

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
FACULTY OF SCIENCE AND HUMANITIES
DEPARTMENT OF COMPUTER APPLICATIONS



PRACTICAL RECORD NOTE

STUDENT NAME :

REGISTER NUMBER :

CLASS : MCA SECTION : G

YEAR & SEMESTER : I YEAR & II SEM

SUBJECT CODE : PCA20C05J

SUBJECT TITLE : COMPUTER NETWORKS

APRIL 2024



SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
FACULTY OF SCIENCE AND HUMANITIES
DEPARTMENT OF COMPUTER APPLICATIONS

SRM Nagar, Kattankulathur – 603 203

CERTIFICATE

Certified to be the bonafide record of practical work done by

Register No. _____ of _____ MCA _____ Degree

course for PCA20C05J – COMPUTER NETWORKS in the Computer lab in SRM

Institute of Science and Technology during the academic year 2023-2024.

Staff In-charge

Head of the Department

Submitted for Semester Practical Examination held on _____.

Internal Examiner

External Examiner

INDEX

S.NO	DATE	TITLE	PAGE. NO	STAFF SIGN
1	03.01.24	FAMILIARIZING WITH WINDOW NETWORK COMMANDS	1	
2	08.01.24	ANALYZING THE PERFORMANCE OF VARIOUS CONFIGURATIONS AND PROTOCOLS OF LAN ESTABLISHING A LOCAL AREA NETWORK (LAN)	7	
3	18.01.24	ANALYZING THE PERFORMANCE OF VARIOUS CONFIGURATIONS AND PROTOCOLS IN LAN CONNECTING TWO LANs USING ROUTER WITH STATIC ROUTER	11	
4	22.01.24	ANALYZING THE PERFORMANCE OF VARIOUS CONFIGURATIONS AND PROTOCOLS IN LAN MULTI-ROUTING CONNECTION	17	
5	30.01.24	CONNECTING TWO LANs USING BRIDGE	23	
6	06.02.24	DESIGNING RING AND MESH TOPOLOGIES USING CISCO PACKET TRACER	27	
7	09.02.24	DESIGNING BUS AND STAR TOPOLOGIES USING CISCO PACKET TRACER	34	
8	13.02.24	DESIGNING HYBRID TOPOLOGIES USING CISCO PACKET TRACER	41	
9	16.02.24	IMPLEMENTING ERROR DETECTING CODE USING PARITY CHECK	54	
10	20.02.24	IMPLEMENTING ERROR DETECTING CODE USING CHECKSUM	57	
11	23.02.24	IMPLEMENTING ERROR DETECTING CODE USING CRC – CCITT	62	
12	01.03.24	IMPLEMENTATION OF GO BACK N PROTOCOL	66	

13	05.03.24	IMPLEMENTATION OF STOP AND WAIT PROTOCOL	69	
14	08.03.24	IMPLEMENTATION OF SELECTIVE REPEAT PROTOCOL	73	
15	12.03.24	IMPLEMENTATION OF WEB PROGRAMMING USING HTML	76	

EX.NO : 01

DATE : 03.01.24

FAMILIARIZING WITH WINDOWS NETWORK COMMANDS

AIM :

To familiarize with windows network commands and their outputs.

PROCEDURE :

1. Open the Command prompt by typing “CMD” in the Run Dialogue.
2. Once the Command prompt opens type the commands.

COMMAND DESCRIPTION :

SI.NO	COMMAND	USE
1)	Ipconfig	This command can be utilized to verify a network connection as well as verify your network settings.
2)	Netstat	Displays active TCP connections, ports on which the computer is listening, Ethernet statistics, the IP routing table etc...
3)	Tracert	The tracert command is used to visually see a network packet being sent and received and the amount of hops required for that packet to get to its destination.
4)	Ping	Helps in determining TCP/IP networks ip address as well as determine issues with the network and assists in resolving them.
5)	Pathping	Provides information about network latency and network loss at intermediate hops between a source and destination pathping sends.

6)	Nslookup	Displays information that you can use to diagnose Domain Name System (DNS) infrastructure.
7)	Nbtstat	MS_DOS utility that displays protocol statistics & current TCP/IP connections using NBT.
8)	Getmac	DOS command used to show both local & remote MAC addresses when run with no parameters (i.e.getmac) it displays MAC addresses for the local system. When run with the /s parameter (Eg. Getmac /s \\too> it displays MAC address for the remote computer).

OUTPUT :

Ipconfig

```
C:\Windows\system32\cmd.exe + 
C:\Users\MSD>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 1:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 2:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . :

Wireless LAN adapter Wi-Fi:

    Connection-specific DNS Suffix . :
    IPv6 Address . . . . . : 2401:4900:2325:1cd1:bf17:7e12:afed:6a20
    Temporary IPv6 Address . . . . . : 2401:4900:2325:1cd1:f47a:5620:6663:c43f
    Link-local IPv6 Address . . . . . : fe80::3d7d:8789:4d20:4a40%16
    IPv4 Address . . . . . : 192.168.153.172
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : fe80::c022:3dff:fe0c:fe56%16
                                         192.168.153.211

Ethernet adapter Bluetooth Network Connection:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . :
```

Netstat

```
C:\Windows\system32\cmd.e: X + ^

Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\MSD>netstat

Active Connections

Proto Local Address          Foreign Address          State
TCP   192.168.153.172:50729  20.198.119.84:https    ESTABLISHED
TCP   192.168.153.172:50807  152.195.38.76:http    TIME_WAIT
TCP   192.168.153.172:50809  152.195.38.76:http    TIME_WAIT
TCP   [2401:4900:2325:1cd1:f47a:5620:6663:c43f]:50797  g2600-140f-f400-0000-0000-0000-1730-e221:http  TIME_WAIT
TCP   [2401:4900:2325:1cd1:f47a:5620:6663:c43f]:50798  g2600-140f-f400-0000-0000-0000-1730-e221:http  TIME_WAIT
TCP   [2401:4900:2325:1cd1:f47a:5620:6663:c43f]:50800  g2600-140f-f400-0000-0000-0000-1730-e221:http  TIME_WAIT
```

Tracert

```
C:\Windows\system32\cmd.e: X + ^

C:\Users\MSD>tracert www.srmist.edu.in

Tracing route to srmist-alb-630144276.ap-south-1.elb.amazonaws.com [35.154.166.26]
over a maximum of 30 hops:

 1  3 ms    3 ms    3 ms  192.168.153.211
 2  *        *        *        Request timed out.
 3  *        *        *        Request timed out.
 4  46 ms   30 ms   22 ms  10.50.221.46
 5  59 ms   40 ms   22 ms  125.19.176.89
 6  74 ms   29 ms   66 ms  182.79.239.197
 7  *        *        *        Request timed out.
 8  *        *        *        Request timed out.
 9  162 ms  82 ms   81 ms  52.95.64.170
10  86 ms   94 ms   76 ms  52.95.64.167
11  103 ms  70 ms   73 ms  52.95.66.127
12  84 ms   74 ms   79 ms  52.95.65.133
13  *        *        *        Request timed out.
14  *        *        *        Request timed out.
15  *        *        *        Request timed out.
16  *        *        *        Request timed out.
17  *        *        *        Request timed out.
18  *        *        *        Request timed out.
19  *        *        *        Request timed out.
20  *        *        *        Request timed out.
21  *        *        *        Request timed out.
22  *        *        *        Request timed out.
23  *        *        *        Request timed out.
24  *        *        *        Request timed out.
25  *        *        *        Request timed out.
26  *        *        *        Request timed out.
27  *        *        *        Request timed out.
28  *        *        *        Request timed out.
29  *        *        *        Request timed out.
30  *        *        *        Request timed out.

Trace complete.
```

Ping

```
C:\Windows\system32\cmd.e: × + ▾

Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\MSD>ping 1.1.1.7

Pinging 1.1.1.7 with 32 bytes of data:
Reply from 1.1.1.7: bytes=32 time=279ms TTL=56
Reply from 1.1.1.7: bytes=32 time=44ms TTL=56
Reply from 1.1.1.7: bytes=32 time=84ms TTL=56
Reply from 1.1.1.7: bytes=32 time=73ms TTL=56

Ping statistics for 1.1.1.7:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 44ms, Maximum = 279ms, Average = 120ms
```

Pathping

```
C:\Users\MSD>pathping www.srmist.edu.in

Tracing route to srmist-alb-630144276.ap-south-1.elb.amazonaws.com [13.127.51.112]
over a maximum of 30 hops:
  0  MSD-K67DBKG [192.168.153.172]
  1  192.168.153.211
  2  *      *      *
Computing statistics for 25 seconds...
          Source to Here   This Node/Link
Hop  RTT     Lost/Sent = Pct  Lost/Sent = Pct  Address
  0           0/ 100 =  0%      |      MSD-K67DBKG [192.168.153.172]
                  0/ 100 =  0%      |
  1    7ms     0/ 100 =  0%     0/ 100 =  0%  192.168.153.211

Trace complete.
```

Nslookup

```
C:\Users\MSD>nslookup www.srmist.edu.in
Server:  Unknown
Address: 192.168.153.211

Non-authoritative answer:
Name:    srmist-alb-630144276.ap-south-1.elb.amazonaws.com
Addresses: 13.127.51.112
            35.154.166.26
Aliases:  www.srmist.edu.in
```

Nbtstat

```
C:\Windows\system32\cmd.e: + ▾
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\MSD>nbtstat

Displays protocol statistics and current TCP/IP connections using NBT
(NetBIOS over TCP/IP).

NBTSTAT [ [-a RemoteName] [-A IP address] [-c] [-n]
           [-r] [-R] [-RR] [-s] [-S] [interval] ]

-a (Adapter status) Lists the remote machine's name table given its name
-A (Adapter status) Lists the remote machine's name table given its
                   IP address.
-c (cache)          Lists NBT's cache of remote [machine] names and their IP addresses
-n (names)          Lists local NetBIOS names.
-r (resolved)       Lists names resolved by broadcast and via WINS
-R (Reload)         Purges and reloads the remote cache name table
-S (Sessions)       Lists sessions table with the destination IP addresses
-s (sessions)       Lists sessions table converting destination IP
                   addresses to computer NETBIOS names.
-RR (ReleaseRefresh) Sends Name Release packets to WINS and then, starts Refresh

RemoteName   Remote host machine name.
IP address   Dotted decimal representation of the IP address.
interval    Redisplays selected statistics, pausing interval seconds
            between each display. Press Ctrl+C to stop redisplaying
            statistics.
```

```
C:\Users\MSD>nbtstat -n
```

```
Ethernet:
NodeIpAddress: [0.0.0.0] Scope Id: []

No names in cache
```

```
Bluetooth Network Connection:
NodeIpAddress: [0.0.0.0] Scope Id: []

No names in cache
```

```
Wi-Fi:
NodeIpAddress: [192.168.153.172] Scope Id: []

NetBIOS Local Name Table
```

Name	Type	Status
MSD-K67DBKG	<20> UNIQUE	Registered
MSD-K67DBKG	<00> UNIQUE	Registered
WORKGROUP	<00> GROUP	Registered

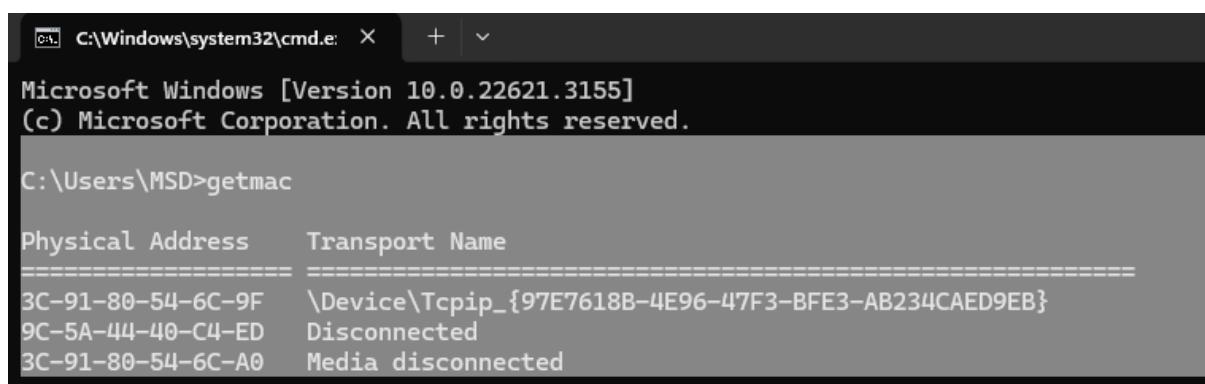
```
Local Area Connection* 1:
NodeIpAddress: [0.0.0.0] Scope Id: []

No names in cache
```

```
Local Area Connection* 2:
NodeIpAddress: [0.0.0.0] Scope Id: []

No names in cache
```

Getmac



```
C:\Windows\system32\cmd.exe + - Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\MSD>getmac

Physical Address      Transport Name
===== =====
3C-91-80-54-6C-9F    \Device\Tcpip_{97E7618B-4E96-47F3-BFE3-AB234CAED9EB}
9C-5A-44-40-C4-ED    Disconnected
3C-91-80-54-6C-A0    Media disconnected
```

RESULT :

Thus, the various network commands are executed and the output is verified.

EX.NO : 02

DATE : 08.01.24

ANALYZING THE PERFORMANCE OF VARIOUS CONFIGURATIONS AND PROTOCOLS OF LAN ESTABLISHING A LOCAL AREA NETWORK (LAN)

AIM :

To set up a Local Area Network using Cisco Packet Tracer.

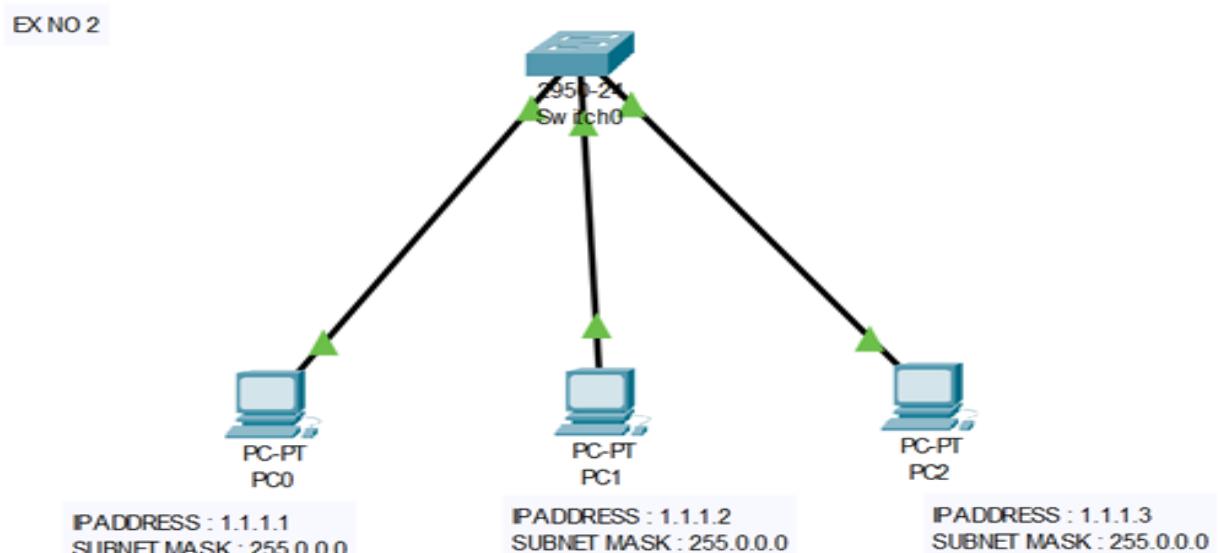
REQUIREMENTS :

- Three Windows PC.
- One Switch(2950-24) or One Hub.
- Three Straight Line LAN Cables.
- Cisco Packet Tracer.

PROCEDURES :

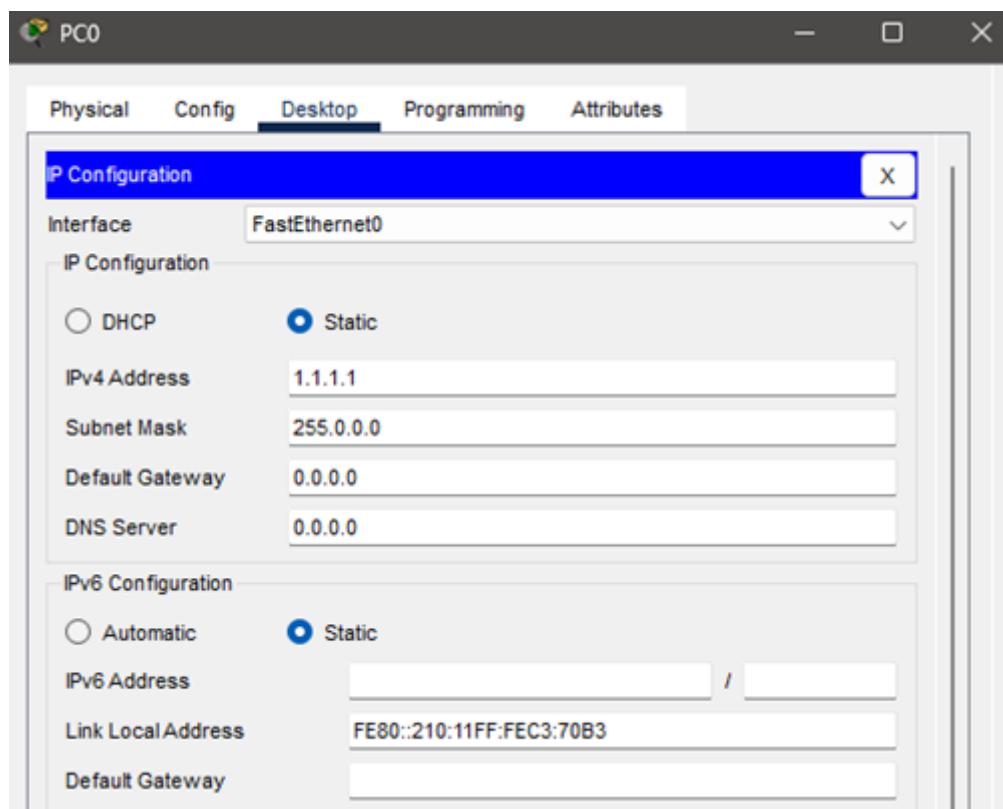
- Open CISCO PACKET TRACER software.
- Draw the Three PC using END Device Icons.
- Draw the Cisco 24 Port Switch Using Switch icon lists.
- Make the Connections using Copper-Straight-Through Ethernet Cables.
- Enter the IP Address To Each Machine.
- Check the Network Connections using Add Simple PDU(P).

NETWORK TOPOLOGY :

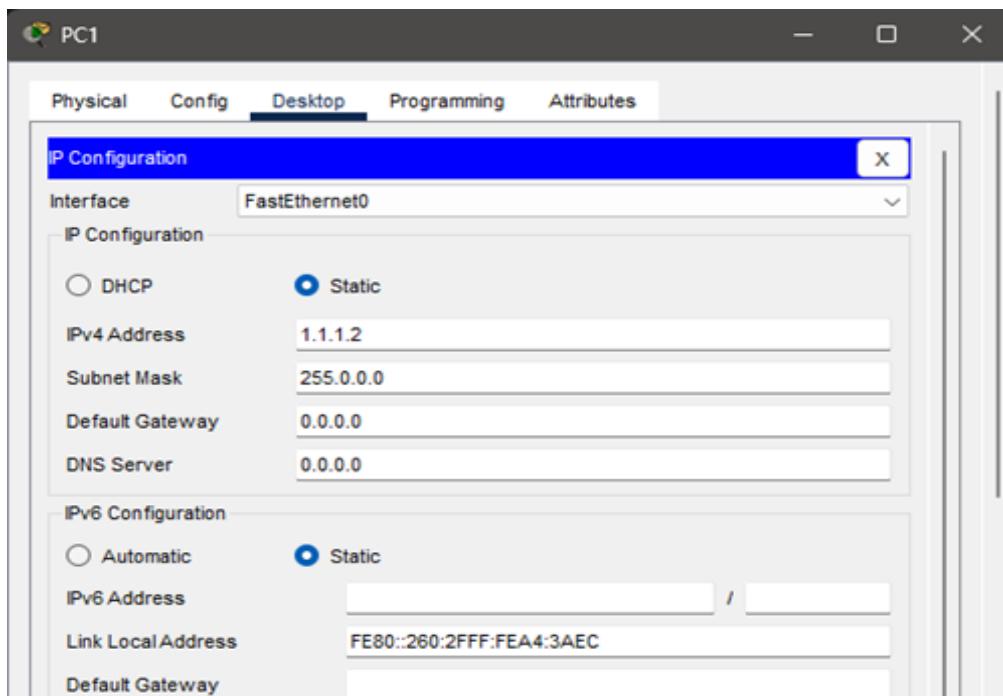


HOST IP ADDRESS :

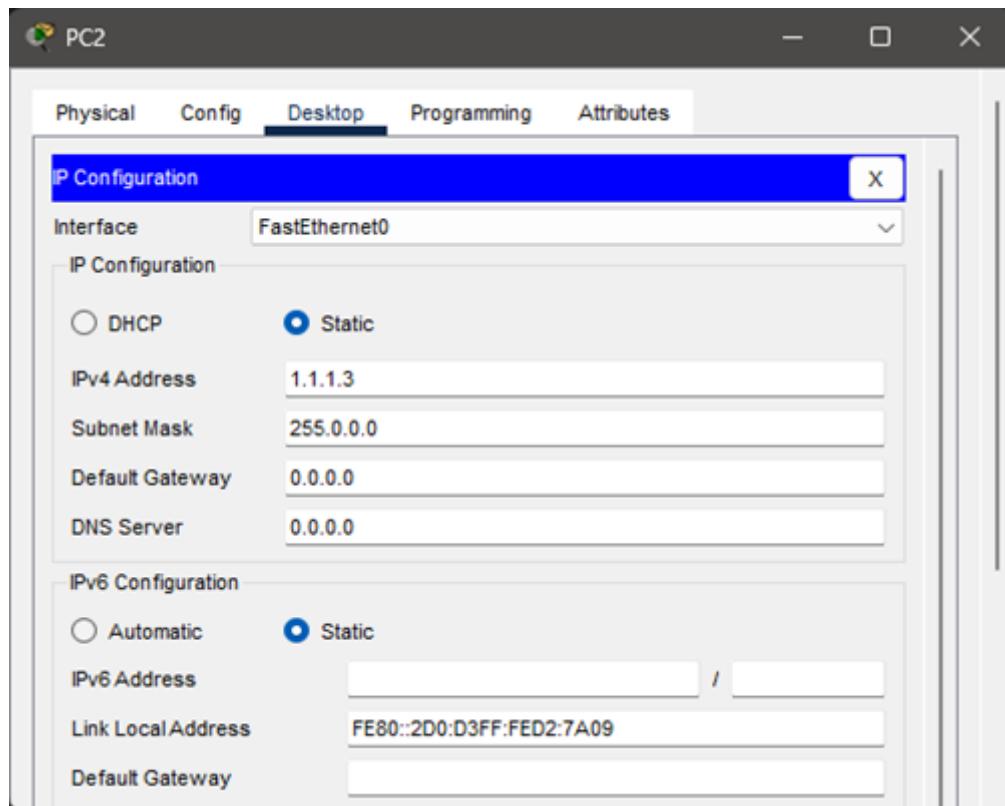
PC0 IP CONFIGURATION



PC1 IP CONFIGURATION



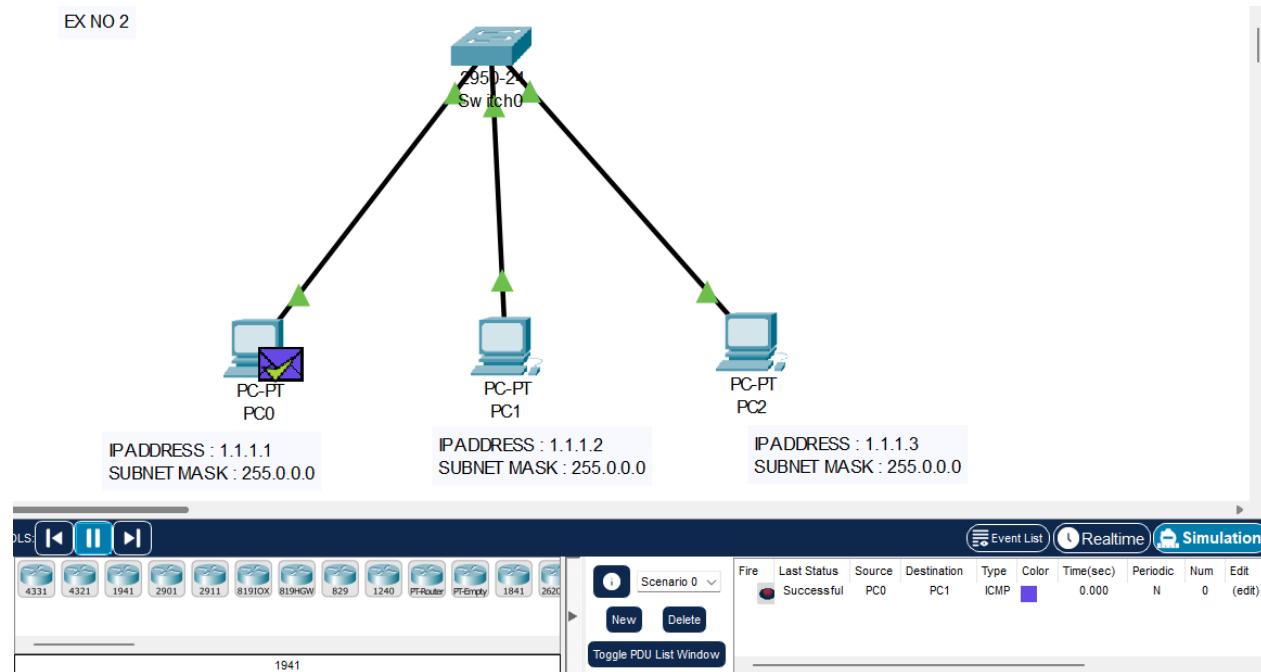
PC2 IP CONFIGURATION



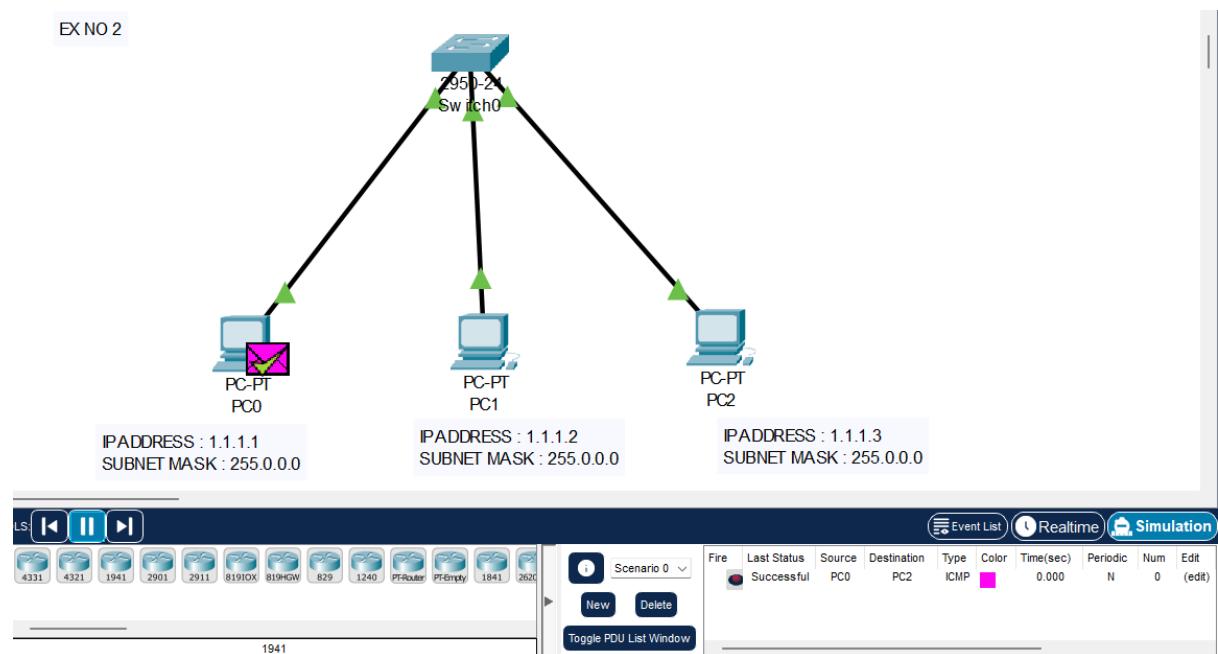
VERIFY LAN NETWORK CONNECTIVITY :

Using Add Simple PDU(p), Click the mail icon and then drop one mail to PC0 and another mail to PC1. If the resultant window shows the successful delivery, then network connectivity is successful.

HOST PC0 TO PC1



HOST PC1 TO PC2



RESULT :

Thus, the LAN connection is established, hosts are configured, the communications among the machines are verified and manipulated successfully.

EX.NO : 03

DATE : 18.01.24

ANALYZING THE PERFORMANCE OF VARIOUS CONFIGURATIONS AND PROTOCOLS IN LAN CONNECTING TWO LANs USING ROUTER WITH STATIC ROUTER

AIM :

To establish connection between two LANs by extending routing connection using router.

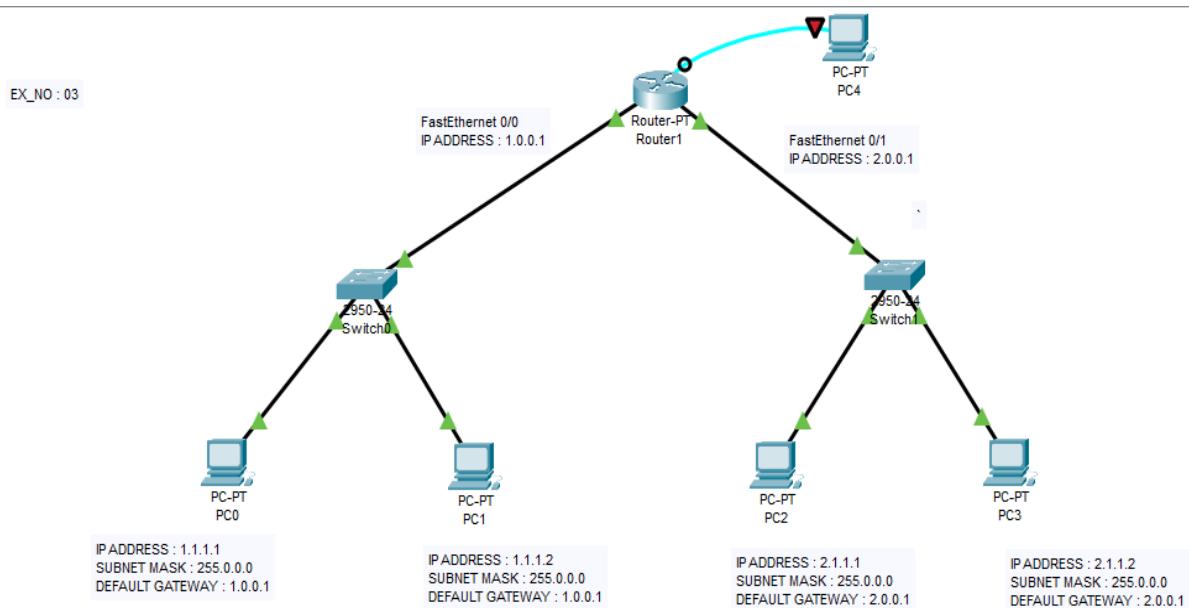
REQUIREMENTS :

- Five windows PC.
- Two Switch (2950-24).
- Six Straight Line LAN Cables.
- One Router (Router PT).
- One console connection of router with PC to configure router.
- Cisco Packet Tracer.

PROCEDURES :

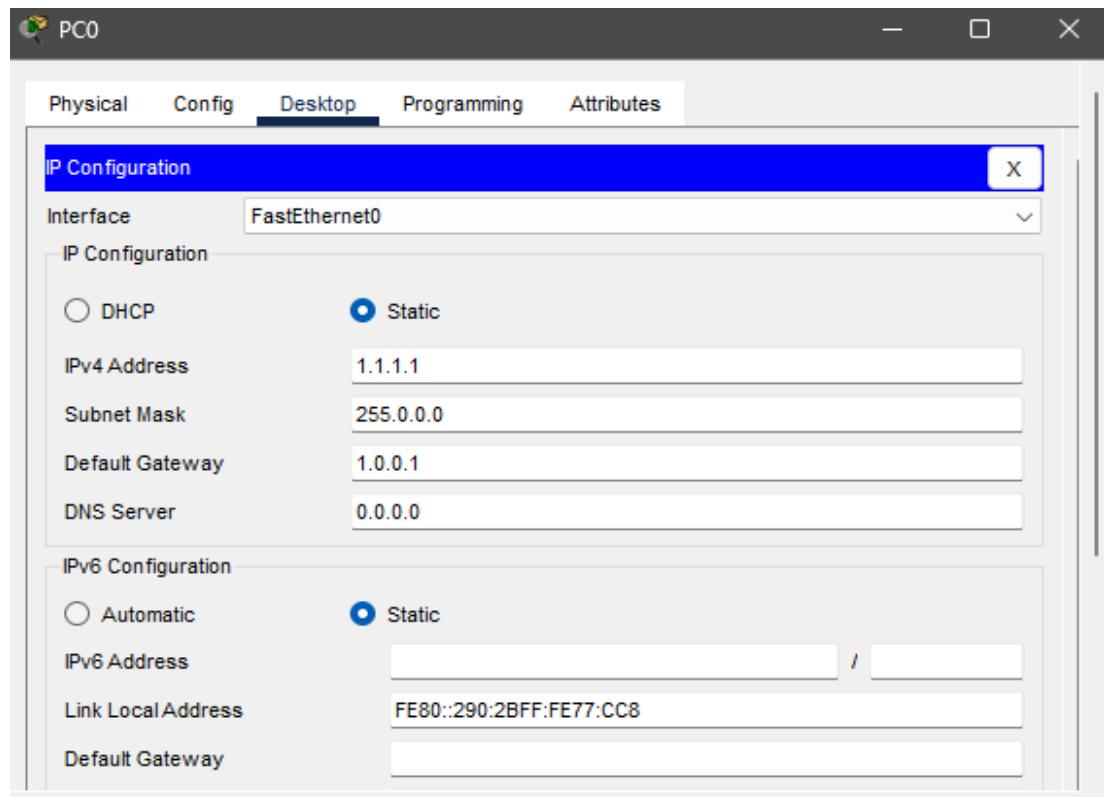
- Open CISCO PACKET TRACER software.
- Draw the Five PC using END Device Icons.
- Draw the Cisco 24 Port Switch Using Switch icon lists.
- Draw the Cisco Generic Routers using Router icon lists.
- Make the Connections using Copper-Straight-Through Ethernet Cables.
- Enter the IP Address To Each Machine.
- Configure Router PT – 0.
- Check the Network Connections using Add Simple PDU(P).

NETWORK TOPOLOGY :

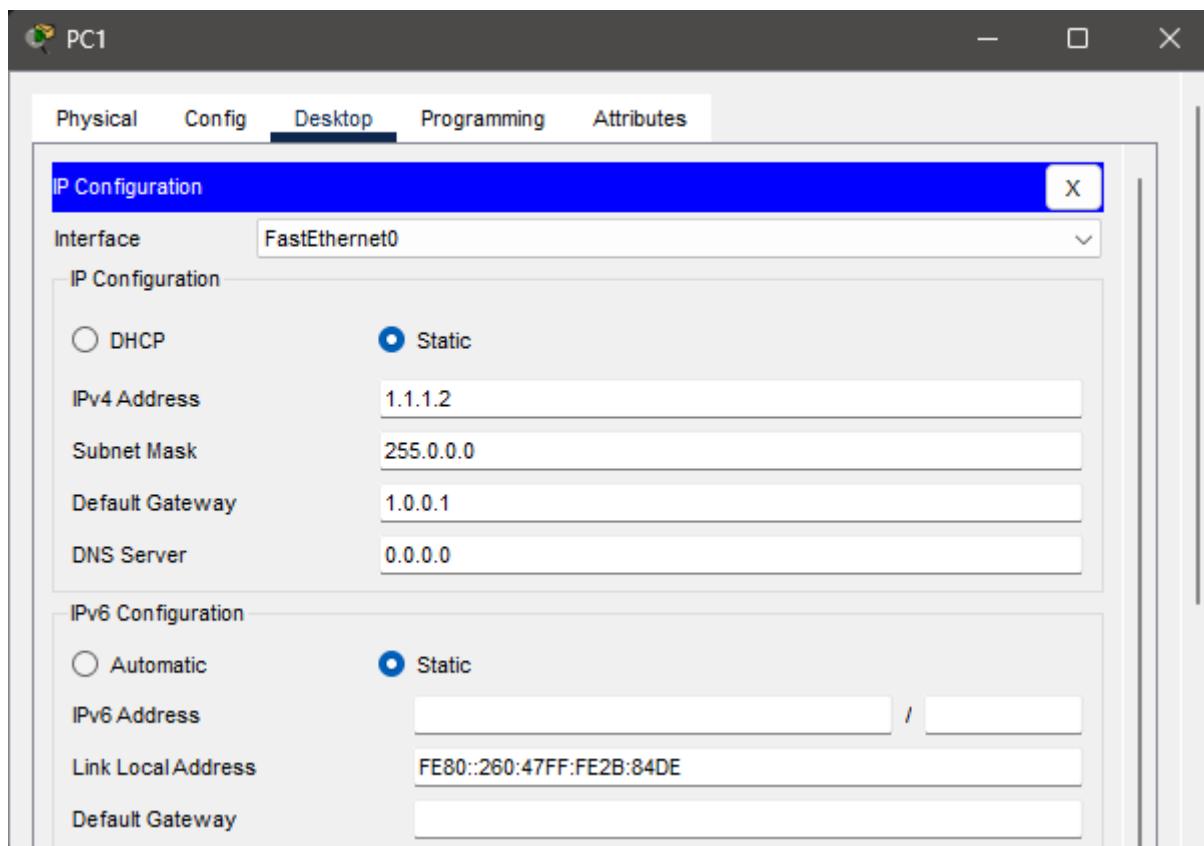


HOST PC IP ADDRESS :

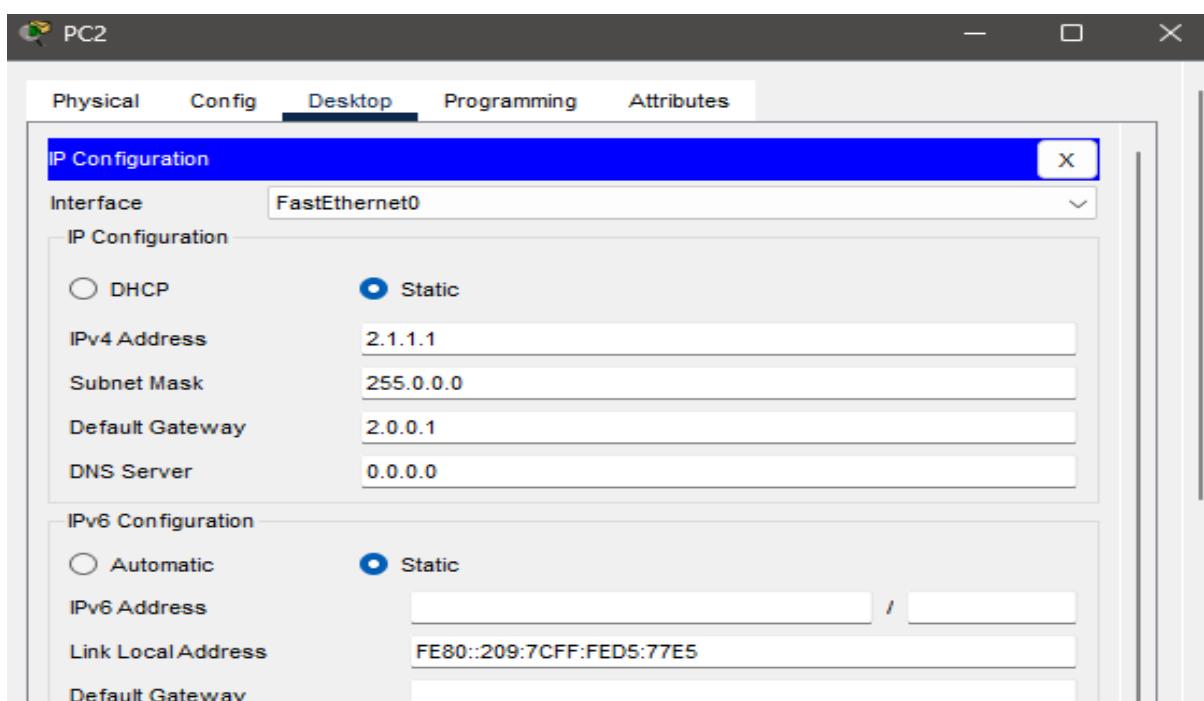
PC0 IP CONFIGURATION



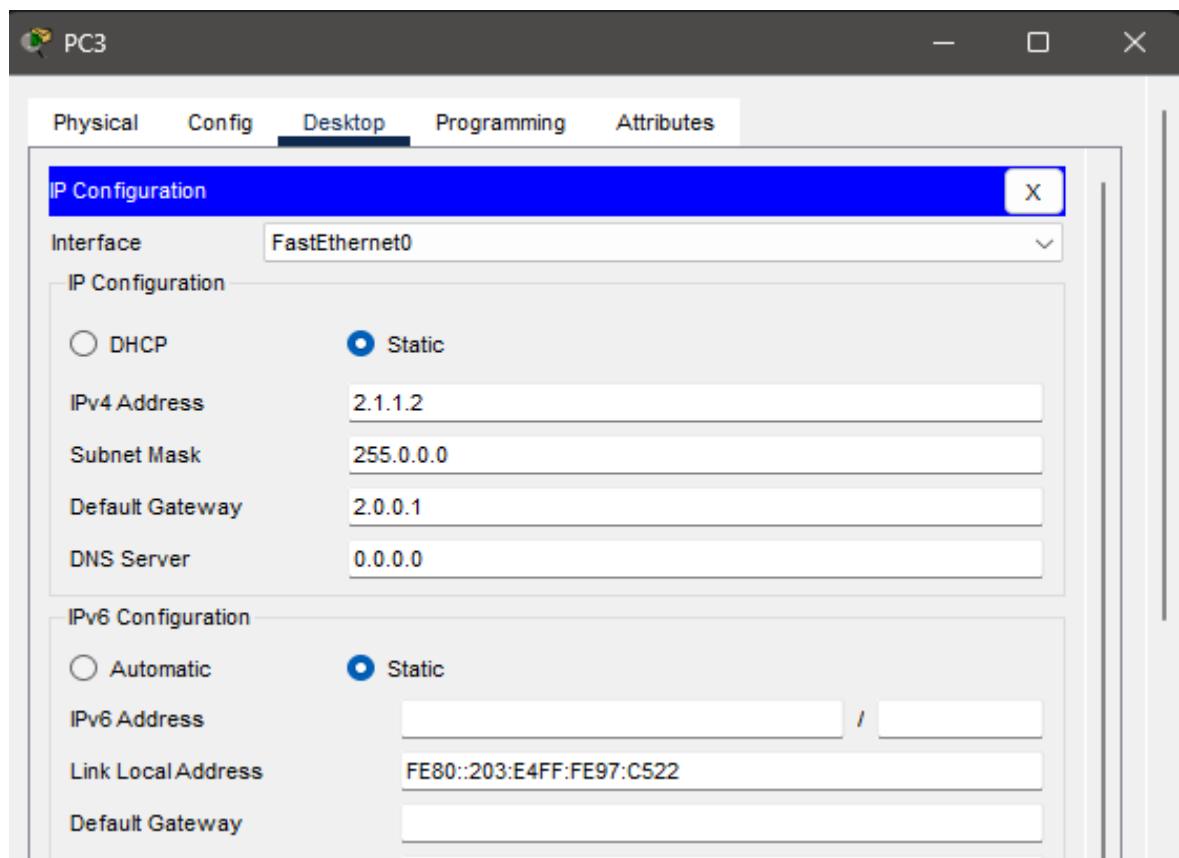
PC1 IP CONFIGURATION :



PC2 IP CONFIGURATION :

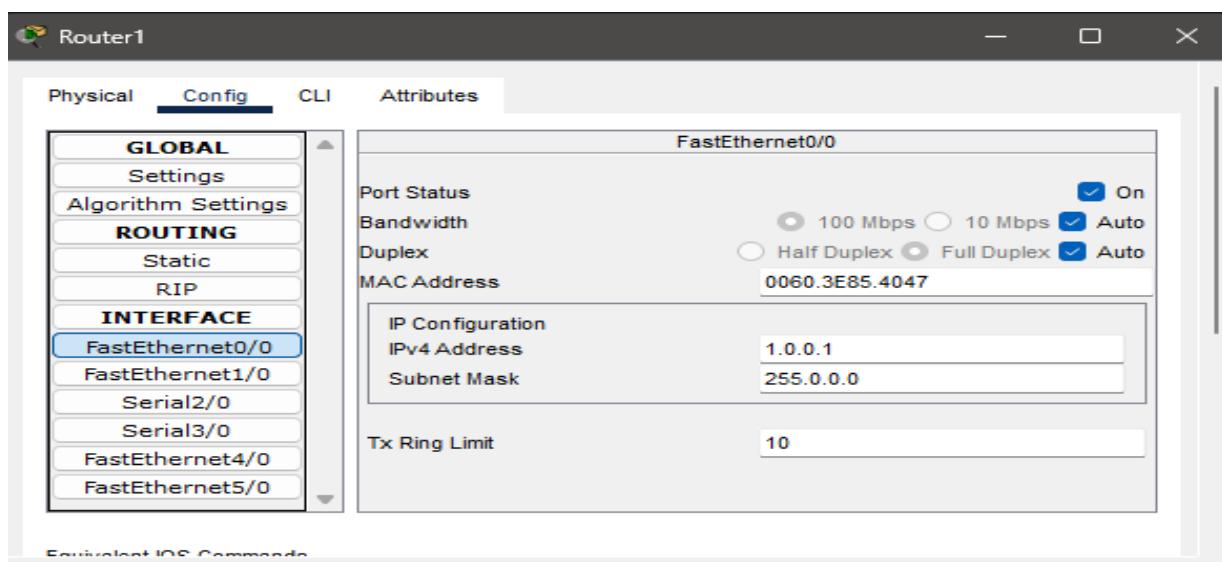


PC3 IP CONFIGURATION :

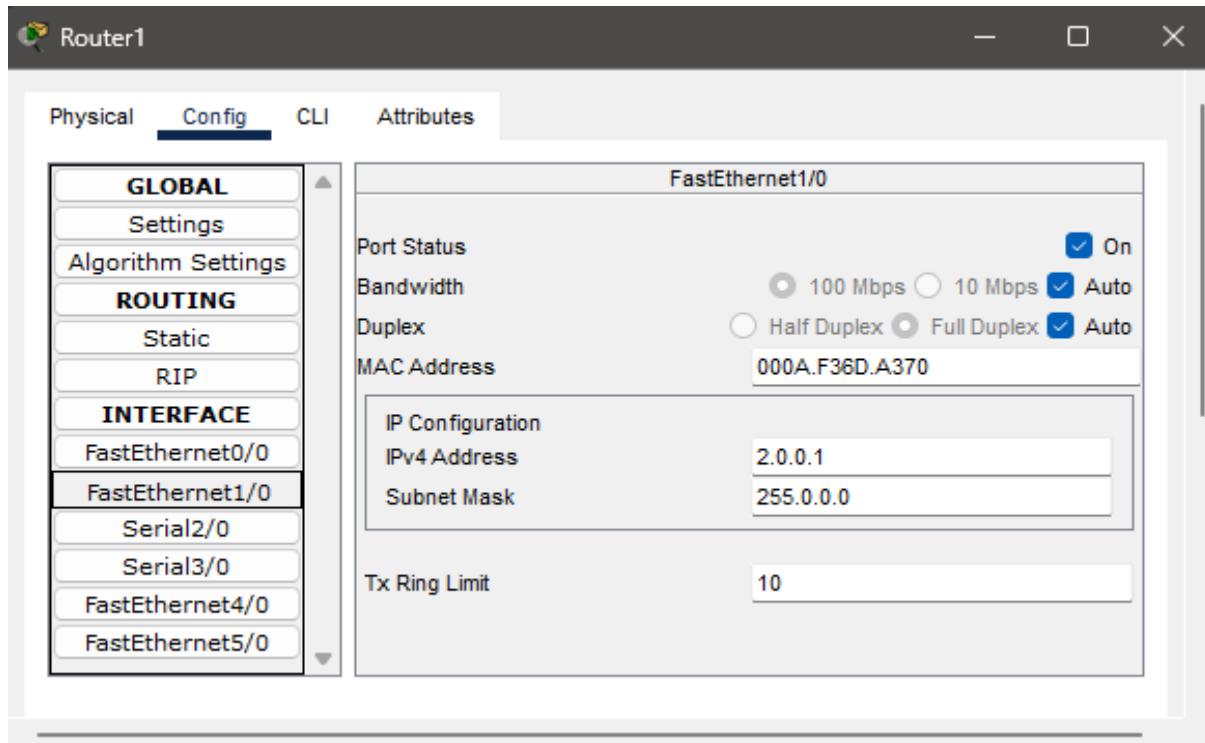


ROUTER 0 CONFIGURATION :

FASTERNET 0/0 CONFIGURATION



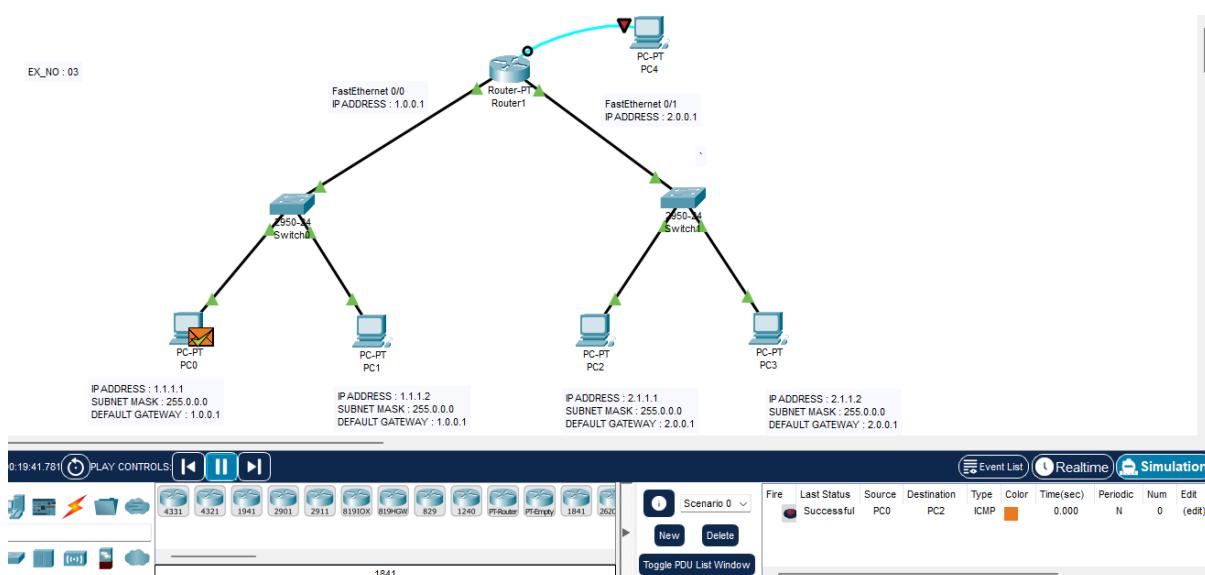
FASTETHERNET 1/0 CONFIGURATION



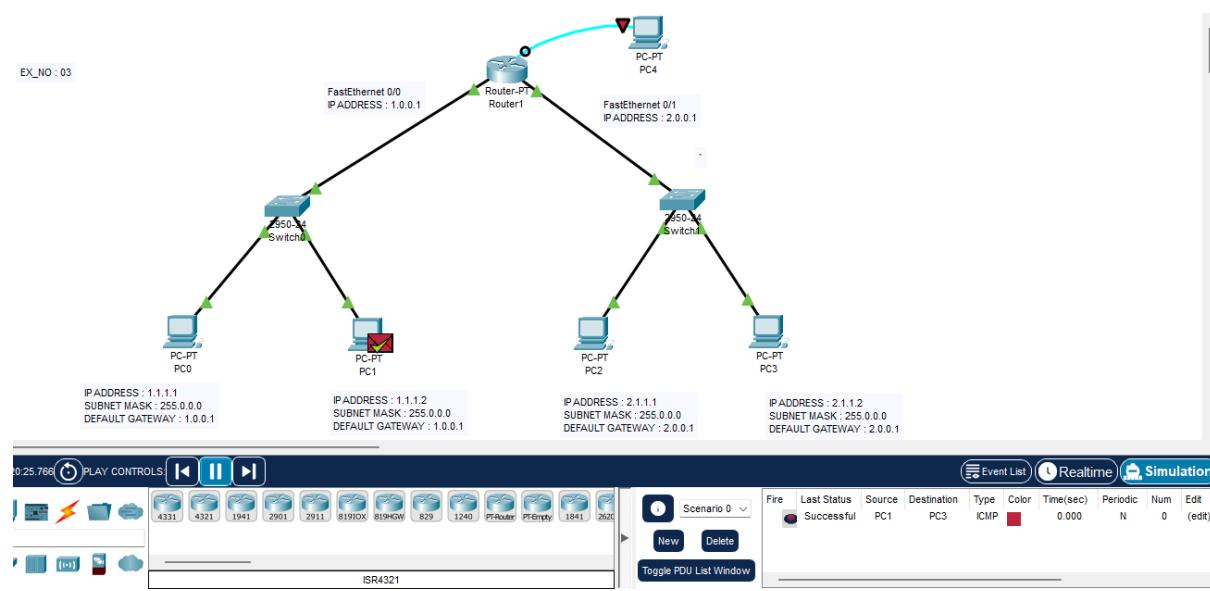
VERIFY LAN NETWORK CONNECTIVITY :

Using Add Simple PDU(p), Click the mail icon and then drop one mail to one of the PC in first lan and another mail to PC in another lan. If the resultant window shows the successful delivery, then network connectivity is successful.

HOST PC0 TO PC2



HOST PC1 TO PC3



RESULT :

Thus, two LANs are connected using router with static router and the communication between LANs is checked successfully.

EX.NO : 04

DATE : 22.01.24

ANALYZING THE PERFORMANCE OF VARIOUS CONFIGURATIONS AND PROTOCOLS IN LAN MULTI-ROUTING CONNECTION

AIM :

To establish connection between two LANs by extending multi-routing connection.

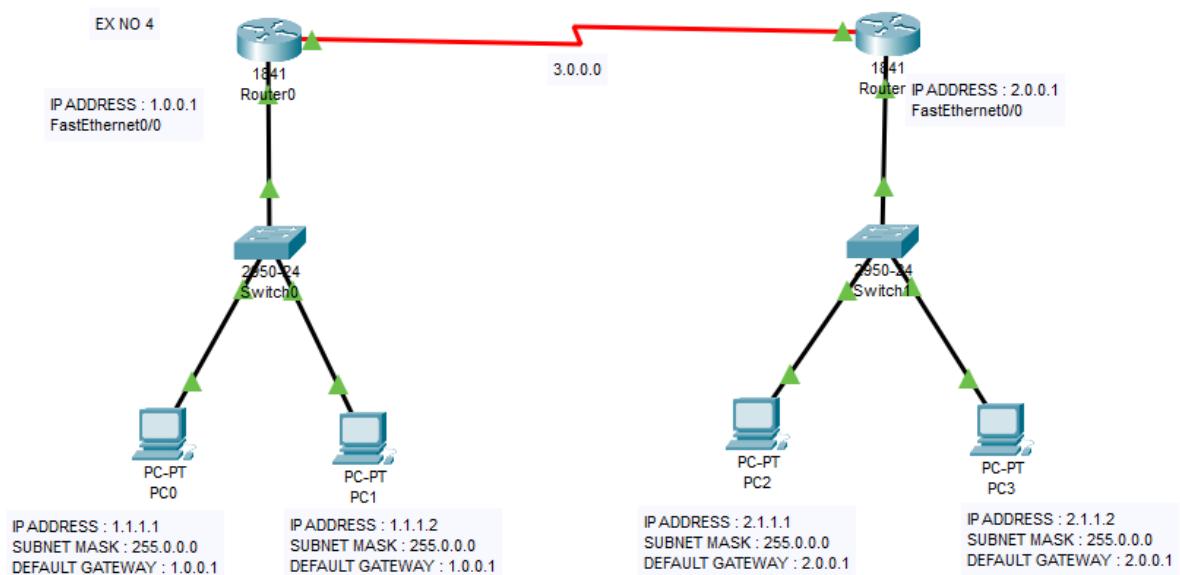
REQUIREMENTS :

- Four windows PC.
- Two Switch (2950-24).
- Six Straight Line LAN Cables.
- Two Router (1841).
- One Serial DTE Cable.
- Cisco Packet Tracer.

PROCEDURES :

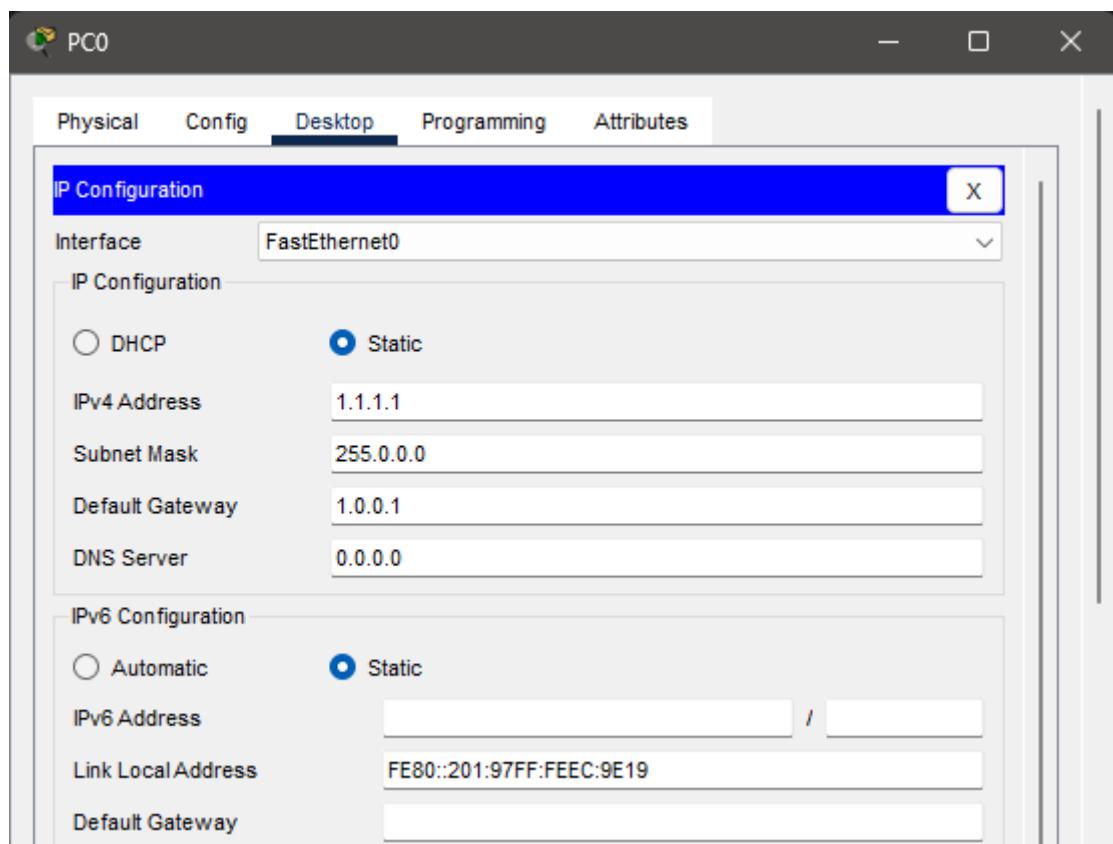
- Open CISCO PACKET TRACER software.
- Draw the Four PC using END Device Icons.
- Draw the Cisco 24 Port Switch Using Switch icon lists.
- Draw the Cisco 1841 Routers using Router icon lists.
- Make the Connections using Copper-Straight-Through Ethernet Cables and Serial DTE Cables.
- Enter the IP Address To Each Machine.
- Configure Routers 0,1 and Serial 0/0 for 2Routers.
- Check the Network Connections using Add Simple PDU(P).

NETWORK TOPOLOGY :

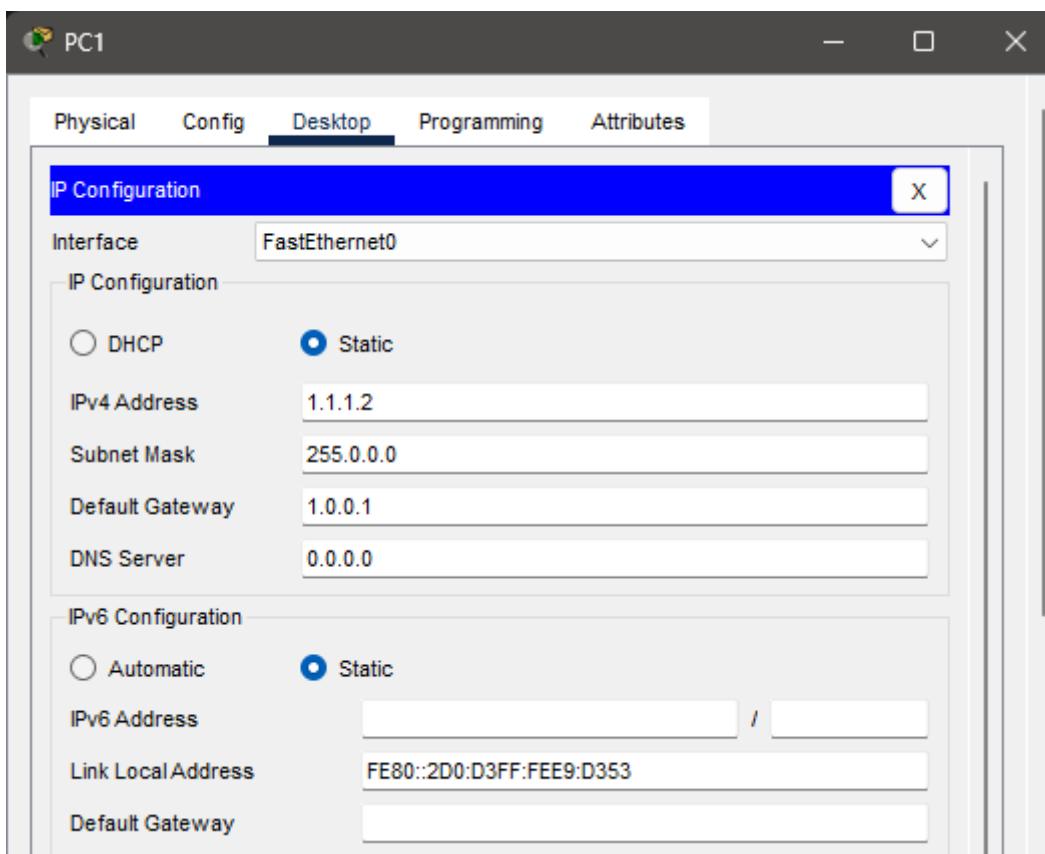


HOST PC IP ADDRESS :

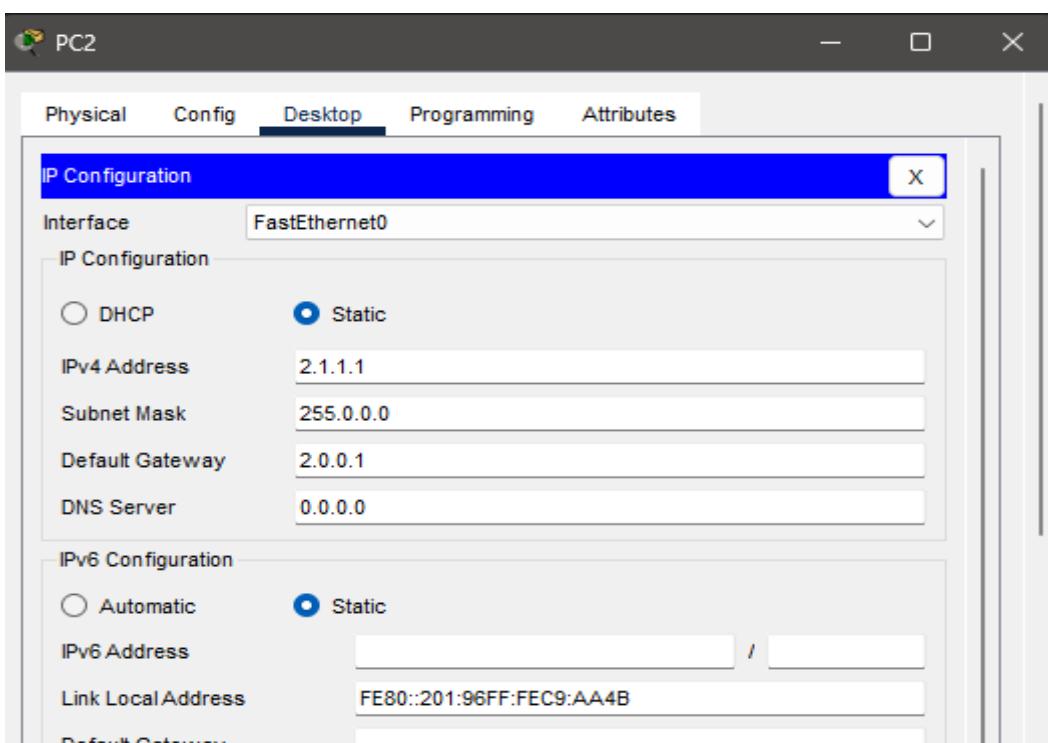
PC0 IP CONFIGURATION



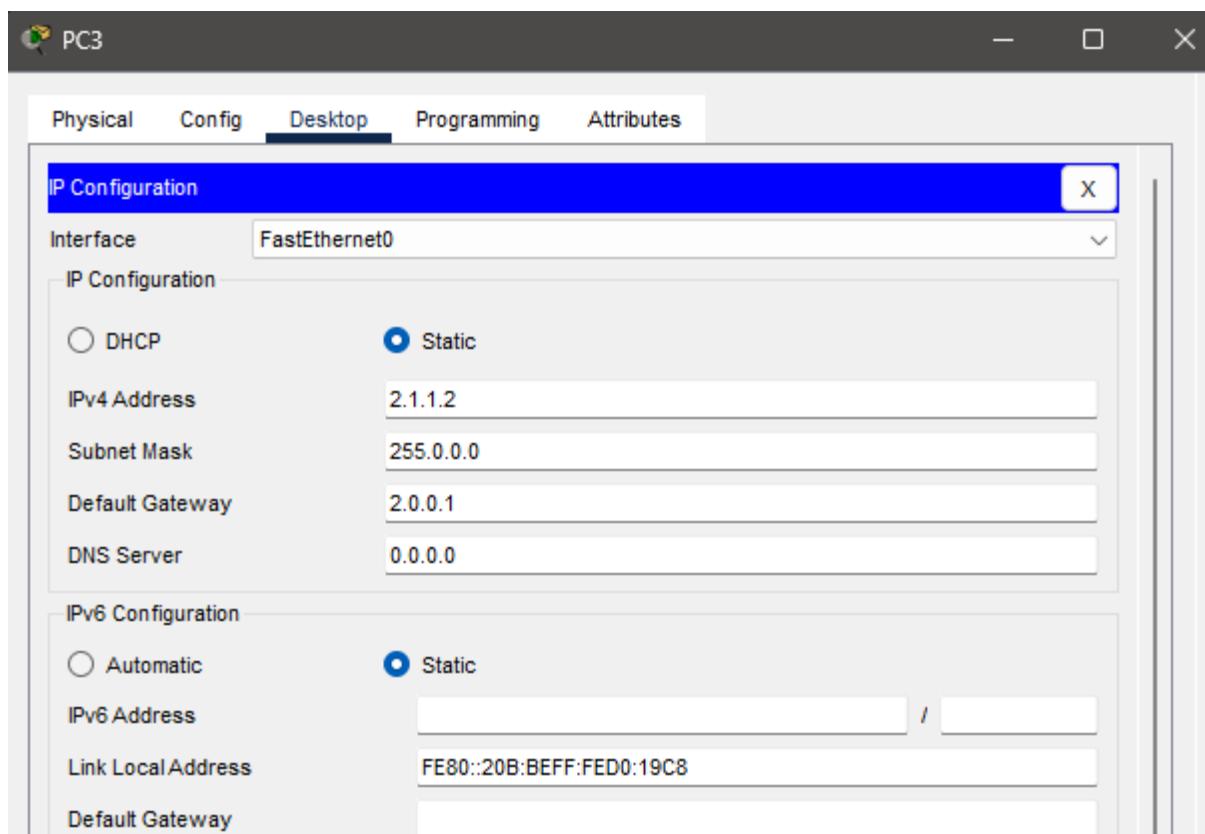
PC1 IP CONFIGURATION



PC2 IP CONFIGURATION

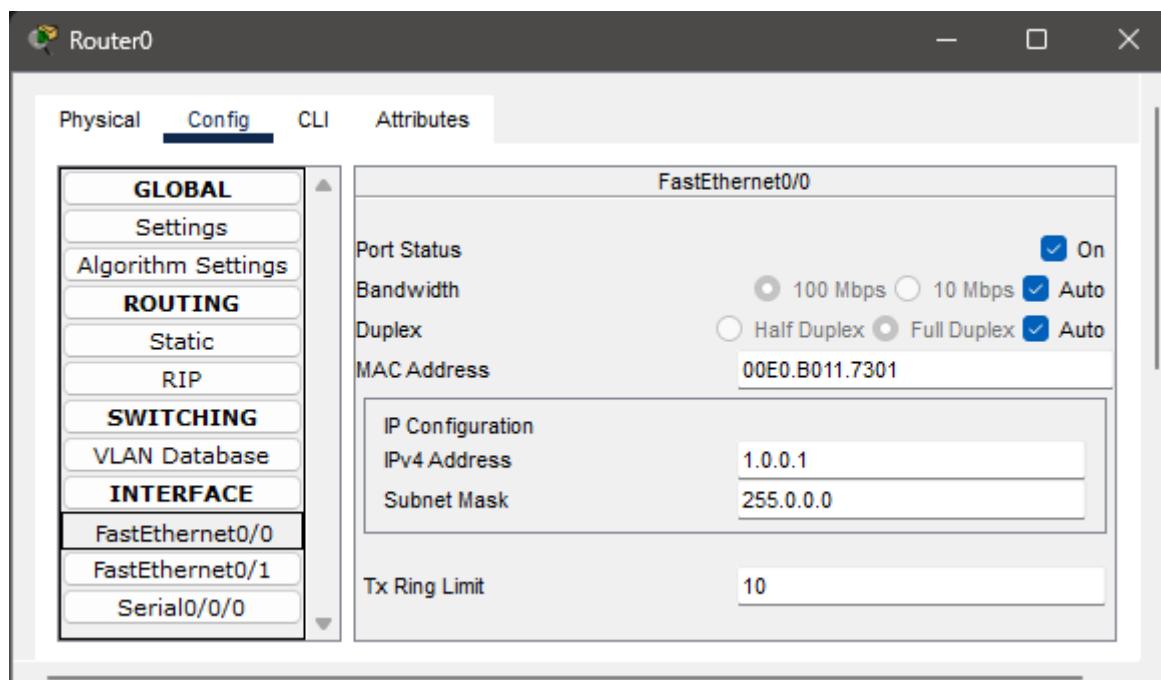


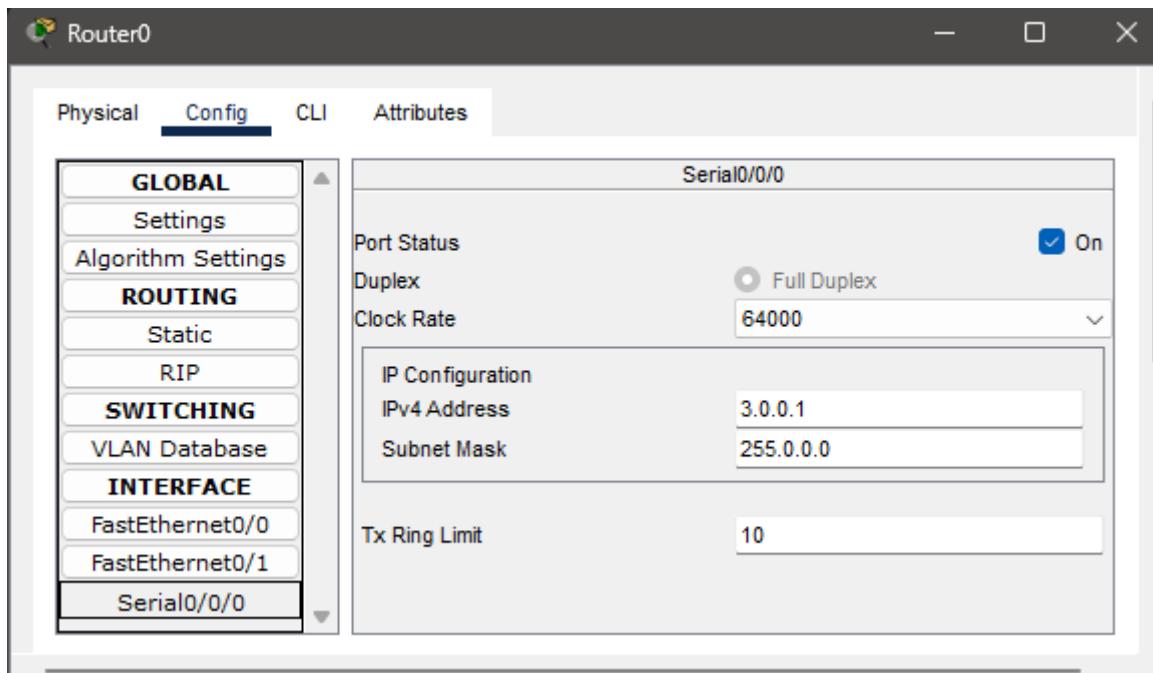
PC3 IP CONFIGURATION



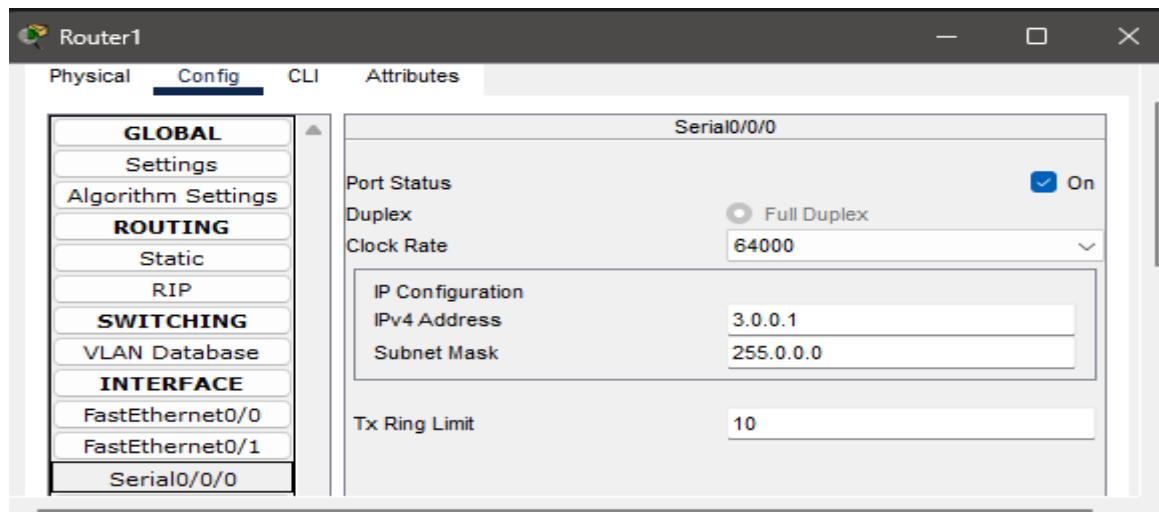
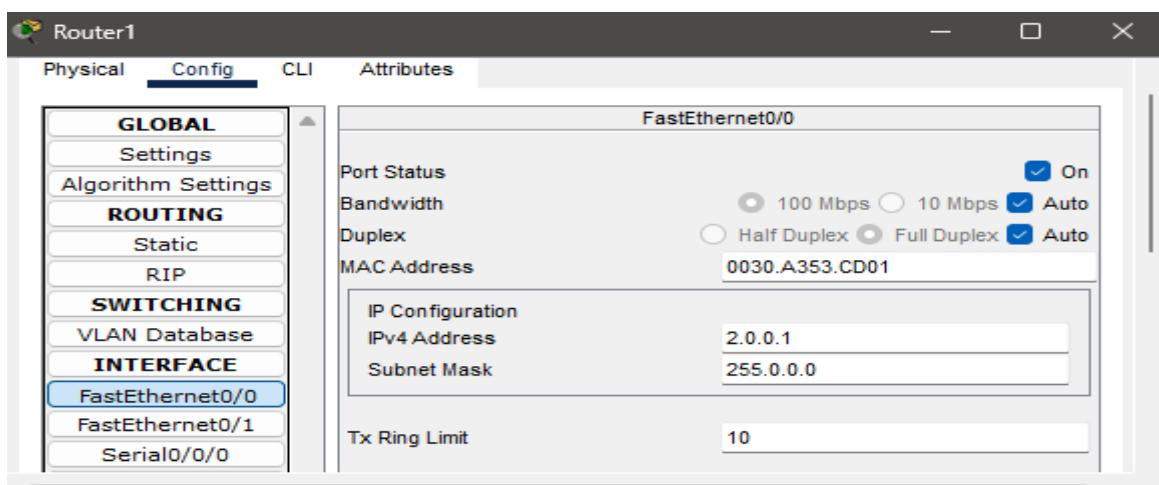
ROUTERS CONFIGURATION :

ROUTER 0 CONFIGURATION





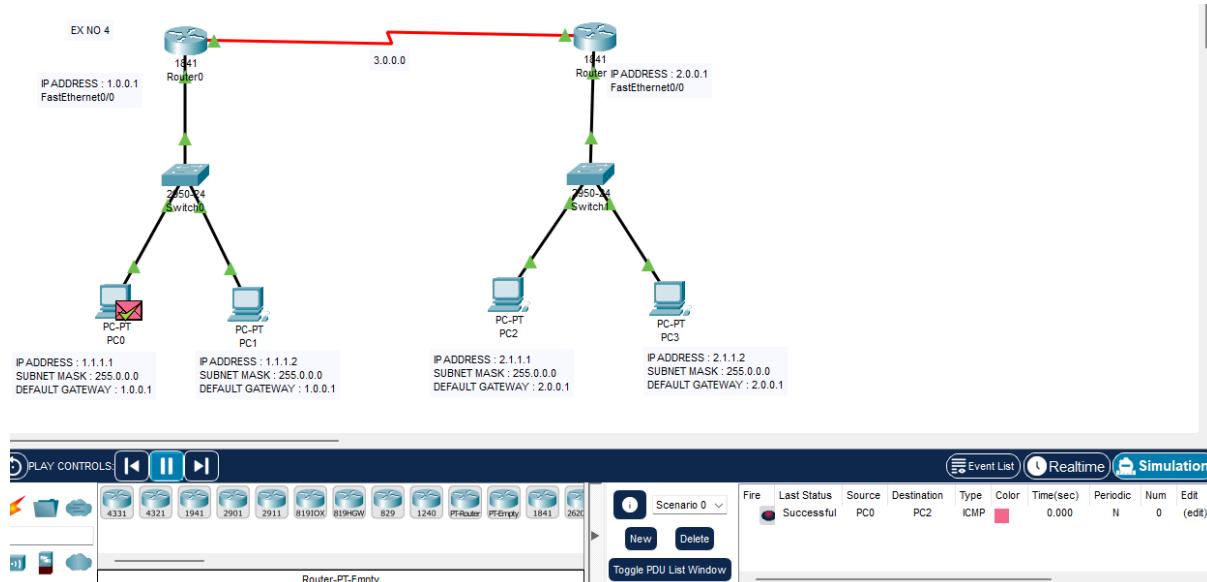
ROUTER 1 CONFIGURATION :



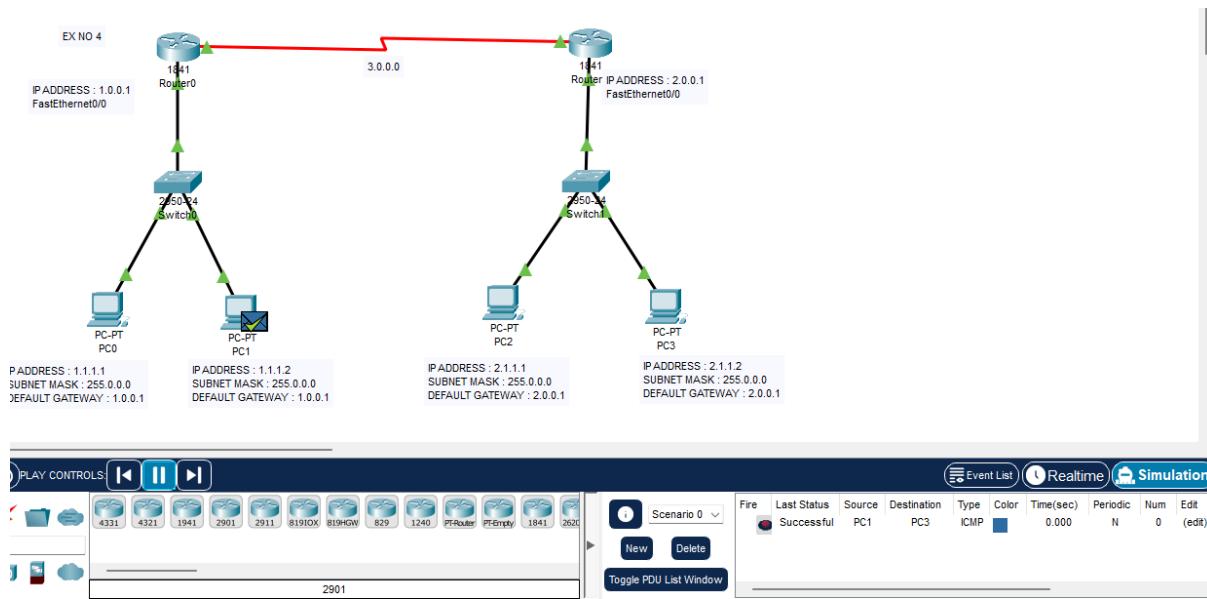
VERIFY LAN NETWORK CONNECTIVITY :

Using Add Simple PDU(p), Click the mail icon and then drop one mail to one of the PC in first lan and another mail to PC in another lan. If the resultant window shows the successful delivery, then network connectivity is successful.

HOST PC0 TO PC2



HOST PC1 TO PC3



RESULT :

Thus, two LANs are connected using multiple routers and the communication between LANs is checked successfully.

EX.NO : 05

DATE : 30.01.24

CONNECTING TWO LANs USING BRIDGE

AIM :

To establish connection between two LANs by using Bridge.

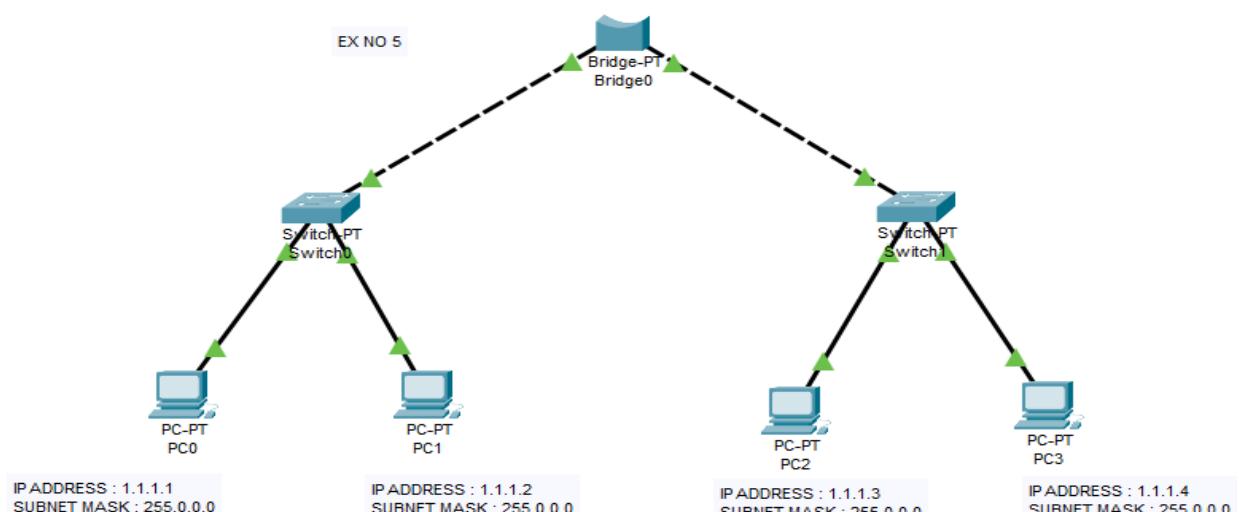
REQUIREMENTS :

- Four windows PC.
- Two Switch (PT).
- Six Straight Line LAN Cables.
- One Bridge (PT).
- Cisco Packet Tracer.

PROCEDURES :

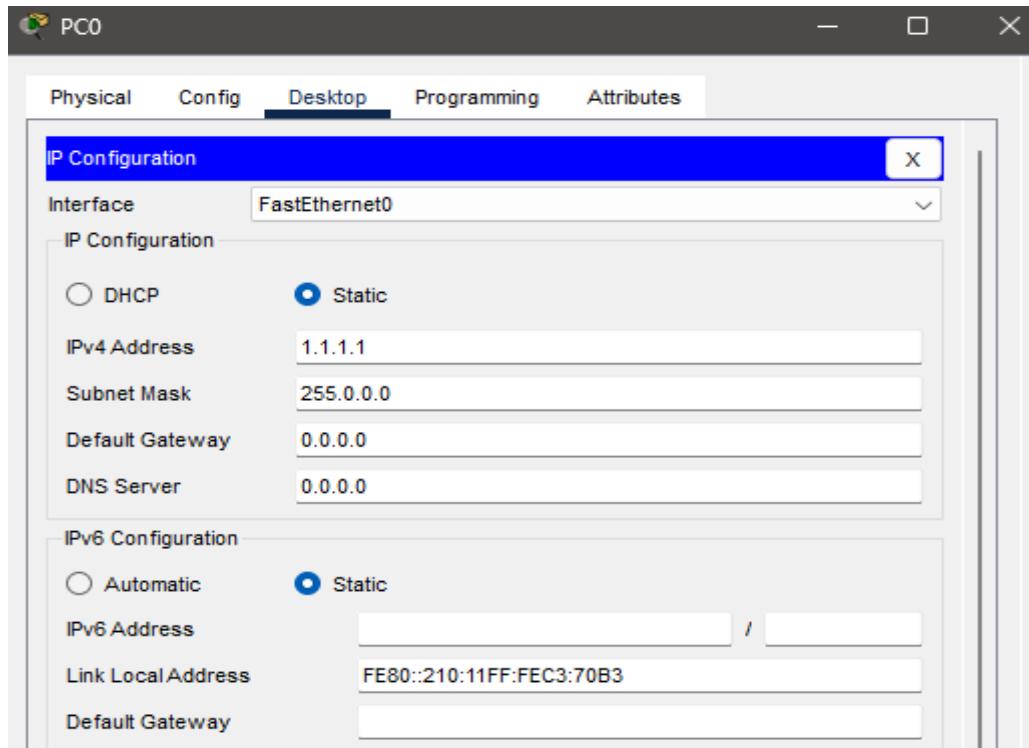
- Open CISCO PACKET TRACER software.
- Draw the Four PC using END Device Icons.
- Draw the Cisco PT Port Switch Using Switch icon lists.
- Draw the Cisco PT Bridge.
- Make the Connections using Copper-Straight-Through Ethernet Cables.
- Enter the IP Address To Each Machine.
- Check the Network Connections using Add Simple PDU(P).

NETWORK TOPOLOGY

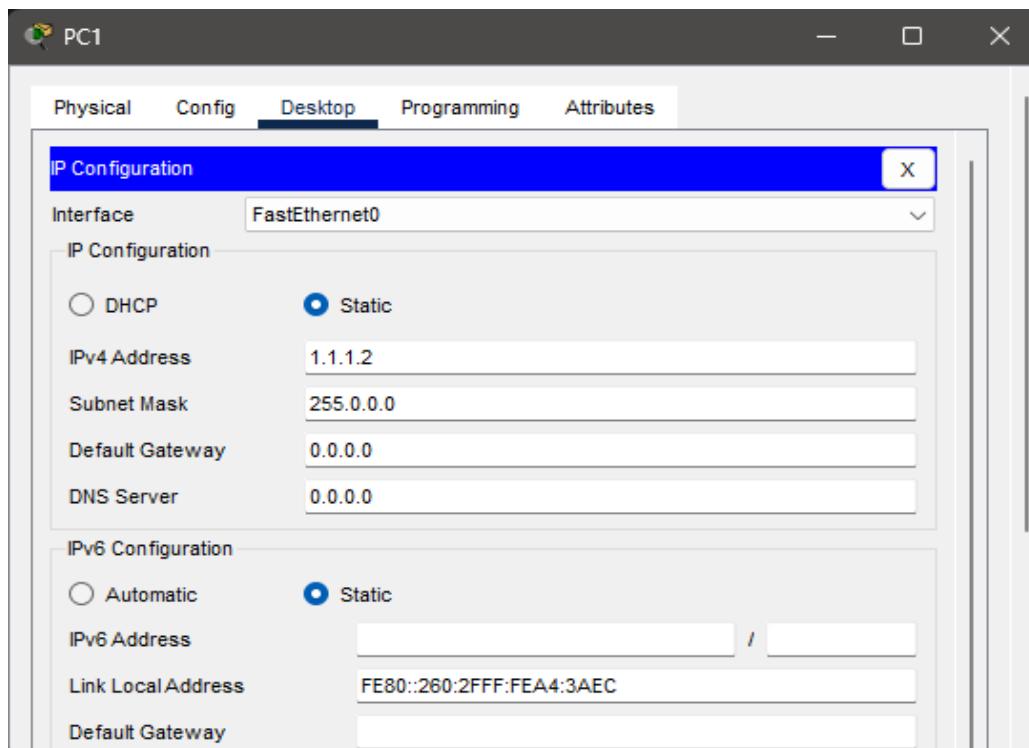


HOST PC IP ADDRESS :

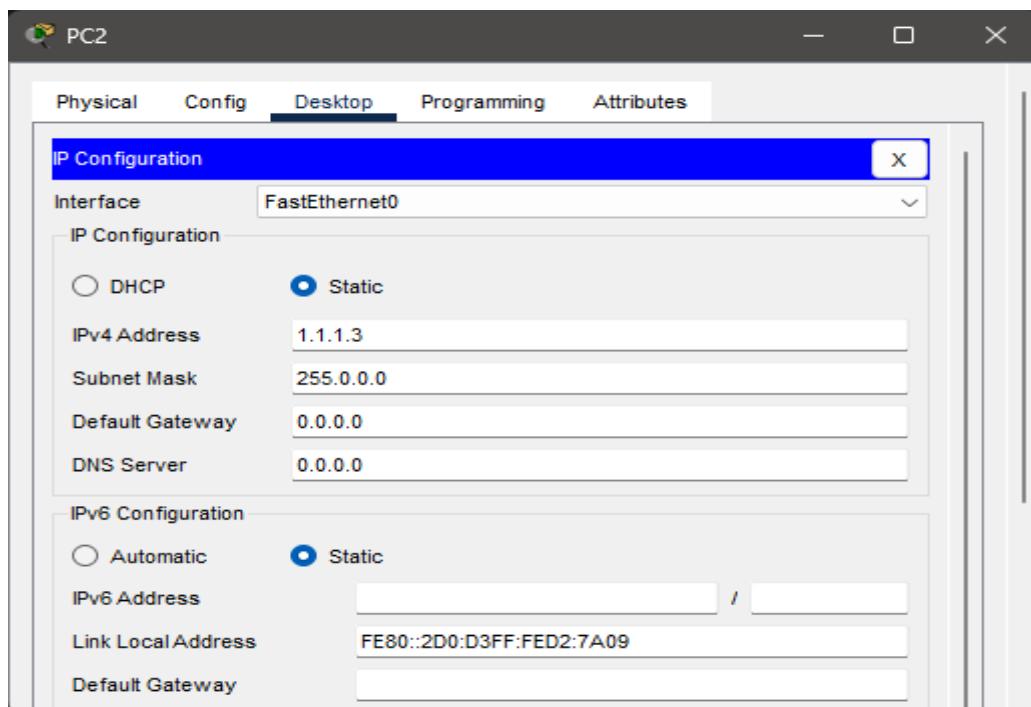
PC0 IP CONFIGURATION



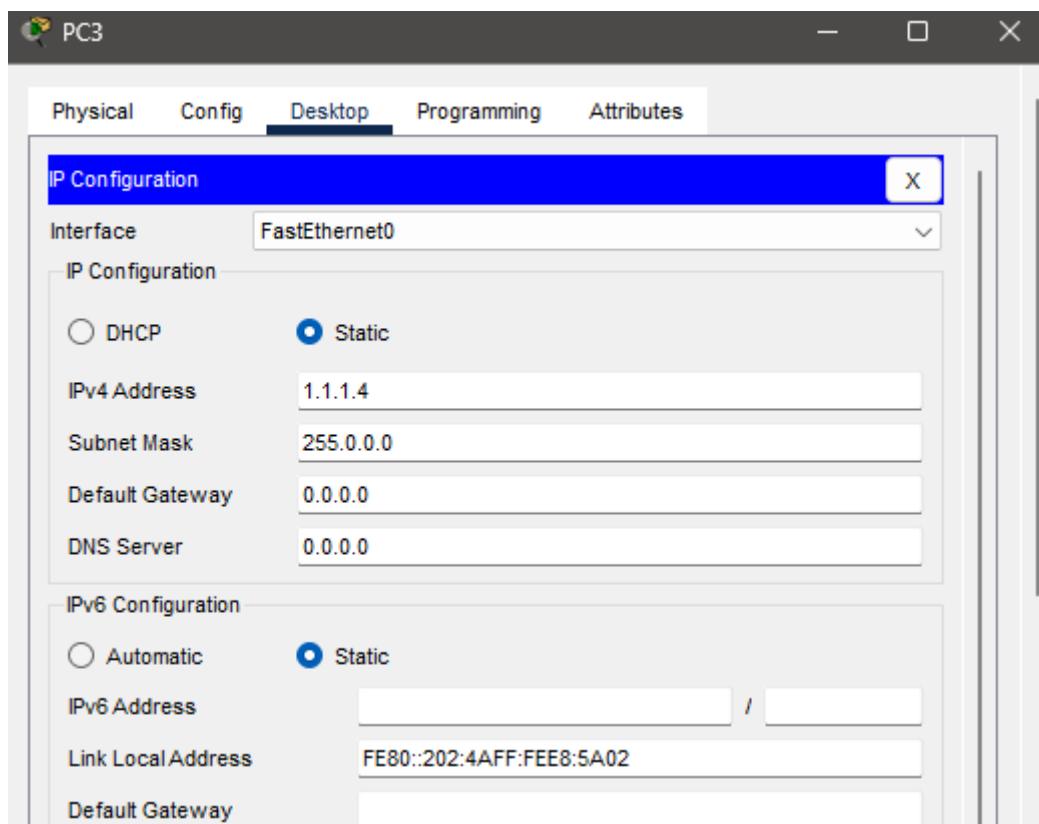
PC1 IP CONFIGURATION



PC2 IP CONFIGURATION



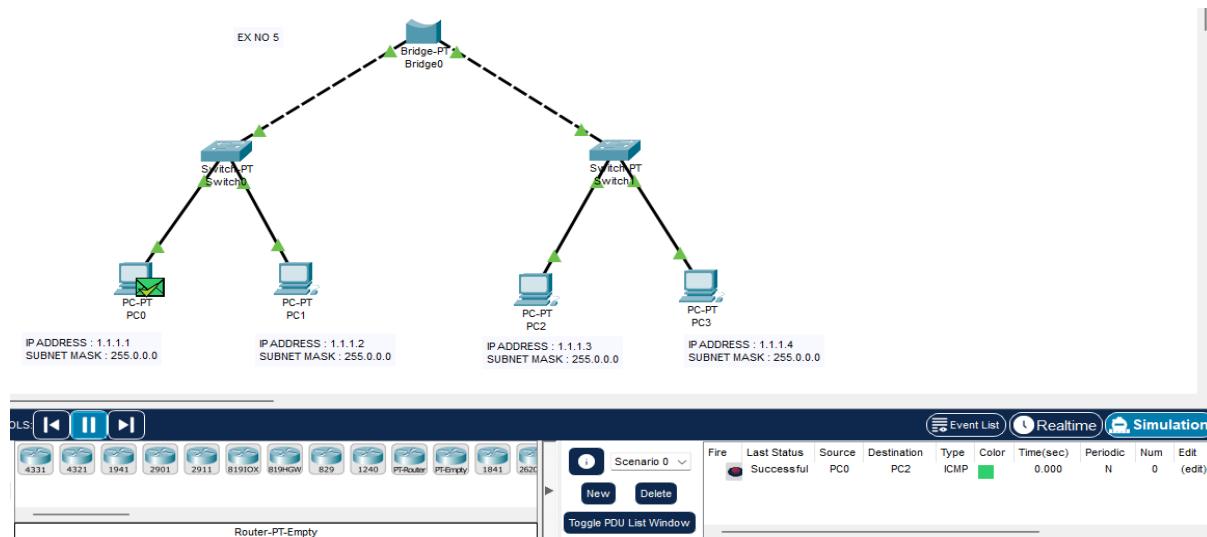
PC3 IP CONFIGURATION



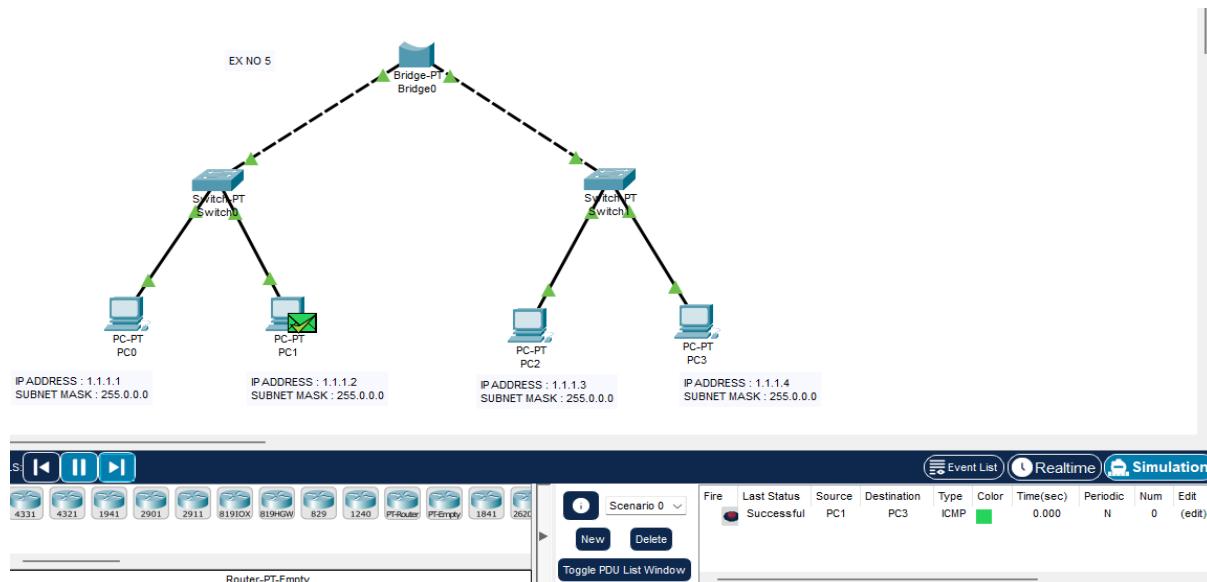
VERIFY LAN NETWORK CONNECTIVITY :

Using Add Simple PDU(p), Click the mail icon and then drop one mail to one of the PC in first lan and another mail to PC in another lan. If the resultant window shows the successful delivery, then network connectivity is successful.

HOST PC0 TO PC2



HOST PC1 TO PC3



RESULT :

Thus, two LANs are connected using Bridges and the communication between LANs is checked successfully.

EX.NO : 06

DATE : 06.02.24

DESIGNING RING AND MESH TOPOLOGIES USING CISCO PACKET TRACER

AIM :

To Designing a Ring and Mesh topologies by using Cisco Packet Tracer.

REQUIREMENTS :

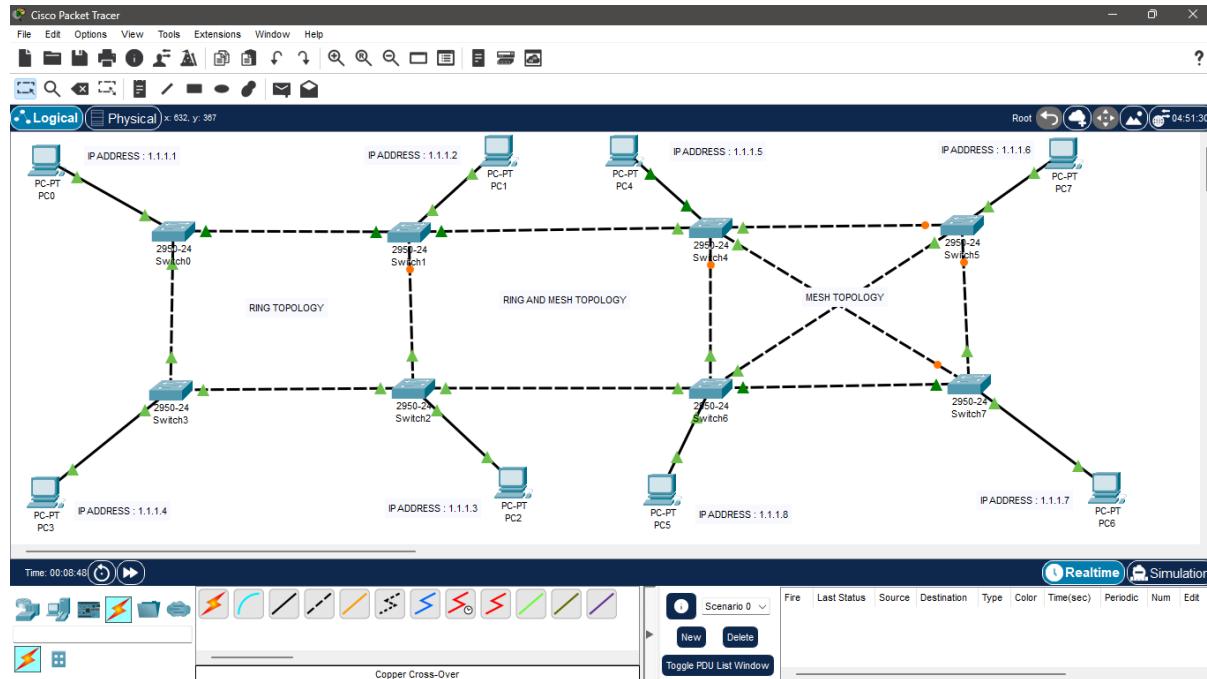
- 8 windows PC.
- 8 Switch (2950-24).
- 8 Straight Line LAN Cables.
- 12 Cross Over Cables.
- Cisco Packet Tracer.

PROCEDURES :

- Open CISCO PACKET TRACER software.
- Draw the 8 PC using END Device Icons.
- Draw the 8 Cisco 2950-24 Switch Using Switch icon lists.
- Make the Connections using Copper-Straight-Through Ethernet Cables.
- Make the Connections between Switches using Cross Overs Cables.
- Enter the IP Address To Each Machine.
- Check the Network Connections using Add Simple PDU(P).

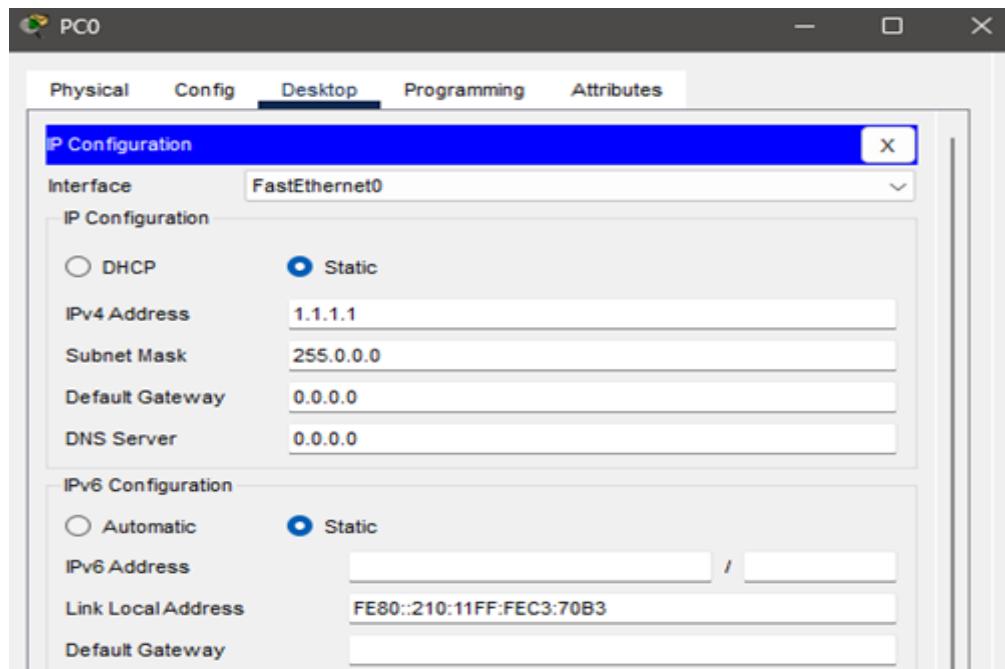
NETWORK TOPOLOGY

RING AND MESH TOPOLOGY

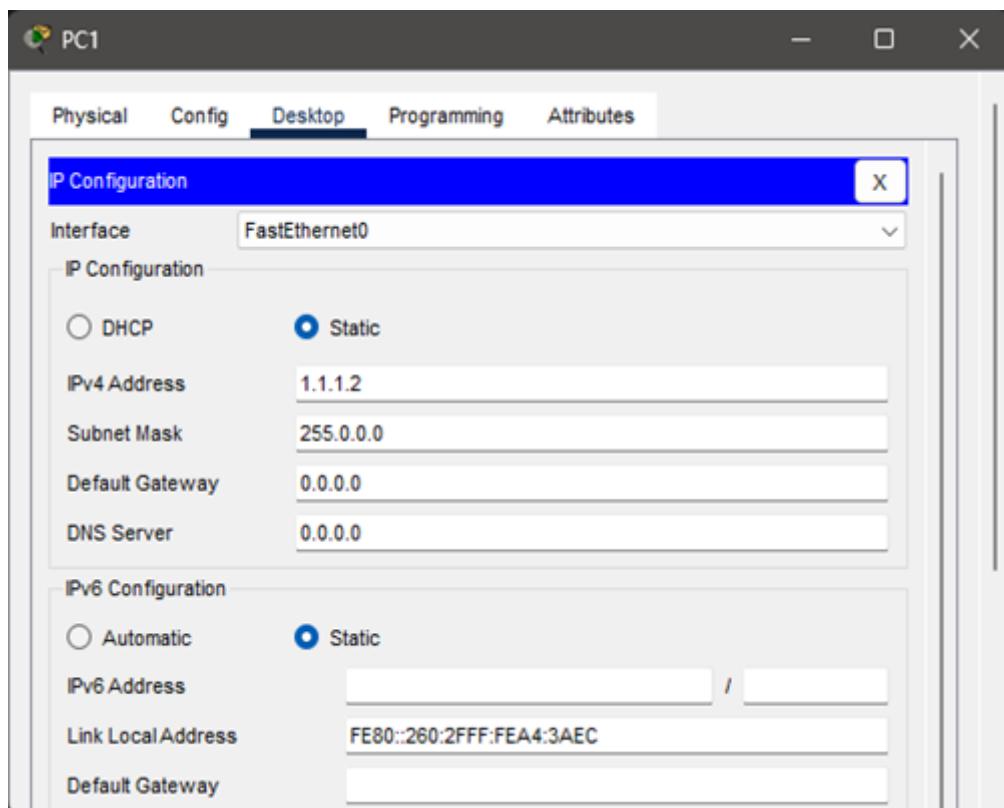


HOST PC IP ADDRESS :

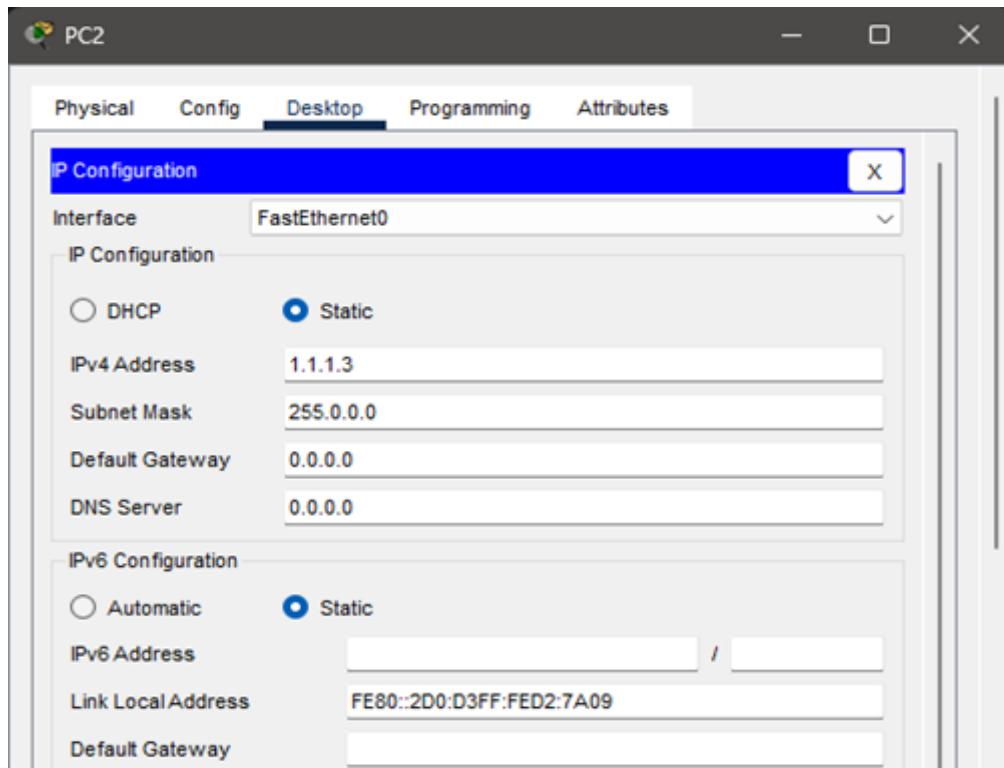
PC0 IP CONFIGURATION



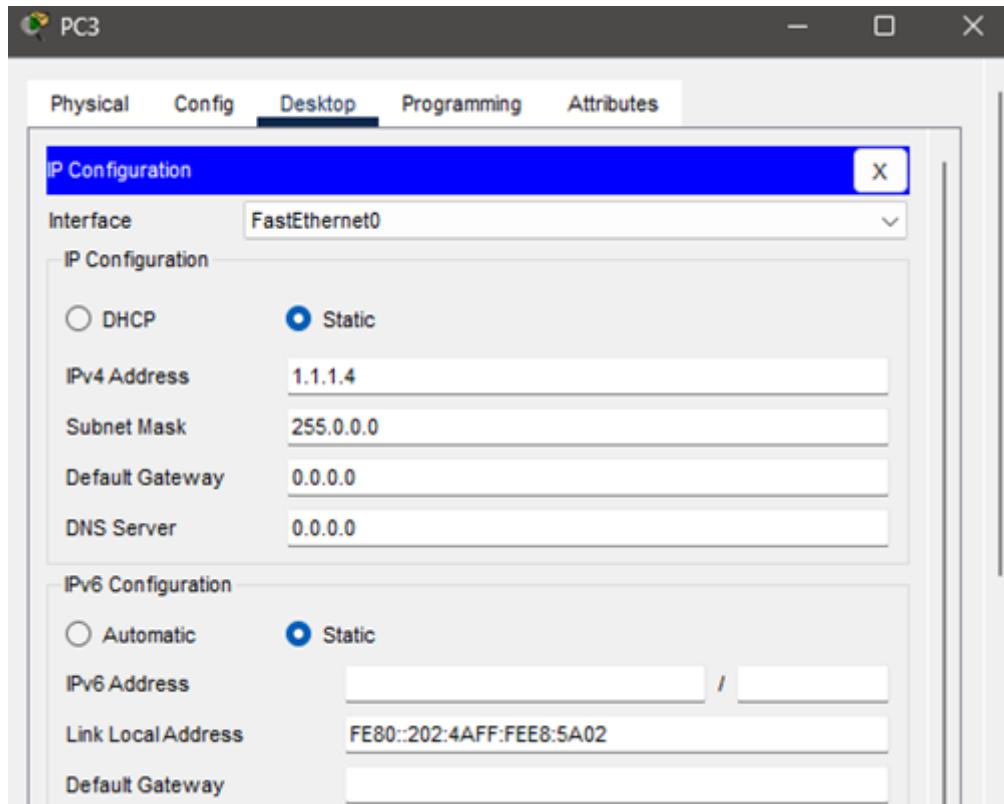
PC1 IP CONFIGURATION



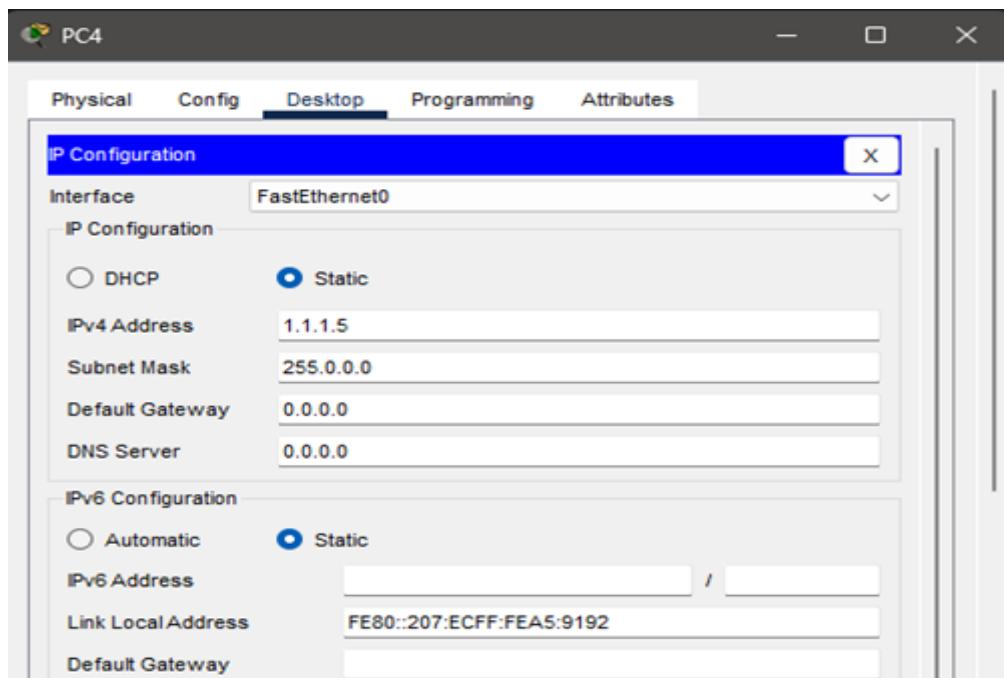
PC2 IP CONFIGURATION



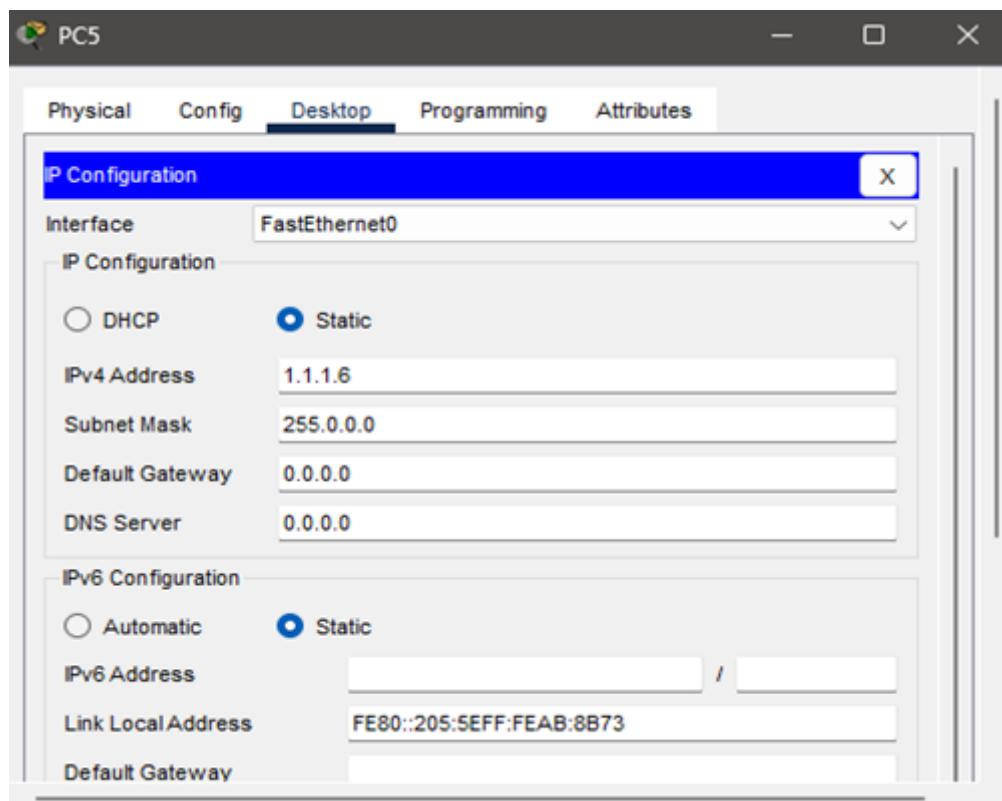
PC3 IP CONFIGURATION



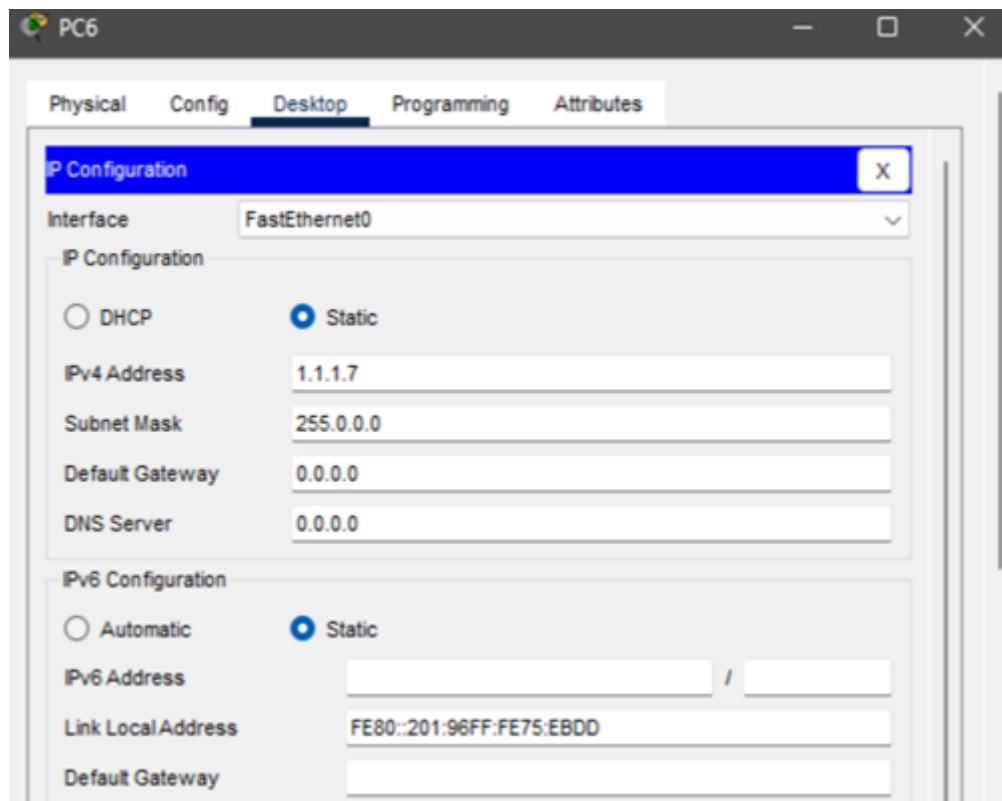
PC4 IP CONFIGURATION



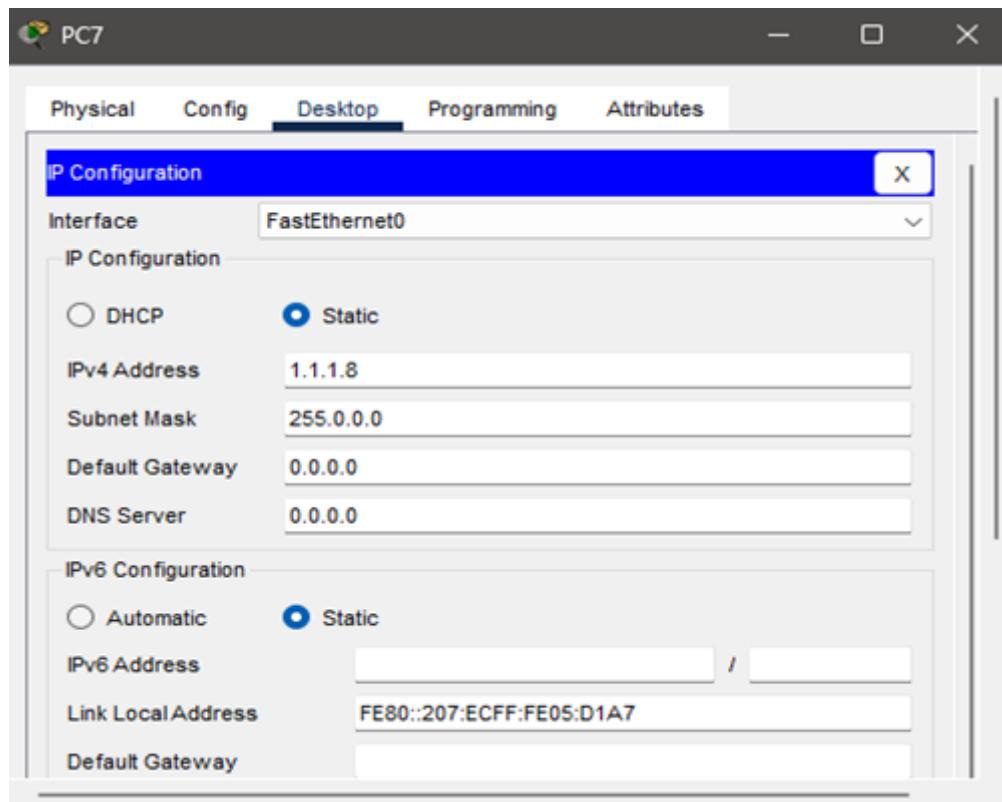
PC5 IP CONFIGURATION



PC6 IP CONFIGURATION



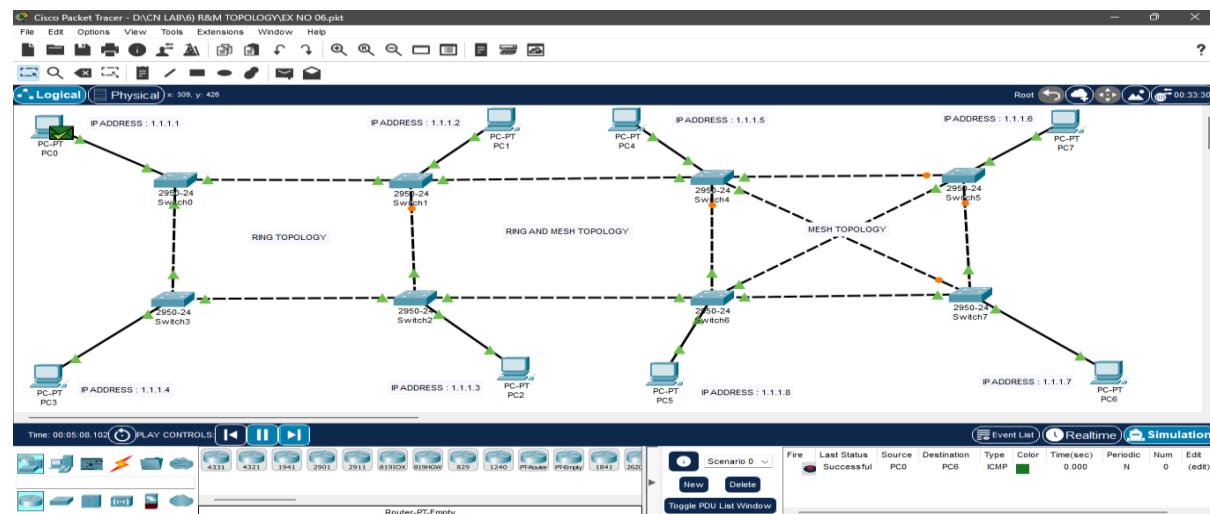
PC7 IP CONFIGURATION



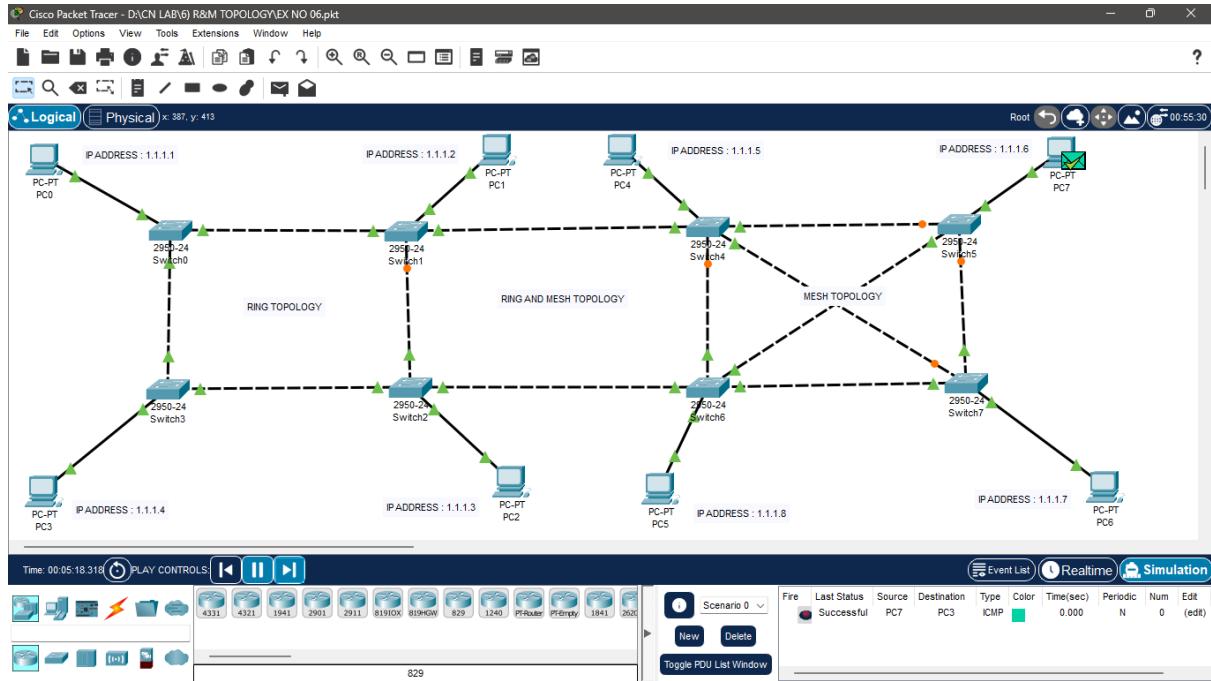
VERIFY LAN NETWORK CONNECTIVITY :

Using Add Simple PDU(p), Click the mail icon and then drop one mail to one of the PC in first lan and another mail to PC in another lan. If the resultant window shows the successful delivery, then network connectivity is successful.

HOST PC0 TO PC6



HOST PC7 TO PC3



RESULT :

Thus, Ring and Mesh topologies are designed using cisco packet tracer and the communication between Ring and Mesh topologies is checked successfully.

EX.NO : 07

DATE : 09.02.24

DESIGNING BUS AND STAR TOPOLOGIES USING CISCO PACKET TRACER

AIM :

To Designing a Bus and Star topologies by using Cisco Packet Tracer.

REQUIREMENTS :

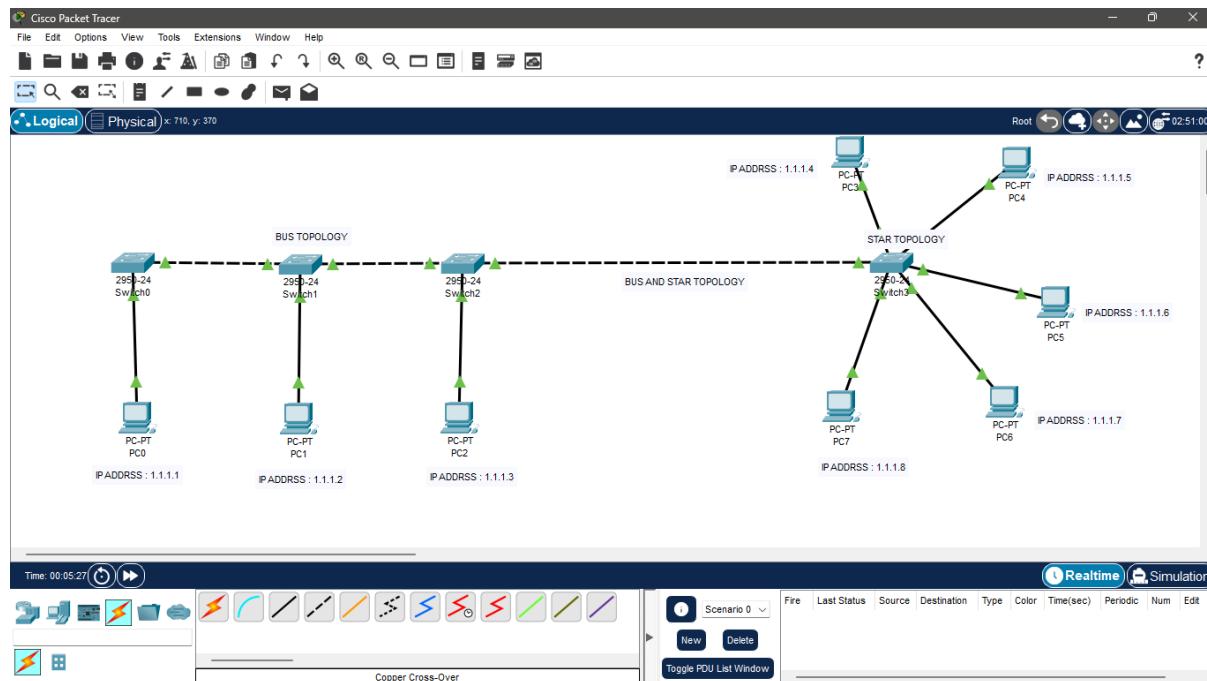
- 8 windows PC.
- 4 Switch (2950-24).
- 8 Straight Line LAN Cables.
- 3 Cross Over Cables.
- Cisco Packet Tracer.

PROCEDURES :

- Open CISCO PACKET TRACER software.
- Draw the 8 PC using END Device Icons.
- Draw the 4 Cisco 2950-24 Switch Using Switch icon lists.
- Make the Connections using Copper-Straight-Through Ethernet Cables.
- Make the Connections between Switches using Cross Overs Cables.
- Enter the IP Address To Each Machine.
- Check the Network Connections using Add Simple PDU(P).

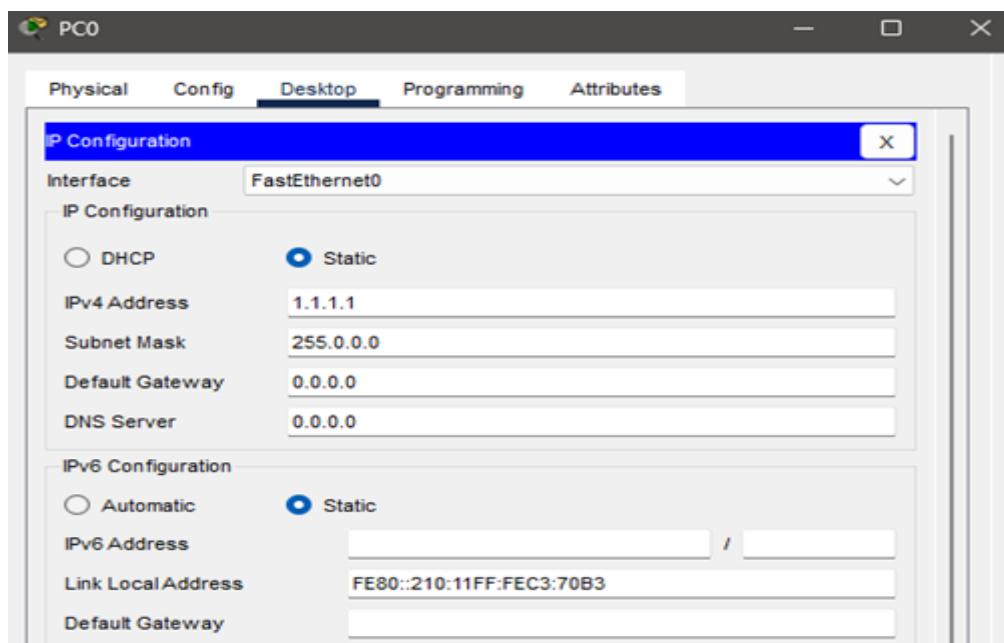
NETWORK TOPOLOGY

BUS AND STAR TOPOLOGY

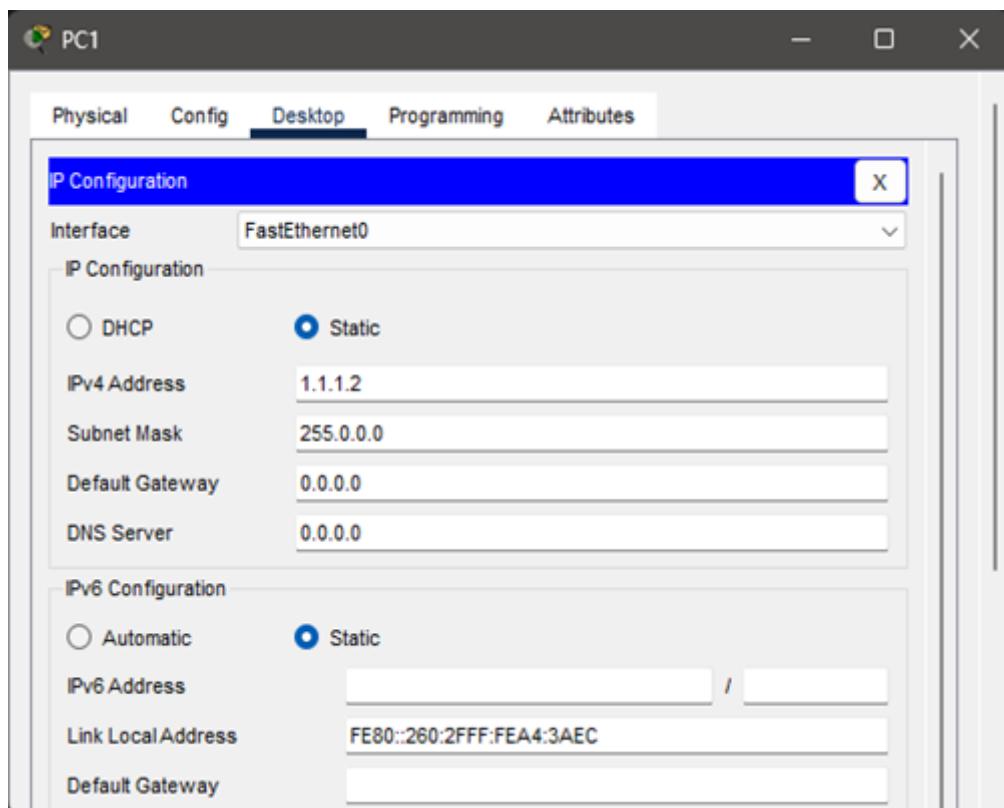


HOST PC IP ADDRESS :

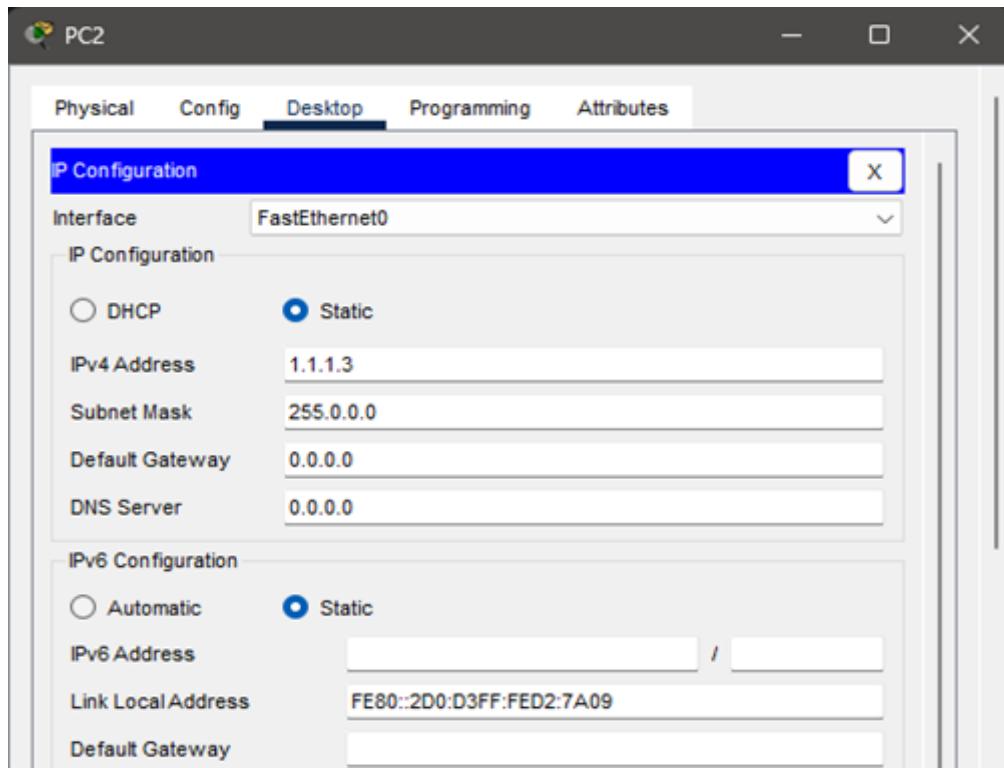
PC0 IP CONFIGURATION



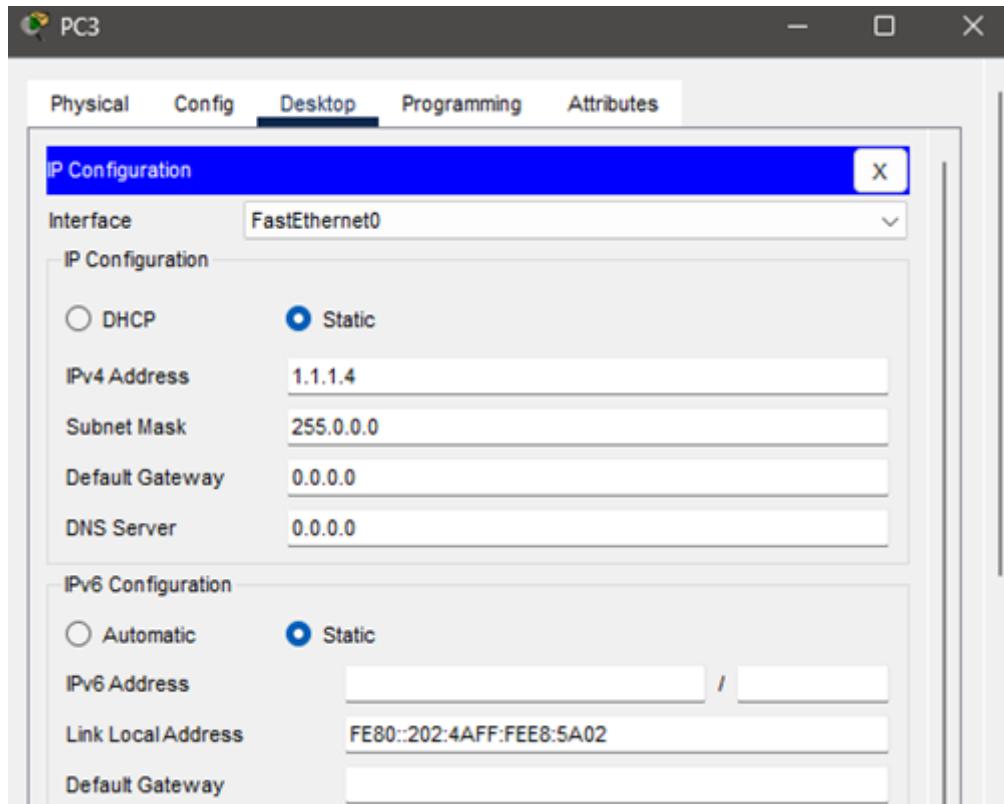
PC1 IP CONFIGURATION



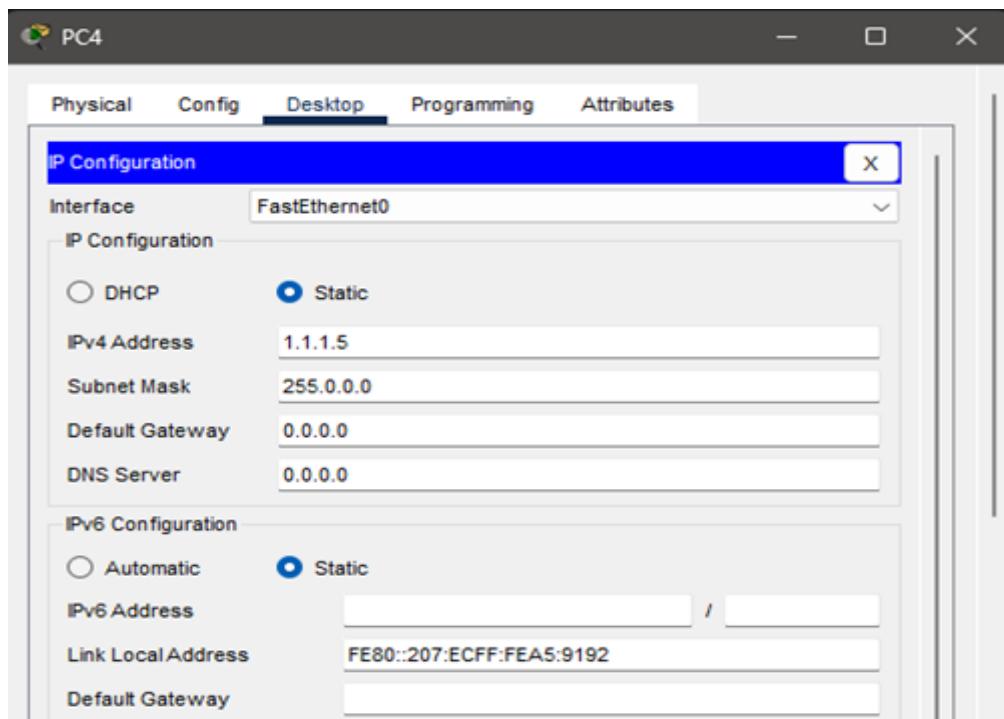
PC2 IP CONFIGURATION



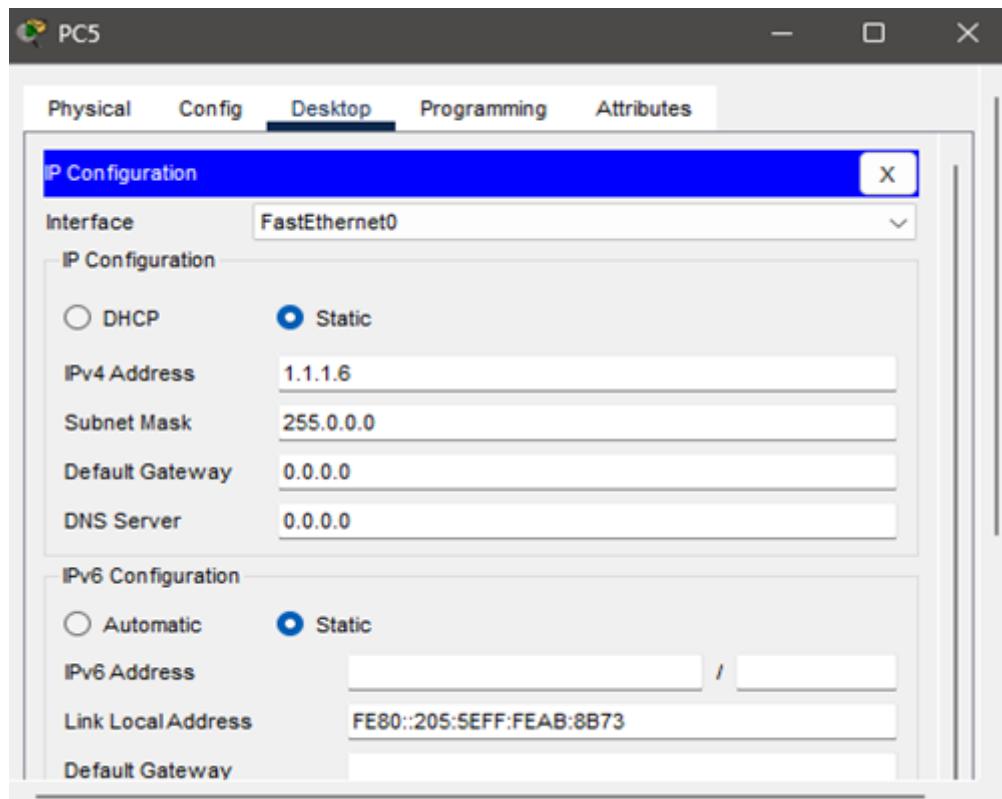
PC3 IP CONFIGURATION



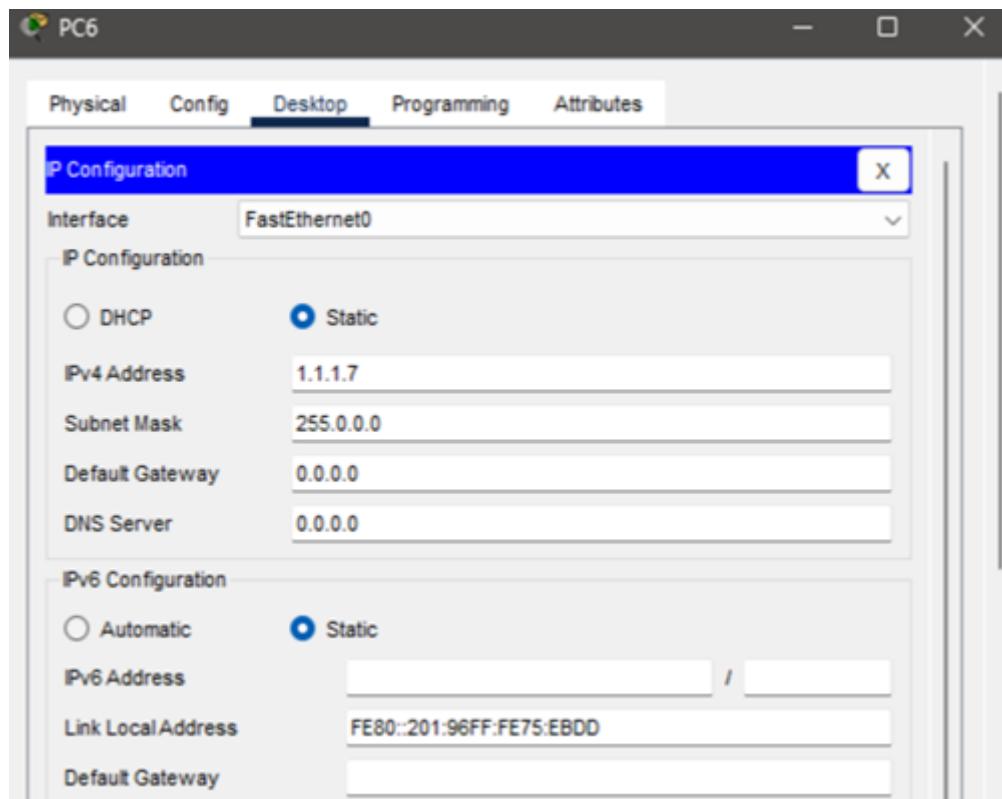
PC4 IP CONFIGURATION



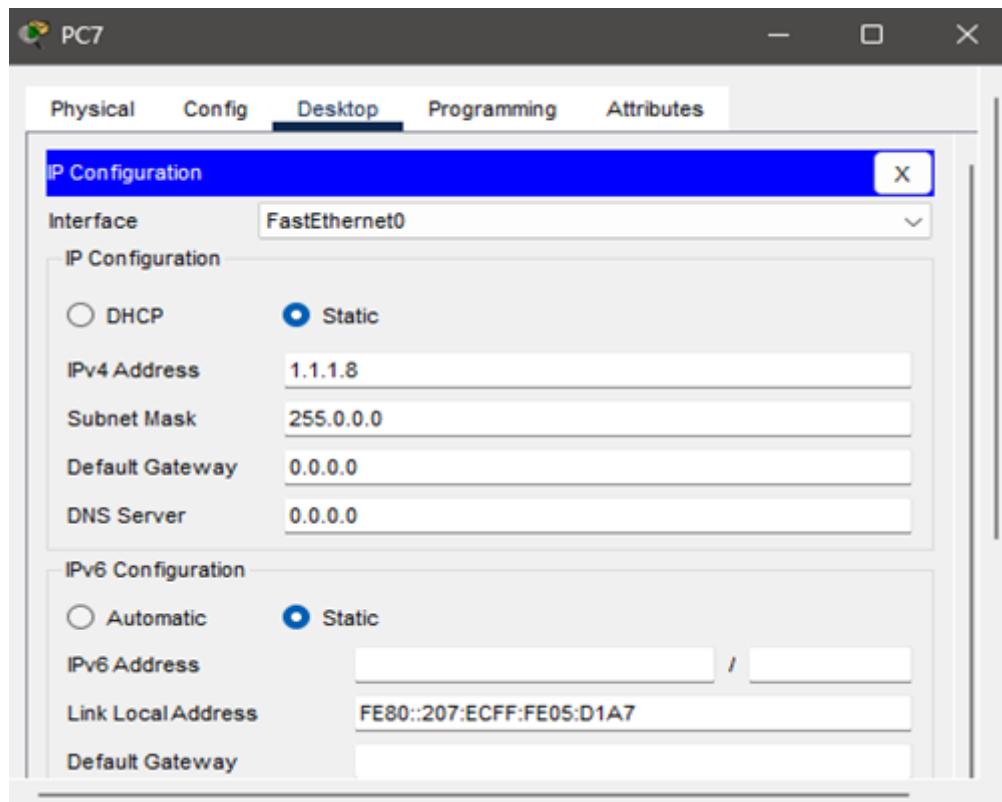
PC5 IP CONFIGURATION



PC6 IP CONFIGURATION



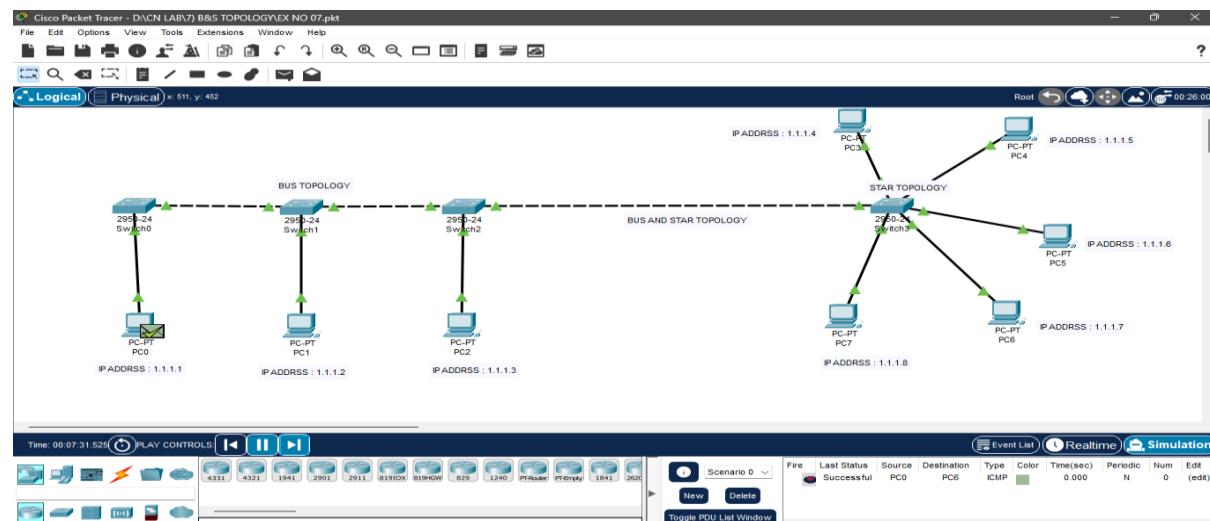
PC7 IP CONFIGURATION



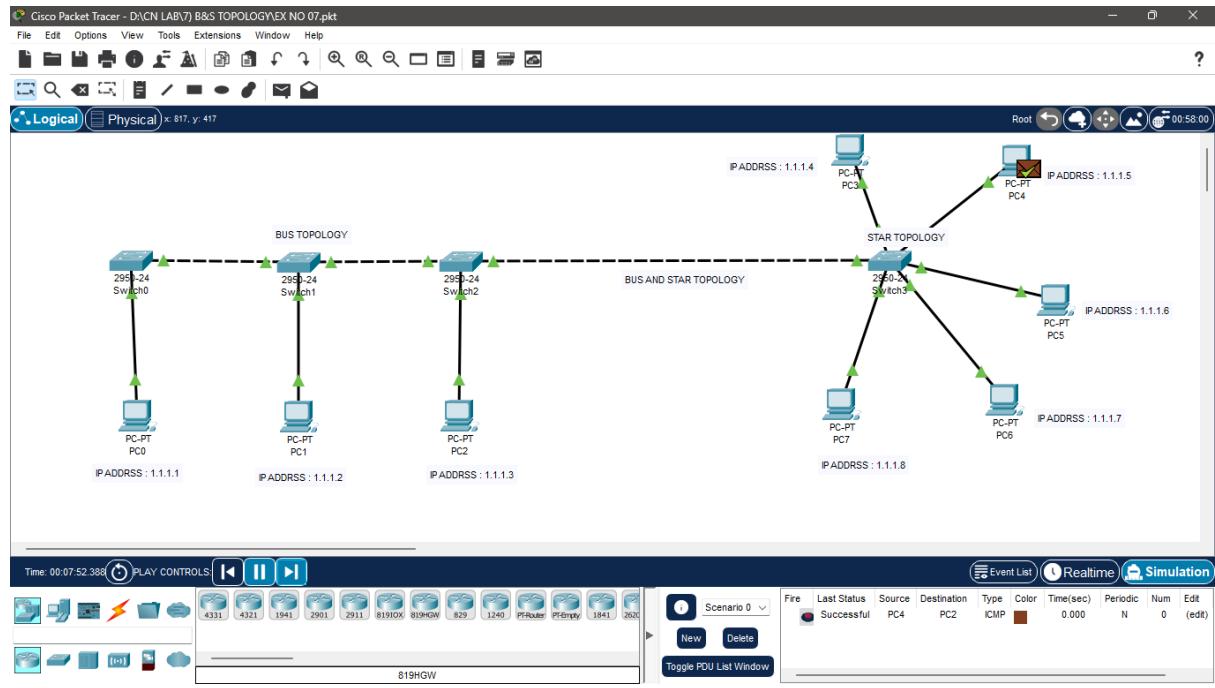
VERIFY LAN NETWORK CONNECTIVITY :

Using Add Simple PDU(p), Click the mail icon and then drop one mail to one of the PC in first lan and another mail to PC in another lan. If the resultant window shows the successful delivery, then network connectivity is successful.

HOST PC0 TO PC6



HOST PC4 TO PC2



RESULT :

Thus, Bus and Star topologies are designed using cisco packet tracer and the communication between Bus and Star topologies is checked successfully.

EX.NO : 08

DATE : 13.02.24

DESIGNING HYBRID TOPOLOGIES USING CISCO PACKET TRACER

AIM :

To Designing a Hybrid topologies by using Cisco Packet Tracer.

REQUIREMENTS :

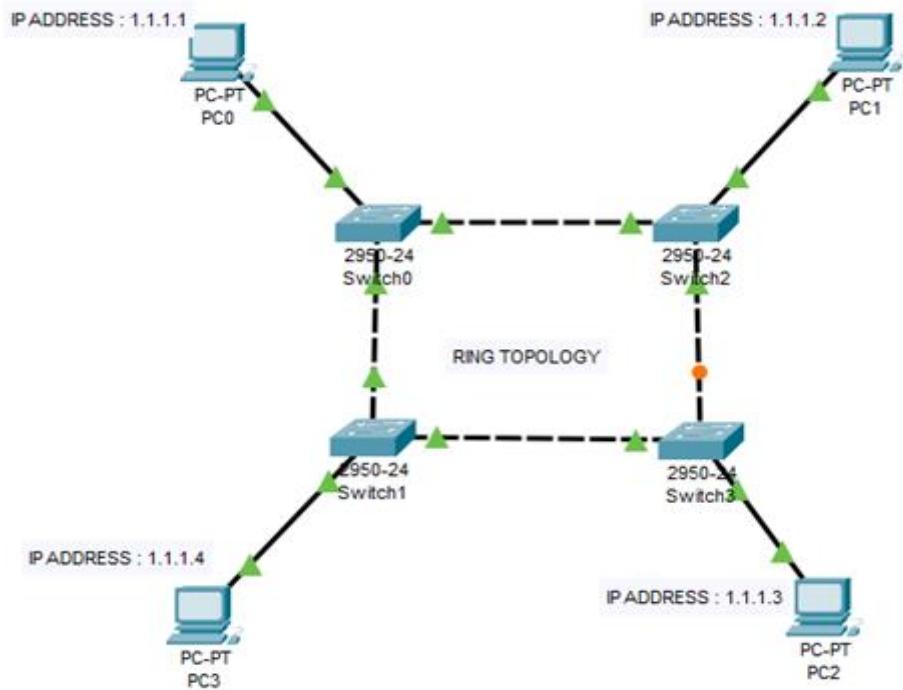
- 16 windows PC.
- 12 Switch (2950-24).
- 16 Straight Line LAN Cables.
- 13 Cross Over Cables.
- Cisco Packet Tracer.

PROCEDURES :

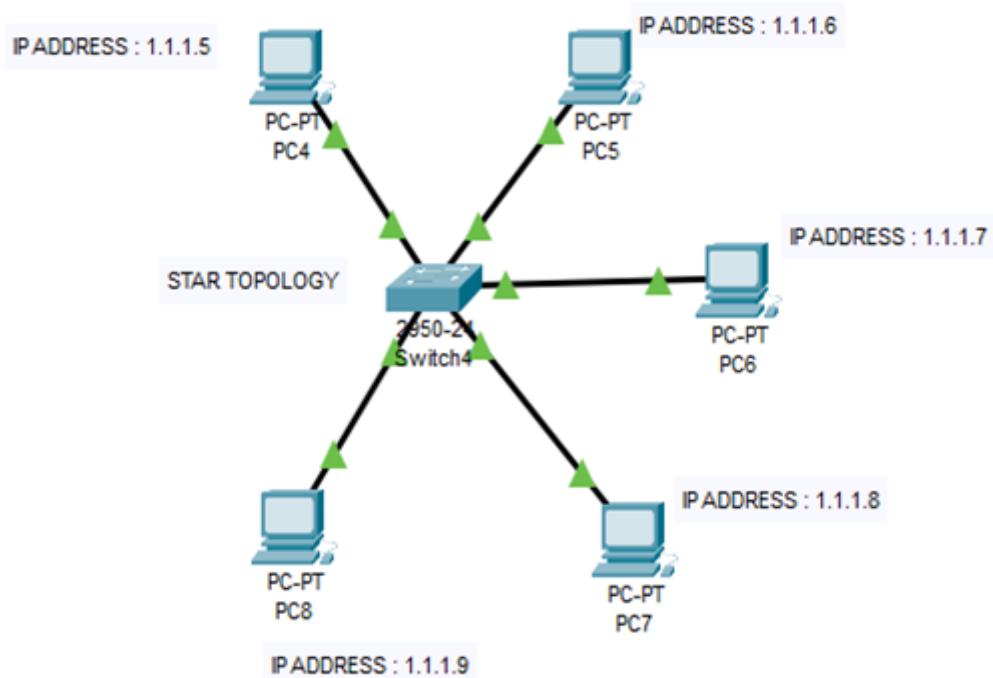
- Open CISCO PACKET TRACER software.
- Draw the 16 PC using END Device Icons.
- Draw the 12 Cisco 2950-24 Switch Using Switch icon lists.
- Make the Connections using Copper-Straight-Through Ethernet Cables.
- Make the Connections between Switches using Cross Overs Cables.
- Enter the IP Address To Each Machine.
- Check the Network Connections using Add Simple PDU(P).

NETWORK TOPOLOGY

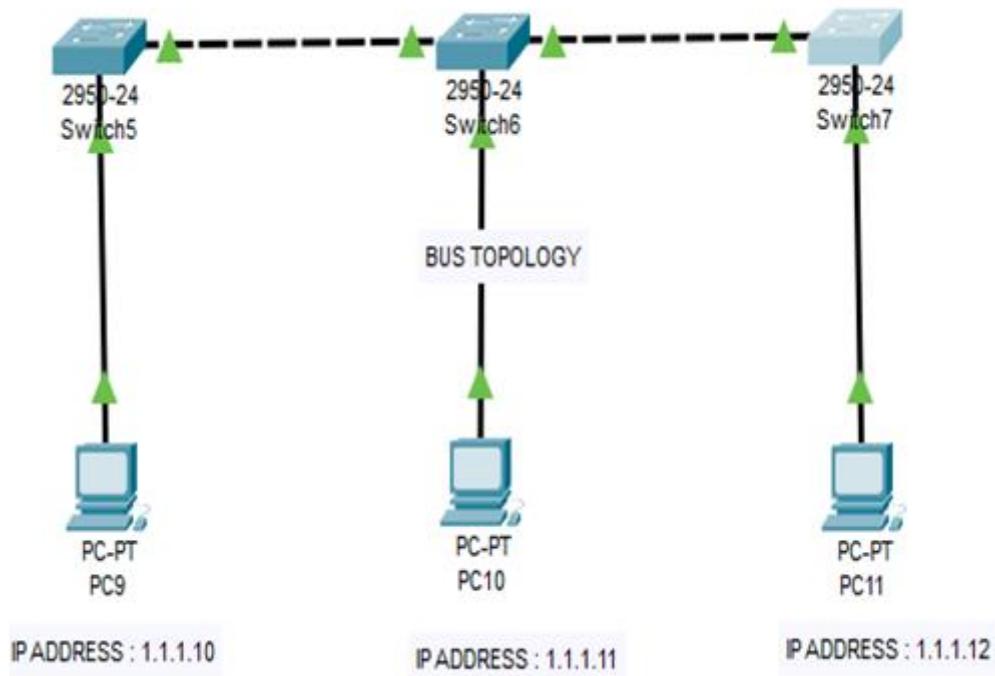
RING TOPOLOGY



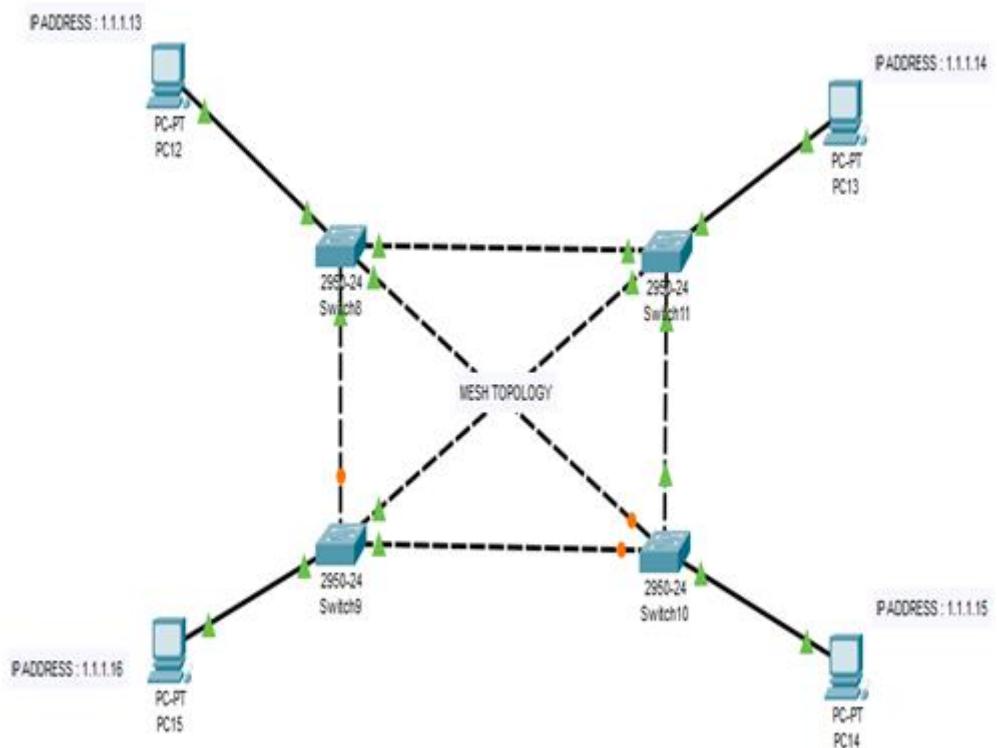
STAR TOPOLOGY



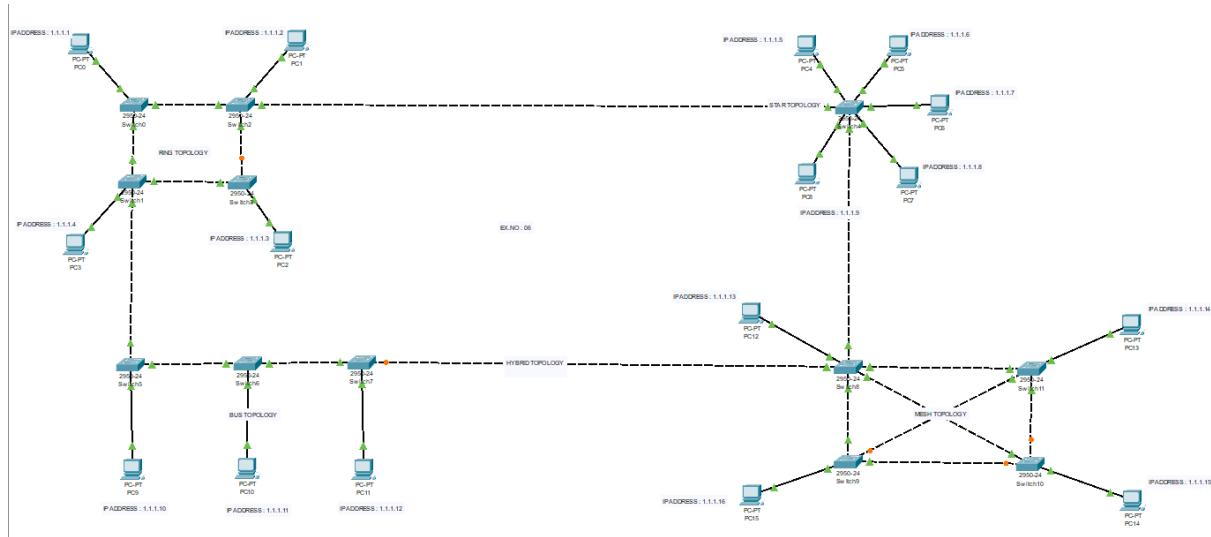
BUS TOPOLOGY



MESH TOPOLOGY

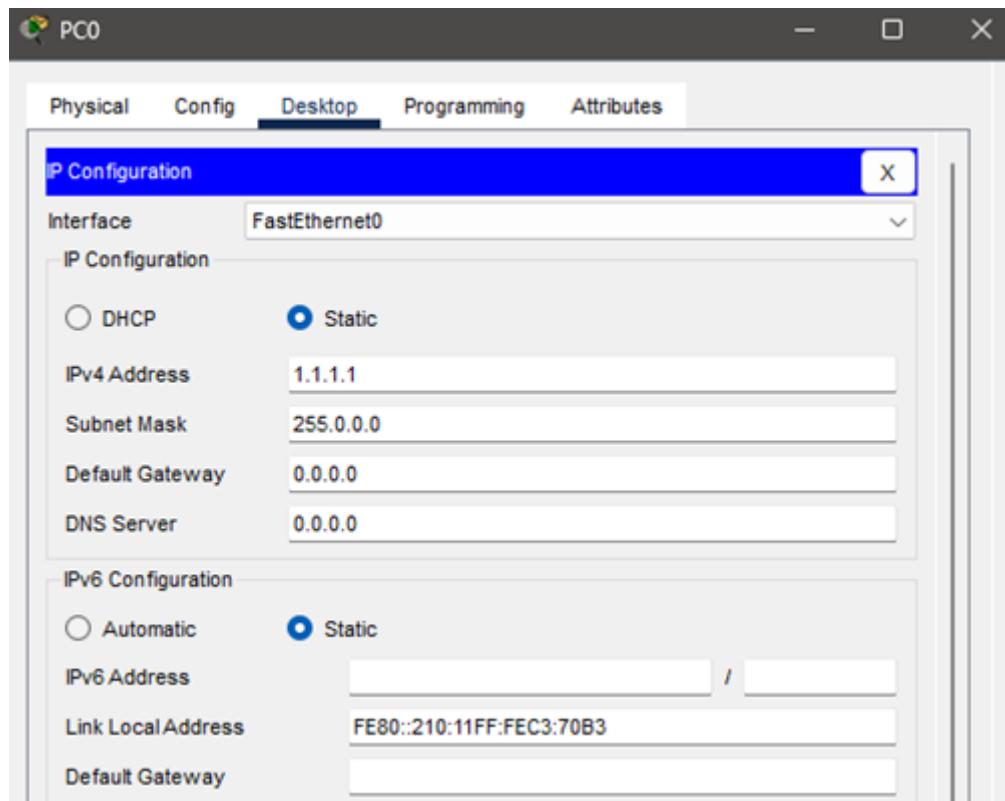


HYBRID TOPOLOGY

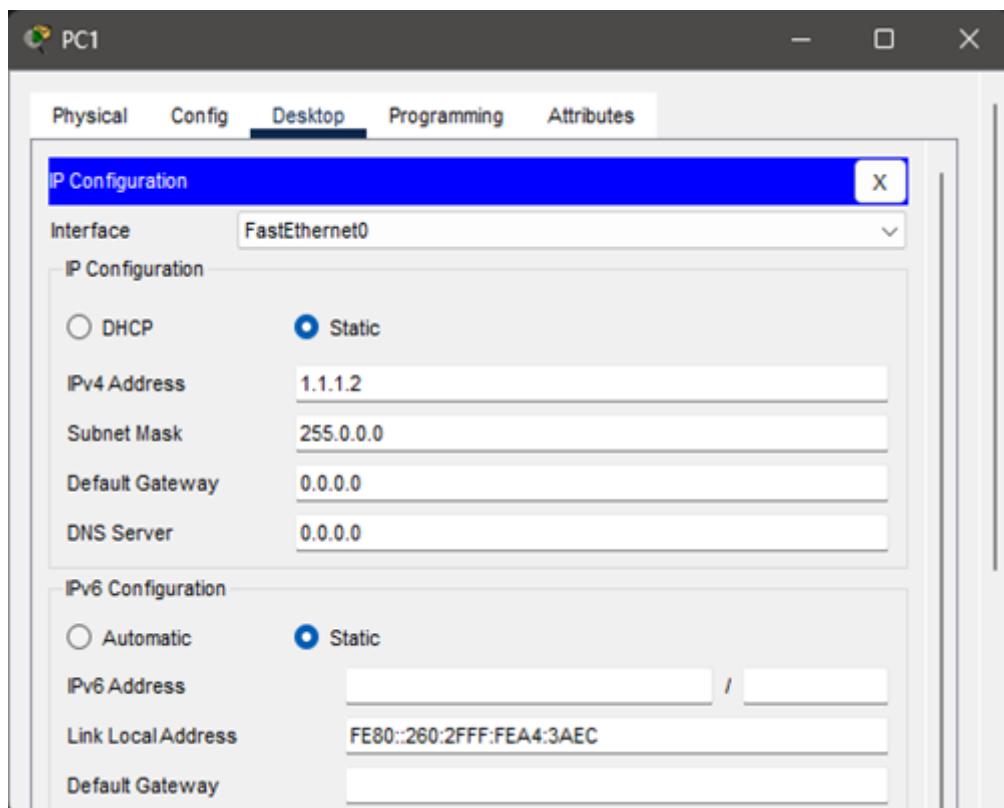


HOST PC IP ADDRESS :

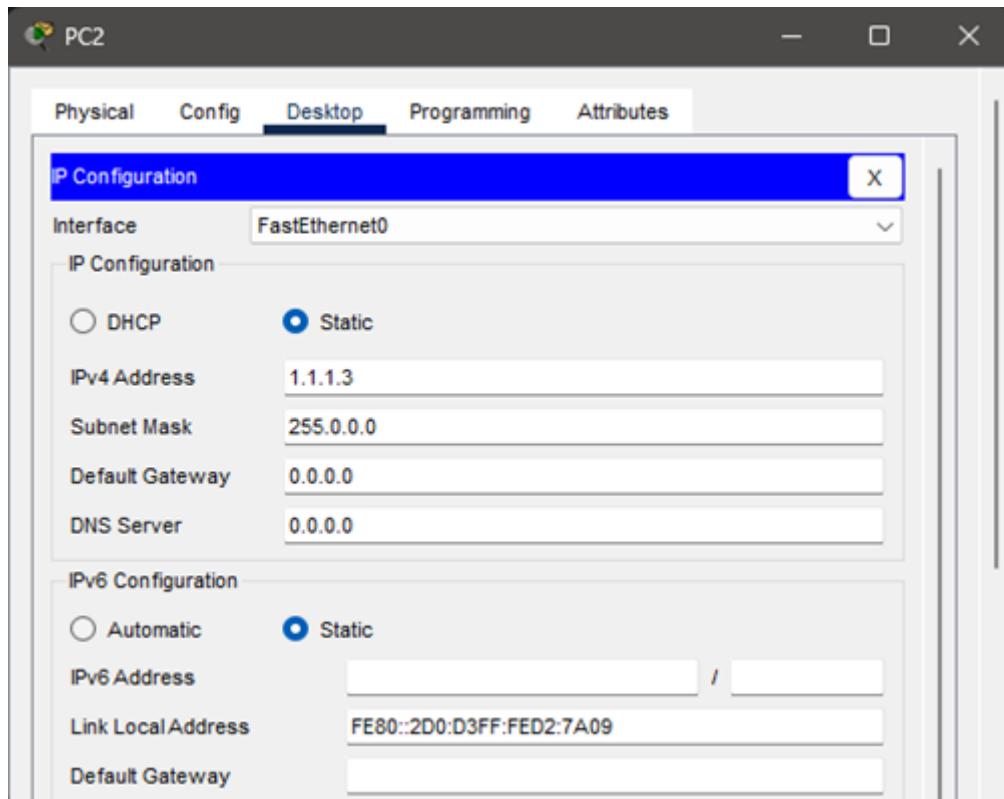
PC0 IP CONFIGURATION



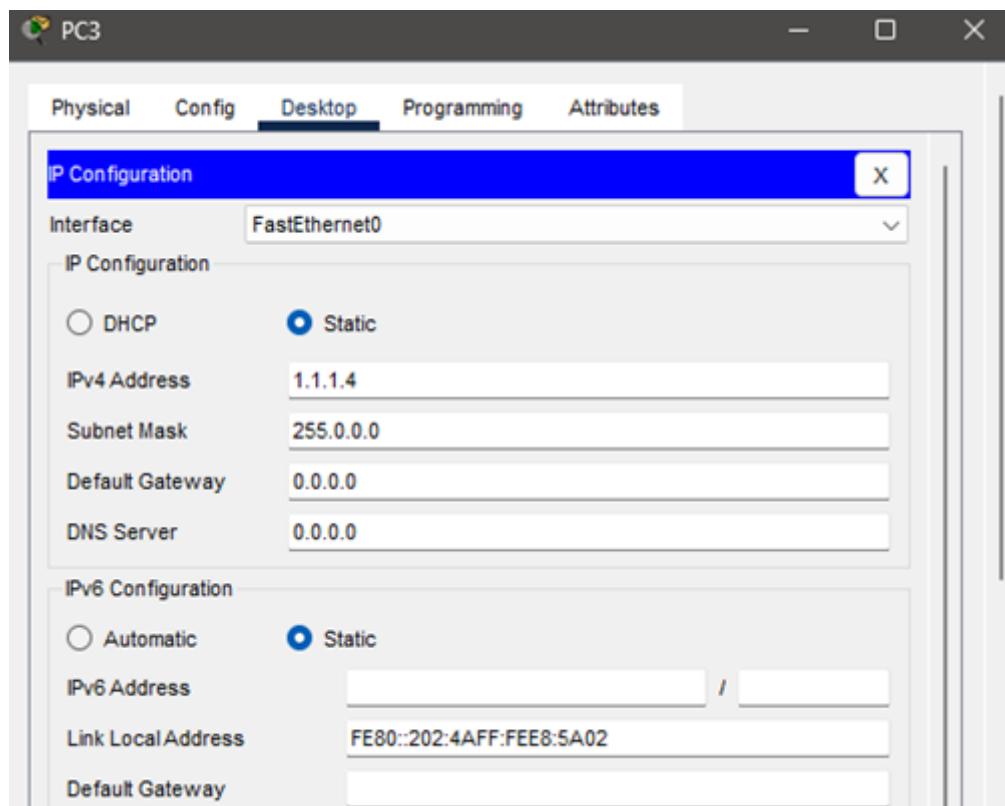
PC1 IP CONFIGURATION



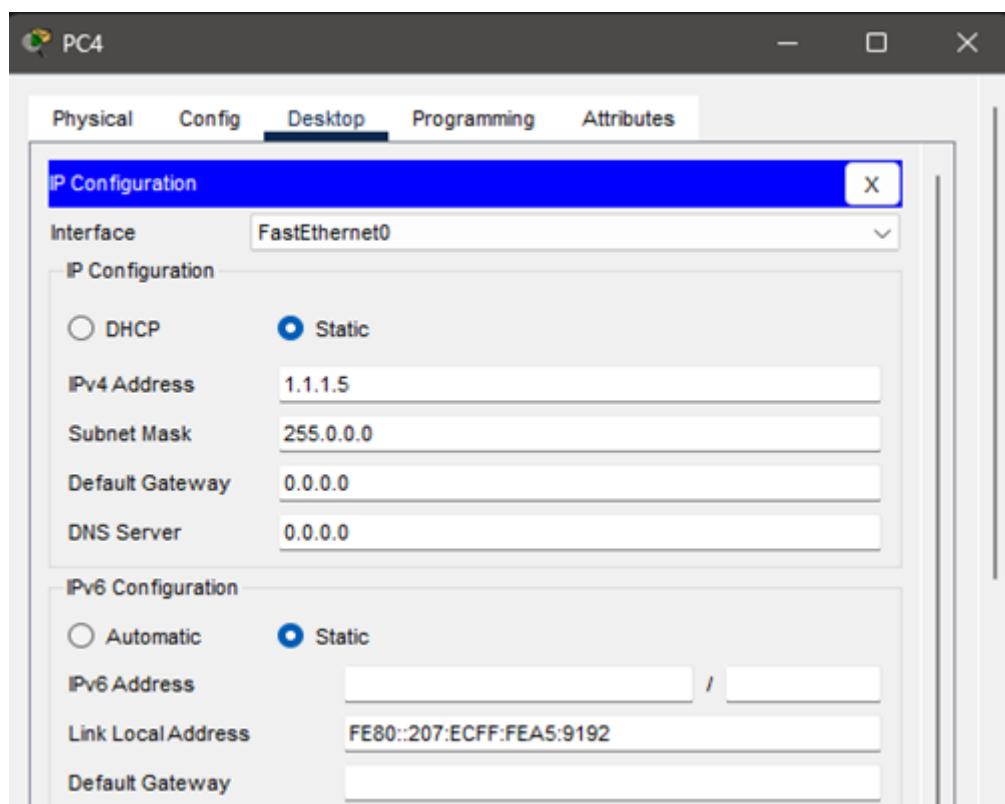
PC2 IP CONFIGURATION



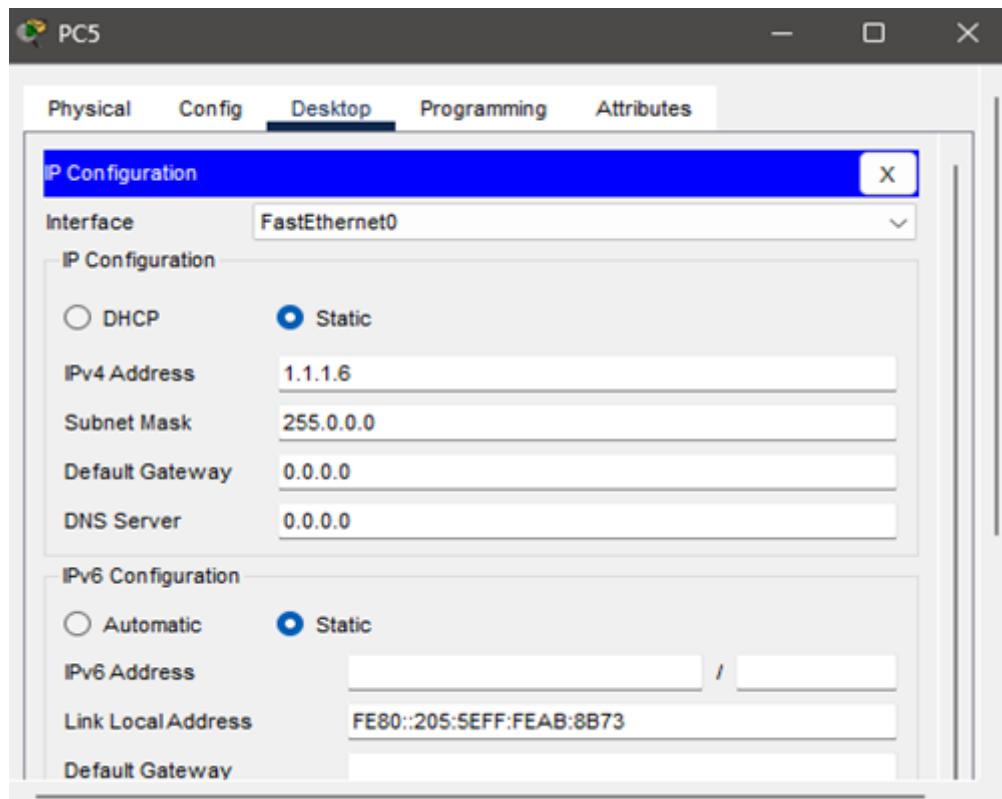
PC3 IP CONFIGURATION



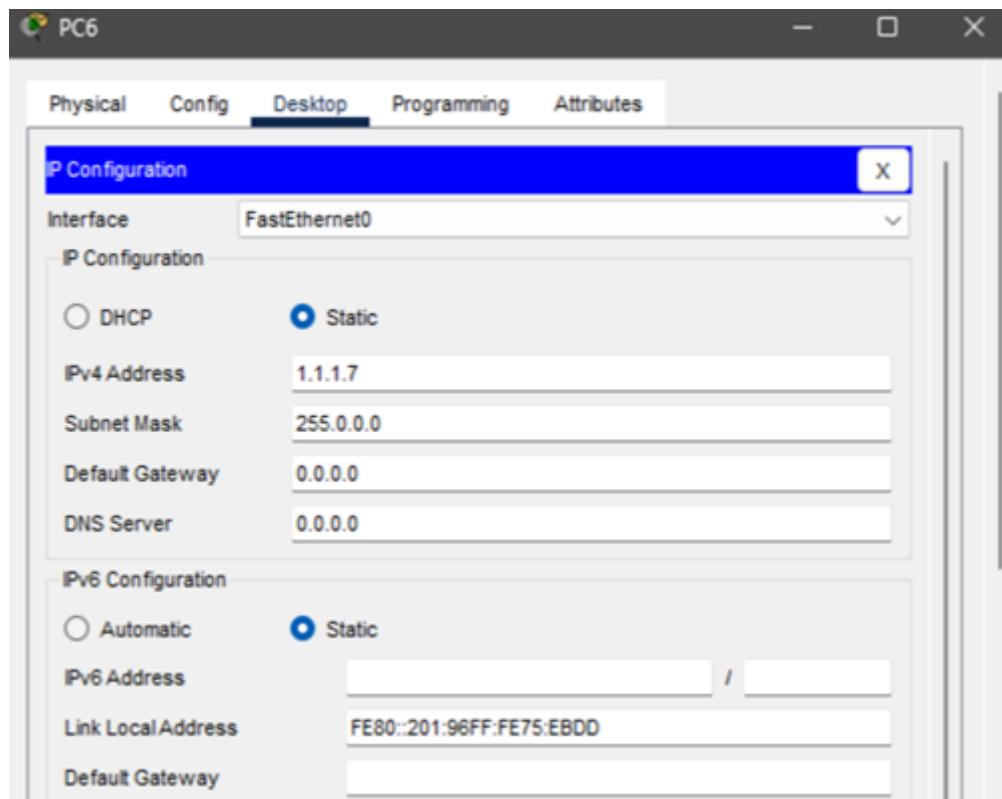
PC4 IP CONFIGURATION



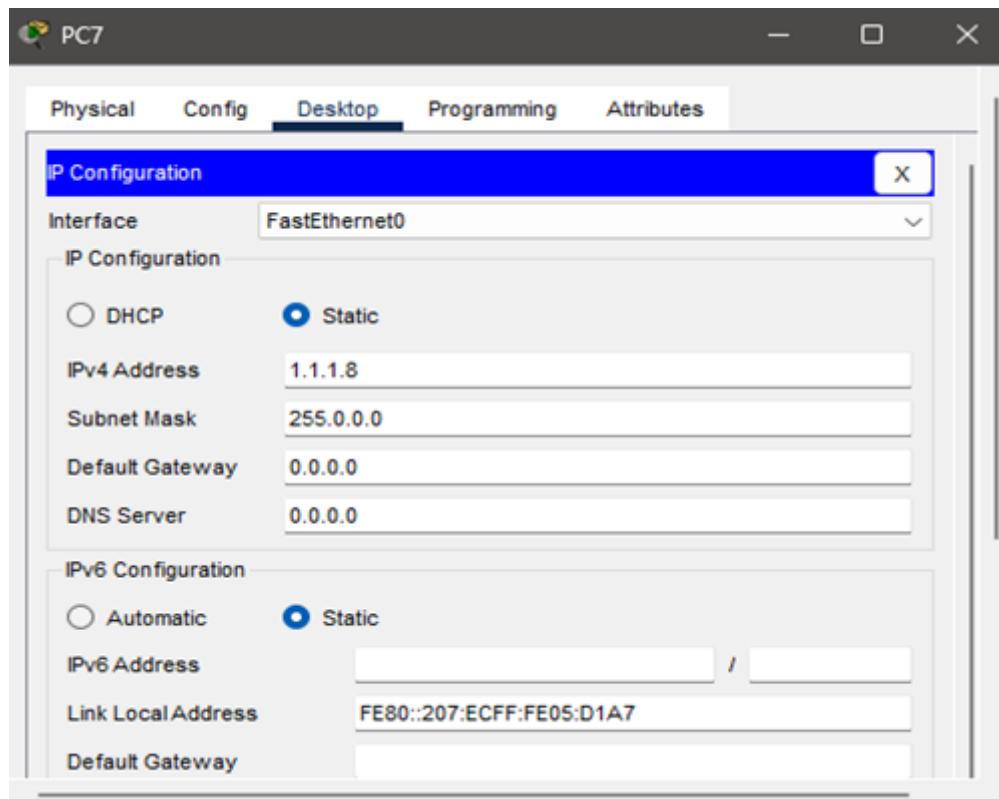
PC5 IP CONFIGURATION



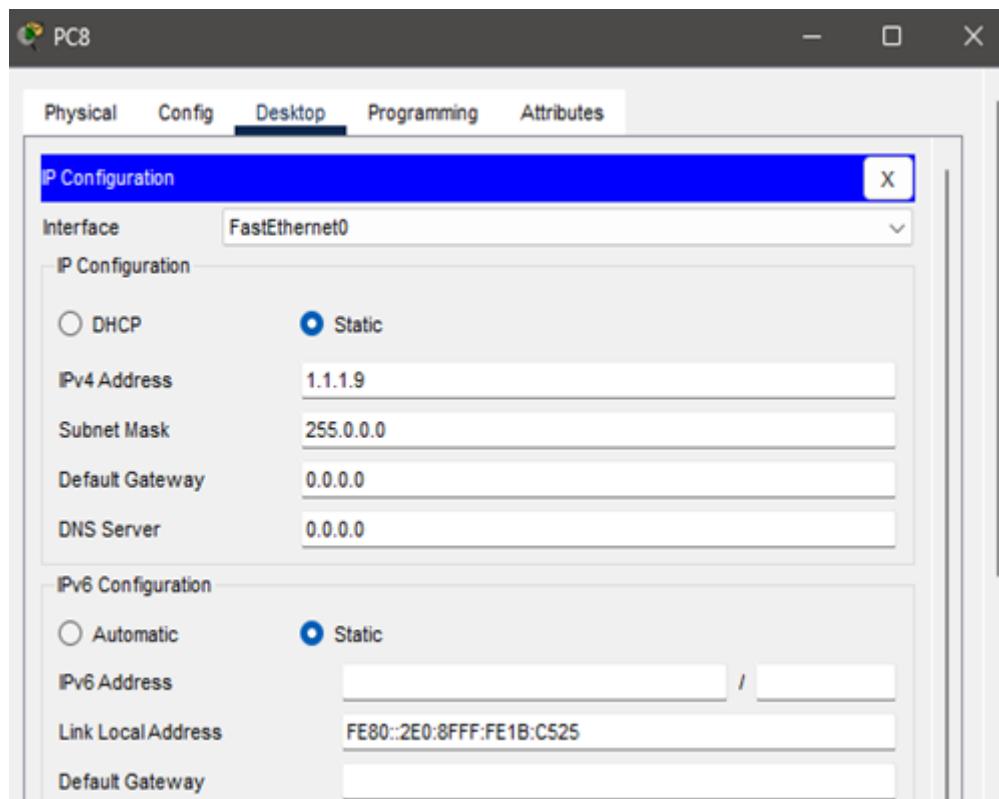
PC6 IP CONFIGURATION



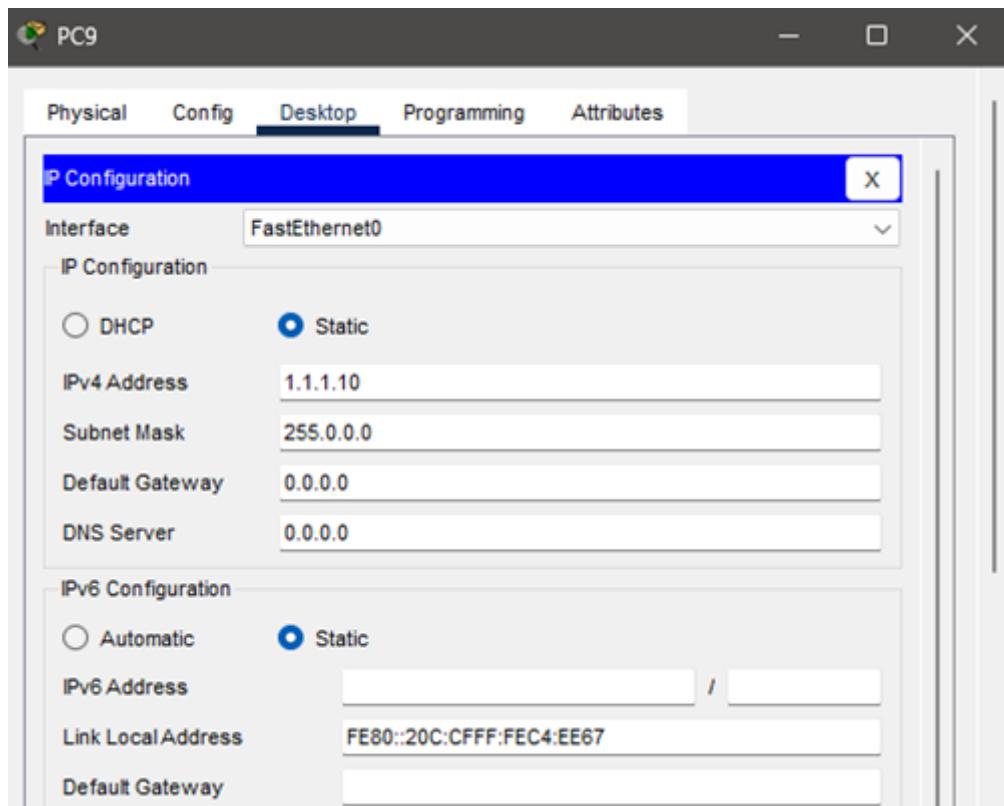
PC7 IP CONFIGURATION



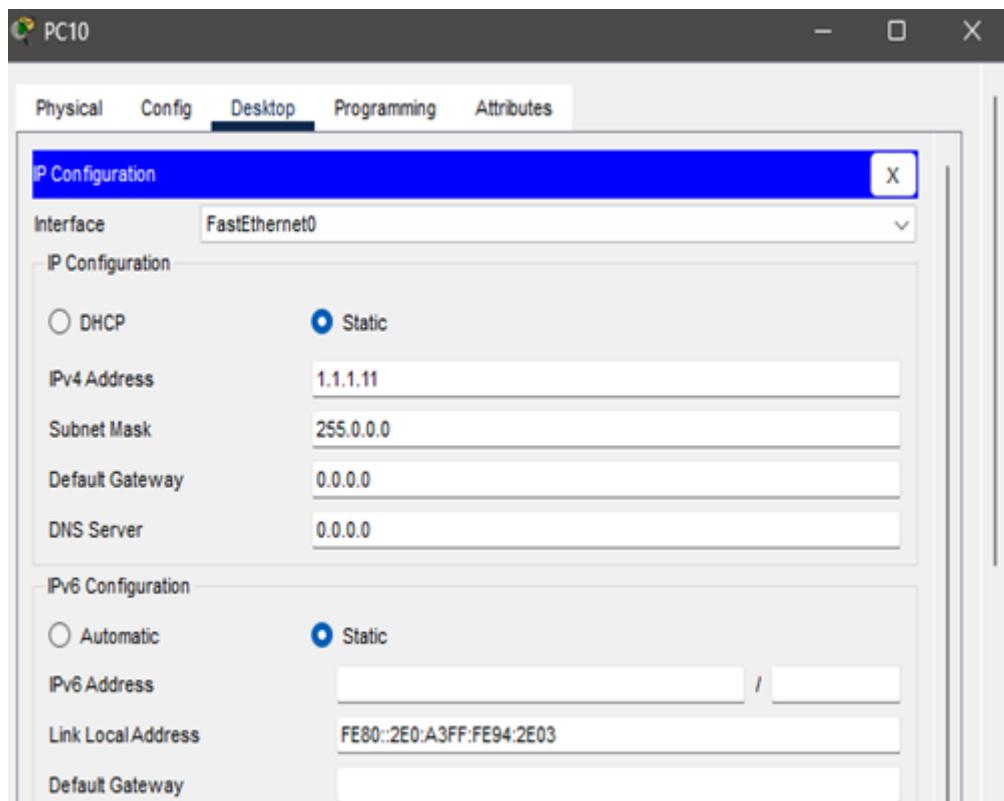
PC8 IP CONFIGURATION



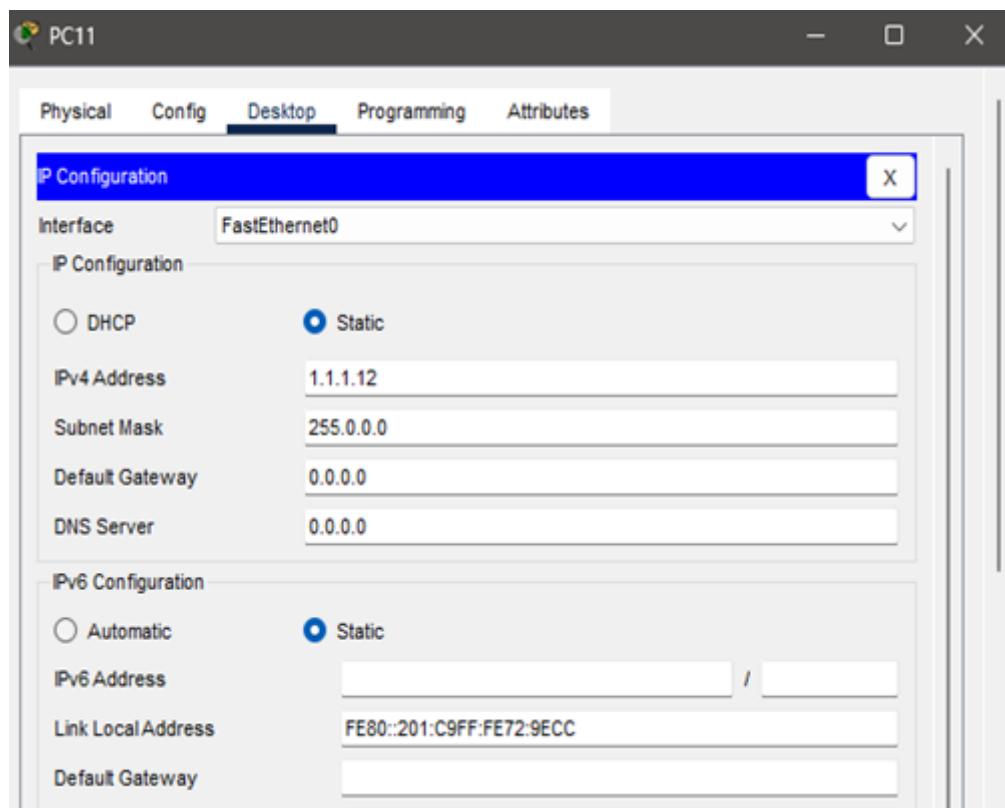
PC9 IP CONFIGURATION



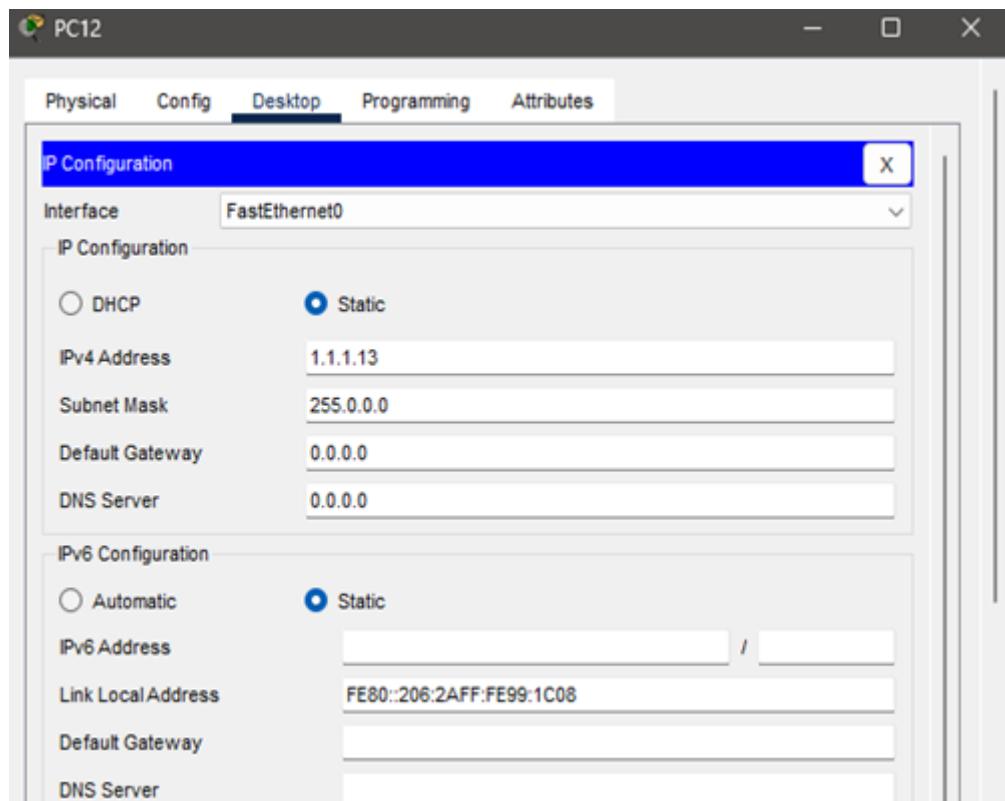
PC10 IP CONFIGURATION



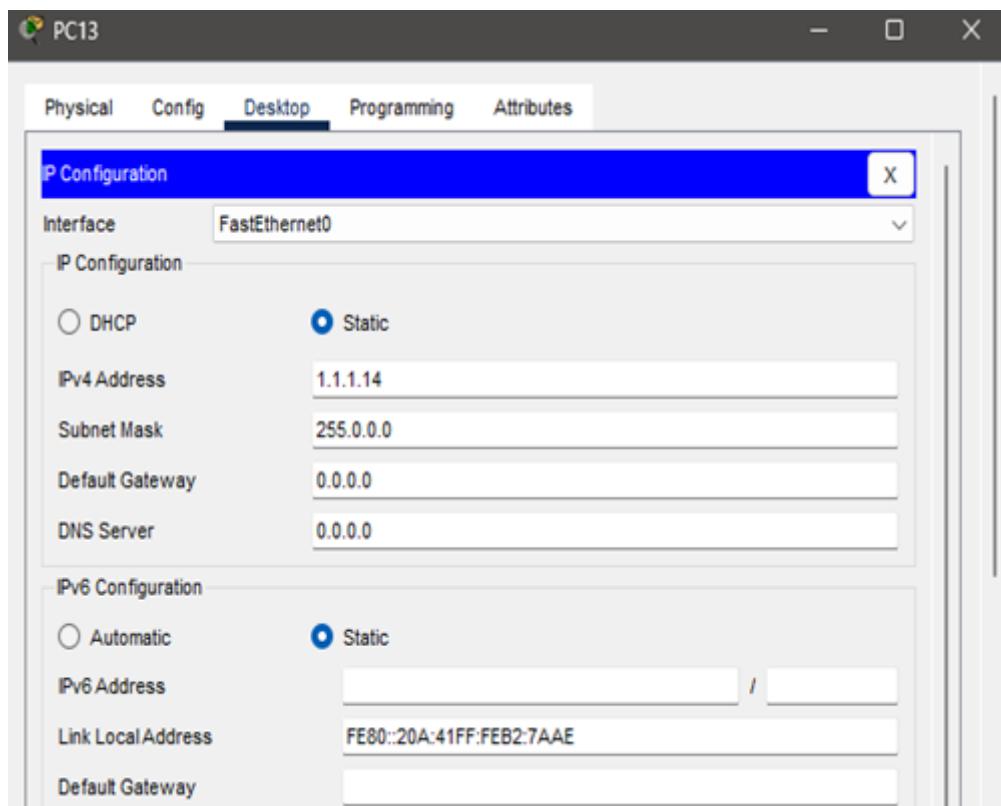
PC11 IP CONFIGURATION



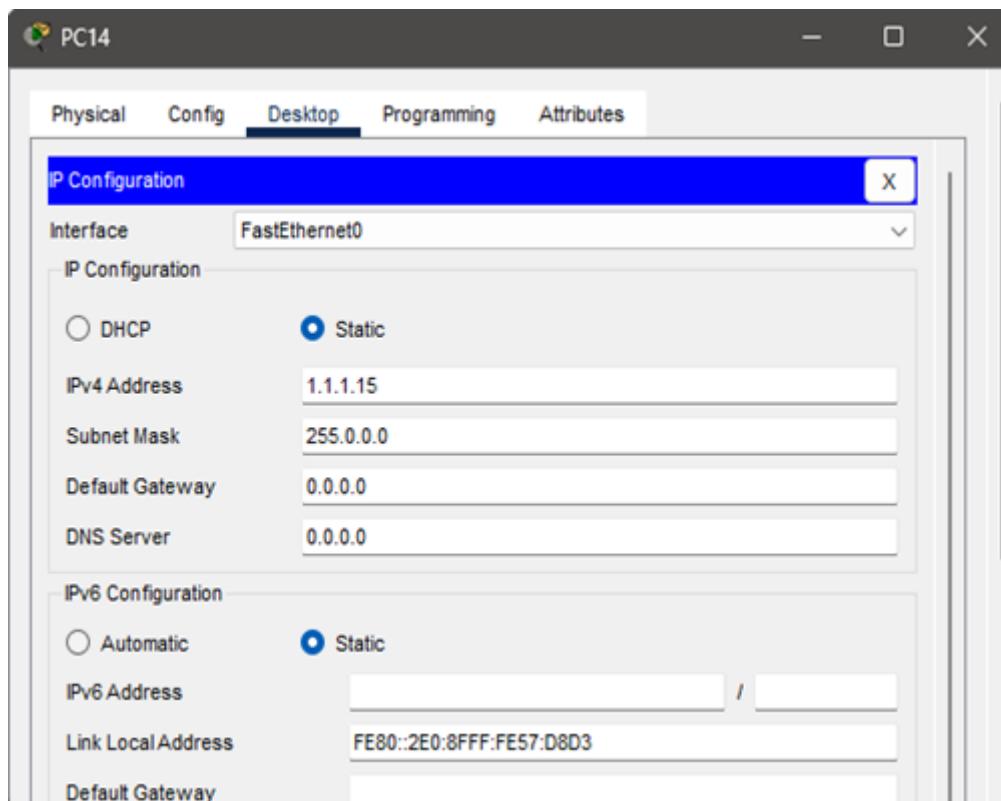
PC12 IP CONFIGURATION



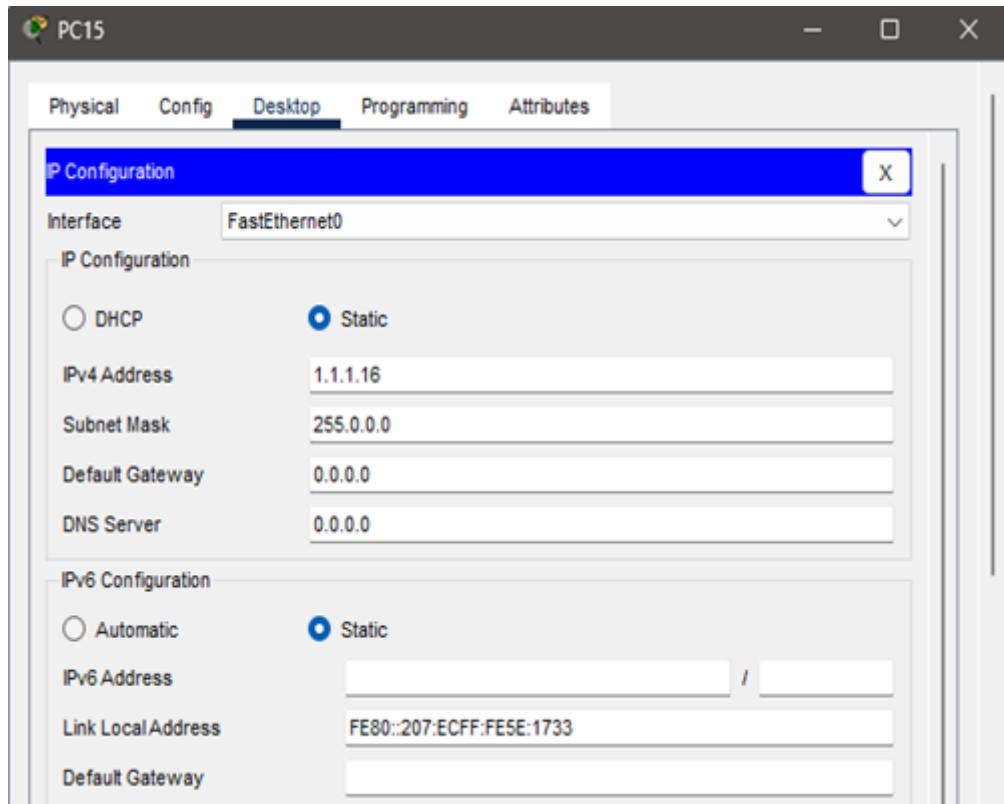
PC13 IP CONFIGURATION



PC14 IP CONFIGURATION



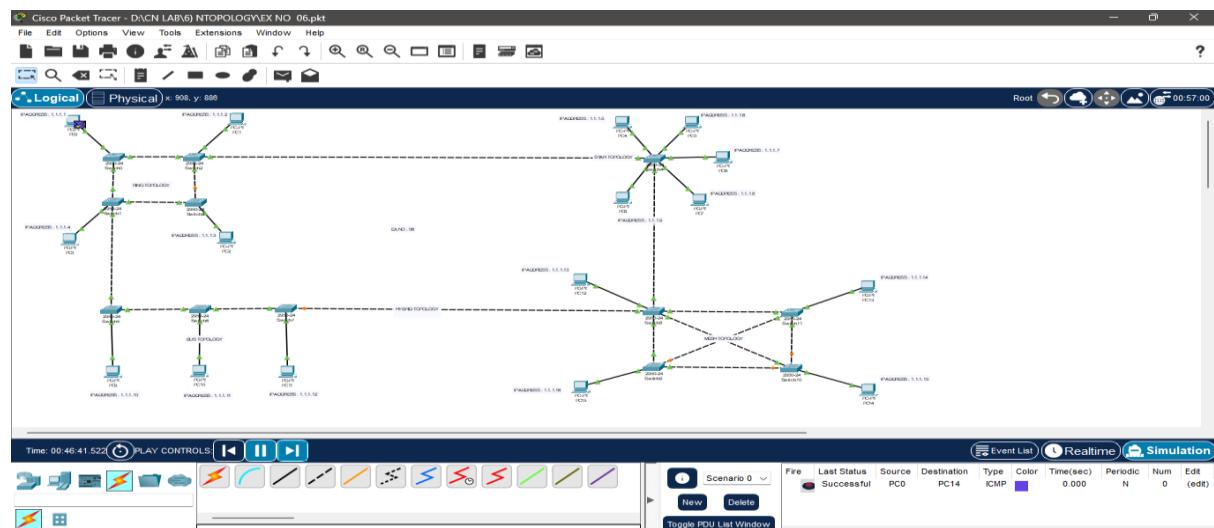
PC15 IP CONFIGURATION



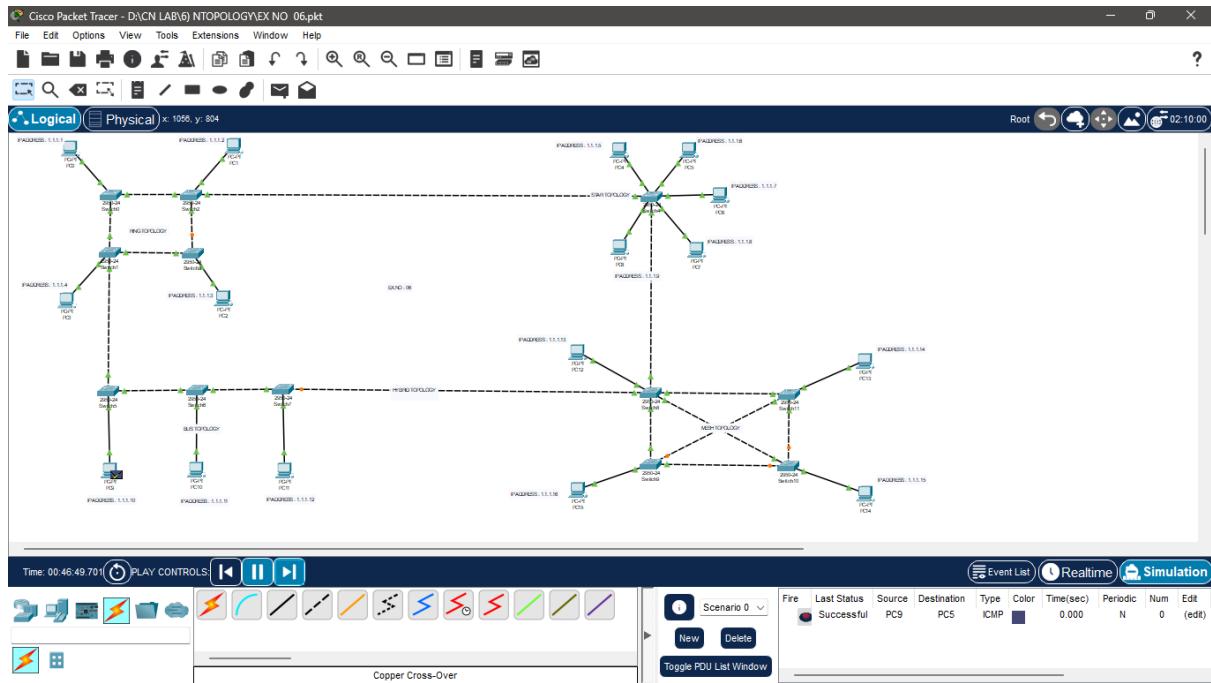
VERIFY LAN NETWORK CONNECTIVITY :

Using Add Simple PDU(p), Click the mail icon and then drop one mail to one of the PC in first lan and another mail to PC in another lan. If the resultant window shows the successful delivery, then network connectivity is successful.

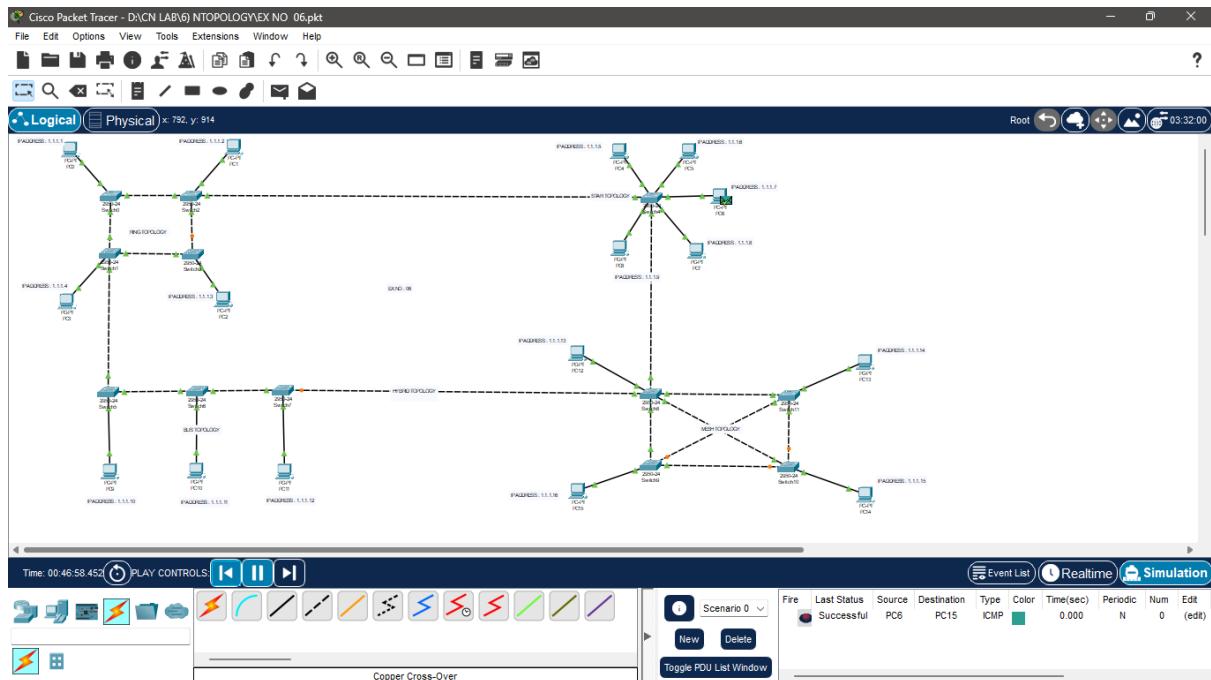
HOST PC0 TO PC14



HOST PC9 TO PC5



HOST PC6 TO PC15



RESULT :

Thus, Hybrid topologies are designed using cisco packet tracer and the communication between Hybrid topologies is checked successfully.

EX.NO : 09

DATE : 16.02.24

IMPLEMENTING ERROR DETECTING CODE USING PARITY CHECK

AIM :

To write a Java program for Error Detecting code using Parity Check.

ALGORITHM :

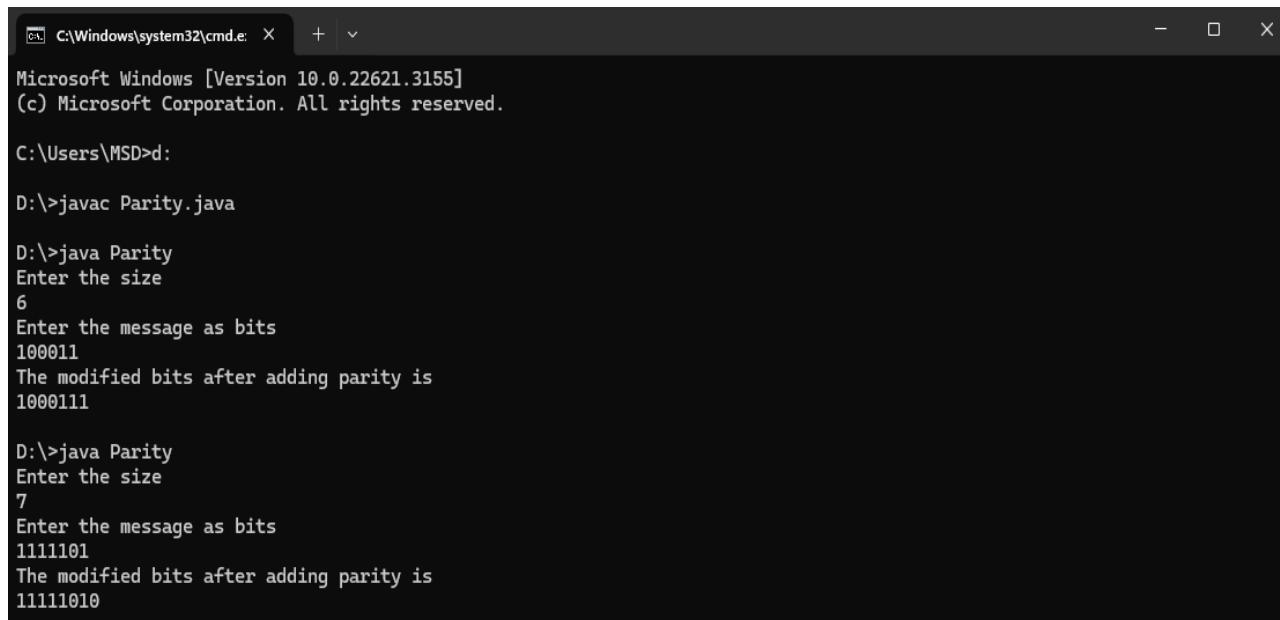
- 1) Start the program.
- 2) Input the size of the message.
- 3) Input the message as bits.
- 4) Calculate the parity bit based on the count of set bits in the message.
- 5) Output the modified message bits with the appended parity bit.
- 6) Stop the program.

PROGRAM :

```
import java.util.*;  
  
class Parity  
{  
    public static void main(String[] args)  
    {  
        Scanner in = new Scanner(System.in);  
        System.out.println("Enter the size");  
        int size = in.nextInt();  
        System.out.println("Enter the message as bits");
```

```
String mess = in.next();
int[] arr = new int[size + 1];
for (int i = 0; i < size; i++)
{
    arr[i] = mess.charAt(i) - '0';
}
int count = 0;
for (int i = 0; i < size; i++)
{
    if (arr[i] == 1)
    {
        count++;
    }
}
arr[size] = (count % 2 == 0) ? 0 : 1;
System.out.println("The modified bits after adding parity is");
for (int i = 0; i < size + 1; i++)
{
    System.out.print(arr[i]);
}
System.out.println();
}
```

OUTPUT :



```
C:\Windows\system32\cmd.exe + - X
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\MSD>d:

D:\>javac Parity.java

D:\>java Parity
Enter the size
6
Enter the message as bits
100011
The modified bits after adding parity is
1000111

D:\>java Parity
Enter the size
7
Enter the message as bits
1111101
The modified bits after adding parity is
11111010
```

RESULT :

Thus, the program for Error Detecting code using Parity Check is successfully executed and the output is verified.

EX.NO : 10

DATE : 20.02.24

IMPLEMENTING ERROR DETECTING CODE USING CHECKSUM

AIM :

To write a Java program for Error Detecting code using Checksum.

ALGORITHM :

- 1)** Start the program.
- 2)** Prompt the user for the number of data segments and the number of bits per segment.
- 3)** Read and store the data segments.
- 4)** Calculate the sender's checksum and its complement.
- 5)** Validate the received checksum and print the conclusion.
- 6)** Stop the program.

PROGRAM :

```
import java.util.Scanner;  
  
public class Checksum  
{  
    static String complement(String sum, int m)  
    {  
        char bits[] = sum.toCharArray();  
        for (int i = 0; i < m; i++)  
        {  
            if (bits[i] == '1')  
                bits[i] = '0';  
            else  
                bits[i] = '1';  
        }  
        return new String(bits);  
    }  
}
```

```

    {
        bits[i] = '0';
    }
    else
    {
        bits[i] = '1';
    }
}

return new String(bits);
}

static String calChecksum(String data[], int k, int m)
{
    int a = Integer.parseInt(data[0], 2);
    int b = 0;
    int c = 0;
    for (int i = 1; i < k; i++)
    {
        b = Integer.parseInt(data[i], 2);
        c = a + b;
        String temp = Integer.toBinaryString(c);
        if (temp.length() > m)
        {
            temp = temp.substring(1);
            c = Integer.parseInt(temp, 2);
        }
    }
}

```

```

        c = c + 1;

    }

    a = c;

}

String sum = Integer.toBinaryString(c);

String t = sum;

if (sum.length() < m)

{

    int diff = m - sum.length();

    for (int i = 0; i < diff; i++)

        t = "0" + t;

}

sum = t;

return sum;

}

static boolean validateChecksum(String data[], int k, int m, String
senderChecksum)

{

String sum = calChecksum(data, k, m);

int s = Integer.parseInt(sum, 2);

int sc = Integer.parseInt(senderChecksum, 2);

s = s + sc;

String finalSum = complement(Integer.toBinaryString(s), m);

System.out.println("Receiver      side      sum:      "      +
Integer.toBinaryString(s));

```

```

        System.out.println("Receiver side complement: " + finalSum);

        return finalSum.equals("00000000");

    }

    public static void main(String[] args)

    {

        System.out.println("How many segments of data? ");

        Scanner input = new Scanner(System.in);

        int k = input.nextInt();

        System.out.println("How many bits per segment? ");

        int m = input.nextInt();

        String data[] = new String[k];

        for (int i = 0; i < k; i++)

        {

            System.out.println("Enter data segment " + (i + 1) + ": ");

            data[i] = input.next();

        }

        String senderChecksum = complement(calChecksum(data, k, m), m);

        System.out.println("Sender side checksum value: " + senderChecksum);

        System.out.println("Receiver      side      complement:      "      +

complement(senderChecksum, m));

        System.out.println("Conclusion:  "  + (validateChecksum(data,  k,  m,

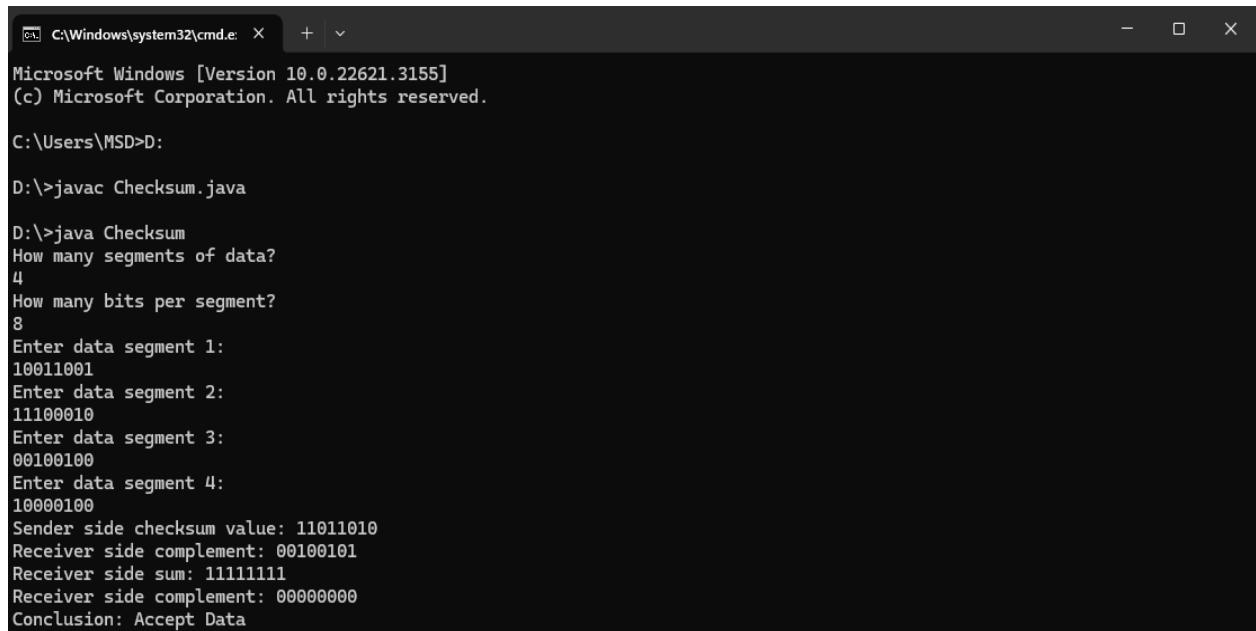
senderChecksum) ? "Accept Data" : "Reject Data"));

    }

}

```

OUTPUT :



```
C:\Windows\system32\cmd.exe + - Microsoft Windows [Version 10.0.22621.3155] (c) Microsoft Corporation. All rights reserved. C:\Users\MSD>D: D:\>javac Checksum.java D:\>java Checksum How many segments of data? 4 How many bits per segment? 8 Enter data segment 1: 10011001 Enter data segment 2: 11100010 Enter data segment 3: 00100100 Enter data segment 4: 10000100 Sender side checksum value: 11011010 Receiver side complement: 00100101 Receiver side sum: 11111111 Receiver side complement: 00000000 Conclusion: Accept Data
```

RESULT :

Thus, the program for Error Detecting code using Checksum is successfully executed and the output is verified.

EX.NO : 11

DATE : 23.02.24

IMPLEMENTING ERROR DETECTING CODE USING CRC – CCITT

AIM :

To write a Java program for Error Detecting code using CRC – CCITT (16 bits).

ALGORITHM :

- 1)** Start the program.
- 2)** Read the message from the user.
- 3)** Calculate and append the CRC remainder to the message.
- 4)** Prompt the user to enter the received data and check its CRC remainder.
- 5)** Print the result.
- 6)** Stop the program.

PROGRAM :

```
import java.util.*;  
  
public class Crc  
{  
    public static int n;  
  
    public static void main(String[] args)  
    {  
        Scanner in=new Scanner(System.in);  
        Crc ob=new Crc();  
        String code, copy, rec,zero="0000000000000000";
```

```

System.out.println("Enter message");
code=in.nextLine();
n=code.length();
copy=code;
code+=zero;
code=ob.divide(code);
System.out.println("Message="+copy);
copy=copy.substring(0,n)+code.substring(n);
System.out.println("CRC=");
System.out.println(code.substring(n));
System.out.println("transmitted frame is "+copy);
System.out.println("Enter received data");
rec=in.nextLine();
if(zero.equals(ob.divide(rec).substring(n)))
    System.out.println("Correct bits received");
else
    System.out.println("Received frame contains one or more
errors");
in.close();
}

public String divide(String s)
{
    int i,j;
    char x;

```

```
String div="1000100000100001";
for(i=0;i<n;i++)
{
    x=s.charAt(i);
    for(j=0;j<17;j++)
    {
        if(x=='1')
        {
            if(s.charAt(i+j)!=div.charAt(j))
                s=s.substring(0,i+j)+"1"+s.substring(i+j+1);
            else
                s=s.substring(0,i+j)+"0"+s.substring(i+j+1);
        }
    }
    return s;
}
}
```

OUTPUT :

```
C:\Windows\system32\cmd.e: + - X
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\MSD>d:

D:\>javac Crc.java

D:\>java Crc
Enter message
1011
Message=1011
CRC=
1011000101101011
transmitted frame is 10111011000101101011
Enter received data
10111011000101101011
Correct bits received

D:\>java Crc
Enter message
11000
Message=11000
CRC=
1001001100111001
transmitted frame is 110001001001100111001
Enter received data
110001001001100111000
Received frame contains one or more errors
```

RESULT :

Thus, the program for Error Detecting code using CRC – CCITT (16 Bits) is successfully executed and the output is verified.

EX.NO : 12

DATE : 01.03.24

IMPLEMENTATION OF GO BACK N PROTOCOL

AIM :

To write a Java program to perform a GoBackN Protocol.

ALGORITHM :

- 1) Start the program.
- 2) Prompt for the frame size.
- 3) Initialize sent to 0.
- 4) Transmit frames up to the frame size, printing each transmission.
- 5) Prompt for lost Acknowledgement.
- 6) If lost Acknowledgement equals frame size, stop; else, update sent and repeat transmission.
- 7) Stop the program.

PROGRAM :

```
import java.io.*;  
  
public class GoBackN  
{  
    public static void main(String args[]) throws IOException  
    {  
        BufferedReader br = new BufferedReader(new  
InputStreamReader(System.in));  
        System.out.println("Enter the Frame Size : ");  
        int window = Integer.parseInt(br.readLine());
```

```
boolean loop = true;  
int sent = 0;  
while(loop)  
{  
    for(int i = 0; i < window; i++)  
    {  
        System.out.println("Frame " + sent + " has been  
transmitted.");  
        sent++;  
        if(sent == window)  
            break;  
    }  
    System.out.println("Enter the lost Acknowledgement : ");  
    int ack = Integer.parseInt(br.readLine());  
    if(ack == window)  
        loop = false;  
    else  
        sent = ack;  
}  
}  
}
```

OUTPUT :

```
C:\Windows\system32\cmd.e: + v
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\MSD>d:

D:>javac GoBackN.java

D:>java GoBackN
Enter the Frame Size :
6
Frame 0 has been transmitted.
Frame 1 has been transmitted.
Frame 2 has been transmitted.
Frame 3 has been transmitted.
Frame 4 has been transmitted.
Frame 5 has been transmitted.
Enter the lost Acknowledgement :
2
Frame 2 has been transmitted.
Frame 3 has been transmitted.
Frame 4 has been transmitted.
Frame 5 has been transmitted.
Enter the lost Acknowledgement :
3
Frame 3 has been transmitted.
Frame 4 has been transmitted.
Frame 5 has been transmitted.
```

RESULT :

Thus, the program for Implementing a GoBackN Protocol is successfully executed and the output is verified.

EX.NO : 13

DATE : 05.03.24

IMPLEMENTATION OF STOP AND WAIT PROTOCOL

AIM :

To write a Java program to perform a Stop and Wait Protocol.

ALGORITHM :

1. Start the program.
2. Run the server on a separate thread.
3. Connect the client to the server.
4. Exchange messages between client and server, repeating three times:
 - a. Client sends a message and waits for acknowledgment.
 - b. Server receives the message, prints it, and sends an acknowledgment.
5. Stop the program.

PROGRAM :

```
import java.io.*;  
import java.net.*;  
  
public class StopAndWait {  
  
    private static final int PORT = 123;  
  
    public static void main(String[] args) {  
  
        new Thread(() -> {  
  
            try {  
  
                runServer();  
            }  
        })  
    }  
}
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
    }).start();
runClient();
}

private static void runServer() throws IOException {
    ServerSocket serverSocket = new ServerSocket(PORT);
    System.out.println("Server waiting for connection...");
    Socket socket = serverSocket.accept();
    System.out.println("Server connected to client");
    BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
    PrintWriter writer = new PrintWriter(socket.getOutputStream(), true);
    String message;
    while ((message = reader.readLine()) != null) {
        System.out.println("Received from client: " + message);
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        writer.println("ACK");
    }
}
```

```

        socket.close();

        serverSocket.close();

    }

private static void runClient() {

    try {

        Socket socket = new Socket("localhost", PORT);

        BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

        PrintWriter writer = new PrintWriter(socket.getOutputStream(), true);

        BufferedReader userInput = new BufferedReader(new
InputStreamReader(System.in));

        for (int i = 1; i <= 3; i++) {

            System.out.print("Enter message to send: ");

            String message = userInput.readLine();

            System.out.println("Sending to server: " + message);

            writer.println(message);

            String ack = reader.readLine();

            System.out.println("Received acknowledgment from server: " + ack);

            try {

                Thread.sleep(1000);

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

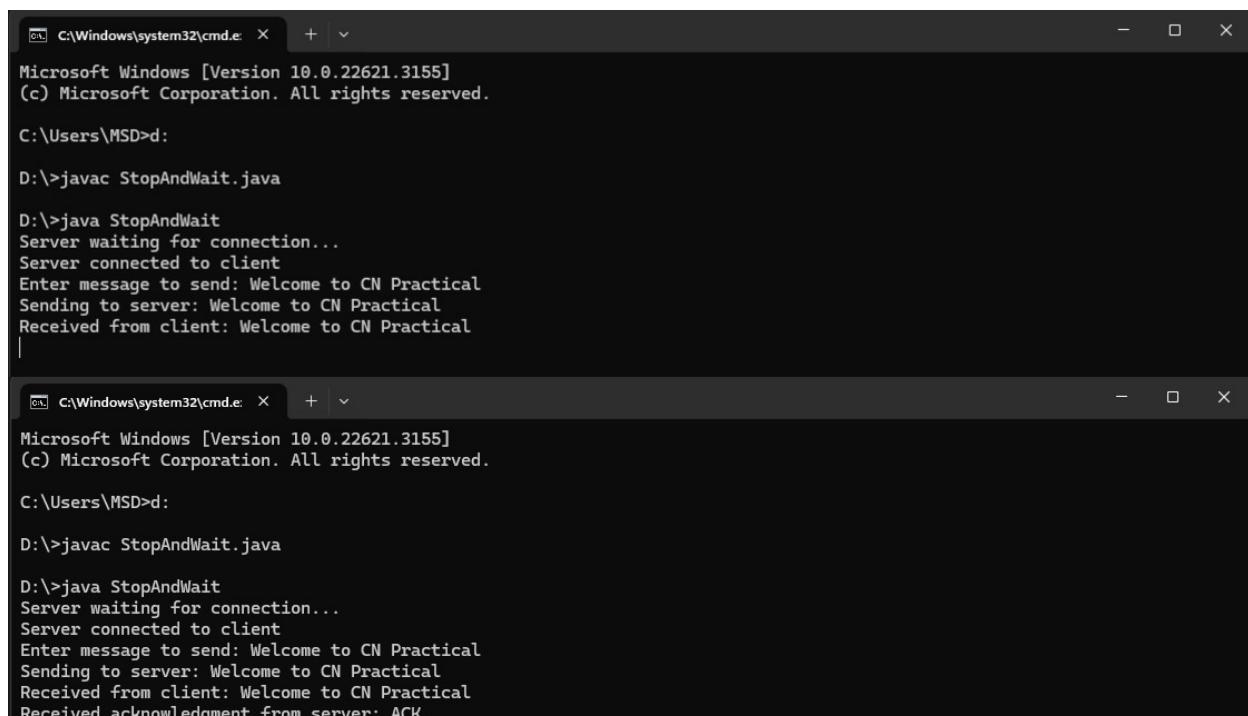
        }

        socket.close();
    }
}

```

```
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

OUTPUT :



```
C:\Windows\system32\cmd.e: + - x  
Microsoft Windows [Version 10.0.22621.3155]  
(c) Microsoft Corporation. All rights reserved.  
C:\Users\MSD>d:  
D:\>javac StopAndWait.java  
D:\>java StopAndWait  
Server waiting for connection...  
Server connected to client  
Enter message to send: Welcome to CN Practical  
Sending to server: Welcome to CN Practical  
Received from client: Welcome to CN Practical  
|  
  
C:\Windows\system32\cmd.e: + - x  
Microsoft Windows [Version 10.0.22621.3155]  
(c) Microsoft Corporation. All rights reserved.  
C:\Users\MSD>d:  
D:\>javac StopAndWait.java  
D:\>java StopAndWait  
Server waiting for connection...  
Server connected to client  
Enter message to send: Welcome to CN Practical  
Sending to server: Welcome to CN Practical  
Received from client: Welcome to CN Practical  
Received acknowledgment from server: ACK
```

RESULT :

Thus, the program for Implementing a Stop and Wait Protocol is successfully executed and the output is verified.

EX.NO : 14

DATE : 08.03.24

IMPLEMENTATION OF SELECTIVE REPEAT PROTOCOL

AIM :

To write a Java program to perform a Selective Repeat Protocol.

ALGORITHM :

1. Start program, prompt frame size.
2. Transmit frames, prompt lost Acknowledgement.
3. Print acknowledgment or retransmission message based on lost Acknowledgement.
4. Repeat until lost Acknowledgement equals frame size.
5. Stop program.

PROGRAM :

```
import java.io.*;  
  
public class SelectiveRepeat  
{  
  
    public static void main(String args[]) throws IOException  
    {  
  
        BufferedReader br = new BufferedReader(new  
InputStreamReader(System.in));  
  
        System.out.println("Enter the Frame Size: ");  
  
        int window = Integer.parseInt(br.readLine());  
  
        for (int sent = 0; sent < window; sent++)  
        {
```

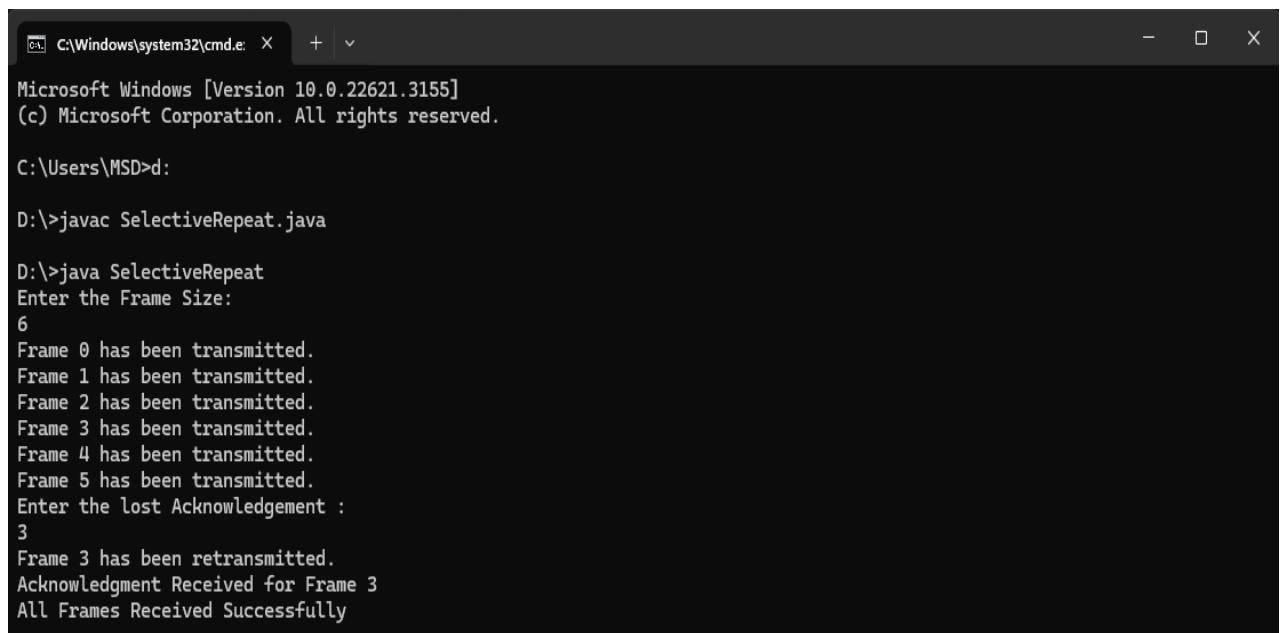
```

        System.out.println("Frame " + sent + " has been transmitted.");
    }

    int ack;
    do
    {
        System.out.println("Enter the lost Acknowledgement : ");
        ack = Integer.parseInt(br.readLine());
        if (ack == window)
        {
            System.out.println("Acknowledgment Received for Frame "
+ (ack - 1));
        }
        else
        {
            System.out.println("Frame " + ack + " has been
retransmitted.");
            System.out.println("Acknowledgment Received for Frame "
+ ack);
            System.out.println("All           Frames       Received
Successfully");
        }
    }
    while (ack == window);
}

```

OUTPUT :



```
C:\Windows\system32\cmd.exe + - X
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\MSD>d:

D:\>javac SelectiveRepeat.java

D:\>java SelectiveRepeat
Enter the Frame Size:
6
Frame 0 has been transmitted.
Frame 1 has been transmitted.
Frame 2 has been transmitted.
Frame 3 has been transmitted.
Frame 4 has been transmitted.
Frame 5 has been transmitted.
Enter the lost Acknowledgement :
3
Frame 3 has been retransmitted.
Acknowledgment Received for Frame 3
All Frames Received Successfully
```

RESULT :

Thus, the program for Implementing a Selective Repeat Protocol is successfully executed and the output is verified.

EX.NO : 15

DATE : 12.03.24

IMPLEMENTATION OF WEB PROGRAMMING USING HTML

AIM :

To Implement a Search Engine Web Programming using Html.

PROCEDURE :

- Create a HTML File.
- In the html file create a form using the <form> tag.
- Set the action attribute of the <form> as http://www.google.com/search.
- Inside the form create a text box for entering the search parameter.
- Set the value of the “GoogleSearch”.
- Create two radio buttons with name as “sitesearch” and one with value as null and the other with value as “srmuniv.ac.in”.
- Save the file with .html or .htm extension.

HOW TO EXECUTE :

- ✓ Double click the file and open it using any available browser.

PROGRAM :

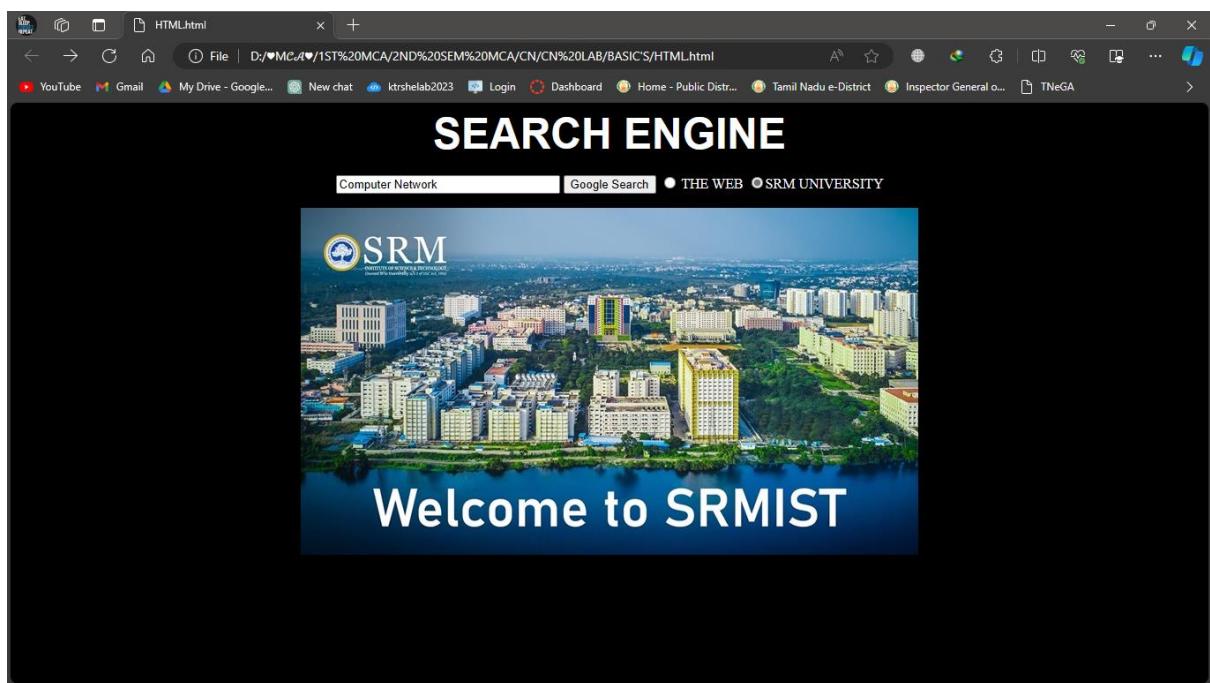
```
<html>
<body bgcolor="black">
<center><font size="36" color = "white" face="Arial"><b>SEARCH
ENGINE</b></font><br>
<br>
<form method="get" action ="http://www.google.com/search">
```

```

<input type="text" name="q" size="31" maxlength="255" value="" />
<input type="Submit" value="Google Search" />
<input type="radio" name="sitesearch" value="" />
<font color="white">THE WEB</font>
<input type="radio" name="sitesearch" value="srmuniv.ac.in" checked /><font color="white">SRM
UNIVERSITY</font><br>
</form></center>
<center></center><center>
</body>
</html>

```

OUTPUT :



Google

Computer Network site:srmuniv.ac.in

All Images Videos Shopping News More Tools SafeSearch

Types Example Components Topology Meaning Advantages Disadvantages Router Switch

About 414 results (0.25 seconds)

Networking Lab - SRMIST
The Networking Laboratory has been designed so with special emphasis on simulation of communication and **computer networks**, while hands-on programming ...

Dr. Savaridassan P - SRMIST
Savaridassan P. Assistant Professor, Cloud Computing, Explainable AI, Cyber Security, Deep Learning, Computer Vision, Machine Learning, **Computer Networks**, ...

Dr. Anand M - SRMIST
An academician and active researcher in Ad-hoc **Networks**, Wireless Sensor Networks ...

HTML.html

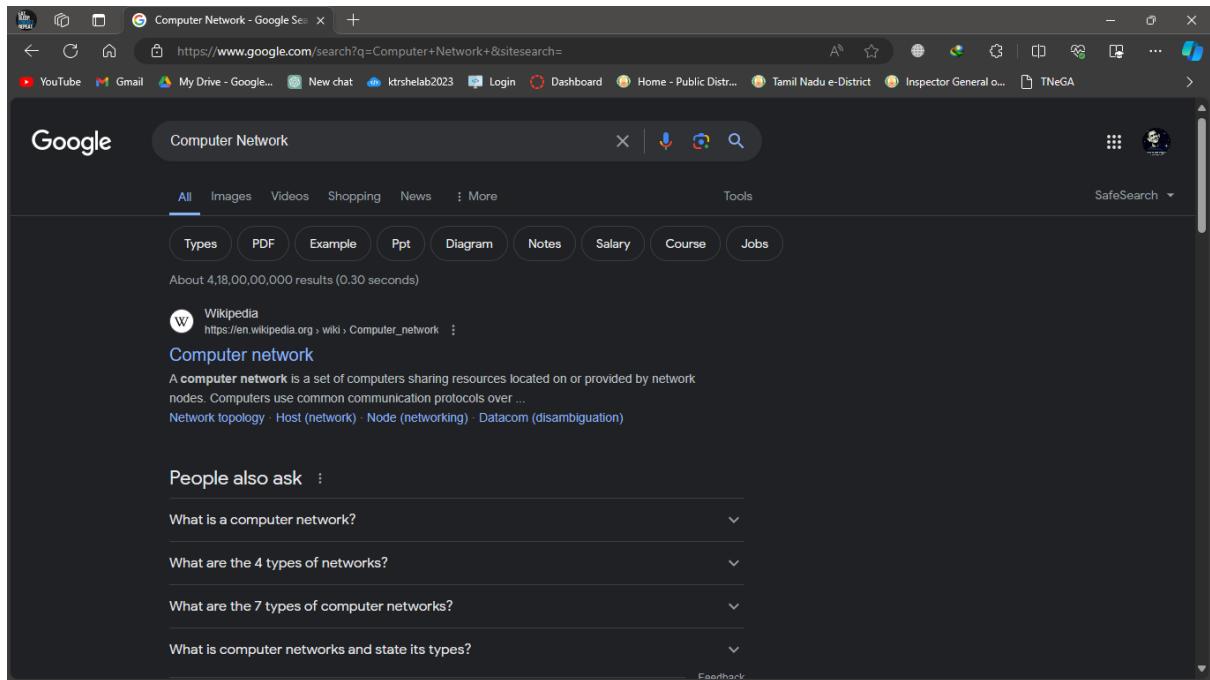
File | D:/McA/1ST%20MCA/2ND%20SEM%20MCA/CN/CN%20LAB/BASIC'S/HTML.html

YouTube Gmail My Drive - Google... New chat ktrshelab2023 Login Dashboard Home - Public Distr... Tamil Nadu e-District Inspector General o... TNGA

SEARCH ENGINE

Computer Network Google Search THE WEB SRM UNIVERSITY

 Welcome to SRMIST



RESULT :

Thus, the implementation of search engine web programming is executed successfully and the output is verified.