (connect to the switch from router) and make it trunk
6) Look into fast ethernet 2/1 and 3/1 and change VLAN1 to 20: VLAN

PAGE NO :
DATE :

7) Router, select VLAN DATABASE, enter the number and name of the VLAN created

In CLI of router

Router (VLAN) # exit
Apply completed
Exiting
Router # config t
Router (config) # interface fastethernet do
Router (config-if) # ip address 192.168.1.1
Router (config-if) # no shutdown 255.255.255.0

Router (config) # interface fastethernet 0/0.1
Router (config-subif) # encapsulation dot1q 20
Router (config-subif) # ip address 192.168.20.1
255.255.255.0

Router (config sub-if) #no shutdown
Router (config-subif) # exit

Result:

(In PC0)

PC > ping 192.168.20.3

pinging 192.168.20.3 with 32 bytes of data

Reply from 192.168.20.3: bytes 32 time=1ms TTL=128
Reply from 192.168.20.3: bytes 32 time=1ms TTL=128
Reply from 192.168.20.3: bytes 32 time=0ms TTL=128
Reply from 192.168.20.3: bytes 32 time=0ms TTL=128

Ping: statistics for 192.168.20.3
Packets: Sent = 4, Received = 4, Lost = 0

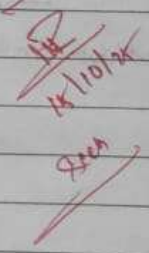
PAGE NO :
DATE :

Approx round trip time in ms
min = 0 ms max = 1 ms, avg = 0 ms

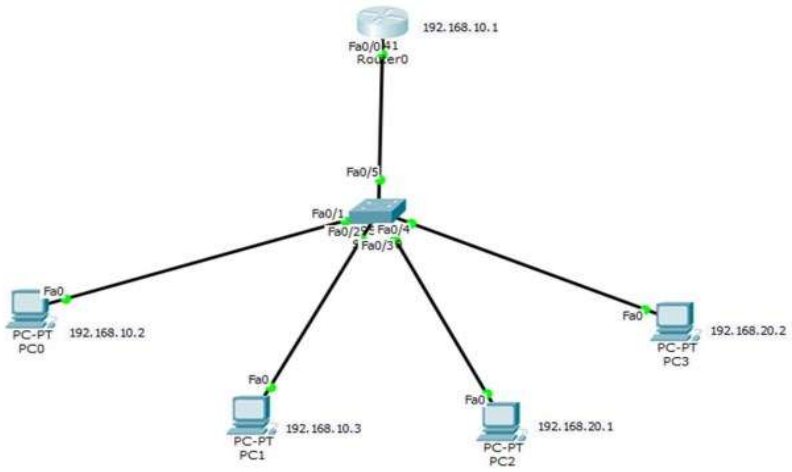
Observation

1) VLAN - Virtual local area network is any broadcast domain that is partitioned and isolated in a computer network at the data link layer

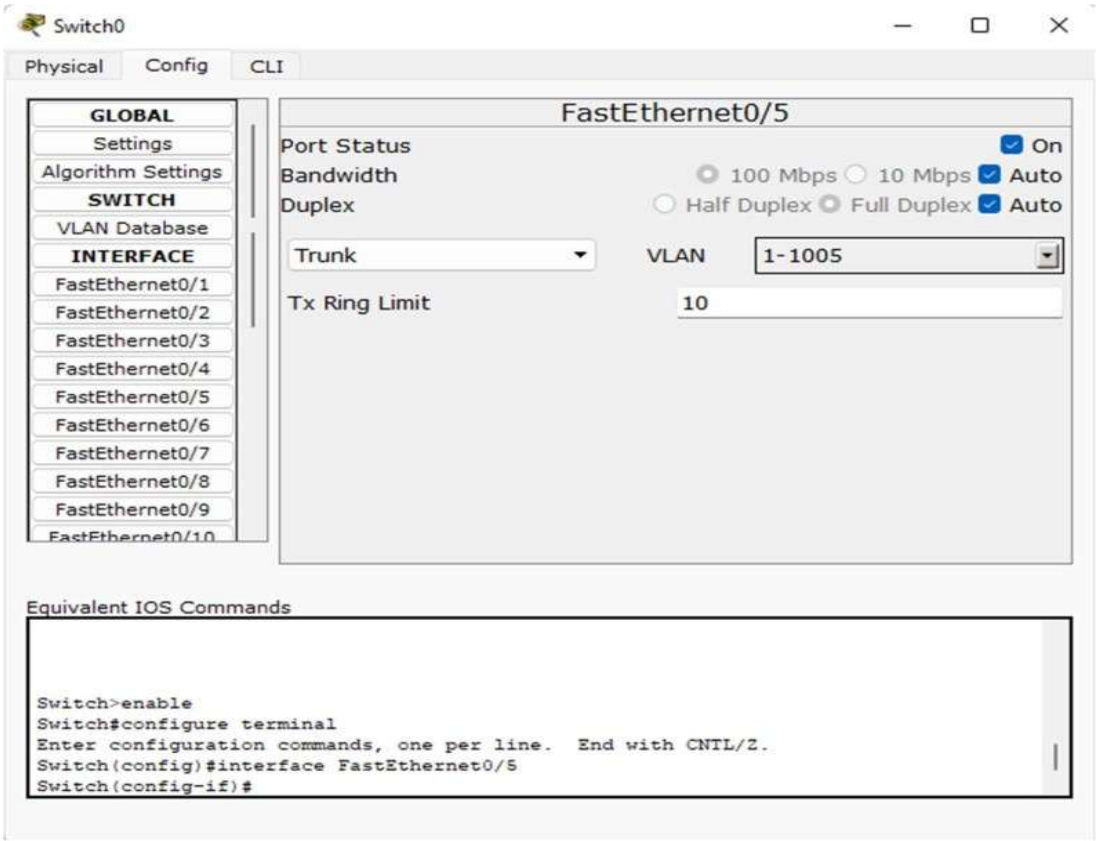
2) It is a virtualised connection that connects multiple devices and network nodes from different LAN's into one ~~locality~~ local (logical) network

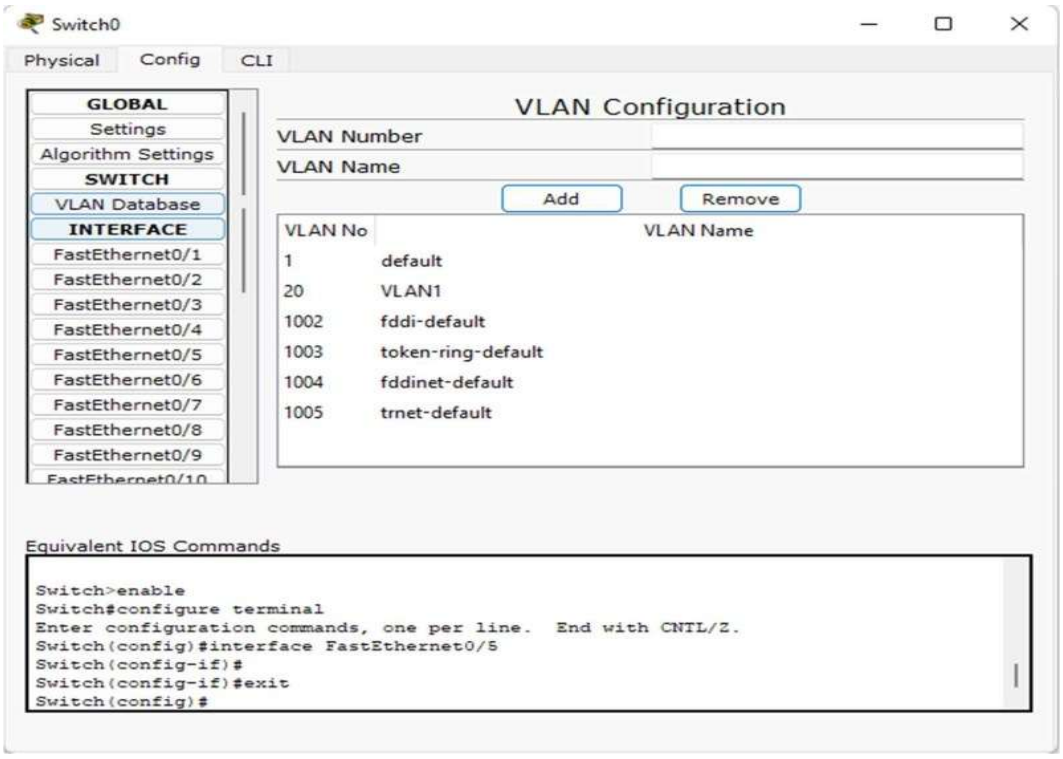


Topology



Output




Switch0
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
SWITCH
VLAN Database
INTERFACE
FastEthernet0/1
FastEthernet0/2
FastEthernet0/3
FastEthernet0/4
FastEthernet0/5
FastEthernet0/6
FastEthernet0/7
FastEthernet0/8
FastEthernet0/9
VLAN Configuration
VLAN Number
VLAN Name
Add
Remove
VLAN No
VLAN Name
1 default
20 VLAN1
1002 fddi-default
1003 token-ring-default
1004 fddinet-default
1005 trnet-default
Equivalent IOS Commands
Switch>enable
Switch#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)#interface FastEthernet0/5
Switch(config-if)#
Switch(config-if)#exit
Switch(config)#

Program 7:

To construct a WLAN and make the nodes communicate wirelessly

Aim:

To set up a WLAN and enable wireless communication between the connected nodes.

Observation

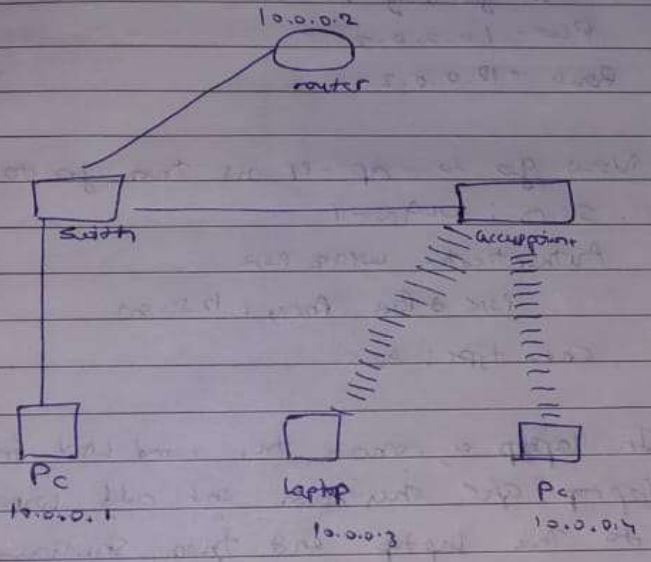
PAGE NO :
DATE :

~~LAB~~ LAB 8

Aim:

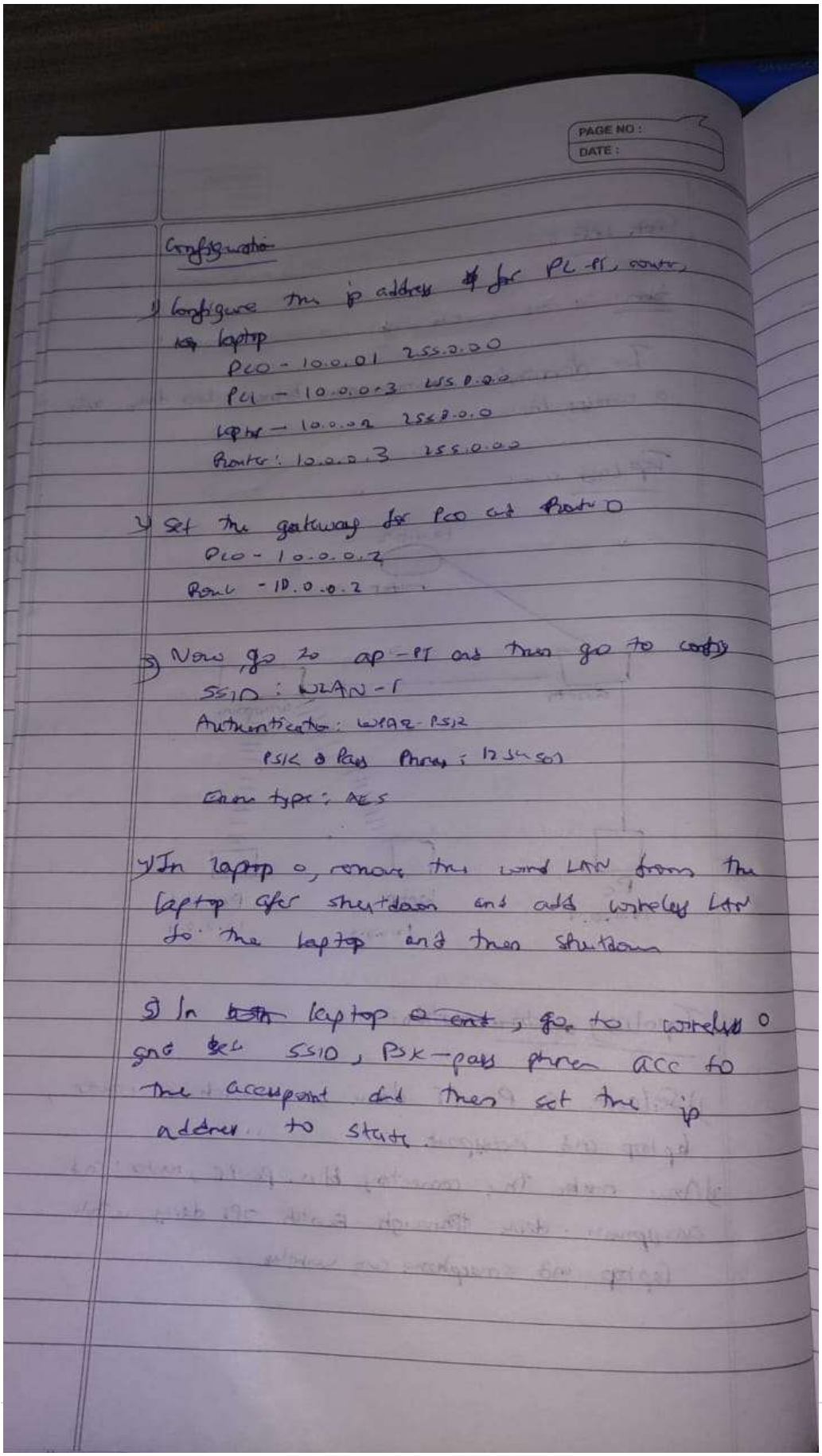
To demonstrate communication between two devices using a wireless LAN.

Topology



Topology ~~creation~~ creation:

1) Select a Pc-PT device, a switch, a router, laptop and accesspoint
2) Now make the connections b/w pc-PT, router and accesspoint device through Switch-PT device, while laptop and smartphone are wirelessly



PAGE NO :
DATE :

Configuration

1) Configure the ip address for PC-PT, router, laptop
PC0 - 10.0.0.1 255.0.0.0
PC1 - 10.0.0.3 255.0.0.0
Laptop - 10.0.0.2 255.0.0.0
Router: 10.0.0.3 255.0.0.0

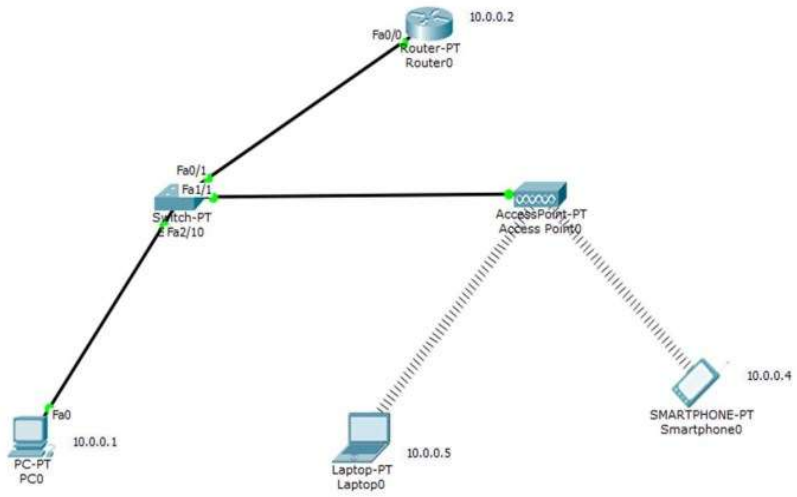
2) Set the gateway for PC0 and Router 0
PC0 - 10.0.0.2
Router - 10.0.0.2

3) Now go to ap-PT and then go to config
SSID : WLAN-1
Authentication: WPA2-PSK2
PSK Pass Phrase : 12345678
Encryption type: AES

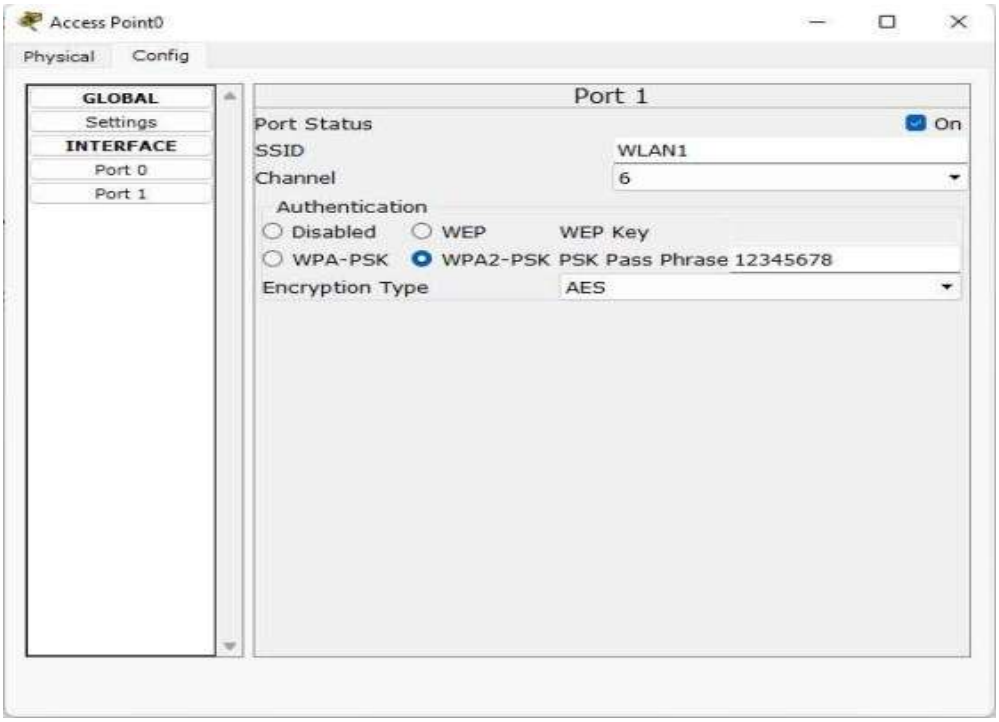
4) In laptop 0, remove the card LAN from the laptop after shutdown and add wireless LAN to the laptop and then shutdown

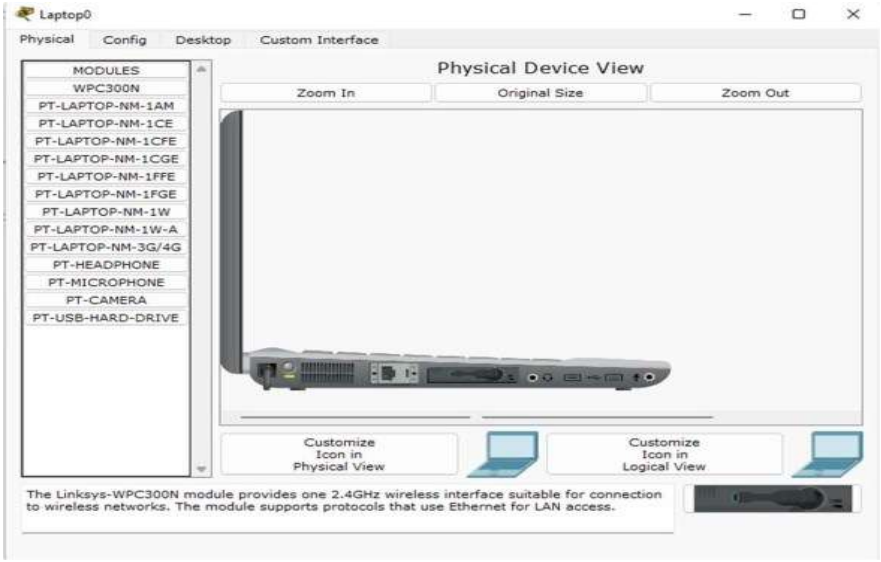
5) In laptop 0 ~~desktop~~, go to wireless 0 and set SSID, PSK-pass phrase acc to the accespoint and then set the ip address to static.

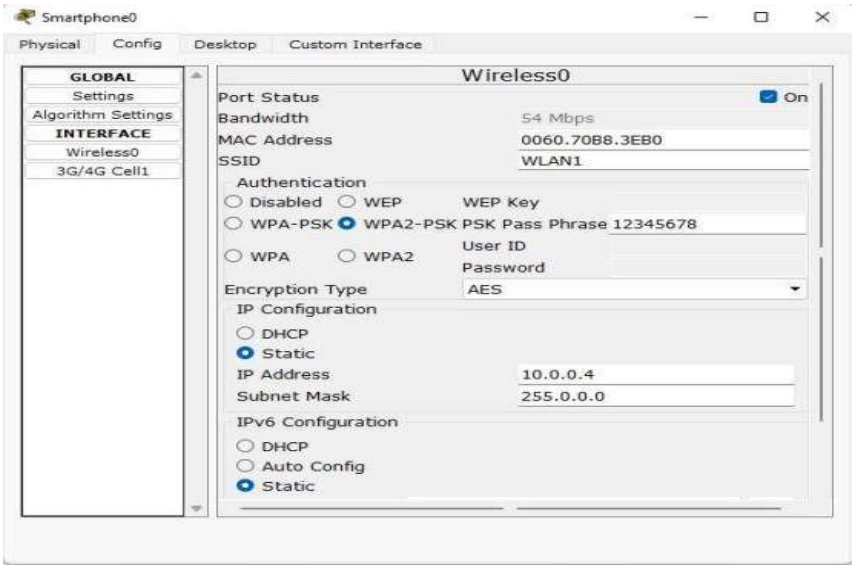
Topology

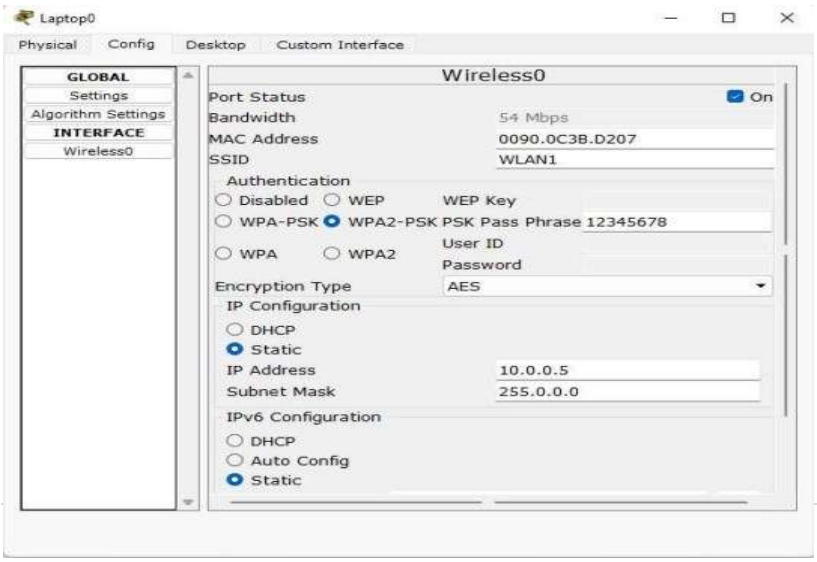


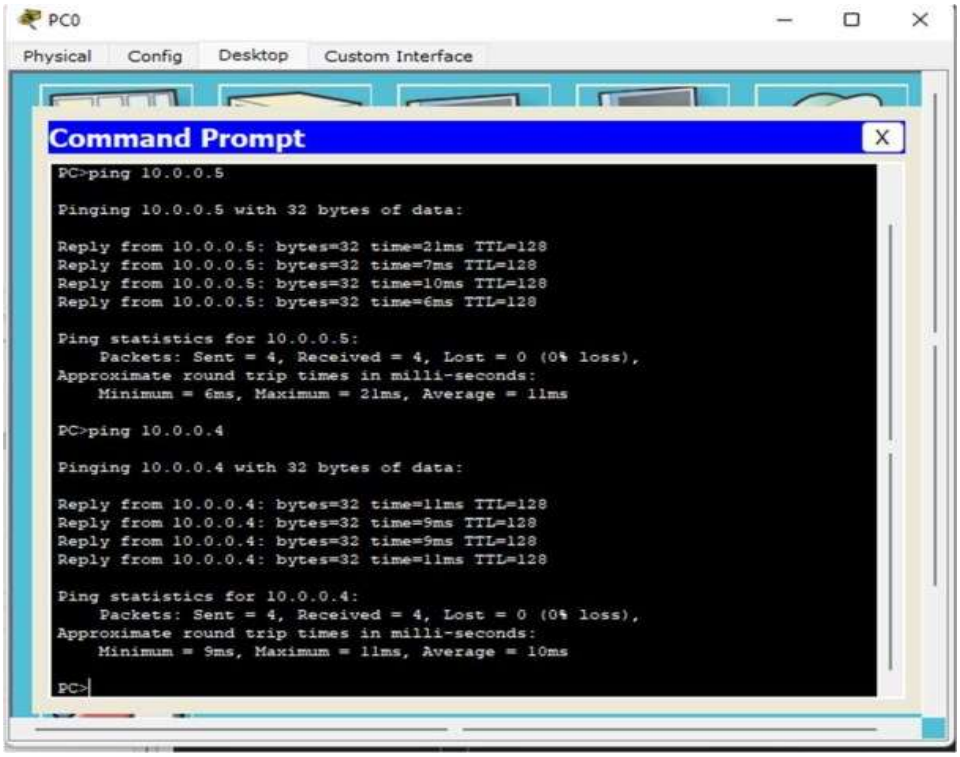
Output




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.


Smartphone0
Physical
Config
Desktop
Custom Interface
GLOBAL
Settings
Algorithm Settings
INTERFACE
Wireless0
3G/4G Cell1
Wireless0
Port Status
On
Bandwidth
54 Mbps
MAC Address
0060.7088.3EB0
SSID
WLAN1
Authentication
Disabled
WEP
WEP Key
WPA-PSK
WPA2-PSK
PSK Pass Phrase
12345678
WPA
WPA2
User ID
Password
Encryption Type
AES
IP Configuration
DHCP
Static
IP Address
10.0.0.4
Subnet Mask
255.0.0.0
IPv6 Configuration
DHCP
Auto Config
Static


Laptop0
Physical
Config
Desktop
Custom Interface
GLOBAL
Settings
Algorithm Settings
INTERFACE
Wireless0
Wireless0
Port Status
On
Bandwidth
54 Mbps
MAC Address
0090.0C3B.D207
SSID
WLAN1
Authentication
Disabled
WEP
WEP Key
WPA-PSK
WPA2-PSK
PSK Pass Phrase
12345678
WPA
WPA2
User ID
Password
Encryption Type
AES
IP Configuration
DHCP
Static
IP Address
10.0.0.5
Subnet Mask
255.0.0.0
IPv6 Configuration
DHCP
Auto Config
Static


PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
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=21ms TTL=128
Reply from 10.0.0.5: bytes=32 time=7ms TTL=128
Reply from 10.0.0.5: bytes=32 time=10ms TTL=128
Reply from 10.0.0.5: bytes=32 time=6ms TTL=128
Ping statistics for 10.0.0.5:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 6ms, Maximum = 21ms, Average = 11ms
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=11ms TTL=128
Reply from 10.0.0.4: bytes=32 time=9ms TTL=128
Reply from 10.0.0.4: bytes=32 time=9ms 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:
Minimum = 9ms, Maximum = 11ms, Average = 10ms
PC>

Program 8: To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)

Aim:

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

Observation

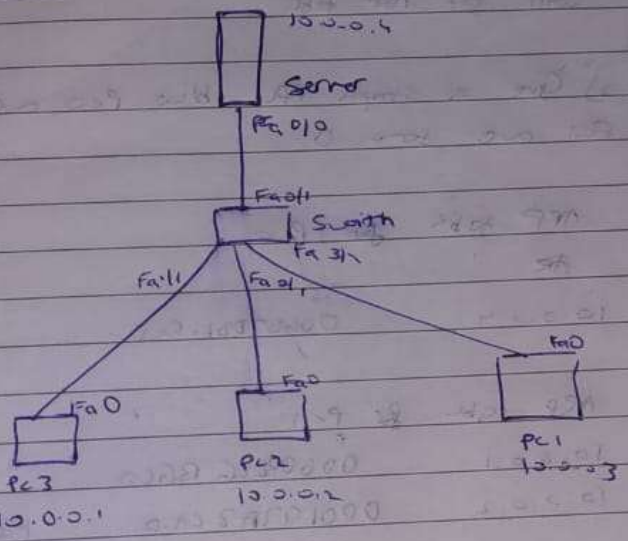
PAGE NO :
DATE :

ARP

To construct simple LAN and understand the concept of operation of ARP

Topic:
Aim: To demonstrate the working of ARP for communication within a LAN

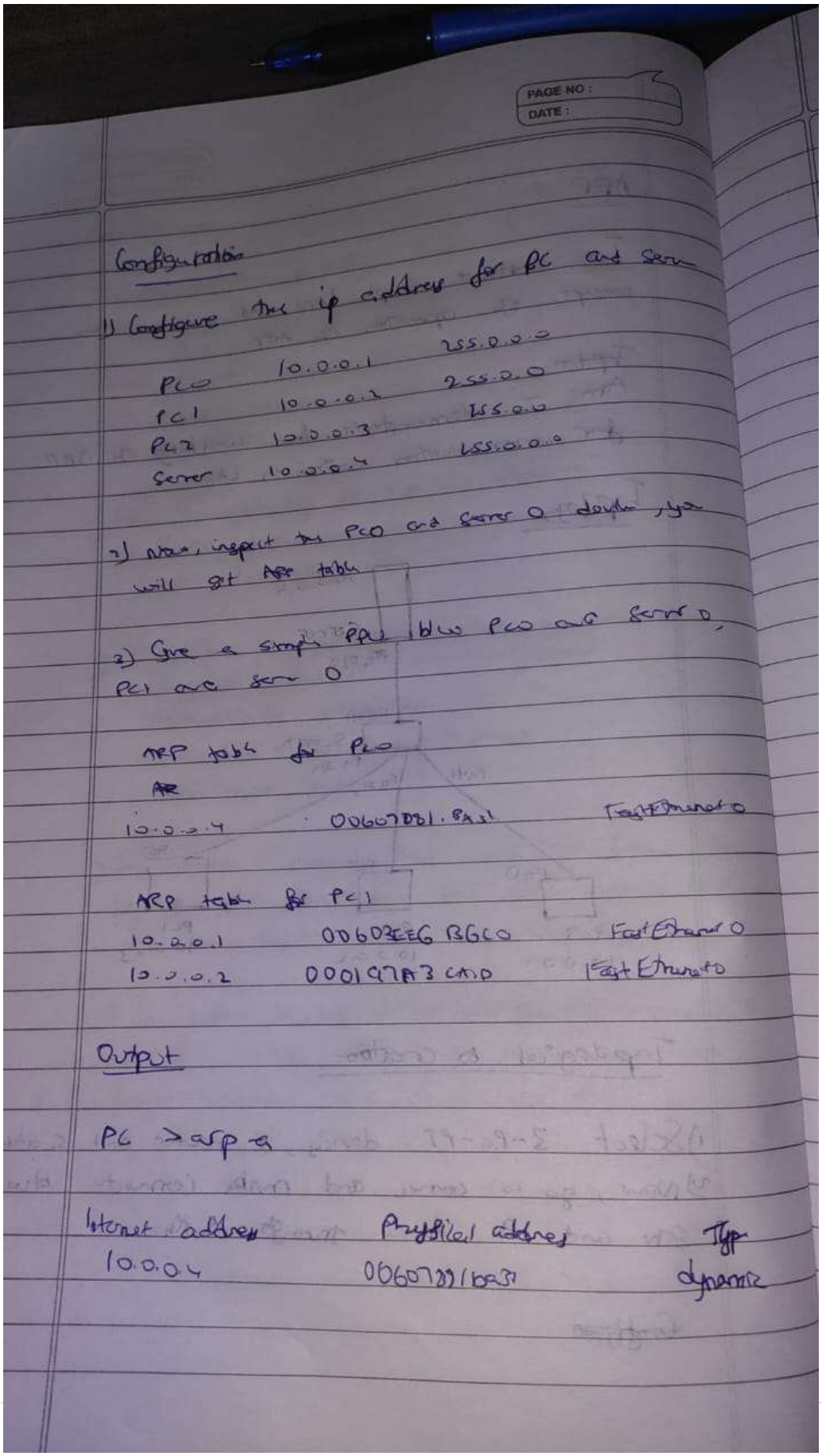
Topology



Topological creation:

1) Select 3-Pc-PT device, a router and switch
2) Now, go to connect and make connect b/w Server and Pc device through Switch

Configure



PAGE NO :
DATE :

Configuration

1) Configure the ip address for PC and Server

PC0	10.0.0.1	255.0.0.0
PC1	10.0.0.2	255.0.0.0
PC2	10.0.0.3	255.0.0.0
Server	10.0.0.4	255.0.0.0

2) Now, inspect the PC0 and Server 0 double, you will get ARP table

2) Give a simple PDU b/w PC0 and Server 0, PC1 and Server 0

ARP table for PC0

Address		
10.0.0.4	0060 7DB1. BA31	FastEthernet0

ARP table for PC1

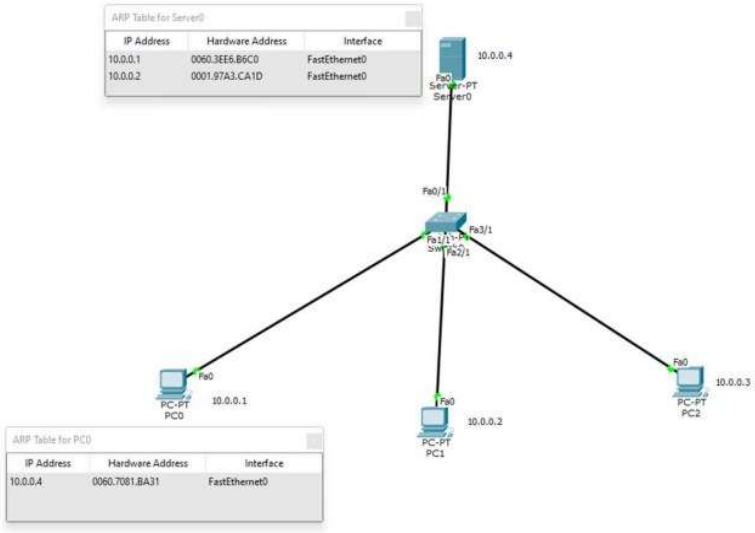
10.0.0.1	0060 3EE6 B6C0	FastEthernet0
10.0.0.2	0001 97A3 CA1D	FastEthernet0

Output

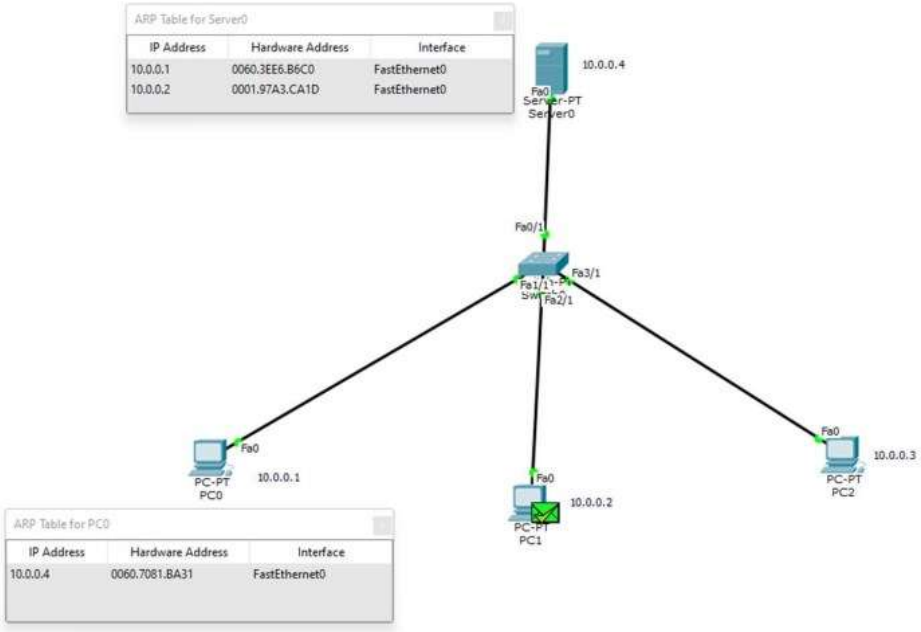
PC > arp -a

Internet address	Physical address	Type
10.0.0.4	0060 7B71 BA31	dynamic

Topology



Output



ARP Table for Server0

IP Address	Hardware Address	Interface
10.0.0.1	0060.3EE6.B6C0	FastEthernet0
10.0.0.2	0001.97A3.CA1D	FastEthernet0

ARP Table for PC0

IP Address	Hardware Address	Interface
10.0.0.4	0060.7081.BA31	FastEthernet0

PDU Information at Device: PC1

OSI Model | Inbound PDU Details

PDU Formats

Ethernet II

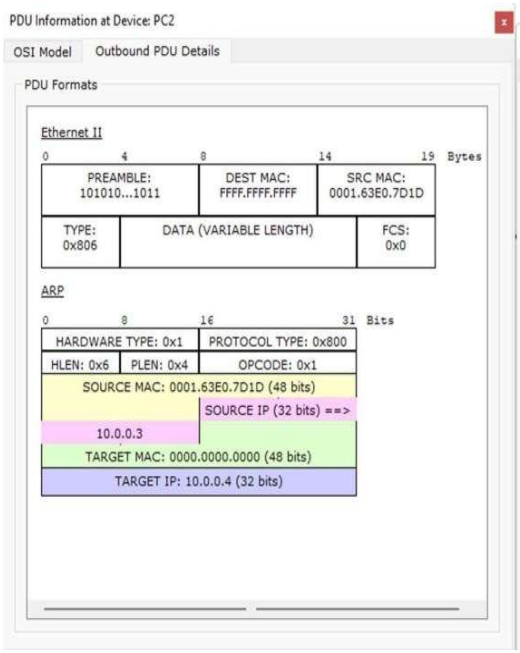
0	4	8	14	19 Byt
PREAMBLE: 101010...1011		DEST MAC: 0001.97A3.CA1D	SRC MAC: 0060.7081.BA31	
TYPE: 0x800	DATA (VARIABLE LENGTH)			FCS: 0x0

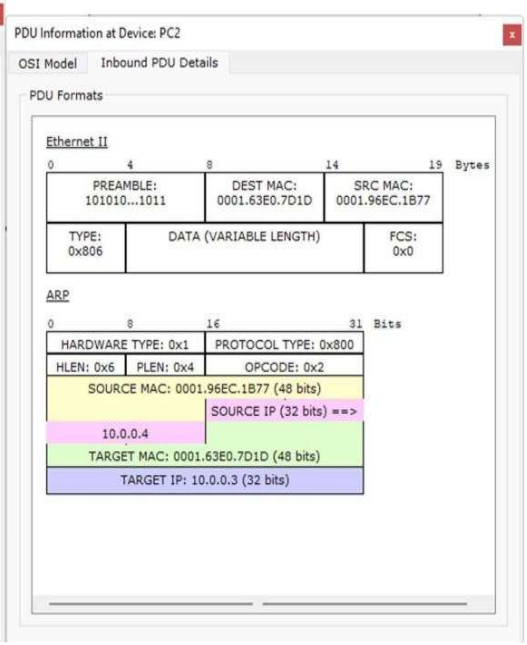
IP

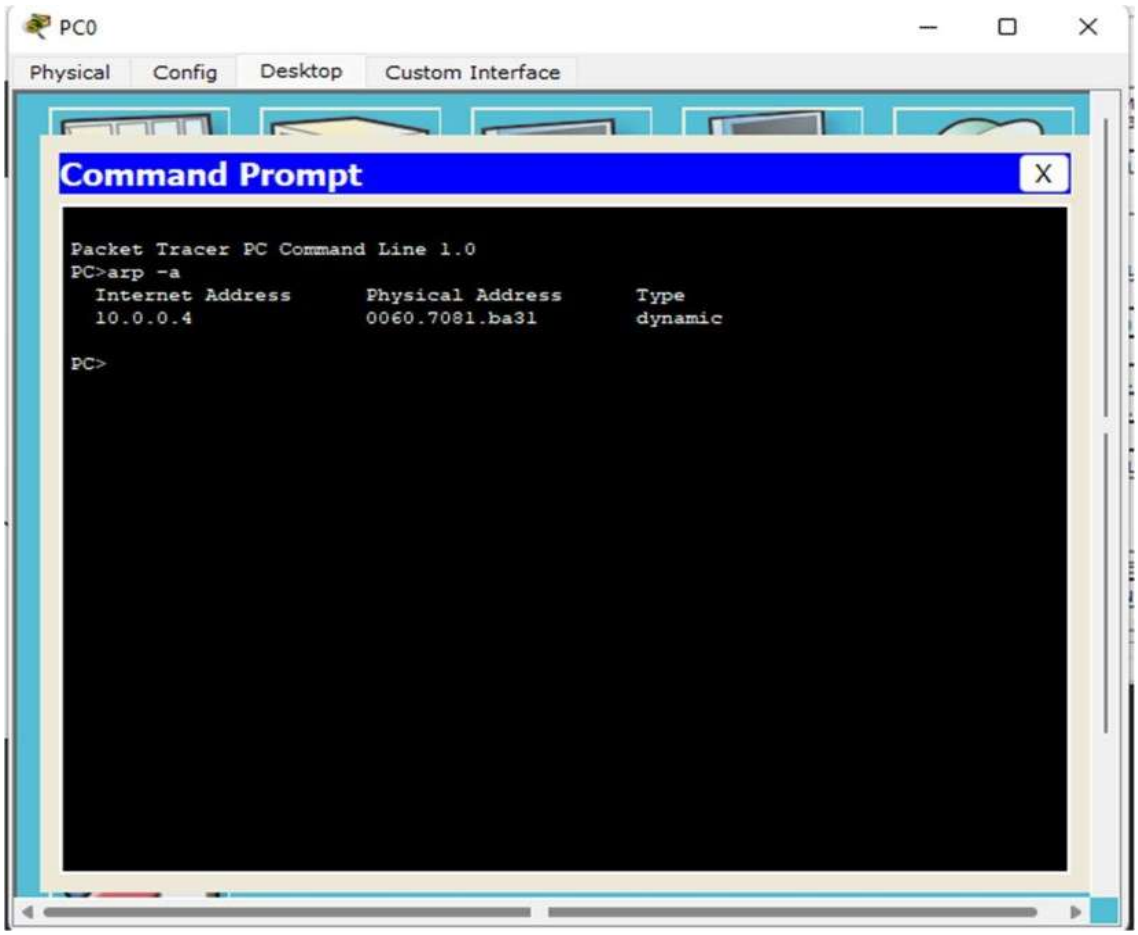
0	4	8	16 19	31 Bits
4	IHL	DSCP: 0x0	TL: 28	
ID: 0xd		0x0	0x0	
TTL: 128		PRO: 0x1	CHKSUM	
SRC IP: 10.0.0.4				
DST IP: 10.0.0.2				
OPT: 0x0				0x0
DATA (VARIABLE LENGTH)				

ICMP

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


PDU Information at Device: PC2
OSI Model
Outbound PDU Details
PDU Formats
Ethernet II
0 4 8 14 19 Bytes
PREAMBLE: 101010...1011
DEST MAC: FFFF.FFFF.FFFF
SRC MAC: 0001.63E0.7D1D
TYPE: 0x806
DATA (VARIABLE LENGTH)
FCS: 0x0
ARP
0 8 16 31 Bits
HARDWARE TYPE: 0x1
PROTOCOL TYPE: 0x800
HLEN: 0x6
PLEN: 0x4
OPCODE: 0x1
SOURCE MAC: 0001.63E0.7D1D (48 bits)
SOURCE IP (32 bits) ==>
10.0.0.3
TARGET MAC: 0000.0000.0000 (48 bits)
TARGET IP: 10.0.0.4 (32 bits)


PDU Information at Device: PC2
OSI Model
Inbound PDU Details
PDU Formats
Ethernet II
0 4 8 14 19 Bytes
PREAMBLE: 101010...1011
DEST MAC: 0001.63E0.7D1D
SRC MAC: 0001.96EC.1B77
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: 0001.96EC.1B77 (48 bits)
SOURCE IP (32 bits) ==>
10.0.0.4
TARGET MAC: 0001.63E0.7D1D (48 bits)
TARGET IP: 10.0.0.3 (32 bits)


PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
X
Packet Tracer PC Command Line 1.0
PC>arp -a
Internet Address Physical Address Type
10.0.0.4 0060.7081.ba31 dynamic
PC>

Program 9:
Configure OSPF routing protocol

Aim:
To configure the OSPF routing protocol and understand its operation in dynamic routing.

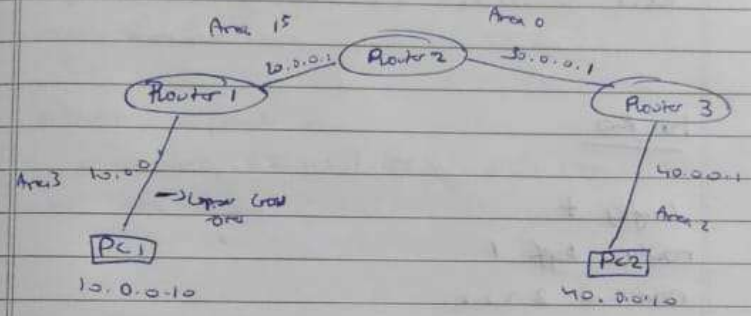
Observation

PAGE NO :
DATE :

OSPF Simulation Program

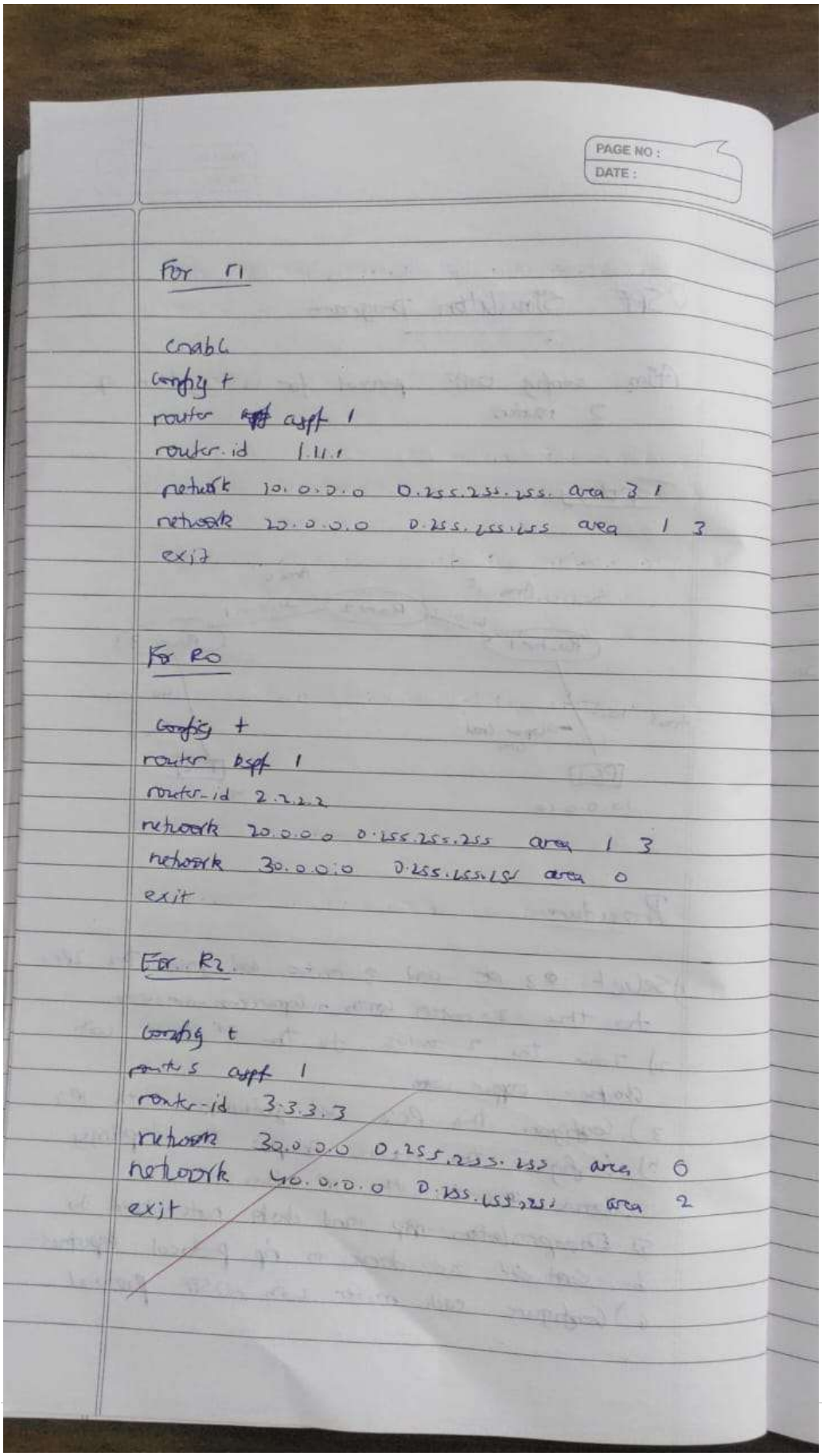
Aim: config OSPF protocol for a system of 3 routers

Topology



Procedure:

1) Select 02 PC and 3 routers and join the PC's to the 2 routers with Copper cross-over wire
2) Join the 2 routers to the 3rd router with straight copper wire
3) Configure the PC's and gateways with IP's
4) Configure the routers as per the topology above with the IP address
5) Encapsulation PPP and clock rate need to be set as done in rip protocol experiment.
6) Configure each router with OSPF protocol



For r1

conf t
router ospf 1
router-id 1.1.1.1
network 10.0.0.0 0.255.255.255 area 3 1
network 20.0.0.0 0.255.255.255 area 1 3
exit

For R0

Config t
router ospf 1
router-id 2.2.2.2
network 20.0.0.0 0.255.255.255 area 1 3
network 30.0.0.0 0.255.255.255 area 0
exit

For R2.

Config t
router ospf 1
router-id 3.3.3.3
network 32.0.0.0 0.255.255.255 area 0
network 40.0.0.0 0.255.255.255 area 2
exit

PAGE NO :
DATE :

Configure the interfaces

interface loopback 0
ip add 172.16.1.254 255.255.0.0
no shutdown

interface loopback 0
ip addres 172.16.1.253 255.255.0.0
no shutdown

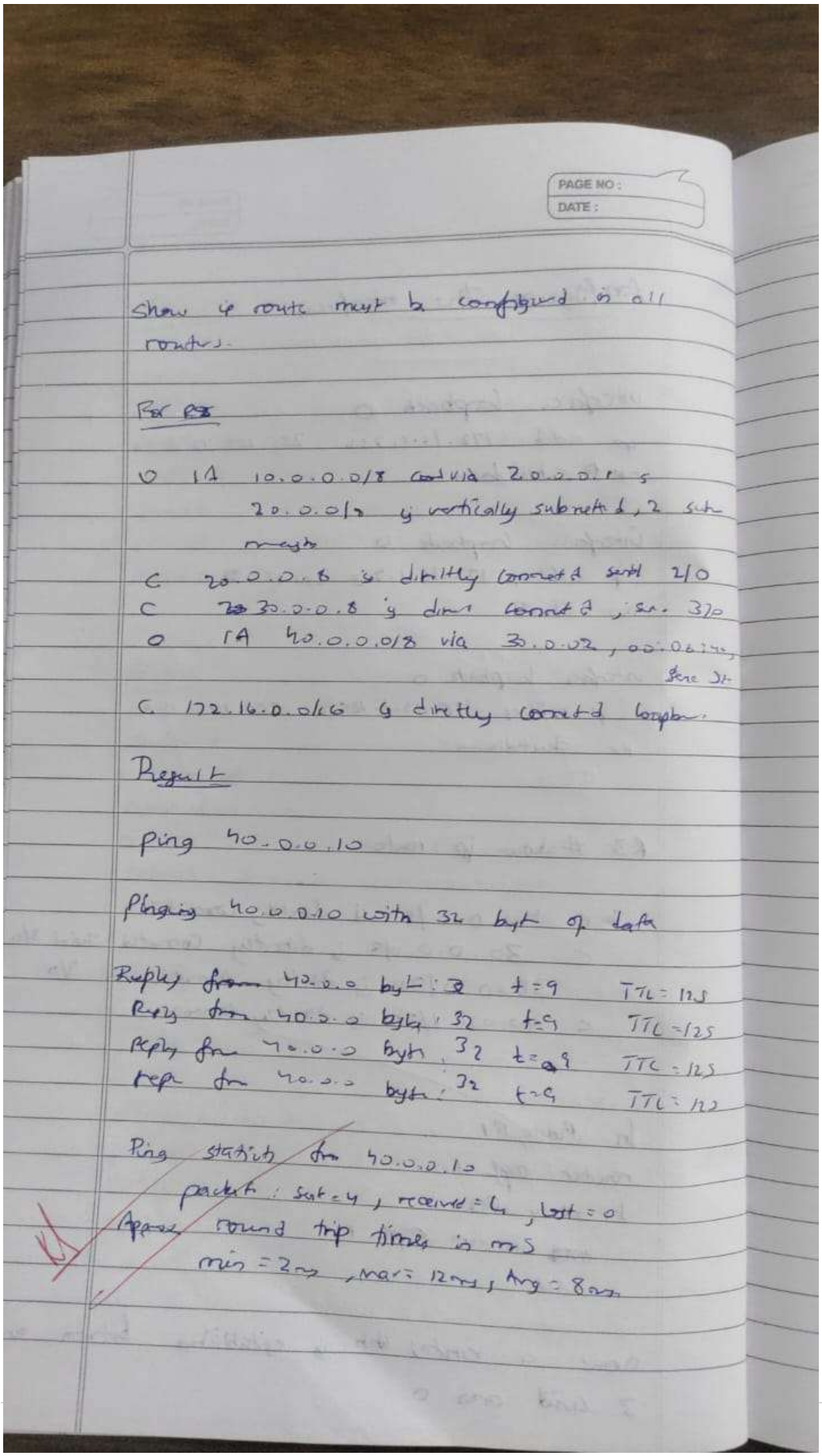
interface loopback 0
ip addres 172.16.1.254 255.255.0.0
no shutdown

R3 #show ip route

C 40.0.0.0/8 is directly connected
C 30.0.0.0/8 is directly connected, Serial 3/0
30.0.0.1/32 is directly connected, Serial 3/0
C 40.0.0.1/32 is directly connected

In Router R1
router ospf 1
In route R2
router ospf 1
exit

now a virtual link is established between area 3 and area 0



PAGE NO :
DATE :

show ip route must be configured in all routers.

For R3

O IA 10.0.0.0/8 [110/2] via 20.0.0.1, 5
20.0.0.0/8 is variably subnetted 2, 2 subnets

C 20.0.0.8 is directly connected, Se 2/0

C 30.0.0.8 is directly connected, Se. 3/0

O IA 40.0.0.0/8 via 30.0.0.2, 00:04:40, Se 3/0

C 172.16.0.0/16 is directly connected loopback.

Result

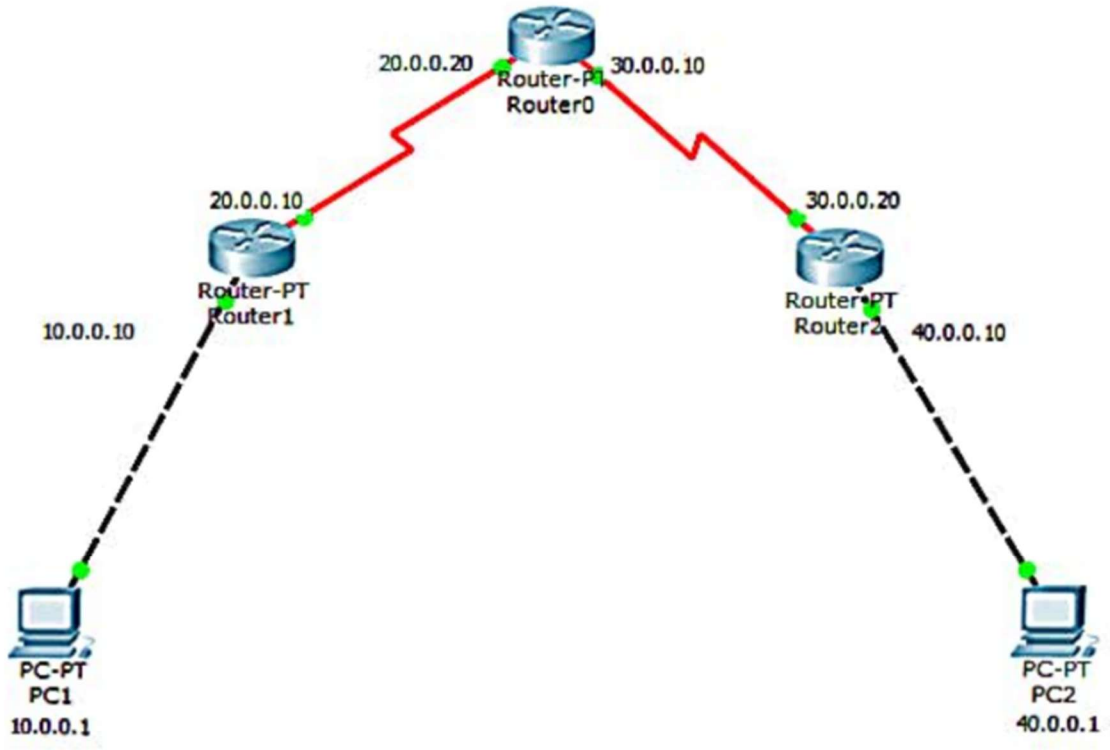
ping 40.0.0.10

Pinging 40.0.0.10 with 32 byte of data

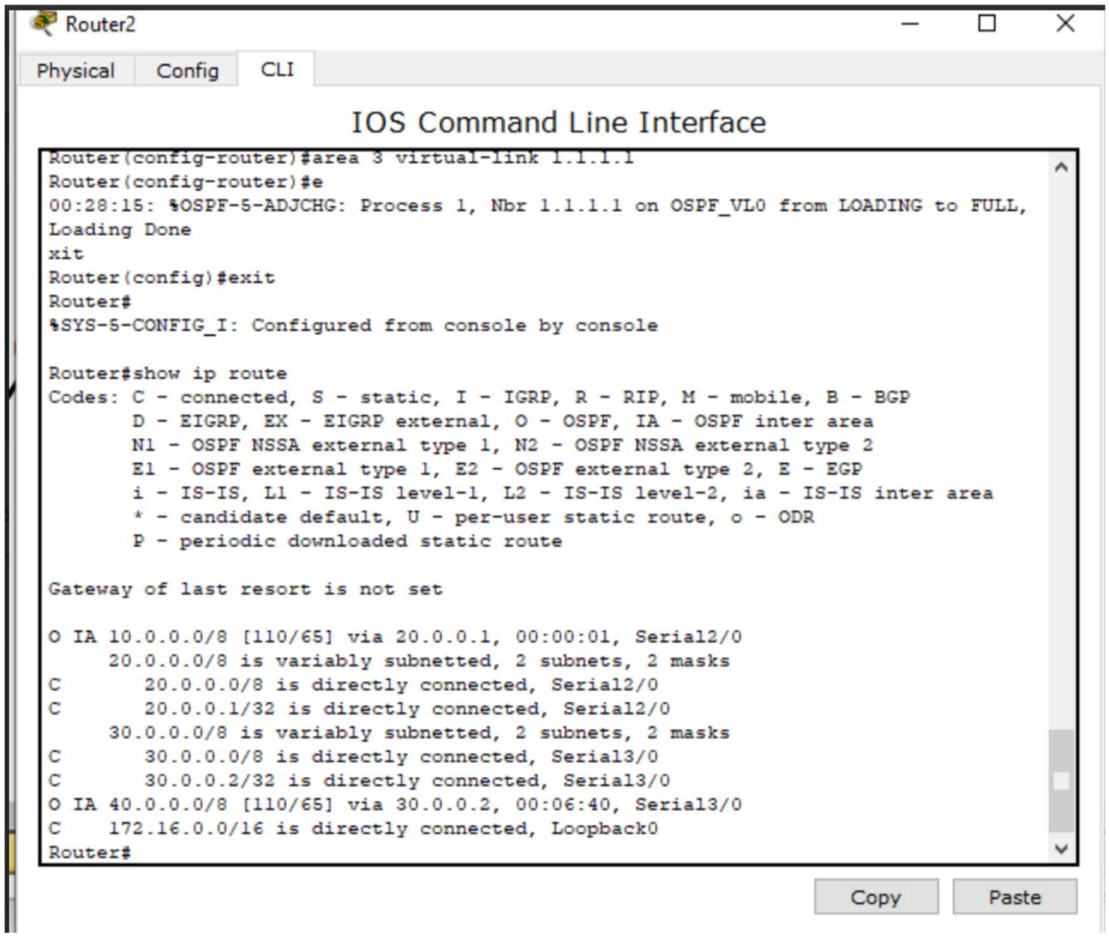
Reply from 40.0.0.0 byte: 32 t=9 TTL=125
Reply from 40.0.0.0 byte: 32 t=9 TTL=125
Reply from 40.0.0.0 byte: 32 t=9 TTL=125
Reply from 40.0.0.0 byte: 32 t=9 TTL=125

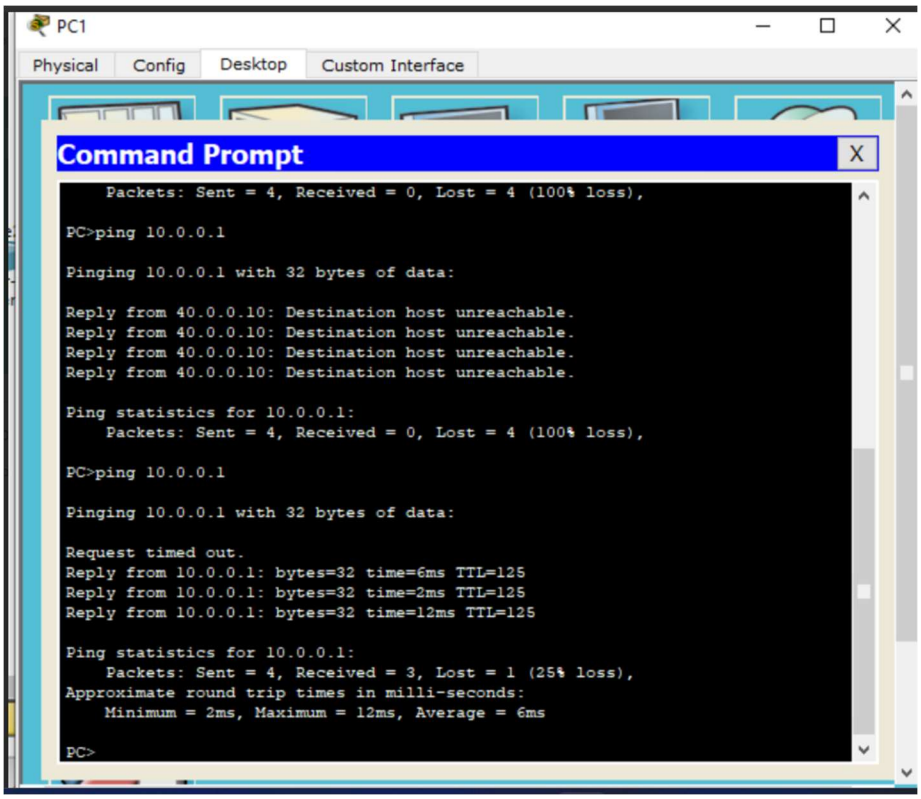
Ping statistics for 40.0.0.10
packet: sent=4, received=4, lost=0
Approx round trip times in ms
min=2ms, max=12ms, Avg=8ms

Topology


20.0.0.20
30.0.0.10
Router-PT
Router0
20.0.0.10
30.0.0.20
Router-PT
Router1
Router-PT
Router2
10.0.0.10
40.0.0.10
PC-PT
PC1
10.0.0.1
PC-PT
PC2
40.0.0.1

Output


Router2
Physical
Config
CLI
IOS Command Line Interface
Router(config-router)#area 3 virtual-link 1.1.1.1
Router(config-router)#e
00:28:15: %OSPF-5-ADJCHG: Process 1, Nbr 1.1.1.1 on OSPF_VL0 from LOADING to FULL,
Loading Done
xit
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
O IA 10.0.0.0/8 [110/65] via 20.0.0.1, 00:00:01, Serial2/0
20.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C 20.0.0.0/8 is directly connected, Serial2/0
C 20.0.0.1/32 is directly connected, Serial2/0
30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C 30.0.0.0/8 is directly connected, Serial3/0
C 30.0.0.2/32 is directly connected, Serial3/0
O IA 40.0.0.0/8 [110/65] via 30.0.0.2, 00:06:40, Serial3/0
C 172.16.0.0/16 is directly connected, Loopback0
Router#
Copy
Paste


PC1
Physical
Config
Desktop
Custom Interface
Command Prompt
X
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 40.0.0.10: Destination host unreachable.
Reply from 40.0.0.10: Destination host unreachable.
Reply from 40.0.0.10: Destination host unreachable.
Reply from 40.0.0.10: Destination host unreachable.
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:
Request timed out.
Reply from 10.0.0.1: bytes=32 time=6ms TTL=125
Reply from 10.0.0.1: bytes=32 time=2ms TTL=125
Reply from 10.0.0.1: bytes=32 time=12ms TTL=125
Ping statistics for 10.0.0.1:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 2ms, Maximum = 12ms, Average = 6ms
PC>

Program 10
Configure RIP routing Protocol in Routers

Aim:

To configure the RIP routing protocol on routers and understand its operation in dynamic routing.

Observation

PAGE NO :
DATE :

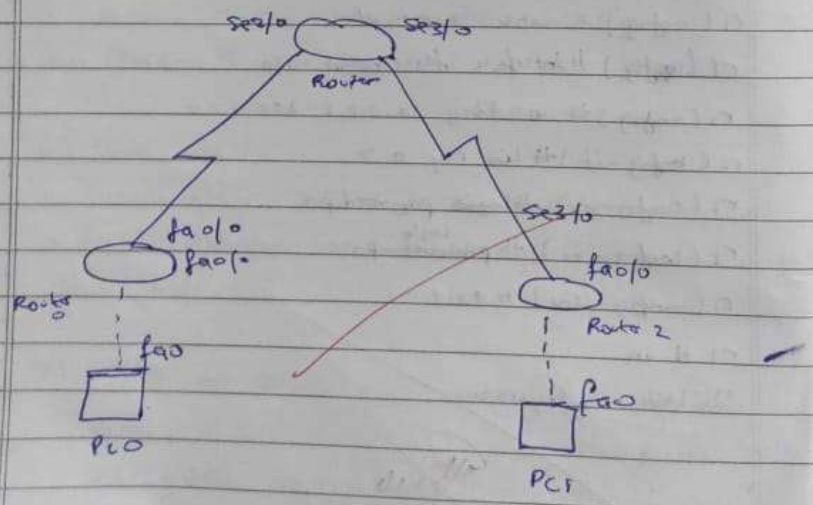
Week - 5

Aim : Routing device using RIP-V2

Procedure :

* Need 2 Pc (PC0 and PC1) and three routers (R0, R1, R2)
* Connect Pc0 to R0 and R1 to R1 through copper cross over cable
* Then connect routers to router 1 through serial dcd cable to router 1 via serial 2 port through that to serial 2 port in router 1. and then router 2 serial 2/ serial 3 port to router 2 (Serial 3 port)

Topology



PAGE NO :
DATE :

Then move to the config the ip address of PC0 and PC1 which is

PC0 :-

IP address → 10.0.0.2
subnet mask → 255.0.0.0
default gateway → 10.0.0.1

PC1 :-

IP address → 40.0.0.2
subnet mask → 255.0.0.0
default gateway → 40.0.0.1

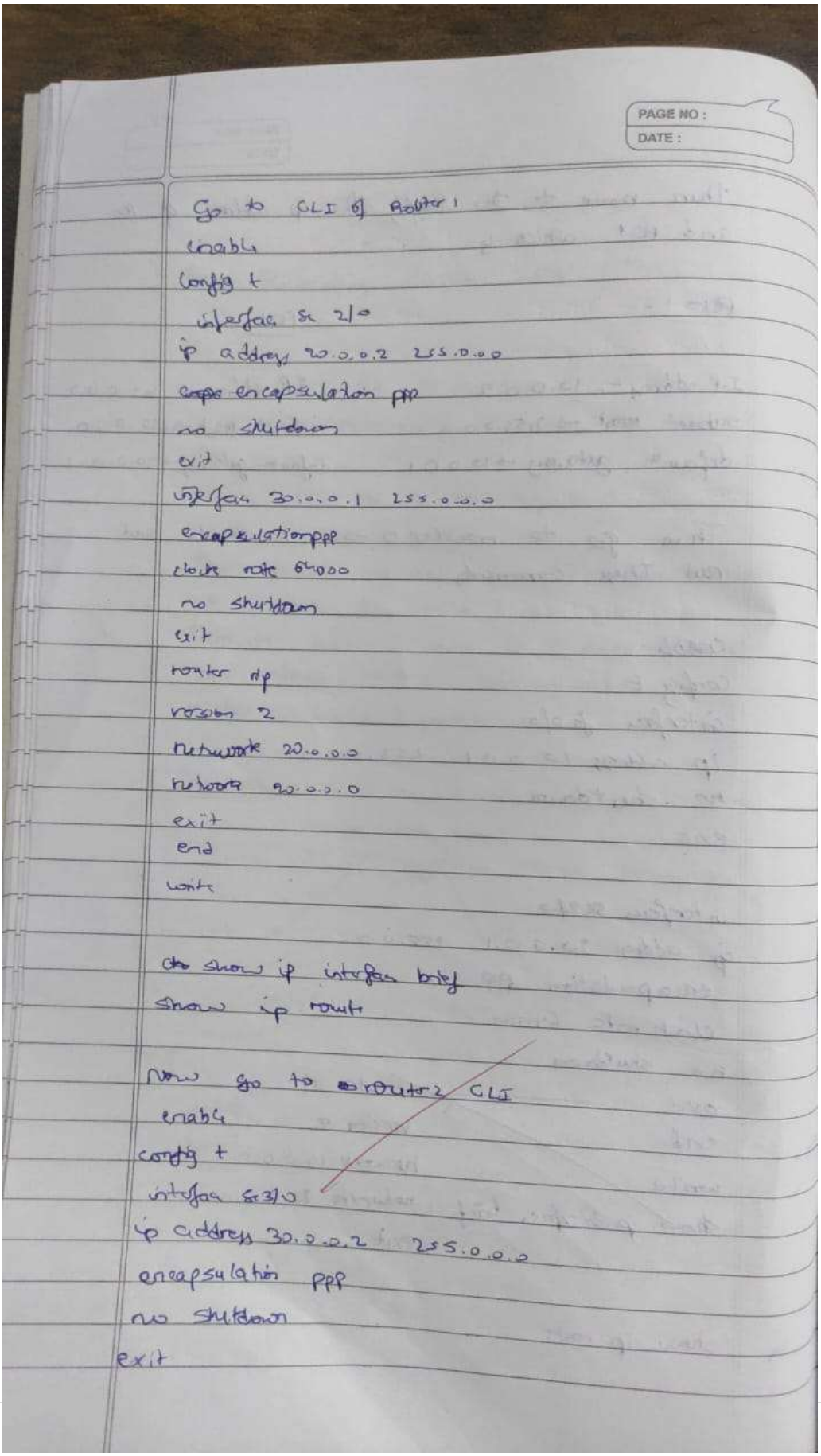
Then go to router 0 → CLI mode and run these commands

enable
config t
interface fa 0/0
ip address 10.0.0.1 255.0.0.0
no shutdown
exit

interface se 2/0
ip address 20.0.0.1 255.0.0.0
encapsulation PPP
clock rate 64000
no shutdown
exit
end
write
show ip interface brief

router rip
version 2
network 10.0.0.0
network 20.0.0.0
exit

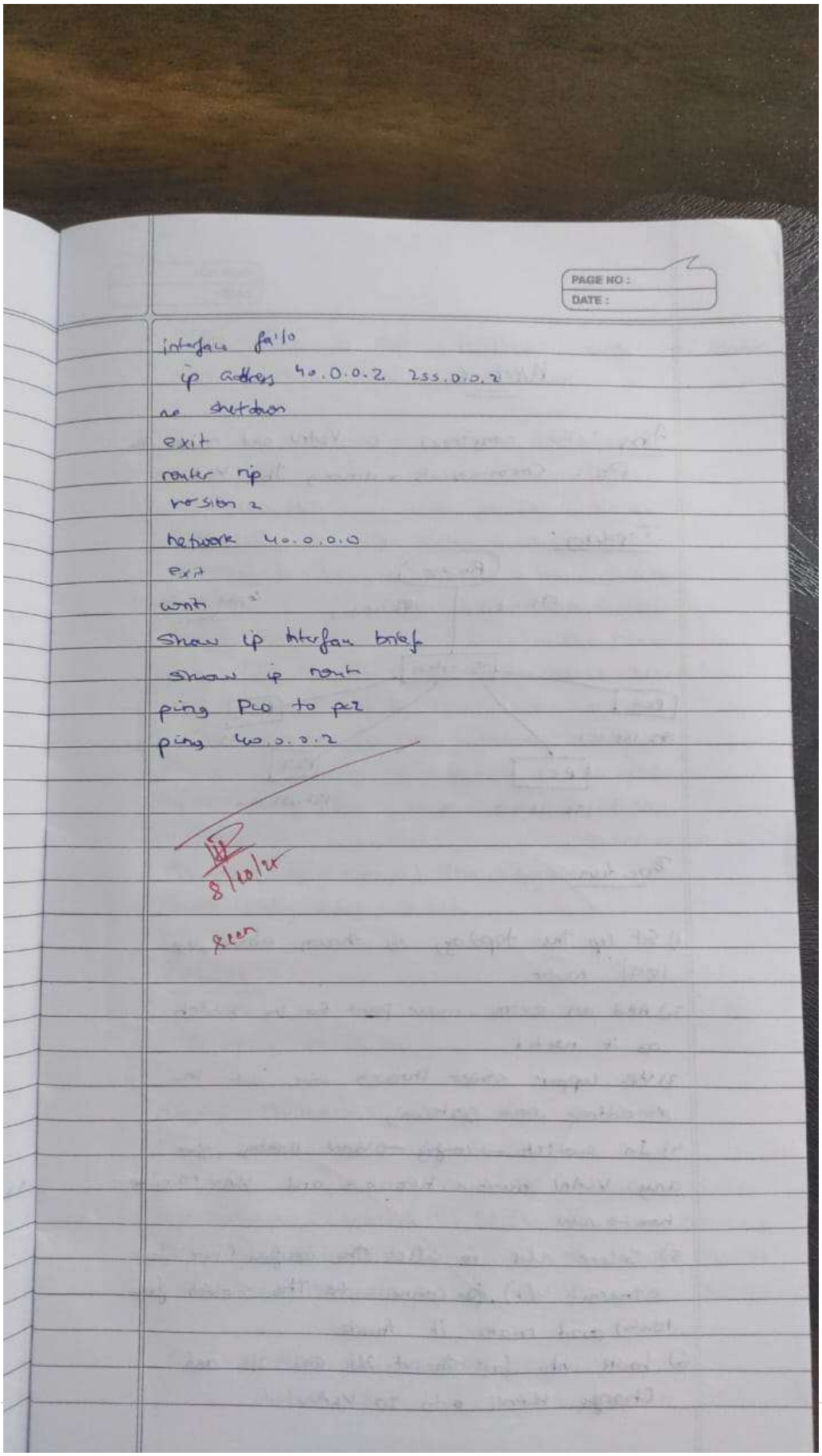
show ip route



Go to CLI of Router1
enable
config t
interface s0 2/0
ip address 20.0.0.2 255.0.0.0
encapsulation ppp
no shutdown
exit
interface 30.0.0.1 255.0.0.0
encapsulation ppp
clock rate 64000
no shutdown
exit
router rip
version 2
network 20.0.0.0
network 30.0.0.0
exit
end
write

do show ip interface brief
show ip route

Now go to router2 CLI
enable
config t
interface s3/0
ip address 30.0.0.2 255.0.0.0
encapsulation ppp
no shutdown
exit

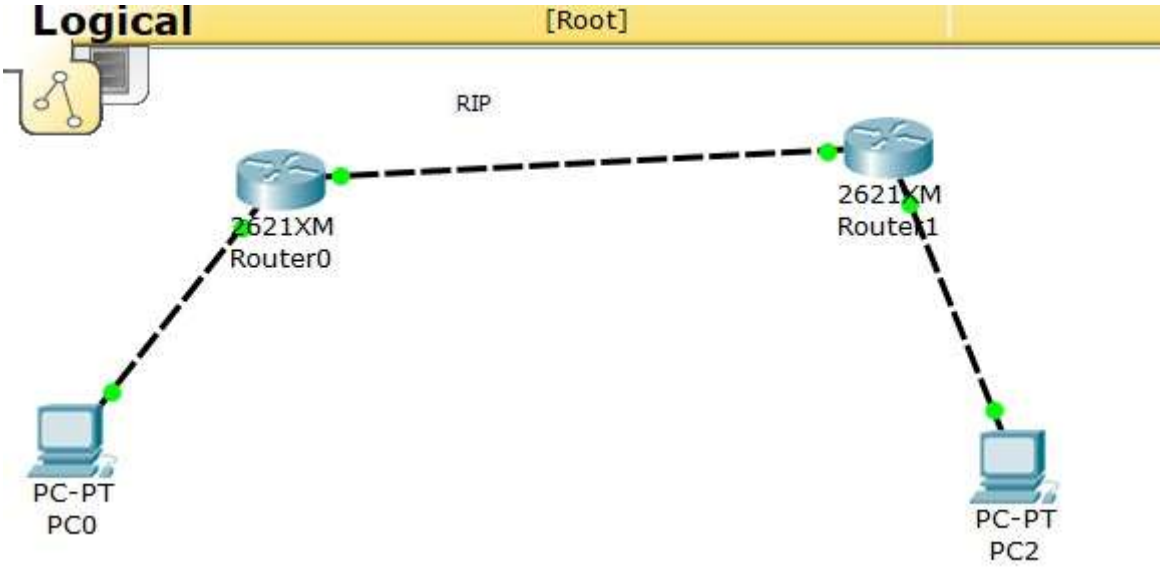

PAGE NO :
DATE :

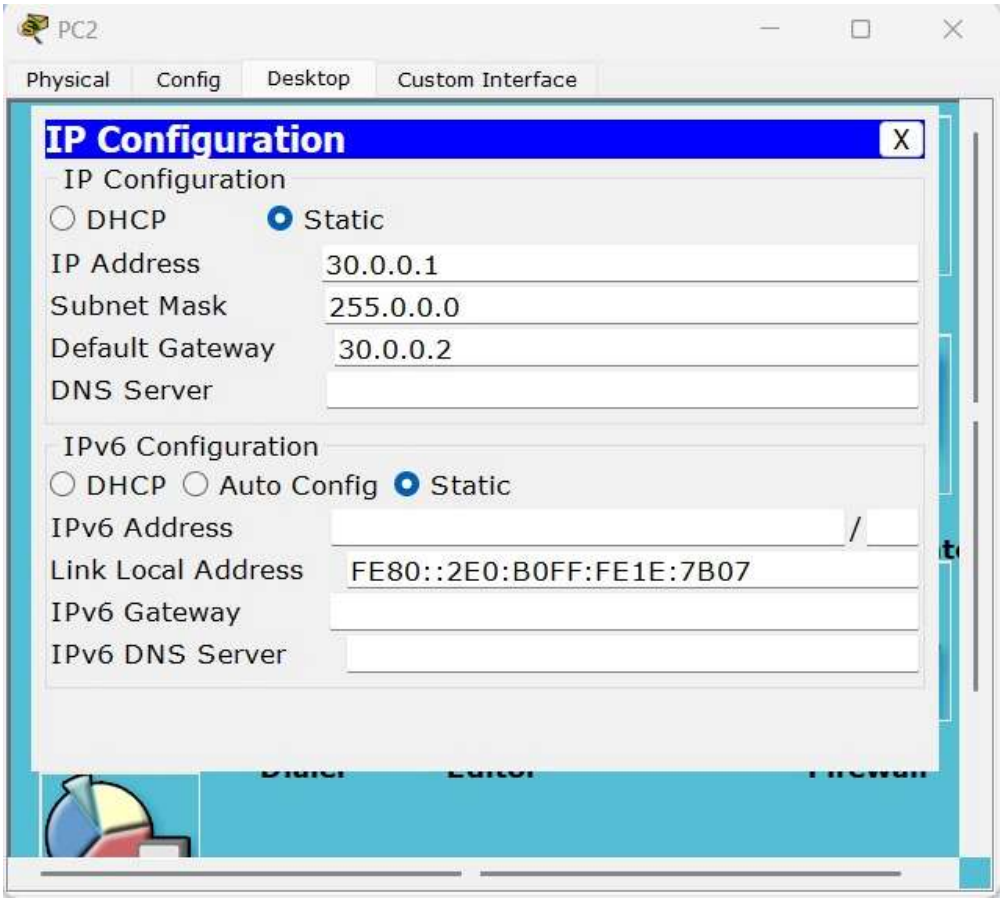
interface fa1/0
ip address 40.0.0.2 255.0.0.2
no shutdown
exit
router rip
version 2
network 40.0.0.0
exit
write
show ip interface brief
show ip route
ping PC0 to pc2
ping 40.0.0.2

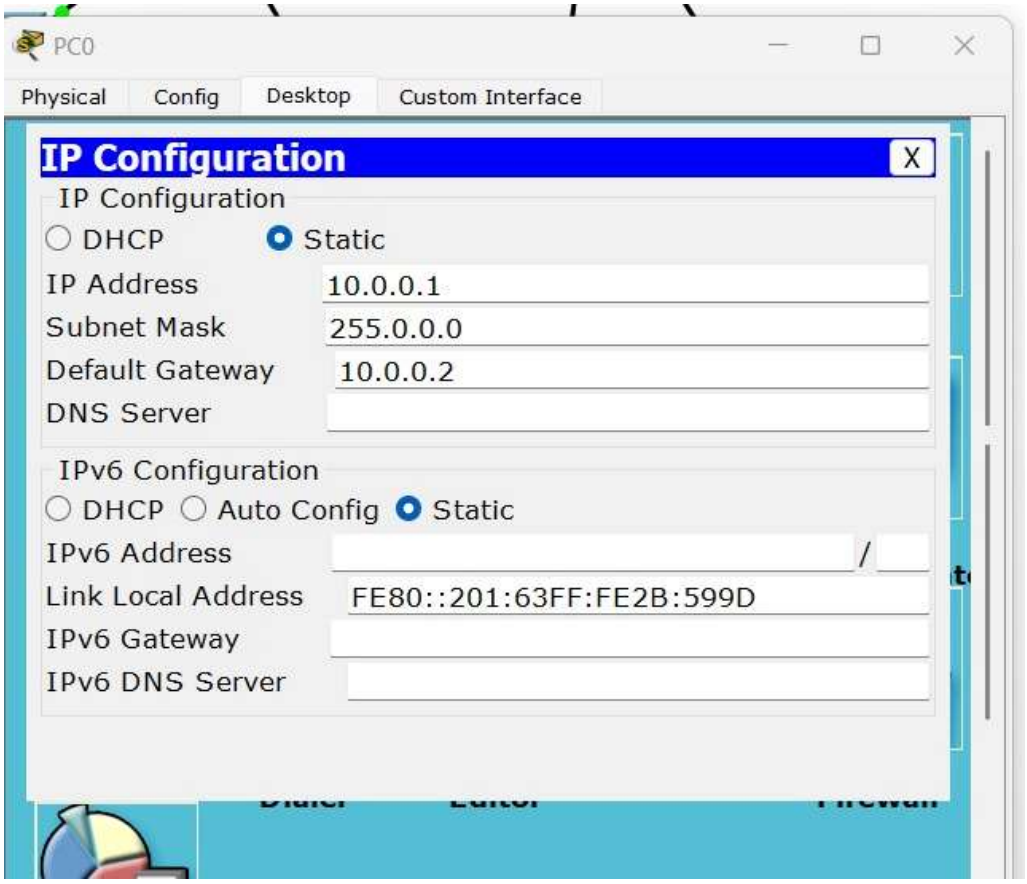
8/10/24

Steps

Topology


Logical
[Root]
RIP
2621XM
Router0
2621XM
Router1
PC-PT
PC0
PC-PT
PC2


PC2
Physical
Config
Desktop
Custom Interface
IP Configuration
X
IP Configuration
DHCP
Static
IP Address
30.0.0.1
Subnet Mask
255.0.0.0
Default Gateway
30.0.0.2
DNS Server
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
Link Local Address
FE80::2E0:B0FF:FE1E:7B07
IPv6 Gateway
IPv6 DNS Server

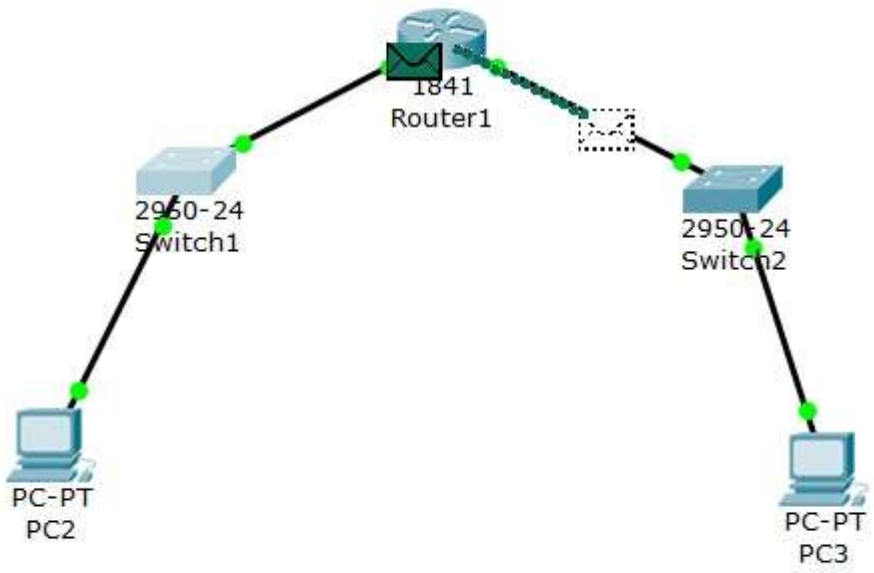

PC0
Physical
Config
Desktop
Custom Interface
IP Configuration
X
IP Configuration
DHCP
Static
IP Address
10.0.0.1
Subnet Mask
255.0.0.0
Default Gateway
10.0.0.2
DNS Server
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
/
Link Local Address
FE80::201:63FF:FE2B:599D
IPv6 Gateway
IPv6 DNS Server

Router0

Physical | Config | CLI

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface FastEthernet0/1
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/1
Router(config-if)#
Router(config-if)#exit
Router(config)#no ip route 30.0.0.0 255.0.0.0 20.0.0.2
Router(config)#
Router(config)#router rip
Router(config-router)#
Router(config-router)#exit
Router(config)#interface FastEthernet0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/1
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#router rip
Router(config-router)#network 10.0.0.0
Router(config-router)#
Router(config-router)#exit
Router(config)#interface FastEthernet0/1
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#router rip
Router(config-router)#network 20.0.0.0
Router(config-router)#
Router(config-router)#end
Router#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
[OK]
Router#
%SYS-5-CONFIG_I: Configured from console by console
```

Output


1841
Router1
2960-24
Switch1
2950-24
Switch2
PC-PT
PC2
PC-PT
PC3

Program 11

Demonstrate the TTL/ Life of a Packet

Aim:

To study and demonstrate the Time-To-Live (TTL) field in an IP packet and observe how the TTL value decreases (hops reduce) as the packet travels through a network using tools like ping or traceroute

Observation

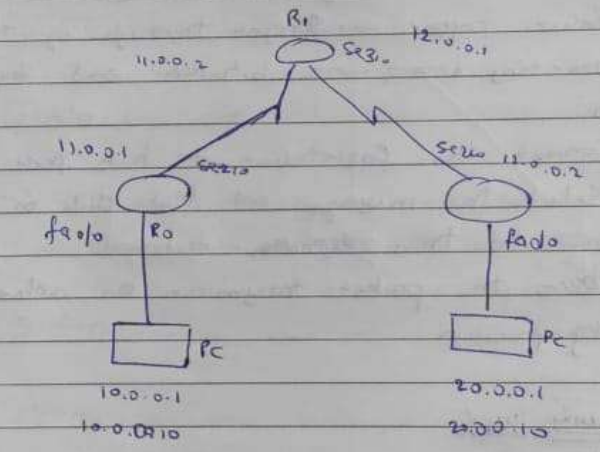
PAGE NO :
DATE :

Week 2

Packet Routing using Router as the connection device

Aim :-

Configure default Route from a static route Through the Router.



step 1 : First we we take 2 Pc's and config the IP address of both the Pc's

Pc-0 IP addres - 10.0.0.1

default gateway - 10.0.0.10

Config IP for Pc1

Pc-1 IP address - 20.0.0.1

default gateway - 20.0.0.10

Now we use the cross cover cable and and fast ethernet

PAGE NO :
DATE :

to connect to routers from both the PC's
So in router 0, the GUI commands are:

Go to CLI
enable
Config terminal
interface fa0/0
ip address 10.0.0.10 255.0.0.0
no shutdown → important
exit

interface s2/0
ip address 21.0.0.1 255.0.0.0
no shutdown
exit

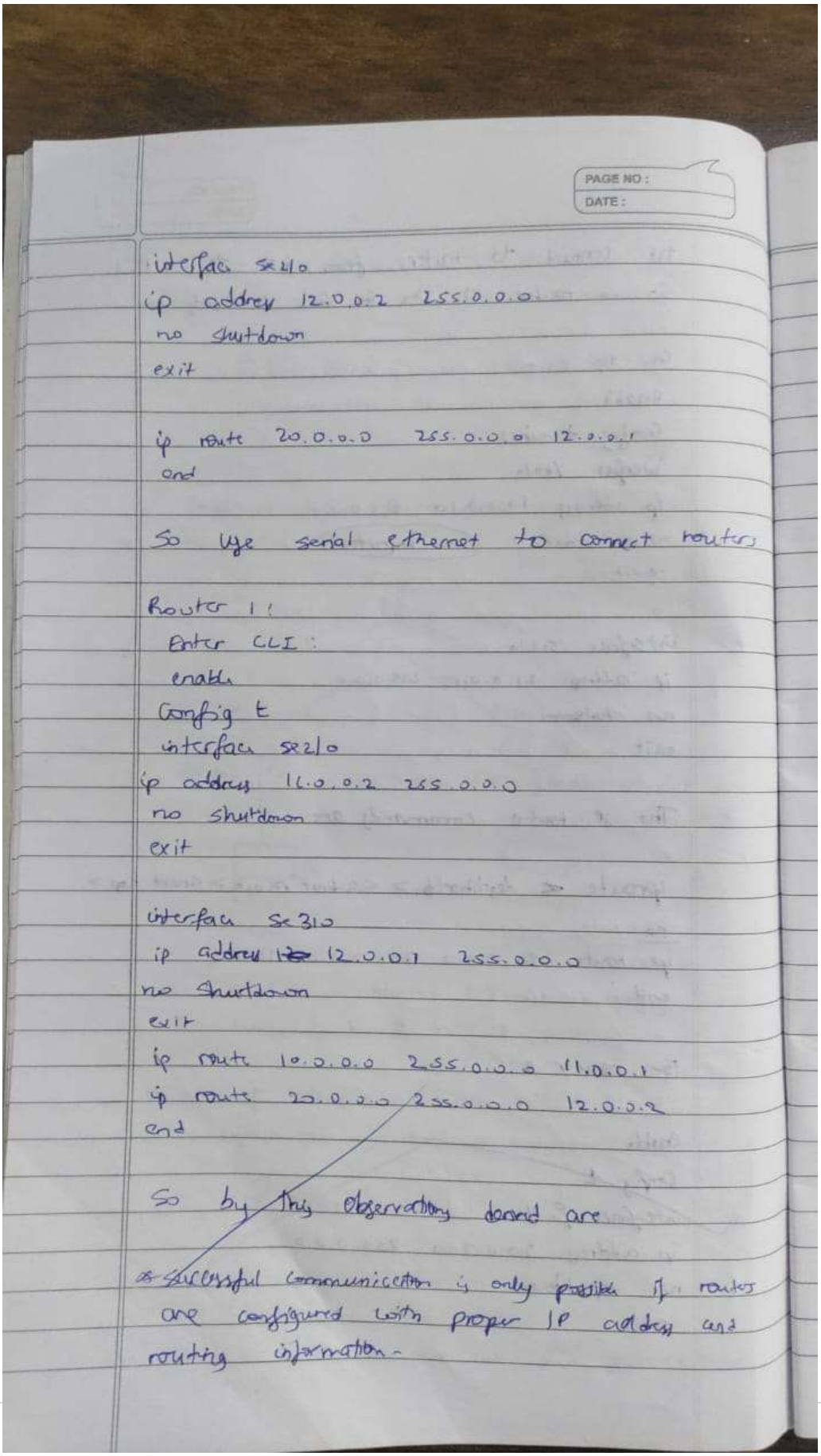
The ip route commands are

ip route <destination> <subnet mask> <next hop>

Ex:
ip route 20.0.0.0 255.0.0.0 11.0D.2
end

For router 2:

Enable
Config t
interface fa0/0
ip address 20.0.0.10 255.0.0.0
no shutdown
exit



PAGE NO :
DATE :

interface se2/0
ip address 12.0.0.2 255.0.0.0
no shutdown
exit

ip route 20.0.0.0 255.0.0.0 12.0.0.1
end

So Use serial ethernet to connect routers

Router 1 :
Enter CLI :
enable
Config t
interface se2/0
ip address 11.0.0.2 255.0.0.0
no shutdown
exit

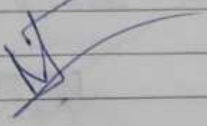
interface se3/0
ip address 12.0.0.1 255.0.0.0
no shutdown
exit
ip route 10.0.0.0 255.0.0.0 11.0.0.1
ip route 20.0.0.0 255.0.0.0 12.0.0.2
end

So by this observation denied are

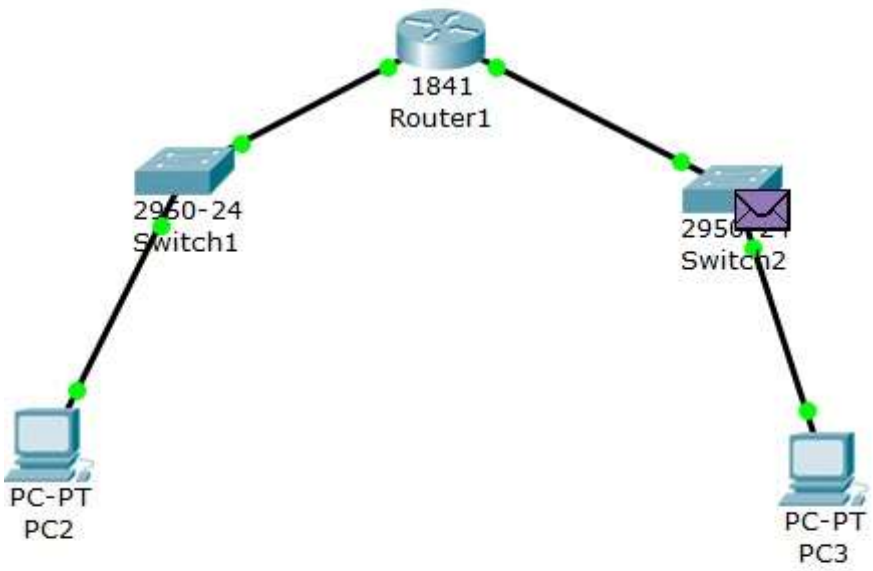
successful communication is only possible if routers are configured with proper IP address and routing information.

PAGE NO :
DATE :

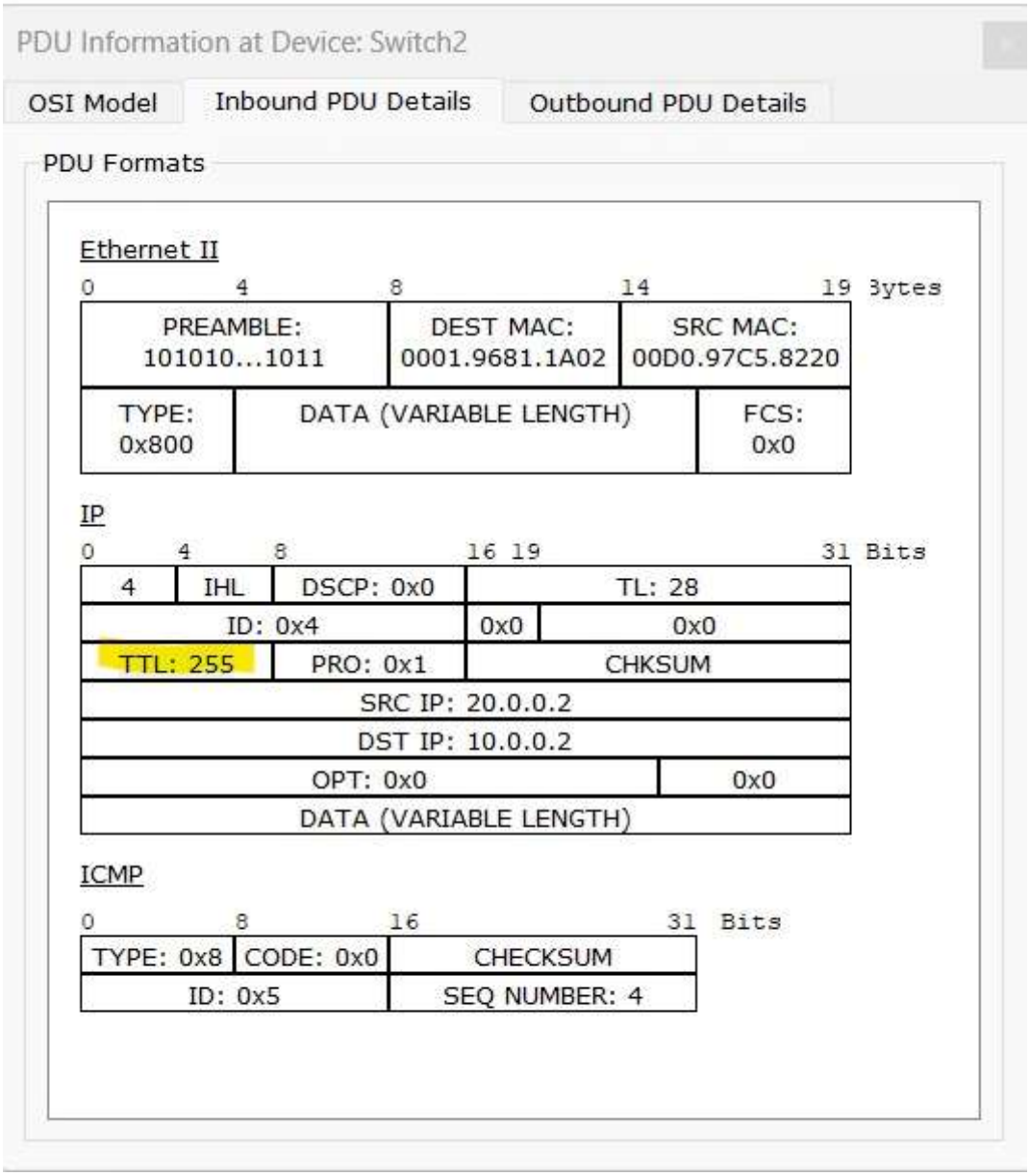
* without routing, packets to other networks fail.
* static and default routes enable successful communication across networks.

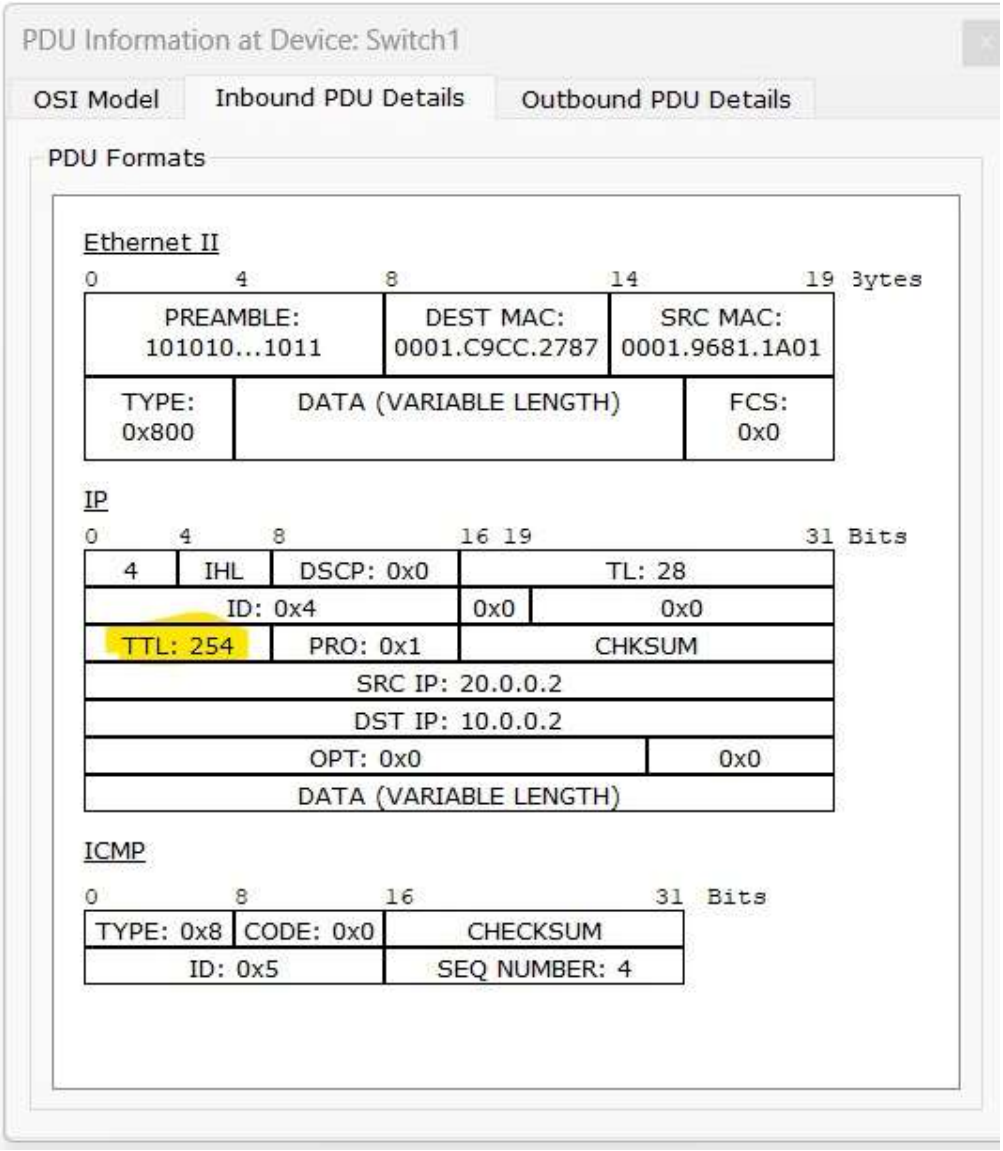


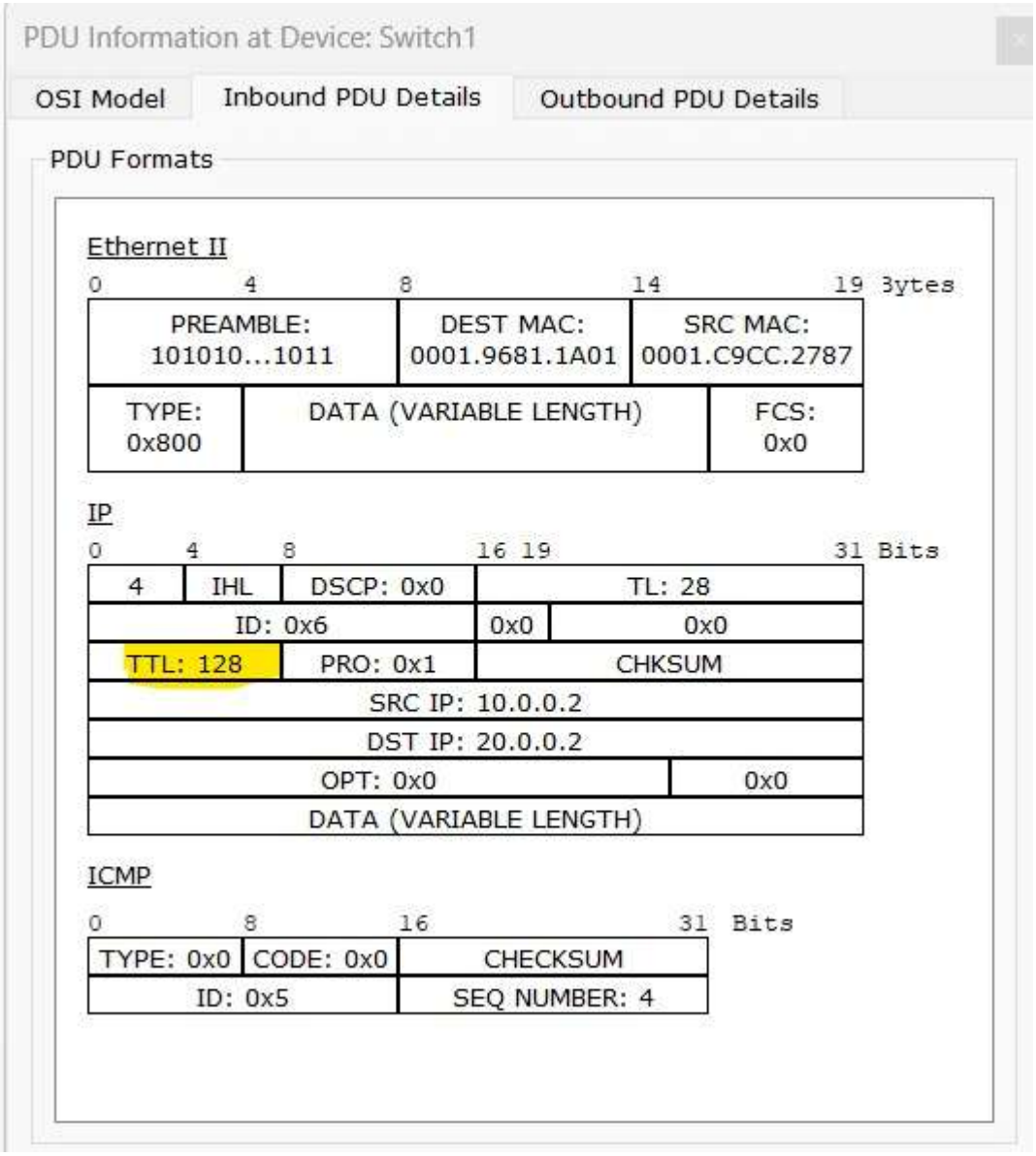
Topology:

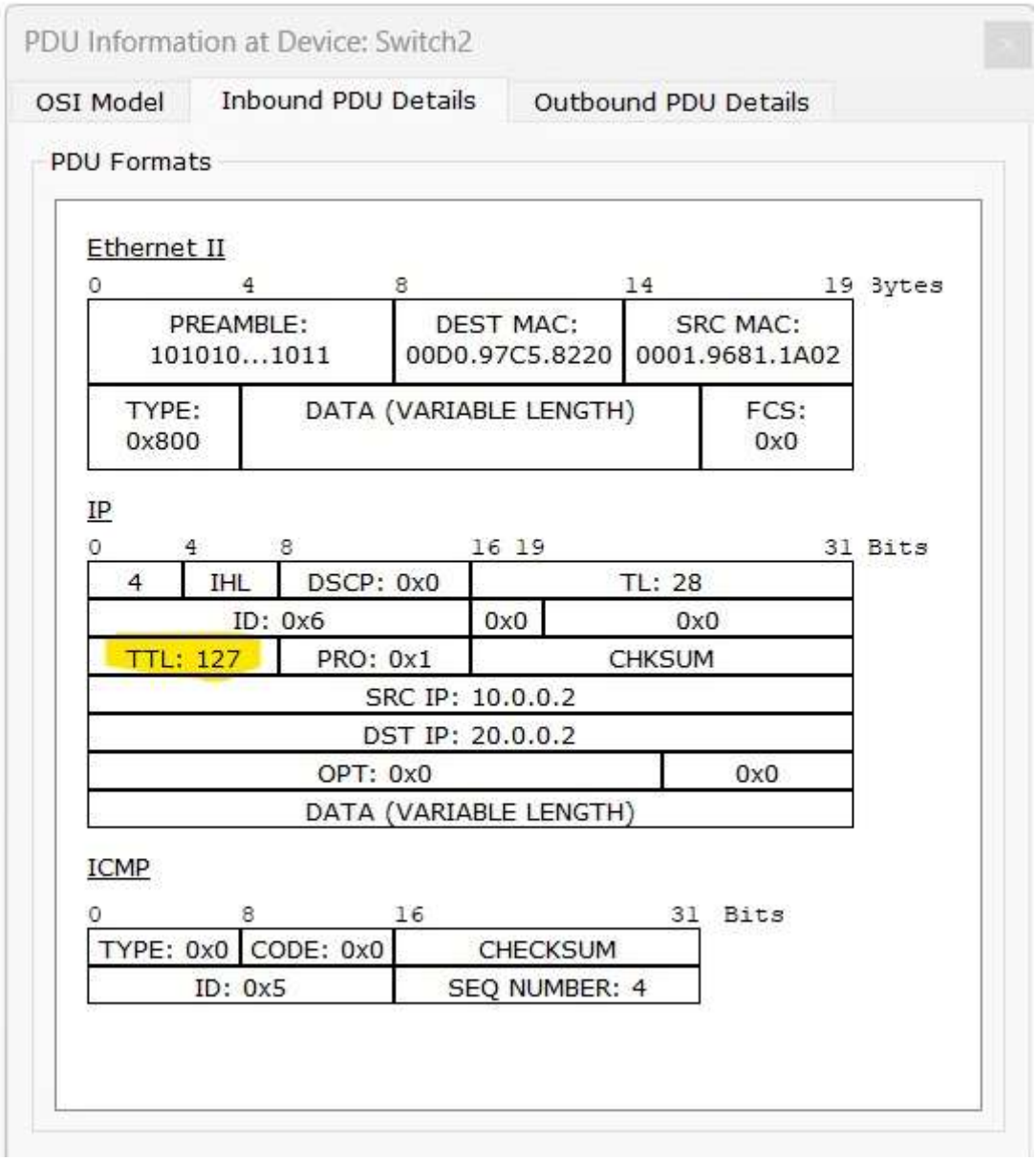


Output


PDU Information at Device: Switch2
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
Ethernet II
0 4 8 14 19 Bytes
PREAMBLE: 101010...1011
DEST MAC: 0001.9681.1A02
SRC MAC: 00D0.97C5.8220
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: 20.0.0.2
DST IP: 10.0.0.2
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: Switch1
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
Ethernet II
0 4 8 14 19 Bytes
PREAMBLE: 101010...1011
DEST MAC: 0001.C9CC.2787
SRC MAC: 0001.9681.1A01
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: 254
PRO: 0x1
CHKSUM
SRC IP: 20.0.0.2
DST IP: 10.0.0.2
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: Switch1
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
Ethernet II
0 4 8 14 19 3ytes
PREAMBLE: 101010...1011
DEST MAC: 0001.9681.1A01
SRC MAC: 0001.C9CC.2787
TYPE: 0x800
DATA (VARIABLE LENGTH)
FCS: 0x0
IP
0 4 8 16 19 31 Bits
4
IHL
DSCP: 0x0
TL: 28
ID: 0x6
0x0
0x0
TTL: 128
PRO: 0x1
CHKSUM
SRC IP: 10.0.0.2
DST IP: 20.0.0.2
OPT: 0x0
0x0
DATA (VARIABLE LENGTH)
ICMP
0 8 16 31 Bits
TYPE: 0x0
CODE: 0x0
CHECKSUM
ID: 0x5
SEQ NUMBER: 4


PDU Information at Device: Switch2
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
Ethernet II
0 4 8 14 19 Bytes
PREAMBLE: 101010...1011
DEST MAC: 00D0.97C5.8220
SRC MAC: 0001.9681.1A02
TYPE: 0x800
DATA (VARIABLE LENGTH)
FCS: 0x0
IP
0 4 8 16 19 31 Bits
4
IHL
DSCP: 0x0
TL: 28
ID: 0x6
0x0
0x0
TTL: 127
PRO: 0x1
CHKSUM
SRC IP: 10.0.0.2
DST IP: 20.0.0.2
OPT: 0x0
0x0
DATA (VARIABLE LENGTH)
ICMP
0 8 16 31 Bits
TYPE: 0x0
CODE: 0x0
CHECKSUM
ID: 0x5
SEQ NUMBER: 4

Cycle 2:

Program 1:

Write a program for error detecting code using CRC (8-bits).

Observation

PAGE NO :
DATE :

Week 7

CRC - Cyclic Redundancy Check

Sender side

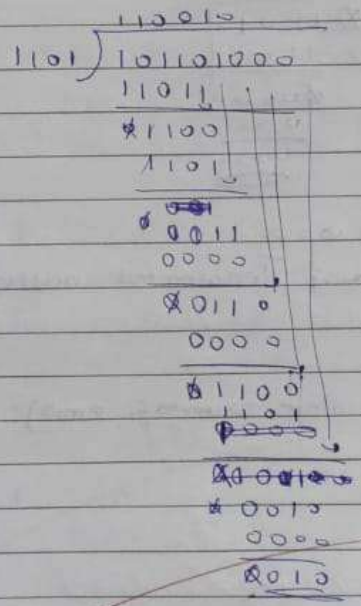
M(x) = 10110 1

g(x) = 1101

4

4-1 = 3

M(x) = 10110 1000



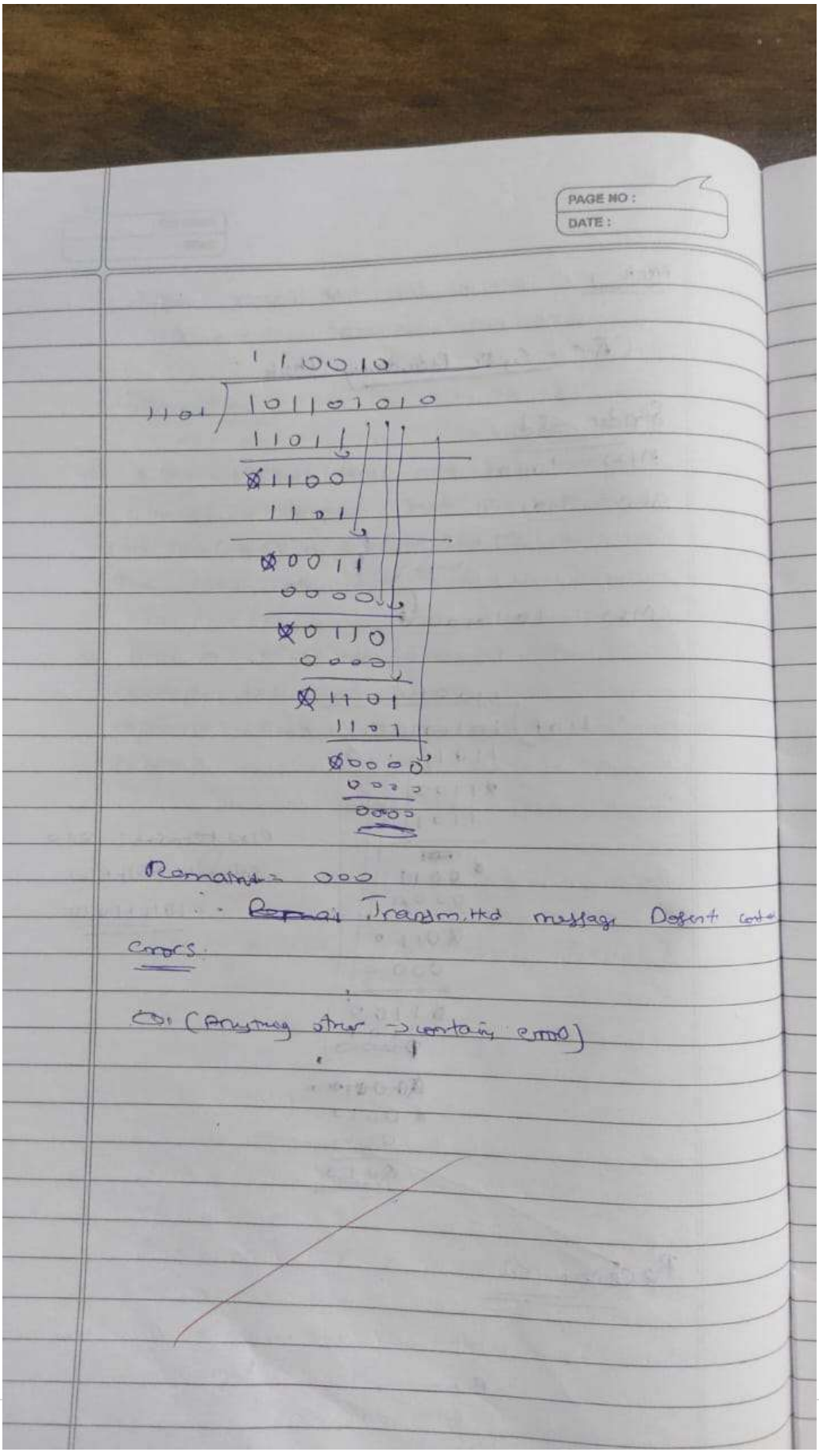
R(x) : remainder = 010

Codeword : M(x) + R(x)

= 10110 1010

Receiver side

R.x

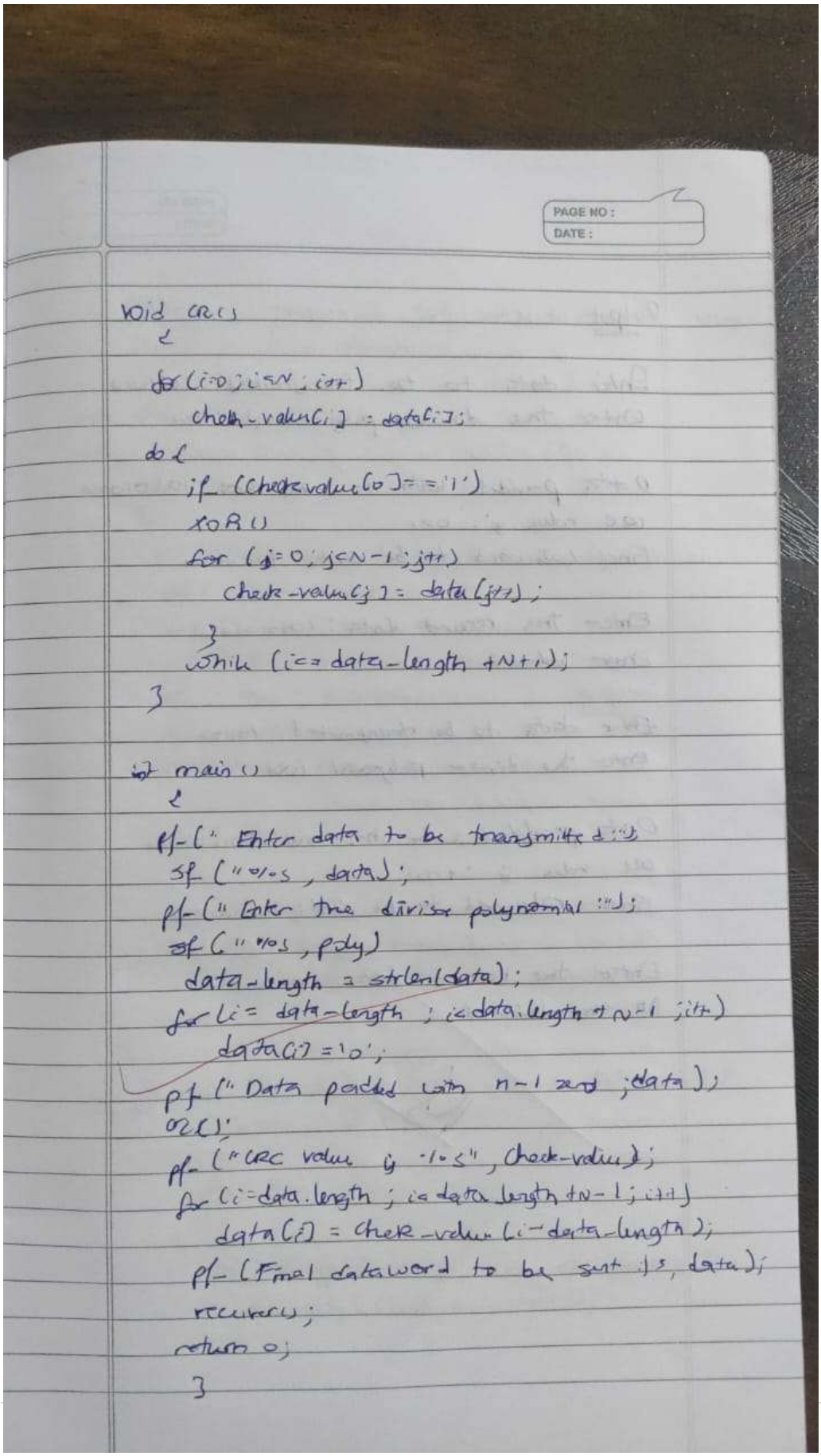

PAGE NO :
DATE :
Remainder = 000
Transmitted message doesn't contain errors.
(Any msg other → contain error)

PAGE NO :
DATE :

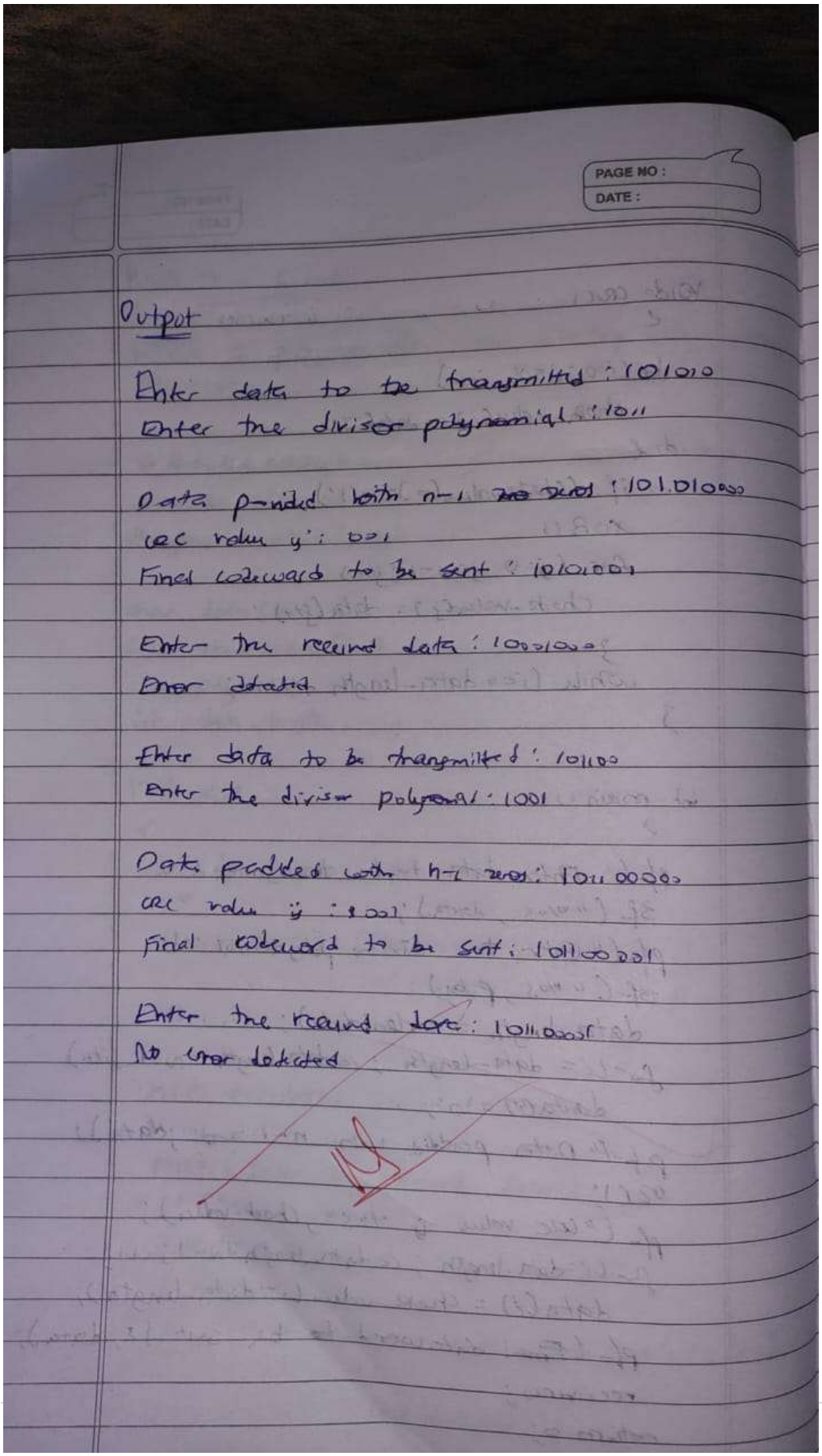
Week 7 - C. code

Write a program for error detecting code using CRC-CCITT

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

char data[30];
char check_value[30];
char poly[10];
int data_length, i, j;

void xor
{
 for (j=1; j<N; j++)
  {
   checkvalue[j] = ((check value[j] == poly[j])?
                    '0' : '1');
  }

void receiver()
{
 printf("Enter the received data");
scanf("%s", data)
 printf("Data received : %s", data);
 crc();
 for (i=0; (i<N-1) && (check_value[i] != '1'); i++)
     if (i<N-1)
         Pf ("Error detected");
     else
         Pf ("No error detected");
}
```



```
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] = data[j+1];
   }
   while (i<=data_length+N+1);
}

int main()
{
pf("Enter data to be transmitted:");
sf("%s", data);
pf("Enter the divisor polynomial:");
sf("%s", poly)
 data_length = strlen(data);
for(i=data_length; i<data.length+N-1; i++)
   data[i]='0';
pf("Data padded with n-1 zeros:%s", data);
crc();
pf("CRC value is -:%s", check_value);
for(i=data.length; i<data_length+N-1; i++)
   data[i] = check_value[i-data_length];
pf("Final data word to be sent is:%s", data);
receiver();
return 0;
}
```



PAGE NO :
DATE :

Output

Enter data to be transmitted : 101010
Enter the divisor polynomial : 1011

Data padded with n-1 zeros : 101.010000
CRC value y' : 001
Final codeword to be sent : 101.010001

Enter the received data : 1000.1000
Error detected

Enter data to be transmitted : 101100
Enter the divisor polynomial : 1001

Data padded with n-1 zeros : 101.100000
CRC value is : 1001
Final codeword to be sent : 101.100001

Enter the received data : 101.100001
No error detected

Code

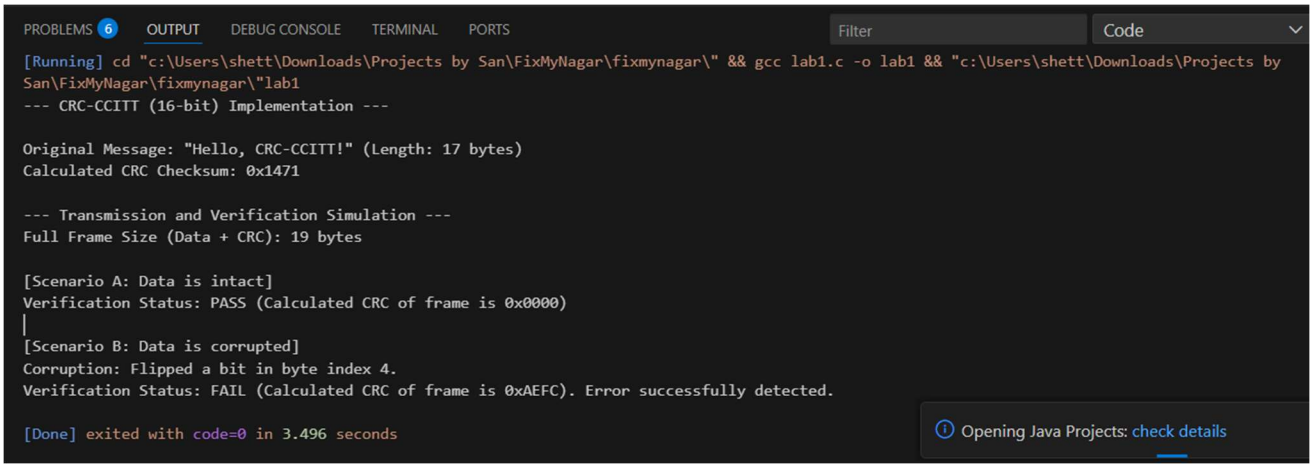
```
#include <stdio.h>
#include <stdint.h>
#include <string.h>

#define CRC_CCITT_POLY 0x1021
#define CRC_CCITT_INIT 0xFFFF

uint16_t calculate_crc16_ccitt(const unsigned char *data, size_t
length) {
    uint16_t crc = CRC_CCITT_INIT;
    size_t i, j;

    for (i = 0; i < length; i++) {
        crc ^= (uint16_t)(data[i] << 8);

        for (j = 0; j < 8; j++) {
            if (crc & 0x8000) {
                crc = (crc << 1) ^ CRC_CCITT_POLY;
            } else {
                crc = (crc << 1);
            }
            crc &= 0xFFFF;
        }
    }
```

```
    return crc;
}

int check_data_integrity(const unsigned char *data, size_t length) {
    uint16_t result = calculate_crc16_ccitt(data, length);
    return result == 0x0000;
}

int main() {
    const char *original_message = "Hello, CRC-CCITT!";
    size_t data_length = strlen(original_message);

    printf("--- CRC-CCITT (16-bit) Implementation ---\n\n");
    printf("Original Message: \"%s\" (Length: %zu bytes)\n",
original_message, data_length);

    uint16_t checksum = calculate_crc16_ccitt((const unsigned char
*)original_message, data_length);

    printf("Calculated CRC Checksum: 0x%04X\n\n", checksum);

    size_t frame_length = data_length + 2;
    unsigned char frame[frame_length];

    memcpy(frame, original_message, data_length);
```

```
    frame[data_length] = (unsigned char)(checksum >> 8);
    frame[data_length + 1] = (unsigned char)(checksum & 0xFF);

    printf("--- Transmission and Verification Simulation ---\n");
    printf("Full Frame Size (Data + CRC): %zu bytes\n",
frame_length);

    printf("\n[Scenario A: Data is intact]\n");
    int intact_check = check_data_integrity(frame, frame_length);

    if (intact_check) {
        printf("Verification Status: PASS (Calculated CRC of frame is
0x0000)\n");
    } else {
        printf("Verification Status: FAIL (Calculated CRC of frame is
0x%04X)\n", calculate_crc16_ccitt(frame, frame_length));
    }

    printf("\n[Scenario B: Data is corrupted]\n");
    size_t error_index = 4;
    frame[error_index] ^= 0x01;

    printf("Corruption: Flipped a bit in byte index %zu.\n",
error_index);

    int corrupted_check = check_data_integrity(frame,
frame_length);
```

```
    if (corrupted_check) {
        printf("Verification Status: FAIL (Calculated CRC of frame is
0x0000, but data was corrupted! - This is a rare, undetected error
case.)\n");
    } else {
        printf("Verification Status: FAIL (Calculated CRC of frame is
0x%04X). Error successfully detected.\n",
calculate_crc16_ccitt(frame, frame_length));
    }

    return 0;
}
```

Output



Program 2: Write a program for congestion control using Leaky bucket algorithm.

Observation

PAGE NO :
DATE :

Week 8

Leaky bucket

Max capacity of bucket = 4, departure = 1

Time	Buffer before	Arrival	Buffer after capture	Dropped	Departure	Buffer end
0	0	3	3	0	1	2
1	2	2	4	0	1	3
2	3	1	4	0	1	3
3	3	4	7	3	1	3
4	3	0	3	0	1	2
5	2	0	2	0	1	1
6	1	0	1	0	1	0

PAGE NO :
DATE :

Write a program for congestion control using leaky bucket algorithm

→
```
#include<stdio.h>

int main()
{
  int incoming, outgoing, buck_siz, n, store=0;
  pf("Enter bucket size ");
  sf("%d", &bucket_siz);
  pf("Enter outgoing size:");
  sf("%d", &outgoing);
  pf("Enter no of inputs");
  sf("%d", &n);

  while(n!=0)
  {
   pf("Enter the incoming bucket size");
   sf("%d", &incoming);
   if(incoming <= (buck_siz - store))
   {
    store += incoming;
   pf("Bucket buffer size %d out of %d",
    store, buck_siz);
   }
   else
   {
   pf("Dropped %d no of packets",
    incoming - (buck_siz - store));
```

```
pf("Bucket buffer size %d out of %d", store,
    buck_size);
    }
    store = store - outgoing;
    pf("After outgoing %d packets
    left out of %d in buffer \n", store, buck_size);
    n--;
    }
}
```

Output

```
Enter bucket size: 5000
Enter outgoing rate: 2000
Enter num of ipts: 2
Enter the incoming packet size: 3000
Bucket buffer size 3000 out of 5000
After outgoing 1000 packets left out of 5000
in buffer
Enter the incoming packet size: 1000
Bucket buffer size 2000 out of 5000
After outgoing 0 packets left out of 5000 in
buffer.
```

Code

```
#include <stdio.h>
#include <stdint.h>
#include <string.h>

// Constants for the Leaky Bucket Algorithm
#define BUCKET_CAPACITY 1000 // Max capacity in units (e.g., bytes or packets)
#define OUTPUT_RATE     100  // Fixed output rate in units/tick
#define SIMULATION_TICKS 10

// Simulated packet stream (units arriving at each time tick)
// This simulates bursty traffic
const int INCOMING_PACKETS[SIMULATION_TICKS] = {
    150, 400, 50, 600, 300, 10, 500, 50, 100, 200
};

void leaky_bucket_simulation() {
    int bucket_level = 0; // Current level of the bucket (initially empty)
    int total_attempted = 0;
    int total_accepted = 0;
    int total_dropped = 0;
    int total_sent = 0;
    int tick;
```

```
    printf("--- Leaky Bucket Congestion Control Simulation ---
\n\n");
    printf("Configuration:\n");
    printf("  Bucket Capacity: %d units\n", BUCKET_CAPACITY);
    printf("  Output Rate (Leak Rate): %d units/tick\n",
OUTPUT_RATE);
    printf("  Total Ticks: %d\n", SIMULATION_TICKS);
    printf("----------------------------------------------------------------
\n");
    printf("| Tick | Incoming | Accepted | Dropped | Current Level |
Sent Out |\n");
    printf("----------------------------------------------------------------
\n");

    for (tick = 0; tick < SIMULATION_TICKS; tick++) {
        int incoming_packet = INCOMING_PACKETS[tick];
        int accepted_packet = 0;
        int dropped_packet = 0;
        int sent_out = 0;
        total_attempted += incoming_packet;

        // --- 1. Handle Incoming Packet (Check for Overflow) ---
        int space_available = BUCKET_CAPACITY - bucket_level;

        if (incoming_packet <= space_available) {
            // Packet fits entirely
```

```
            accepted_packet = incoming_packet;
            bucket_level += accepted_packet;
        } else {
            // Bucket overflow: accept only what fits, drop the rest.
            accepted_packet = space_available;
            dropped_packet = incoming_packet - accepted_packet;
            bucket_level = BUCKET_CAPACITY; // Bucket is now
full
        }

        total_accepted += accepted_packet;
        total_dropped += dropped_packet;

        // --- 2. Drain the Bucket (Fixed Output Rate) ---
        if (bucket_level > 0) {
            // Send out at most the OUTPUT_RATE or whatever is left
in the bucket
            if (bucket_level >= OUTPUT_RATE) {
                sent_out = OUTPUT_RATE;
                bucket_level -= OUTPUT_RATE;
            } else {
                sent_out = bucket_level;
                bucket_level = 0; // Bucket becomes empty
            }
            total_sent += sent_out;
        }
```

```
        // --- 3. Report Status ---
        printf("| %4d | %8d | %8d | %7d | %13d | %8d |\n",
               tick + 1,
               incoming_packet,
               accepted_packet,
               dropped_packet,
               bucket_level,
               sent_out);
    }
    printf("----------------------------------------------------------------
\n");
    printf("\nSimulation Summary:\n");
    printf("Total Attempted Input: %d\n", total_attempted);
    printf("Total Accepted: %d\n", total_accepted);
    printf("Total Dropped: %d\n", total_dropped);
    printf("Total Sent (Smoothed Output): %d\n", total_sent);
    printf("Final Bucket Level: %d\n", bucket_level);
}

int main() {
    leaky_bucket_simulation();
    return 0;
}
```

Output

```
[Running] cd "c:\Users\shett\Downloads\Projects by San\FixMyNagar\fixmynagar\" && gcc lab1.c -o lab1 && "c:\Users\shett\Downloads\Projects by
San\FixMyNagar\fixmynagar\"lab1
--- Leaky Bucket Congestion Control Simulation ---

Configuration:
  Bucket Capacity: 1000 units
  Output Rate (Leak Rate): 100 units/tick
  Total Ticks: 10
-----------------------------------------------------------------
| Tick | Incoming | Accepted | Dropped | Current Level | Sent Out |
-----------------------------------------------------------------
|    1 |      150 |      150 |       0 |            50 |      100 |
|    2 |      400 |      400 |       0 |           350 |      100 |
|    3 |       50 |       50 |       0 |           300 |      100 |
|    4 |      600 |      600 |       0 |           800 |      100 |
|    5 |      300 |      200 |     100 |           900 |      100 |
|    6 |       10 |       10 |       0 |           810 |      100 |
|    7 |      500 |      190 |     310 |           900 |      100 |
|    8 |       50 |       50 |       0 |           850 |      100 |
|    9 |      100 |      100 |       0 |           850 |      100 |
|   10 |      200 |      150 |      50 |           900 |      100 |
-----------------------------------------------------------------

Simulation Summary:
Total Attempted Input: 2360
Total Accepted: 1900
Total Dropped: 460
Total Sent (Smoothed Output): 1000
Final Bucket Level: 900

[Done] exited with code=0 in 0.577 seconds
```

Program 3: 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.

Observation:

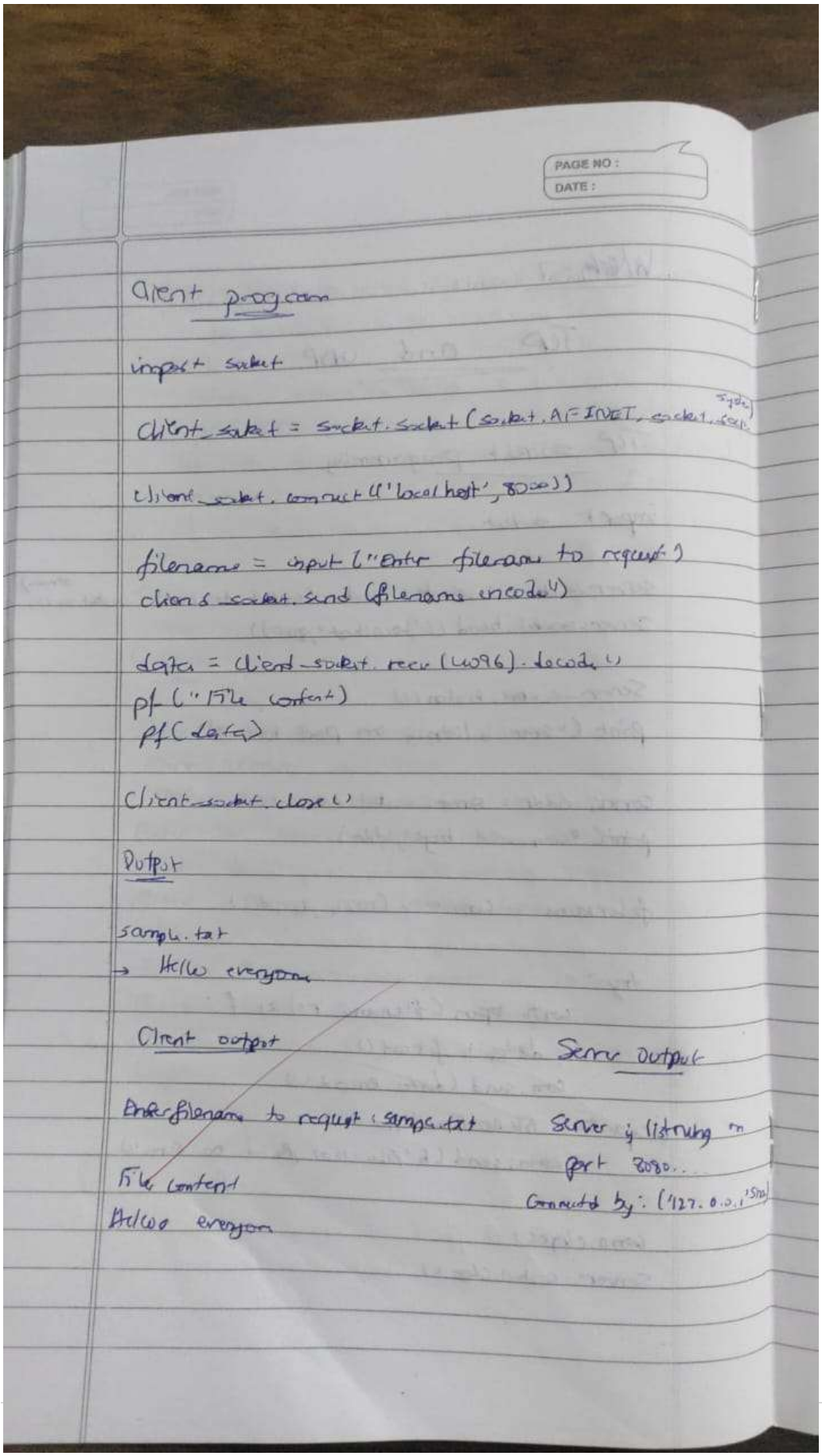
PAGE NO :
DATE :

Week-7

TCP and UDP

TCP Socket Programming

```
import socket

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('localhost', 8080))

server_socket.listen(1)
print("Server is listening on port 8080...")

conn, addr = server_socket.accept()
print("Connected by:", addr)

filename = conn.recv(1024).decode()

try:
    with open(filename, 'r') as f:
        data = f.read()
    conn.send(data.encode())
except FileNotFoundError:
    conn.send(b"File not found on server")

conn.close()
server_socket.close()
```



PAGE NO :
DATE :

Client program

import socket

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

client_socket.connect(('localhost', 8080))

filename = input("Enter filename to request")
client_socket.send(filename.encode())

data = client_socket.recv(4096).decode()
print("File content")
print(data)

client_socket.close()

Output

sample.txt
→ Hello everyone

Client output

Enter filename to request : sample.txt
File content
Hello everyone

Server Output

Server is listening on port 8080...
Connected by: ('127.0.0.1', 5000)

Code

File Server

```
import socket
import os
import sys

PORT = 8080
HOST = '127.0.0.1'
BUFFER_SIZE = 1024
ENCODING = 'utf-8'

def handle_client(conn, addr):
    """Handles the file request from a single client connection."""
    print(f"Connection established with {addr[0]}:{addr[1]}")

    try:
        # 1. Receive filename from client
        data = conn.recv(BUFFER_SIZE)
        if not data:
            print("Client disconnected.")
            return

        filename = data.decode(ENCODING).strip()
        print(f"Client requested file: '{filename}'")

        # 2. Attempt to check and open the file
        if not os.path.exists(filename) or not os.path.isfile(filename):
```

```
            # File not found: Send error header
            error_msg = "ERROR: File Not Found."
            conn.sendall(error_msg.encode(ENCODING))
            print(f"Sent {error_msg}")
            return

        # File found:
        file_size = os.path.getsize(filename)
        print(f"File found (Size: {file_size} bytes). Sending
content...")

        # Send a success header (SUCCESS:<size>)
        success_msg = f"SUCCESS:{file_size}"
        conn.sendall(success_msg.encode(ENCODING))

        # 3. Read and send file contents chunk by chunk
        with open(filename, 'rb') as f:
            while True:
                bytes_read = f.read(BUFFER_SIZE)
                if not bytes_read:
                    # End of file
                    break
                conn.sendall(bytes_read)

        print("File content transmission complete.")

    except Exception as e:
```

```
        print(f"An error occurred while handling client {addr}: {e}")
    finally:
        # 4. Close the connection
        conn.close()

def main():
    """Main function to set up and run the server."""
    # Create socket file descriptor
    try:
        server_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
        # Allows reuse of the port immediately after closing the server
        server_socket.setsockopt(socket.SOL_SOCKET,
socket.SO_REUSEADDR, 1)
        server_socket.bind((HOST, PORT))
        server_socket.listen(5)
        print(f"File Server listening on {HOST}:{PORT}...")
    except socket.error as e:
        print(f"Failed to set up server: {e}")
        sys.exit(1)

    while True:
        print("\nWaiting for a new client connection...")
        try:
            # Accept new connection
            conn, addr = server_socket.accept()
            handle_client(conn, addr)
```

```
        except KeyboardInterrupt:
            print("\nServer shutting down.")
            break
        except Exception as e:
            print(f"Accept failed: {e}")
            continue

    server_socket.close()

if __name__ == "__main__":
    main()
```

File Client

```
import socket
import sys

PORT = 8080
HOST = '127.0.0.1'
BUFFER_SIZE = 1024
ENCODING = 'utf-8'

def main():
    if len(sys.argv) != 2:
        print(f"Usage: python3 {sys.argv[0]} <filename>")
        sys.exit(1)
```

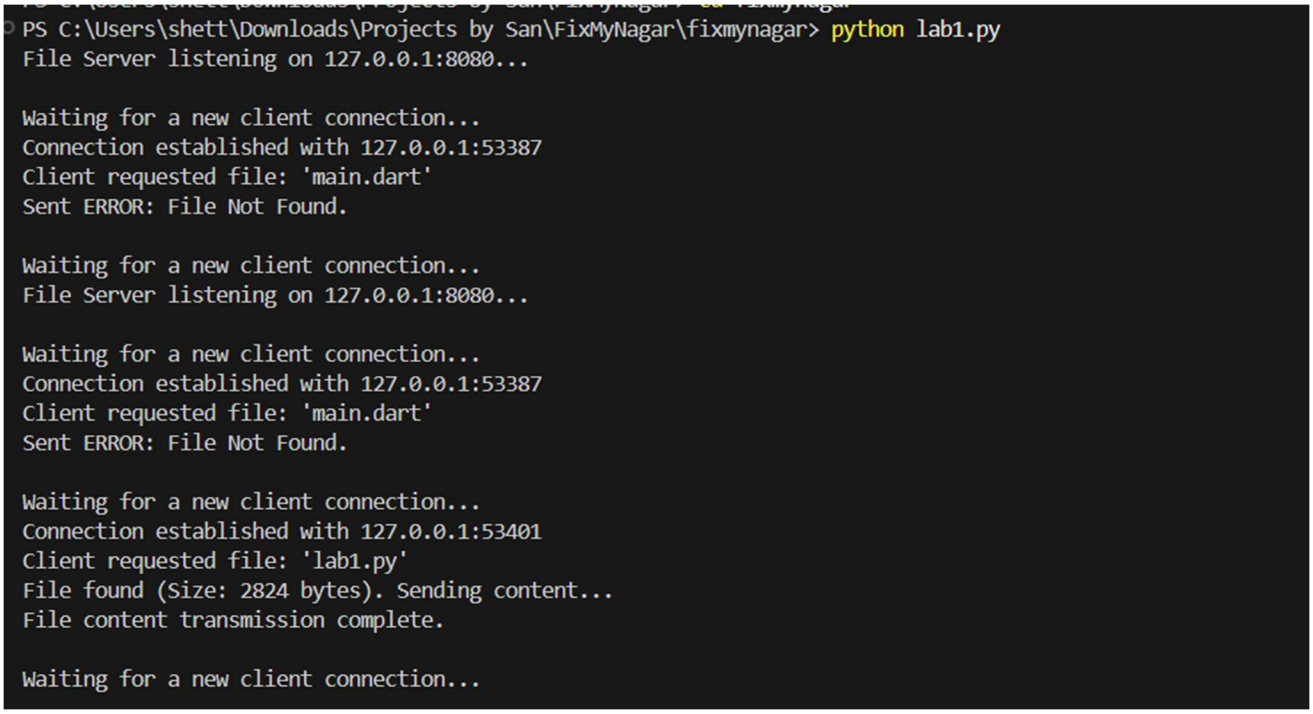
```
    filename = sys.argv[1]

    print(f"Attempting to request file: '{filename}' from
{HOST}:{PORT}")

    # 1. Create socket file descriptor
    try:
        client_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
        client_socket.connect((HOST, PORT))
        print("Connection established. Sending filename...")
    except socket.error as e:
        print(f"\nConnection Failed: {e}")
        sys.exit(1)

    try:
        # 2. Send the filename
        client_socket.sendall(filename.encode(ENCODING))

        # 3. Receive the initial response/header
        response_header = b''
        # Read the first chunk, which should contain the header
        chunk = client_socket.recv(BUFFER_SIZE)
        if not chunk:
            print("Server closed connection without response.")
            return
```

```
        response_header = chunk.decode(ENCODING)

        print("\n--- Server Response ---")

        # Check for ERROR response
        if response_header.startswith("ERROR:"):
            print(f"Error: {response_header.split(':', 1)[1].strip()}")

        # Check for SUCCESS header and print content
        elif response_header.startswith("SUCCESS:"):
            parts = response_header.split(':', 1)
            if len(parts) < 2:
                print("Invalid SUCCESS header received.")
                return

            expected_size = int(parts[1])
            print(f"File transfer initiated (Expected size:
{expected_size} bytes).")
            print("File Contents:")
            print("==========================================
")

            # NOTE: For simplicity, the Python server sends the header
and then the content immediately.
            # This logic assumes the header contains no part of the file,
which is a simplification.
```

```
            # Since the header is small, the next `recv` should get the
actual content.

            total_received = 0

            # Loop to receive the rest of the file chunks
            while True:
                data_chunk = client_socket.recv(BUFFER_SIZE)
                if not data_chunk:
                    break

                # We assume the content is primarily text for printing
                print(data_chunk.decode(ENCODING), end='')
                total_received += len(data_chunk)

            print("\n==================================================
==")
            print(f"Transmission complete. Total received data:
{total_received} bytes.")

        else:
            # Unexpected response, just print raw text
            print("Unexpected response received.")
            print("==========================================
")
            print(response_header)
```

```
            print("==========================================
")

    except Exception as e:
        print(f"An error occurred during communication: {e}")
    finally:
        # 4. Close the socket
        client_socket.close()

if __name__ == "__main__":
    main()
```

Output



```
PS C:\Users\shett\Downloads\Projects by San\FixMyNagar\fixmynagar> python lab2.py text.txt
Attempting to request file: 'text.txt' from 127.0.0.1:8080
Connection established. Sending filename...

--- Server Response ---
File transfer initiated (Expected size: 19 bytes).
File Contents:
====================================
hellooooo yaayyyyyy
====================================
Transmission complete. Total received data: 19 bytes.
PS C:\Users\shett\Downloads\Projects by San\FixMyNagar\fixmynagar>
```

<u>Program 4</u>: 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.

Observation

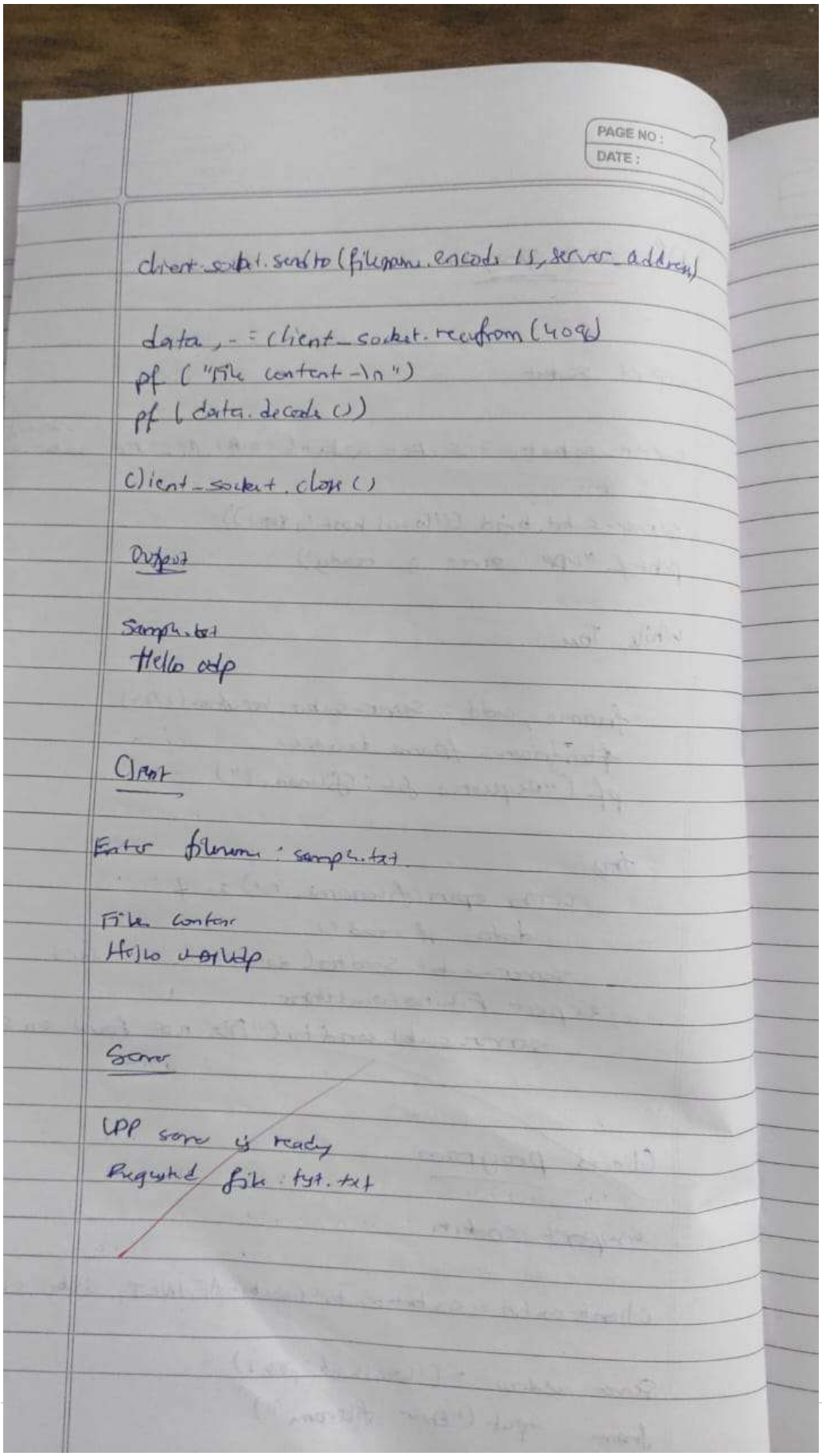
PAGE NO :
DATE :

UDP socket Program

```
import socket

server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

server_socket.bind(('localhost', 8021))
print("UDP server is ready")

while True:

    fname, addr = server_socket.recvfrom(1024)
    filename = fname.decode()
    print("Requested file: {filename}")

    try:
        with open(filename, 'r') as f:
            data = f.read()
        server_socket.sendto(data.encode(), addr)
    except FileNotFoundError:
        server_socket.sendto("File not found on server", addr)
```

Client program

```
import socket

client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

server_address = ('localhost', 8021)
fname = input("Enter filename")
```



client_socket.sendto(filename.encode(), server_address)

data, _ = client_socket.recvfrom(4096)
pf("File content:\n")
pf(data.decode())

client_socket.close()

Output

Sample.txt
Hello udp

Client

Enter filename: sample.txt

File content:
Hello udp

Server

UDP server is ready
Requested file: tyt.txt

Code
Server

```
import socket
import os
import sys

PORT = 8080
HOST = '127.0.0.1'
BUFFER_SIZE = 1024 # Max size to receive (filename)
CHUNK_SIZE = 1000  # Size of file data per packet
ENCODING = 'utf-8'
SEQUENCE_HEADER_LEN = 5 # 4 digits for seq num + 1 colon
(e.g., "0001:")

def main():
    """Main function to set up and run the UDP server."""
    try:
        # Create UDP socket
        server_socket = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
        server_socket.bind((HOST, PORT))
        print(f"UDP File Server listening on {HOST}:{PORT}...")
    except socket.error as e:
        print(f"Failed to set up server: {e}")
        sys.exit(1)
```

```
    while True:
        print("\nWaiting for client request (filename)...")

        try:
            # 1. Receive filename and client address
            data, client_address =
server_socket.recvfrom(BUFFER_SIZE)
            filename = data.decode(ENCODING).strip()
            print(f"Received request from
{client_address[0]}:{client_address[1]} for file: '{filename}'")

            # 2. Attempt to check and open the file
            if not os.path.exists(filename) or not
os.path.isfile(filename):
                # File not found: Send error message
                error_msg = "ERROR: File Not Found."
                server_socket.sendto(error_msg.encode(ENCODING),
client_address)
                print(f"Sent {error_msg}")
                continue

            # File found:
            file_size = os.path.getsize(filename)
            print(f"File found (Size: {file_size} bytes). Sending content
in chunks...")

            sequence_number = 1
```

```
            # 3. Read file in binary mode and send chunks
            with open(filename, 'rb') as f:
                while True:
                    bytes_read = f.read(CHUNK_SIZE)
                    if not bytes_read:
                        break # End of file

                    # Construct the packet: [Seq Num (4 digits)]:[Data]
                    seq_header =
f"{sequence_number:04d}:".encode(ENCODING)
                    packet_data = seq_header + bytes_read

                    server_socket.sendto(packet_data, client_address)
                    sequence_number += 1

            # 4. Send the completion marker
            completion_msg =
f"{sequence_number:04d}:FILE_TRANSFER_COMPLETE".enco
de(ENCODING)
            server_socket.sendto(completion_msg, client_address)

            print(f"Transmission complete. Total chunks sent:
{sequence_number - 1}")

        except KeyboardInterrupt:
            print("\nServer shutting down.")
```

```
            break
        except Exception as e:
            print(f"An error occurred: {e}")
            continue

    server_socket.close()

if __name__ == "__main__":
    main()
```

Client

```
import socket
import sys

PORT = 8080
HOST = '127.0.0.1'
BUFFER_SIZE = 1500 # Slightly larger than the expected max
packet size (CHUNK_SIZE + headers)
ENCODING = 'utf-8'

def main():
    if len(sys.argv) != 2:
        print(f"Usage: python3 {sys.argv[0]} <filename>")
        sys.exit(1)

    filename = sys.argv[1]
```

```
    # Dictionary to store received chunks {seq_num: data_bytes}
    received_chunks = {}
    last_seq_num = 0

    print(f"Attempting to request file: '{filename}' from
{HOST}:{PORT}")

    # 1. Create UDP socket
    try:
        client_socket = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
        # Set a timeout for receiving data (UDP is unreliable, so we
don't wait forever)
        client_socket.settimeout(5.0)
    except socket.error as e:
        print(f"\nSocket creation error: {e}")
        sys.exit(1)

    try:
        server_address = (HOST, PORT)

        # 2. Send the filename request
        client_socket.sendto(filename.encode(ENCODING),
server_address)
        print("Filename sent. Waiting for file chunks...")
```

```
        # 3. Receive chunks
        while True:
            try:
                data, server = client_socket.recvfrom(BUFFER_SIZE)

                # Check for ERROR response (no sequence number)
                if data.decode(ENCODING).startswith("ERROR:"):
                    print("\n--- Server Response ---")
                    print(f"Error: {data.decode(ENCODING).split(':',
1)[1].strip()}")
                    return

                # Split packet into header and payload
                header_end = data.find(b':')
                if header_end == -1:
                    print(f"Warning: Received malformed packet.
Skipping.")
                    continue

                seq_num_str = data[:header_end].decode(ENCODING)
                payload = data[header_end + 1:]

                try:
                    seq_num = int(seq_num_str)
                except ValueError:
                    print(f"Warning: Invalid sequence number
'{seq_num_str}'. Skipping.")
```

```
                continue

            # Check for completion marker
            if payload.decode(ENCODING).strip() ==
"FILE_TRANSFER_COMPLETE":
                last_seq_num = seq_num - 1 # The sequence number
of the last data packet
                print(f"Received completion marker. Total data
packets expected: {last_seq_num}")
                break

            # Store the chunk
            if seq_num not in received_chunks:
                received_chunks[seq_num] = payload

        except socket.timeout:
            print("Socket timeout reached. Assuming transmission
complete or lost packets.")
            # We break here because UDP doesn't guarantee
delivery; we take what we got.
            break

    # 4. Reassemble and Print Content
    print("\n--- Reassembly & Output ---")
    if not received_chunks:
        print("No data received or file was empty.")
        return
```

```
        # Sort chunks by sequence number
        sorted_keys = sorted(received_chunks.keys())
        total_received_bytes = 0

        print("File Contents:")
        print("==========================================")

        # Concatenate and print (assuming text content)
        for seq_num in sorted_keys:
            chunk = received_chunks[seq_num]
            print(chunk.decode(ENCODING), end='')
            total_received_bytes += len(chunk)

        # Check for missing packets (for debugging/diagnostic)
        missing_packets = set(range(1, last_seq_num + 1)) -
set(received_chunks.keys())
        if missing_packets:
            print(f"\n\nWARNING: Missing packets detected:
{len(missing_packets)} packet(s).")
            # print(f"Missing sequence numbers:
{sorted(list(missing_packets))}") # Uncomment for verbose error

        print("\n==========================================
")
        print(f"Transmission complete. Total received data:
{total_received_bytes} bytes.")
```

```
    except Exception as e:
        print(f"An error occurred during communication: {e}")
    finally:
        # 5. Close the socket
        client_socket.close()

if __name__ == "__main__":
    main()
```

Output

```
PS C:\Users\shett\Downloads\Projects by San\FixMyNagar> cd fixmynagar
PS C:\Users\shett\Downloads\Projects by San\FixMyNagar\fixmynagar> python lab2.py text.txt
Attempting to request file: 'text.txt' from 127.0.0.1:8080
Filename sent. Waiting for file chunks...
Received completion marker. Total data packets expected: 1

--- Reassembly & Output ---
File Contents:
====================================
hellooooo yaayyyyyy
====================================
Transmission complete. Total received data: 19 bytes.
PS C:\Users\shett\Downloads\Projects by San\FixMyNagar\fixmynagar>
```

```
PS C:\Users\shett\Downloads\Projects by San\FixMyNagar> cd fixmynagar
PS C:\Users\shett\Downloads\Projects by San\FixMyNagar\fixmynagar> python lab1.py
UDP File Server listening on 127.0.0.1:8080...

Waiting for client request (filename)...
Received request from 127.0.0.1:60310 for file: 'text.txt'
File found (Size: 19 bytes). Sending content in chunks...
Transmission complete. Total chunks sent: 1

Waiting for client request (filename)...
```