INDEX

EX.NO : 01

DATE :

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.

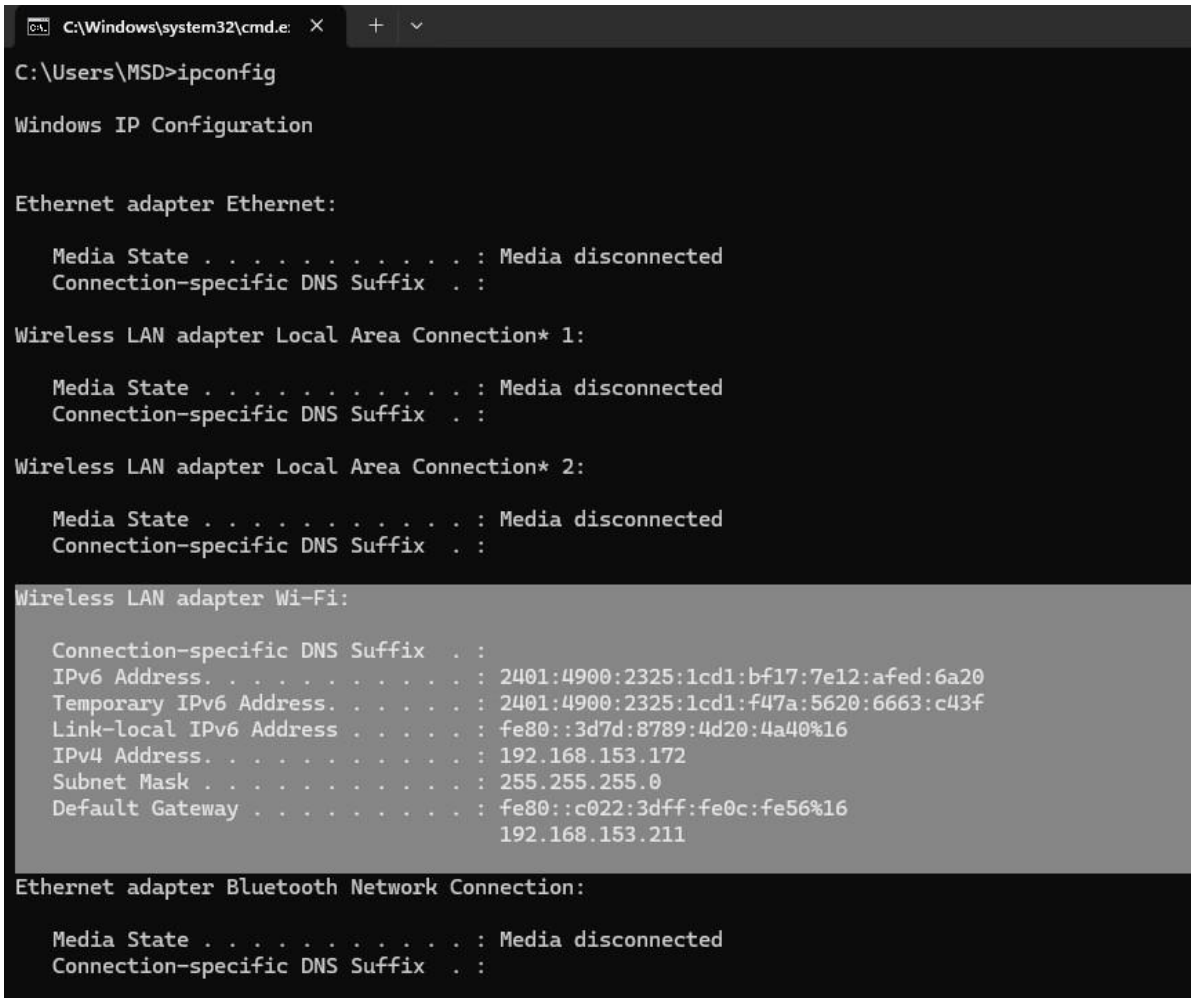
COMMAND DESCRIPTION :

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 a 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.

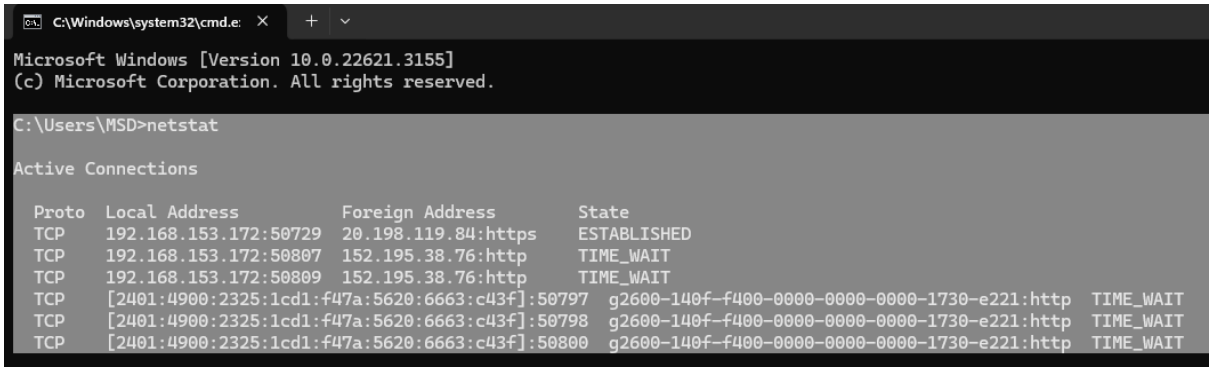
6)	Nslookup	Displays information that you can use to diagnose Domain Name System (DNS) infrastructure.
7)	Nbtstat	MS_DOS utility that displays protocol statistics & current TCP/IP connections using NBT.
8)	Getmac	DOS command used to show both local & remote MAC addresses when run with no parameters (i.egetmac) 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 :

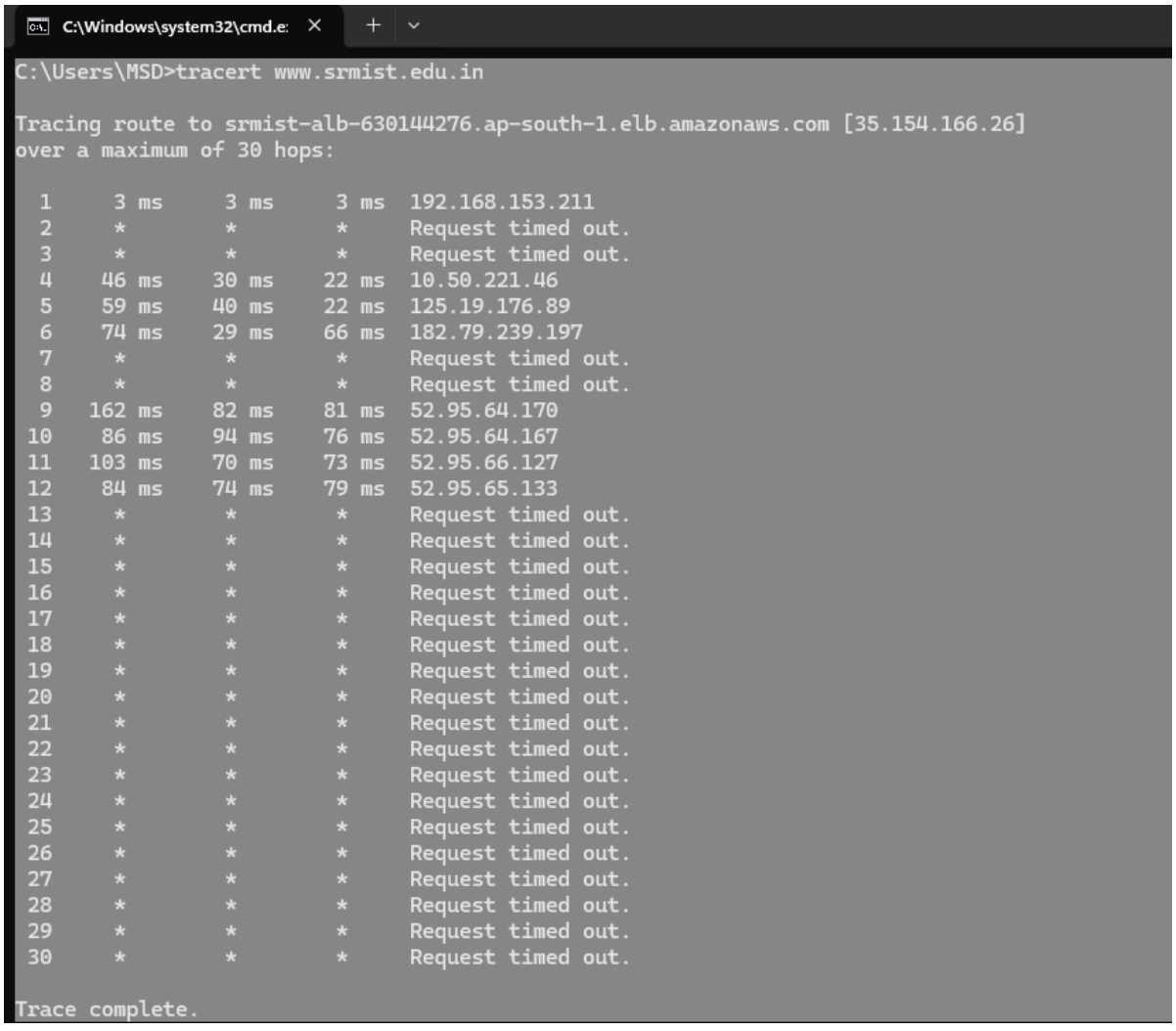
Ipconfig



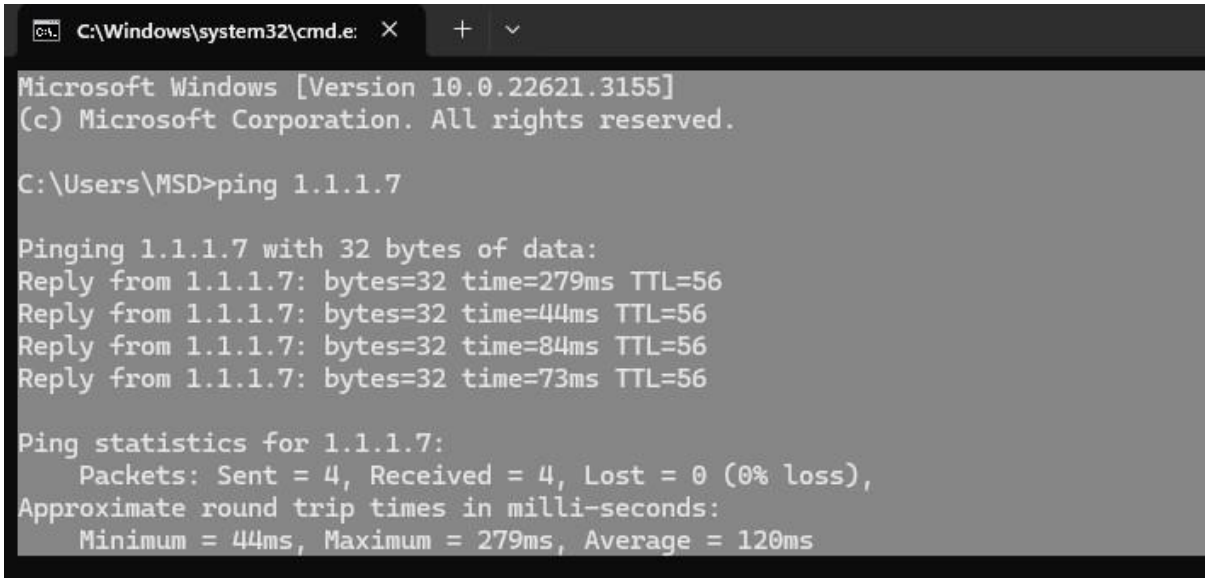
Netstat



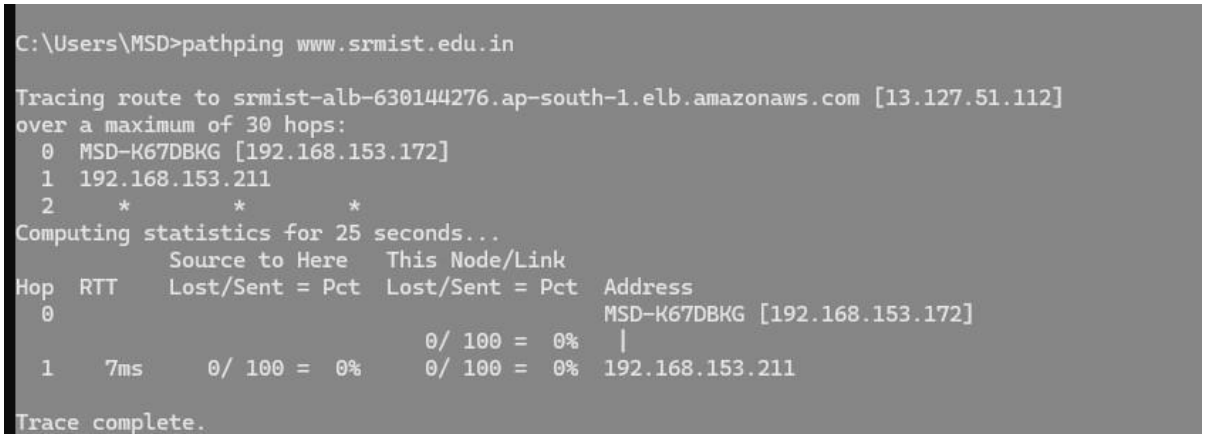
Tracert



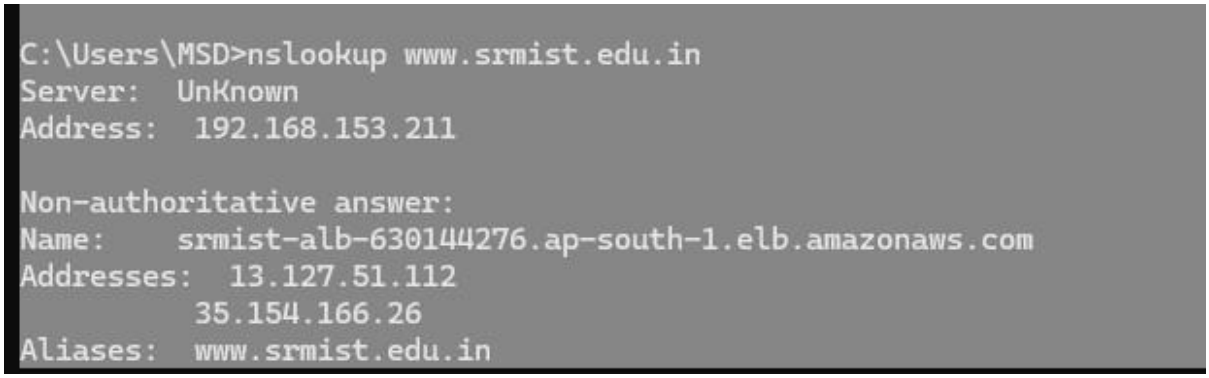
Ping



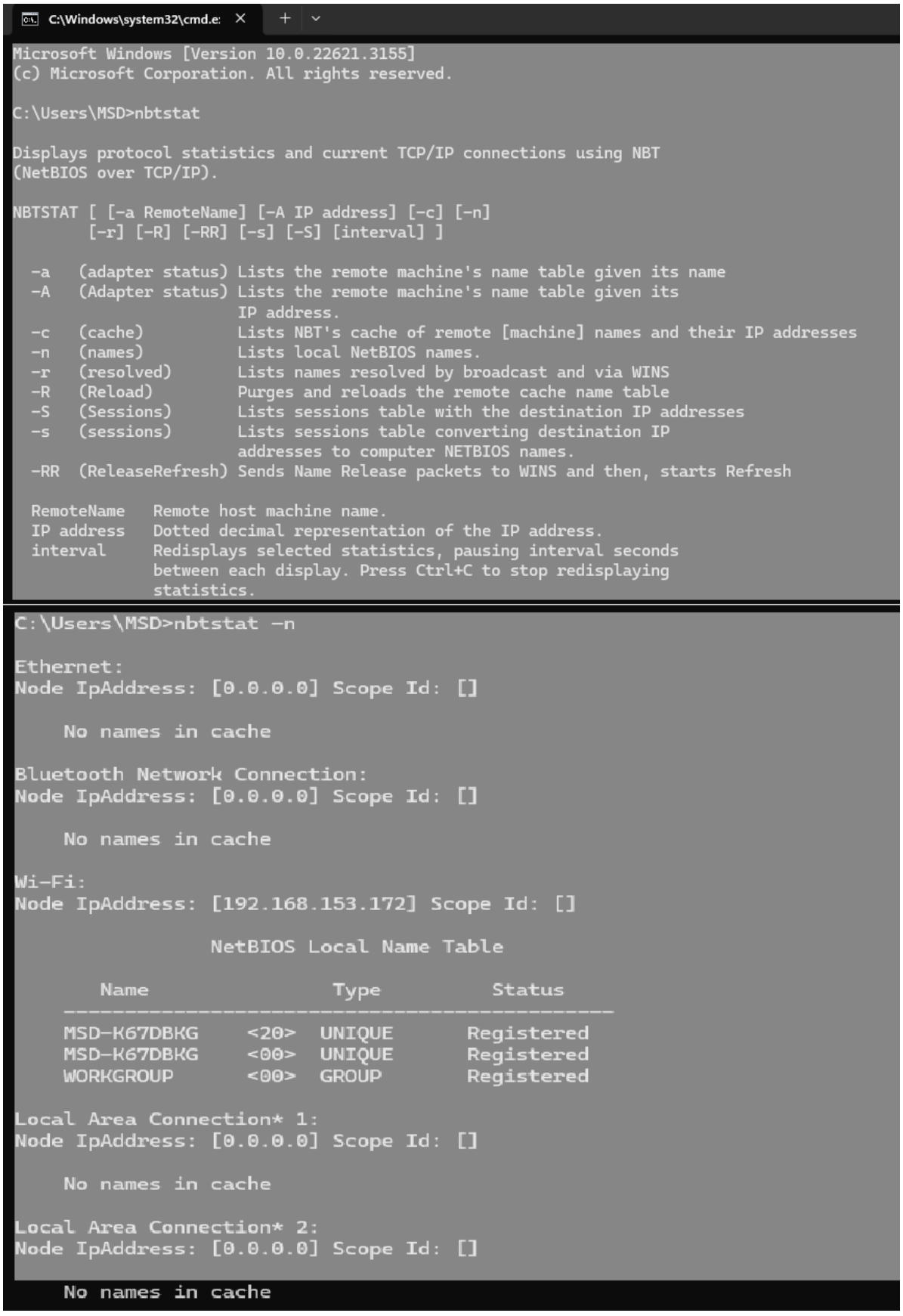
Pathping



Nslookup



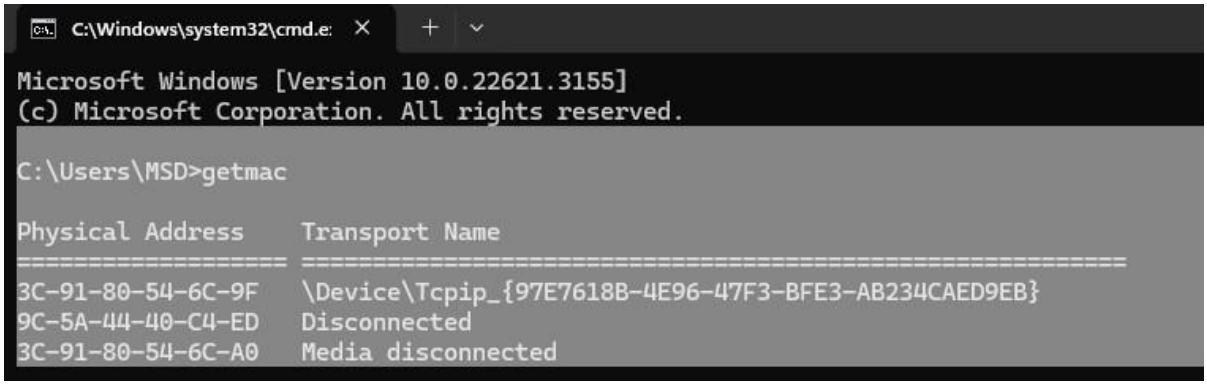
Nbtstat



```
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:
Node IpAddress: [0.0.0.0] Scope Id: []

    No names in cache

Bluetooth Network Connection:
Node IpAddress: [0.0.0.0] Scope Id: []

    No names in cache

Wi-Fi:
Node IpAddress: [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:
Node IpAddress: [0.0.0.0] Scope Id: []

    No names in cache

Local Area Connection* 2:
Node IpAddress: [0.0.0.0] Scope Id: []

    No names in cache
```

Getmac



RESULT :

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

EX.NO : 02

DATE :

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

AIM :

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

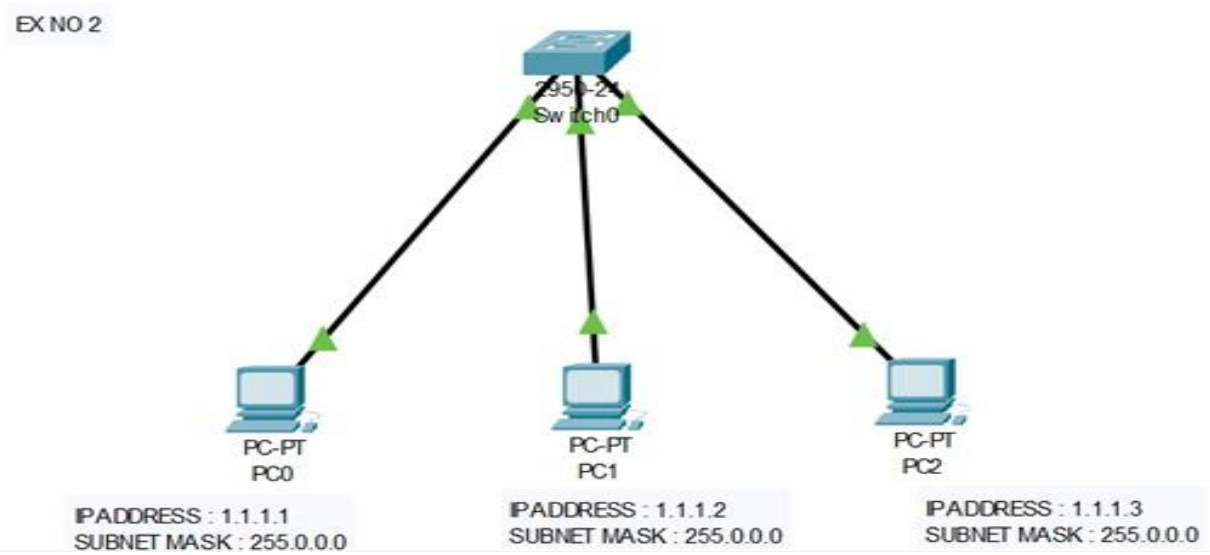
REQUIREMENTS :

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

PROCEDURES :

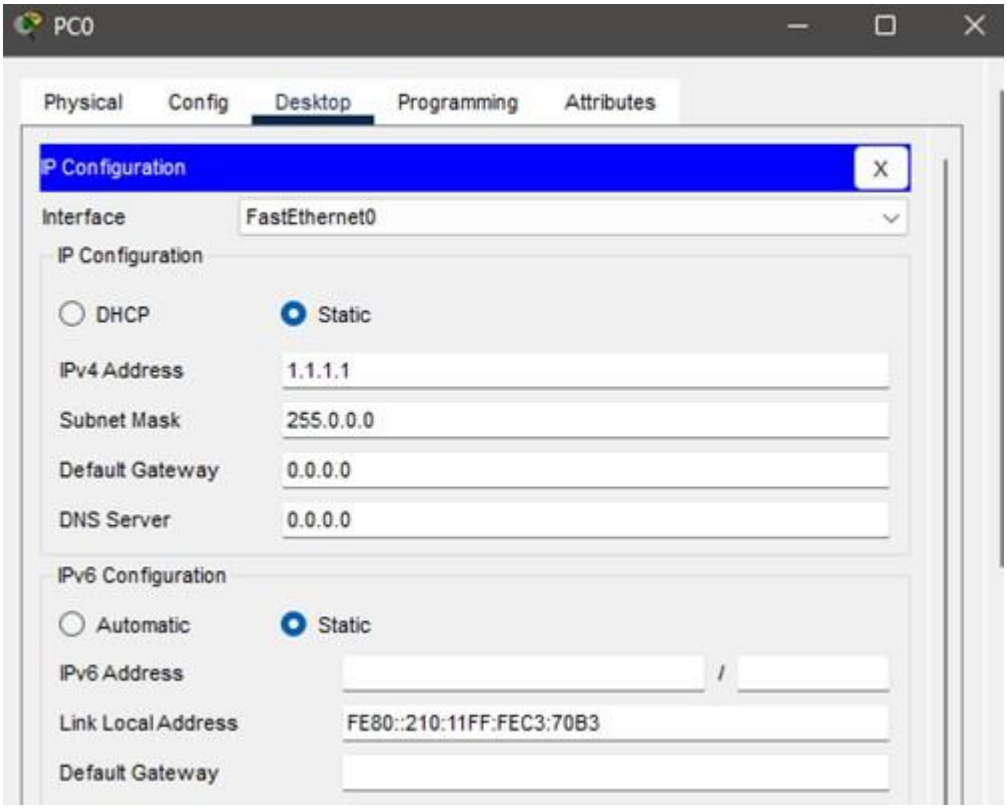
- Open CISCO PACKET TRACER software.
- Draw the Three PC using END Device Icons.
- Draw the Cisco 24 Port Switch Using Switch icon lists.
- Make the Connections using Copper-Straight-Through Ethernet Cables.
- Enter the IP Address To Each Machine.
- Check the Network Connections using Add Simple PDU(P).

NETWORK TOPOLOGY :

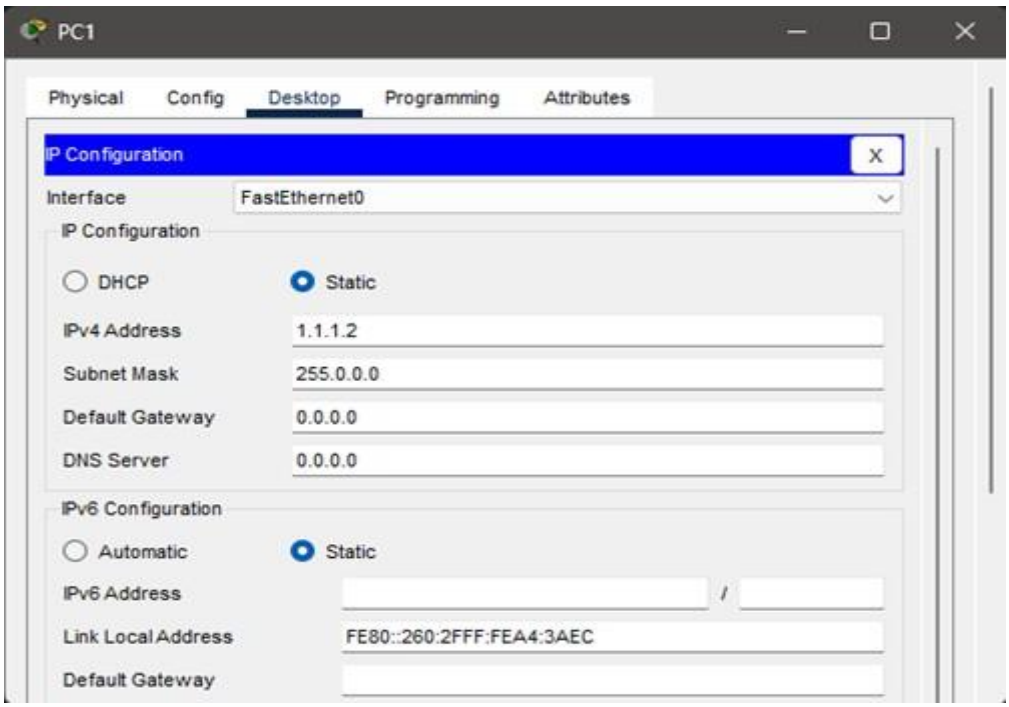


HOST IP ADDRESS :

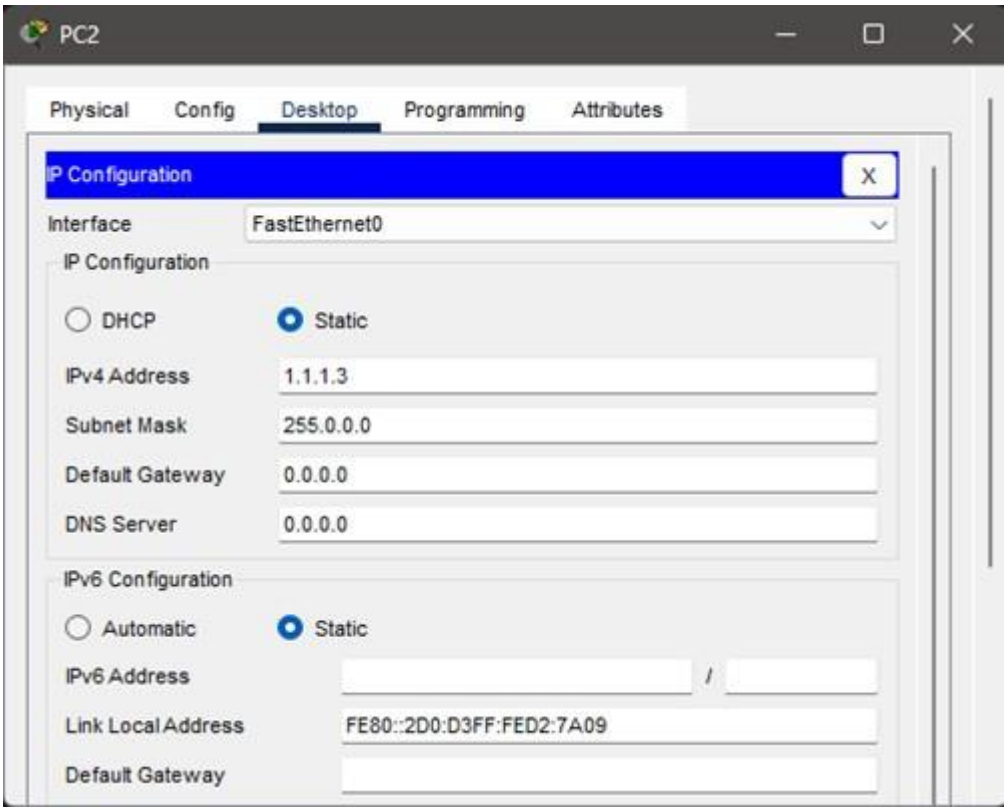
PC0 IP CONFIGURATION



PC1 IP CONFIGURATION



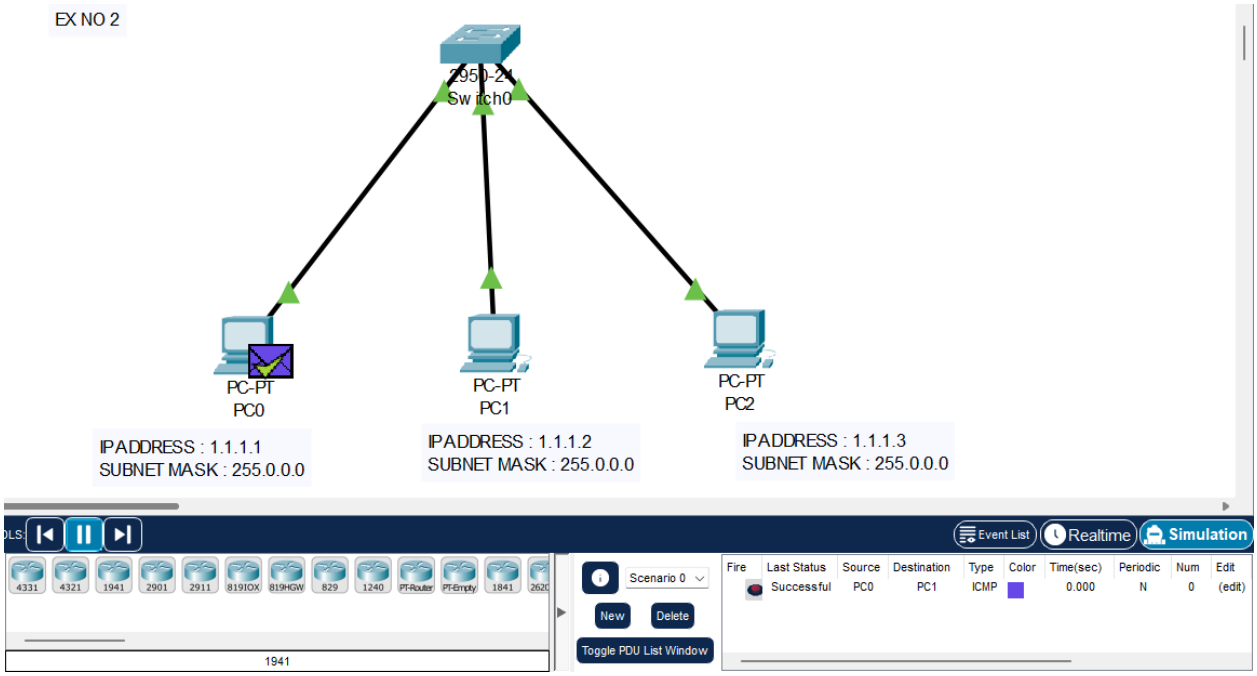
PC2 IP CONFIGURATION



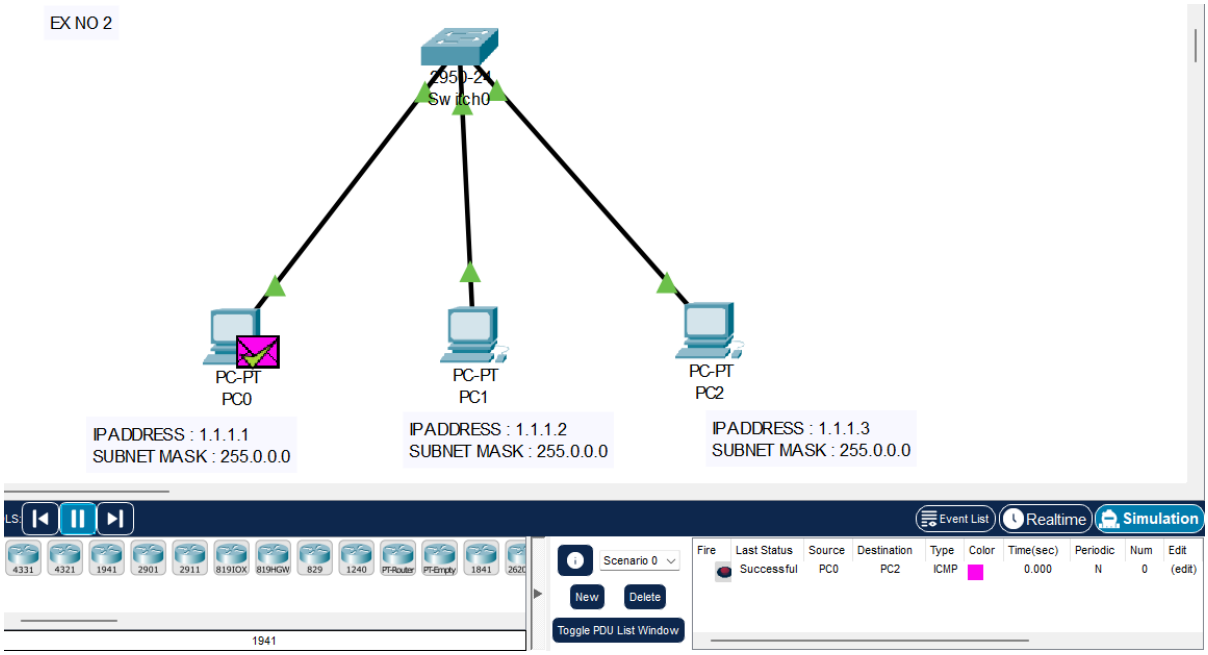
VERIFY LAN NETWORK CONNECTIVITY :

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 :

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

AIM :

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

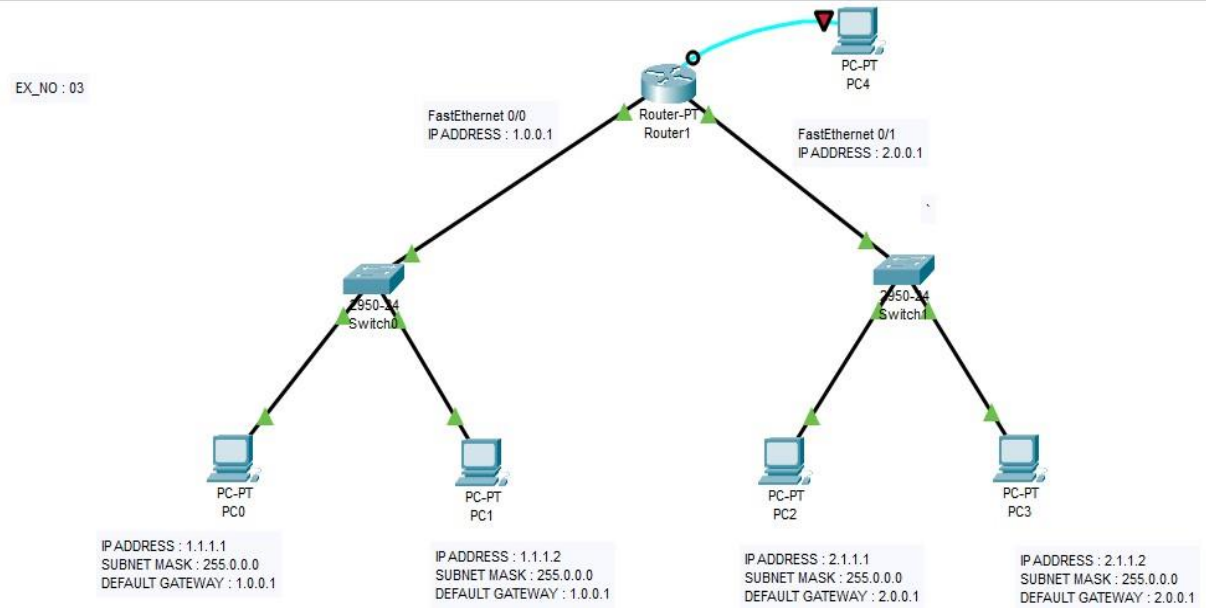
REQUIREMENTS :

- 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 :

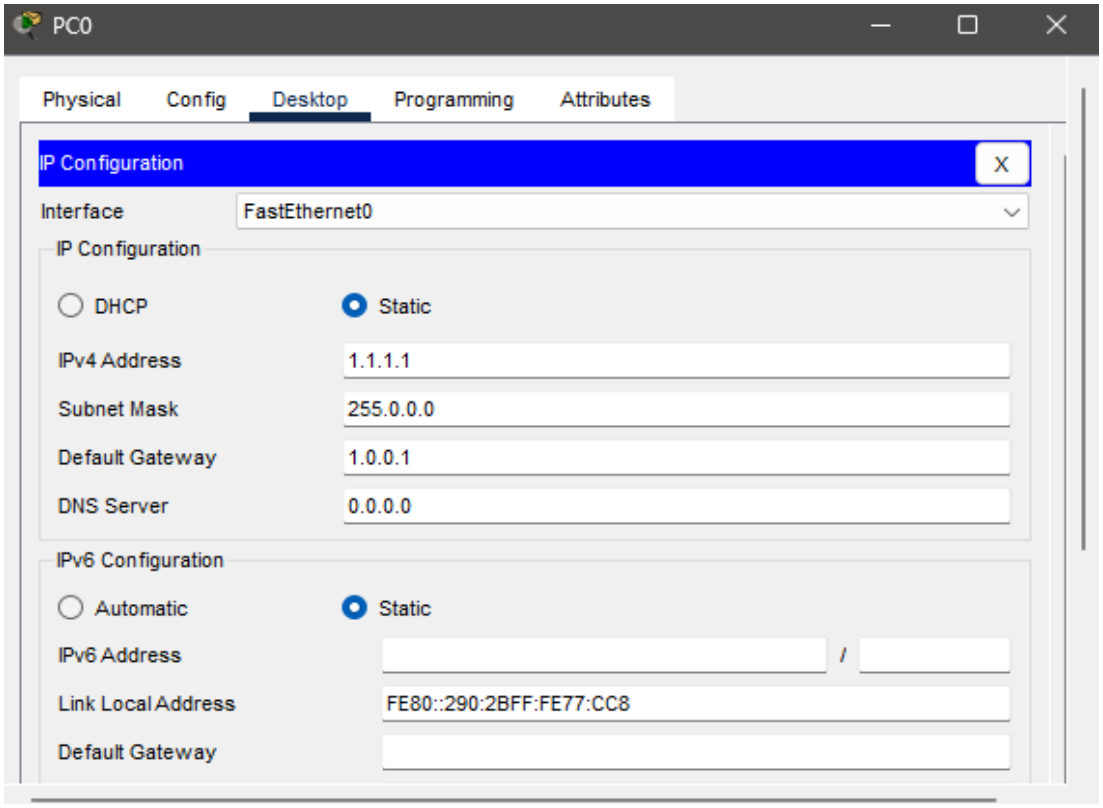
- Open CISCO PACKET TRACER software.
- Draw the Five PC using END Device Icons.
- Draw the Cisco 24 Port Switch Using Switch icon lists.
- Draw the Cisco Generic Routers using Router icon lists.
- Make the Connections using Copper-Straight-Through Ethernet Cables.
- Enter the IP Address To Each Machine.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

NETWORK TOPOLOGY :

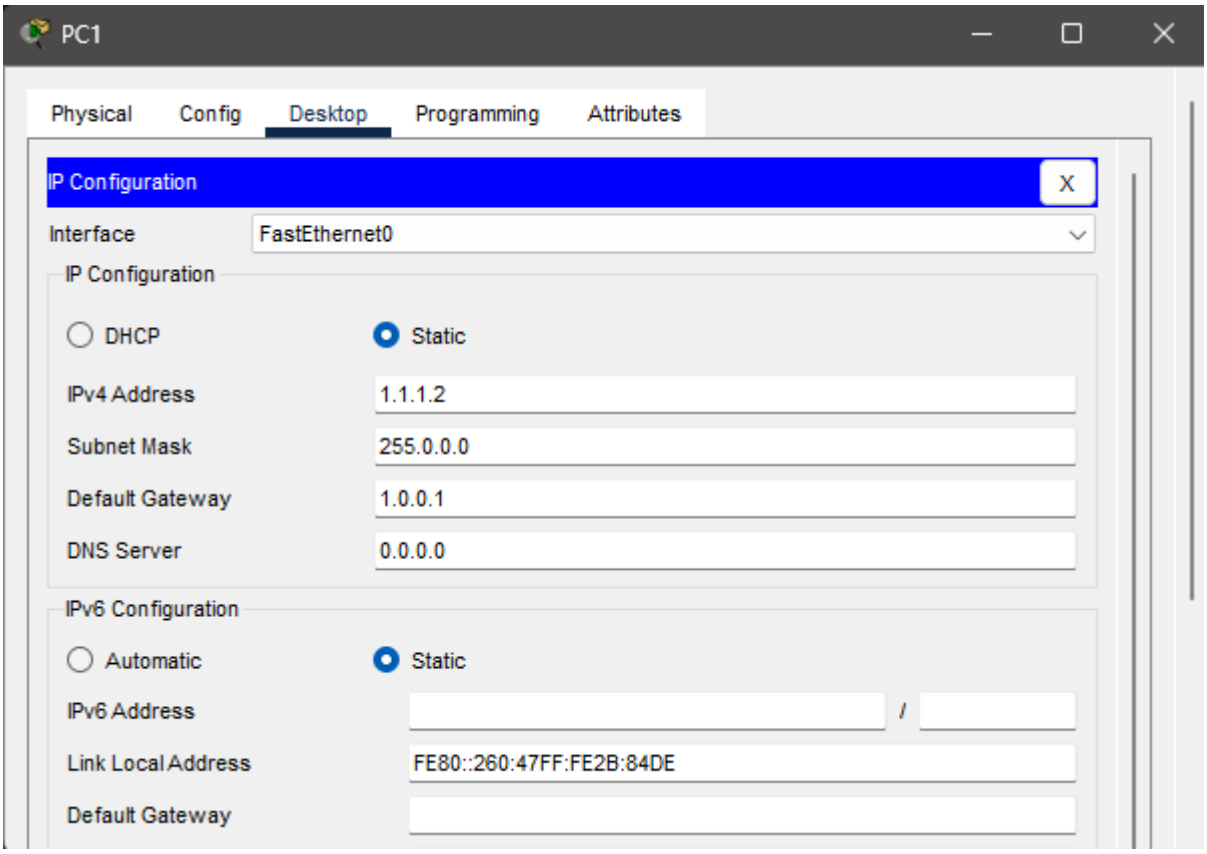


HOST PC IP ADDRESS :

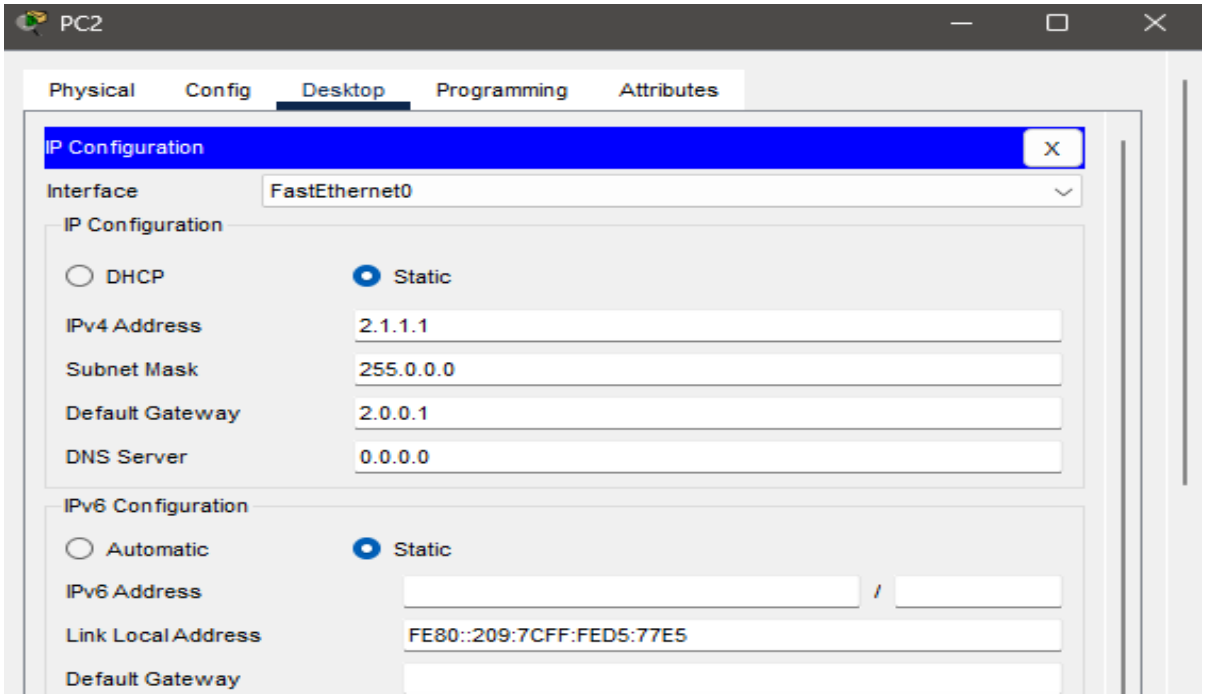
PC0 IP CONFIGURATION



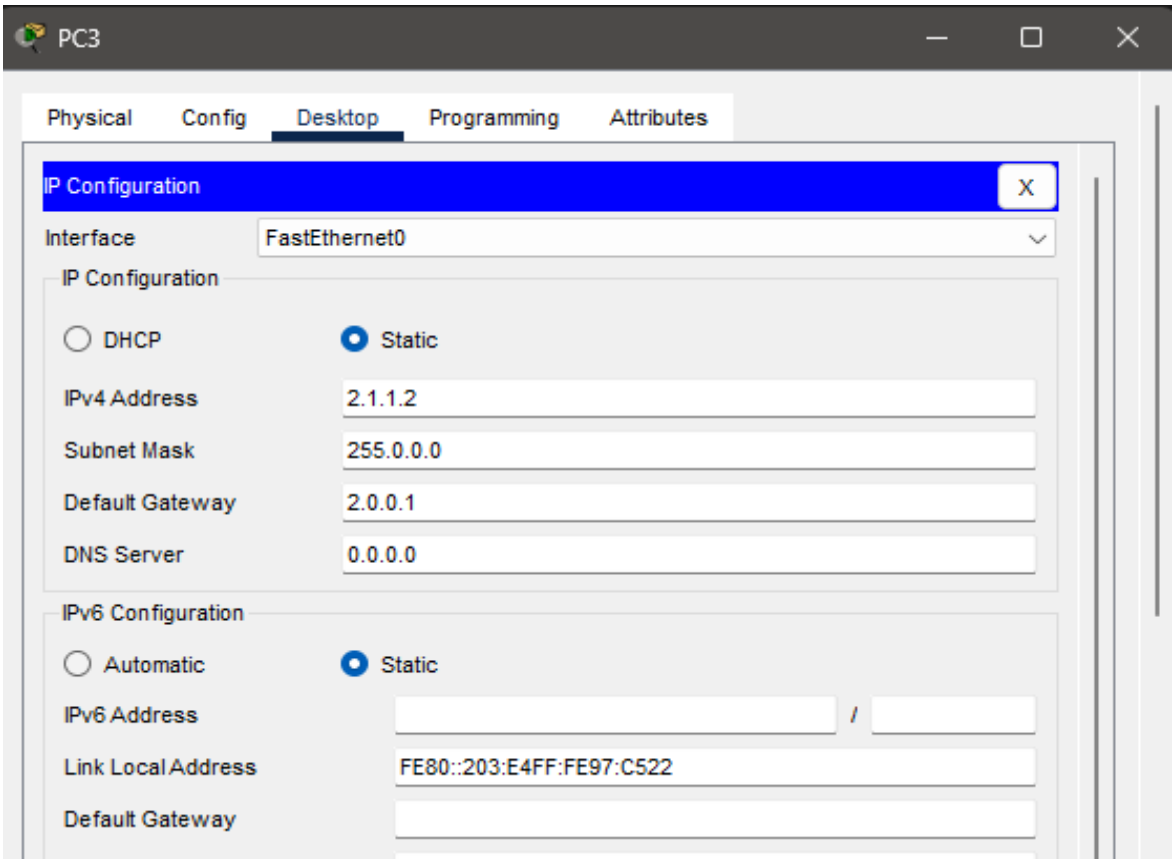
PC1 IP CONFIGURATION :



PC2 IP CONFIGURATION :

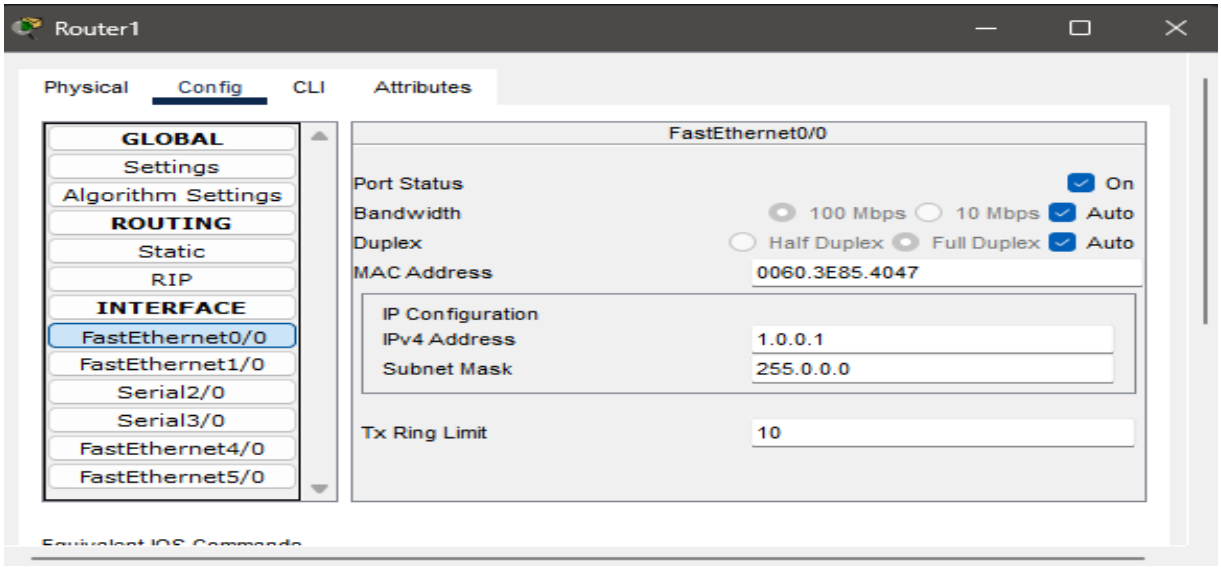


PC3 IP CONFIGURATION :

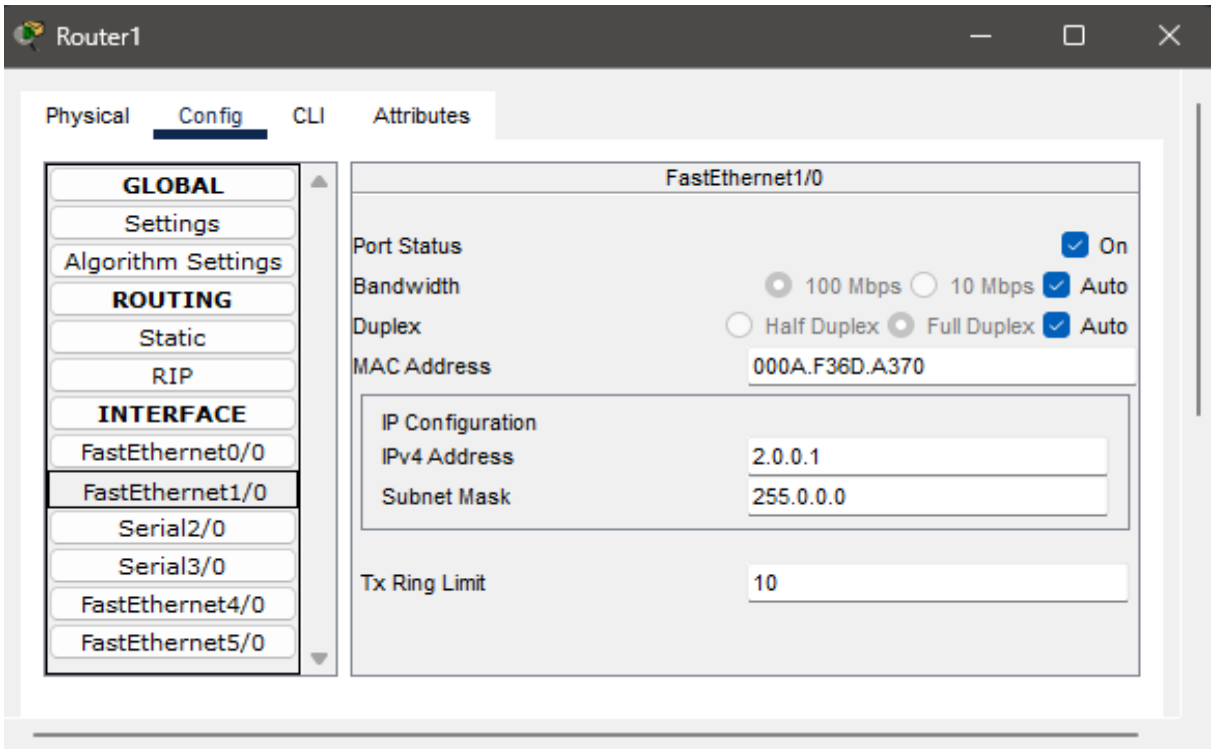


ROUTER 0 CONFIGURATION :

FASTETHERNET 0/0 CONFIGURATION



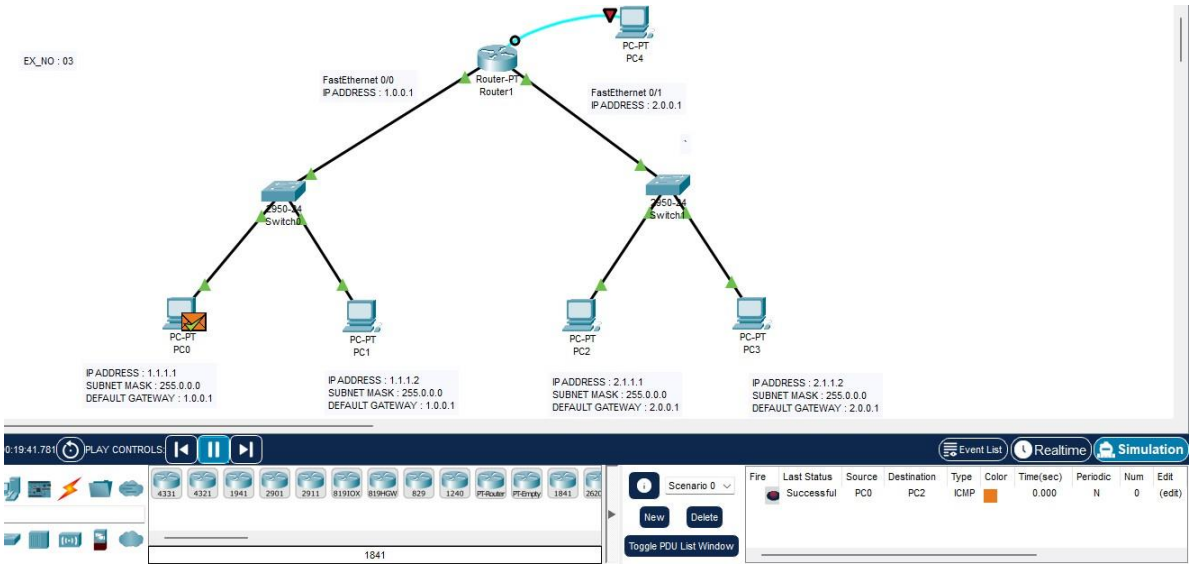
FASTETHERNET 1/0 CONFIGURATION



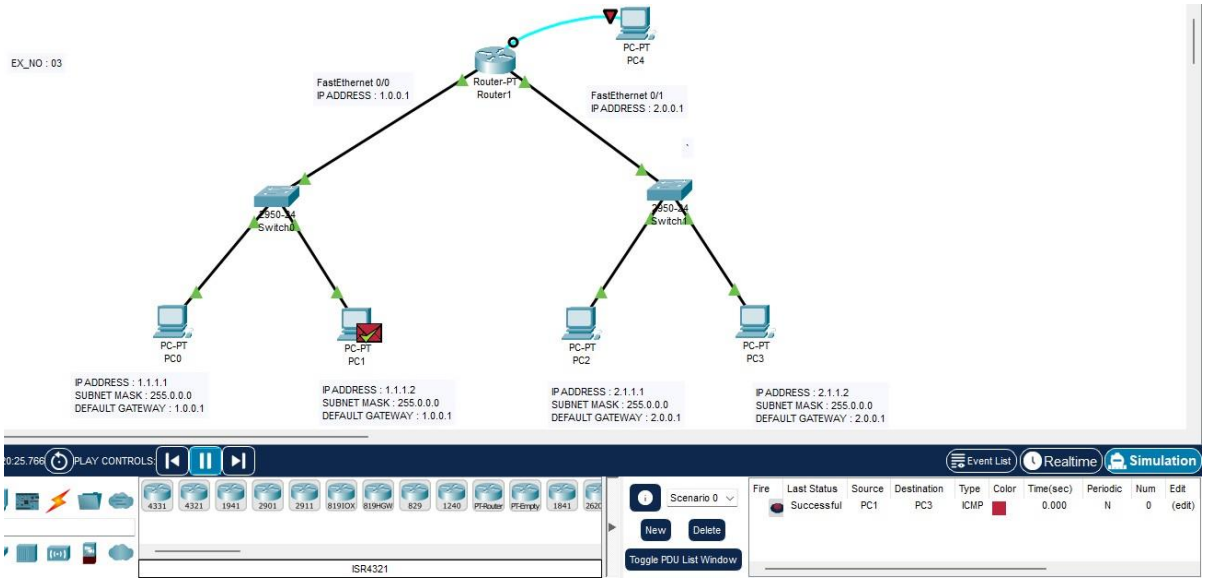
VERIFY LAN NETWORK CONNECTIVITY :

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 :

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

AIM :

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

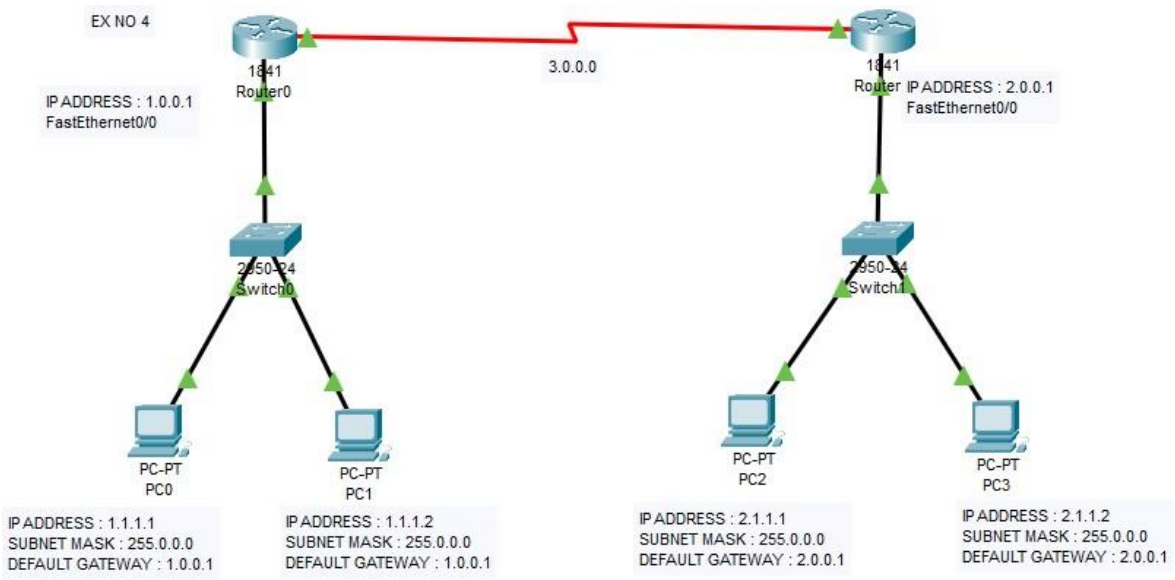
REQUIREMENTS :

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

PROCEDURES :

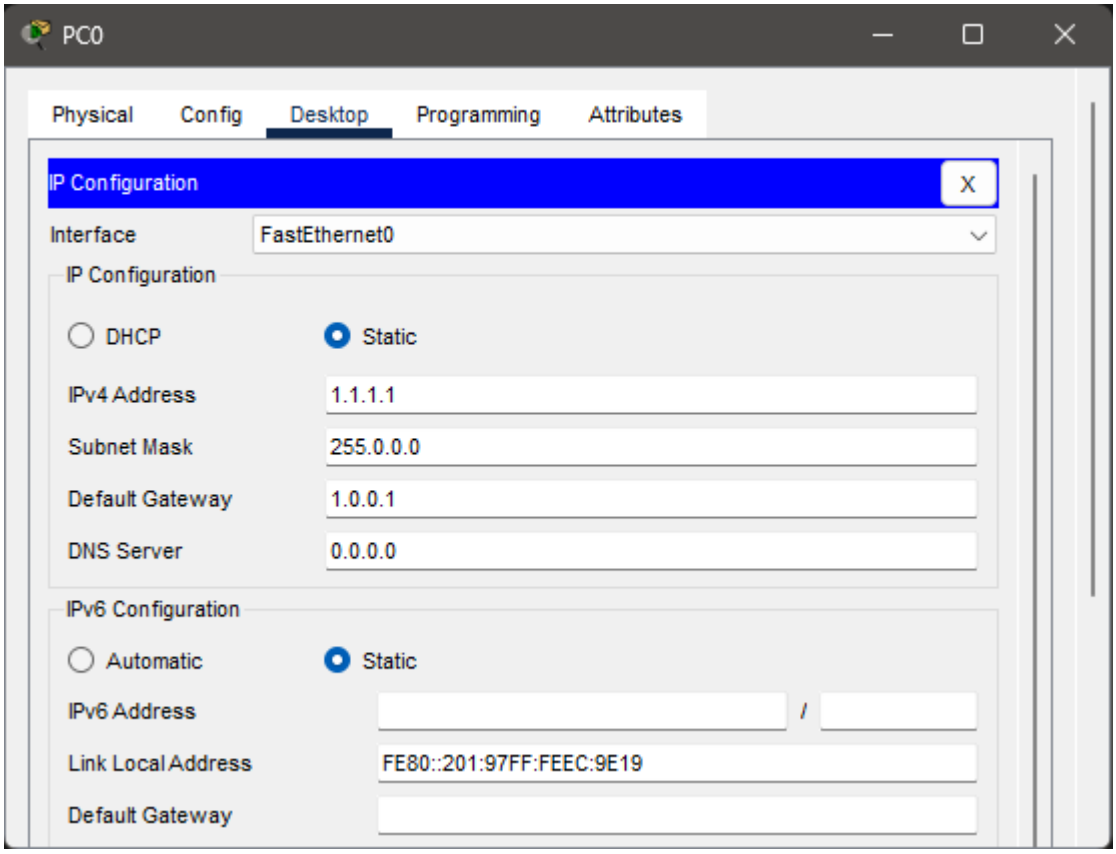
- Open CISCO PACKET TRACER software.
- Draw the Four PC using END Device Icons.
- Draw the Cisco 24 Port Switch Using Switch icon lists.
- Draw the Cisco 1841 Routers using Router icon lists.
- Make the Connections using Copper-Straight-Through Ethernet Cables and Serial DTE Cables.
- Enter the IP Address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

NETWORK TOPOLOGY :

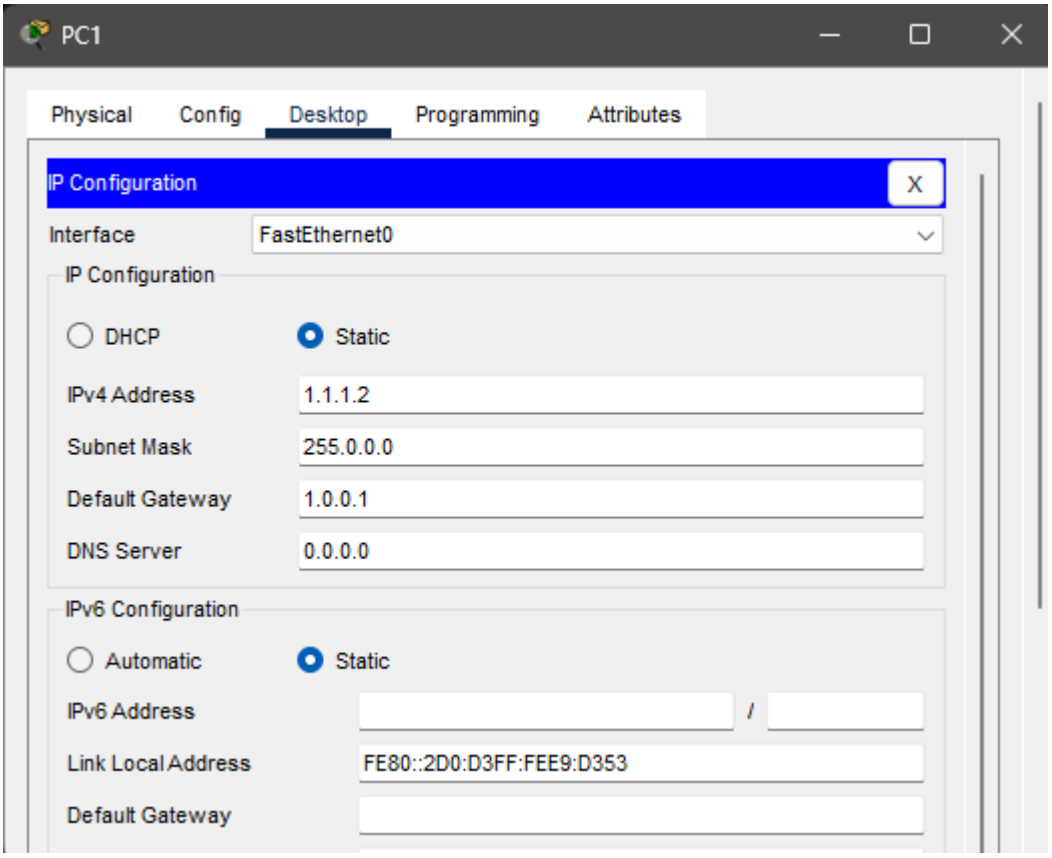


HOST PC IP ADDRESS :

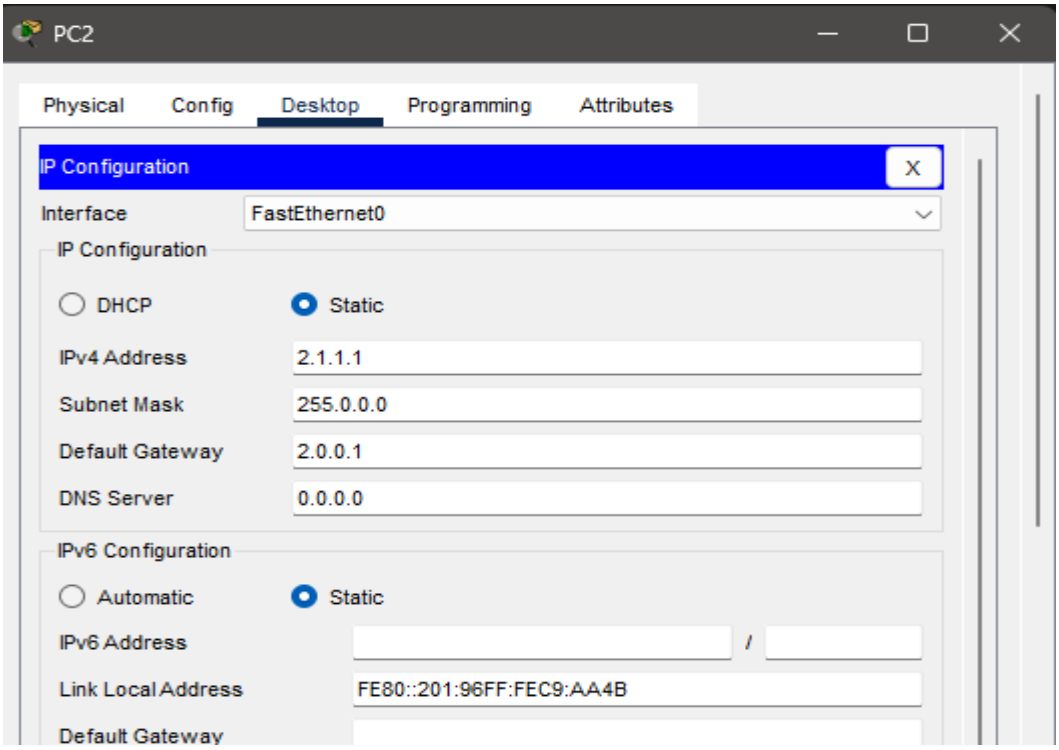
PC0 IP CONFIGURATION



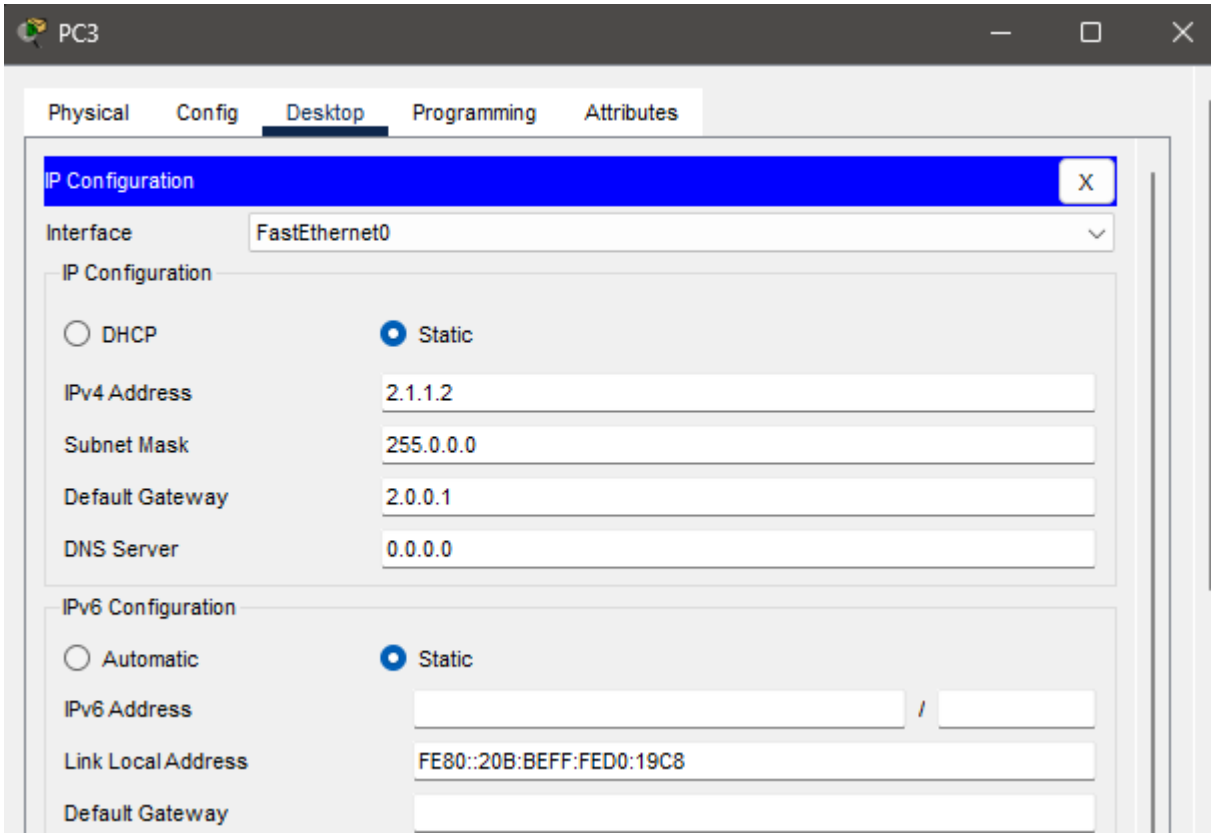
PC1 IP CONFIGURATION



PC2 IP CONFIGURATION

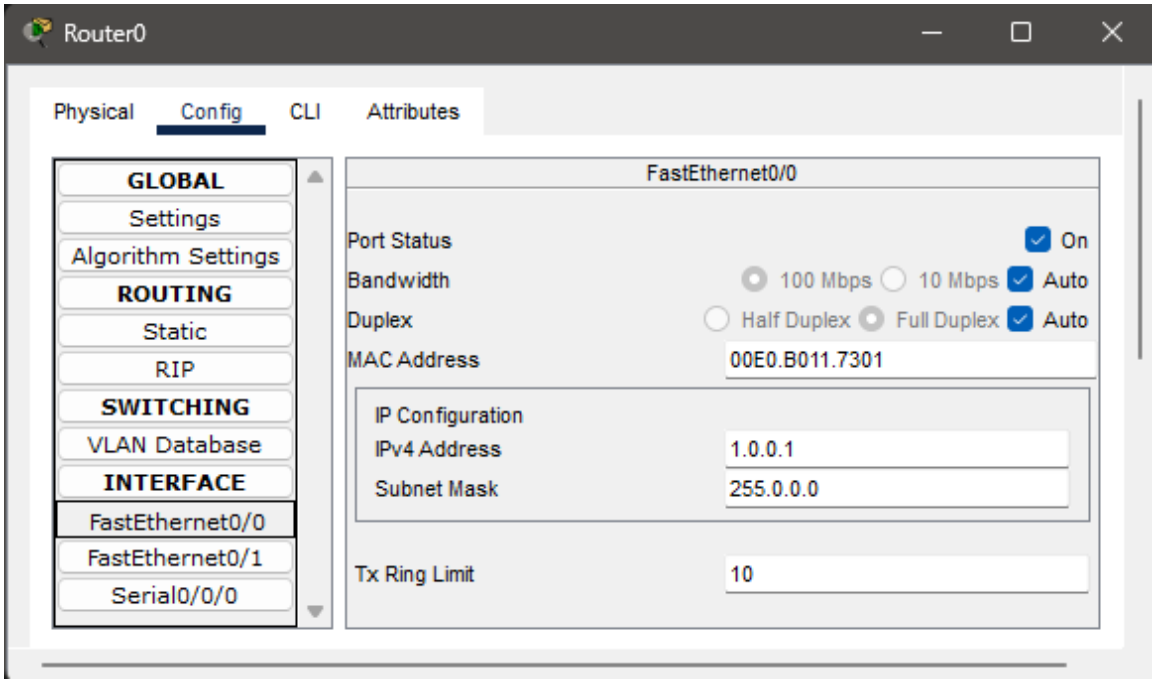


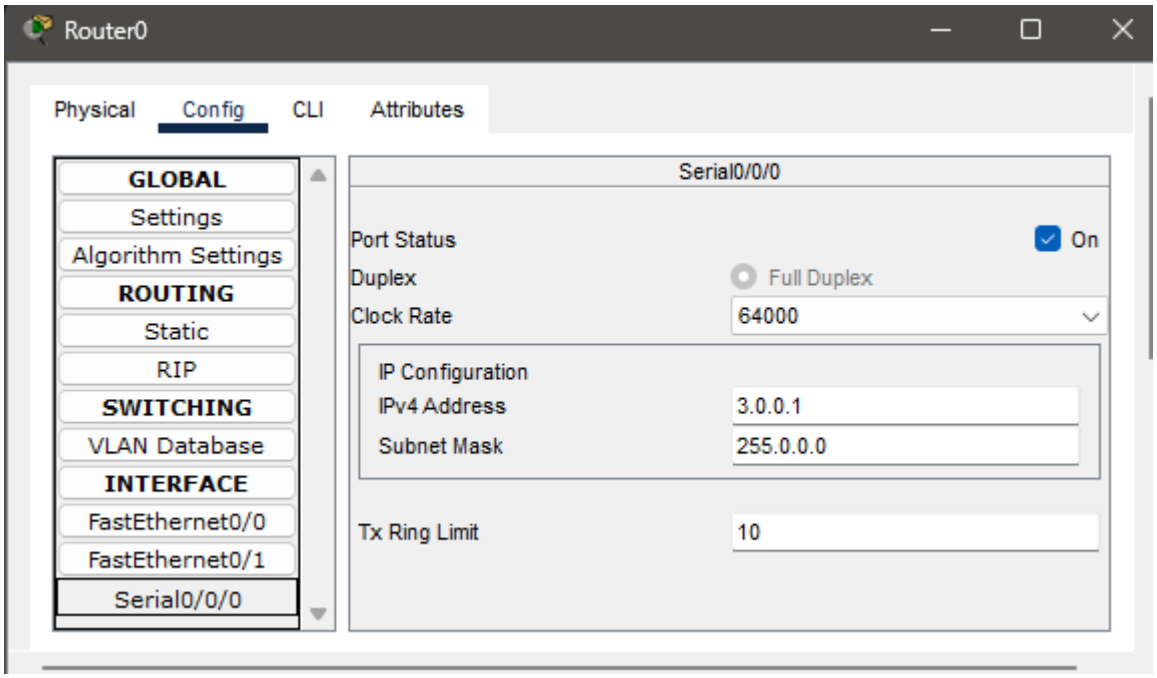
PC3 IP CONFIGURATION



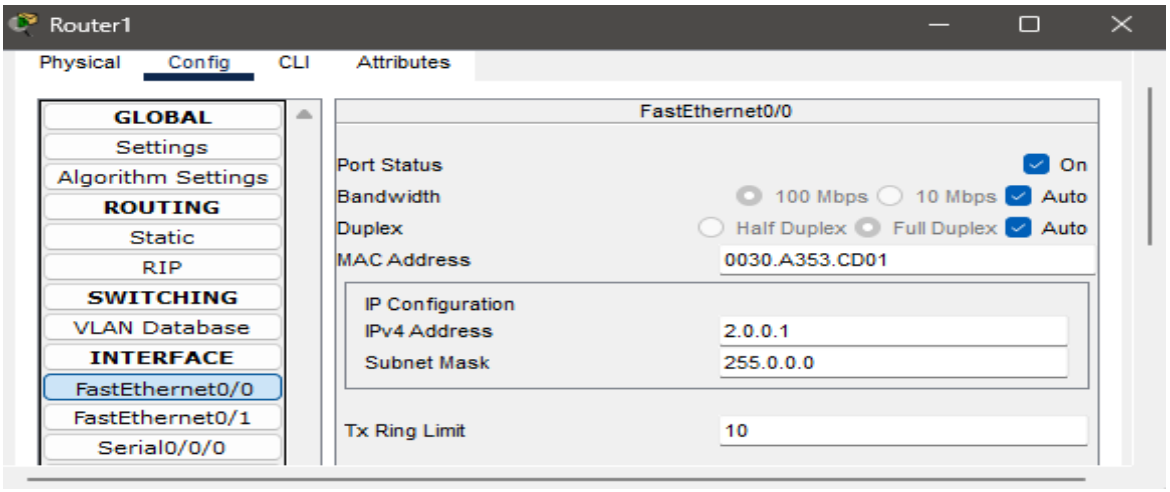
ROUTERS CONFIGURATION :

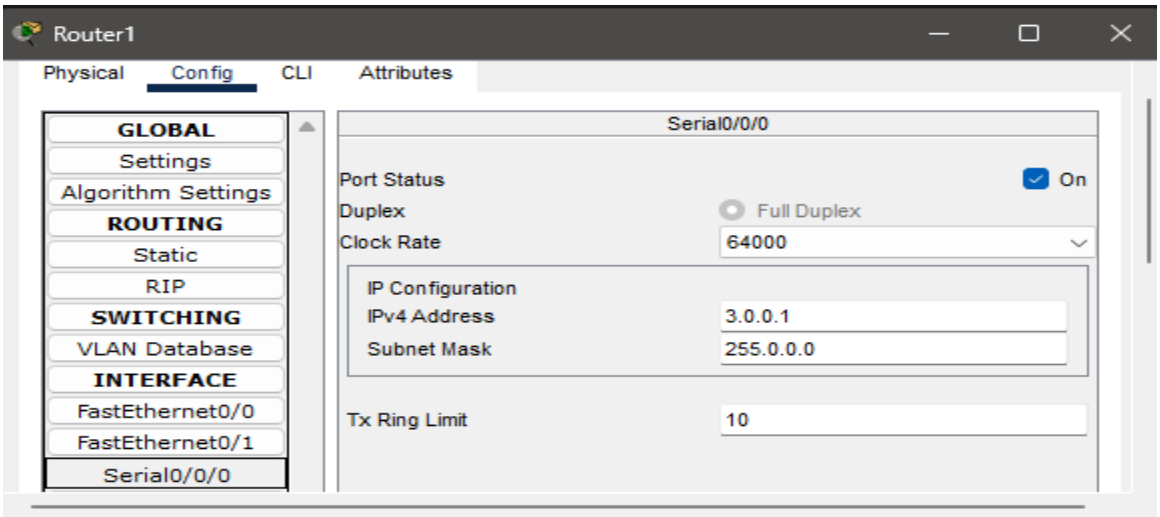
ROUTER 0 CONFIGURATION





ROUTER 1 CONFIGURATION :

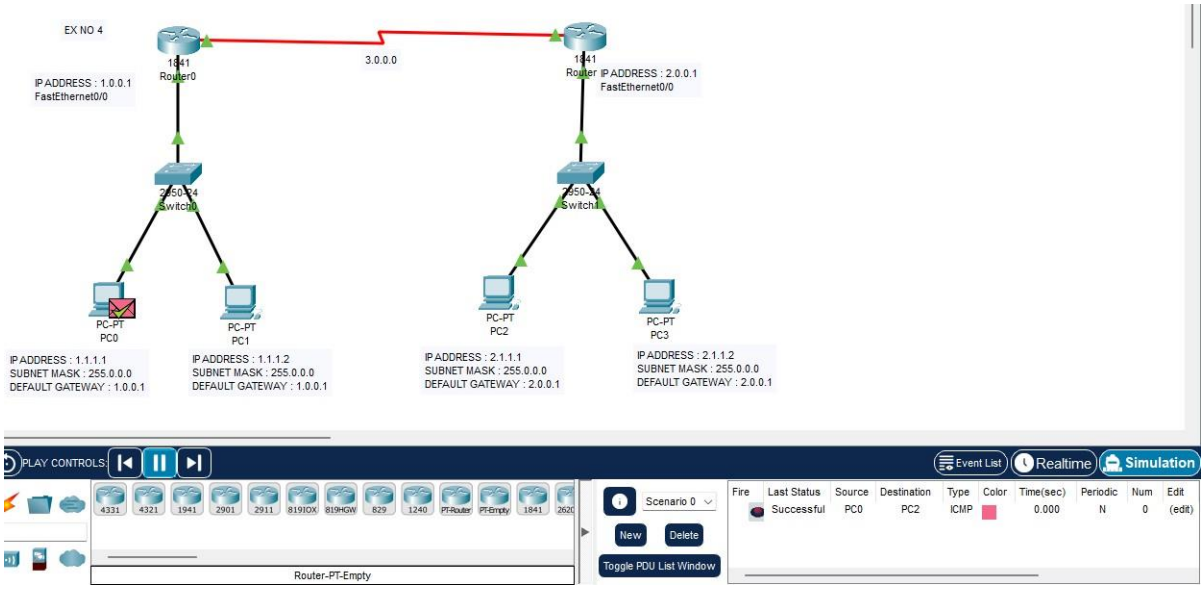




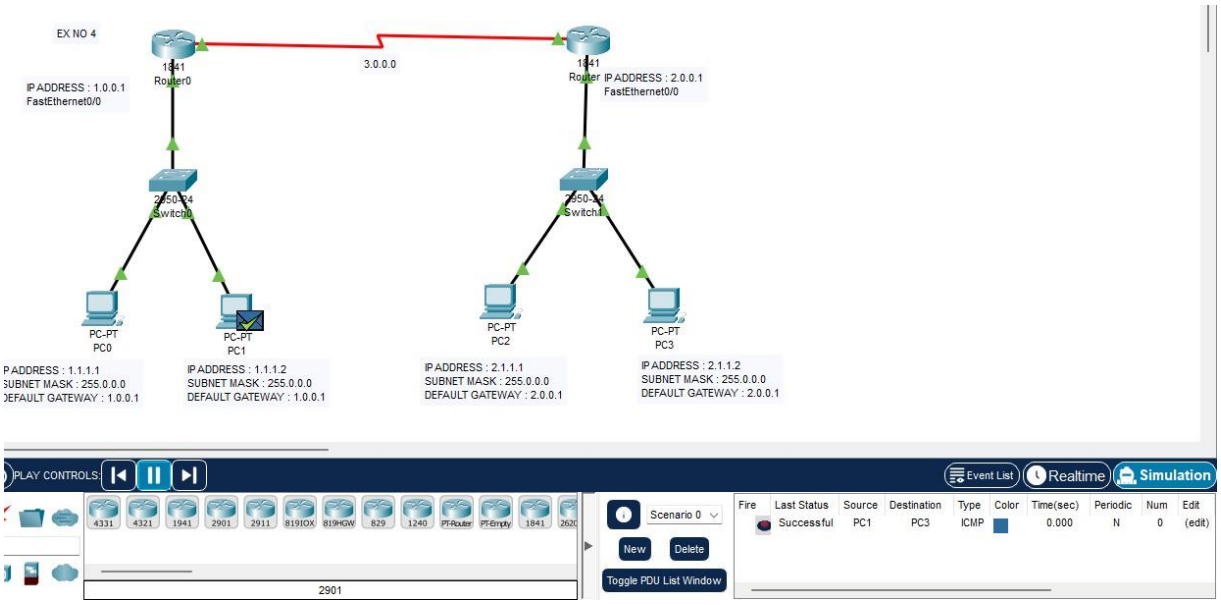
VERIFY LAN NETWORK CONNECTIVITY :

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 :

CONNECTING TWO LANs USING BRIDGE

AIM :

To establish connection between two LANs by using Bridge.

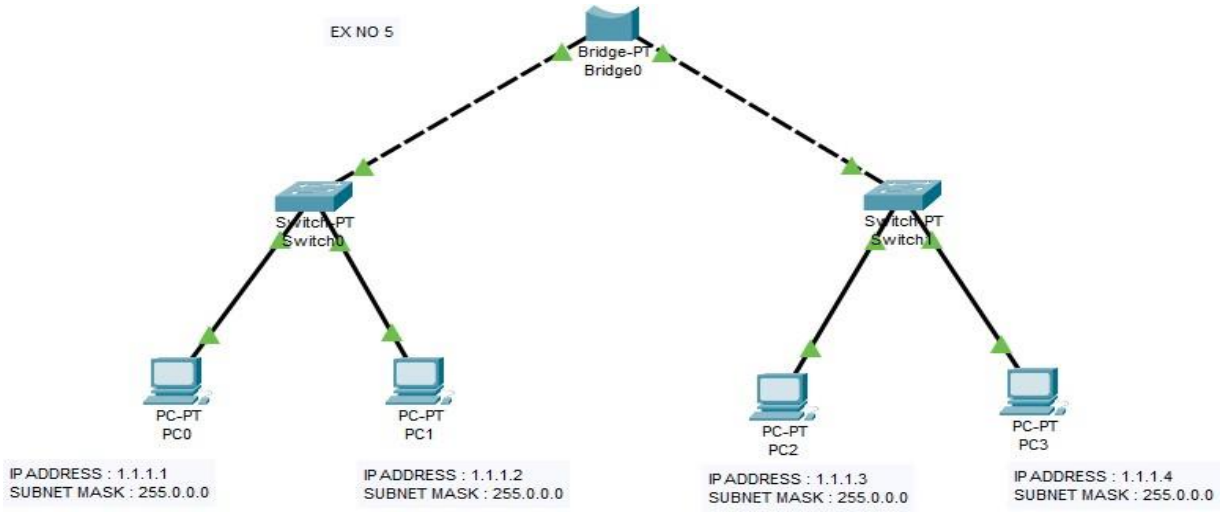
REQUIREMENTS :

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

PROCEDURES :

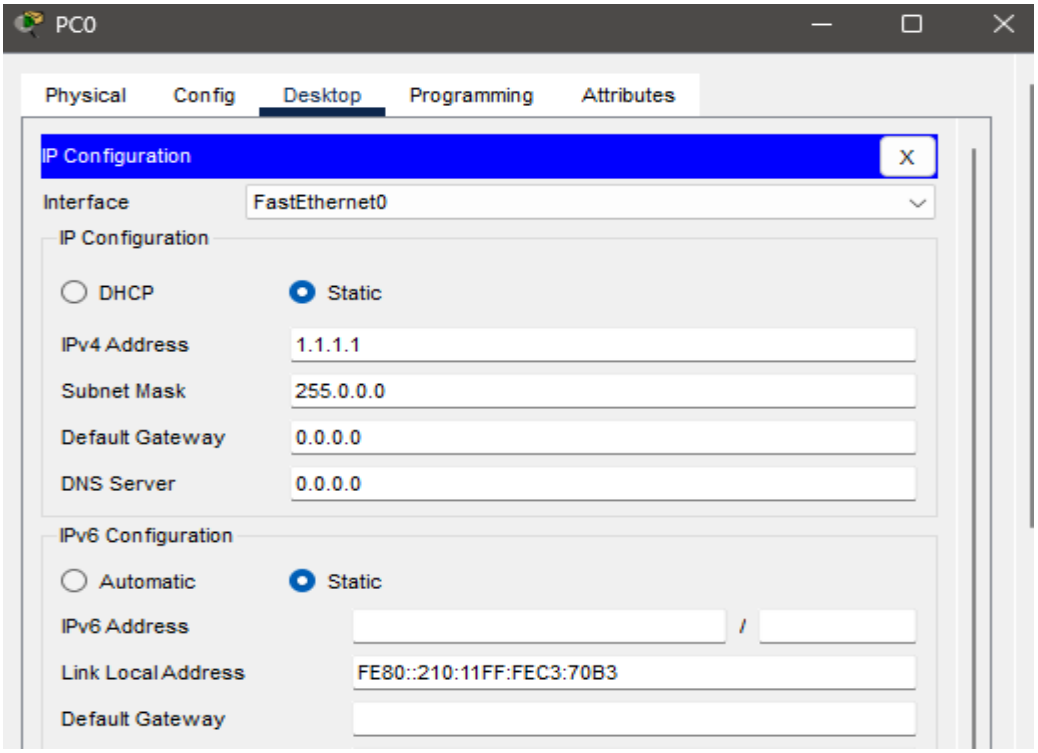
- Open CISCO PACKET TRACER software.
- Draw the Four PC using END Device Icons.
- Draw the Cisco PT Port Switch Using Switch icon lists.
- Draw the Cisco PT Bridge.
- Make the Connections using Copper-Straight-Through Ethernet Cables.
- Enter the IP Address To Each Machine.
- Check the Network Connections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

NETWORK TOPOLOGY

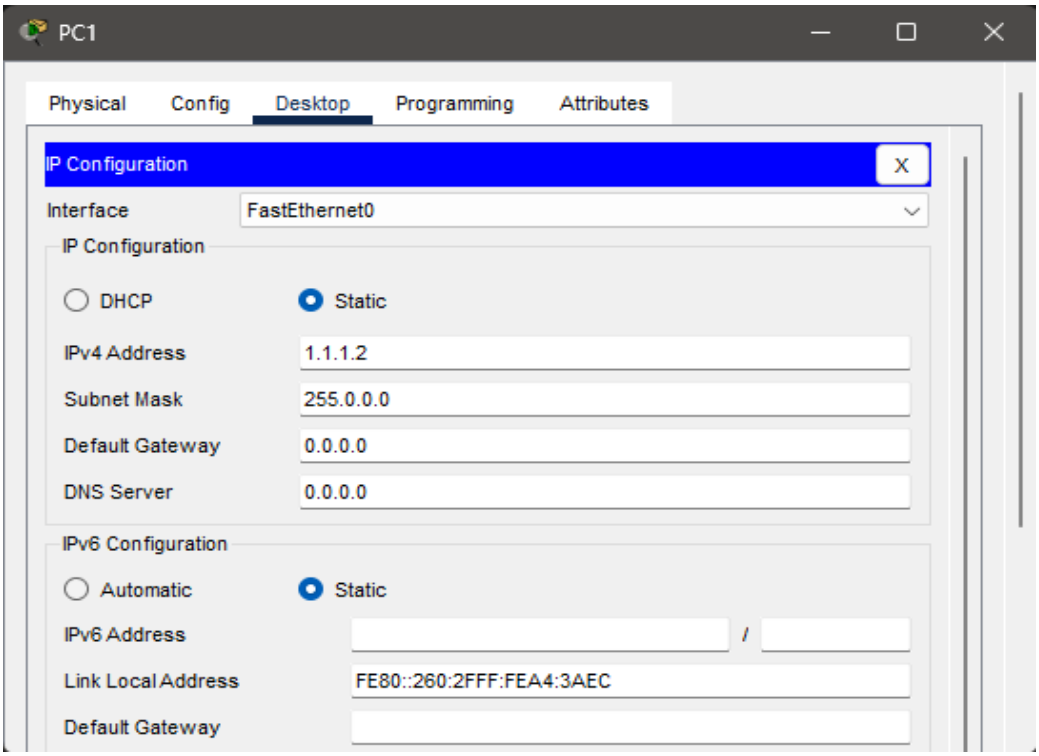


HOST PC IP ADDRESS :

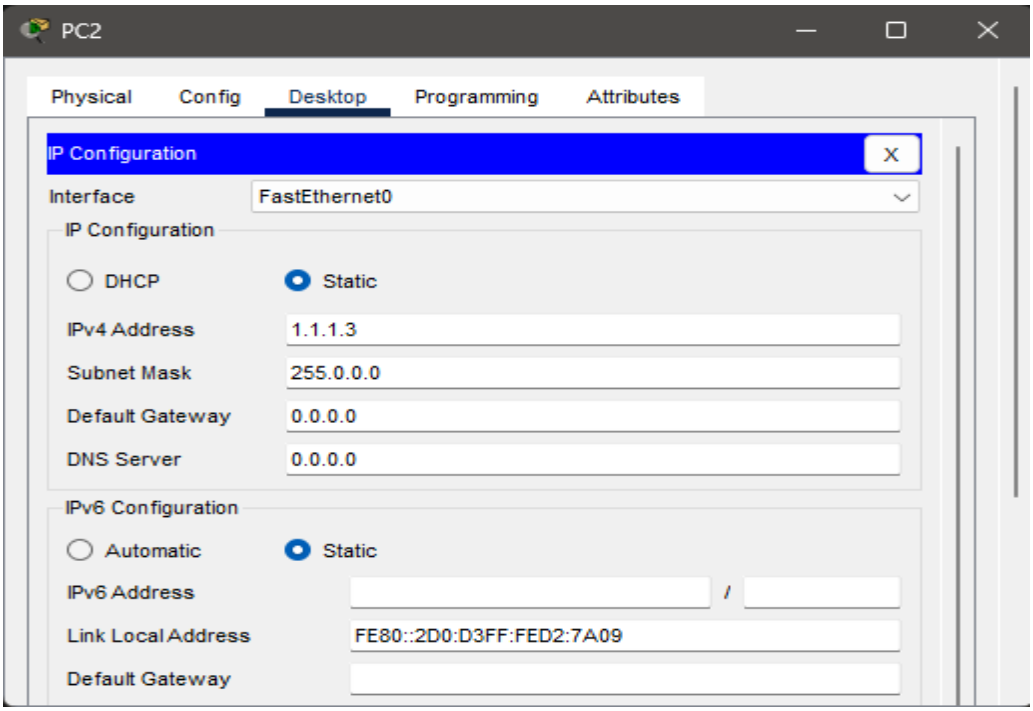
PC0 IP CONFIGURATION



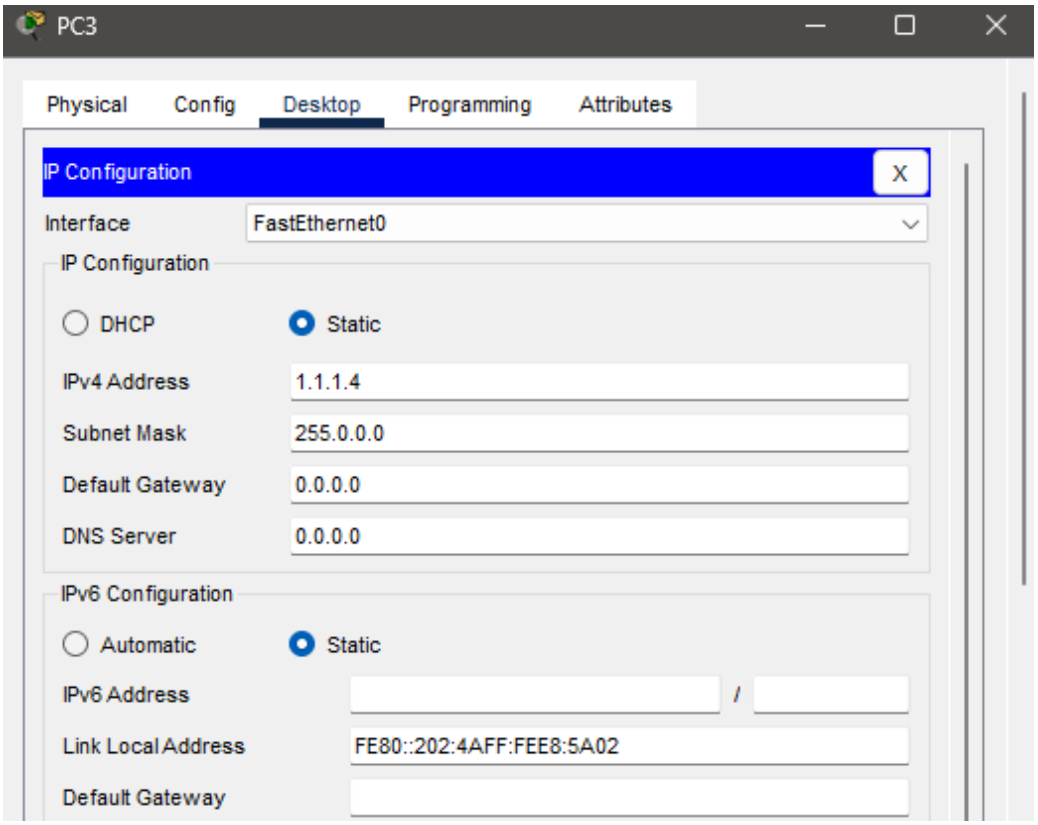
PC1 IP CONFIGURATION



PC2 IP CONFIGURATION



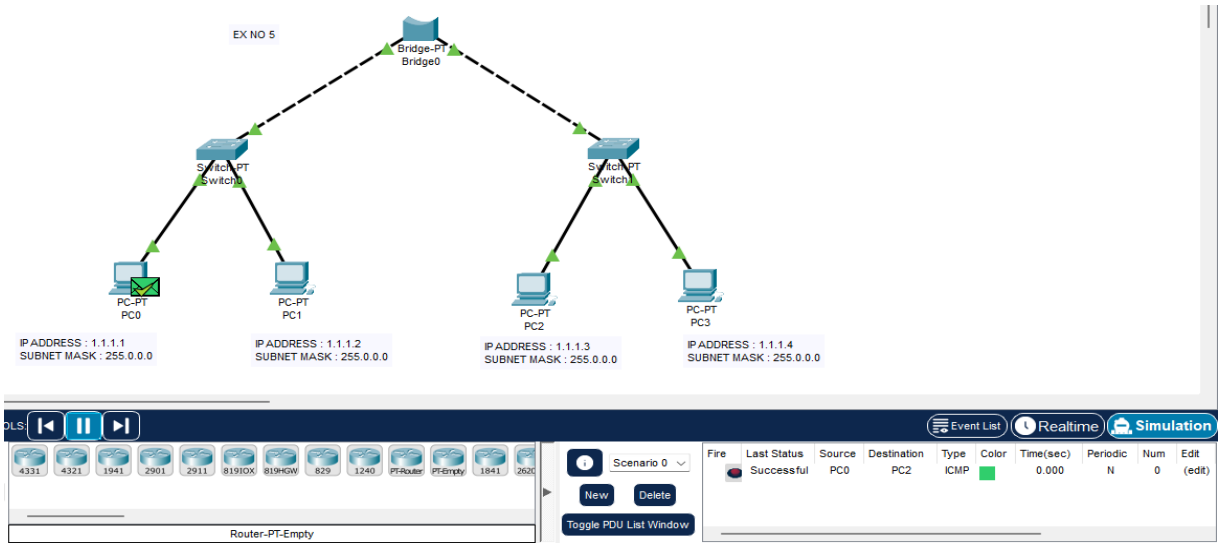
PC3 IP CONFIGURATION



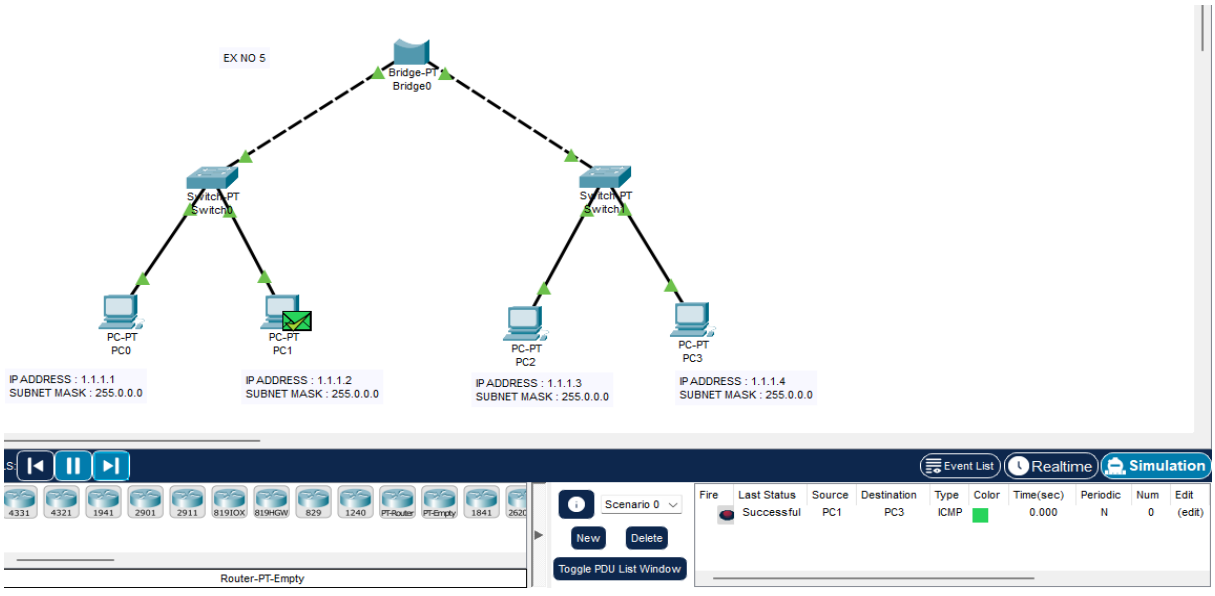
VERIFY LAN NETWORK CONNECTIVITY :

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 :

DESIGNING RING AND MESH TOPOLOGIES USING CISCO PACKET TRACER

AIM :

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

REQUIREMENTS :

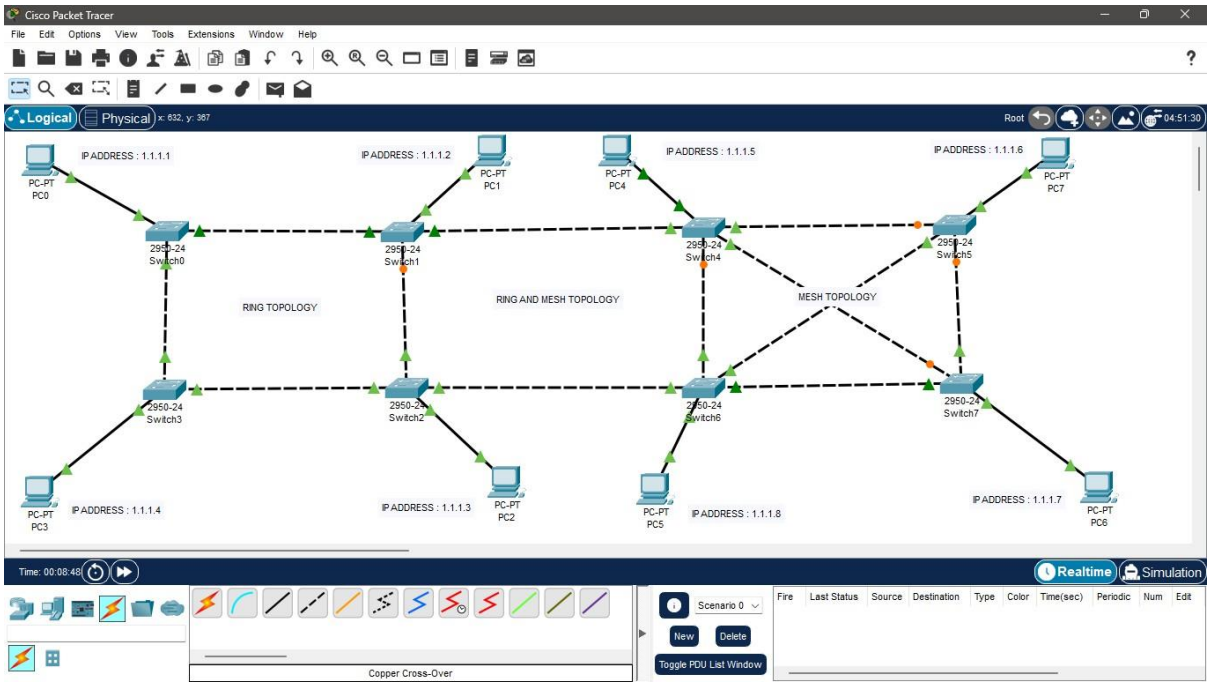
- 8 windows PC.
- 8 Switch (2950-24).
- 8 Straight Line LAN Cables.
- 12 Cross Over Cables.
- Cisco Packet Tracer.

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).

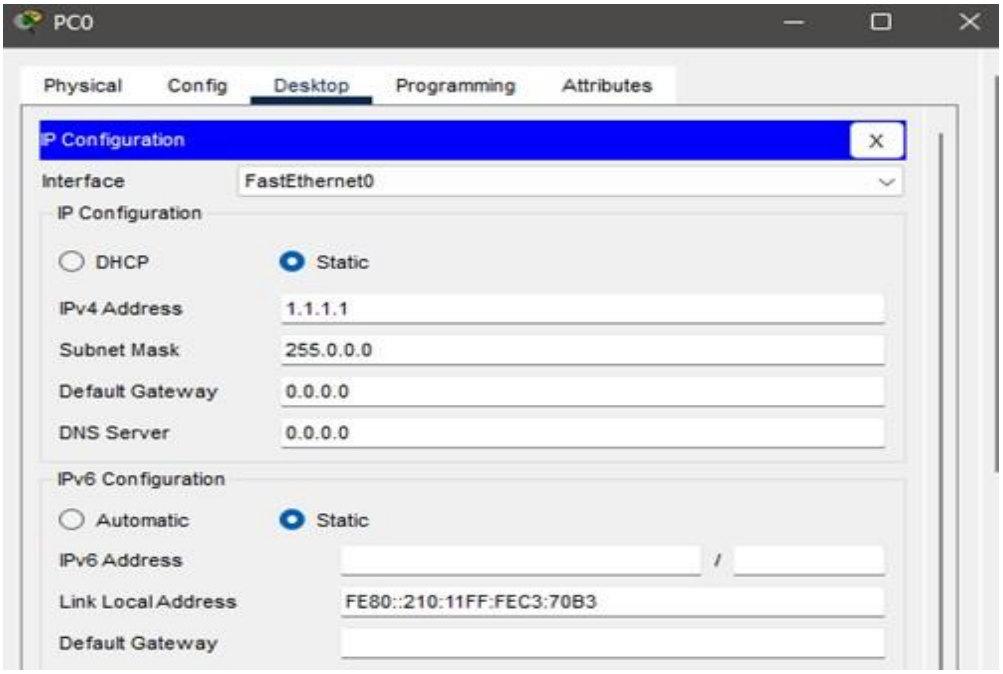
NETWORK TOPOLOGY

RING AND MESH TOPOLOGY

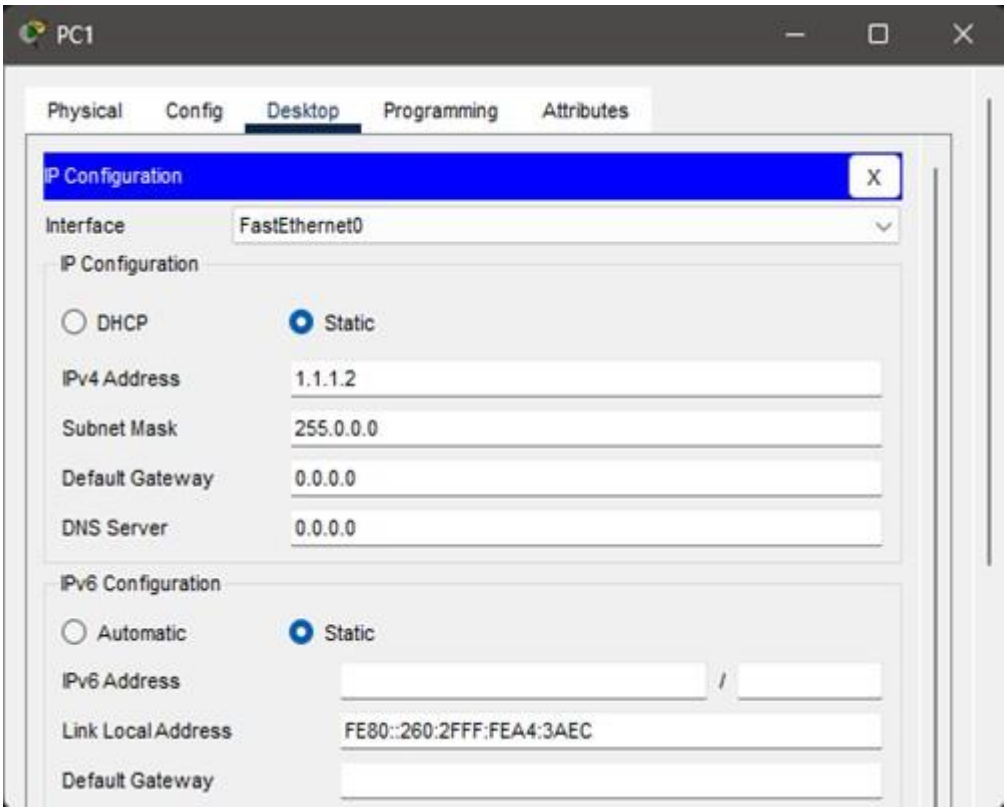


HOST PC IP ADDRESS :

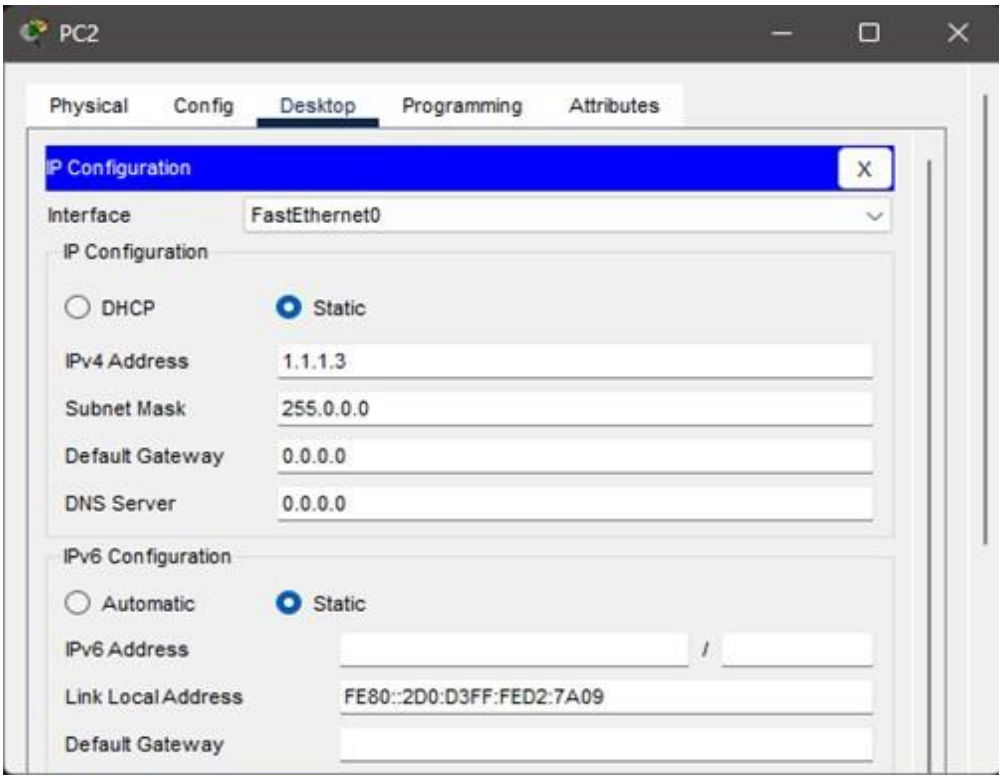
PC0 IP CONFIGURATION



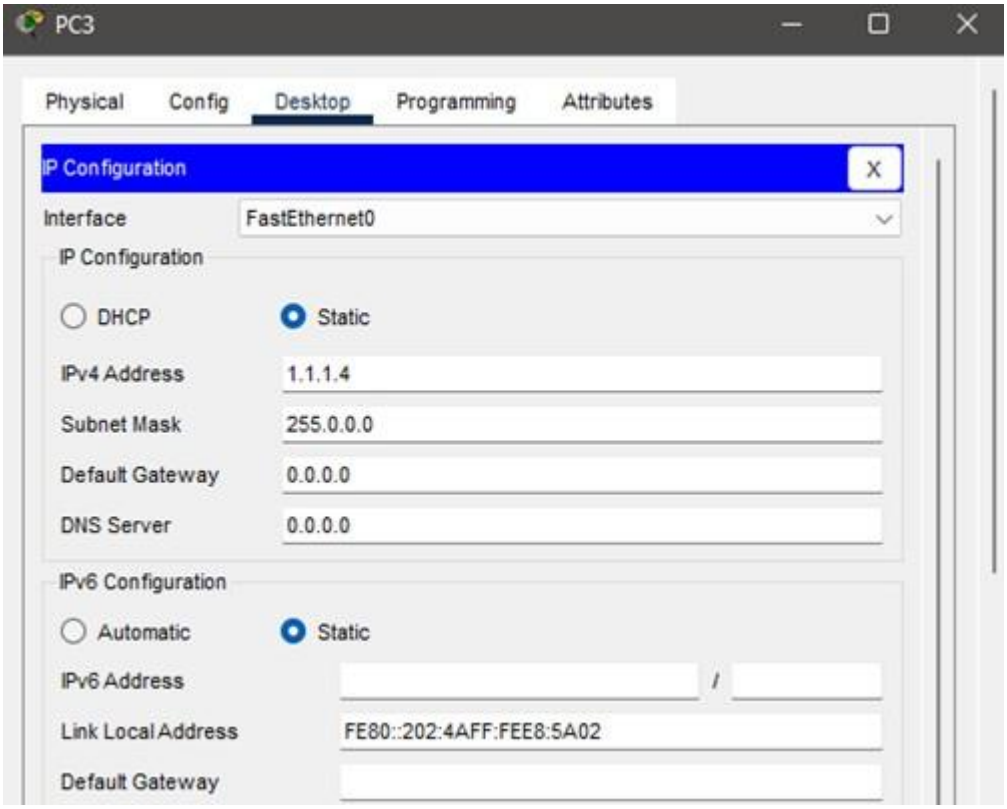
PC1 IP CONFIGURATION



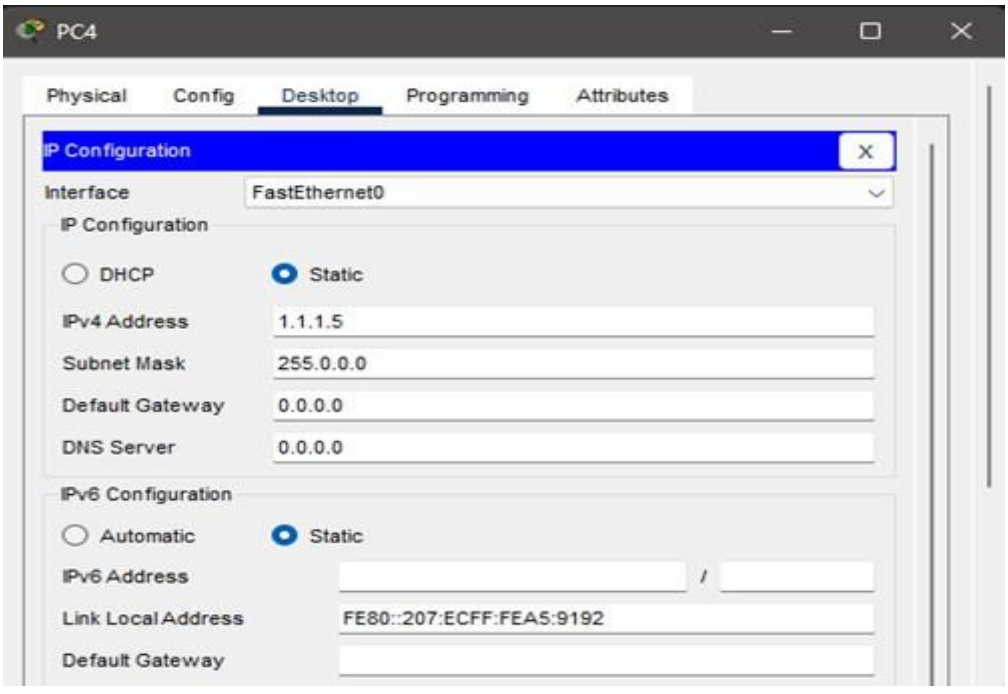
PC2 IP CONFIGURATION



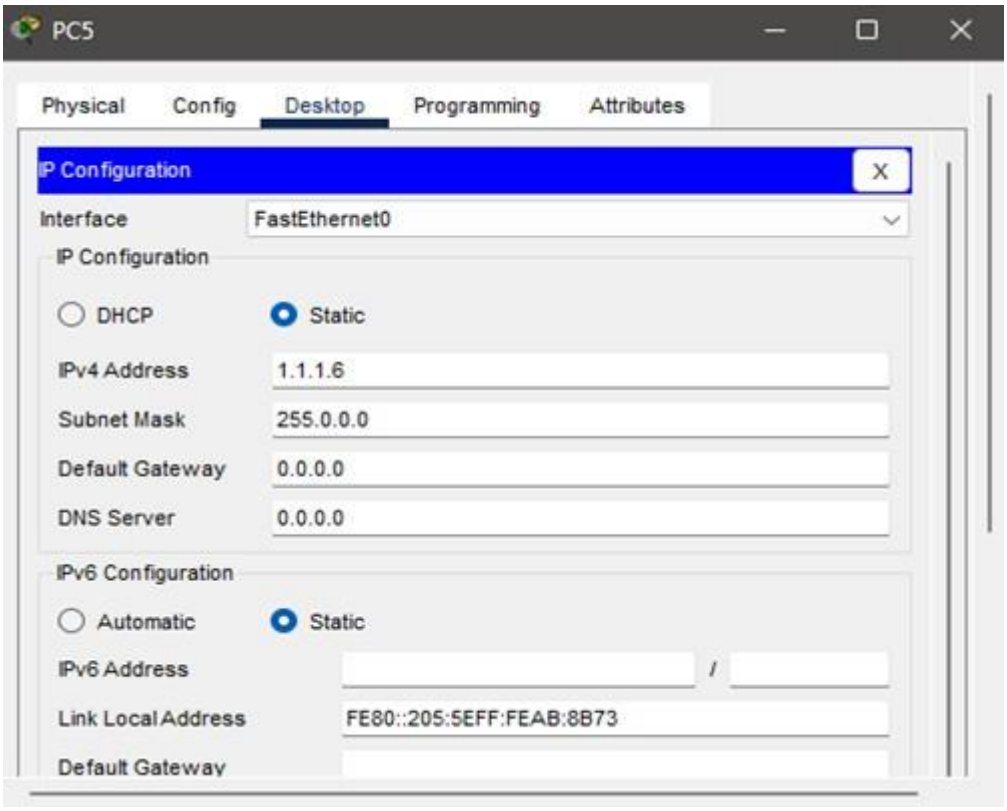
PC3 IP CONFIGURATION



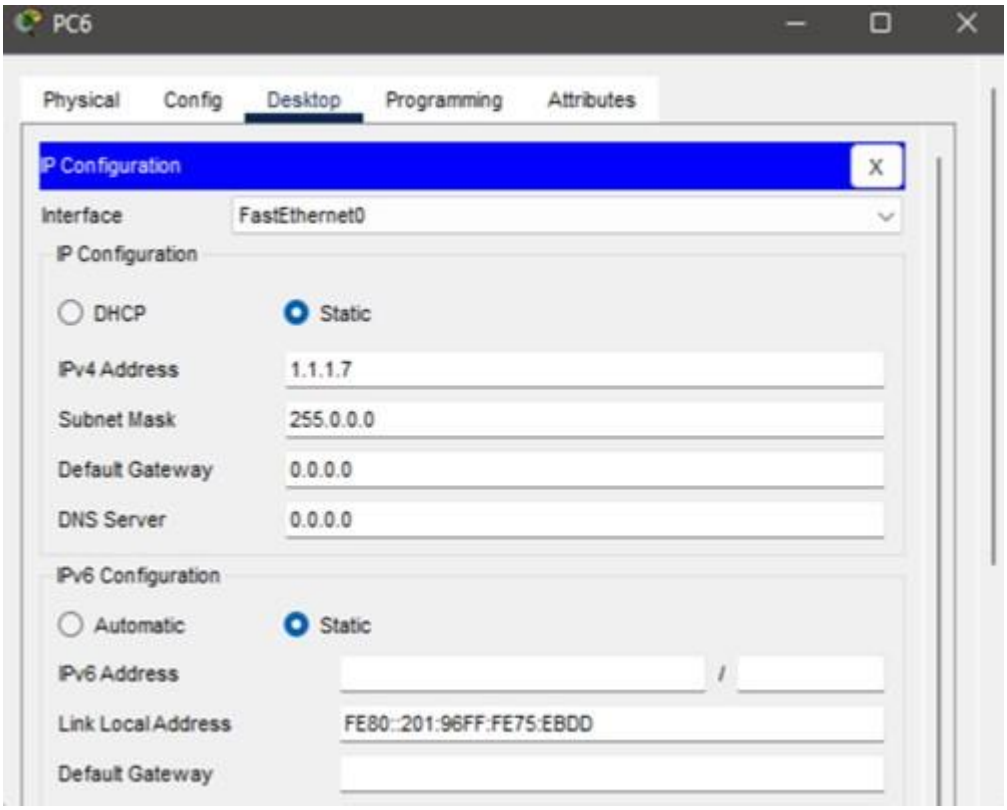
PC4 IP CONFIGURATION



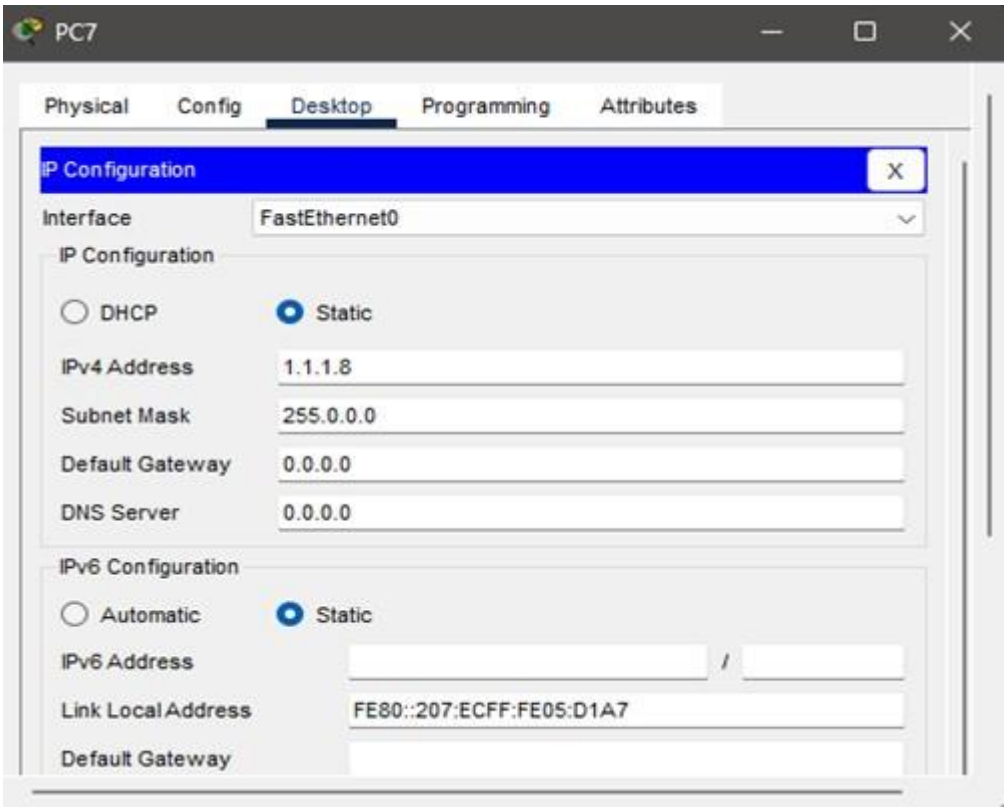
PC5 IP CONFIGURATION



PC6 IP CONFIGURATION



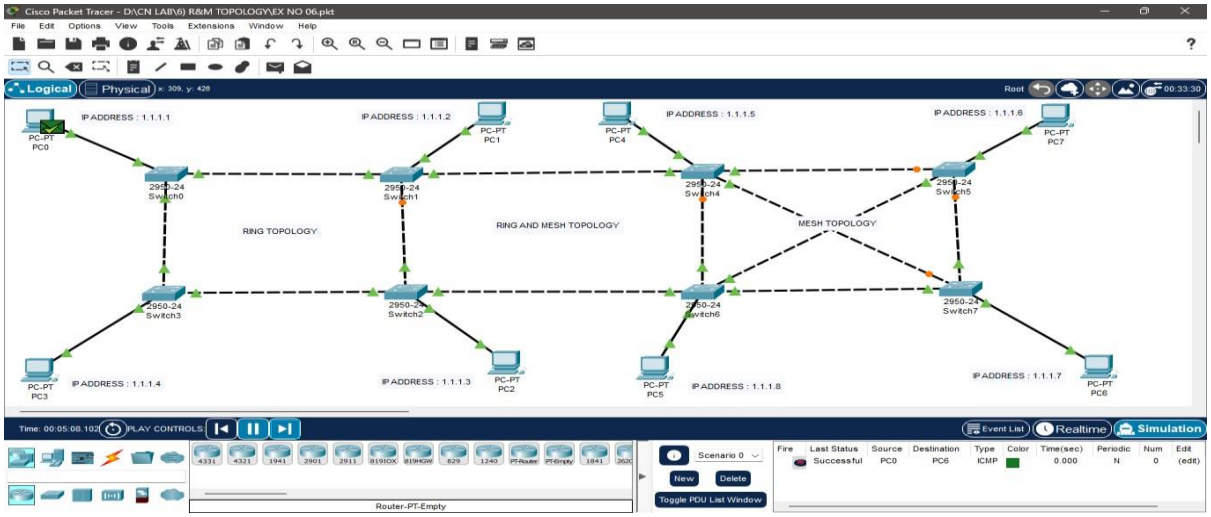
PC7 IP CONFIGURATION



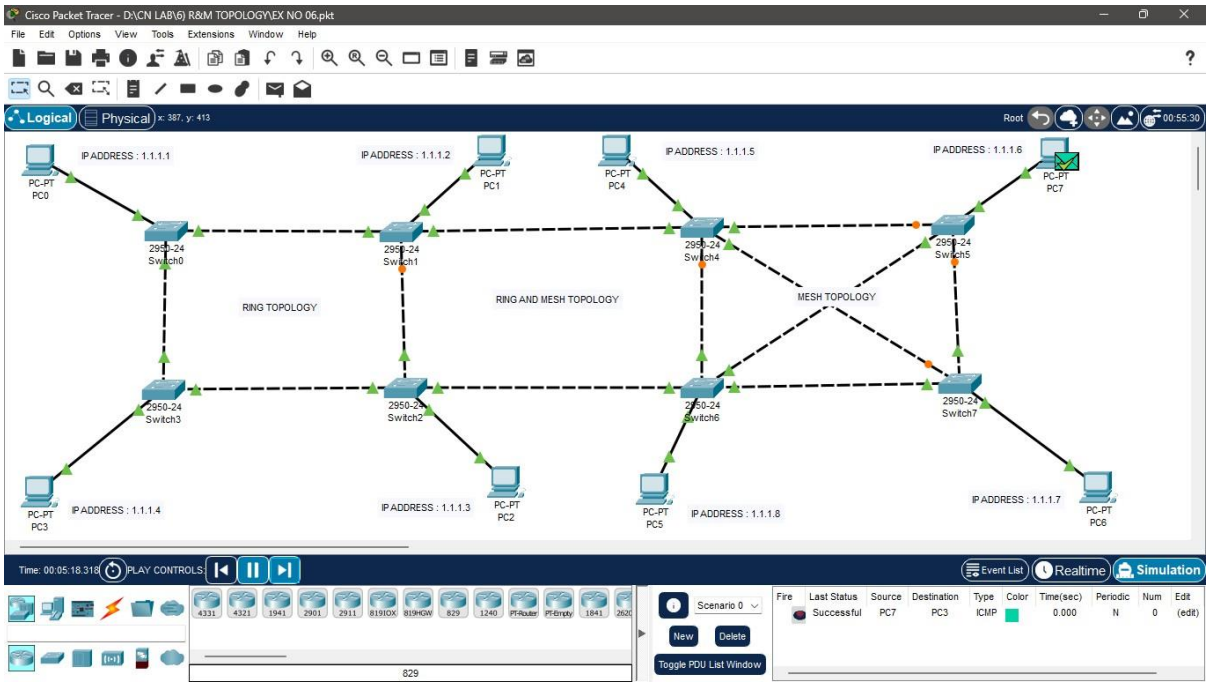
VERIFY LAN NETWORK CONNECTIVITY :

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 :

DESIGNING BUS AND STAR TOPOLOGIES USING CISCO PACKET TRACER

AIM :

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

REQUIREMENTS :

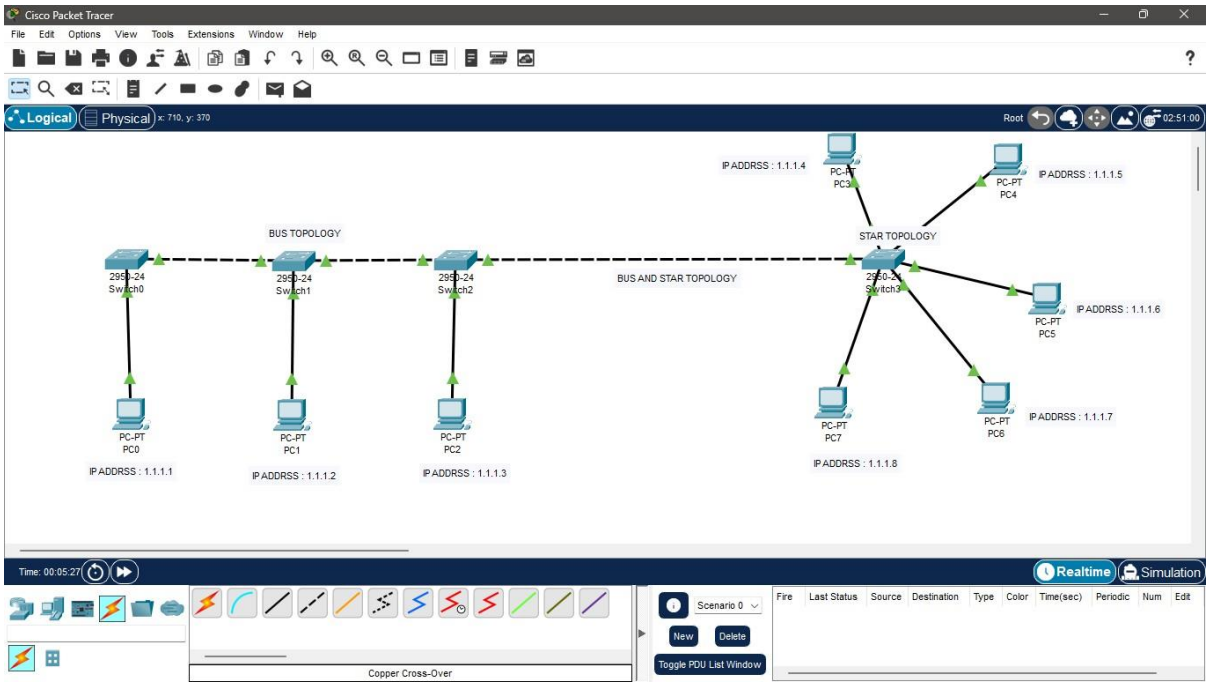
- 8 windows PC.
- 4 Switch (2950-24).
- 8 Straight Line LAN Cables.
- 3 Cross Over Cables.
- Cisco Packet Tracer.

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).

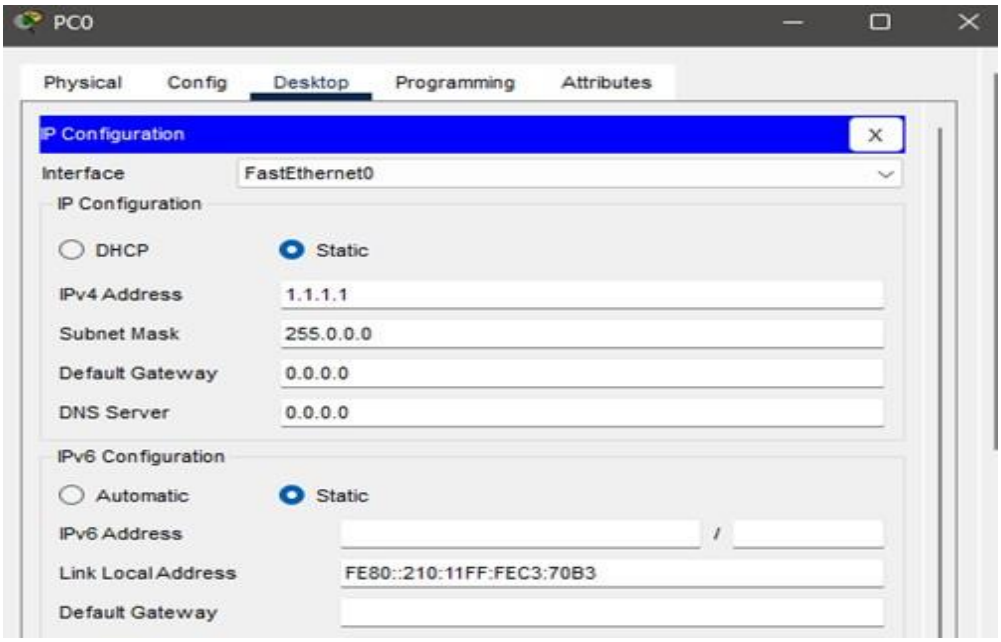
NETWORK TOPOLOGY

BUS AND STAR TOPOLOGY

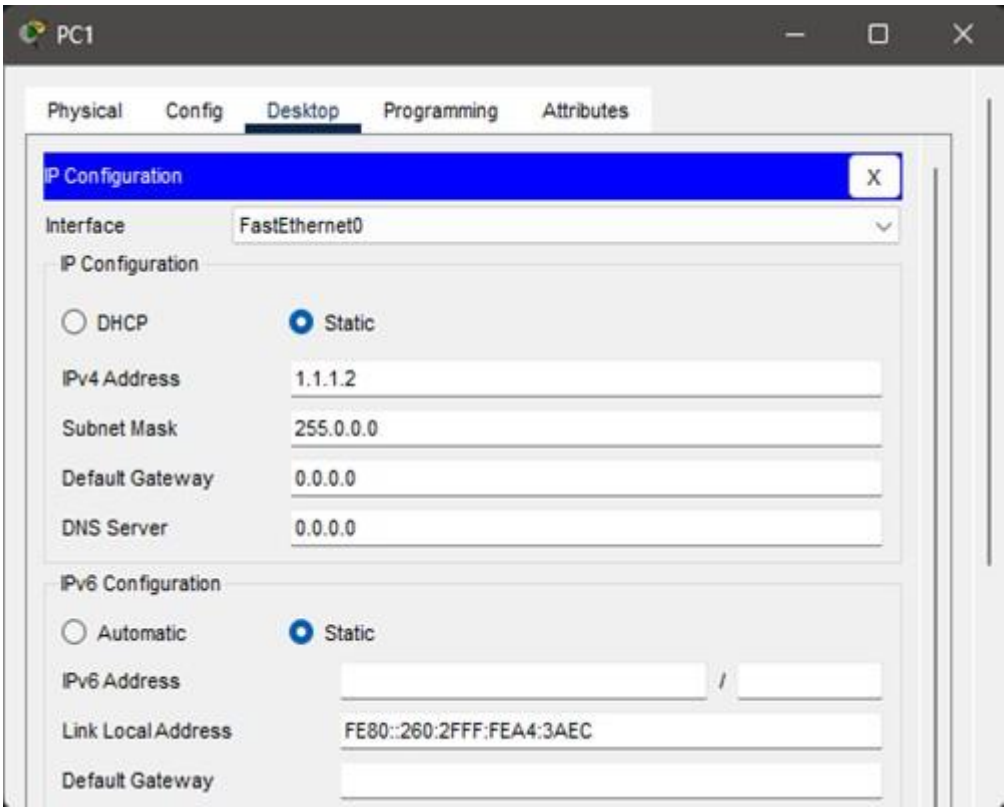


HOST PC IP ADDRESS :

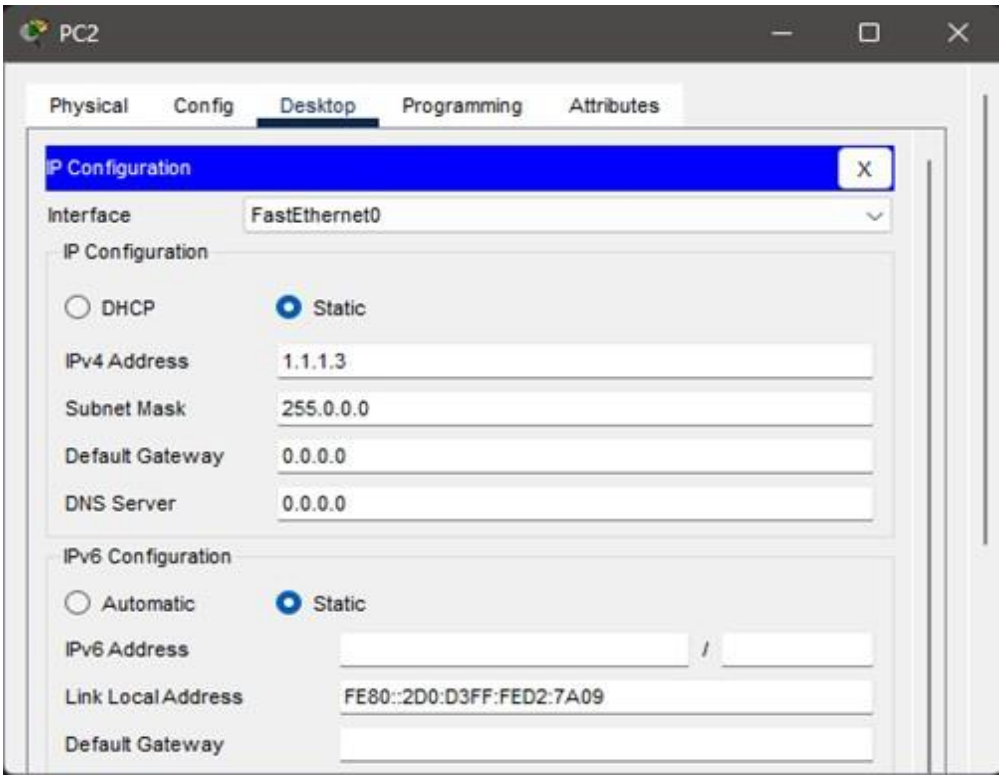
PC0 IP CONFIGURATION



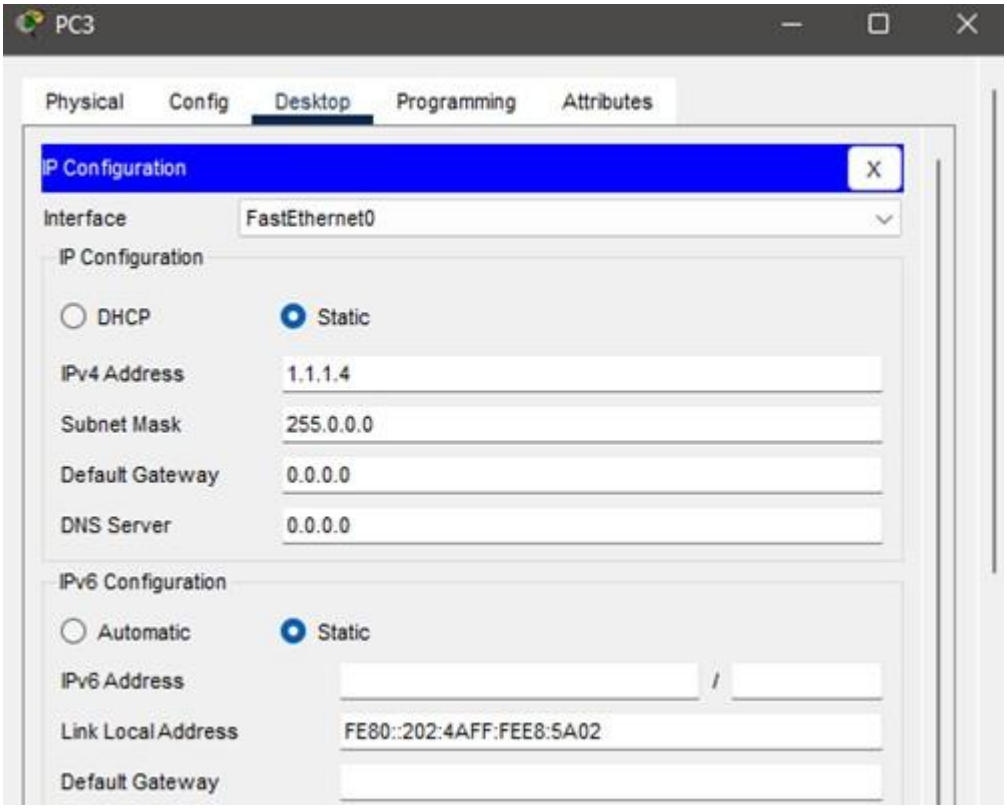
PC1 IP CONFIGURATION



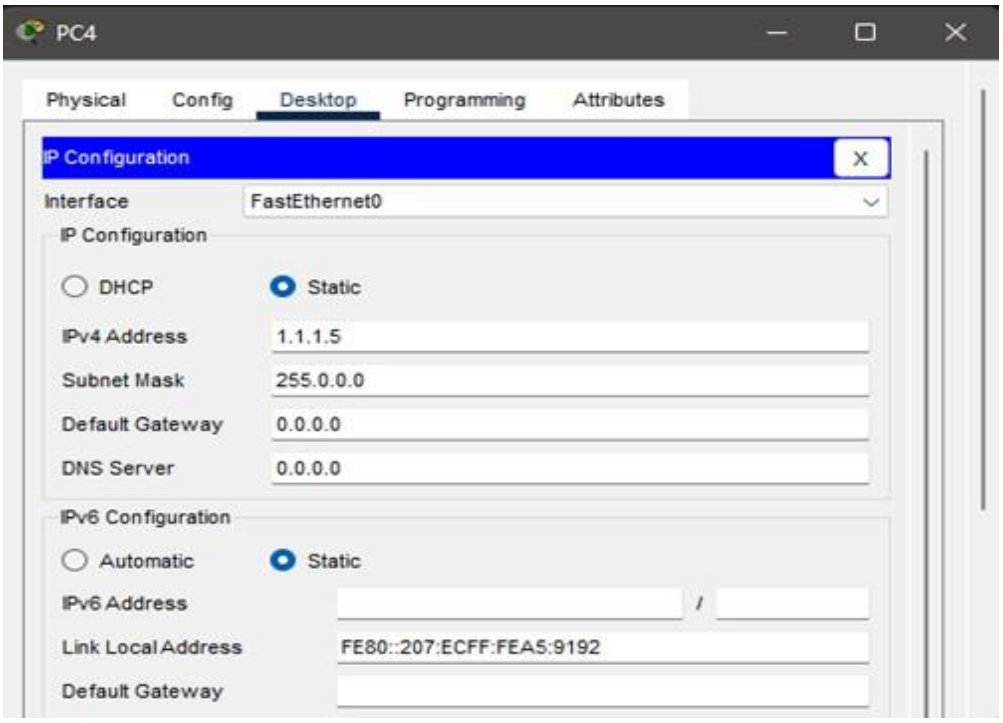
PC2 IP CONFIGURATION



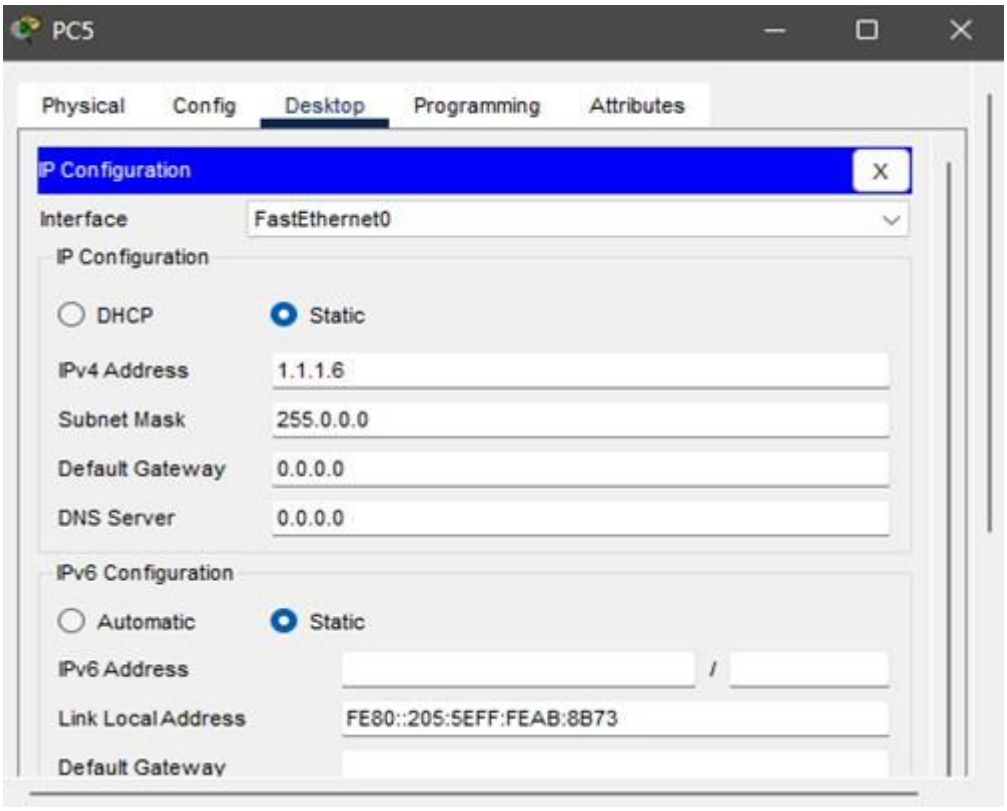
PC3 IP CONFIGURATION



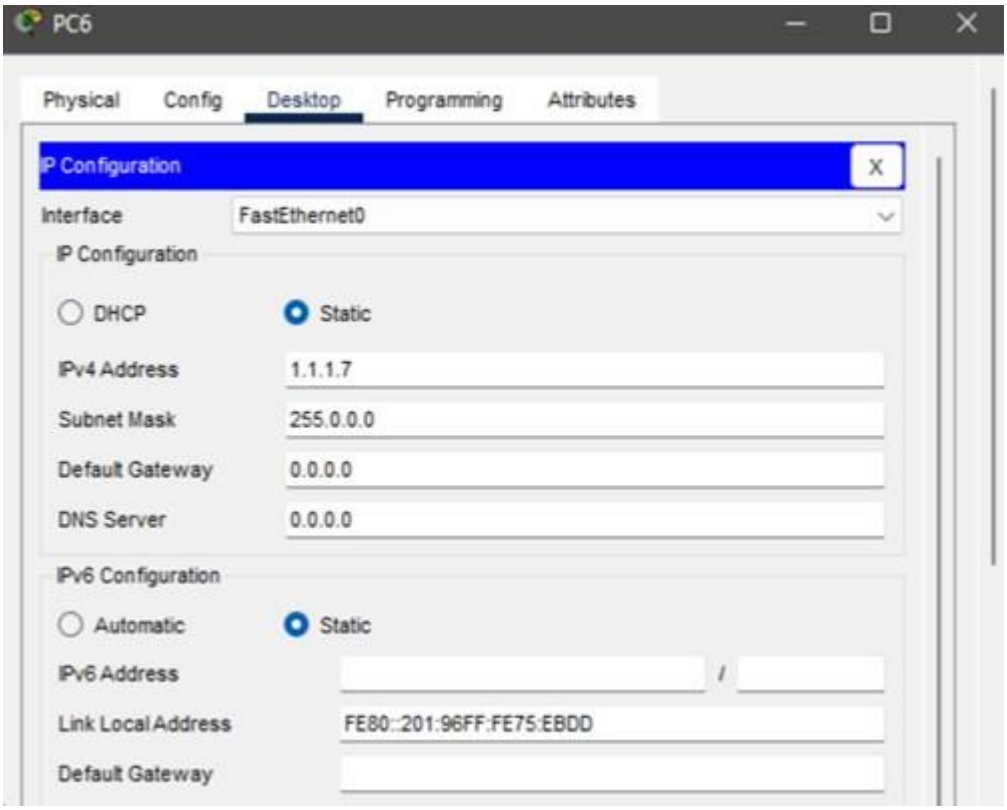
PC4 IP CONFIGURATION



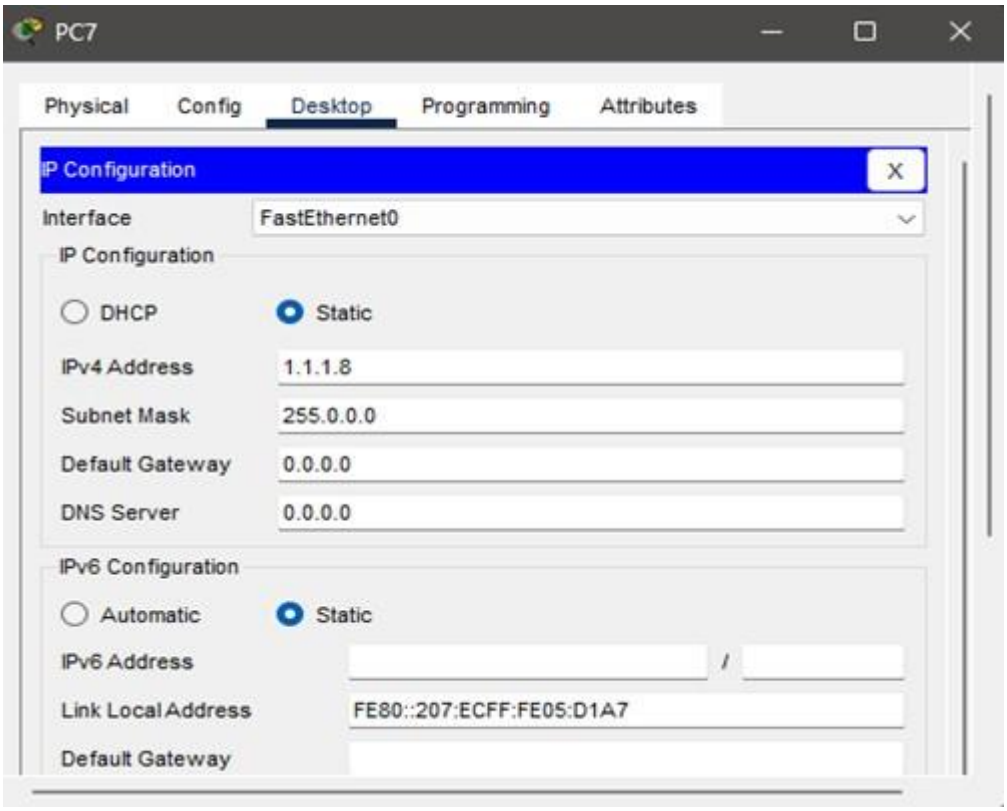
PC5 IP CONFIGURATION



PC6 IP CONFIGURATION



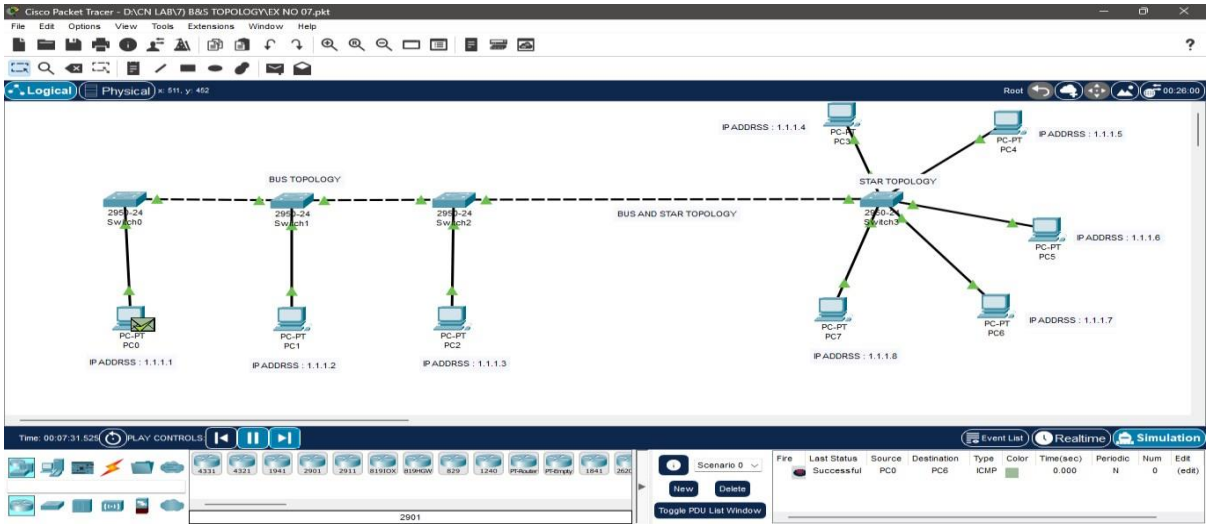
PC7 IP CONFIGURATION



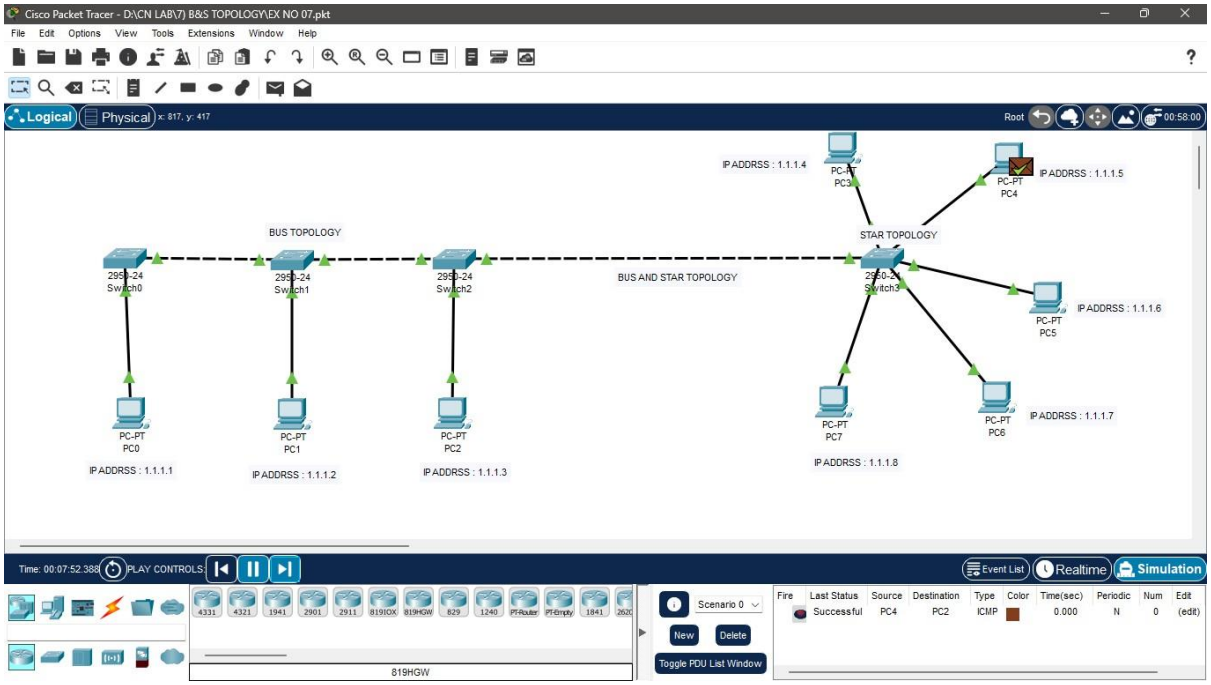
VERIFY LAN NETWORK CONNECTIVITY :

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 :

DESIGNING HYBRID TOPOLOGIES USING CISCO PACKET TRACER

AIM :

To Designing a Hybrid topologies by using Cisco Packet Tracer.

REQUIREMENTS :

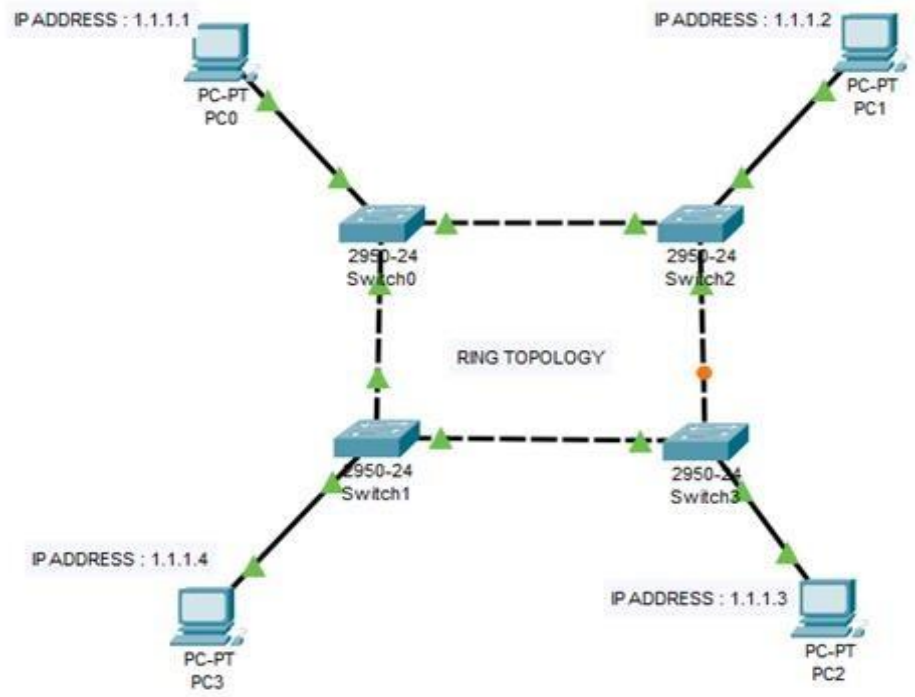
- 16 windows PC.
- 12 Switch (2950-24).
- 16 Straight Line LAN Cables.
- 13 Cross Over Cables.
- Cisco Packet Tracer.

PROCEDURES :

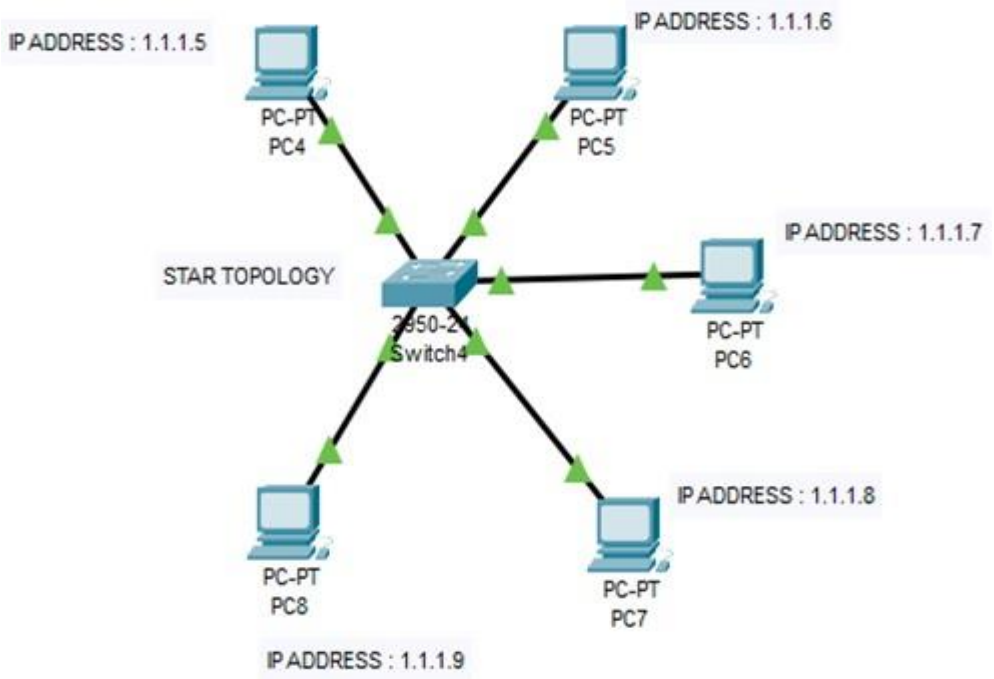
- Open CISCO PACKET TRACER software.
- Draw the 16 PC using END Device Icons.
- Draw the 12 Cisco 2950-24 Switch Using Switch icon lists.
- Make the Connections using Copper-Straight-Through Ethernet Cables.
- Make the Connections between Switches using Cross Overs Cables.
- Enter the IP Address To Each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

NETWORK TOPOLOGY

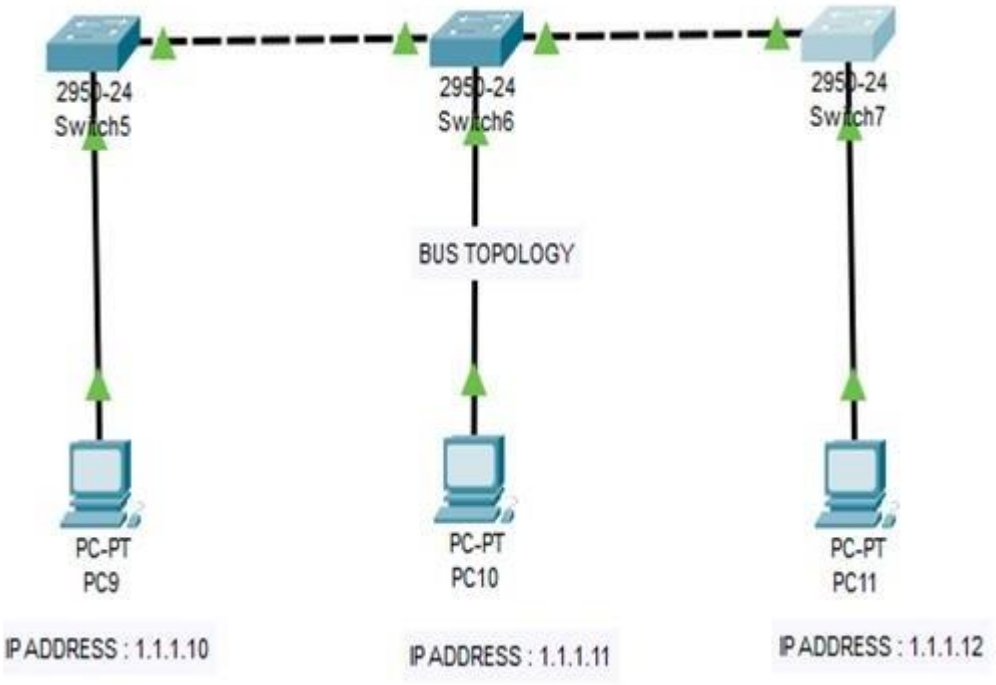
RING TOPOLOGY



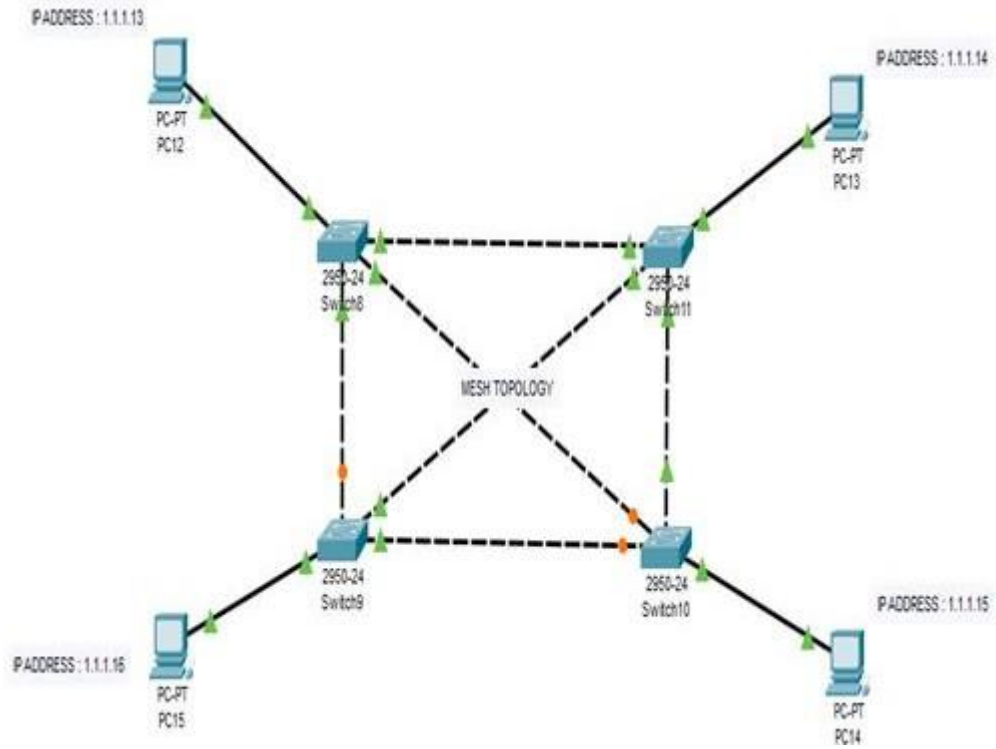
STAR TOPOLOGY



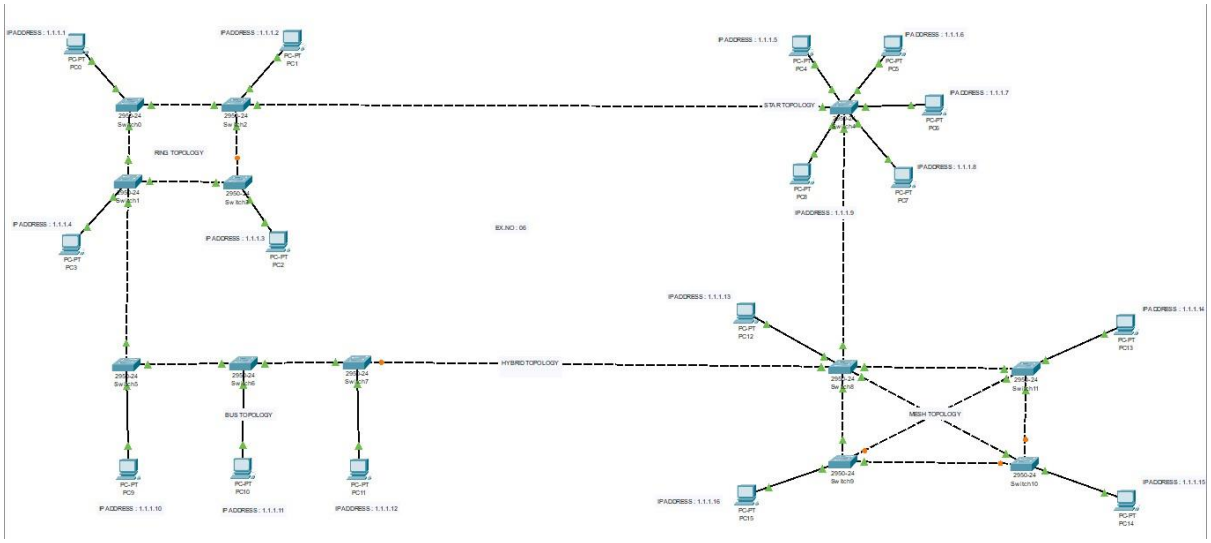
BUS TOPOLOGY



MESH TOPOLOGY

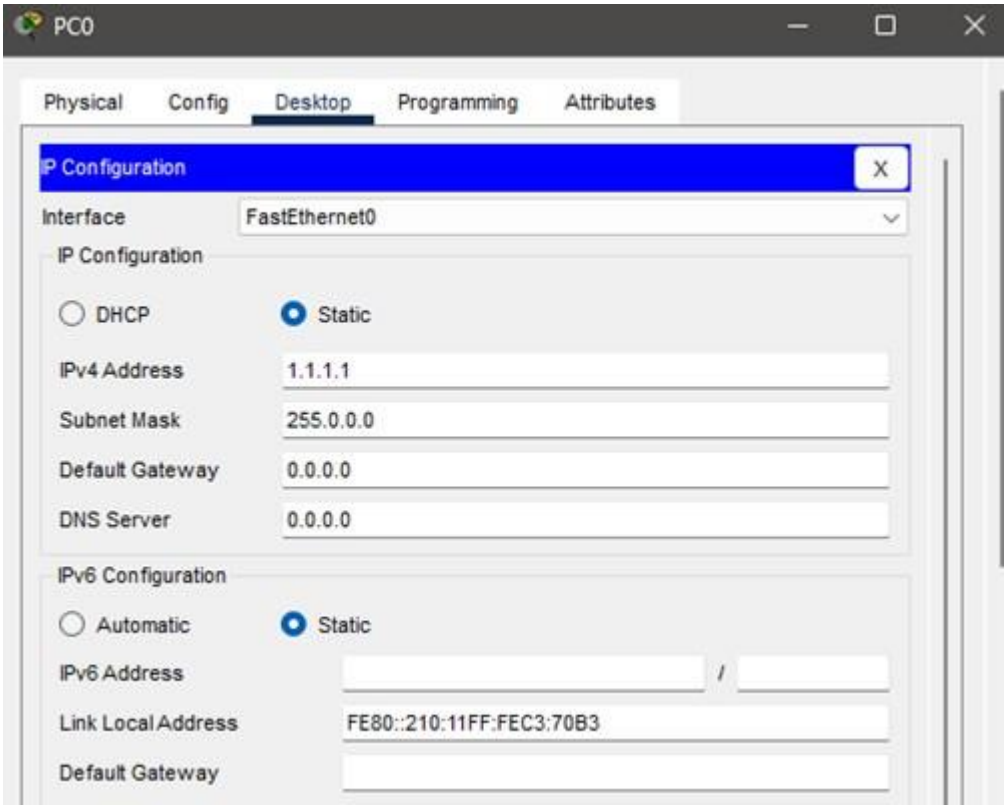


HYBRID TOPOLOGY

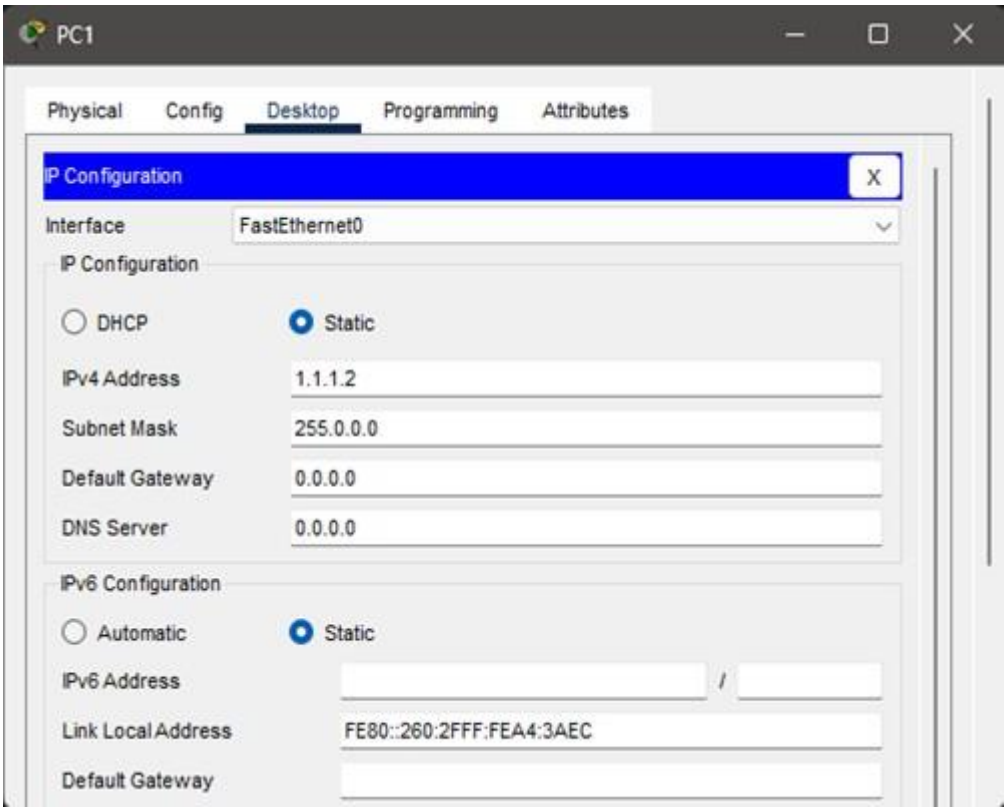


HOST PC IP ADDRESS :

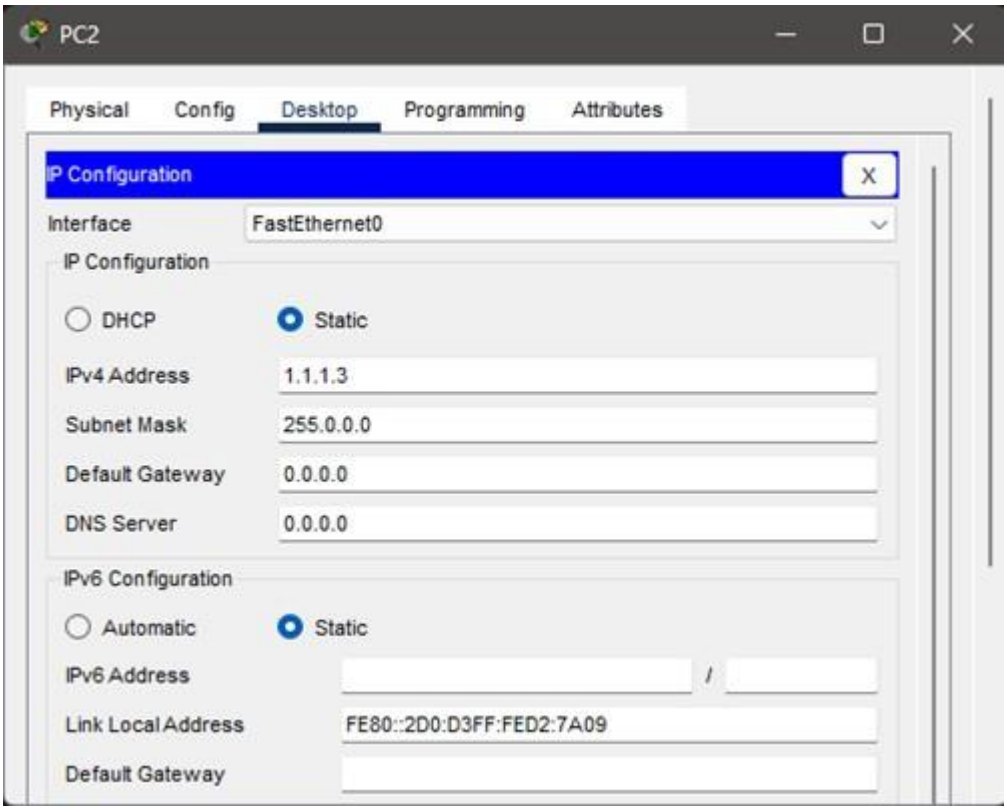
PC0 IP CONFIGURATION



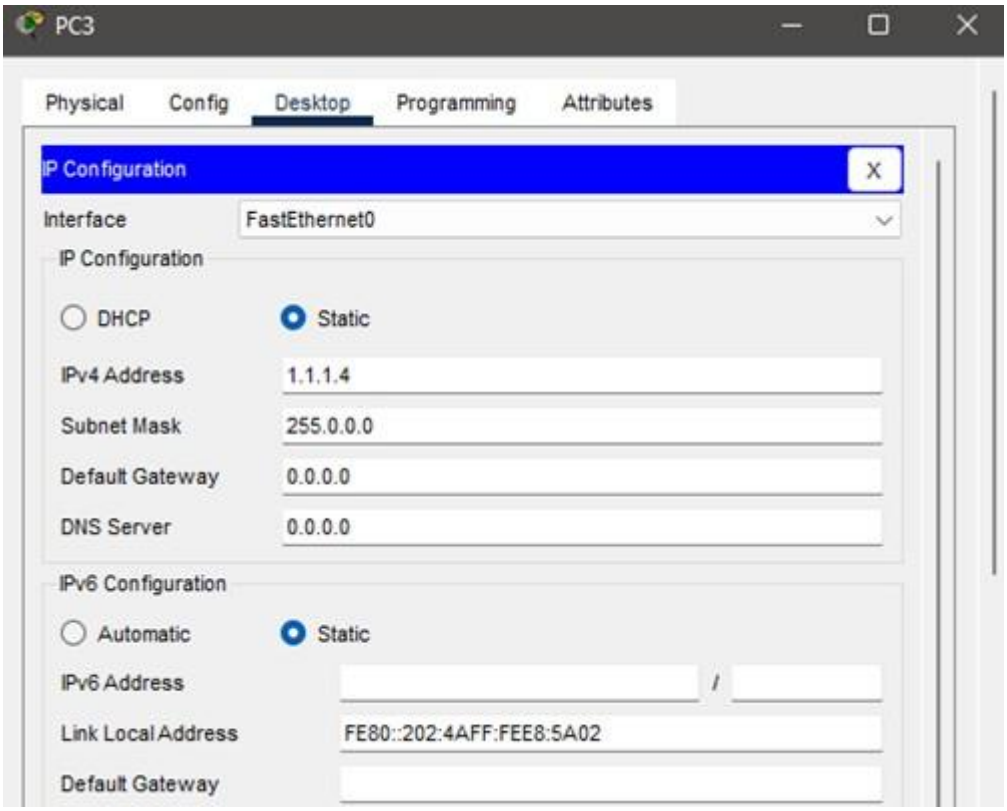
PC1 IP CONFIGURATION



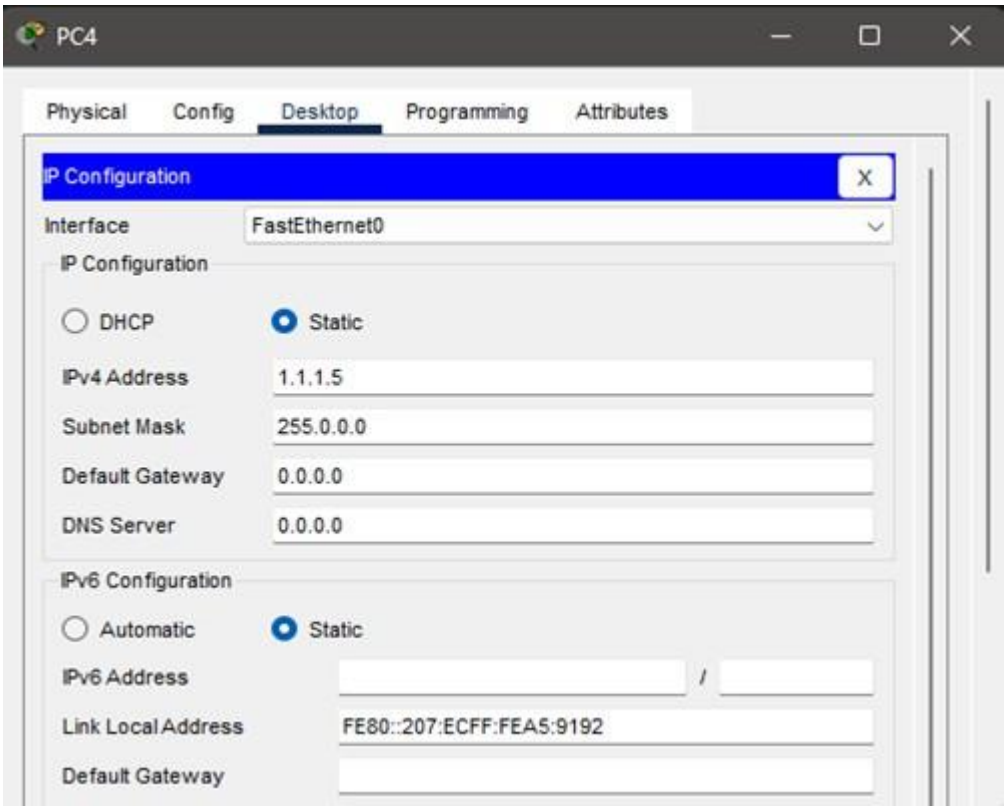
PC2 IP CONFIGURATION



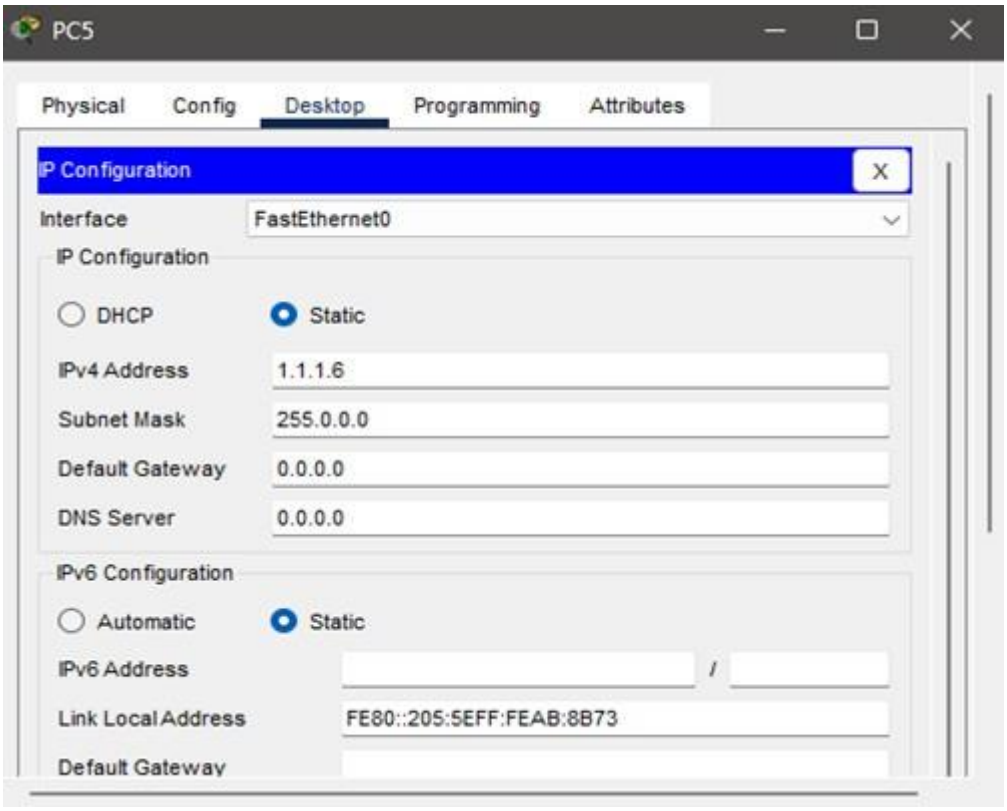
PC3 IP CONFIGURATION



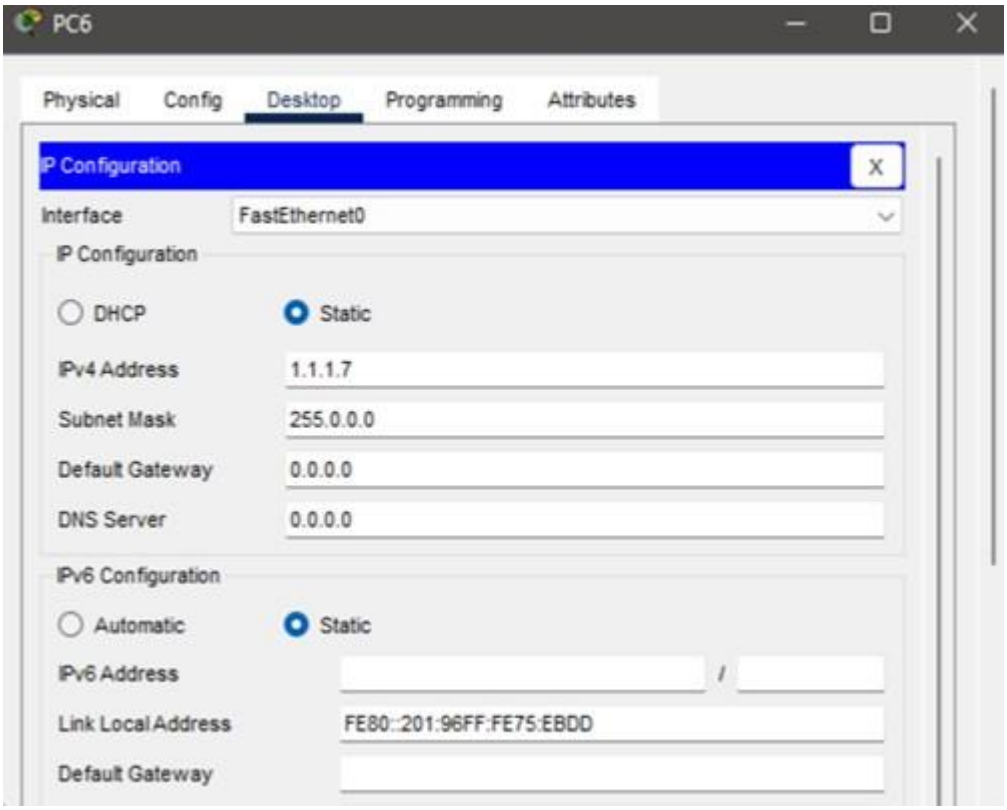
PC4 IP CONFIGURATION



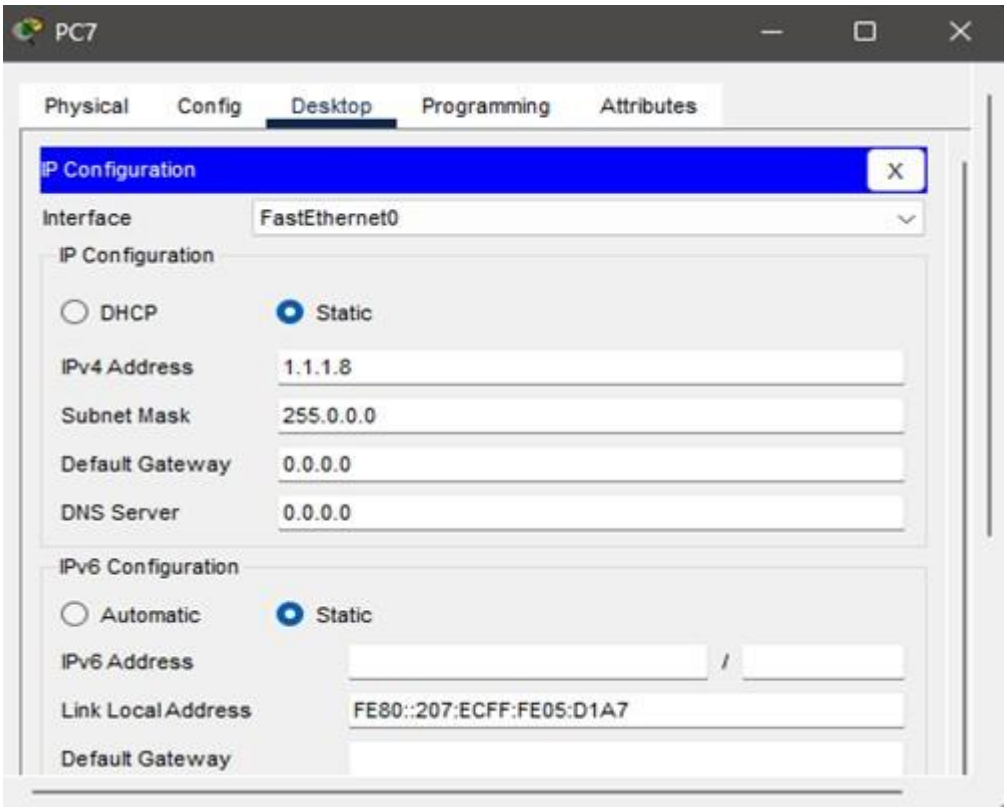
PC5 IP CONFIGURATION



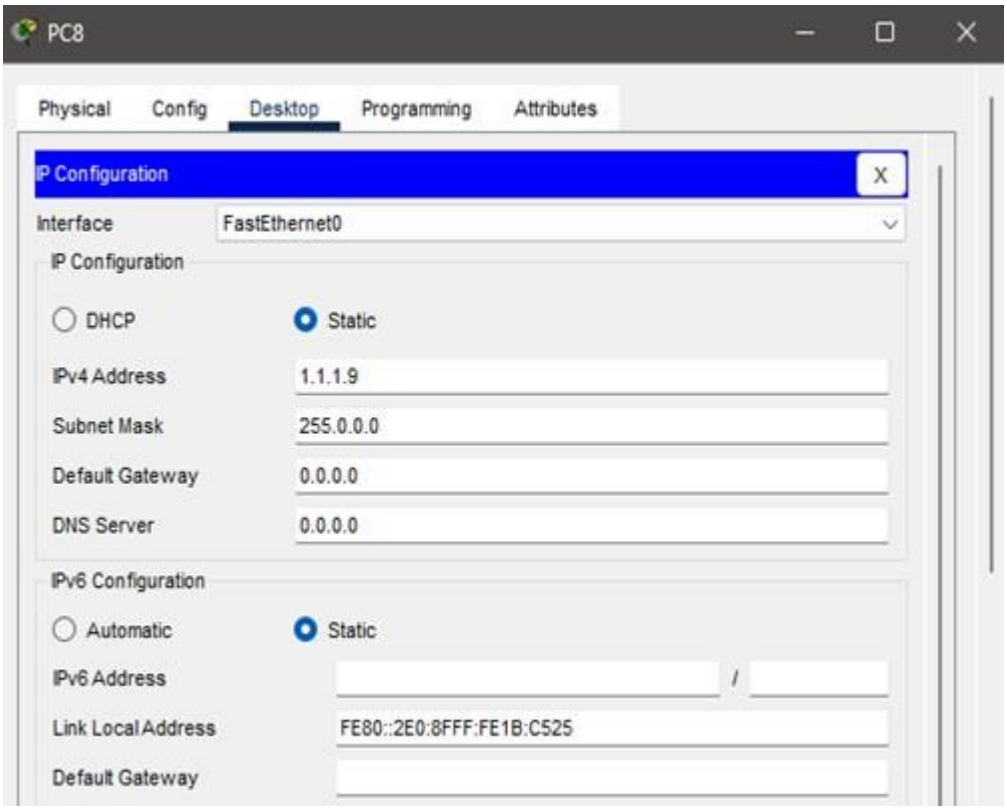
PC6 IP CONFIGURATION



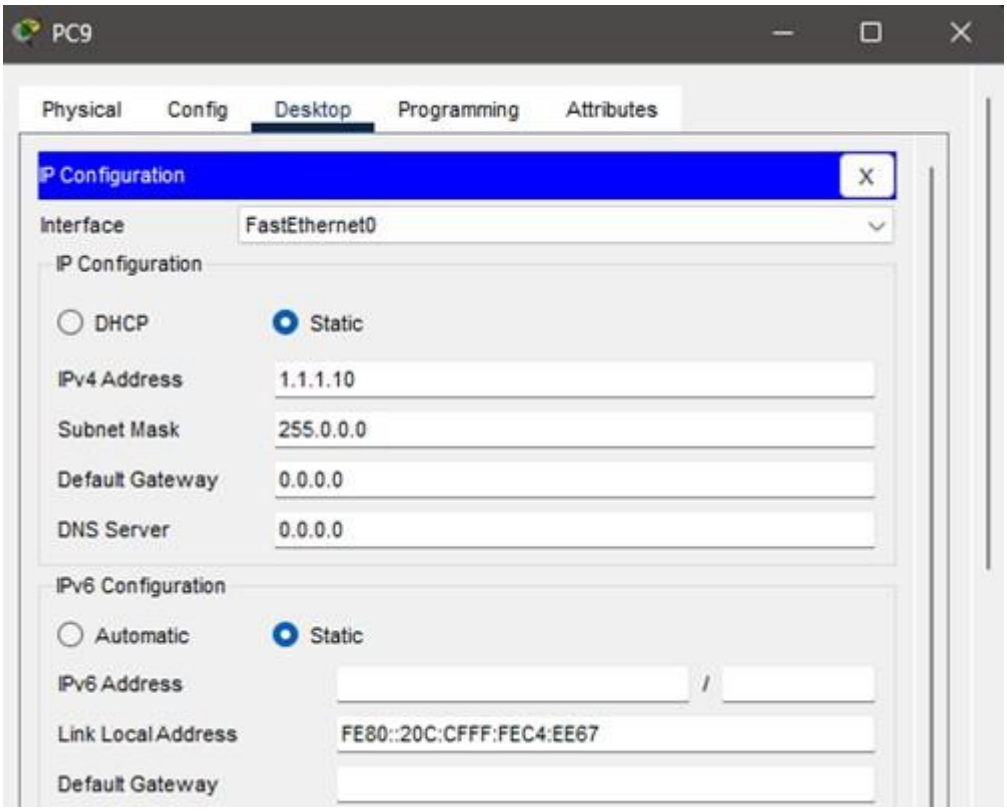
PC7 IP CONFIGURATION



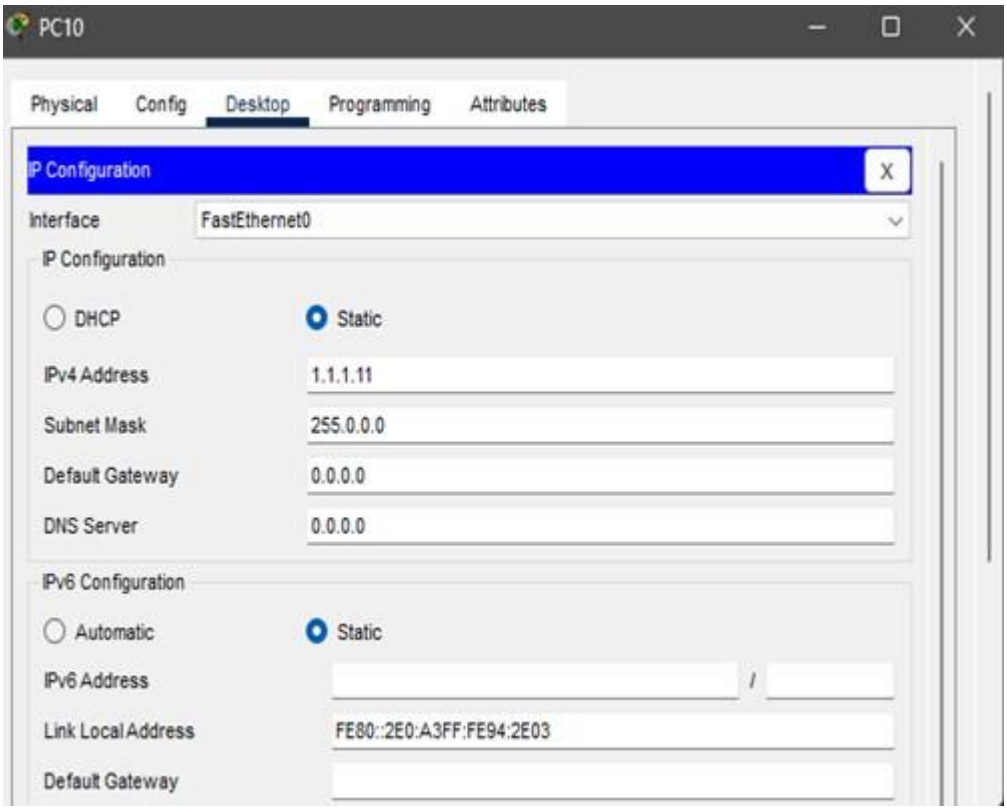
PC8 IP CONFIGURATION



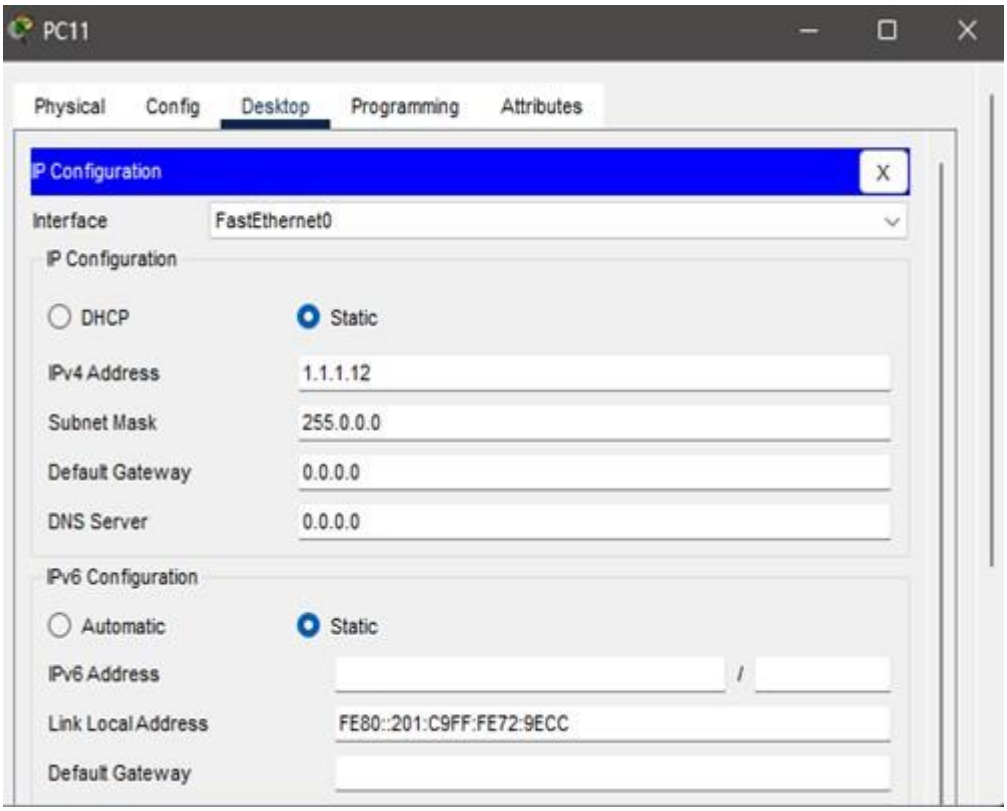
PC9 IP CONFIGURATION



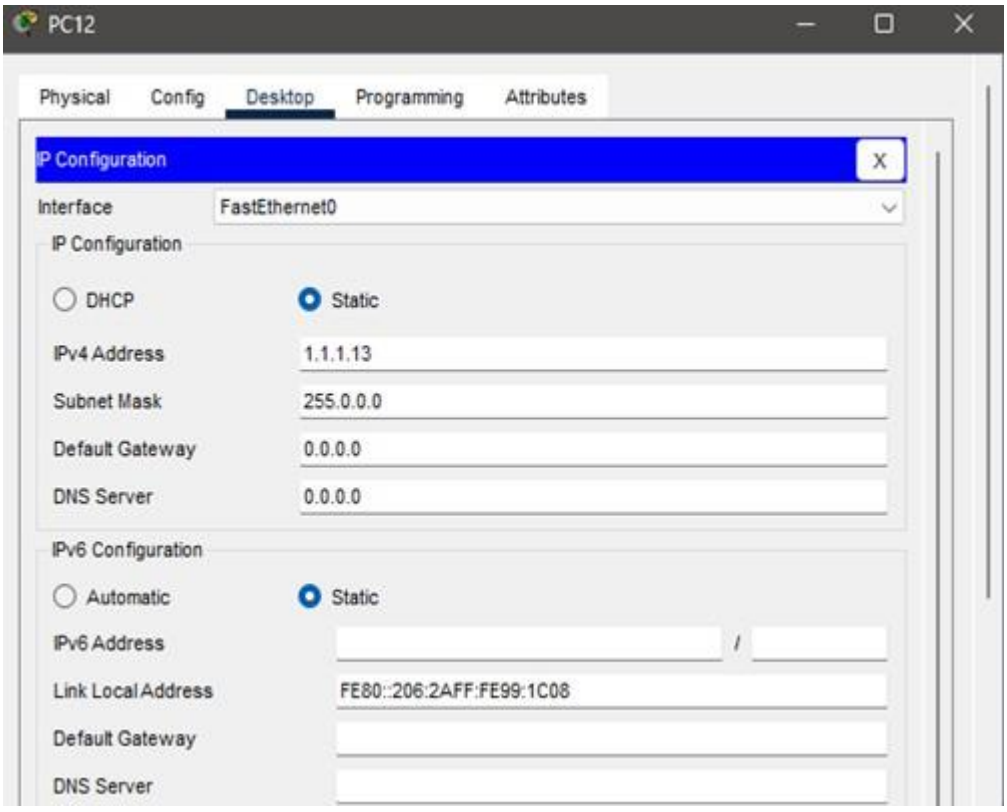
PC10 IP CONFIGURATION



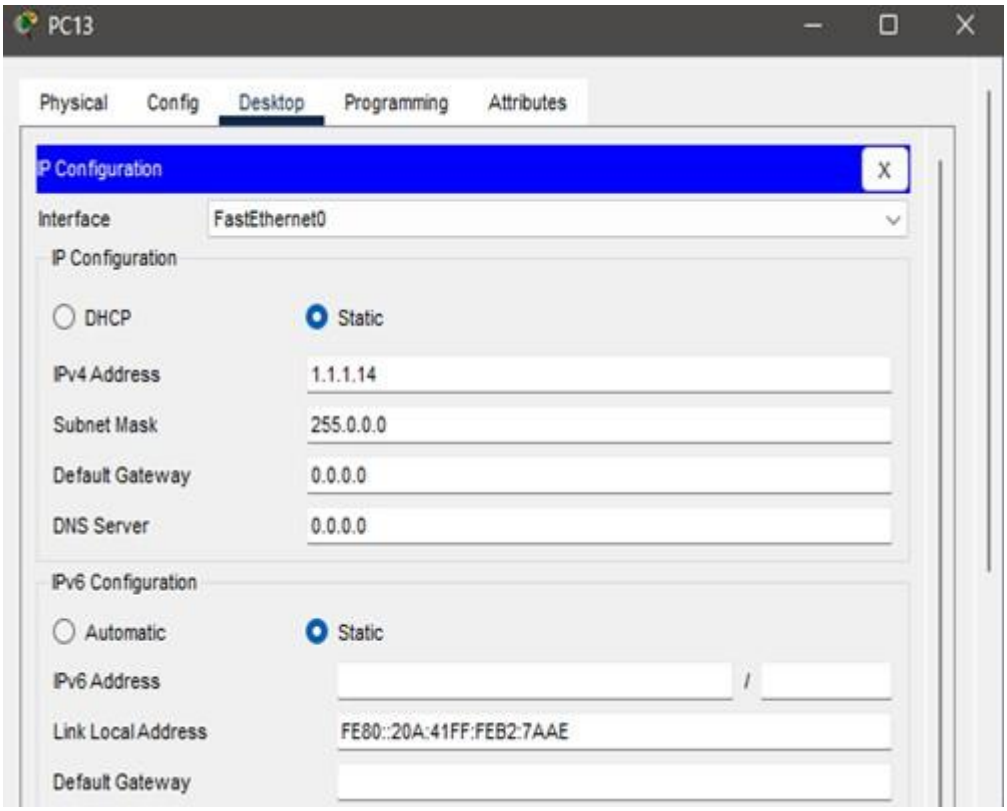
PC11 IP CONFIGURATION



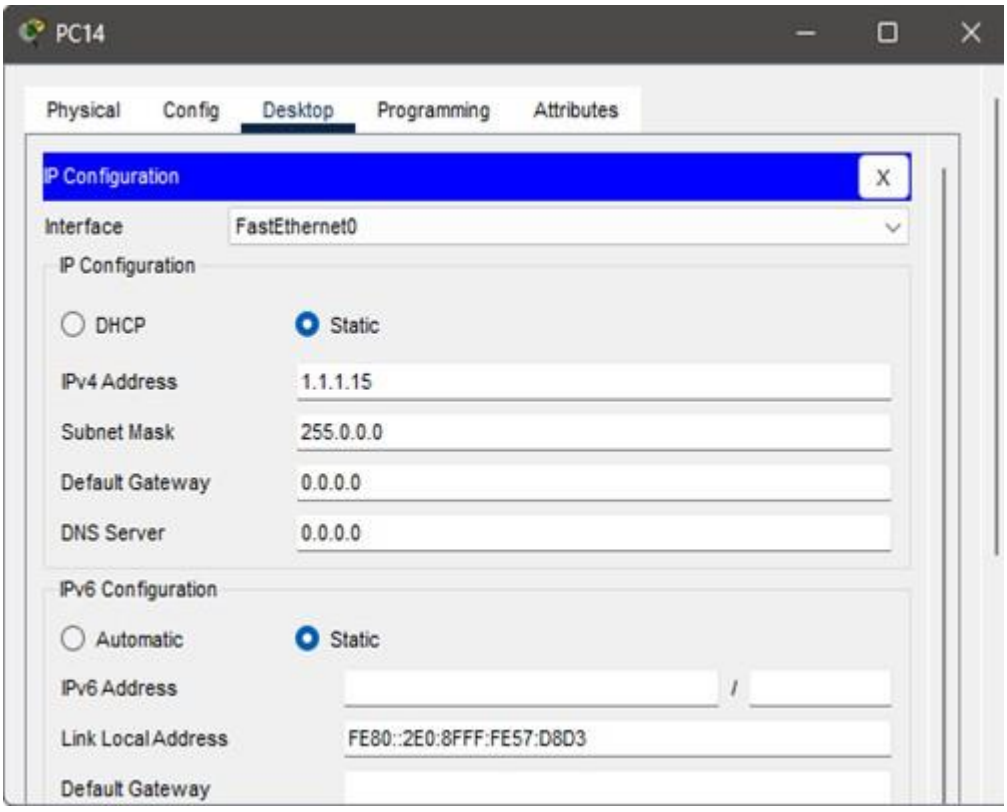
PC12 IP CONFIGURATION



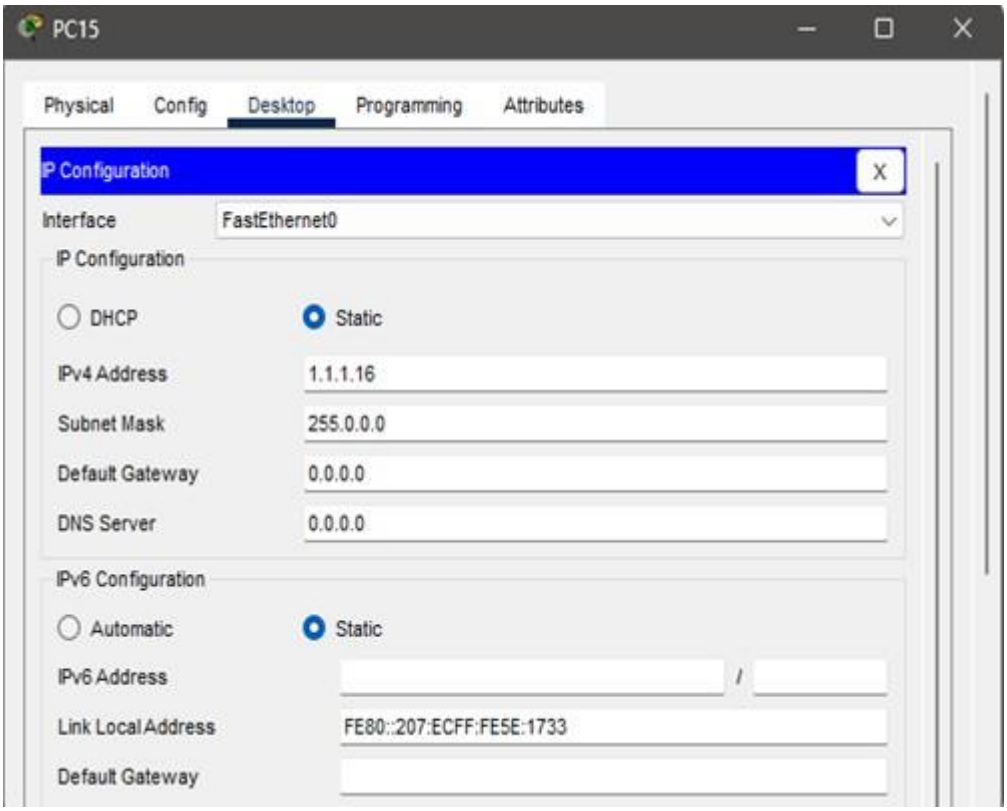
PC13 IP CONFIGURATION



PC14 IP CONFIGURATION



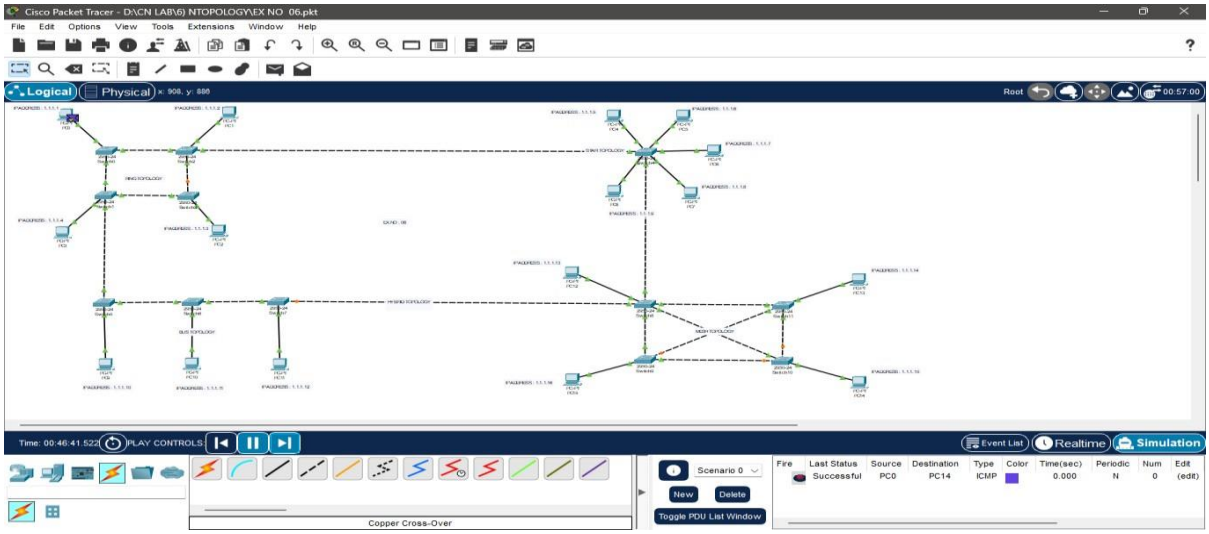
PC15 IP CONFIGURATION



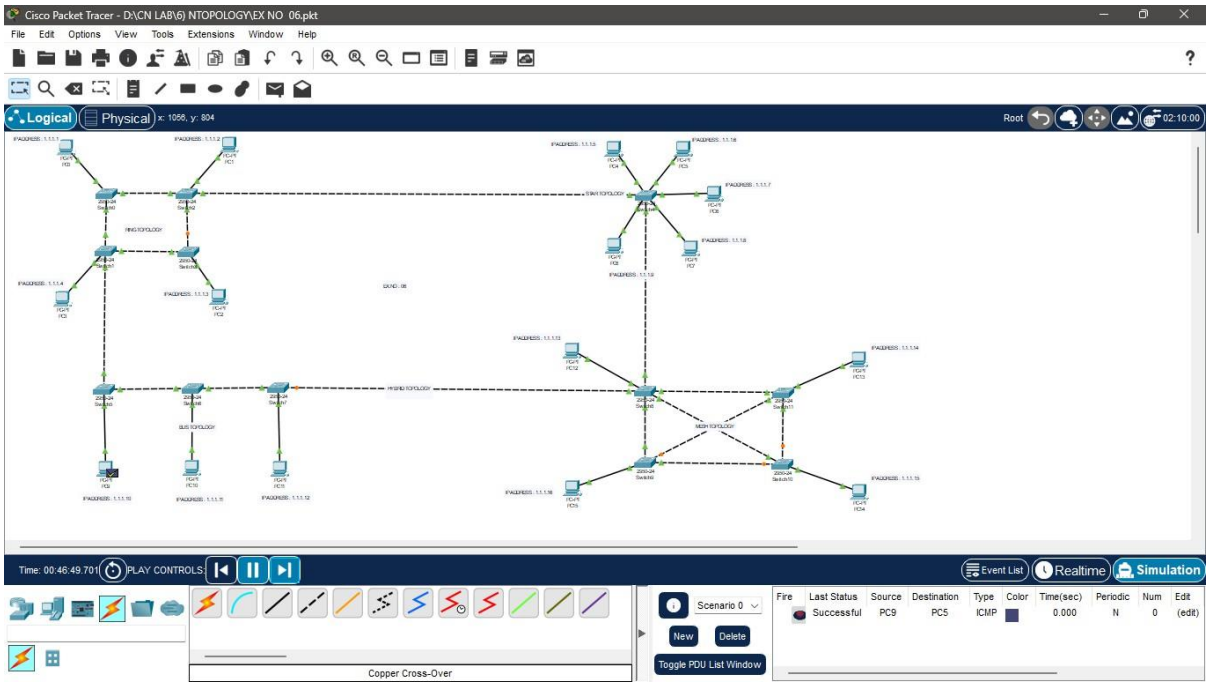
VERIFY LAN NETWORK CONNECTIVITY :

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.

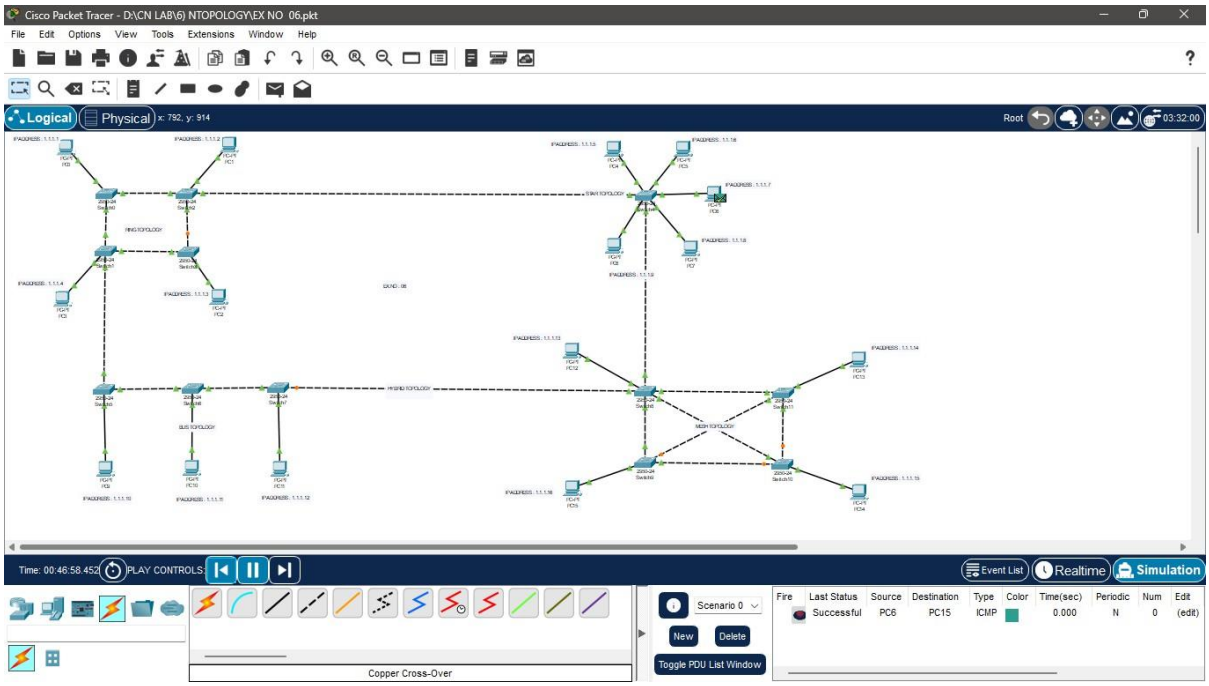
HOST PC0 TO PC14



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 :

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.

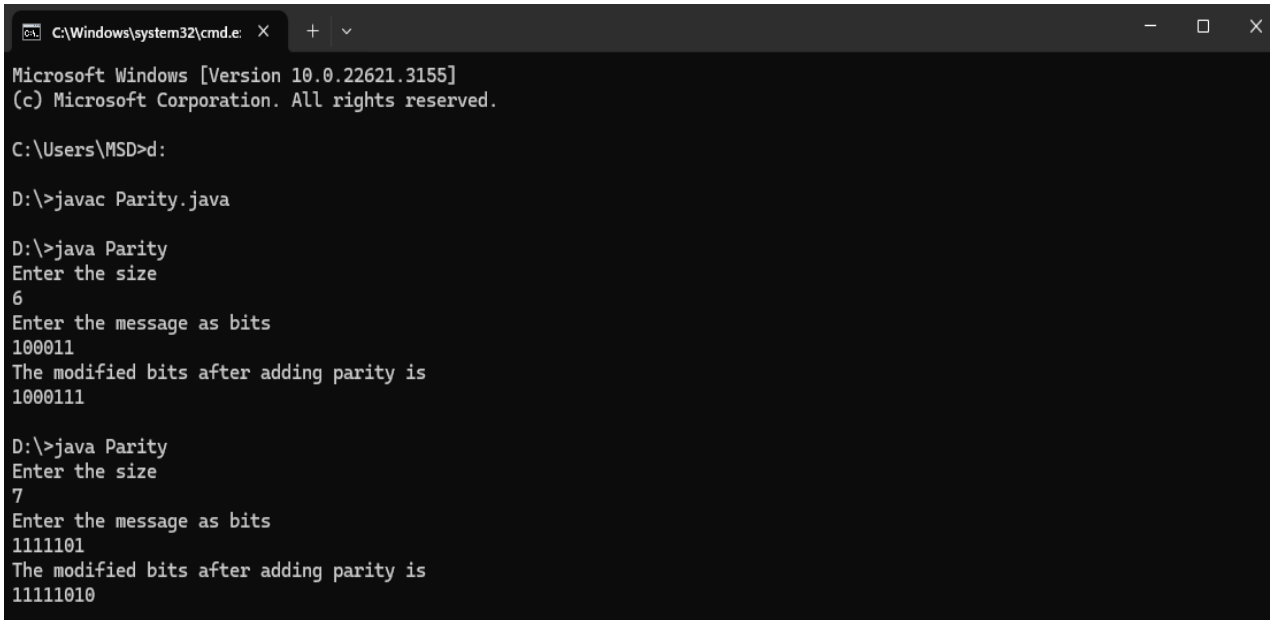
6) Stop the program.

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 :



RESULT :

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

EX.NO : 10

DATE :

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 :

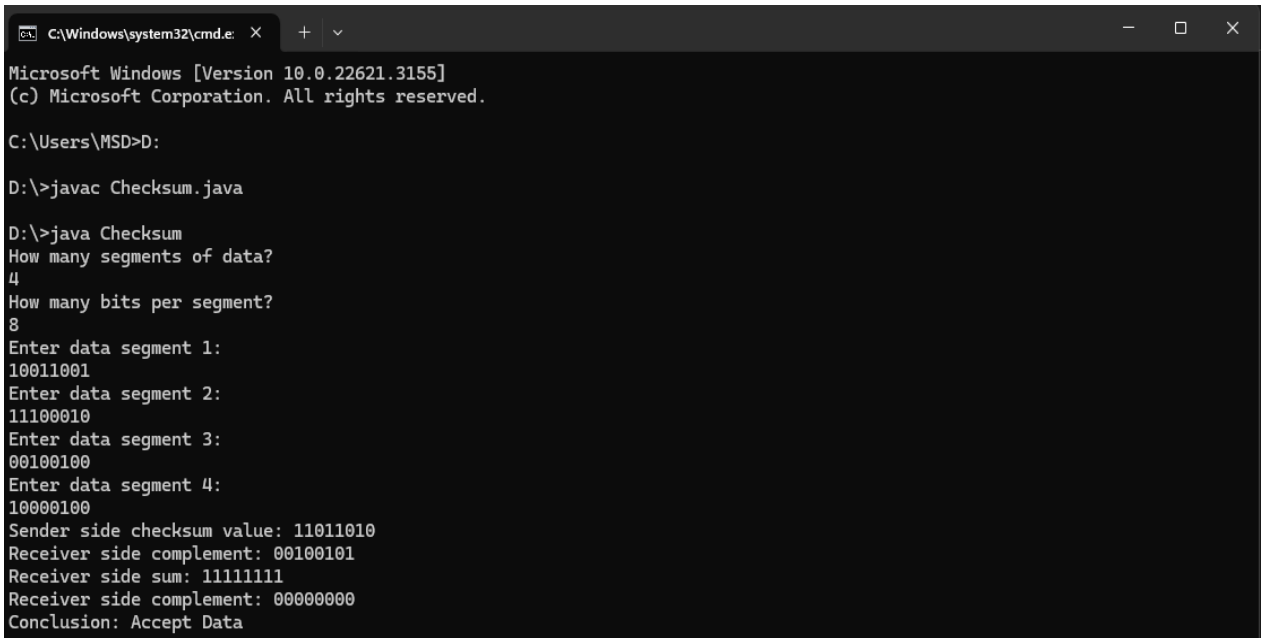
```
import java.util.Scanner;
public class Checksum
{
    static String complement(String sum, int m)
    {
    char bits[] = sum.toCharArray();
    for (int i = 0; i < m; i++)
        {
            if (bits[i] == '1')
```

```
                {
                bits[i] = '0';
                }
                else
                {
                bits[i] = '1';
                }
}
return new String(bits);
}
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 :



RESULT :

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

EX.NO : 11

DATE :

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.

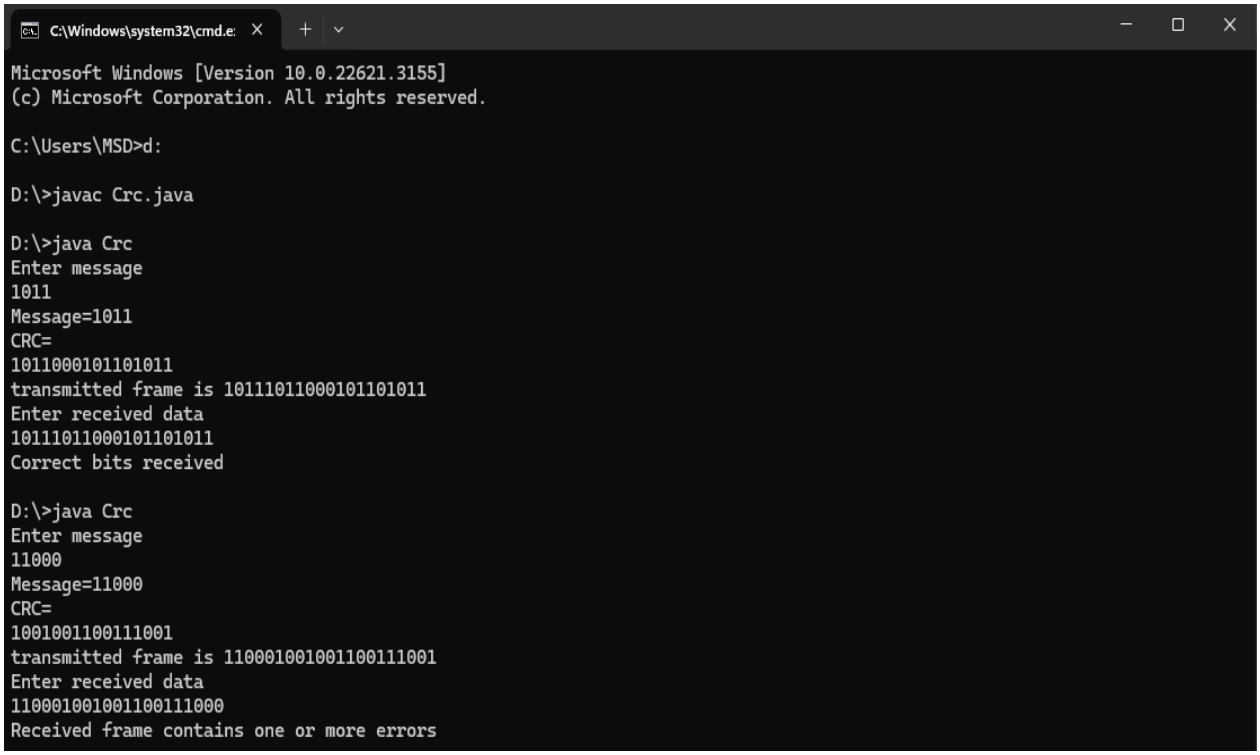
PROGRAM :

```
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="10001000000100001";
        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 :



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 :

IMPLEMENTATION OF GO BACK N PROTOCOL

AIM :

To write a Java program to perform a GoBackN Protocol.

ALGORITHM :

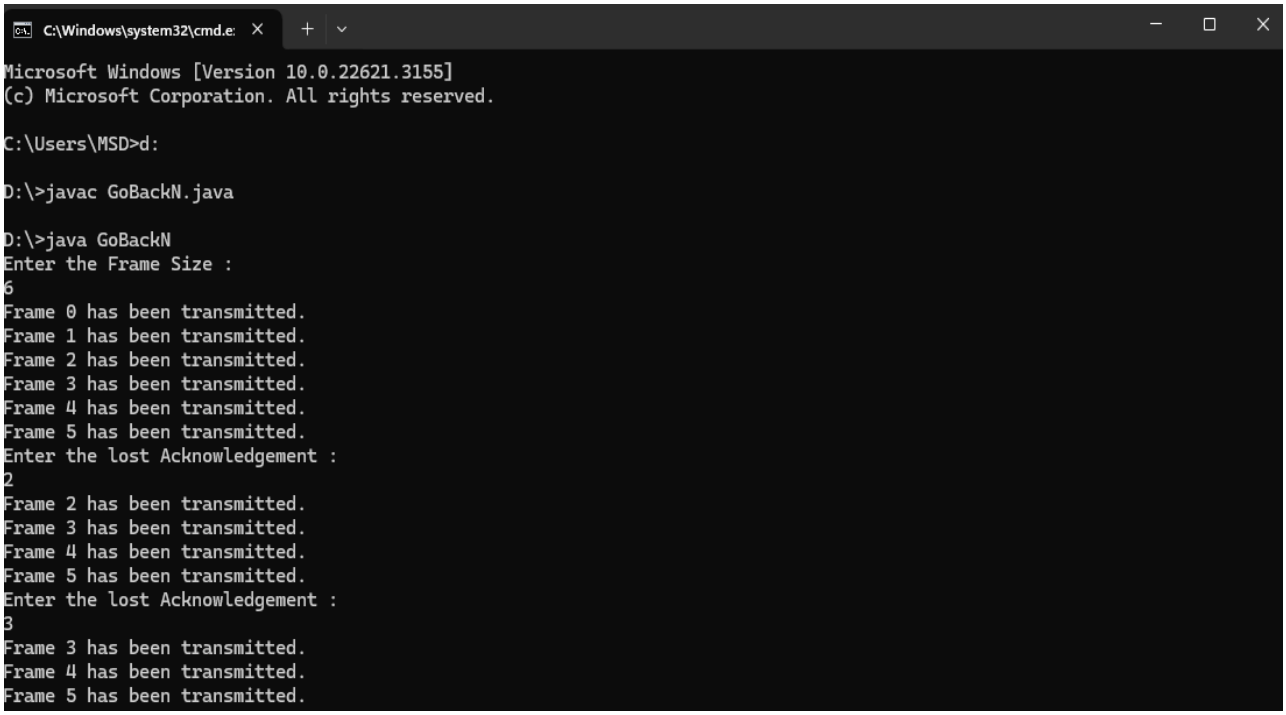
1) Start the program.
2) Prompt for the frame size.
3) Initialize sent to 0.
4) Transmit frames up to the frame size, printing each transmission.
5) Prompt for lost Acknowledgement.
6) If lost Acknowledgement equals frame size, stop; else, update sent and repeat transmission.
7) Stop the program.

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 :



RESULT :

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

EX.NO : 13

DATE :

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 :

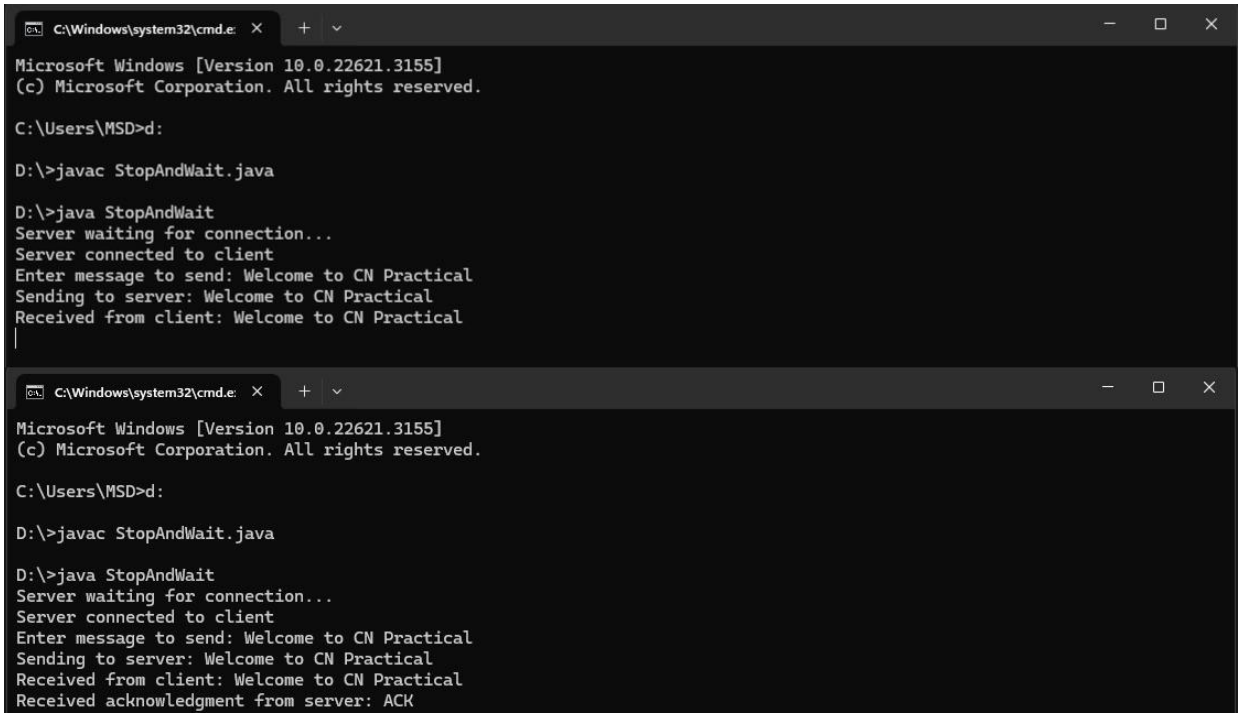
```
import java.io.*;
import java.net.*;
public class StopAndWait {
   private static final int PORT = 123;
   public static void main(String[] args) {
      new Thread(() -> {
         try {
            runServer();
```

```
            } 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 :



RESULT :

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

EX.NO : 14

DATE :

IMPLEMENTATION OF SELECTIVE REPEAT PROTOCOL

AIM :

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

ALGORITHM :

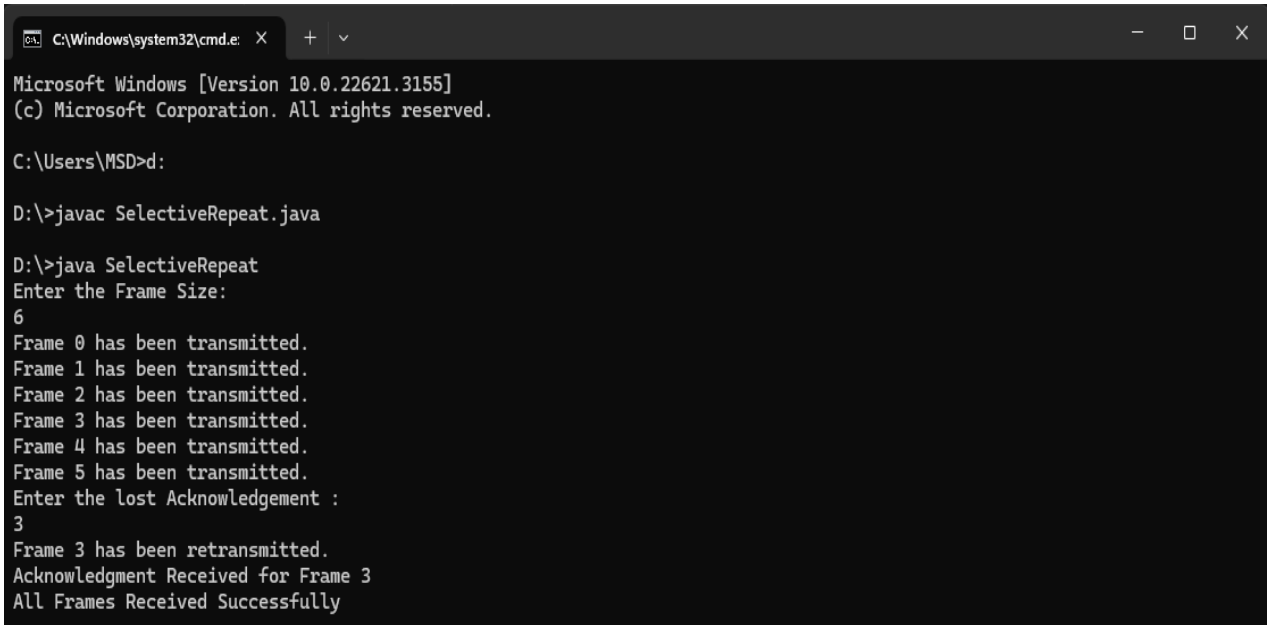
1. Start program, prompt frame size.
2. Transmit frames, prompt lost Acknowledgement.
3. Print acknowledgment or retransmission message based on lost Acknowledgement.
4. Repeat until lost Acknowledgement equals frame size.
5. Stop program.

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 :



RESULT :

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

EX.NO : 15

DATE :

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 :

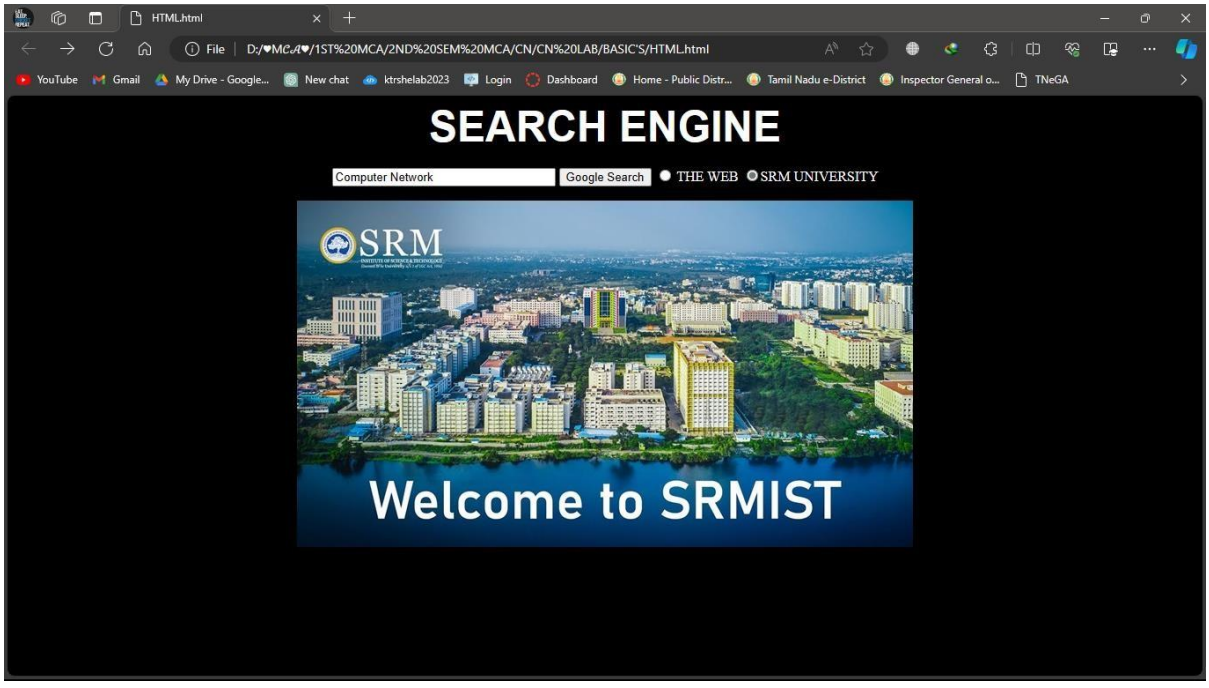
- ✓ Double click the file and open it using any available browser.

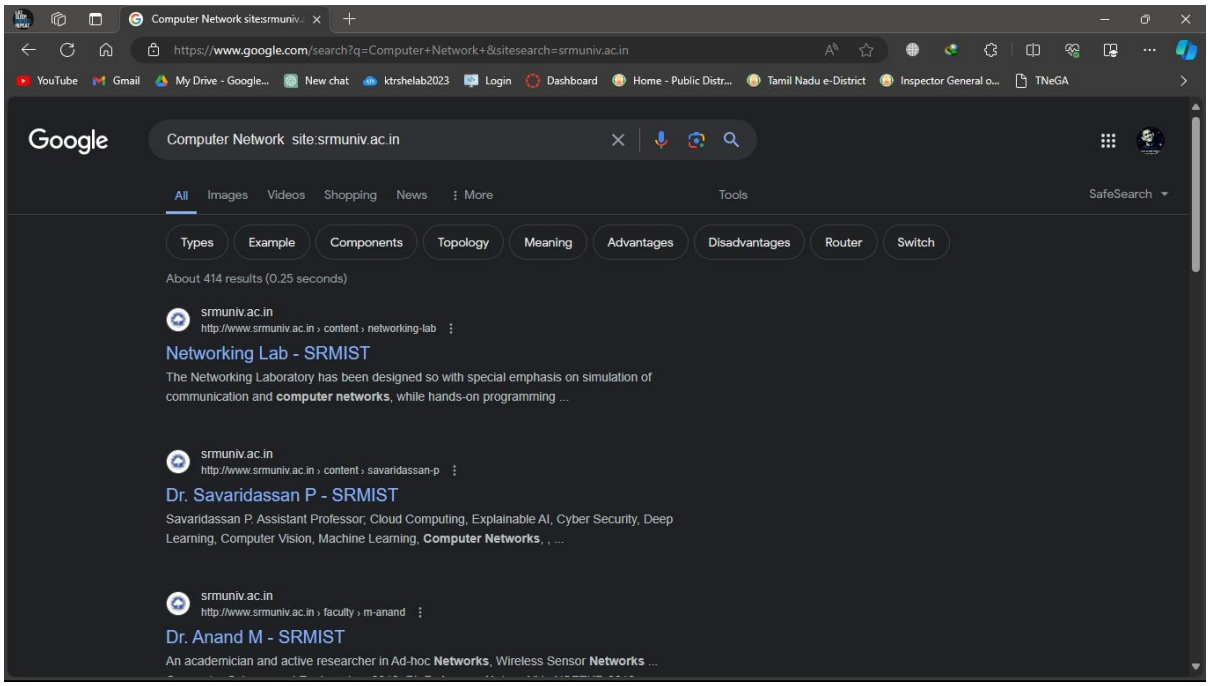
PROGRAM :

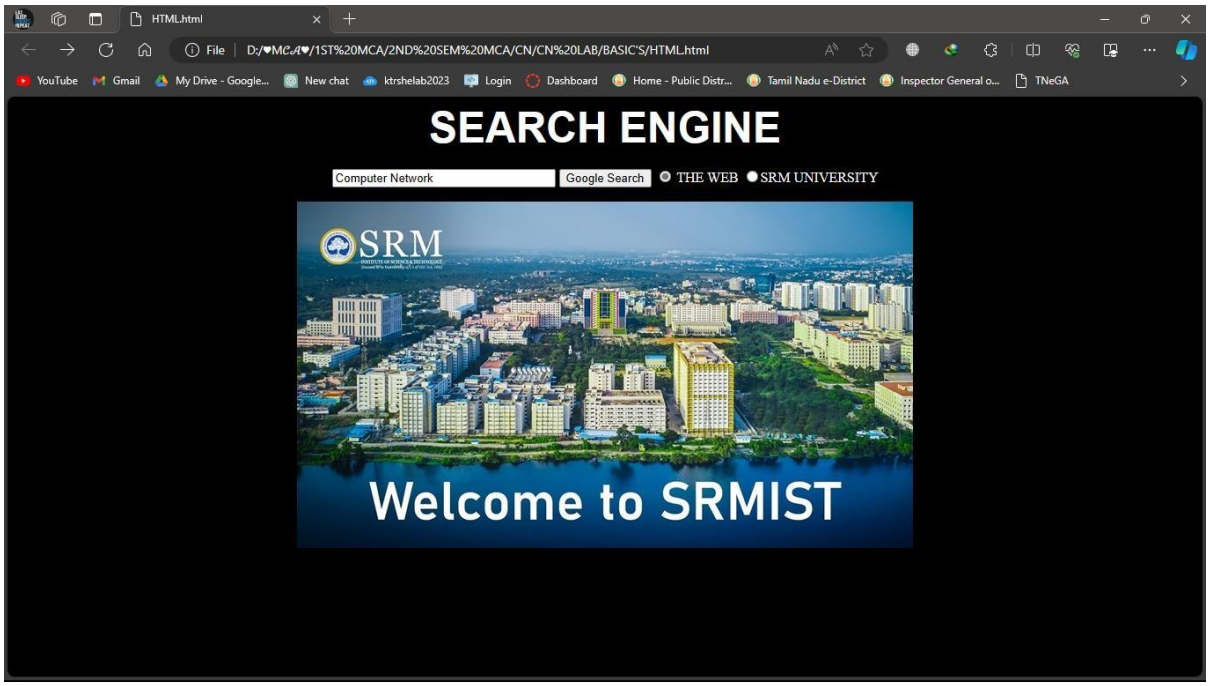
```
<html>
<body bgcolor="black">
<center><font size="36" color = "white" face="Arial"><b>SEARCH
ENGINE</b></font><br>
<br>
<form method="get" action ="http://www.google.com/search">
```

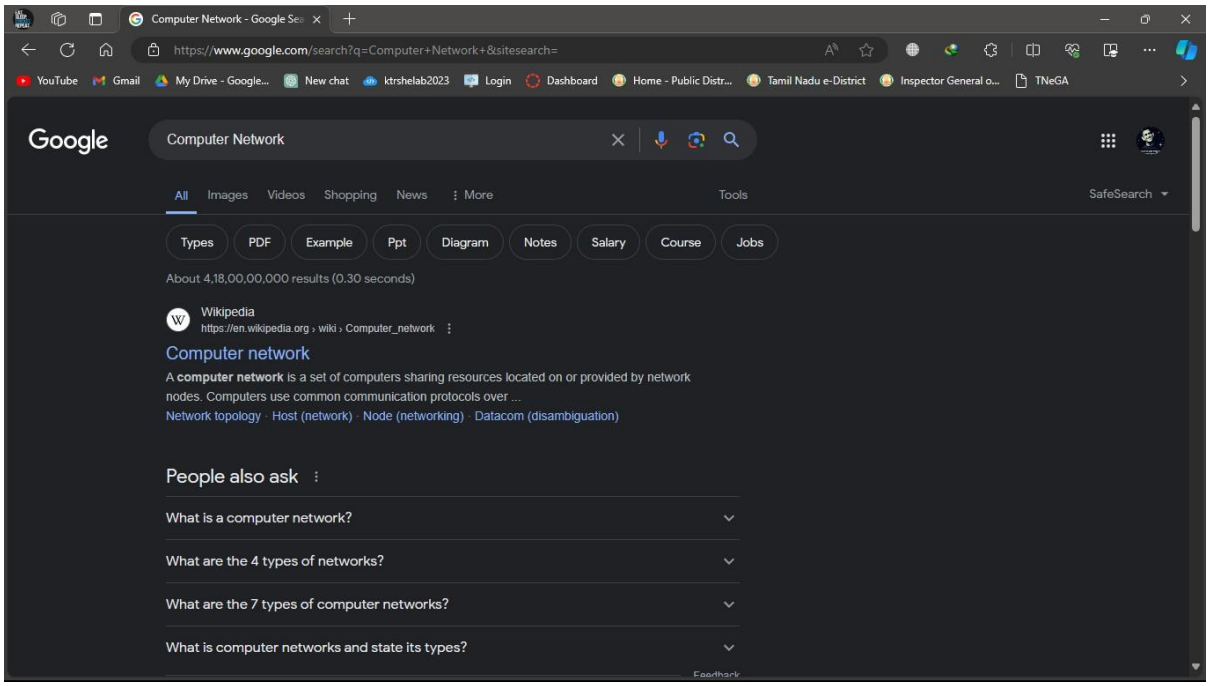
```
<input type="text" name="q" size="31" maxlength="255" value=""/>

<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><img src="file:///D:/IDM/srm.jpg"></center><center>

</body>

</html>
```

OUTPUT :




Computer Network site:srmuniv...
https://www.google.com/search?q=Computer+Network+&sitesearch=srmuniv.ac.in
YouTube
Gmail
My Drive - Google...
New chat
ktrshelab2023
Login
Dashboard
Home - Public Distr...
Tamil Nadu e-District
Inspector General o...
TNeGA
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)
srmuniv.ac.in
http://www.srmuniv.ac.in › content › networking-lab
Networking Lab - SRMIST
The Networking Laboratory has been designed so with special emphasis on simulation of communication and computer networks, while hands-on programming ...
srmuniv.ac.in
http://www.srmuniv.ac.in › content › savaridassan-p
Dr. Savaridassan P - SRMIST
Savaridassan P, Assistant Professor; Cloud Computing, Explainable AI, Cyber Security, Deep Learning, Computer Vision, Machine Learning, Computer Networks, , ...
srmuniv.ac.in
http://www.srmuniv.ac.in › faculty › m-anand
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...
TNeGA
SEARCH ENGINE
Computer Network
Google Search
THE WEB
SRM UNIVERSITY
SRM
Welcome to SRMIST



RESULT :

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