**EX.NO: 1**

**DATE: 31/01/2022**

# FAMILIARIZING WITH WINDOWS NETWORK COMMANDS

## AIM:

To familiarize with windows network commands and their outputs.

## PROCEDURE:

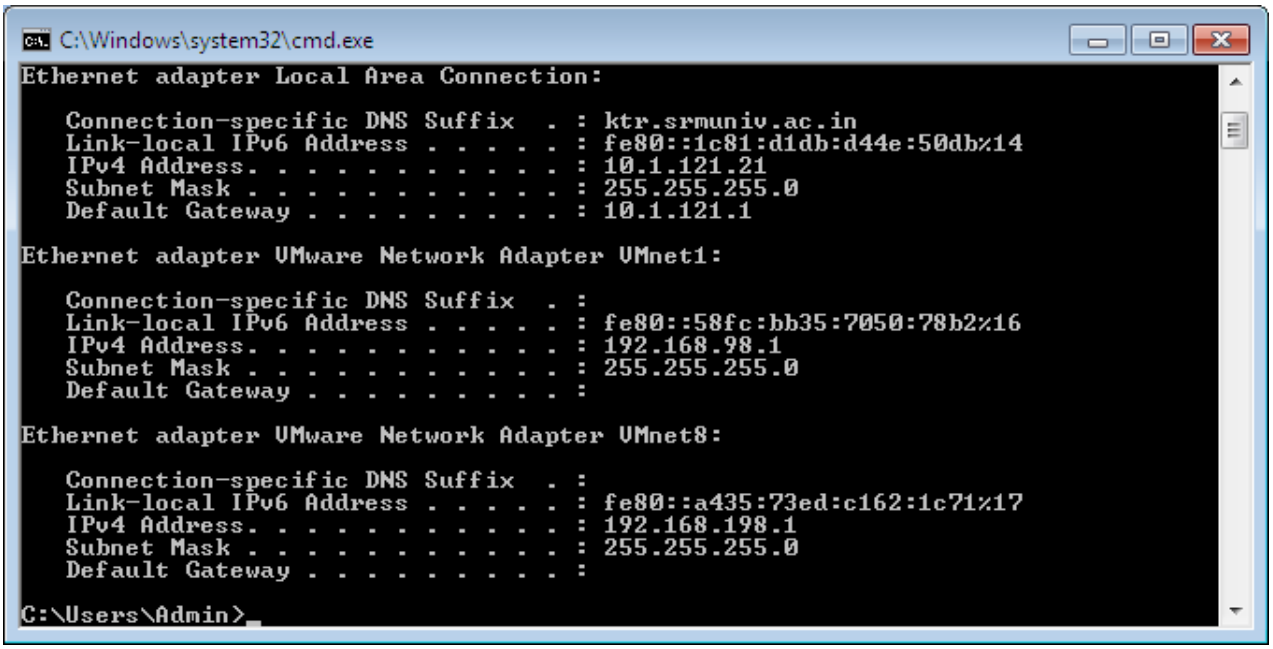
1. Open the Command prompt by typing “CMD” in the Run Dialogue
2. Once the Command prompt opens type the commands

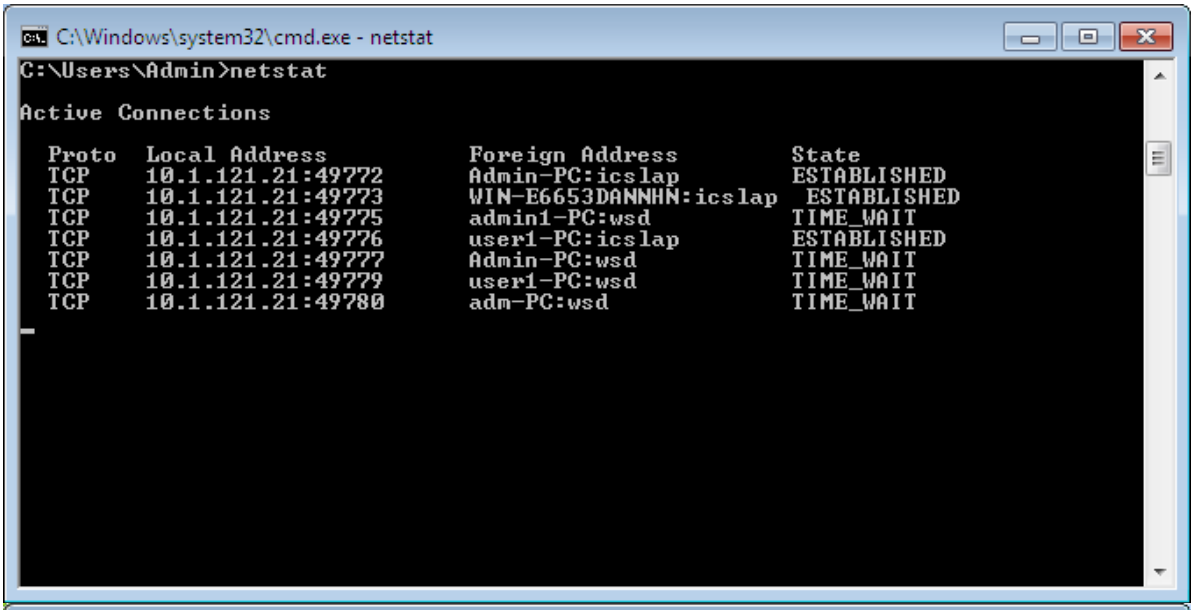
## COMMAND DESCRIPTION:

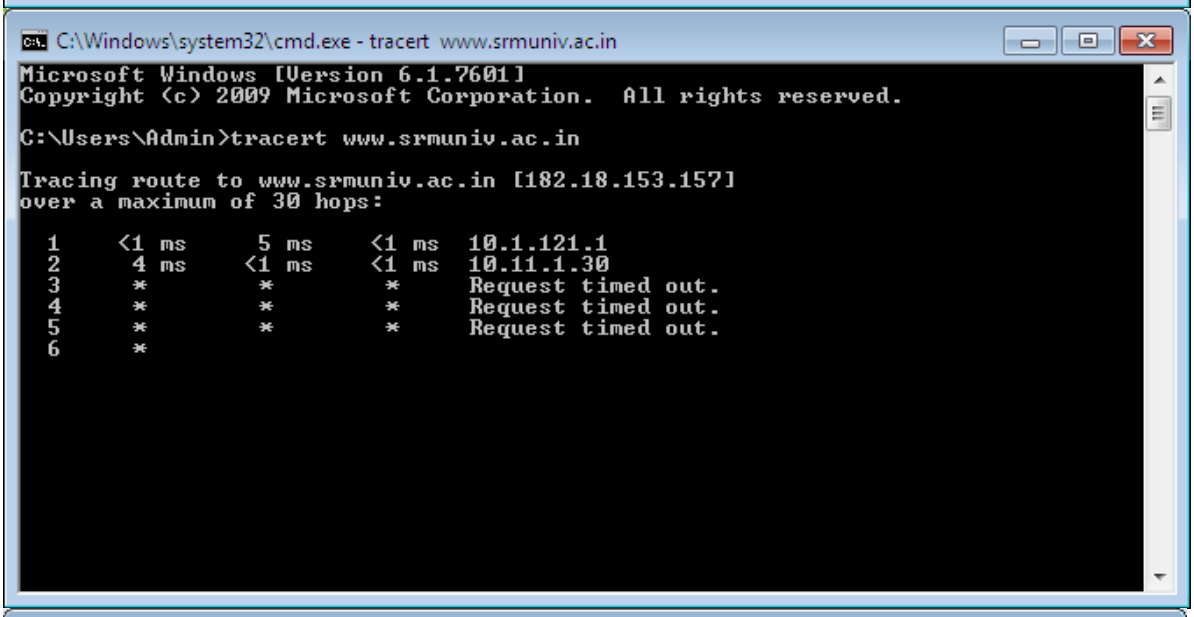
| S.no | Command | Use |
|---|---|---|
| 1. | ipconfig | This command can be utilized to verify a network connection as well as verify your network settings |
| 2. | Netstat | Displays active e TCP connections, ports on which the computer is listening, Ethernet statistics, the IP routing table etc. |
| 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 |

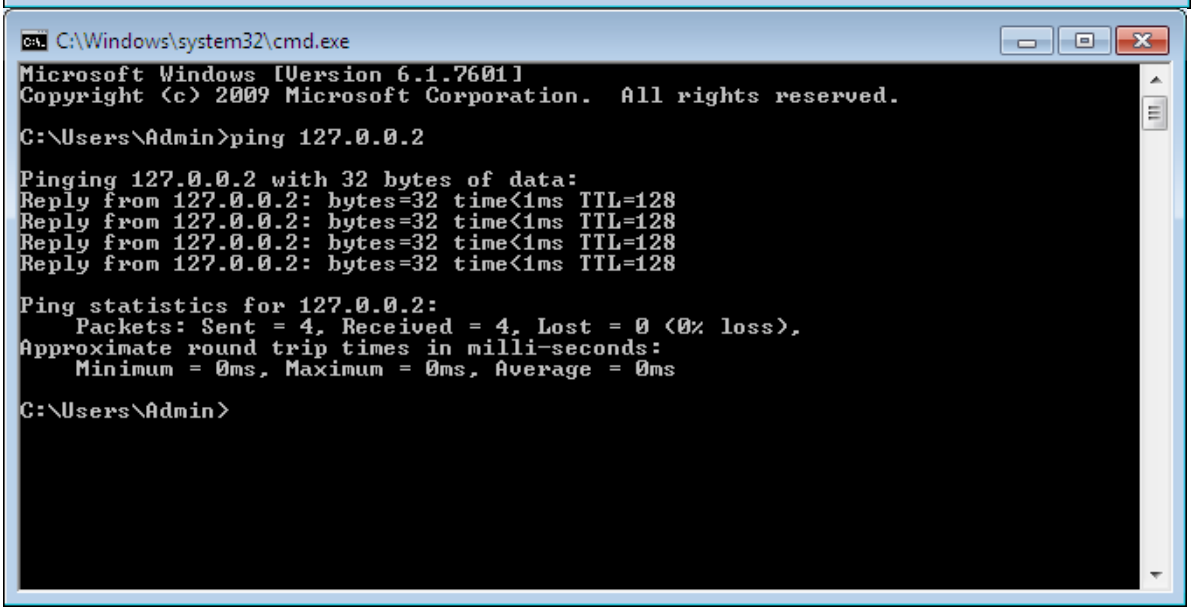
| | | |
|---|---|---|
| | | and destination pathping sends. |
| 6. | Nslookup | Displays information that you can use to diagnose Domain Name System (DNS) infrastructure |
| 7. | Nbtstat | MS_DOS utility that displays protocol statistics & current TCP/IP connections using NBT |
| 8. | getmac | DOS command 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<br>MAC address for the remote computer. |

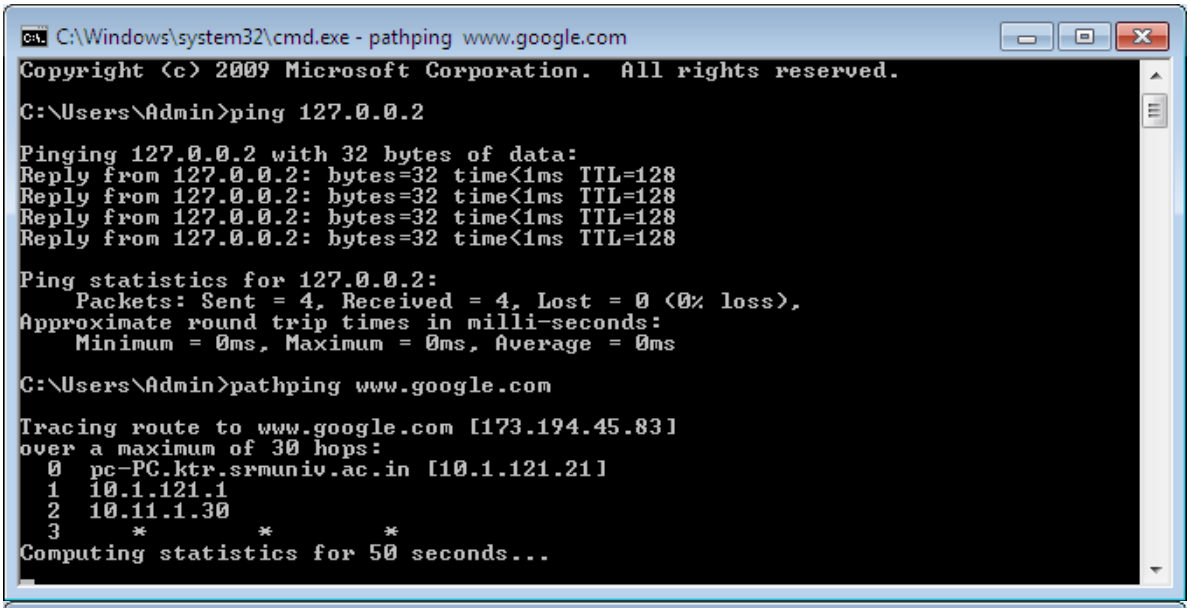
**OUTPUT:**

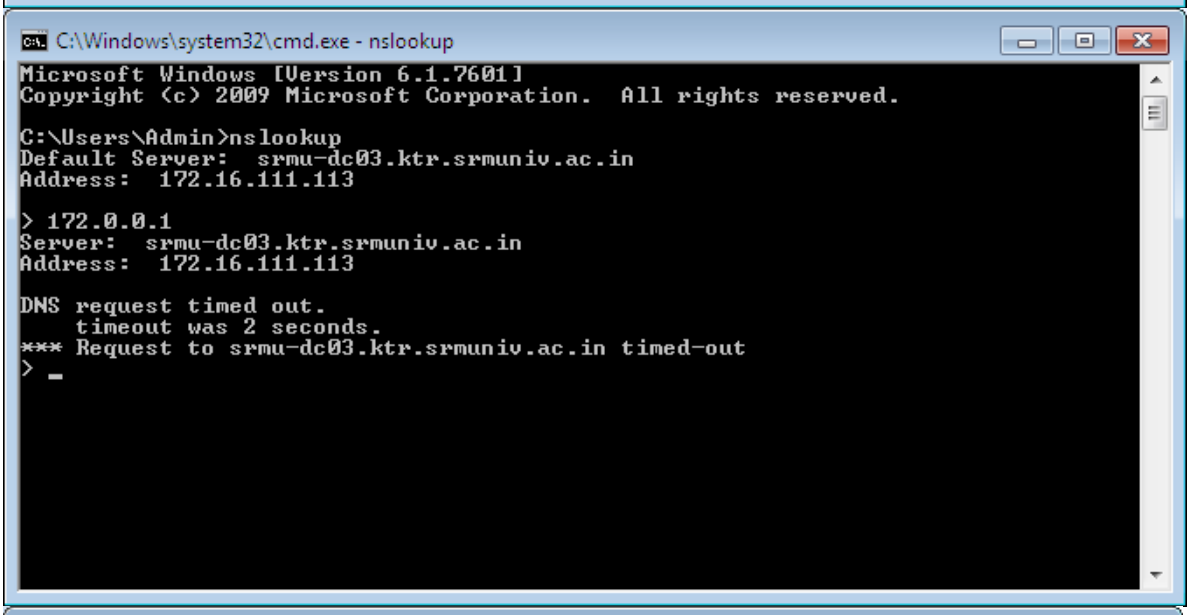


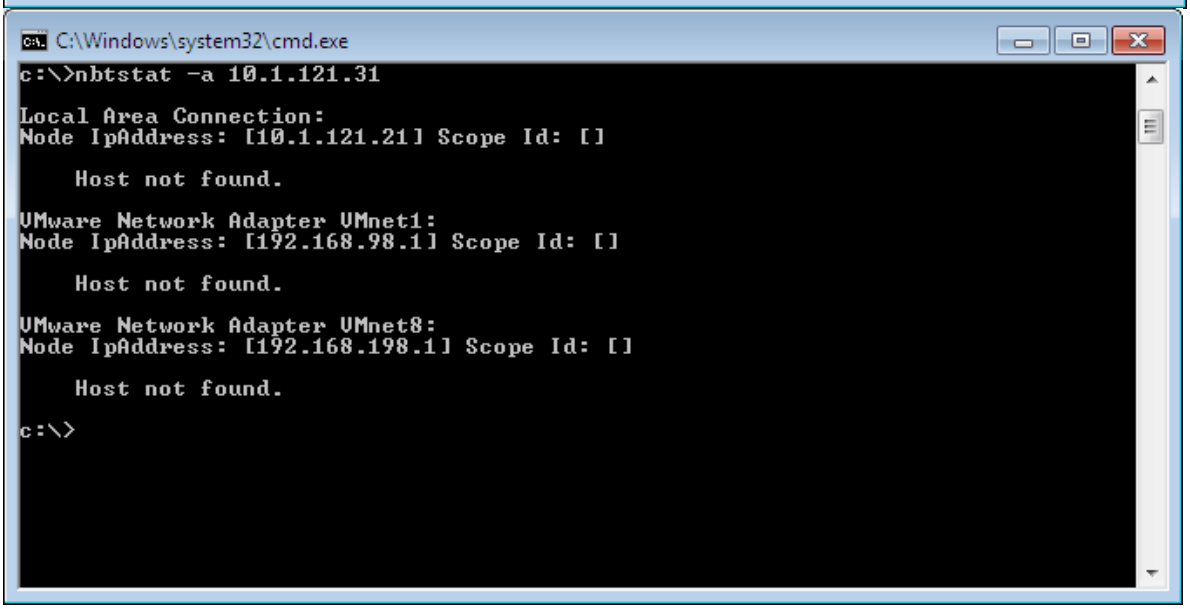

C:\Windows\system32\cmd.exe - netstat
C:\Users\Admin>netstat
Active Connections
Proto Local Address Foreign Address State
TCP 10.1.121.21:49772 Admin-PC:icslap ESTABLISHED
TCP 10.1.121.21:49773 WIN-E6653DANNHN:icslap ESTABLISHED
TCP 10.1.121.21:49775 admin1-PC:wsd TIME_WAIT
TCP 10.1.121.21:49776 user1-PC:icslap ESTABLISHED
TCP 10.1.121.21:49777 Admin-PC:wsd TIME_WAIT
TCP 10.1.121.21:49779 user1-PC:wsd TIME_WAIT
TCP 10.1.121.21:49780 adm-PC:wsd TIME_WAIT

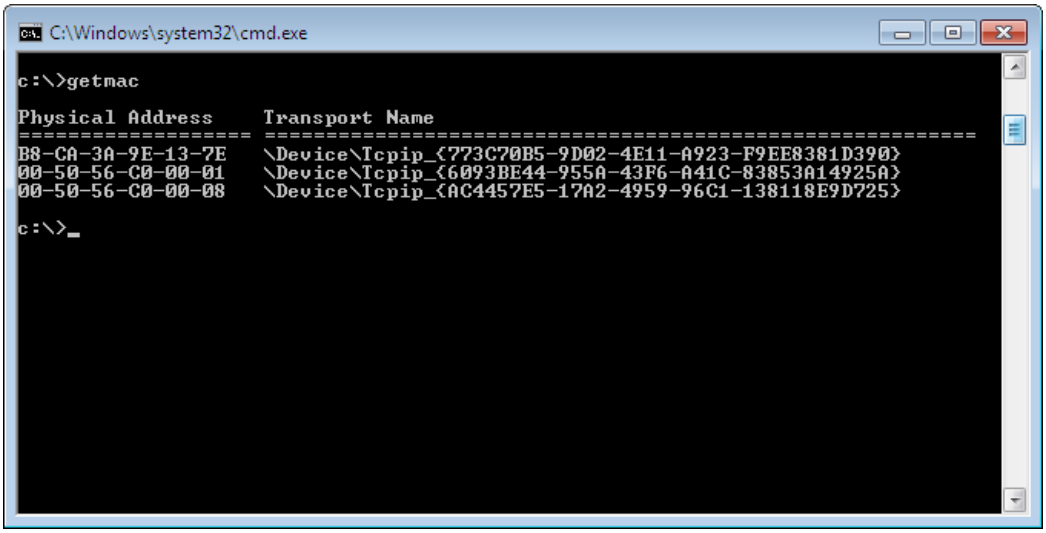

C:\Windows\system32\cmd.exe - tracert www.srmuniv.ac.in
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\Admin>tracert www.srmuniv.ac.in
Tracing route to www.srmuniv.ac.in [182.18.153.157]
over a maximum of 30 hops:
1 <1 ms 5 ms <1 ms 10.1.121.1
2 4 ms <1 ms <1 ms 10.11.1.30
3 * * * Request timed out.
4 * * * Request timed out.
5 * * * Request timed out.
6 *


C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\Admin>ping 127.0.0.2
Pinging 127.0.0.2 with 32 bytes of data:
Reply from 127.0.0.2: bytes=32 time<1ms TTL=128
Reply from 127.0.0.2: bytes=32 time<1ms TTL=128
Reply from 127.0.0.2: bytes=32 time<1ms TTL=128
Reply from 127.0.0.2: bytes=32 time<1ms TTL=128
Ping statistics for 127.0.0.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
C:\Users\Admin>


C:\Windows\system32\cmd.exe - pathping www.google.com
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\Admin>ping 127.0.0.2
Pinging 127.0.0.2 with 32 bytes of data:
Reply from 127.0.0.2: bytes=32 time<1ms TTL=128
Reply from 127.0.0.2: bytes=32 time<1ms TTL=128
Reply from 127.0.0.2: bytes=32 time<1ms TTL=128
Reply from 127.0.0.2: bytes=32 time<1ms TTL=128
Ping statistics for 127.0.0.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
C:\Users\Admin>pathping www.google.com
Tracing route to www.google.com [173.194.45.83]
over a maximum of 30 hops:
0 pc-PC.ktr.srmuniv.ac.in [10.1.121.21]
1 10.1.121.1
2 10.11.1.30
3 * * *
Computing statistics for 50 seconds...


C:\Windows\system32\cmd.exe - nslookup
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\Admin>nslookup
Default Server: srmu-dc03.ktr.srmuniv.ac.in
Address: 172.16.111.113
> 172.0.0.1
Server: srmu-dc03.ktr.srmuniv.ac.in
Address: 172.16.111.113
DNS request timed out.
timeout was 2 seconds.
*** Request to srmu-dc03.ktr.srmuniv.ac.in timed-out
>


C:\Windows\system32\cmd.exe
c:\>nbtstat -a 10.1.121.31
Local Area Connection:
Node IpAddress: [10.1.121.21] Scope Id: []
Host not found.
VMware Network Adapter VMnet1:
Node IpAddress: [192.168.98.1] Scope Id: []
Host not found.
VMware Network Adapter VMnet8:
Node IpAddress: [192.168.198.1] Scope Id: []
Host not found.
c:\>




**RESULT:**

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

**EX.NO: 2**

**DATE: 07/02/2022**

# 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.

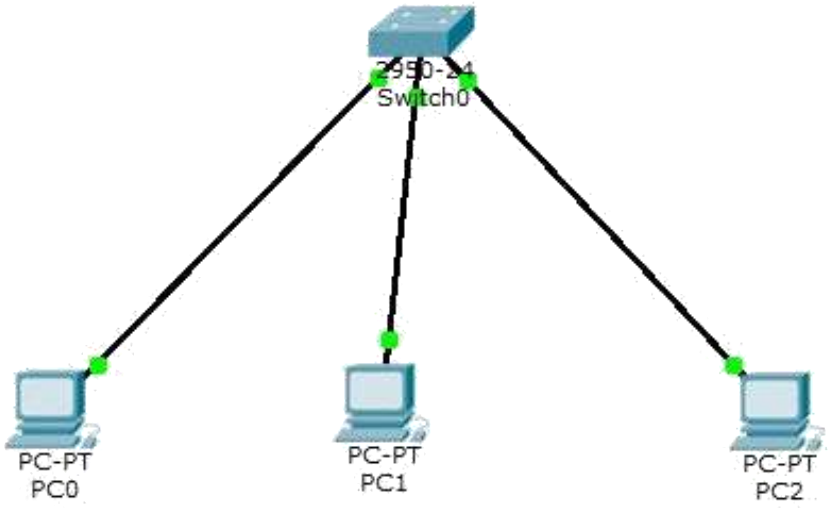
### REQUIREMENTS:

- Three Windows PC or 3 Linux PC.
- One Switch or One Hub.
- Three Copper-Straight Line LAN (cat-5) Cables.
- Power Supply.
- Cisco Packet Tracer 6.0.1

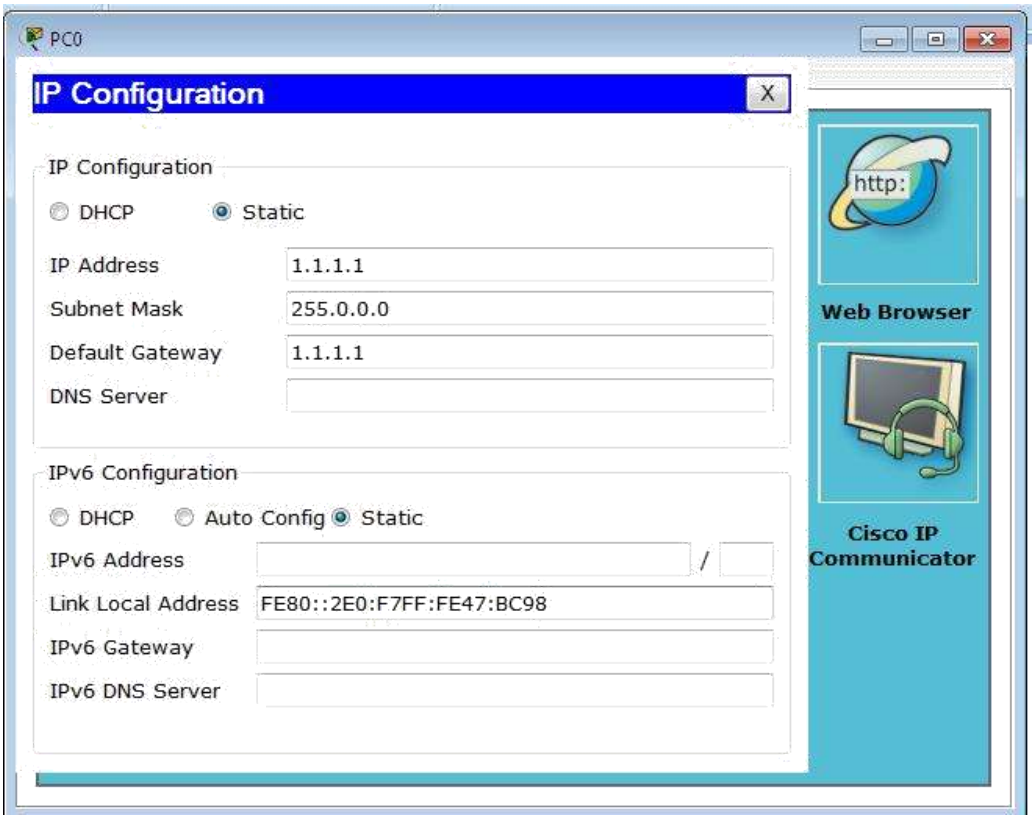
### PROCEDURES:

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

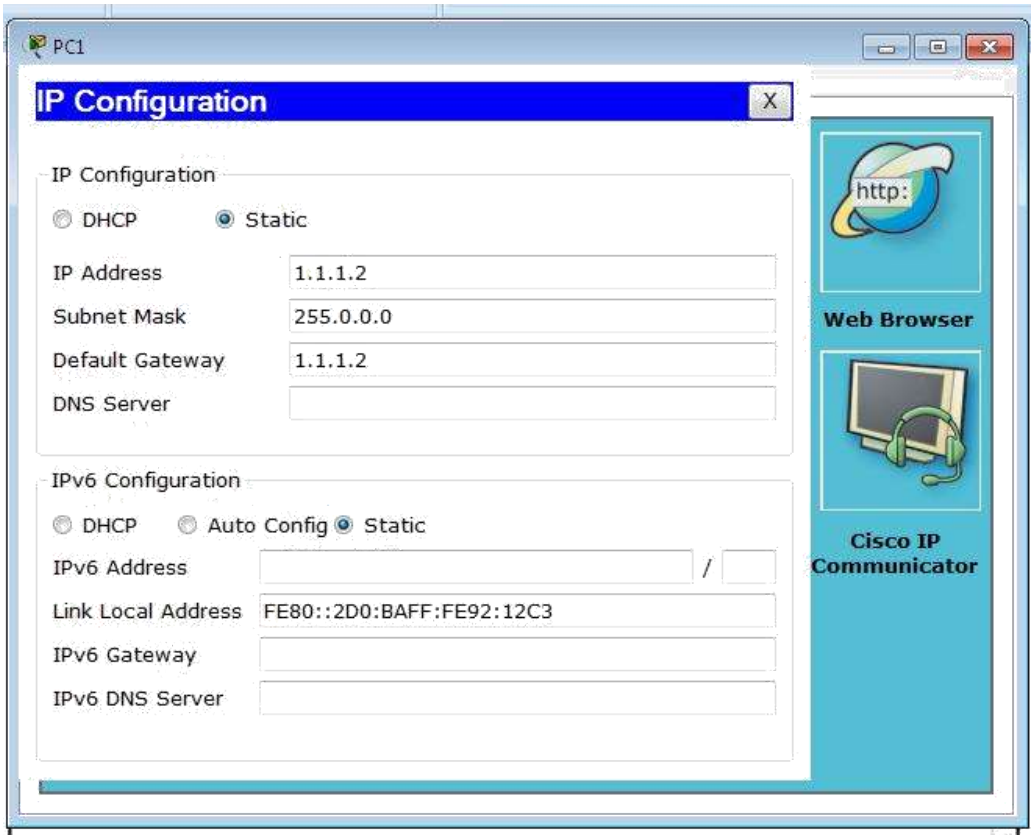
## NETWORK TOPOLOGY:



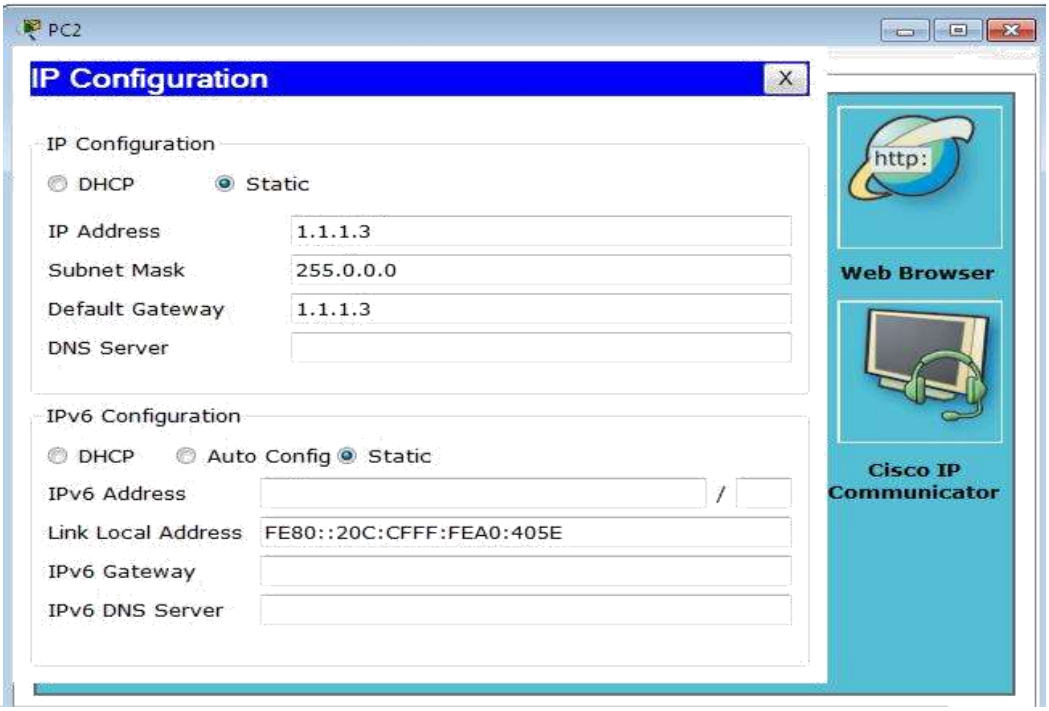

## HOST PC0 IP ADDRESS:



## HOST PC1 IP ADDRESS:



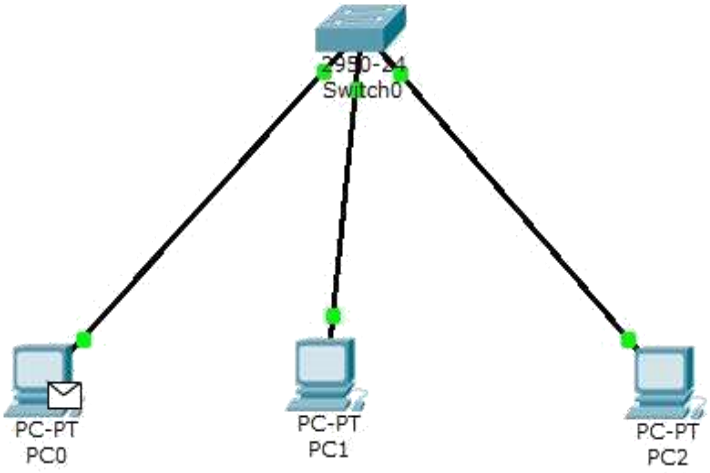

## HOST PC2 IP ADDRESS:

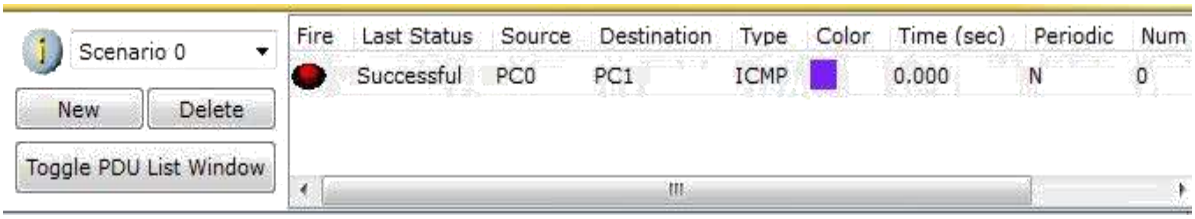


## VERIFY LAN NETWORK CONNECTIVITY:

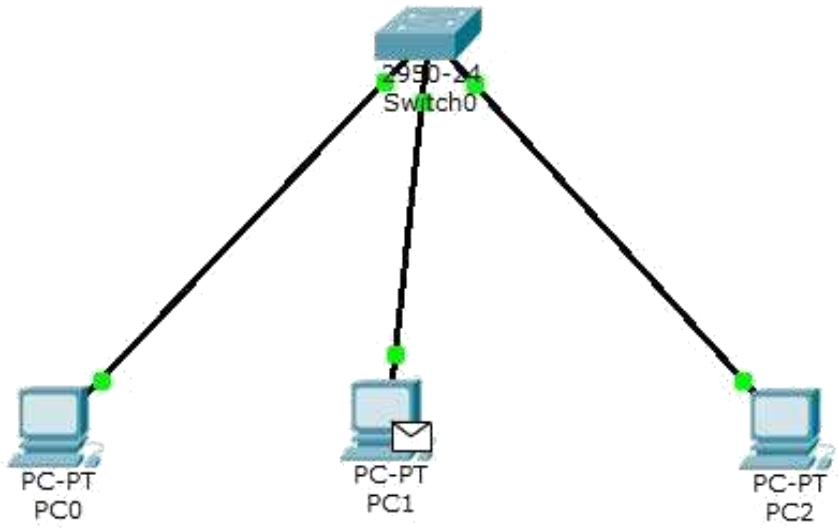
Using Add Simple PDU(p), Click the mail icon and then drop one mail to PC0 and another mail to PC1. If the resultant window shows the successful delivery, then network connectivity is successful or up.

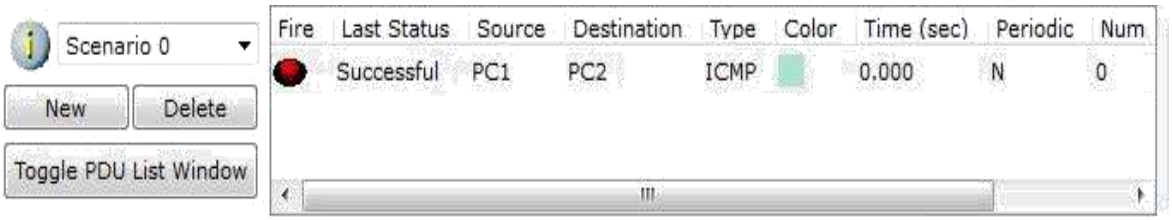
## HOST PC0 TO HOST PC1:



**HOST PC1 TO HOST PC2:**




**RESULT:**

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

**EX.NO: 3**

**DATE: 14/02/2022**

# 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.

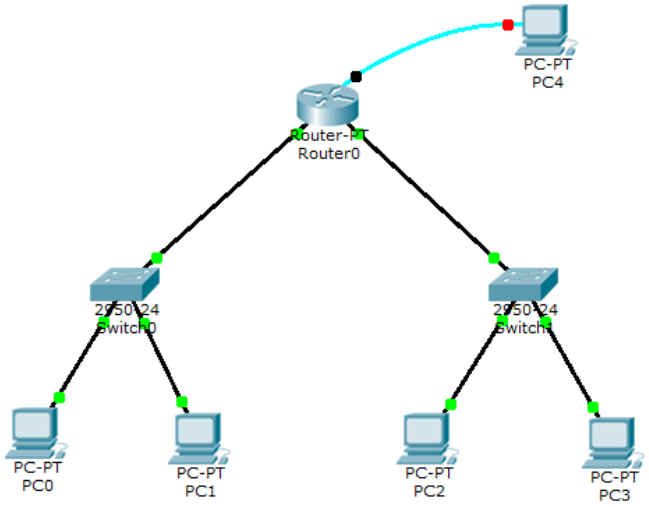
### REQUIREMENTS:

- Four windows PC or Four Linux PC.
- Two Switch (8 port) or 2 Hub.
- Six Straight Line LAN (cat-5) Cables with RJ-45 Sockets.
- Power supply
- Basic Network Configuration Commands. For Router, Switch and PCs.
- Cisco Packet Tracer 6.0.1
- Cisco Router (Model 1841)
- One console connection of router with PC to configure router.

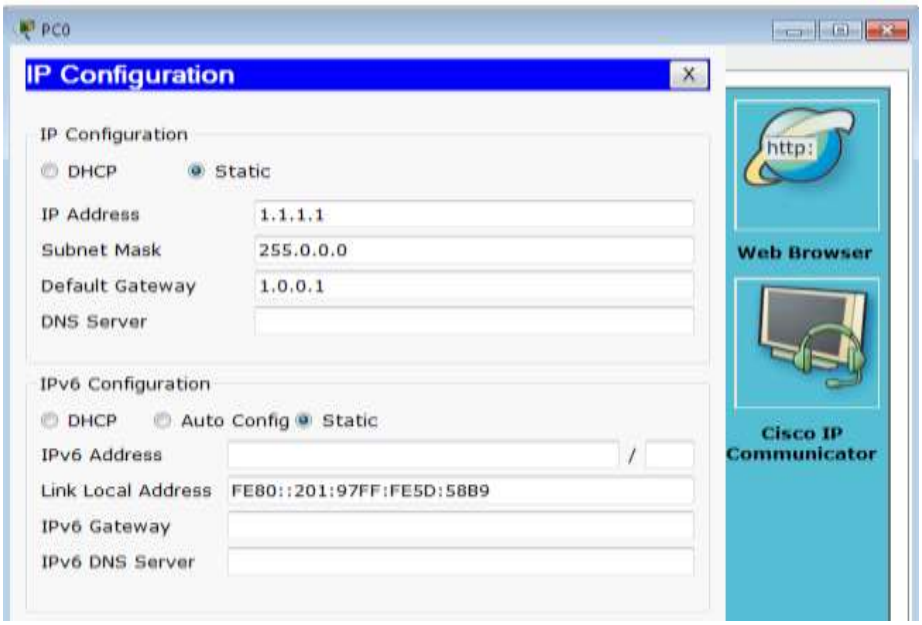
### PROCEDURES:

- Open the CISCO PACKET TRACER software.
- Draw the 4 PC using End Device Icons.
- Draw the 2 CISCO 24 Port Switch using Switch icon lists.
- Draw the Cisco Generic Routers using Router icon lists.
- Make the Connections using Straight-Through Ethernet Cables.
- Configure Router R0.
- Enter the IP Address to Each Machine.
- Check the Connections using Add Simple PDU(P).

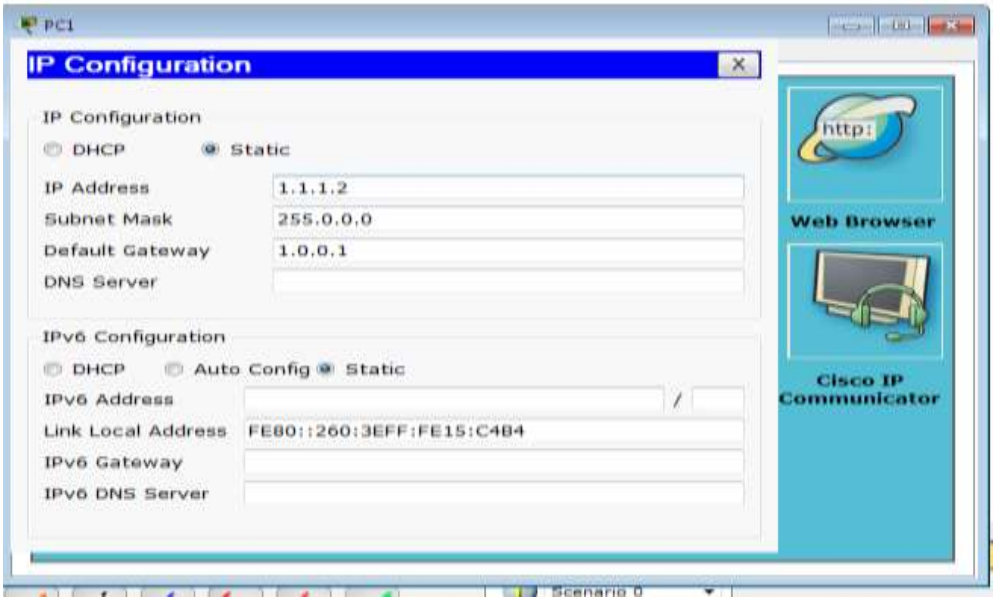
## NETWORK TOPOLOGY:



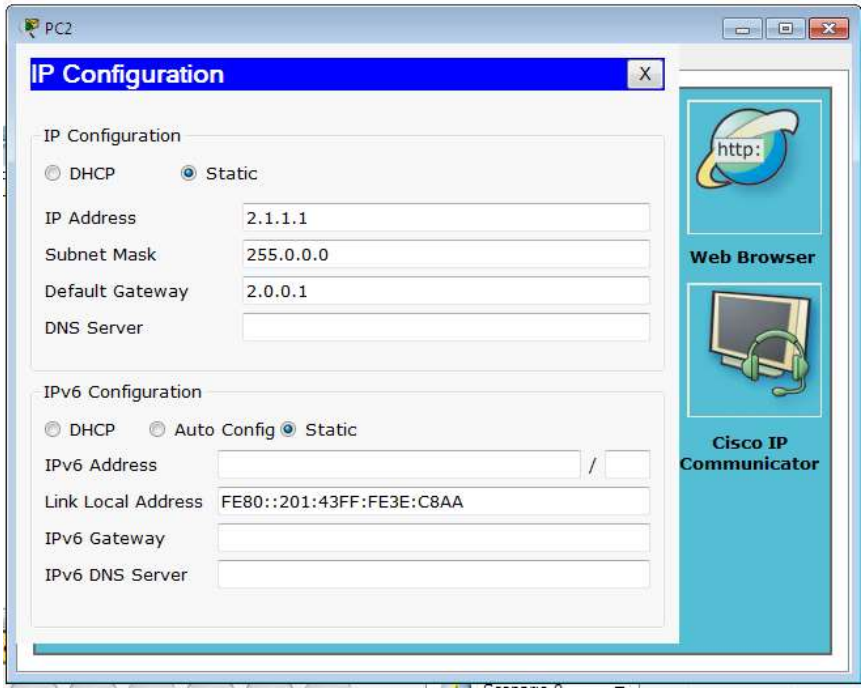

## PC0 IP CONFIGURATION ADDRESS:



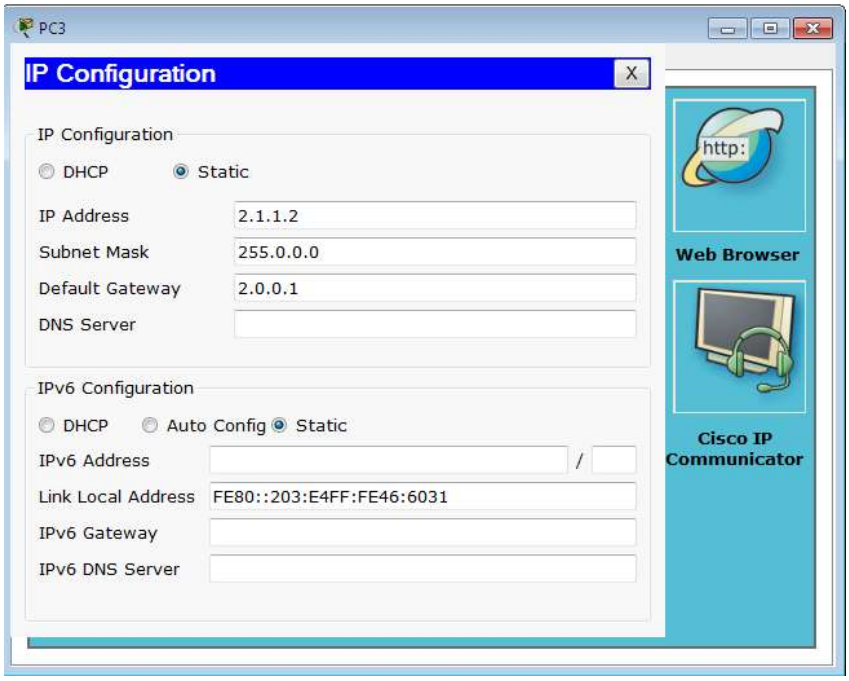
## PC1 IP CONFIGURATION ADDRESS:




## PC2 IP CONFIGURATION ADDRESS:



**PC3 IP CONFIGURATION ADDRESS:**




**ROUTER 0 CONFIGURATION:**

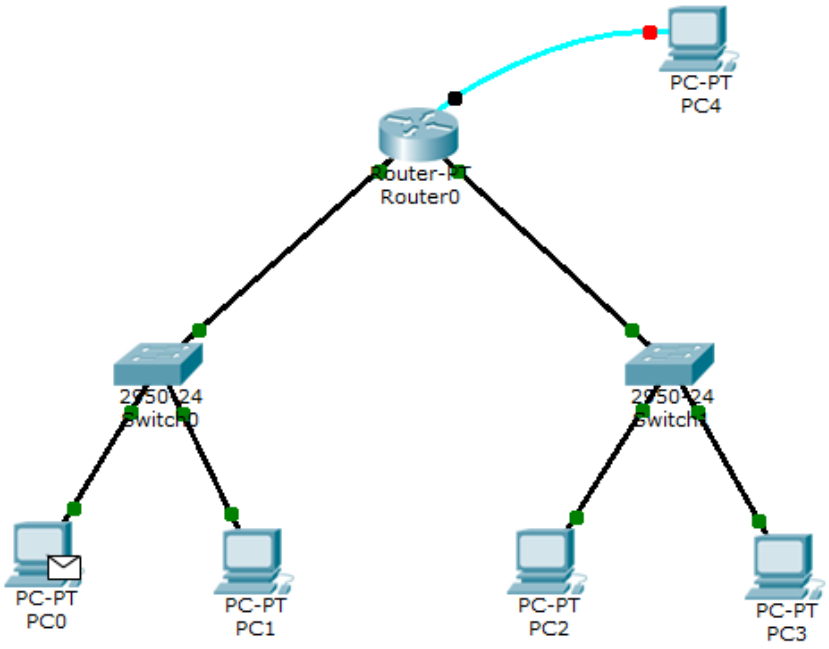
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface fastethernet 0/0
Router(config-if)#ip address 1.0.0.1 255.0.0.0
Router(config-if)#no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
Router(config-if)#exit

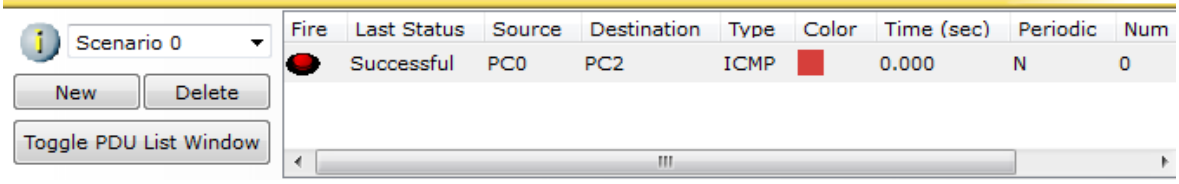
Router(config)#interface fastethernet 0/0
Router(config-if)#interface fastethernet 1/0
Router(config-if)#ip address 2.0.0.1 255.0.0.0
Router(config-if)#no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface FastEthernet1/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet1/0, changed state to up
Router(config-if)#exit

## VERIFY LAN NETWORK CONNECTIVITY

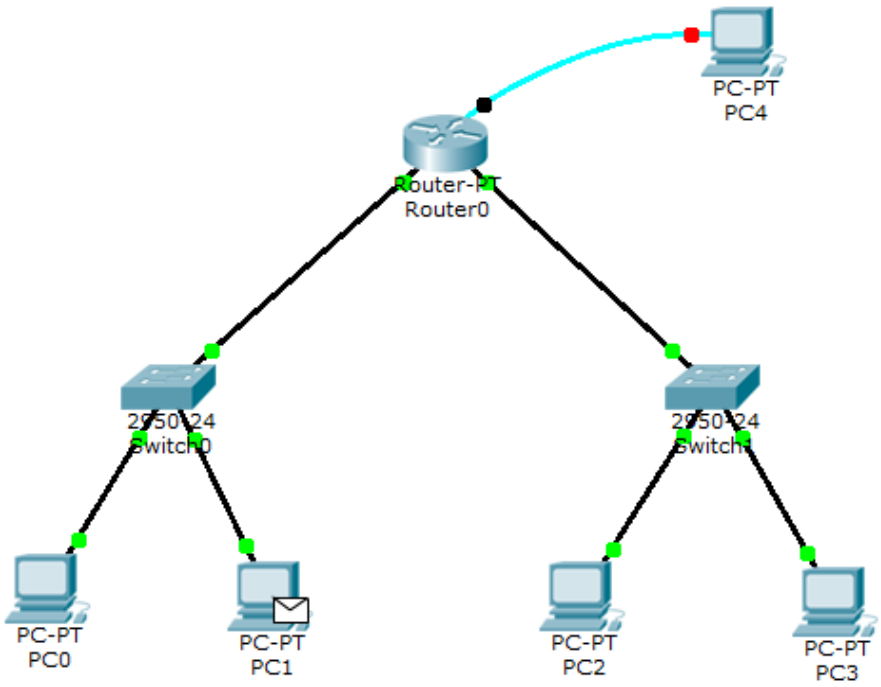
Using Add Simple PDU(p), Click the mail icon and then drop one mail to one of the PC in first LAN and another mail to PC in another LAN. If the resultant window shows the successful delivery of the mail then network connectivity is successful.

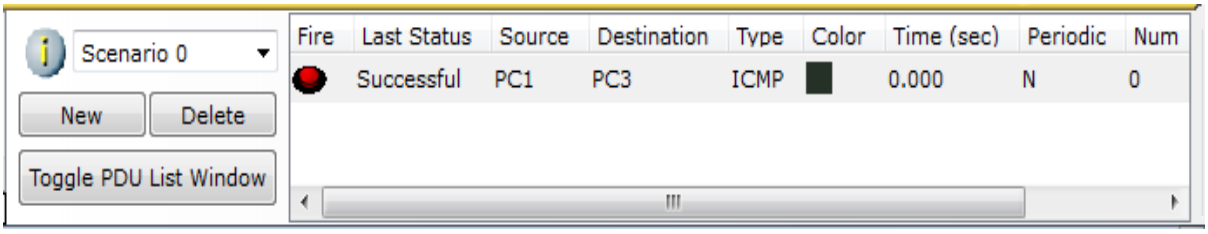
## PC0 TO PC2



**PC1 TO PC3**




**RESULT:**

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

**EX.NO: 4**

**DATE: 21/02/2022**

# ANALYZING THE PERFORMANCE OF VARIOUS CONFIGURATIONS AND PROTCOLS IN LAN

## MULTI-ROUTING CONNECTION WITH STATIC ROUTER

## AIM:

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

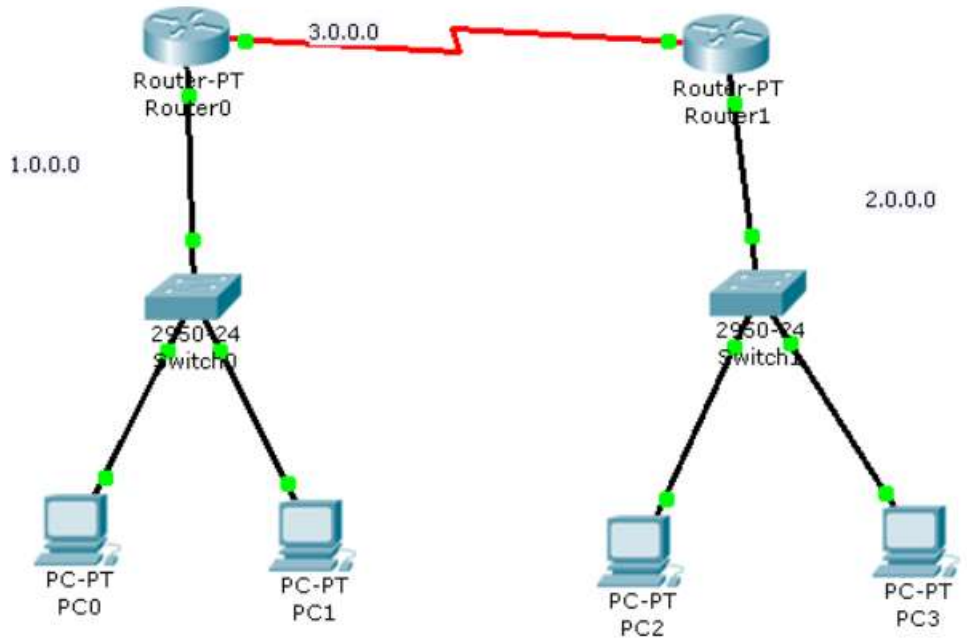
## REQUIREMENTS:

- Four windows PC or Four Linux PC.
- Two Switch (8 port).
- Six Straight Line LAN (cat-5) Cables with RJ-45 Sockets.
- Power supply
- Basic Network Configuration Commands. For Router, Switch and PCs.
- Cisco Packet Tracer 6.0.1
- Two Cisco Routers (Model 1841)

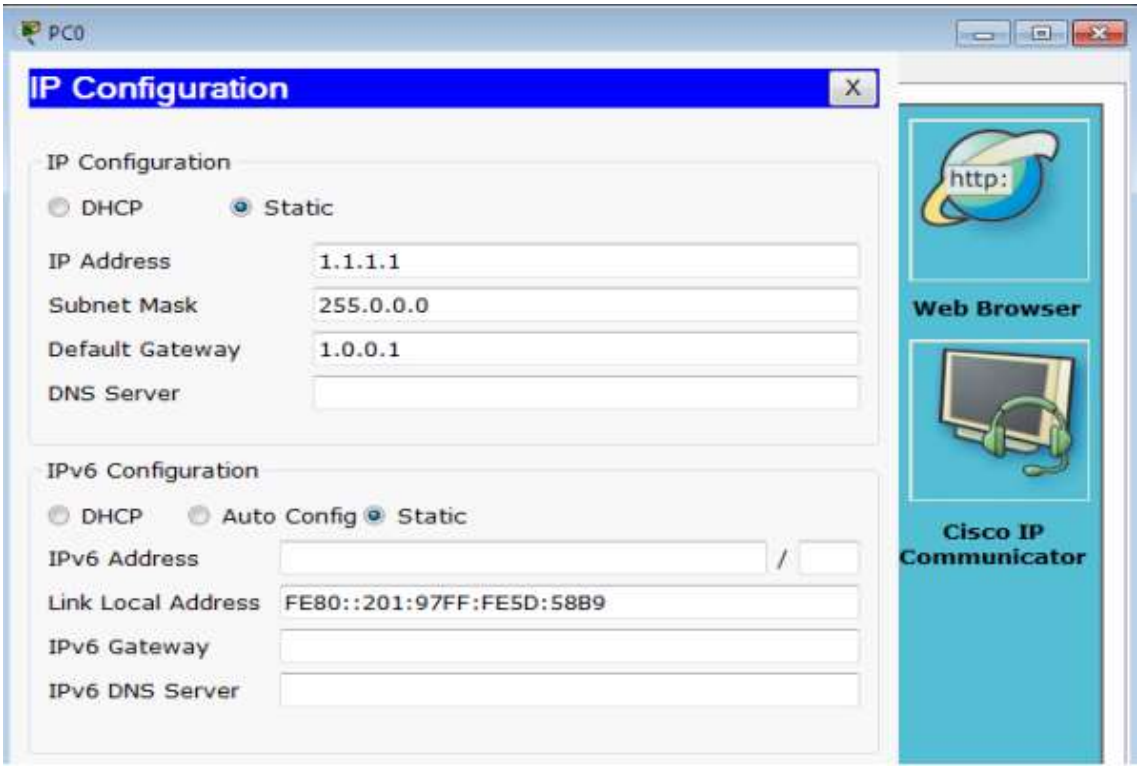
## PROCEDURES:

1. Open the CISCO PACKET TRACER software.
2. Draw the 4 PC using End Device Icons.
3. Draw the 2 CISCO 24 Port Switch using Switch icon lists.
4. Draw the two Cisco Generic Routers using Router icon lists.
5. Make the Connections using Straight-Through Ethernet Cables.
6. Configure Routers R0 and R1.
7. Enter the IP Address to Each Machine.
8. Configuring Static Routing for Each router.
9. Check the Connections using Add Simple PDU(P).

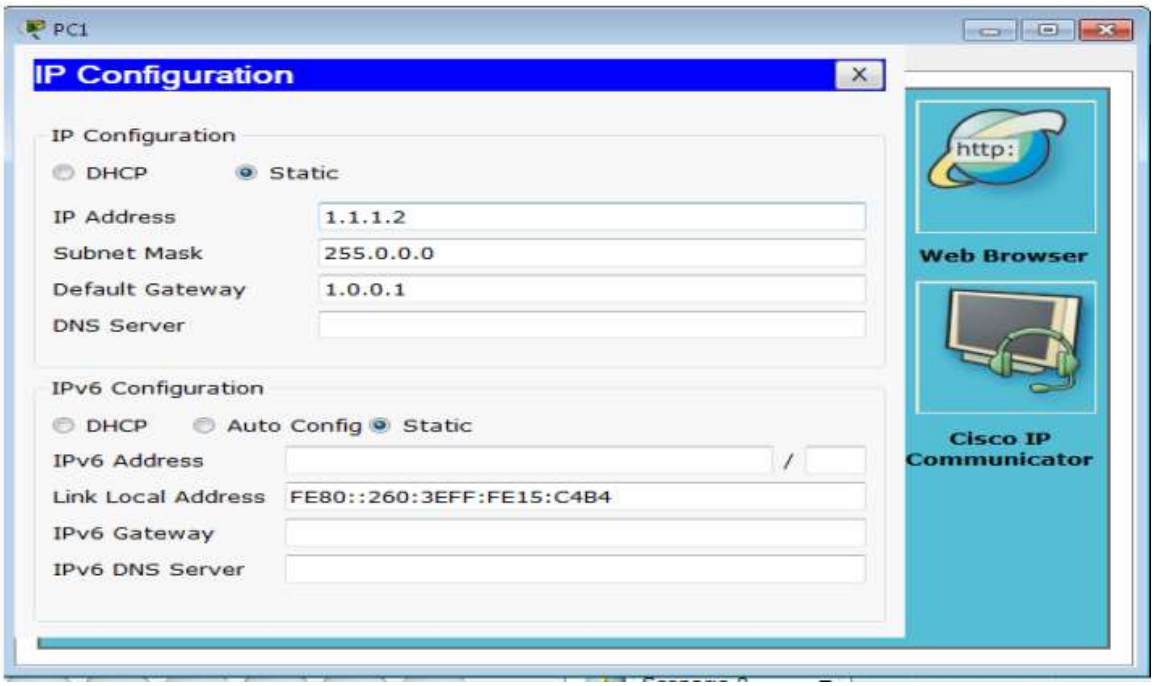
## NETWORK TOPOLOGY:



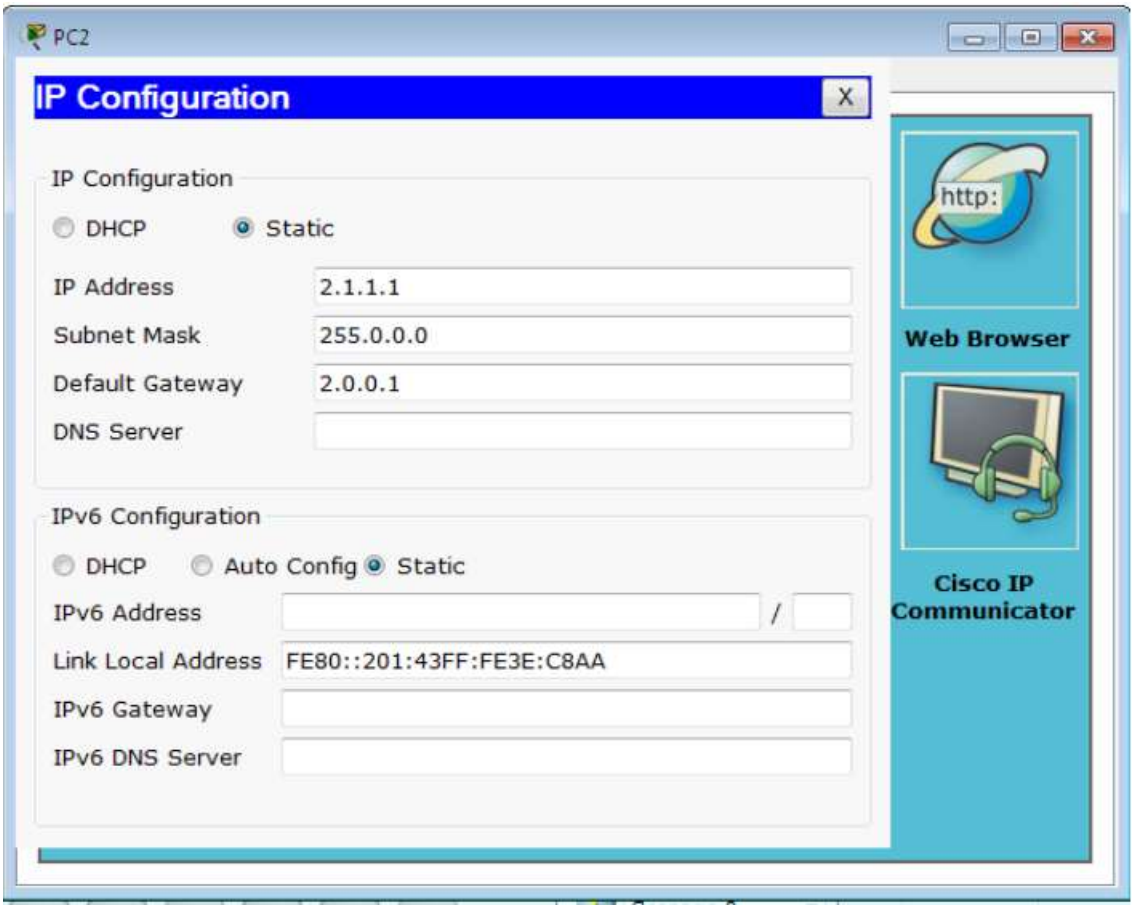

## PC0 IP CONFIGURATION ADDRESS:



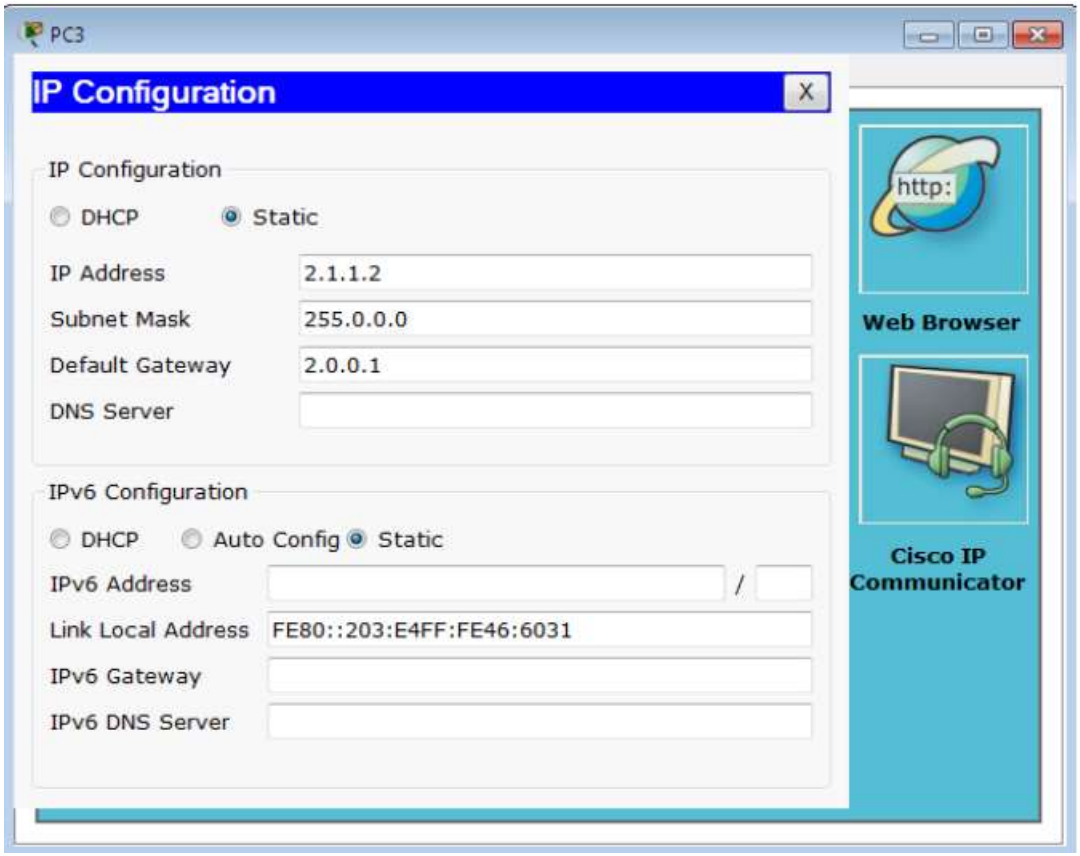
## PC1 IP CONFIGURATION ADDRESS:




## PC2 IP CONFIGURATION ADDRESS:



**PC3 IP CONFIGURATION ADDRESS:**




**ROUTER 0 CONFIGURATION:**

Router>enable

Router#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)#interface fastethernet 0/0

Router(config-if)#ip address 1.0.0.1 255.0.0.0

Router(config-if)#no shutdown

Router(config-if)#

%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to

up

%LINEPROTO-5-UPDOWN: Line protocol on Interface

FastEthernet0/0, changed state to up

Router(config-if)#exit

Router#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)#interface serial 2/0

Router(config-if)#ip address 3.0.0.1 255.0.0.0

Router(config-if)#clock rate 64000

Router(config-if)#no shutdown

%LINK-5-CHANGED: Interface Serial2/0, changed state to down

Router(config-if)#exit

Router(config)#ip route 2.0.0.0 255.0.0.0 3.0.0.0

**ROUTER 1 CONFIGURATION:**

Router>enable

Router#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)#interface fastethernet 0/0

Router(config-if)#ip address 2.0.0.1 255.0.0.0

Router(config-if)#no shutdown

Router(config-if)#

%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to

up

%LINEPROTO-5-UPDOWN: Line protocol on Interface

FastEthernet0/0, changed state to up

Router(config-if)#exit

Router#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)#interface serial 2/0

Router(config-if)#ip address 3.0.0.0 255.0.0.0

Bad mask /8 for address 3.0.0.0

Router(config-if)#ip address 3.0.0.1 255.0.0.0

Router(config-if)#clock rate 64000

Router(config-if)#no shutdown

%LINK-5-CHANGED: Interface Serial2/0, changed state to down

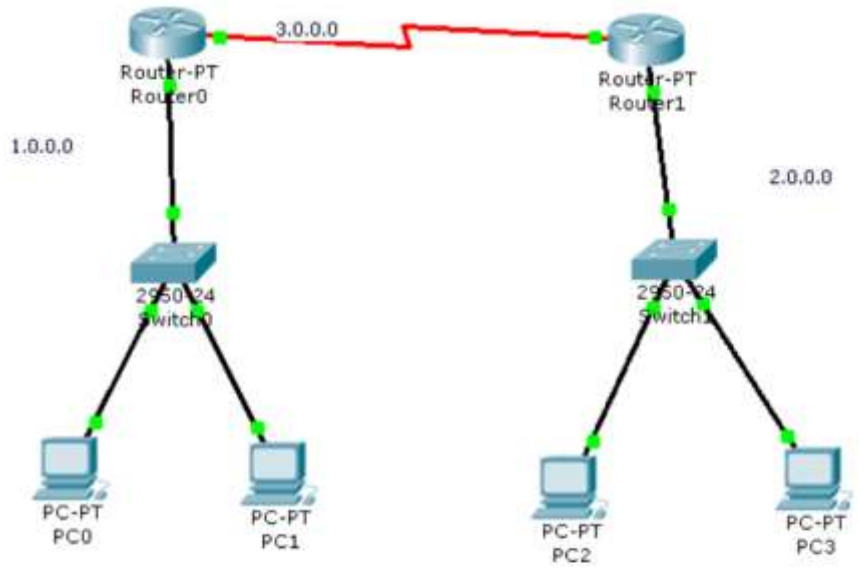
Router(config-if)#exit

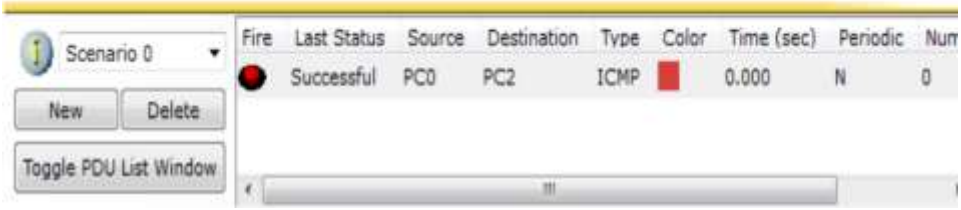
Router(config)#ip route 1.0.0.0 255.0.0.0 3.0.0.0

**VERIFY LAN NETWORK CONNECTIVITY:**

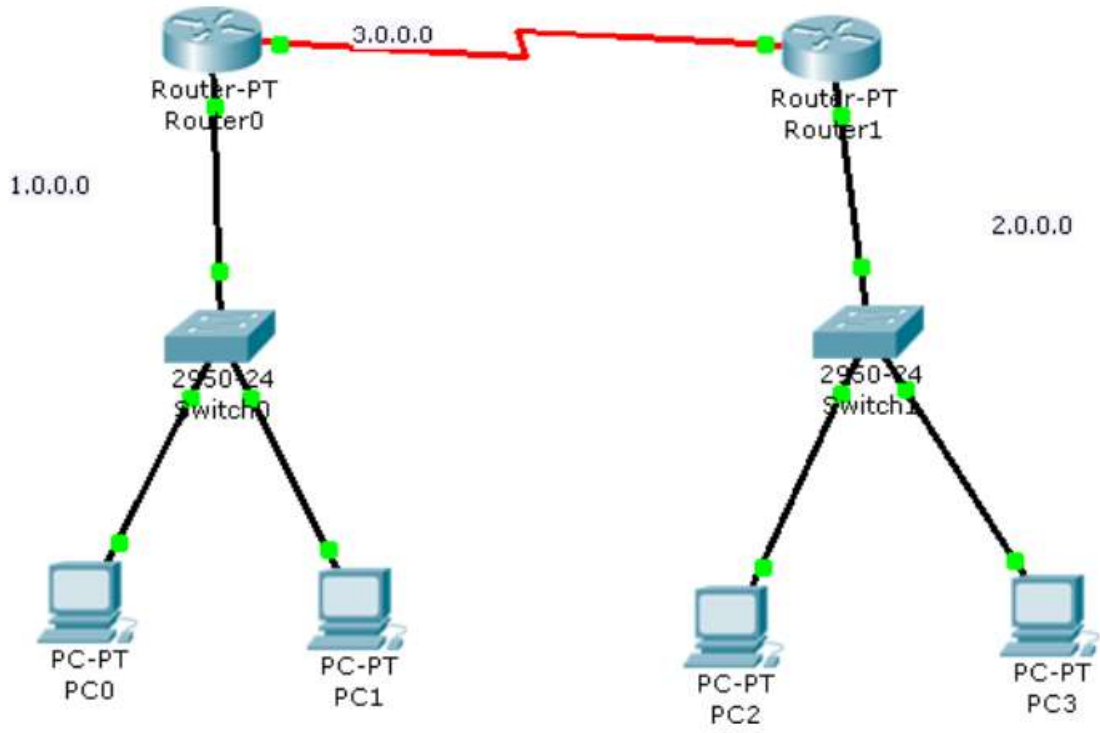
Using Add Simple PDU(p), Click the mail icon and then drop one mail to one of the PC in first LAN and another mail to PC in another LAN. If the resultant windows show the successful delivery of the mail then network connectivity is successful.

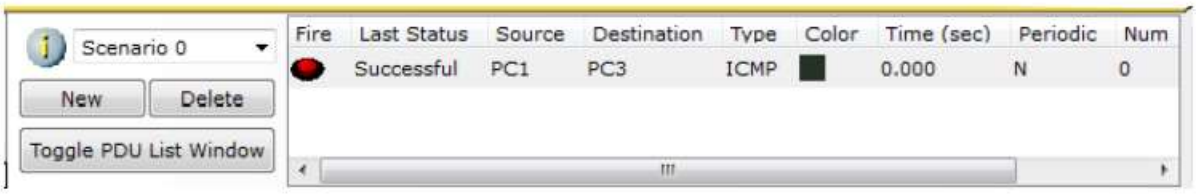
**PC0 TO PC2:**



**PC1 TO PC3:**




**RESULT:**

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

**EX.NO: 5**

**DATE: 28/02/2022**

## IMPLEMENTING MINI SEARCH ENGINE

### AIM:

To implement Google search in a web page.

### 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:

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

### PROGRAM:

```
<html>
<body bgcolor="skyblue">
<center><font size="36" color = "blue" 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=""/>
THE WEB
```

<input type="radio" name="sitesearch" value="srmuniv.ac.in" checked />SRM UNIVERSITY<br>

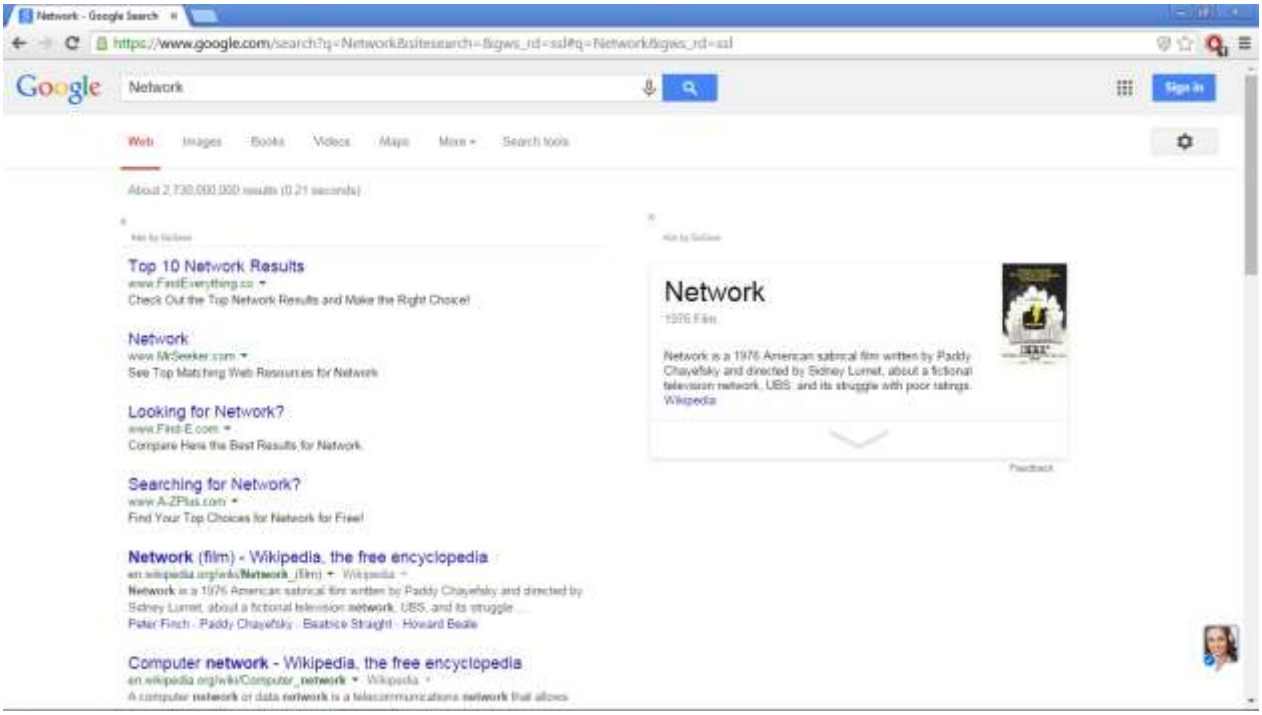
</form>

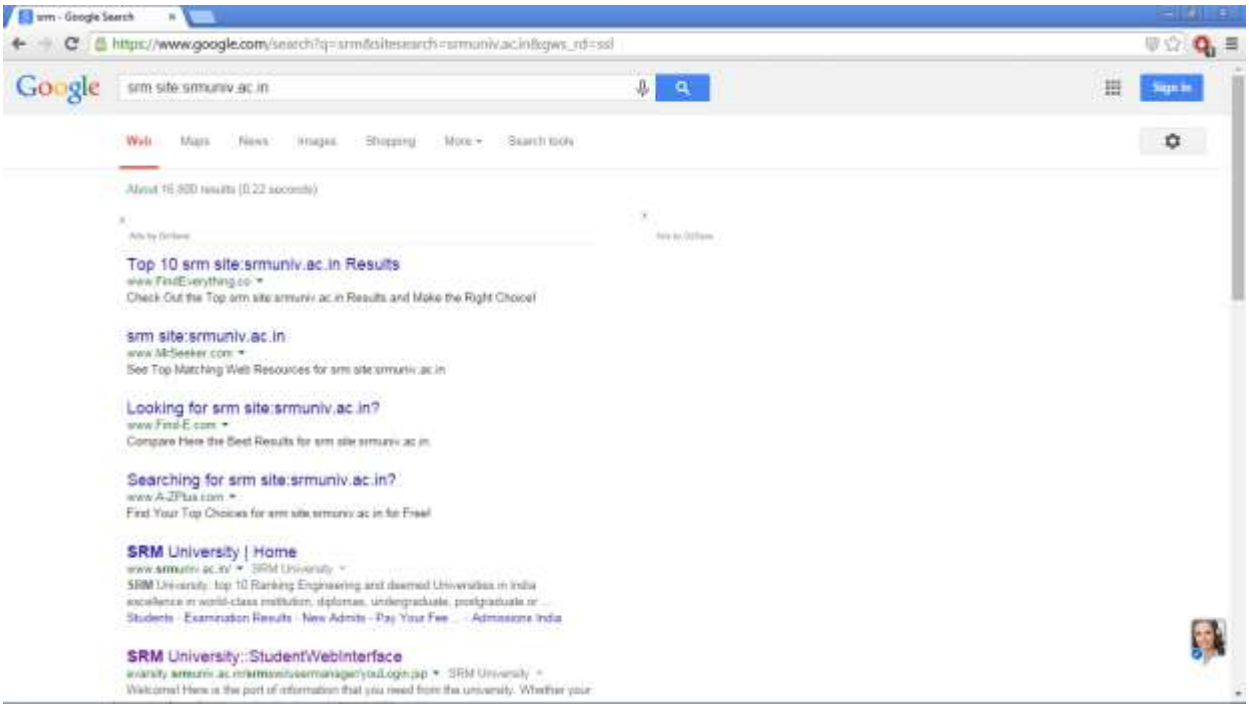
<imgsrc="E:\Networkslab\srmlogo.gif">

</body>

</html>

**OUTPUT:**


## RESULT:

Thus, the program to implement a mini search engine is executed successfully and the output is verified.

**EX.NO: 6**

**DATE: 14/03/2022**

# IMPLEMENTING SIMPLE WEB SERVER

## AIM:

To write a java web server that will read and process simple HTTP requests from the client

## PROCEDURE:

- In the main method of the java class, create a server socket and bind it to the port
- Read the HTTP request from the client
- Depending on the type of file requested, retrieve the appropriate file
- Send the file back to the client.
- Close the streams
- Close the server socket.

## HOW TO EXECUTE:

- Open the command prompt
- Save and compile the java file using the command javac<file name>.java
- Run the server program using the command java <classname>
- Open the browser and type the URL of the server in the browsers address bar
- The URL is of the form http://localhost:portno/file.html
- Note the web server is developed only to support html, gif and jpeg file types.

**SOURCE CODE:**

**WebServer.java:**

```
import java.io.*;
import java.net.*;
importjava.util.*;
classWebServer
{
public static void main(String args[]) throws Exception
  {
        String request;
        String file;
        ServerSocket server = new ServerSocket(6001);
        Socket socket = server.accept();
        BufferedReader in = new BufferedReader(new
                                InputStreamReader(socket.getInputStream()));
        DataOutputStream out = new DataOutputStream(socket.getOutputStream());
                                        request = in.readLine();
        StringTokenizer token = new StringTokenizer(request);
        if (token.nextToken().equals("GET")){
        file = token.nextToken();
        if (file.startsWith("/") == true )
        file = file.substring(1);
        File filef = new File(file);
        intlen = (int) filef.length();
        FileInputStreaminFile = new FileInputStream (filef);
        byte[] byt = new byte[len];
        inFile.read(byt);
        out.writeBytes("HTTP/1.0 200 Document Follows\r\n");
```

```
        if (file.endsWith(".jpg"))
        out.writeBytes("Content-Type: image/jpeg\r\n");
        if (file.endsWith(".gif"))
        out.writeBytes("Content-Type: image/gif\r\n");
        out.writeBytes("Content-Length: " + len + "\r\n");
        out.writeBytes("\r\n");
        out.write(byt, 0, len);
        socket.close();
    }
else
        System.out.println("Bad Request Message");
    }
}
```

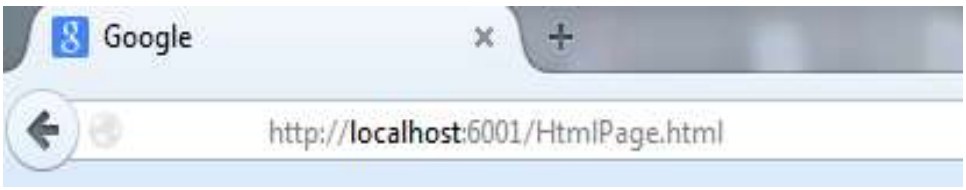
**HtmlPage.html:**

```
<html>
<head>
<title>Web Server</title>
</head>
<body>
<h1>Hello SRM!!!!</h1>
</body>
</html>
```

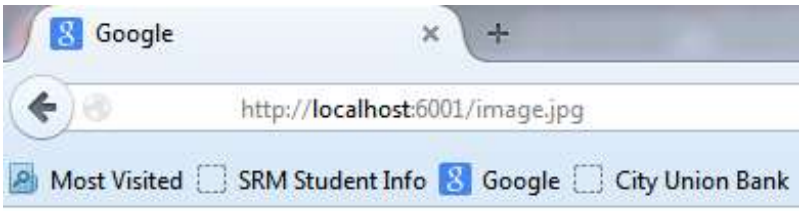
## OUTPUT:



## WORKING WITH HTML PAGE:




**WORKING WITH IMAGE:**





**RESULT:**

Thus, the program for implementing a simple web server is successfully executed and the output is verified.

**EX.NO: 7**

**DATE: 21/03/2022**

## IMPLEMENTING ERROR DETECTING CODE USING CRC-CCITT

**AIM:**

To write a java program for error detecting code using CRC-CCITT (16 bits)

**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:**

1)
Enter message

1011

Message=1011

CRC=

1011000101101011

transmitted frame is 10111011000101101011

Enter received data

10111011000101101011

Correct bits received

2)
Enter message

11000

Message=11000

CRC=

1001001100111001

transmitted frame is 110001001001100111001

Enter received data

110001001001100111000

Received frame contains one or more errors

3)
Enter message

1001

Message=1001

CRC=

1001000100101001

transmitted frame is 10011001000100101001

Enter received data

10011001000100101001

Correct bits received

**RESULT:**

Thus, the program for error detecting code using CRC-CCITT (16 bits) is successfully executed and the output is verified

**EX.NO: 8**

**DATE: 28/03/2022**

## IMPLEMENTATION OF SLIDING WINDOW PROTOCOL

**AIM:**

To write a java program to perform sliding window protocol

**ALGORITHM:**

1.Start the program.
2.Get the frame size from the user
3.To create the frame based on the user request.
4.To send frames to server from the client side.
5.If your frames reach the server, it will send ACK signal to client otherwise it will send NACK signal to client.
6.Stop the program

**PROGRAM:**

**slidsender.java:**

```
import java.net.*;
import java.io.*;
import java.rmi.*;
 public class slidsender{
    public static void main(String a[])throws Exception{
       ServerSocket ser=new ServerSocket(10);
       Socket s=ser.accept();
       DataInputStream in=new DataInputStream(System.in);
       DataInputStream in1=new DataInputStream(s.getInputStream());
       String sbuff[]=new String[8];
       PrintStream p;
       int sptr=0,sws=8,nf,ano,i;
       String ch;
       do{
          p=new PrintStream(s.getOutputStream());
          System.out.print("Enter theno. of frames : ");
          nf=Integer.parseInt(in.readLine());
```

```
            p.println(nf);
            if(nf<=sws-1){
               System.out.println("Enter "+nf+" Messages to be send\n");
               for(i=1;i<=nf;i++){
                  sbuff[sptr]=in.readLine();
                  p.println(sbuff[sptr]);
                  sptr=++sptr%8;
               }
            sws-=nf;
            System.out.print("Acknowledgment received");
            ano=Integer.parseInt(in1.readLine());
            System.out.println(" for "+ano+" frames");
            sws+=nf;
           }
           else{
               System.out.println("The no. of frames exceeds window size");
               break;
            }
           System.out.print("\nDo you wants to send some more frames : ");
           ch=in.readLine();
           p.println(ch);
         }
         while(ch.equals("yes"));
         s.close();
     } }
```

**slidreceiver.java:**

```
import java.net.*;
import java.io.*;
class slidreceiver{
   public static void main(String a[])throws Exception{
      Socket s=new Socket(InetAddress.getLocalHost(),10);
      DataInputStream in=new DataInputStream(s.getInputStream());
      PrintStream p=new PrintStream(s.getOutputStream());
      int i=0,rptr=-1,nf,rws=8;
      String rbuf[]=new String[8];
      String ch;System.out.println();
      do{
         nf=Integer.parseInt(in.readLine());
         if(nf<=rws-1){
            for(i=1;i<=nf;i++){
```

```
                rptr=++rptr%8;rbuf[rptr]=in.readLine();
                System.out.println("The received Frame " +rptr+" is : "+rbuf[rptr]);
              }
          rws-=nf;System.out.println("\nAcknowledgment sent\n");
          p.println(rptr+1); rws+=nf;
          }
          else
             break;
          ch=in.readLine();
       }while(ch.equals("yes"));
    }
}
```

**OUTPUT:**

**slidsender.java**

```
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

D:\Netbeans\JavaApplication2\src\javaapplication2>javac slidsender.java
Note: slidsender.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

D:\Netbeans\JavaApplication2\src\javaapplication2>java slidsender
Enter theno. of frames : 2
Enter 2 Messages to be send

How are you
Hello
Acknowledgment received for 2 frames

Do you wants to send some more frames : no

D:\Netbeans\JavaApplication2\src\javaapplication2>
```

**slidreceiver.java**

```
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

D:\Netbeans\JavaApplication2\src\javaapplication2>javac slidreceiver.java
Note: slidreceiver.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

D:\Netbeans\JavaApplication2\src\javaapplication2>java slidreceiver

The received Frame 0 is : How are you
The received Frame 1 is : Hello

Acknowledgment sent


D:\Netbeans\JavaApplication2\src\javaapplication2>
```

**RESULT:**

Thus, the program for implementing a sliding window protocol is successfully executed and the output is verified.

**EX.NO: 9**

**DATE: 04/04/2022**

## IMPLEMENTATION OF STOP AND WAIT PROTOCOL

**AIM:**

To write a java program to perform Stop and Wait protocol

**ALGORITHM:**

1.Start the program.
2.Get the frame size from the user
3.To create the frame based on the user request.
4.To send frames to server from the client side.
5.If your frames reach the server, it will send ACK signal to client otherwise it will send NACK signal to client.
6.Stop the program

**PROGRAM:**

**Sender.java**

```
import java.io.*;
import java.net.*;
public class Sender{
   Socket sender;ObjectOutputStream out;
   ObjectInputStream in;
   String packet,ack,str, msg;
   int n,i=0,sequence=0;
   Sender(){}
   public void run(){
      try{
         BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
         System.out.println("Waiting for Connection....");
         sender = new Socket("localhost",2004);
         sequence=0;
         out=new ObjectOutputStream(sender.getOutputStream());
         out.flush();
         in=new ObjectInputStream(sender.getInputStream());
```

```
            str=(String)in.readObject();
            System.out.println("reciver > "+str);
            System.out.println("Enter the data to send....");
            packet=br.readLine();
            n=packet.length();
        do{
            try{
                if(i<n){
                    msg=String.valueOf(sequence);
                    msg=msg.concat(packet.substring(i,i+1));
                }
                else if(i==n){
                    msg="end";
                    out.writeObject(msg);
                    break;
                }
                out.writeObject(msg);
                sequence=(sequence==0)?1:0;
                out.flush();
                System.out.println("data sent>"+msg);
                ack=(String)in.readObject();
                System.out.println("waiting for ack.....\n\n");
                if(ack.equals(String.valueOf(sequence))){
                    i++;System.out.println("receiver > "+" packet recieved\n\n");
                }
                else{
                    System.out.println("Time out resending data....\n\n");
                    sequence=(sequence==0)?1:0;
                }
            }
            catch(Exception e){}
        }while(i<n+1);
        System.out.println("All data sent. exiting.");
        }
        catch(Exception e){}
        finally{
            try{in.close();
            out.close();sender.close();
            }catch(Exception e){}
        }
    }
    public static void main(String args[]){
```

```
        Sender s=new Sender();
        s.run();
    }
}
```

**Receiver.java**

```
import java.io.*;
 import java.net.*;
 public class Reciever{
    ServerSocket reciever;
    Socket connection=null;
    ObjectOutputStream out;
    ObjectInputStream in;
    String packet,ack,data="";
    int i=0,sequence=0;
    Reciever(){}
    public void run(){
       try{
          BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
          reciever = new ServerSocket(2004,10);
          System.out.println("waiting for connection...");
          connection=reciever.accept();
          sequence=0;
          System.out.println("Connection established :");
          out=new ObjectOutputStream(connection.getOutputStream());
          out.flush();
          in=new ObjectInputStream(connection.getInputStream());
          out.writeObject("connected .");
          do{
             try{
                packet=(String)in.readObject();
                if(Integer.valueOf(packet.substring(0,1))==sequence){
                   data+=packet.substring(1);sequence=(sequence==0)?1:0;
                   System.out.println("\n\nreceiver >"+packet);
                }
                else{
                   System.out.println("\n\nreceiver >"+packet +" duplicate data");
                }
                if(i<3){
                   out.writeObject(String.valueOf(sequence));i++;
```

```
                }
                else{
                    out.writeObject(String.valueOf((sequence+1)%2));i=0;
                }
            }
            catch(Exception e){}
        }while(!packet.equals("end"));
      System.out.println("Data recived="+data);
      out.writeObject("connection ended .");
       }catch(Exception e){}
       finally{
          try{
             in.close();
             out.close();
             reciever.close();
          }catch(Exception e){}
       }}
    public static void main(String args[]){
       Reciever s=new Reciever();
       while(true){
        s.run();
       }
    }
 }
```

**OUTPUT:**

**Sender.java**

```
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

D:\Netbeans\JavaApplication2\src\javaapplication2>javac Sender.java

D:\Netbeans\JavaApplication2\src\javaapplication2>java Sender
Waiting for Connection....
reciver > connected .
Enter the data to send....
hii
data sent>0h
waiting for ack.....


receiver >  packet recieved


data sent>1i
waiting for ack.....
```

```
receiver >  packet recieved


data sent>0i
waiting for ack.....


receiver >  packet recieved


All data sent. exiting.

D:\Netbeans\JavaApplication2\src\javaapplication2>
```

**Receiver.java**

```
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

D:\Netbeans\JavaApplication2\src\javaapplication2>javac Reciever.java

D:\Netbeans\JavaApplication2\src\javaapplication2>java Reciever
waiting for connection...
Connection established :


receiver >0h


receiver >1i


receiver >0i
Data recived=hii
waiting for connection...
```

**RESULT:**

Thus, the program for implementing a stop and wait protocol is successfully executed and the output is verified.

**EX.NO: 10**

**DATE: 09/04/2022**

## CONNECTING TWO LANs USING BRIDGE

### AIM:

To establish connection between two LANs using Bridge.

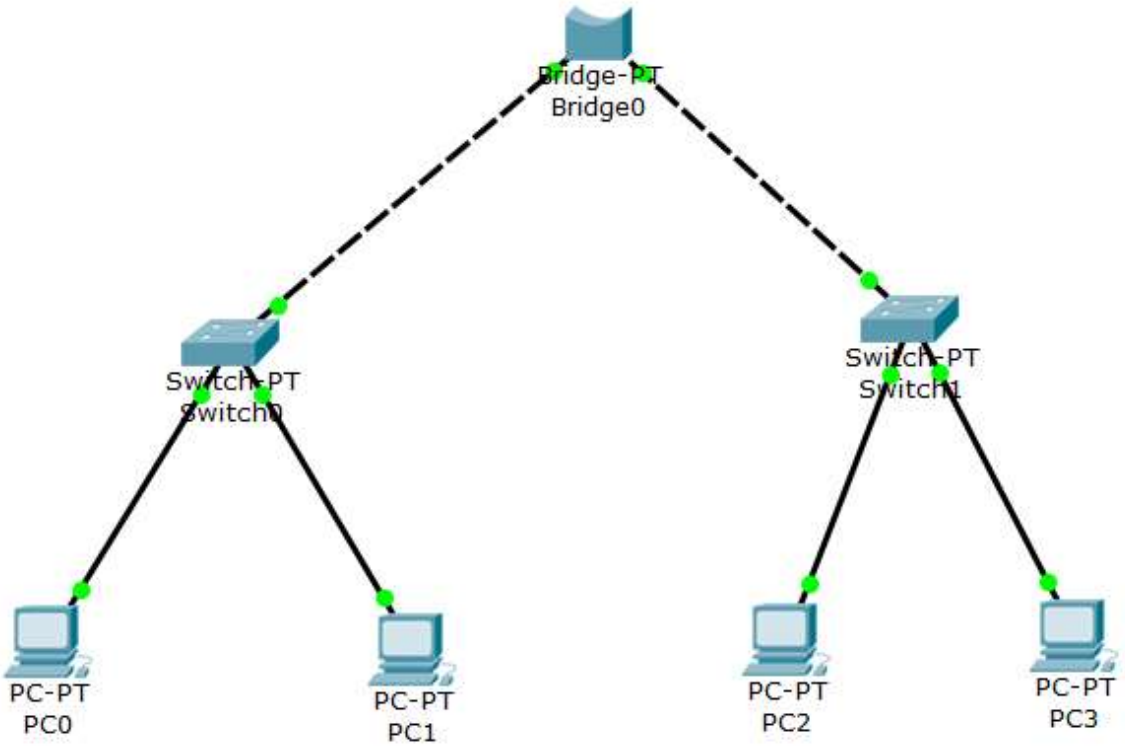
### REQUIREMENTS:

- Four windows PC or Four Linux PC.
- Two Switch - PT.
- Six Straight Line LAN (cat-5) Cables with RJ-45 Sockets.
- Power supply
- Bridge-PT
- Cisco Packet Tracer 6.0.1

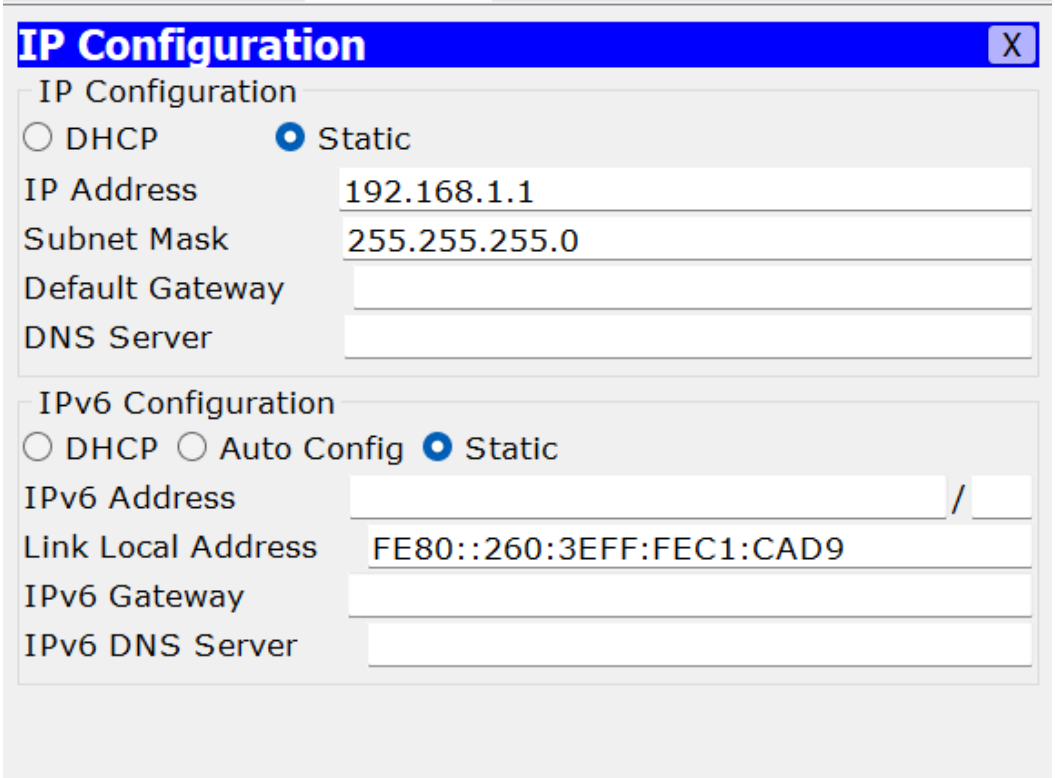
### PROCEDURES:

- Open the CISCO PACKET TRACER software.
- Draw the 4 PC using End Device Icons.
- Draw the 2 CISCO 24 Port Switch using Switch icon lists.
- Draw the Cisco Generic Bridge.
- Make the Connections using Straight-Through Ethernet Cables.
- Enter the IP Address to Each Machine.
- Check the Connections using Add Simple PDU(P)

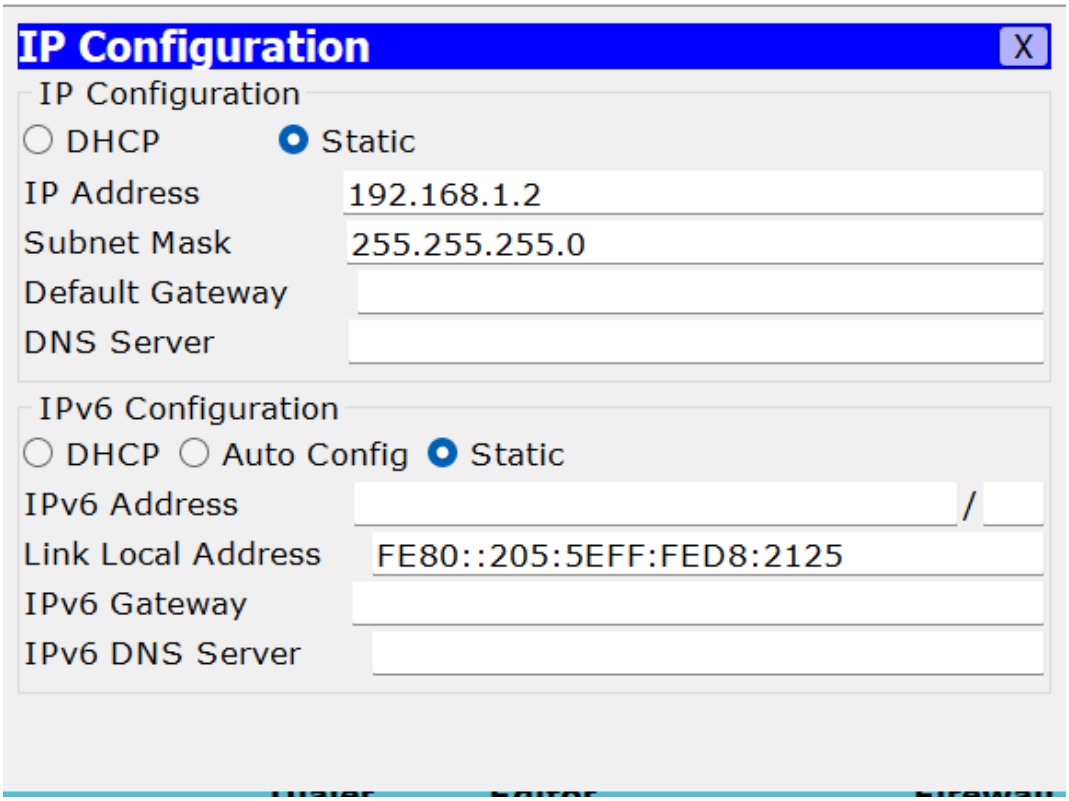
## NETWORK TOPOLOGY:



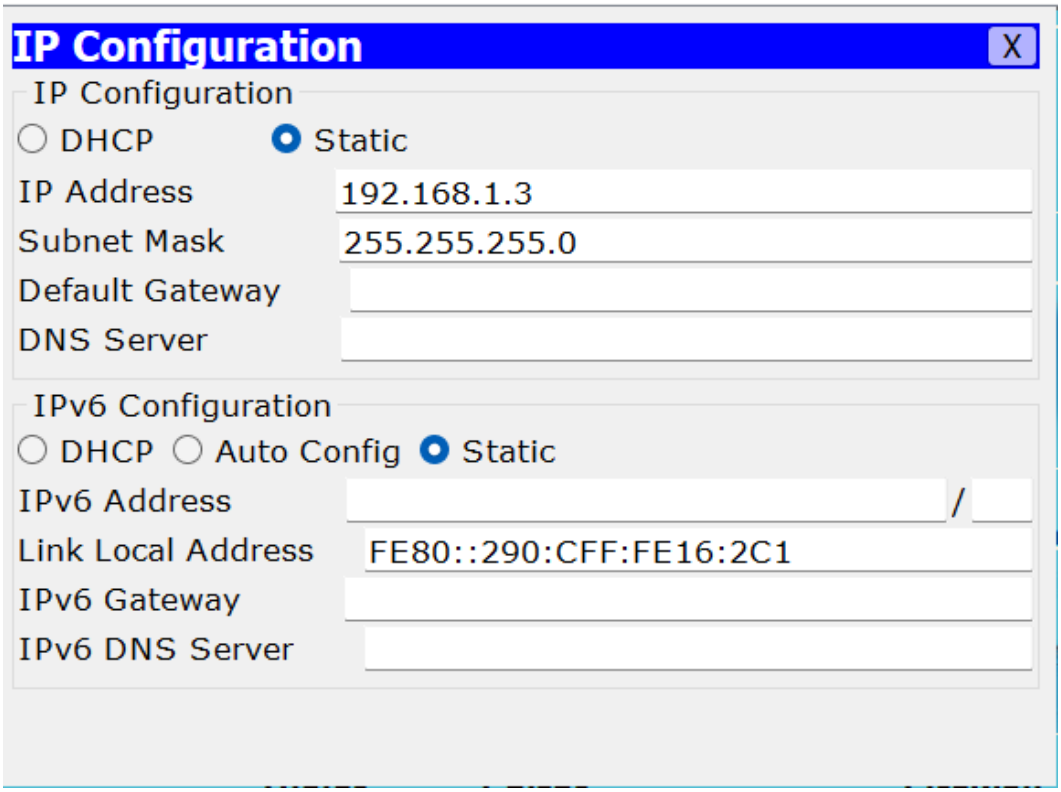

## PC0 IP CONFIGURATION ADDRESS:



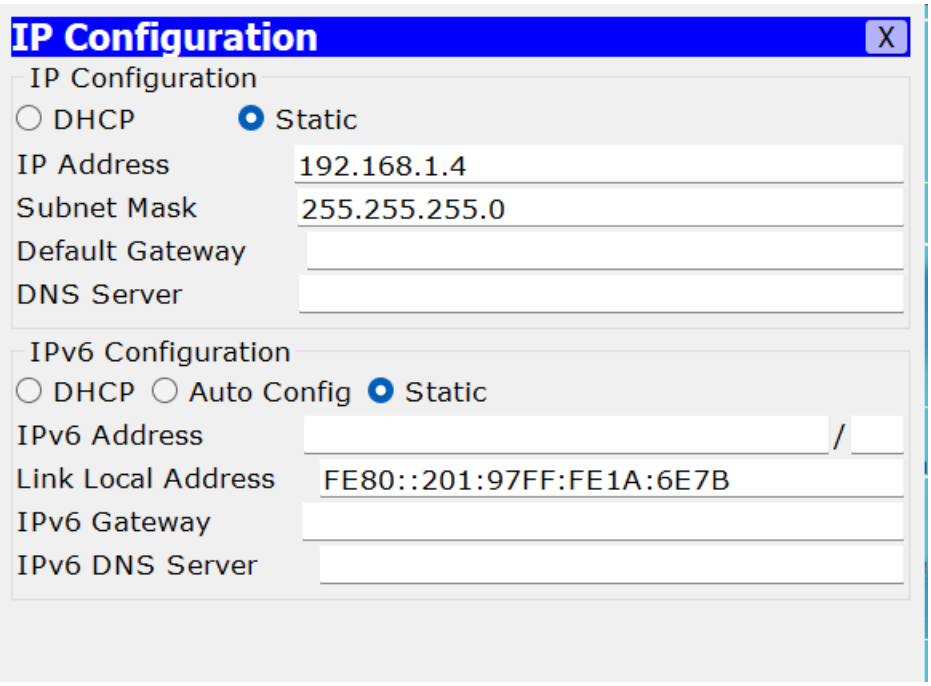
**PC1 IP CONFIGURATION ADDRESS:**




**PC2 IP CONFIGURATION ADDRESS:**



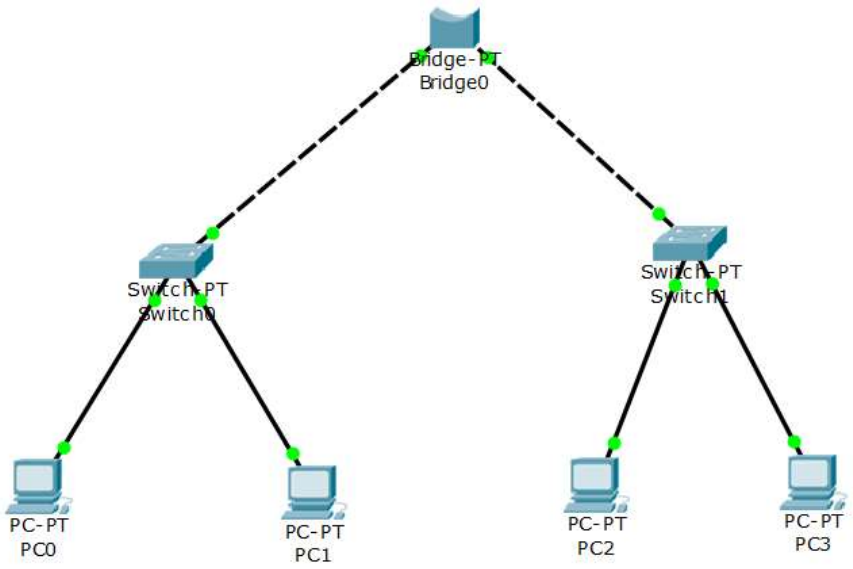
## PC3 IP CONFIGURATION ADDRESS:




## VERIFY LAN NETWORK CONNECTIVITY

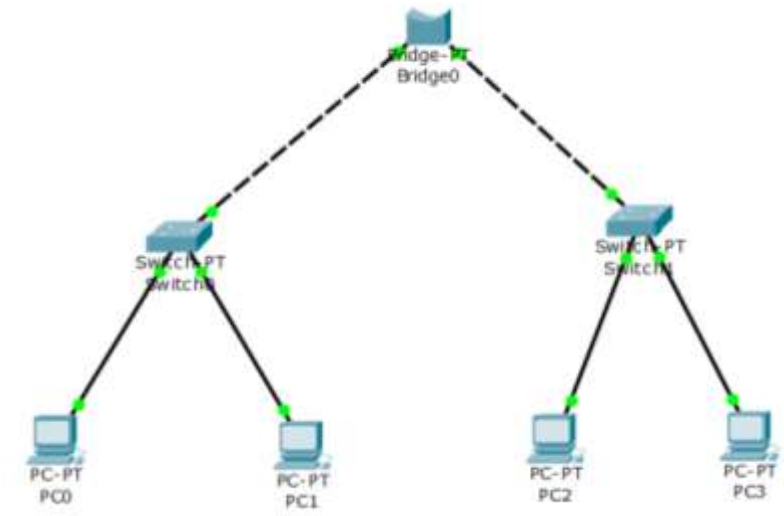
Using Add Simple PDU(p), Click the mail icon and then drop one mail to one of the PC in first LAN and another mail to PC in another LAN. If the resultant window shows the successful delivery of the mail, then network connectivity is successful.

## PC0 TO PC2



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

## PC1 TO PC3




| Fire | Last Status | Source | Destination | Type | Color | Time(se | Periodic | Num | Edit | Delete |
|---|---|---|---|---|---|---|---|---|---|---|
| | Successful | PC1 | PC3 | ICMP | | 0.000 | N | 0 | (edit) | (delete) |

## RESULT:

Thus, tow LANs are connected to the bridge and the communication between LANs is checked successfully

**EX.NO: 11**

**DATE: 19/04/2022**

## DESIGNING VARIOUS TOPOLOGIES USING CISCO PACKET TRACER

### AIM:

To Designing various topologies using cisco packet tracer.

### REQUIREMENTS:

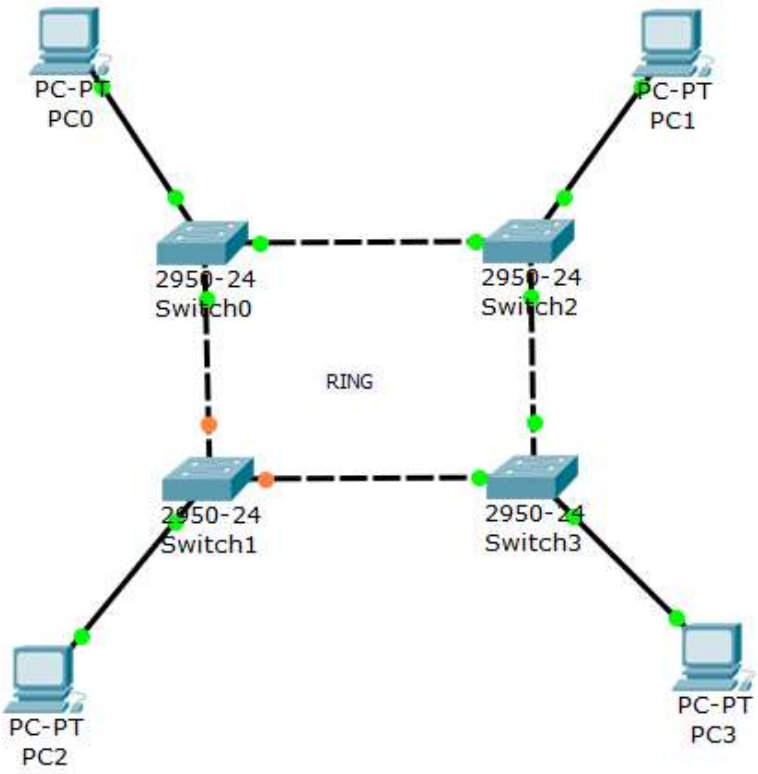
- 16 windows PC or Four Linux PC.
- 12 2950-24 Switch.
- 13 Copper Cross-over cables.
- 16 Copper Straight – Through cables.
- Power supply
- Cisco Packet Tracer 6.0.1

### PROCEDURES:

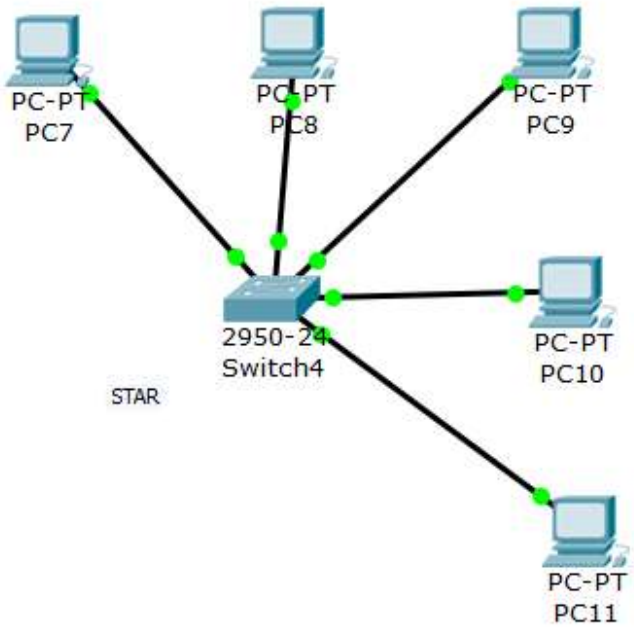
- Open the CISCO PACKET TRACER software.
- Draw the 16 PC using End Device Icons.
- Draw the 12 2950-24 Switch using Switch icon lists.
- Make the Connections using Straight-Through Ethernet Cables.
- Make the Connections between switches using Cross – Over Cables.
- Enter the IP Address to Each Machine.
- Check the Connections using Add Simple PDU(P)

## NETWORK TOPOLOGY:

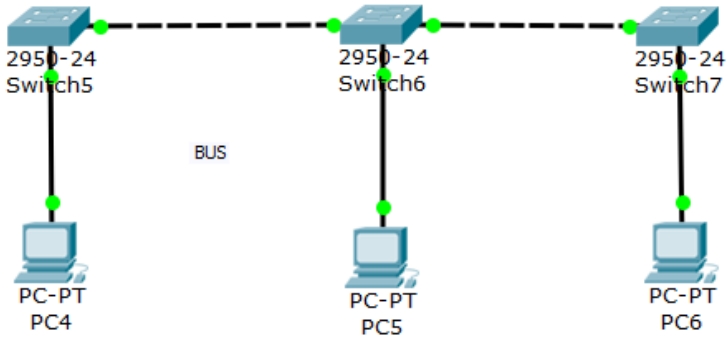
## RING TOPOLOGY:



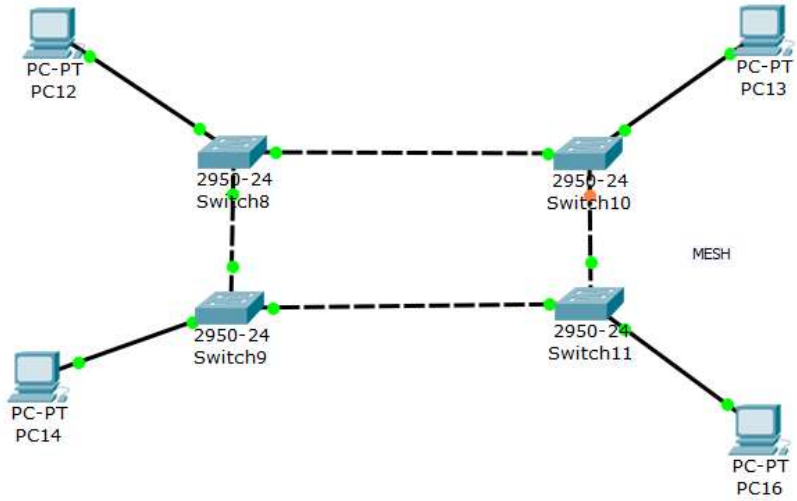

## STAR TOPOLOGY:



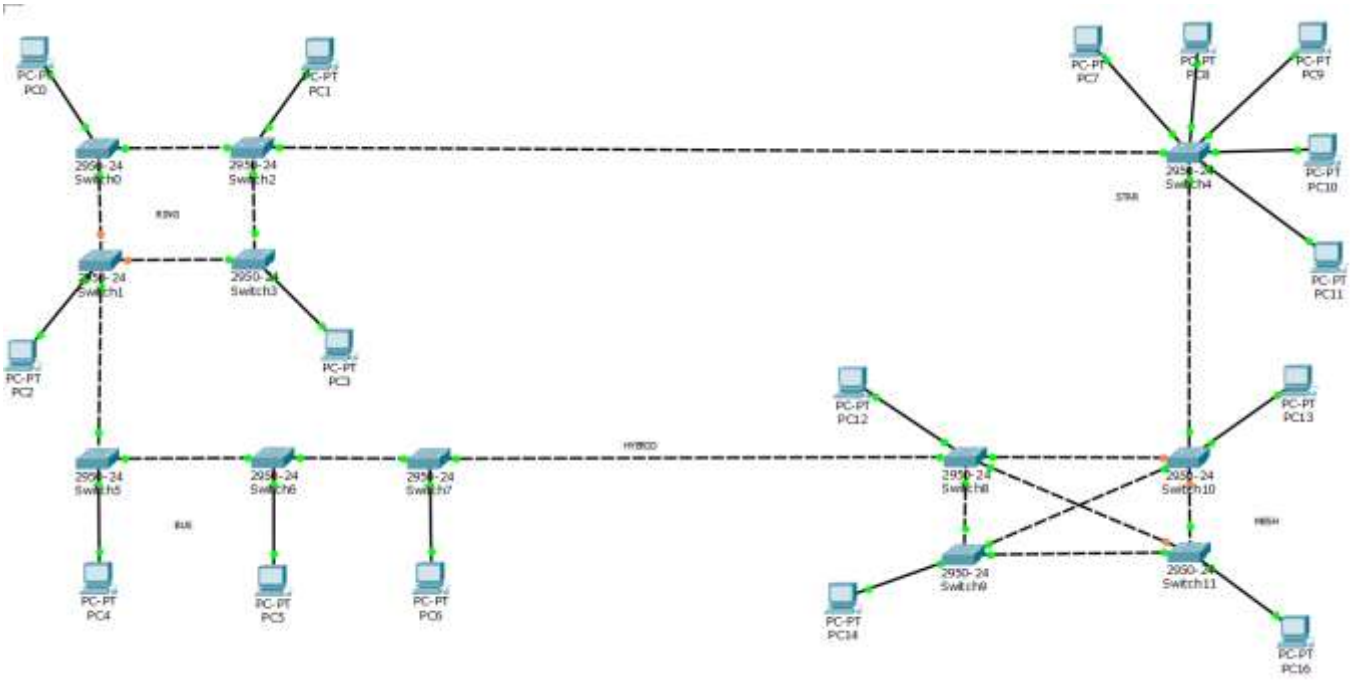
## BUS TOPOLOGY:



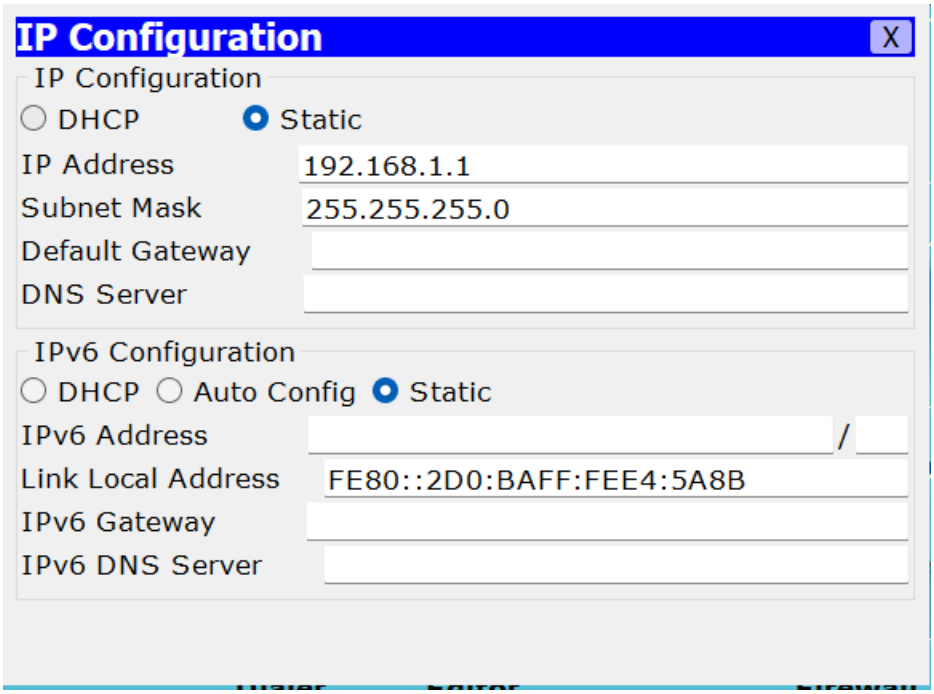

## MESH TOPOLOGY:



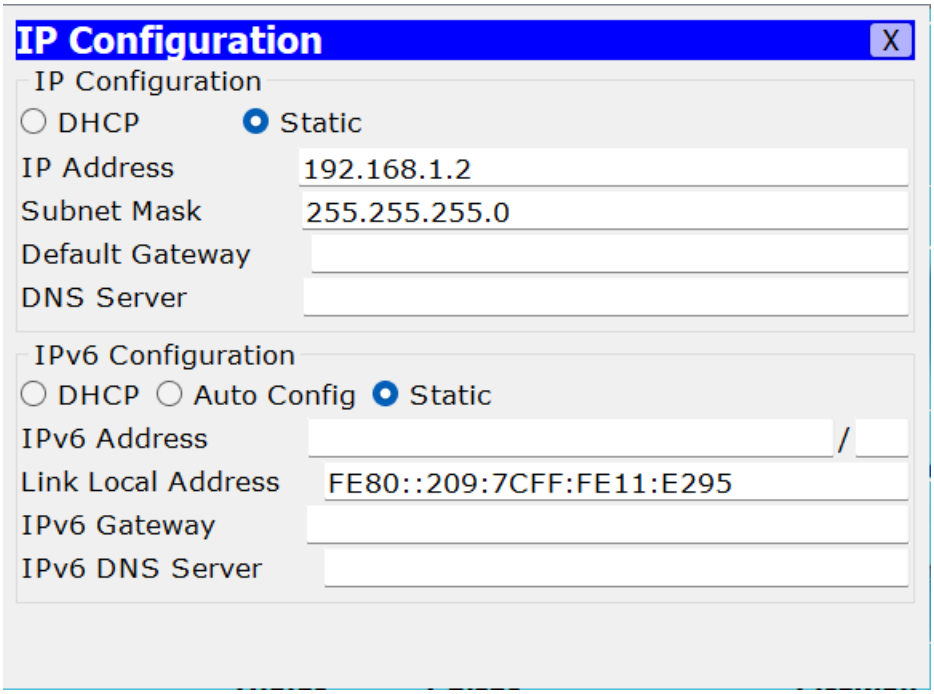

## HYBRID TOPOLOGY:



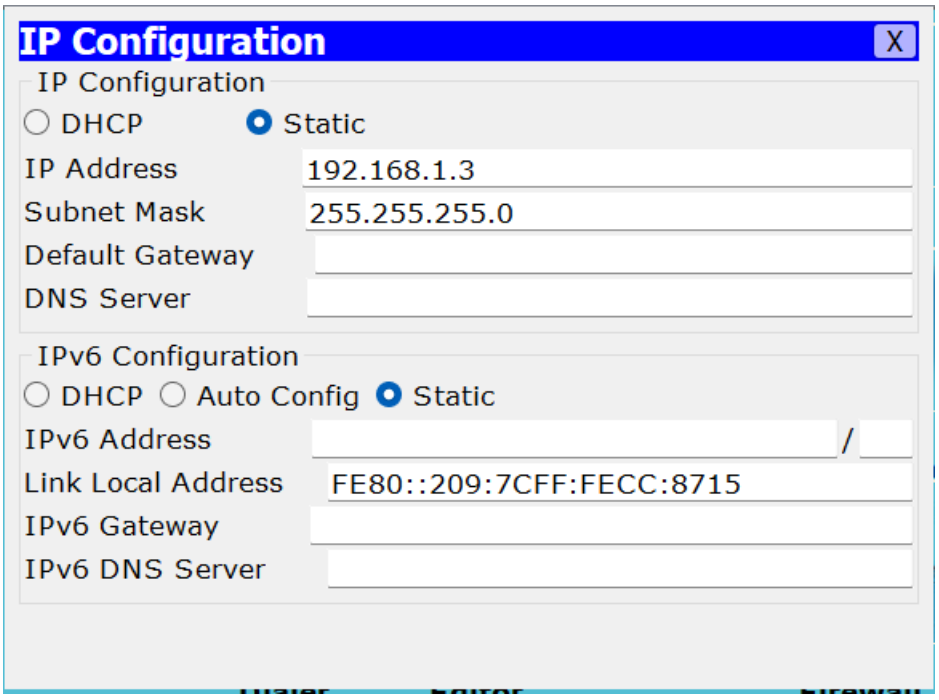
**PC0 IP CONFIGURATION ADDRESS:**



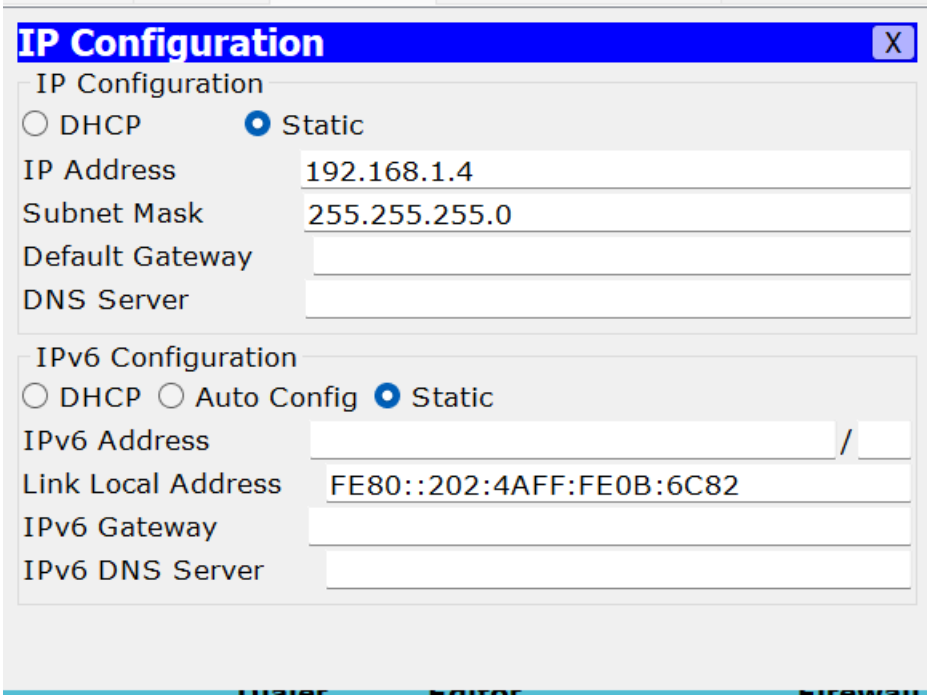

**PC1 IP CONFIGURATION ADDRESS:**



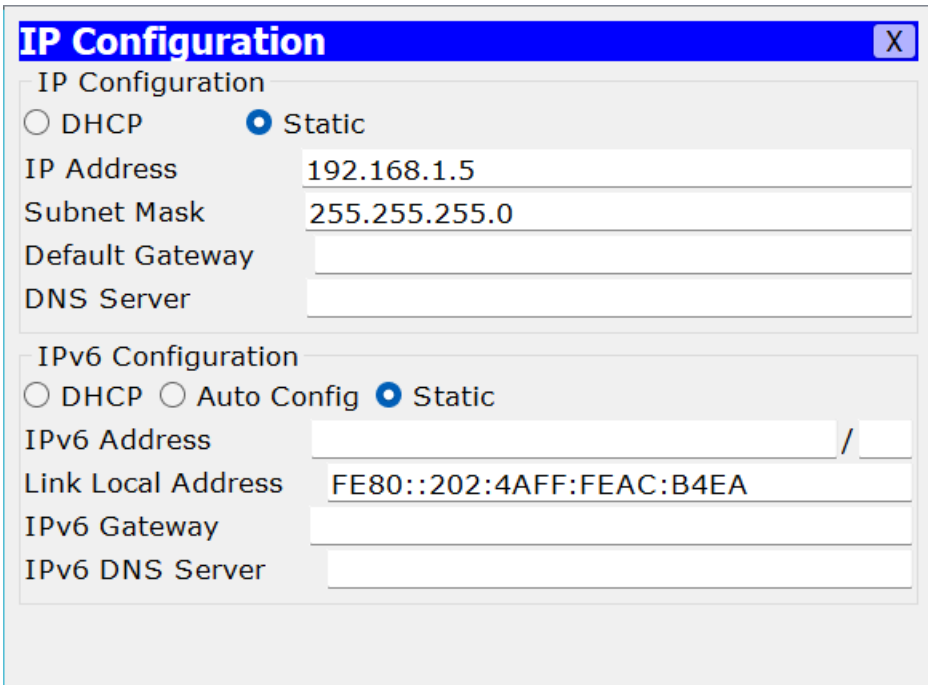
## PC2 IP CONFIGURATION ADDRESS:



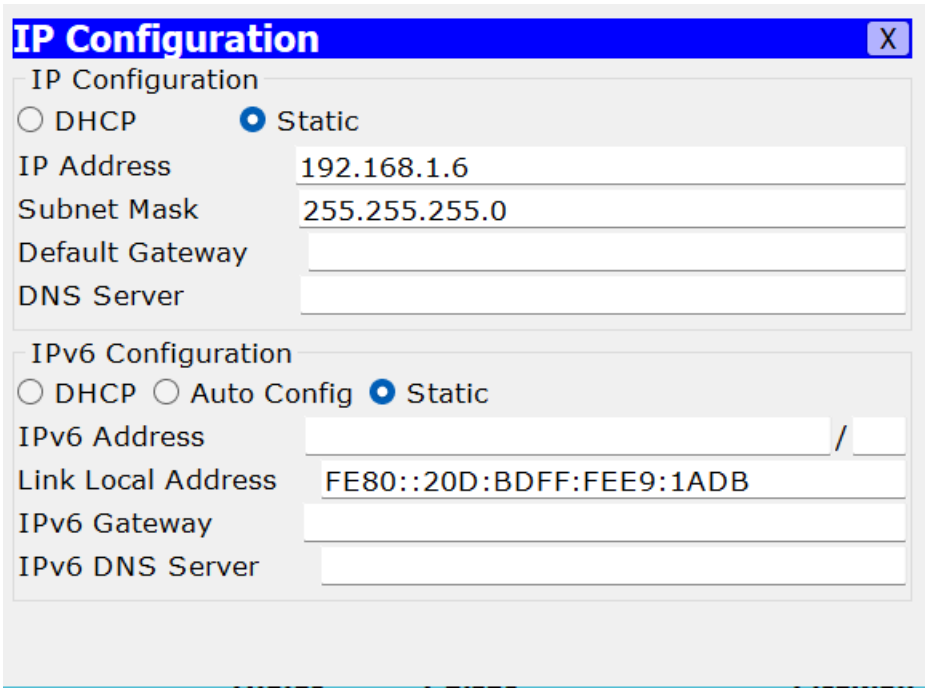

## PC3 IP CONFIGURATION ADDRESS:



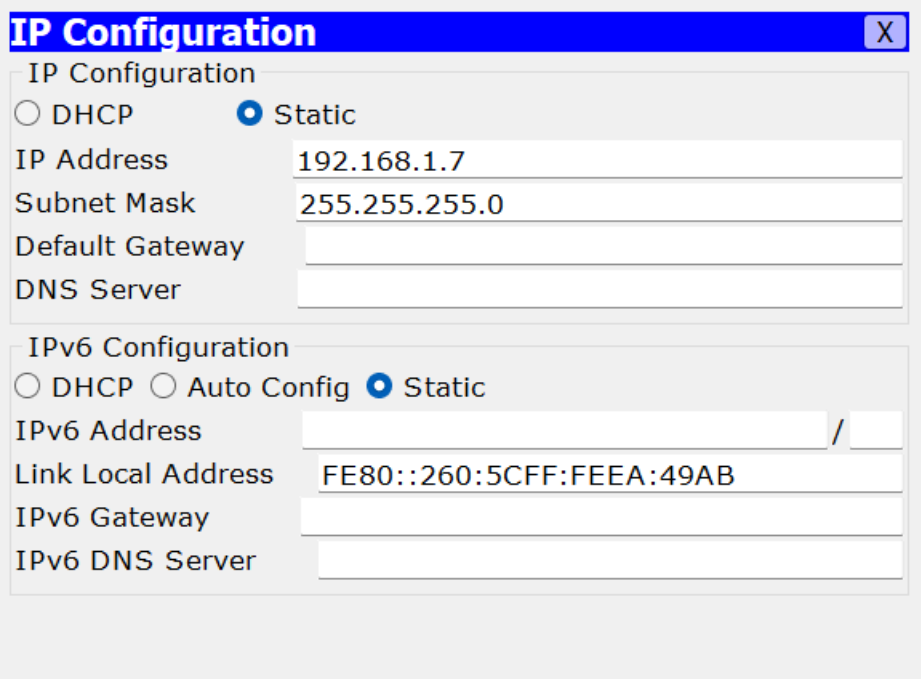
## PC4 IP CONFIGURATION ADDRESS:



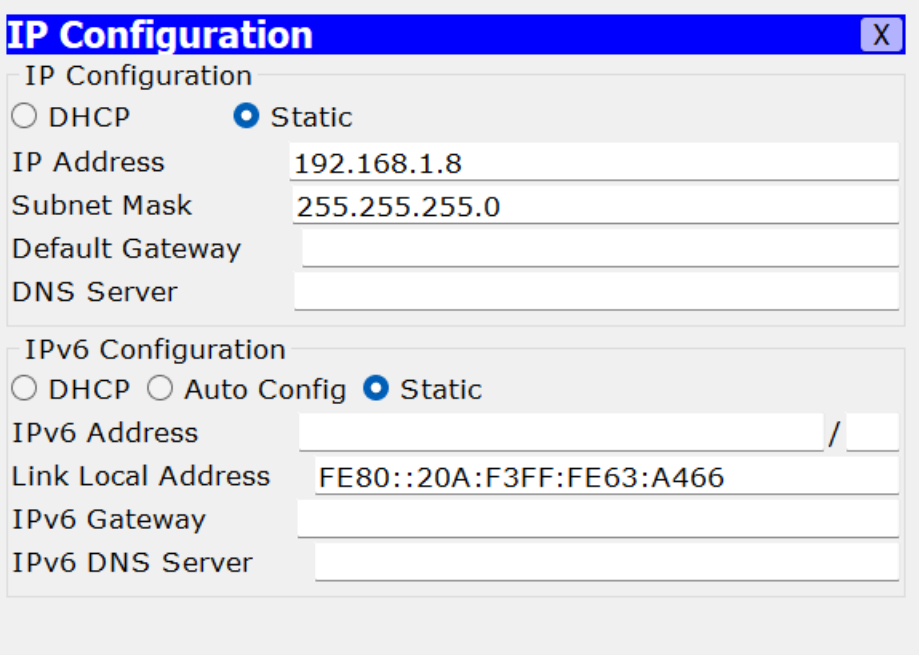

## PC5 IP CONFIGURATION ADDRESS:



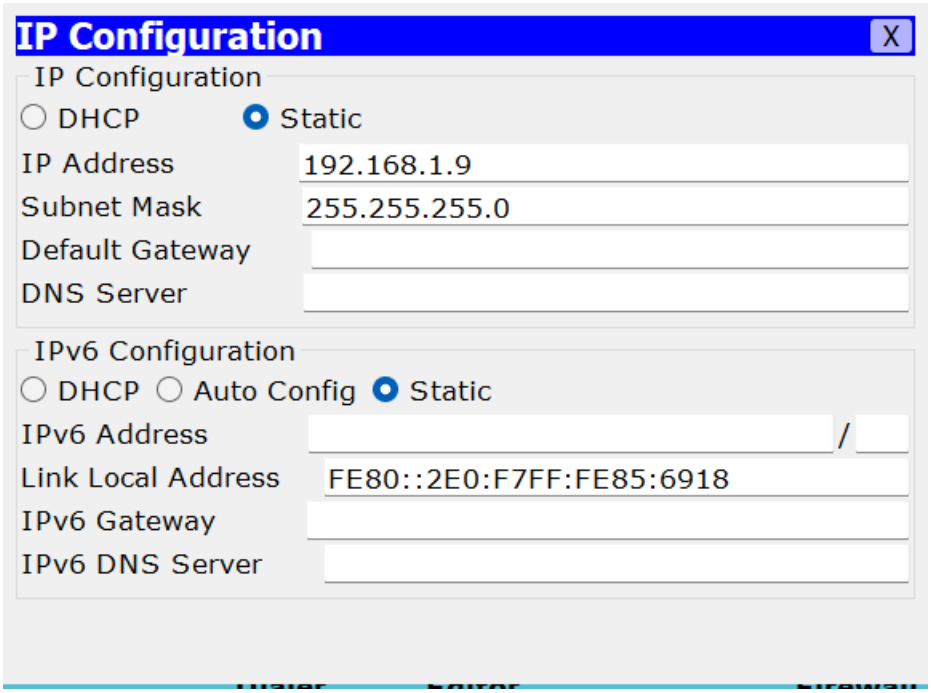
## PC6 IP CONFIGURATION ADDRESS:



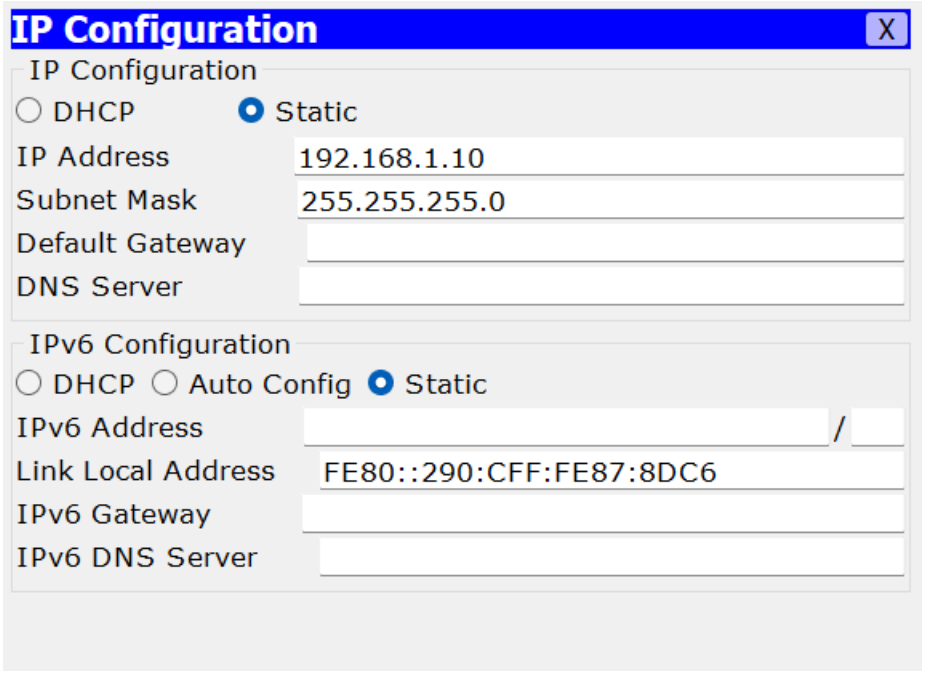

## PC7 IP CONFIGURATION ADDRESS:



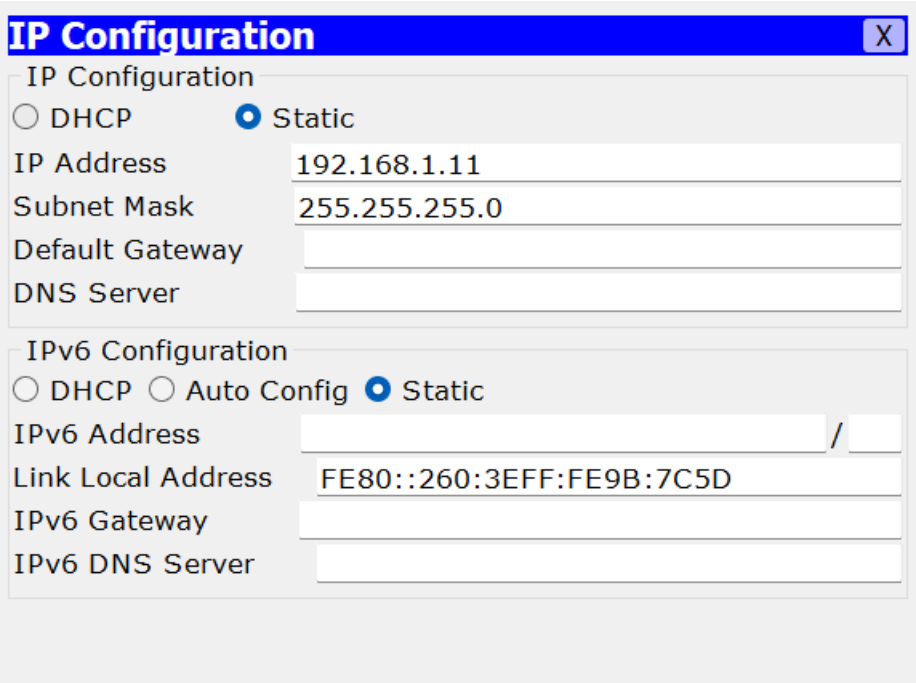
## PC8 IP CONFIGURATION ADDRESS:



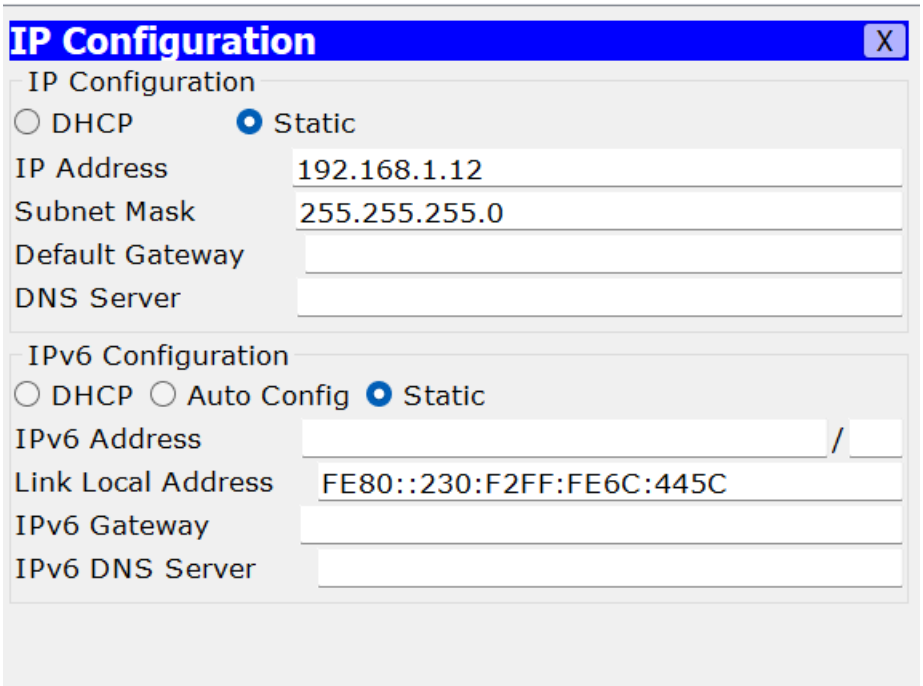

## PC9 IP CONFIGURATION ADDRESS:



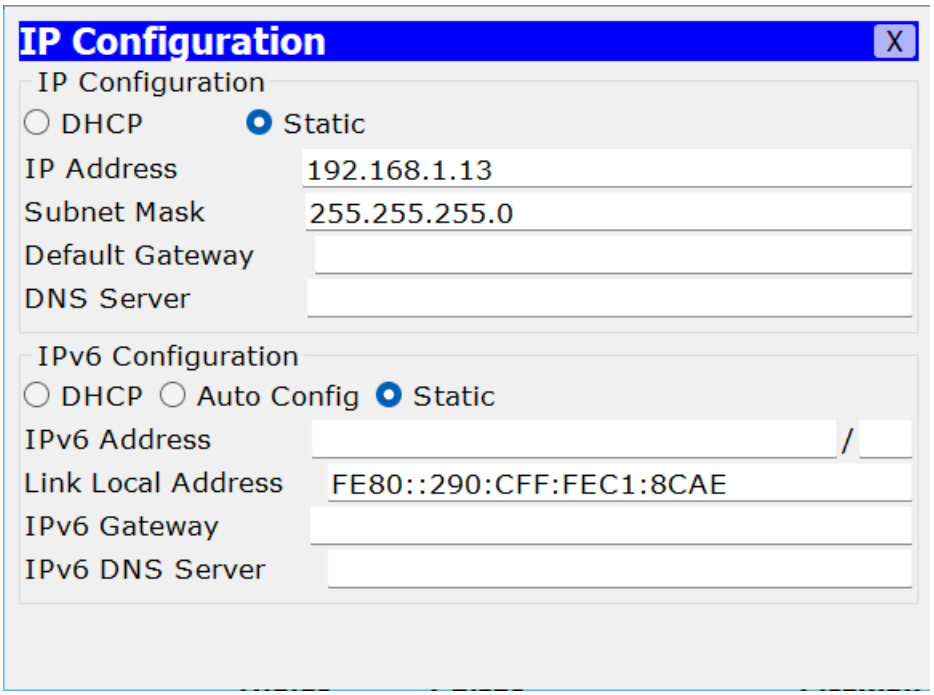
**PC10 IP CONFIGURATION ADDRESS:**



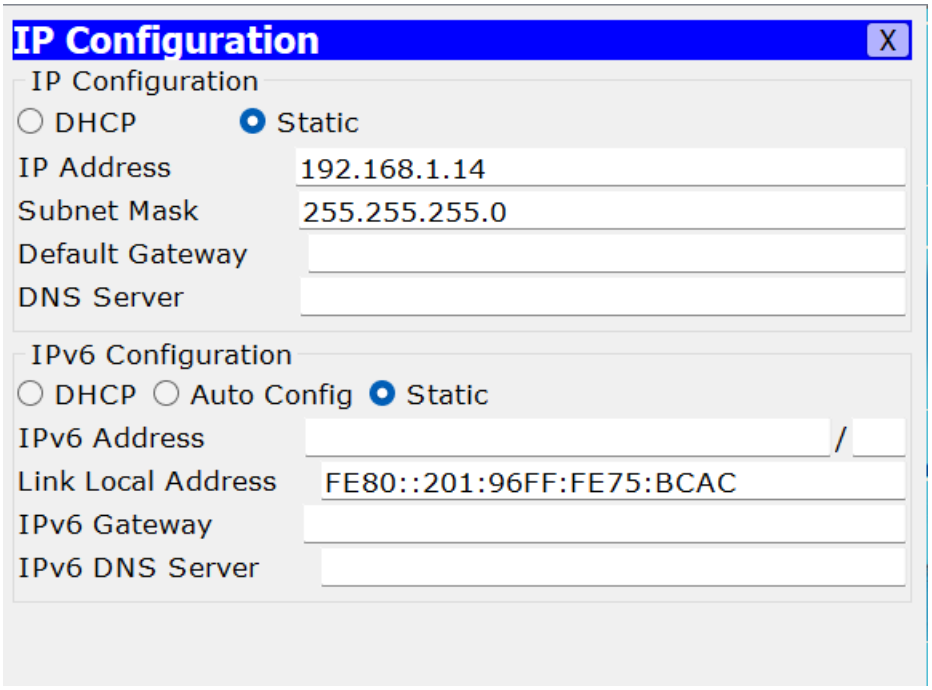

**PC11 IP CONFIGURATION ADDRESS:**



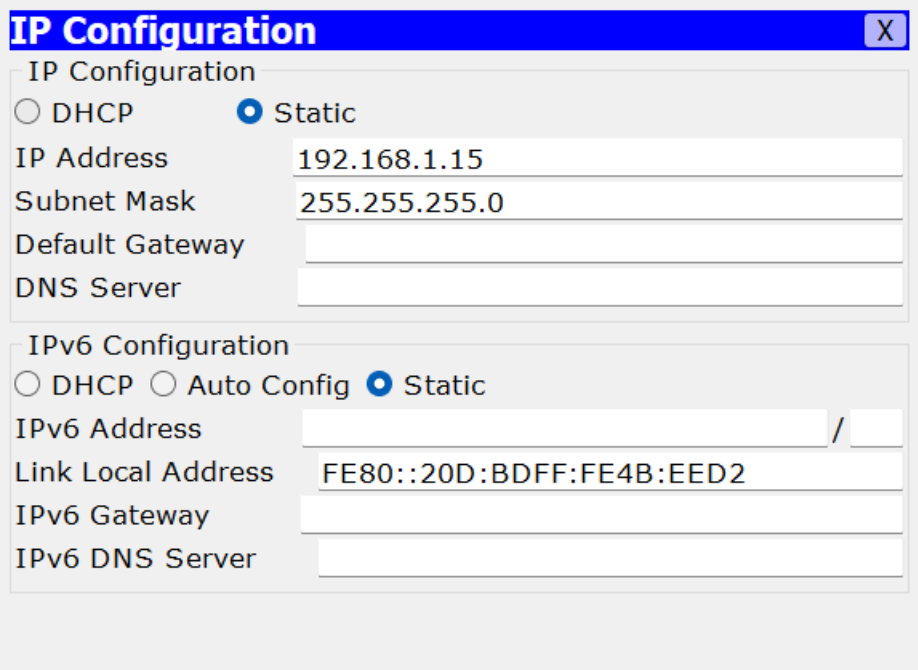
## PC12 IP CONFIGURATION ADDRESS:



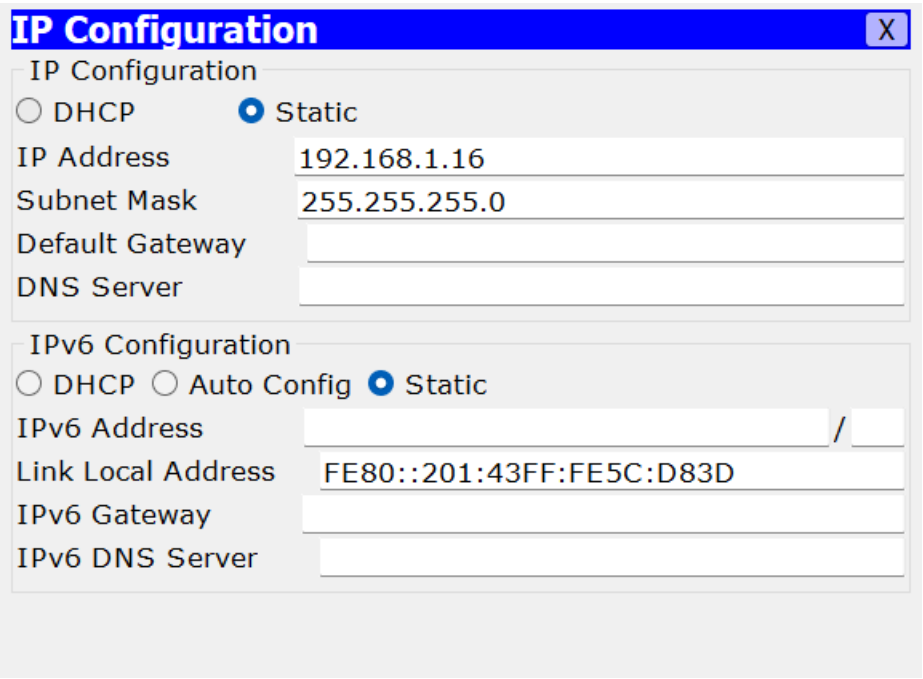

## PC13 IP CONFIGURATION ADDRESS:



## PC14 IP CONFIGURATION ADDRESS:



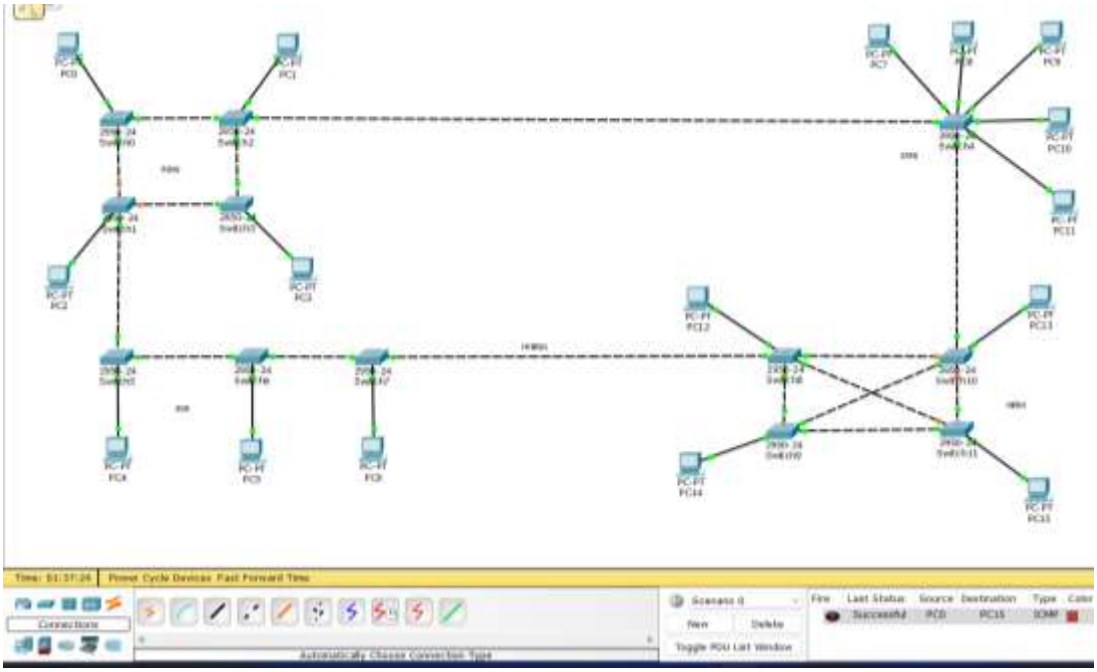

## PC15 IP CONFIGURATION ADDRESS:



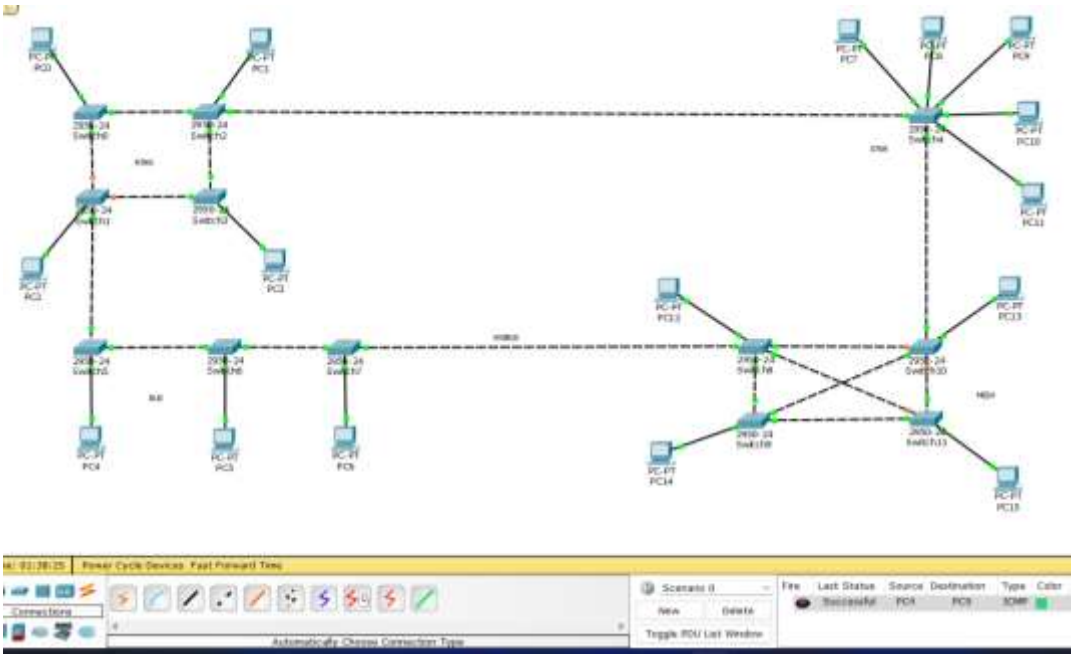
## VERIFY LAN NETWORK CONNECTIVITY

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 of the mail, then network connectivity is successful.

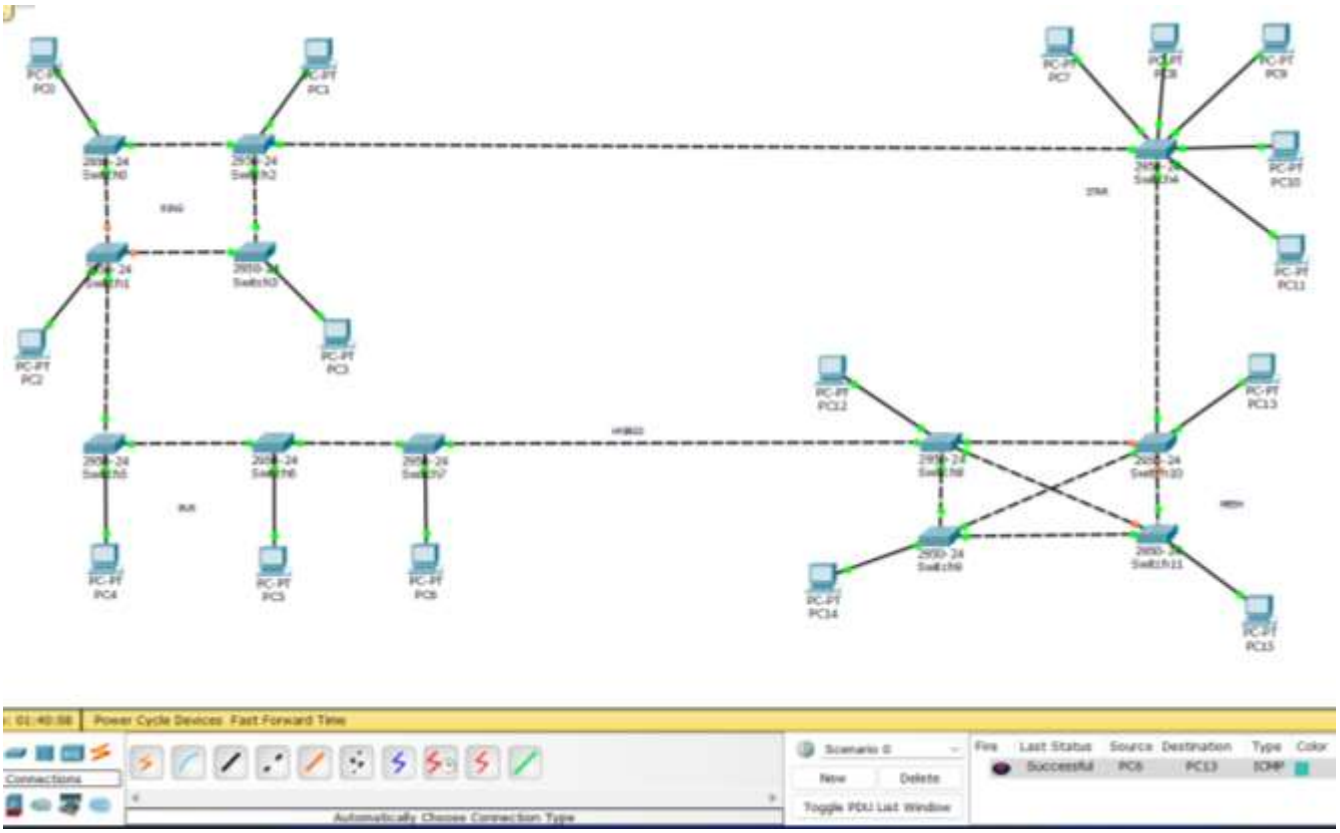
### PC0 TO PC15



### PC4 TO PC9



**PC6 TO PC13**



**RESULT:**

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