



L  
A  
B  
M  
A  
N  
U  
A  
L

# EASWARI ENGINEERING COLLEGE

(An Autonomous Institution)

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

### 231CSC511L – COMPUTER NETWORKS LABORATORY

#### MANUAL



III YEAR  
2025-2026

PREPARED BY

Mrs.A.Jeba Sheela,  
Mrs.M.Gowthami,  
Mrs.Fatima Vincy R

APPROVED BY

HOD/CSE

231CSC511L  Regulation - R2023_V1.0	<b>COMPUTER NETWORKS LABORATORY</b>  (Common to CSE (AI & ML), CSE, CSE (CS), IT)	Periods per week				Credits
		L	T	P	R	
		0	0	3	1	2

### SCHEME OF EXAMINATION

Duration of End Semester Examination in Hours	Marks			Minimum marks for Pass	
	Continuous assessment Examination	End Semester Examination	Maximum marks	End Semester Examination	Total
3	60	40	100	45	50

### COURSE OBJECTIVES:

1.	To learn and use network commands
2.	To learn socket programming.
3.	To implement and analyze various network protocols
4.	To learn and use simulation tools.
5.	To use simulation tools to analyze the performance of various network protocols.

### COURSE OUTCOMES:

Upon completion of this course, student will be able to:	Bloom's level
CO1: Apply networking commands for various operating systems.	K3
CO2: Implement various protocols using TCP / UDP sockets.	K3
CO3: Implement Cyclic Redundancy Check for error detection and correction	K3
CO4: Analyze the performance of various network protocols using simulation tools.	K4
CO5: Evaluate the various routing algorithms for finding optimal path.	K5

### MAPPING OF COURSE OUTCOMES (CO) WITH PROGRAMME OUTCOME (PO)

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	3	3	2	3	-	-	-	3	3	3	3
CO2	3	2	3	2	3	-	-	-	3	3	3	3
CO3	2	3	2	2	3	-	-	-	3	2	3	2
CO4	3	3	3	3	3	-	-	-	2	2	3	3
CO5	2	3	3	3	3	-	-	-	2	2	3	3

3 – High : 2 - Medium : 1 – Low : ‘-’ - No correlation

<b>LIST OF EXPERIMENTS</b>	
1.	Configuration of network components like router, hub, switch using simulators
2.	Write a program to implement socket programming.
3.	Write a HTTP web client program to download a web page using TCP / UDP sockets.
4.	Write a program to implement DNS using TCP / UDP sockets
5.	Write a program to implement Echo client and echo server and chat application using Transport layer protocol.
6.	Implementation of File Transfer using TCP / UDP.
7.	Network packet analysis using tools like Wireshark, tcpdump.
8.	Simulation of Congestion / flow control Algorithms using NS
9.	Performance of TCP and UDP using Simulation tool.
10.	Simulation of Distance Vector and Link State Routing algorithm.
11.	Implementation of IPv4 and IPv6
12.	Implementation of SMTP
13.	Implementation of error correction code.
14.	Implementation of RPC using simulator.

**TOTAL PERIODS:** **60**

#### **CONTENT BEYOND SYLLABUS**

1.	Running commands – Trace route, ARP, TelNet, FTP, DHCP – DNS configurations.
2.	Network packet analysis using tools like Wireshark, tcpdump.

#### **LIST OF EQUIPMENTS:**

<b>1. HARDWARE:</b>
Standalone desktops.
<b>2. SOFTWARE</b>
• Network simulator like NS2/Glomosim/OPNET/ Packet Tracer / Equivalent /C / Java / Python

#### **COURSE DESIGNERS**

1.	Dr K P K Devan	Assistant Professor	Computer Science and Engineering
2.	D Kavitha	Assistant Professor	Computer Science and Engineering

<b>Recommended by Board of Studies</b>	Date: 10.05.2024	Syllabus version	1
<b>Approved by the Academic Council</b>	Date: 22.07.2024	Meeting No.	7



FORMAT NO: LP 01

ISSUE NO: 2

ISSUE DATE: 28.01.12

**EASWARI ENGINEERING COLLEGE**  
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**COURSE PLAN**  
**Regulation – 2023**

Course/Branch	:	<b>B.E / CSE</b>	<b>Total no. of hours given in syllabus:</b>
Subject Code	:	<b>231CSC511L</b>	
Subject Title	:	Computer Networks Laboratory	
Year/Semester	:	III/V	
Faculty Name	:	Mrs. A.Jeba Sheela	

**COURSE OBJECTIVES:**

**The student should be made to:**

- To learn and use network commands.
- To learn socket programming.
- To implement and analyze various network protocols.
- To learn and use simulation tools.
- To use simulation tools to analyze the performance of various network protocols.

<b>Ex. No.</b>	<b>Experiment Name</b>	<b>Allotted hours</b>	<b>CO Mapping</b>
1	Configuration of network components like router, hub, switch using simulators	4 Hours	CO1
2	Write a program to implement socket programming.	4 Hours	CO2
3	Write a HTTP web client program to download a web page using TCP / UDP sockets.	4 Hours	CO5
4	Write a program to implement DNS using TCP / UDP sockets	4 Hours	CO4
5	Write a program to implement Echo client and echo server and chat application using Transport layer protocol.	4 Hours	CO4
6	Implementation of File Transfer using TCP / UDP.	4 Hours	CO5

7	Network packet analysis using tools like Wireshark, tcpdump.	4 Hours	CO3
8	Simulation of Congestion / flow control Algorithms using NS	4 Hours	CO2
9	Performance of TCP and UDP using Simulation tool.	4 Hours	CO4
10	Simulation of Distance Vector and Link State Routing algorithm.	4 Hours	CO3
11	Implementation of IPv4 and IPv6	4 Hours	CO3
12	Implementation of SMTP	4 Hours	CO5
13	Implementation of error correction code.	4 Hours	CO2
14	Implementation of RPC using simulator.	4 Hours	CO5
<b>Content Beyond Syllabus</b>			
1.	Running commands – Trace route, ARP, TelNet, FTP, DHCP – DNS configurations.	2 Hours	CO5

#### **COURSE OUTCOMES:**

Upon completion of this course, student will be able to:

CO1: Apply networking commands for various operating systems.

CO2: Implement various protocols using TCP / UDP sockets.

CO3: Implement Cyclic Redundancy Check for error detection and correction.

CO4: Analyse the performance of various network protocols using simulation tools.

CO5: Evaluate the various routing algorithms for finding optimal path.

#### **LABORATORY REQUIREMENT FOR BATCH OF 30 STUDENTS:**

##### **HARDWARE:**

1. Standalone desktops 30 Nos

##### **SOFTWARE:**

1. C / C++ / Java / Python / Equivalent Compiler 30
2. Network simulator like NS2/GLOMOSIM/OPNET/ Packet Tracer / Equivalent

**FACULTY INCHARGE**

**HOD/CSE**

## **Program Outcomes (PO)**

- [1] **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- [2] **Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- [3] **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- [4] **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- [5] **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- [6] **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- [7] **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- [8] **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- [9] **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- [10] **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- [11] **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- [12] **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

## **Program Specific Outcomes (PSO)**

- [1] To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.
- [2] To apply software engineering principles and practices for developing quality software for scientific and business applications.
- [3] To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems.

**EASWARI ENGINEERING COLLEGE**  
**(An Autonomous Institution)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**231CSC511L – COMPUTER NETWORKS LABORATORY**

**INDEX**

<b>EXP.NO</b>	<b>DATE</b>	<b>TITLE</b>	<b>PAGE NUMBER</b>	<b>MARKS</b>	<b>SIGN</b>
1		Configuration of network components like router, hub, switch using simulators			
2		Write a program to implement socket programming.			
3		Write a HTTP web client program to download a web page using TCP / UDP sockets.			
4		Write a program to implement DNS using TCP / UDP sockets			
5a		Write a program to implement Echo client and echo server using Transport layer protocol.			
5b		Write a program to implement chat application using Transport layer protocol.			
6		Implementation of File Transfer using TCP / UDP.			
7		Network packet analysis using tools like Wireshark, tcpdump.			
8		Simulation of Congestion / flow control Algorithms using NS			
9		Performance of TCP and UDP using Simulation tool.			
10		Simulation of Distance Vector and Link State Routing algorithm.			
11		Implementation of IPv4 and IPv6			

12		Implementation of SMTP			
13		Implementation of error correction code.			
14		Implementation of RPC using simulator.			
<b>CONTENT BEYOND SYLLABUS</b>					
15		Running commands – Trace route, ARP, TelNet, FTP, DHCP – DNS configurations.			

**Ex.No:1**

## **CONFIGURATION OF NETWORK COMPONENTS LIKE ROUTER, HUB, SWITCH USING SIMULATORS**

**Aim:**

To perform configuration of network components like router, hub, switch using CISCO packet tracer.

**Steps:**

**1. Setting Up the Simulator**

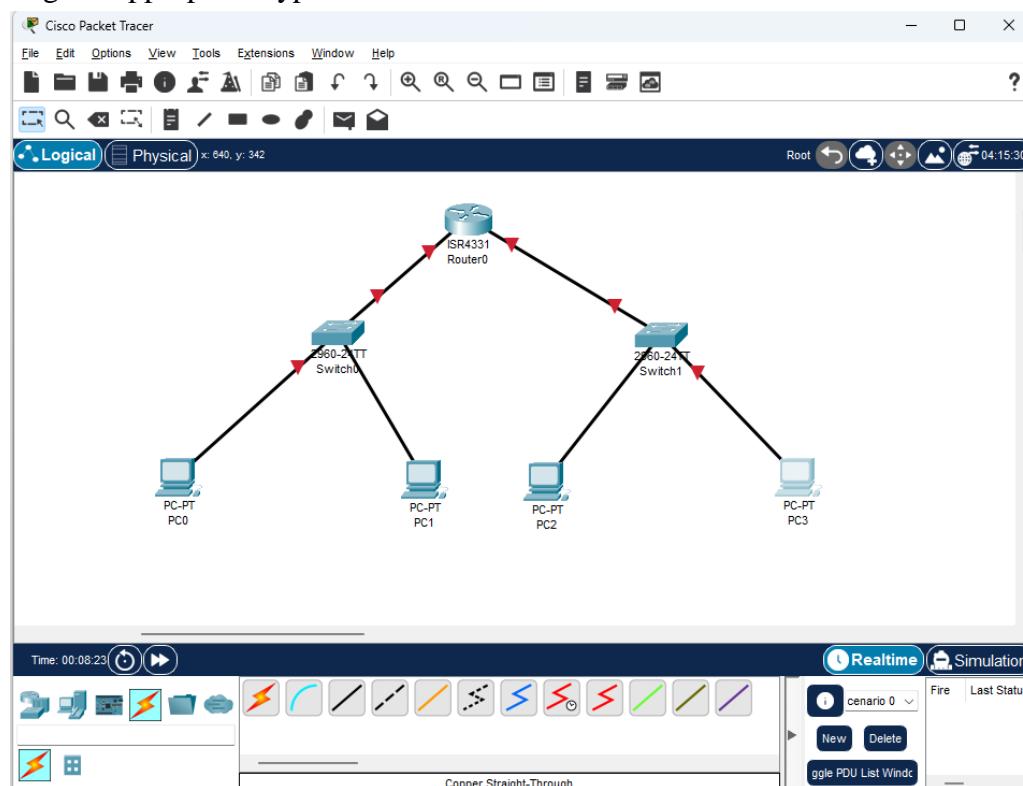
**Install the Simulator:** Download and install a network simulator like Cisco Packet Tracer.

**Create a New Workspace:** Open the application and start a new project or workspace.

**2. Adding Network Components**

**Select Devices:** Drag and drop the required components like routers, switches, hubs, PCs, into the workspace.

**Interconnect Devices:** Use cables (crossover or straight-through) to connect devices, ensuring the appropriate type is used based on the connection.



Connection	Cable Type
PC to Switch	Straight - through
Switch to Router	Straight - through
Router to Hub	Straight - through
Hub to PC	Straight - through

Ensure green dots appear on all connections.

### 3. Configuring Routers

**Access the CLI:** Open the Command-Line Interface (CLI) of the router within the simulator.

#### Basic Configuration:

##### Assign host name

*Router>enable*

*Router#configure terminal*

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

*Router(config)#hostname JSrouter*

#### Configure Interface

*JSrouter(config)#interface gigabitEthernet 0/0/0*

*JSrouter(config-if)#ip address 192.168.1.1 255.255.255.0*

*JSrouter(config-if)#no shutdown*

*JSrouter(config-if)#*

*%LINK-5-CHANGED: Interface GigabitEthernet0/0/0, changed state to up*

*%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0/0, changed state to up*

*JSrouter(config)#interface gigabitEthernet 0/0/1*

*JSrouter (config-if)# ip address 192.168.2.1 255.255.255.0*

*JSrouter (config-if)# no shutdown*

*JSrouter (config-if)# exit*

*JSrouter (config)# exit*

*JSrouter # copy running-config startup-config*

#### Configure PCs

1. Click each PC - > Desktop tab -> IP Configuration.
2. Assign IP address and subnet mask as in the table.

Device	Interface	IP Address	Subnet Mask
PC0	FastEthernet0	192.168.1.2	255.255.255.0
Router	G0/0	192.168.1.1	255.255.255.0
Router	G0/1	192.168.2.1	255.255.255.0
PC1	FastEthernet0	192.168.2.2	255.255.255.0
PC2	FastEthernet0	192.168.2.3	255.255.255.0

### 3. Set Default Gateway:

For PC0 -> 192.168.1.1

For PC1 & PC2 -> 192.168.2.1

### Configure switch settings

- Click the switch → CLI tab.
- Enter privileged mode:

**Switch> enable**

**Switch#**

Enter global configuration:

**Switch# configure terminal**

**Switch(config)#**

Suppose you want to set **fast Ethernet 0/1** as an access port in VLAN 1 (default):

**Switch(config)# interface fastEthernet 0/1**

**Switch(config-if)# switchport mode access**

**Switch(config-if)# switchport access vlan 1**

**Switch(config-if)# no shutdown**

**Switch(config-if)# exit**

For the port connecting to the hub (say Fa0/2):

**Switch(config)# interface fastEthernet 0/2**

**Switch(config-if)# switchport mode access**

**Switch(config-if)# no shutdown**

**Switch(config-if)# exit**

Disable unused ports:

**Switch(config)# interface range fastEthernet 0/3 – 0/24**

**Switch(config-if-range)# shutdown**

**Switch(config-if-range)# exit**

Exit and save:

**Switch(config)# end**

**Switch# write memory**

**Verify Connectivity**

- Use Command Prompt in any PC.
- Type:  
Ping 192.168.2.2  
Ping 192.168.1.1  
Ping 192.168.2.3

**Result:**

Thus the configuration of network components like router, hub, switch using CISCO packet tracer was done and verified successfully.

**Ex.No: 2**

**PROGRAM TO IMPLEMENT SOCKET PROGRAMMING**

**Aim**

To write a java program to implement socket programming.

**Algorithm**

**Client Side**

1. Start the program.
2. Create a socket which binds the IP address of server and the port address to acquire service.
3. After establishing connection send a data to server.
4. Receive and print the same data from server.
5. Close the socket.
6. End the program.

**Server Side**

1. Start the program.
2. Create a server socket to activate the port address.
3. Create a socket for the server socket which accepts the connection.
4. After establishing connection receive the data from client.
5. Print and send the same data to client.
6. Close the socket.
7. End the program.

**Program**

**SServer.java**

```
import java.net.*;
import java.io.*;

public class SServer {
    public static void main(String args[]) {
        try {
            ServerSocket ss = new ServerSocket(6000);
            Socket s = ss.accept();
            DataInputStream dis = new DataInputStream(s.getInputStream());
```

```
        String str = (String) dis.readUTF();
        System.out.println("message: " + str);
        ss.close();
    } catch (Exception ex) {
        System.out.println(ex);
    }
}
```

### SClient.java

```
import java.net.*;
import java.io.*;
import java.util.Scanner;

public class SClient {
    public static void main(String args[]) {
        try {
            Scanner sc = new Scanner(System.in);
            Socket s = new Socket("localhost", 6000);
            DataOutputStream dout = new DataOutputStream(s.getOutputStream());
            System.out.print("Enter message: ");
            String msg = sc.nextLine();
            dout.writeUTF(msg);
            dout.flush();
            dout.close();
            s.close();
            sc.close();
        } catch (Exception ex) {
            System.out.println(ex);
        }
    }
}
```

### Output

```
C:\Users\reddy\.jdks\openjdk-16.0.1\bin\java.exe "-javaagent
```

```
Enter message: Hello
```

```
Process finished with exit code 0
```

```
C:\Users\reddy\.jdks\openjdk-16.0.1\bin\java.exe "-javaagent
```

```
Message: Hello
```

```
Process finished with exit code 0
```

## **Result**

Thus the program for socket implementation was written and executed.

## **Ex.No: 3**

### **HTTP WEB CLIENT PROGRAM TO DOWNLOAD A WEB PAGE USING TCP / UDP SOCKETS**

#### **Aim**

To Write a HTTP web client program to download a web page using TCP / UDP sockets.

#### **Algorithm**

1. Start the program.
2. Import all the necessary packages
3. Create an object for URL class and pass the Web page URL to download.
4. Create input Stream object and store the content in stream
5. Receive and print the same data from the stream.
6. End the program.

#### **Program**

##### **WebPage.java**

```
import java.io.*;
import java.util.Scanner;
import java.net.*;

public class WebPage {
    public static void main(String[] args) {
        URL url;
        InputStream is = null;
        BufferedReader br;
        String line;
        try {
            Scanner sc = new Scanner(System.in);
```

```
System.out.print("Enter the url [eg: www.google.com] : ");
String msg = sc.nextLine();
url = new URL("https://" + msg);
is = url.openStream(); // throws an IOException
br = new BufferedReader(new InputStreamReader(is));

while ((line = br.readLine()) != null) {
    System.out.println(line);
}

} catch (MalformedURLException mue) {
    mue.printStackTrace();
} catch (IOException ioe) {
    ioe.printStackTrace();
} finally {
    try {
        if (is != null)
            is.close();
    } catch (IOException ioe) {
    }
}
}
```

## Output

## **Result**

Thus the web client program to download a web page was written and executed.

### **Ex.No: 4**

### **PROGRAM TO IMPLEMENT DNS USING TCP / UDP SOCKETS**

#### **Aim**

To write a program to implement DNS using TCP / UDP sockets.

#### **Algorithm**

##### **Client Side**

1. Start the program.
2. Create a socket which binds the Ip address of server and the port address to acquire service.
3. After establishing connection, client needs to send IP address/ URL to lookup the DNS to server.
4. Receive and print the same data from server.
5. Close the socket.
6. End the program.

##### **Server Side**

1. Start the program.
2. Create a server socket to activate the port address.
3. Create a socket for the server socket which accepts the connection.
4. After establishing connection receive the data from client.
5. Process the data and send the same data to client.
6. Close the socket.
7. End the program.

#### **Program**

**cdns.java**

```

import java.io.*;
import java.net.*;
import java.util.*;

class cdns {
    public static void main(String args[]) {
        try {
            DatagramSocket client = new DatagramSocket();
            InetAddress addr = InetAddress.getByName("127.0.0.1");

            byte[] sendbyte = new byte[1024];
            byte[] receivebyte = new byte[1024];
            BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
            System.out.println("Enter the DOMAIN NAME or IP address:");
            String str = in.readLine();
            sendbyte = str.getBytes();
            DatagramPacket sender = new DatagramPacket(sendbyte, sendbyte.length,
addr, 1309);
            client.send(sender);
            DatagramPacket receiver = new DatagramPacket(receivebyte,
receivebyte.length);
            client.receive(receiver);
            String s = new String(receiver.getData());
            System.out.println("IP address or DOMAIN NAME: " + s.trim());
            client.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}

```

### **sdns.java**

```

import java.io.*;
import java.net.*;
import java.util.*;

class sdns {
    public static void main(String args[]) {
        try {
            DatagramSocket server = new DatagramSocket(1309);
            while (true) {
                byte[] sendbyte = new byte[1024];
                byte[] receivebyte = new byte[1024];
                DatagramPacket receiver = new DatagramPacket(receivebyte,
receivebyte.length);
                server.receive(receiver);
                String str = new String(receiver.getData());
                String s = str.trim();
                InetAddress addr = receiver.getAddress();
                int port = receiver.getPort();
                String ip[] = { "165.165.80.80", "165.165.79.1" };

```

## Output

```
C:\Users\reddy\.jdks\openjdk-16.0.1\bin\java.exe "-javaagent:C:\Users\reddy\IdeaProjects\Java\lib\junit-jupiter-engine-5.7.1.jar" -jar C:\Users\reddy\IdeaProjects\Java\target\Java-Assignment-1.jar  
Process finished with exit code 0
```

```
C:\Users\reddy\.jdks\openjdk-16.0.1\bin\java.exe "-javaagent:C:\Program Files\  
Enter the DOMAIN NAME or IP address:  
www.aptitudeguru.com  
IP address or DOMAIN NAME: 165.165.80.80  
  
Process finished with exit code 0
```

## **Result**

Thus the program implement DNS using TCP / UDP sockets was written and executed.

### **Ex. No: 5a**

#### **PROGRAM TO IMPLEMENT ECHO CLIENT AND ECHO SERVER USING TRANSPORT LAYER PROTOCOL**

##### **Aim**

To write a program to implement echo client and echo server using Transport layer protocol.

##### **Algorithm**

###### **Client Side**

1. Start the program.
2. Create a socket which binds the Ip address of server and the port address to acquire service.
3. After establishing connection send a data to server.
4. Receive and print the same data from server.
5. Close the socket.
6. End the program.

###### **Server Side**

1. Start the program.
2. Create a server socket to activate the port address.
3. Create a socket for the server socket which accepts the connection.
4. After establishing connection receive the data from client.

5. Print and send the same data to client.
6. Close the socket.
7. End the program.

### **Program**

#### **echoc.java**

```
import java.io.*;
import java.net.*;

public class echoc {
    public static void main(String args[]) {
        Socket c = null;
        String line;
        DataInputStream is, is1;
        PrintStream os;
        try {
            c = new Socket("localhost", 8080);
        } catch (IOException e) {
            System.out.println(e);
        }
        try {

            os = new PrintStream(c.getOutputStream());
            is = new DataInputStream(System.in);
            is1 = new DataInputStream(c.getInputStream());
            do {
                System.out.print("Client: ");
                line = is.readLine();
                os.println(line);
                if (!line.equals("exit"))
                    System.out.println("Server: " + is1.readLine());
            } while (!line.equals("exit"));
        } catch (IOException e) {
            System.out.println("socket closed");
        }
    }
}
```

#### **echos.java**

```
import java.io.*;
import java.net.*;
import java.lang.*;

public class echos {
    public static void main(String args[]) throws IOException {
        ServerSocket s = null;
        String line;
        DataInputStream is;
        PrintStream ps;
        Socket c = null;
        try {
            s = new ServerSocket(8080);
```

```

        } catch (IOException e) {
            System.out.println(e);
        }
    try {
        c = s.accept();
        is = new DataInputStream(c.getInputStream());
        ps = new PrintStream(c.getOutputStream());

        while (true) {
            line = is.readLine();
            System.out.println("Message received and sent back to client.");
            ps.println(line);
        }
    } catch (IOException e) {
        System.out.println(e);
    }
}
}

```

## Output

```
C:\Users\reddy\.jdks\openjdk-16.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.1\lib\idea_rt.jar=53945:C:
msg received and sent back to client
msg received and sent back to client
|
```

```
C:\Users\reddy\.jdks\openjdk-16.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.1\lib\
client
hi
server:hi
client
hello
server:hello
client
|
```

**Result**

Thus the program for simulation of echo server and echo client was written and executed.

**Ex. No: 5 b****PROGRAM TO IMPLEMENT CHAT APPLICATION USING TRANSPORT  
LAYER PROTOCOL****Aim**

To write a program to implement a client -server application for chat using TCP.

**Algorithm****Client Side**

1. Start the program
2. Include necessary package in java
3. To create a socket in client to server.
4. The client establishes a connection to the server.
5. The client accepts the connection and to send the data from client to server.

6. The client communicates the server to send the end of the message
7. Stop the program.

### **Server Side**

1. Start the program
2. Include necessary package in java
3. To create a socket in server to client
4. The server establishes a connection to the client.
5. The server accepts the connection and to send the data from server to client and
6. vice versa
7. The server communicates the client to send the end of the message.
8. Stop the program.

### **Program**

#### **tcpc.java**

```
import java.net.*;
import java.io.*;

public class tcpc {
    public static void main(String arg[]) {
        Socket c = null;
        String line;
        DataInputStream is, is1;
        PrintStream os;
        try {
            c = new Socket("localhost", 9998);
        } catch (IOException e) {
            System.out.println(e);
        }
        try {
            os = new PrintStream(c.getOutputStream());
            is = new DataInputStream(System.in);
            is1 = new DataInputStream(c.getInputStream());
            do {
                System.out.print(" Client: ");
                line = is.readLine();
                os.println(line);
                System.out.println(" Server:" + is1.readLine());
            } while (line.equalsIgnoreCase(" quit ") == false);
            is1.close();
            os.close();
        } catch (IOException e) {
            System.out.println(" Socket Closed !Message Passing is over ");
        }
    }
}
```

#### **tcps.java**

```
import java.net.*;
import java.io.*;

public class tcps {
```

```

public static void main(String arg[]) {
    ServerSocket s = null;
    String line;
    DataInputStream is = null, is1 = null;
    PrintStream os = null;
    Socket c = null;
    try {
        s = new ServerSocket(9998);
    } catch (IOException e) {
        System.out.println(e);

    }
    try {
        c = s.accept();
        is = new DataInputStream(c.getInputStream());
        is1 = new DataInputStream(System.in);
        os = new PrintStream(c.getOutputStream());
        do {
            line = is.readLine();

            System.out.println("Client: " + line);
            System.out.print("Server: ");
            line = is1.readLine();
            os.println(line);
        } while (line.equalsIgnoreCase("quit") == false);
        is.close();
        os.close();
    } catch (IOException e) {
        System.out.println(e);
    }
}
}

```

## Output

```

C:\Users\reddy\.jdks\openjdk-16.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=5314,C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\bin" -Dfile.encoding=UTF-8 com.intellij.rt.execution.junit.JUnitStarter -idePath=C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1 Client:hi
Server:
hello
Client:how are u?
Server:
im gud

```

```
C:\Users\reddy\.jdks\openjdk-16.0.1\bin\java.exe "-javaagent:C:\Program Files\  
Client:  
hi  
Server:hello  
Client:  
how are u?  
Server:im gud  
Client:
```

### **Result**

Thus the program to implement a client-server application for chat using TCP was written and executed.

### **Ex. No: 6**

### **IMPLEMENTATION OF FILE TRANSFER USING TCP / UDP**

#### **Aim**

To write a java program for file transfer using TCP Sockets.

#### **Algorithm**

### **Server side**

1. Import java packages and create class file server.
2. Create a new server socket and bind it to the port.
3. Accept the client connection
4. Get the file name and stored into the BufferedReader.
5. Create a new object class file and realine.
6. If file exists, then FileReader read the content until EOF is reached.
7. Stop the program.

### **Client side**

1. Import java packages and create class file server.
2. Create a new server socket and bind it to the port.
3. Now connection is established.
4. The object of a BufferedReader class is used for storing data content which has been retrieved from socket object.
5. The connection is closed.
6. Stop the program.

### **Program**

#### **filec.java**

```
import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.net.InetAddress;
import java.net.Socket;

public class filec {
    public static void main(String[] args) throws Exception {
        // Initialize socket
        Socket socket = new Socket(InetAddress.getByName("localhost"), 5000);
        byte[] contents = new byte[10000];
        // Initialize the FileOutputStream to the output file's full path.
        FileOutputStream fos = new FileOutputStream("D:\\College\\Semester 5\\CN
LAB\\Programs\\demo.png");
        BufferedOutputStream bos = new BufferedOutputStream(fos);
        InputStream is = socket.getInputStream();
        // No of bytes read in one read() call
        int bytesRead = 0;
        while ((bytesRead = is.read(contents)) != -1)
            bos.write(contents, 0, bytesRead);
        bos.flush();
        socket.close();
        System.out.println("File saved successfully!");
    }
}
```

#### **files.java**

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.net.ServerSocket;
```

```

import java.net.Socket;

public class files {
    public static void main(String[] args) throws Exception {
        // Initialize Sockets
        ServerSocket ssock = new ServerSocket(5000);
        Socket socket = ssock.accept();
        // The InetAddress specification
        InetAddress IA = InetAddress.getByName("localhost");

        // Specify the file
        File file = new File("D:\\College\\Semester 5\\CN LAB\\Outputs\\arp.png");
        FileInputStream fis = new FileInputStream(file);
        BufferedInputStream bis = new BufferedInputStream(fis);
        // Get socket's output stream
        OutputStream os = socket.getOutputStream();
        // Read File Contents into contents array
        byte[] contents;
        long fileLength = file.length();
        long current = 0;
        long start = System.nanoTime();
        while (current != fileLength) {
            int size = 10000;
            if (fileLength - current >= size)
                current += size;
            else {
                size = (int) (fileLength - current);
                current = fileLength;
            }
            contents = new byte[size];
            bis.read(contents, 0, size);
            os.write(contents);
            System.out.print("Sending file ... " + (current * 100) / fileLength + "% complete!");
        }
        os.flush();
        // File transfer done. Close the socket connection!
        socket.close();
        ssock.close();
        System.out.println("File sent successfully!");
    }
}

```

## Output

```
C:\Users\reddy\jdk\openjdk-16.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.1\lib\idea_rt.jar=55936:C:\Program Files\JetBrains\I
```

```
Sending file ... 21% complete! Sending file ... 42% complete! Sending file ... 64% complete! Sending file ... 85% complete! Sending file ... 100% complete! File sent successfully!
```

```
Process finished with exit code 0
```

```
C:\Users\reddy\.jdks\openjdk-16.0.1\bin\java.exe "-jar"  
File saved successfully!
```

```
Process finished with exit code 0
```

### **Result**

Thus the file transfer operation is done and executed successfully.

## **Ex. No: 7**

### **NETWORK PACKET ANALYSIS USING TOOLS LIKE WIRESHARK, TCPDUMP**

#### **Study about Wireshark**

Wireshark is an open-source packet analyzer, which is used for **education, analysis, software development, communication protocol development, and network troubleshooting.**

It is used to track the packets so that each one is filtered to meet our specific needs. It is commonly called as a **sniffer, network protocol analyzer, and network analyzer.** It is also used by network security engineers to examine security problems.

Wireshark is a free to use application which is used to apprehend the data back and forth. It is often called as a free packet sniffer computer application. It puts the network card into an unselective mode, i.e., to accept all the packets which it receives.

A packet is a unit of data which is transmitted over a network between the origin and the destination. Network packets are small, i.e., maximum **1.5 Kilobytes for Ethernet packets and 64 Kilobytes for IP packets.** The data packets in the Wireshark can be viewed online and can be analyzed offline.

#### **Installation of Wireshark Software**

Below are the steps to install the Wireshark software on the computer:

- Open the web browser.
- Search for '**Download Wireshark.**'
- Select the Windows installer according to your system configuration, either 32-bit or 64-bit. Save the program and close the browser.
- Now, open the software, and follow the install instruction by accepting the license.
- The Wireshark is ready for use.

On the network and Internet settings option, we can check the interface connected to our computer.

If Linux users, then you will find Wireshark in its package repositories.

#### **i. Packet Capture Using Wireshark**

Capturing your traffic with Wireshark

After starting Wireshark, do the following:

1. Select **Capture | Interfaces**
2. Select the interface on which packets need to be captured. This will usually be the interface where the Packet/s column is constantly changing, which would indicate the presence of live traffic). If you have multiple network interface cards (i.e. LAN card and Wi-Fi adapter) you may need to check with your IT administrator to determine the right interface.
3. Click the **Start** button to start the capture.
4. Recreate the problem. The capture dialog should show the number of packets increasing. Try to avoid running any other internet applications while capturing, closing other browsers, Instant messengers etc.
5. Once the problem which is to be analyzed has been reproduced, click on **Stop**. It may take a few seconds for Wireshark to display the packets captured.
6. Save the packet trace in the default format. Click on the **File** menu option and select **Save As**. By default Wireshark will save the packet trace in libpcap format. This is a filename with a.pcap extension.

## **ii. Starting Wireshark using commandline**

You can start Wireshark from the commandline, but it can also be started from most Window managers as well. In this section we will look at starting it from the commandline.

Wireshark supports a large number of command line parameters. To see what they are, simply enter the command *wireshark -h*

Wireshark3.7.3(v3.7.3rc0-25-g72703582d587)

Interactively dump and analyze network traffic.

See <https://www.wireshark.org> for more information. Usage:

`wireshark [options] ... [ <infile> ]`

Captureinterface:

```
-i<interface>,--interface<interface>
      name or idx of interface(def: first non-loopback)
-f<capturefilter>    packetfilter in libpcap filter syntax
```

## **iii. Viewing Captured Traffic**

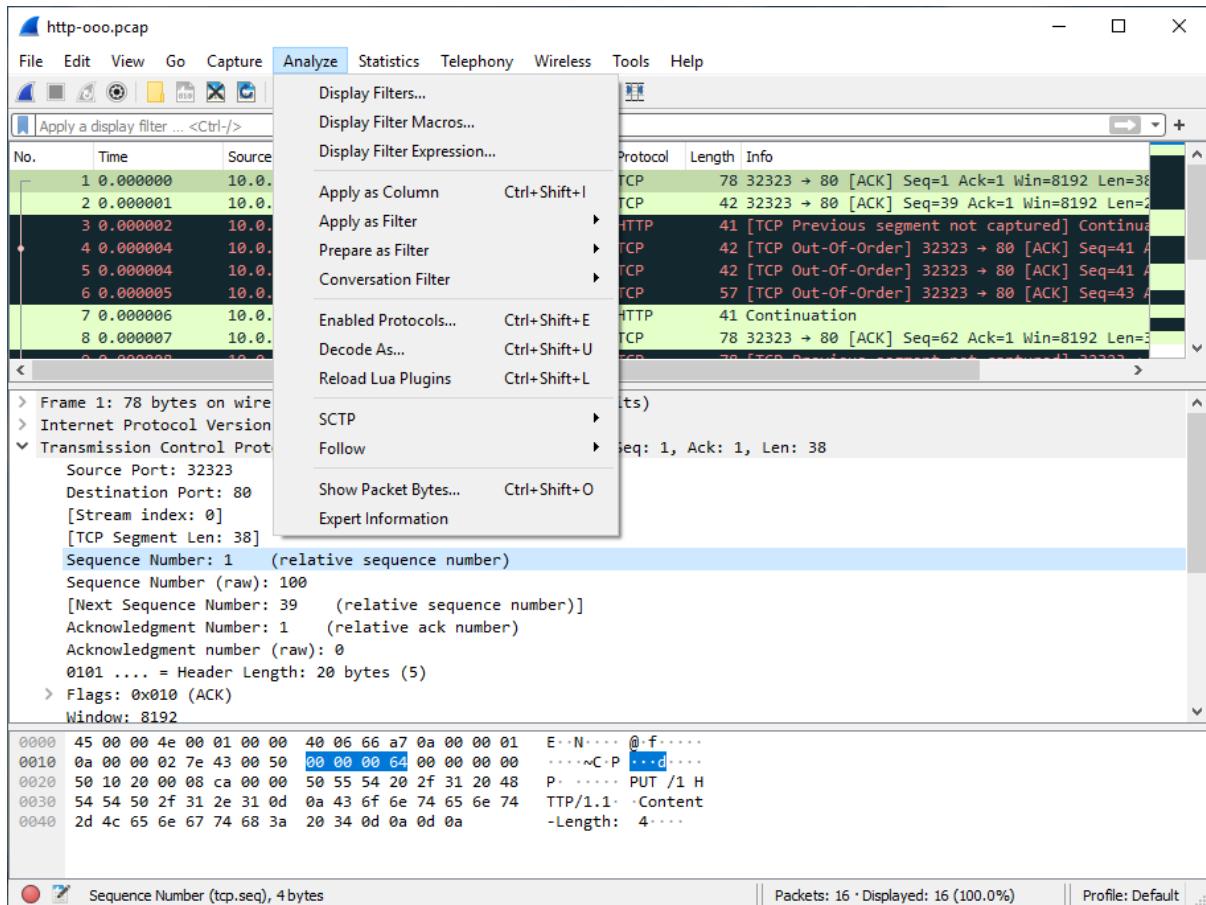
Once you have captured some packets or you have opened a previously saved capture file, you can view the packets that are displayed in the packet list pane by simply **clicking on a packet in the packet list pane**, which will bring up the selected packet in the tree view and byte view panes. Once you have captured some packets or you have opened a previously

saved capture file, you can view the packets that are displayed in the packet list pane by simply **clicking on a packet in the packet list pane**, which will bring up the selected packet in the tree view and byte view panes.

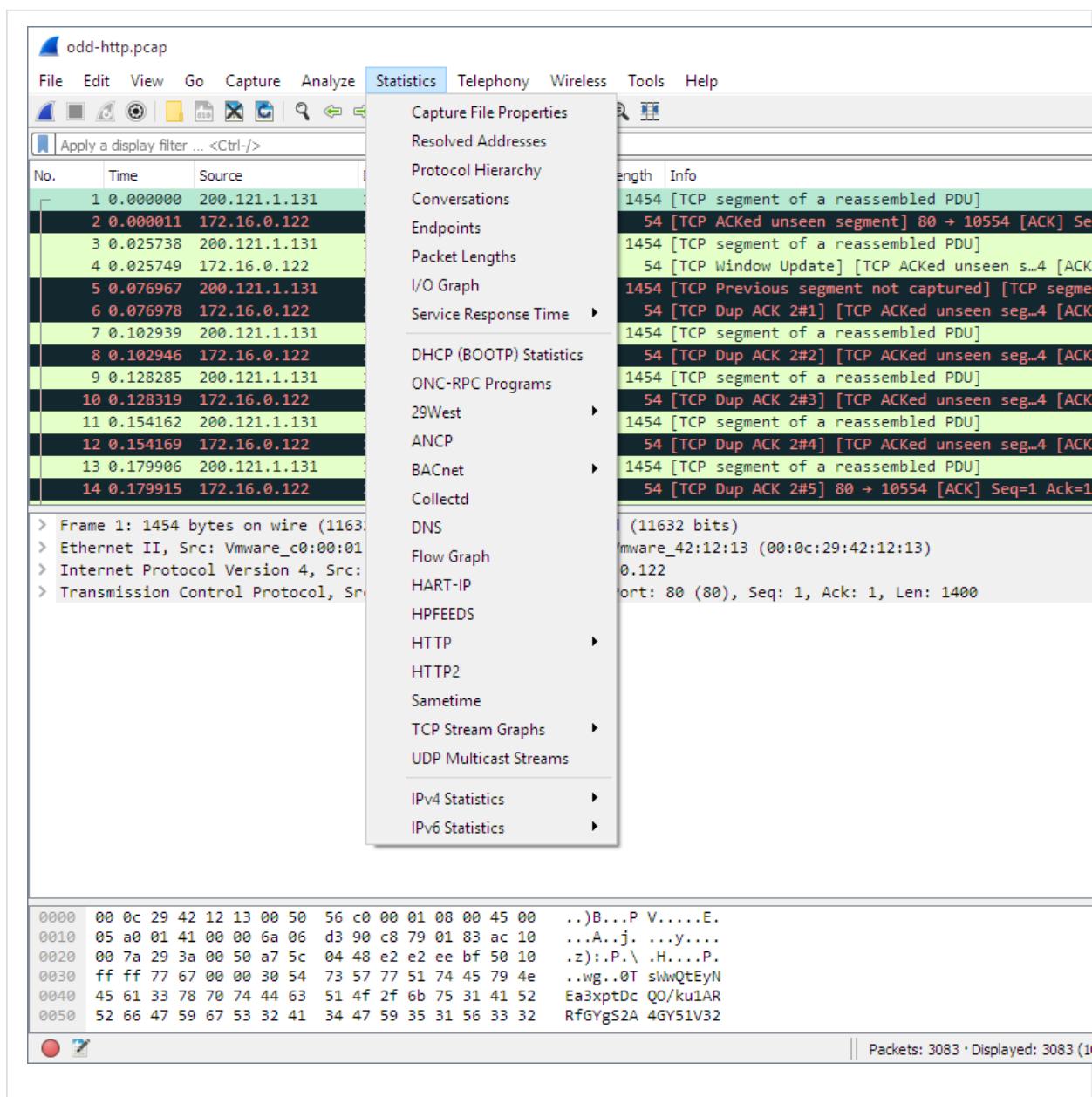
#### iv. Analysis and Statistics & Filters.

##### Analysis menu in wireshark

The “Analyze” Menu The Wireshark Analyze menu contains the fields shown in Analyze menu items.

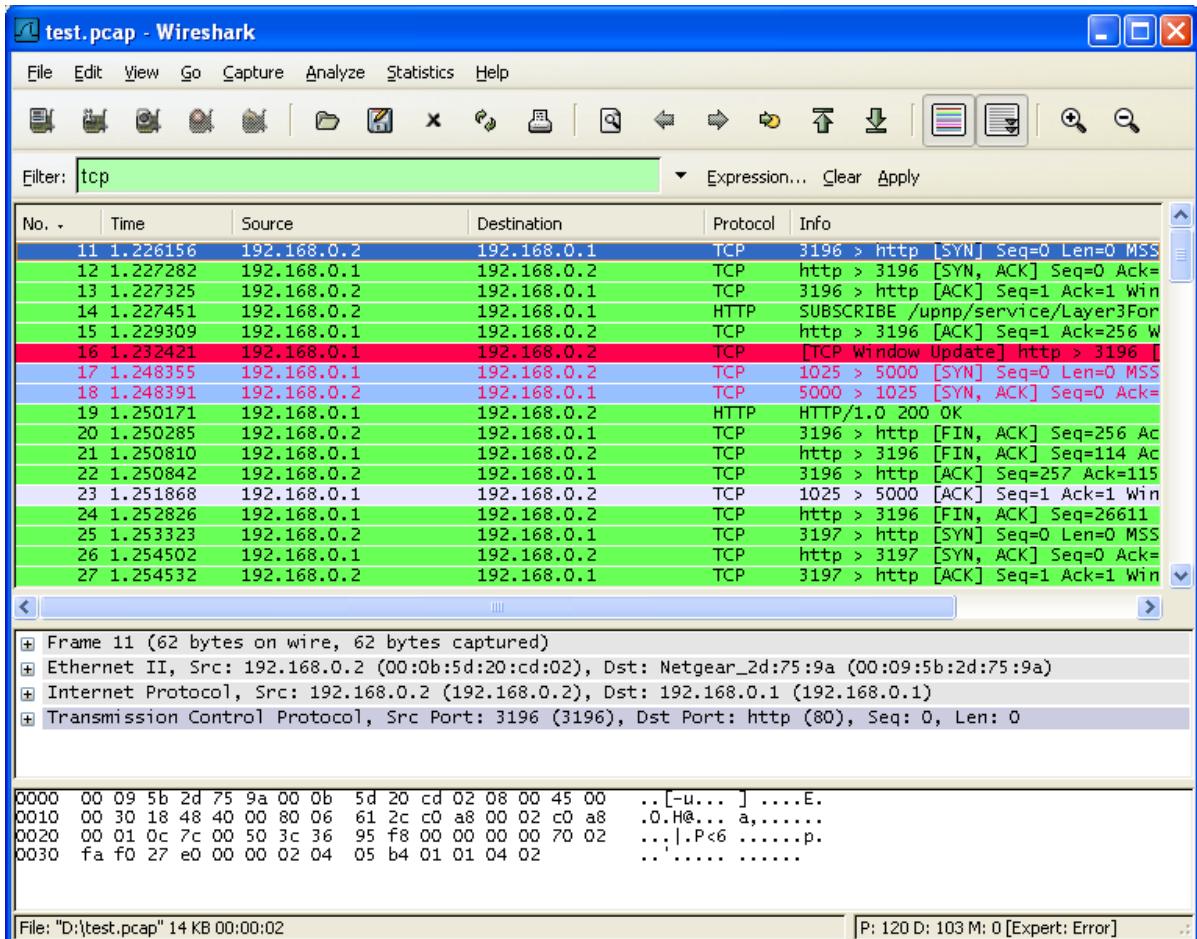


## The “Statistics” Menu in wire shark



## Filters in wireshark

Wireshark has two filtering languages: **capture filters** and **display filters**. Capture filters are used for filtering when capturing packets, “Filtering while capturing”. Display filters are used for filtering which packets are displayed.



# Capture network traffic with `tcpdump`

To capture all traffic on a specific interface, use the `-i` flag followed by the interface name. For example, to capture traffic on the `eth0` interface:

**tcpdump -i eth0**

To see a list of all available interfaces, use the command:

## tcpdump -D

### **Capturing traffic to/from a specific host**

To capture traffic to or from a specific host, use the **host** keyword followed by the hostname or IP address:

```
tcpdump host 192.168.1.100
```

This will capture all traffic to and from the host with the IP address **192.168.1.100**.

### **Capturing traffic on a specific port**

To capture traffic on a specific port, use the `port` keyword followed by the port number:

**tcpdump port 80**

## *Combining filters*

Combine filters using and, or, and not operators. For example, to capture all traffic to or from host 192.168.1.100 on port 80, use:

**tcpdump host 192.168.1.100 and port 80**

To capture traffic from **192.168.1.100** on either port 80 or 443, use:

**tcpdump src host 192.168.1.100 and \(|( port 80 or port 443 |)**

## **Filtering by protocol**

To filter by protocol, use the **ip**, **tcp**, **udp**, or other protocol keywords. For example, to capture only TCP traffic:

**tcpdump tcp**

To capture only UDP traffic:

**tcpdump udp**

## **Filtering by source or destination**

To filter by source or destination host or port, use the **src** or **dst** keywords:

**tcpdump src host 192.168.1.100**

**tcpdump dst port 443**

## **Result**

Thus the Network Packet Analysis using tools like Wireshark and TCPDUMP has been studied successfully.

## Ex. No: 8

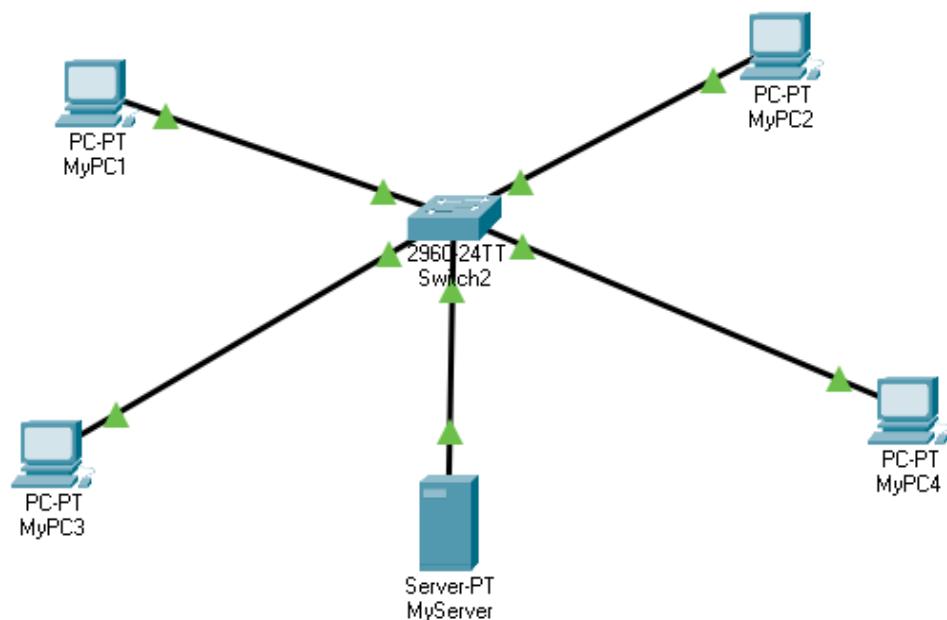
### SIMULATION OF CONGESTION / FLOW CONTROL ALGORITHMS USING NS

#### Aim

To simulate congestion / flow control algorithms using NS.

#### Procedure

##### TCP Congestion Control



1. Create the Network as shown in the Figure. Which Contains 4 PC's and 1 Server. All the devices are connected through 2960 – 24 TT Switch.

S No	Device Name	IP Address
1	PC-PT MyPC1	192.168.10.1
2	PC-PT MyPC2	192.168.10.2
3	PC-PT MyPC3	192.168.10.3
4	PC-PT MyPC4	192.168.10.4
5	Server-PT MyServer	192.168.10.250

2. After establishing the network, Check the connection by using Ping Command.

```

Packet Tracer PC Command Line 1.0
C:\>Ping 192.168.10.1

Pinging 192.168.10.1 with 32 bytes of data:

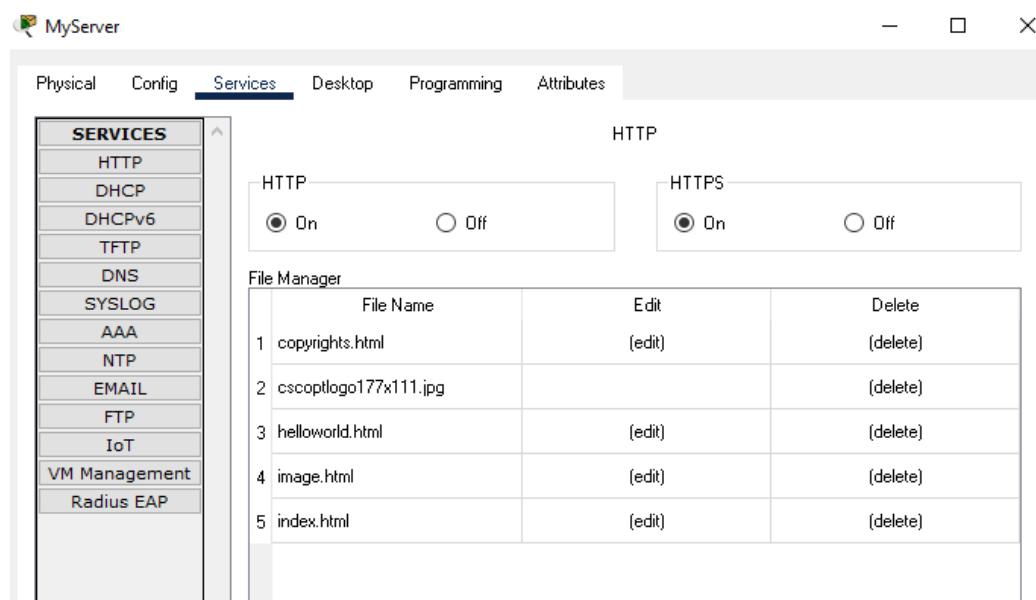
Reply from 192.168.10.1: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.10.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

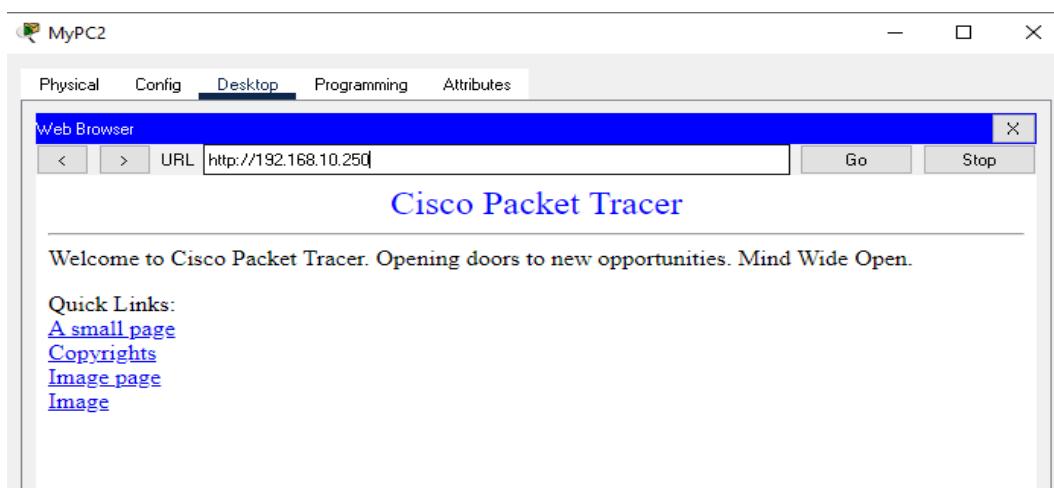
C:\>

```

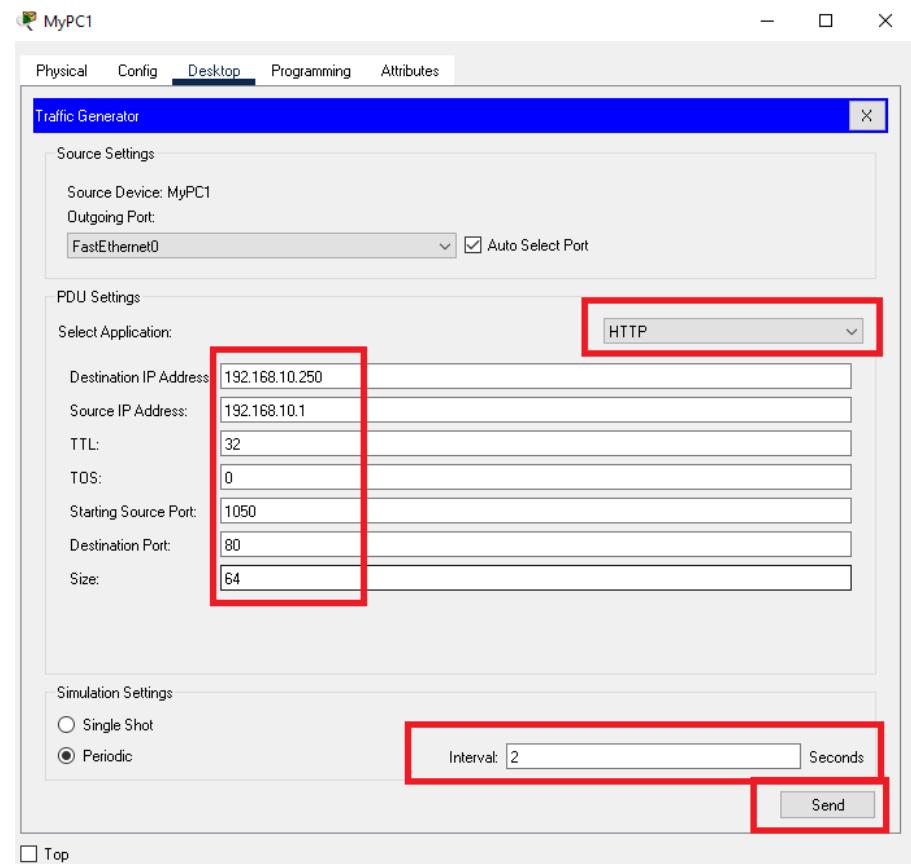
- Check the services of server, it should be enabled HTTP – ON and HTTPS – ON.



- From anyone of the PC, Go to web browser from Desktop tab, Type the URL <http://192.168.10.250>



- To Generate Traffic, You can choose any PC → Traffic Generator.

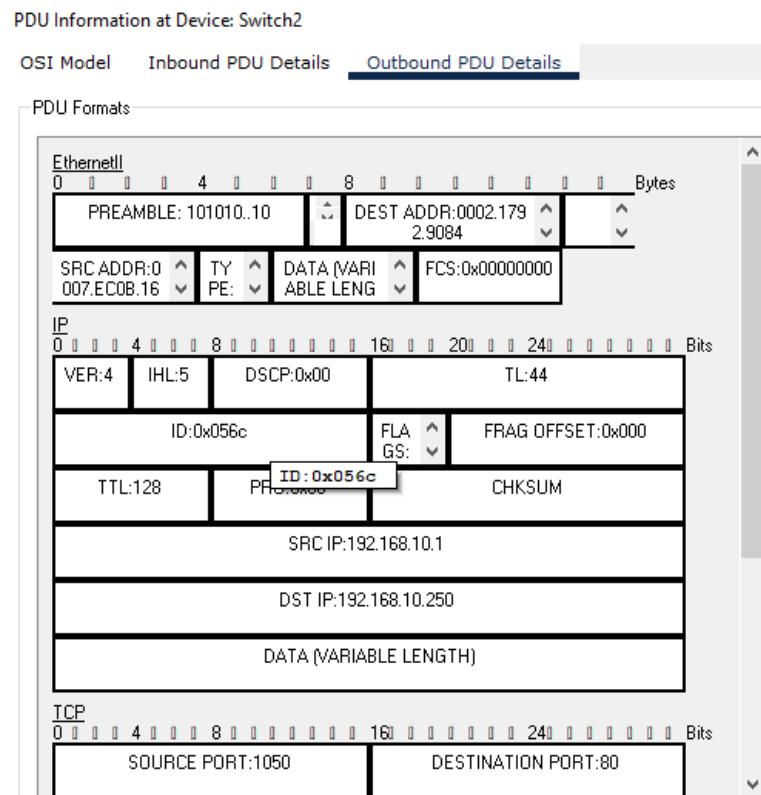
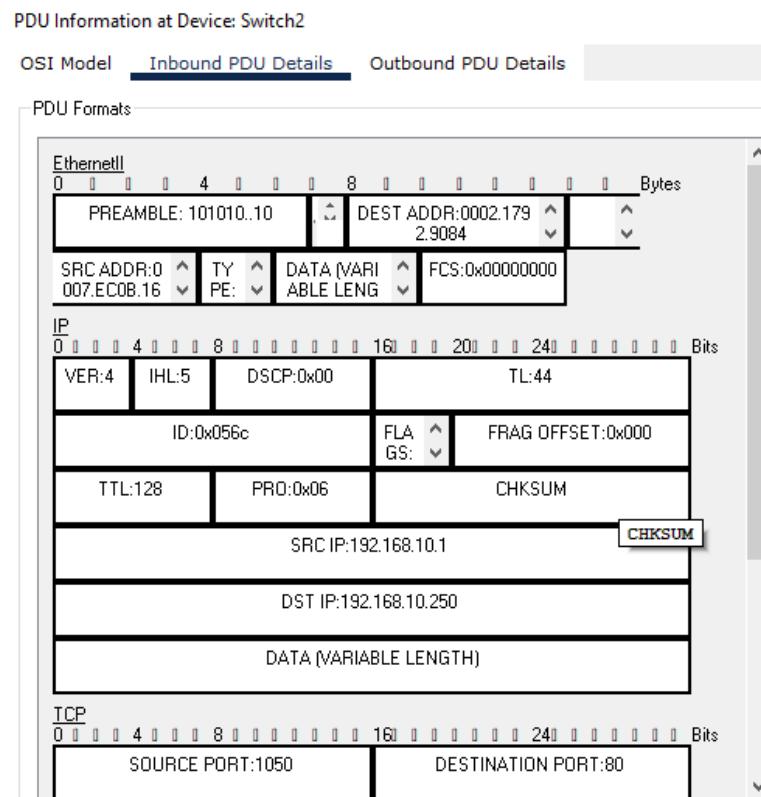


## 6. Trace the Packet Transmission of Every Node.

PDU Information at Device: Switch2

OSI Model	Inbound PDU Details	Outbound PDU Details
At Device: Switch2 Source: MyPC1 Destination: 192.168.10.250		
<b>In Layers</b>	<b>Out Layers</b>	
Layer7	Layer7	
Layer6	Layer6	
Layer5	Layer5	
Layer4	Layer4	
Layer3	Layer3	
Layer 2: Ethernet II Header 0007.EC0B.1669 >> 0002.1792.9084	Layer 2: Ethernet II Header 0007.EC0B.1669 >> 0002.1792.9084	
Layer 1: Port FastEthernet0/5	Layer 1: Port(s): FastEthernet0/1	

1. FastEthernet0/5 receives the frame.



## Result

Thus the simulation for congestion / flow control was done using NS.

## Ex. No: 9

### PERFORMANCE OF TCP AND UDP USING SIMULATION TOOL

#### Aim

To perform TCP and UDP using simulation tool.

#### Procedure

#### CONNECT TWO DIFFERENT NETWORKS

##### Step: 1: Create Two Different Networks

**Network 1** → should be in an IP Group → 192.168.1.0 With Default Gateway → 192.168.1.1

Network 1 → PC0 → Assign IP Address → 192.168.1.2

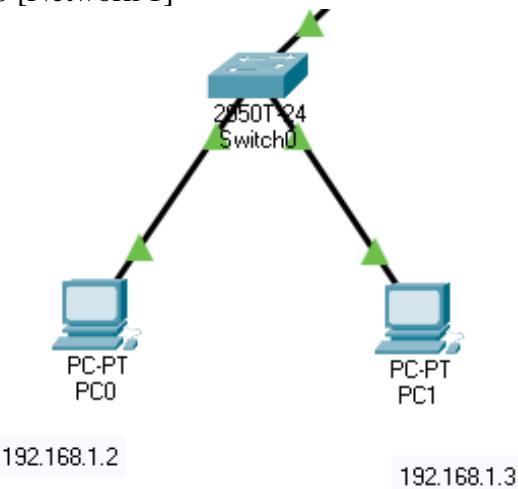
Network 1 → PC1 → Assign IP Address → 192.168.1.3

**Network 2** → should be in an IP Group → 192.168.2.0 With Default Gateway → 192.168.2.1

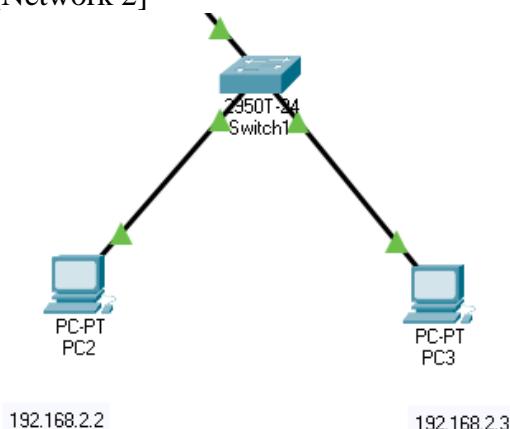
Network 2 → PC2 → Assign IP Address → 192.168.2.2

Network 2 → PC3 → Assign IP Address → 192.168.2.3

**Step:2:** Connect the PC's With Network Switch 2950T using Copper Straight through Cable  
PC0 & PC1 → 2950T Switch0 [Network 1]



PC2 & PC3 → 2950T Switch1 [Network 2]



**Step 3:** Add Routers 1841 for Each Network

Network 1 → 1841 Router0

Network 2 → 1841 Router1

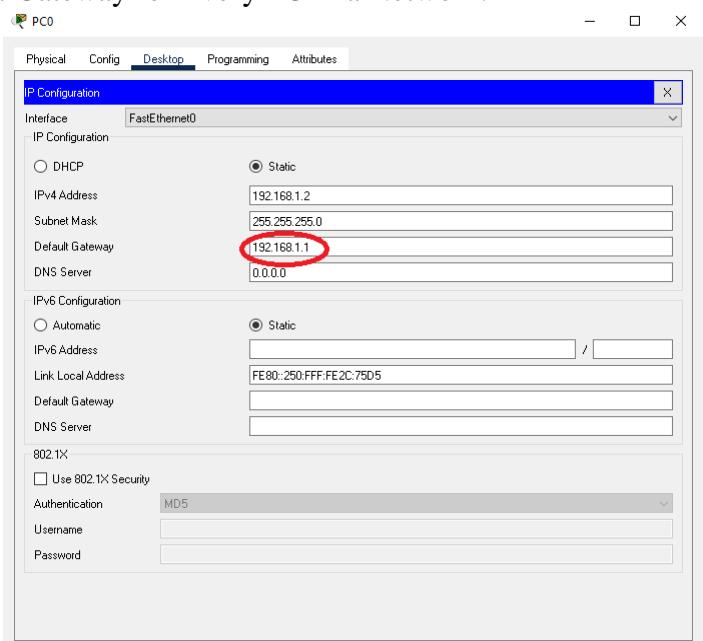
**Step 4:** Connect Both the Router with Serial Cable

Router0 → Serial 0/0/0

Router1→Serial 0/0/1



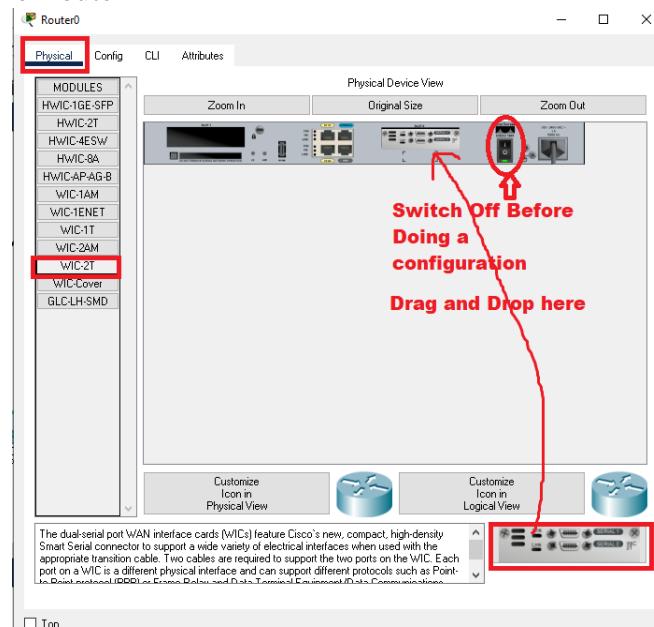
**Step: 5:** Assign Default Gateway for Every PC in a Network.



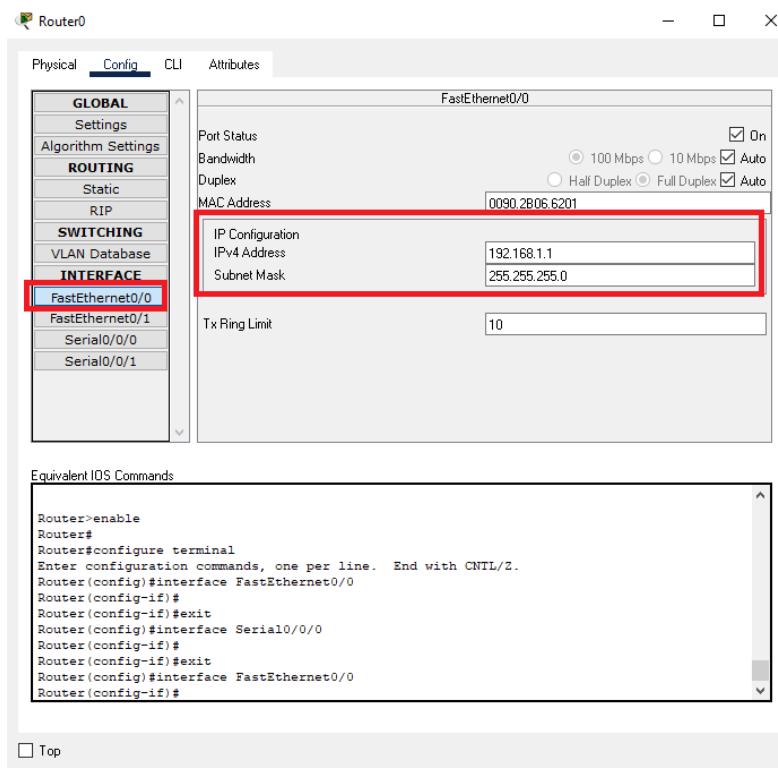
**For Network 1 → PC0 & PC1** → Default Gateway → 192.068.1.1

**For Network 2 → PC2 & PC3** → Default Gateway → 192.068.2.1

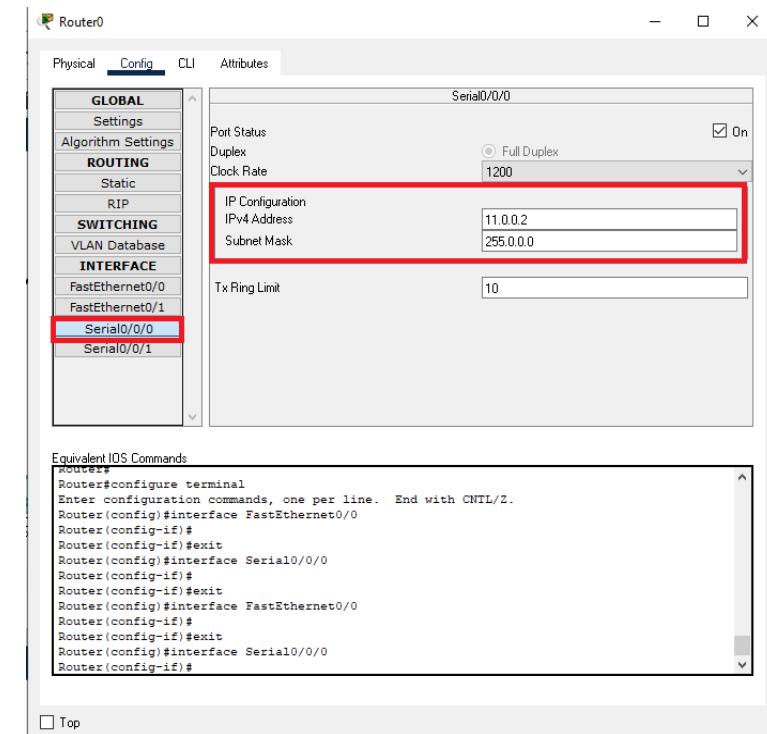
**Step 6:** Configure Both the Router



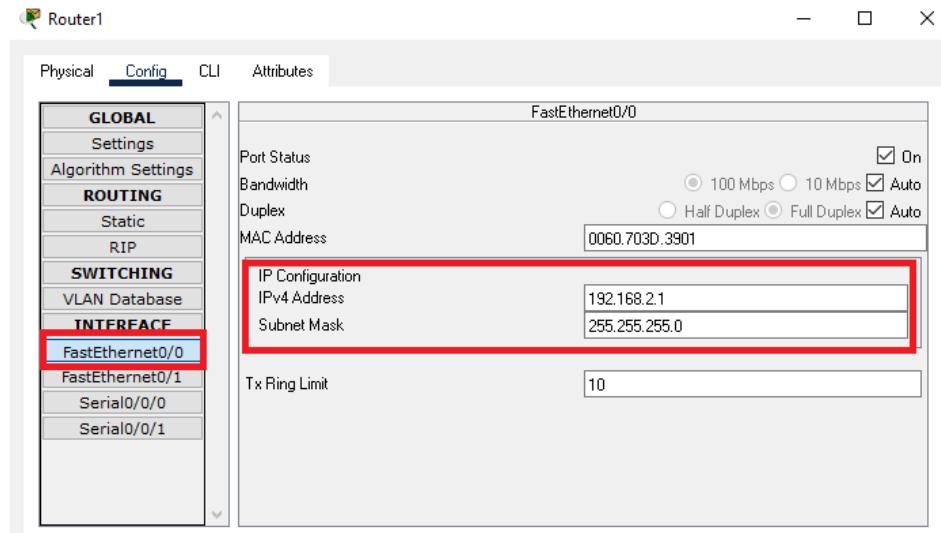
**Step 7:** Assign IP Configuration for Router0 FastEthernet 0/0 & Serial 0/0/0  
IP Address → 192.168.1.1 for FastEthernet 0/0



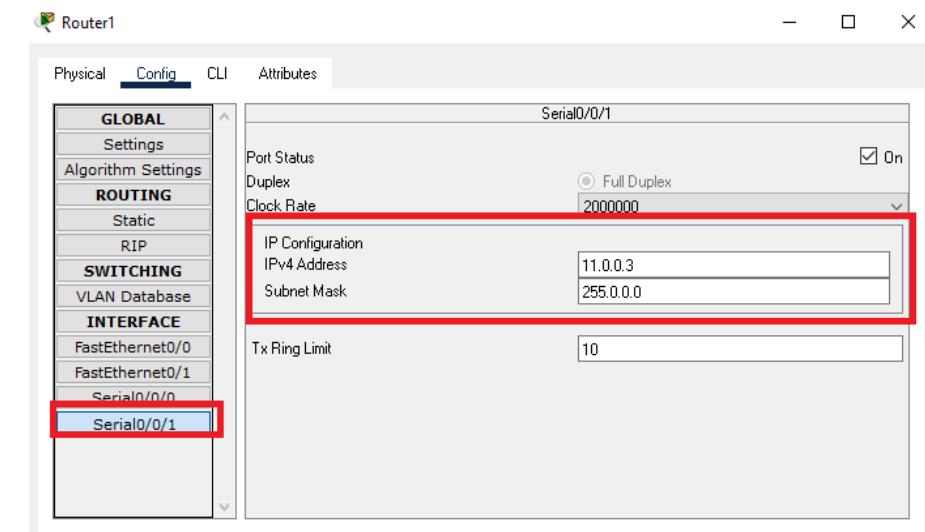
IP Address → 11.0.0.2 for Serial 0/0/0



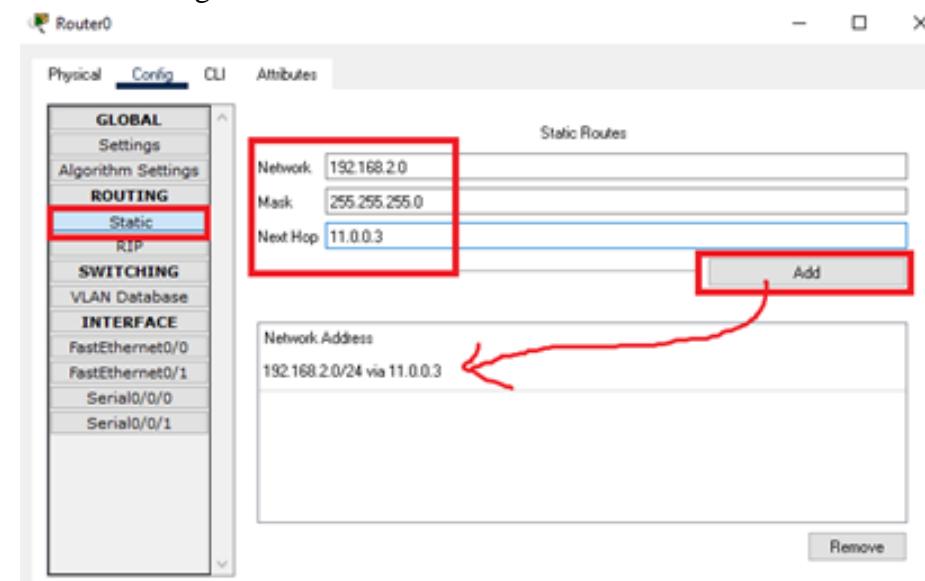
**Step 8:** Assign IP Configuration for Router1 FastEthernet 0/0 & Serial 0/0/1  
IP Address → 192.168.2.1 for FastEthernet 0/0



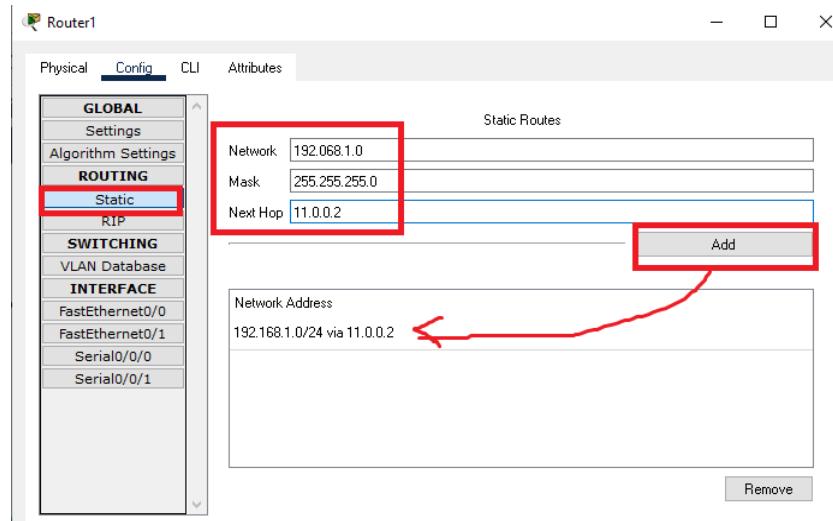
IP Address → 11.0.0.3 for Serial 0/0/1



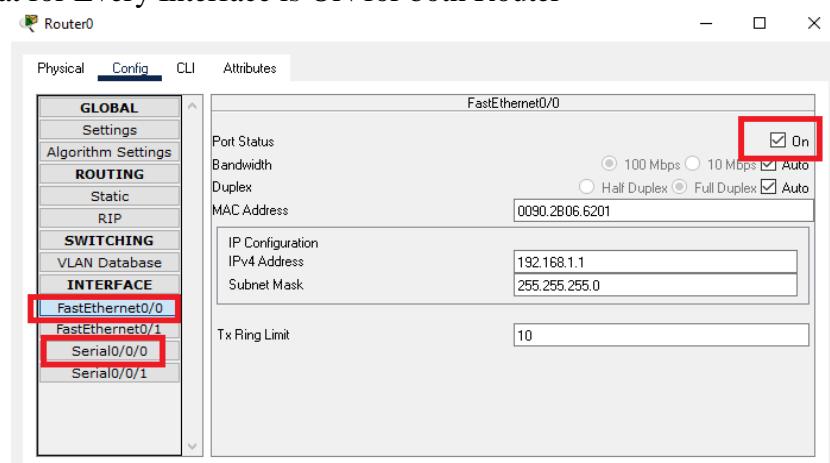
**Step9:** Add the Static Routing Information for the **Router 0**



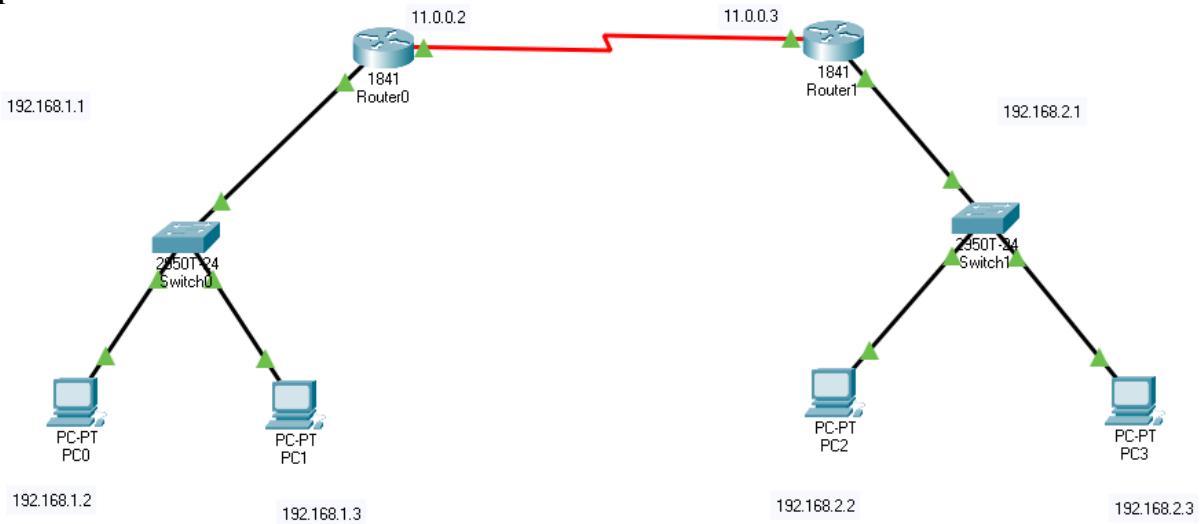
**Step 10:** Add the Static Routing Information for the **Router 1**



### Step11: Ensure that for Every Interface is ON for both Router



### Output



### Result

Thus the performance of TCP and UDP using simulation tools has been implemented.

## Ex.No: 10

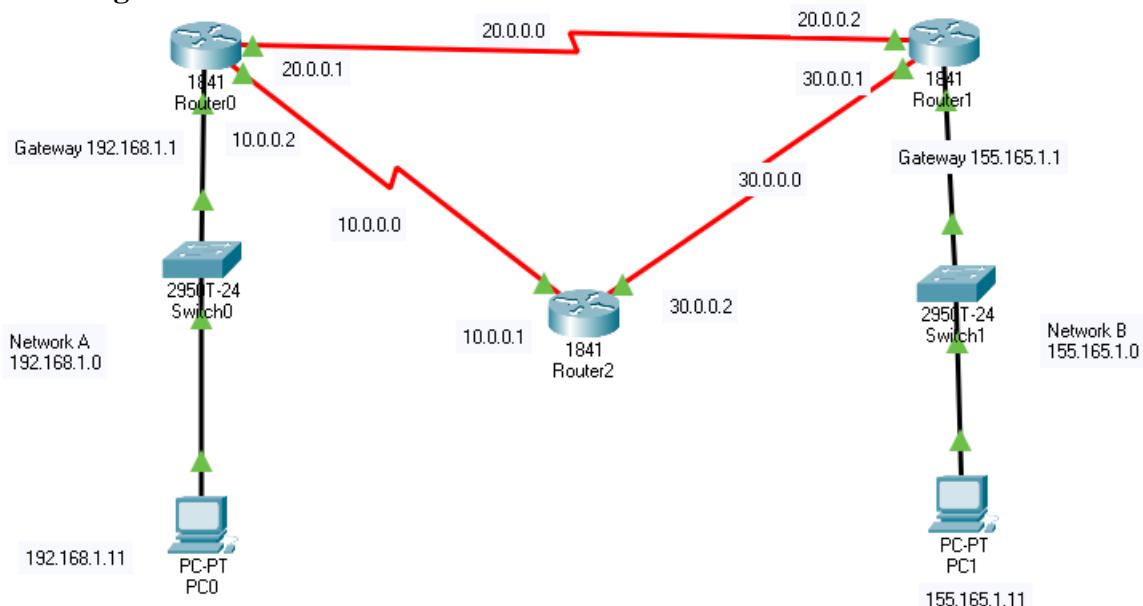
### SIMULATION OF DISTANCE VECTOR AND LINK STATE ROUTING ALGORITHM

#### Aim

To simulate distance vector and link state routing algorithm.

#### Procedure

##### OSPF - Routing



**Step: 1:** Create a Networks with following devices

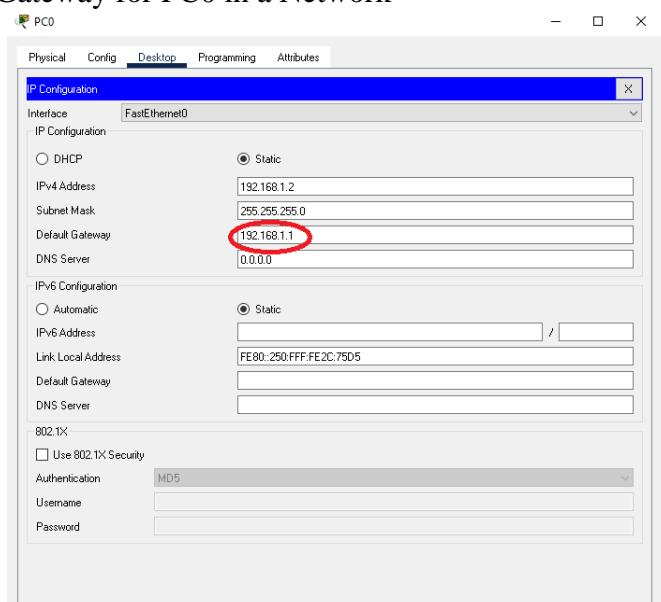
**Network A [192.168.1.0]**

PC0 → IP Address → 192.168.1.1

2950 T Switch0

1841 Router0

**Step: 2:** Assign Default Gateway for PC0 in a Network



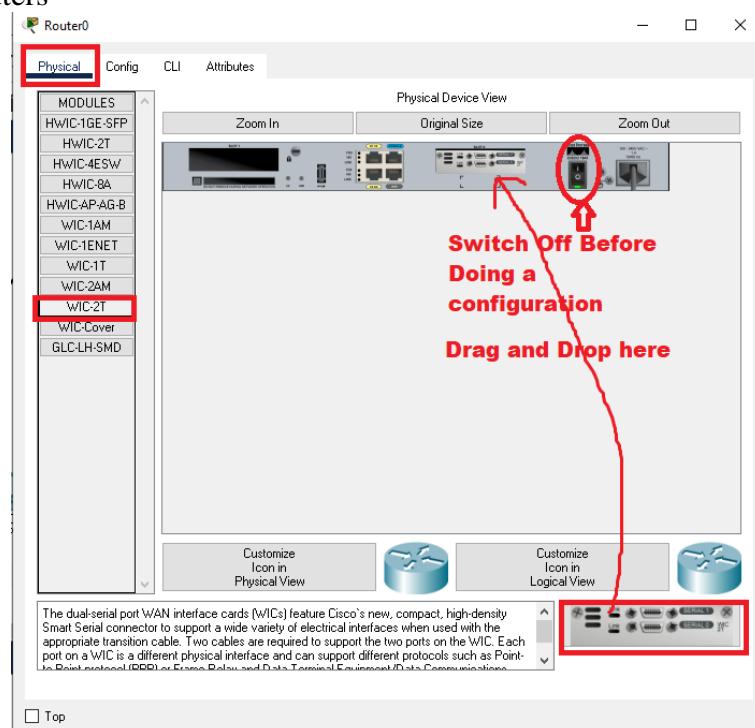
For Network A → PC0 → Default Gateway → 192.168.1.1

Similarly, do it for Network B

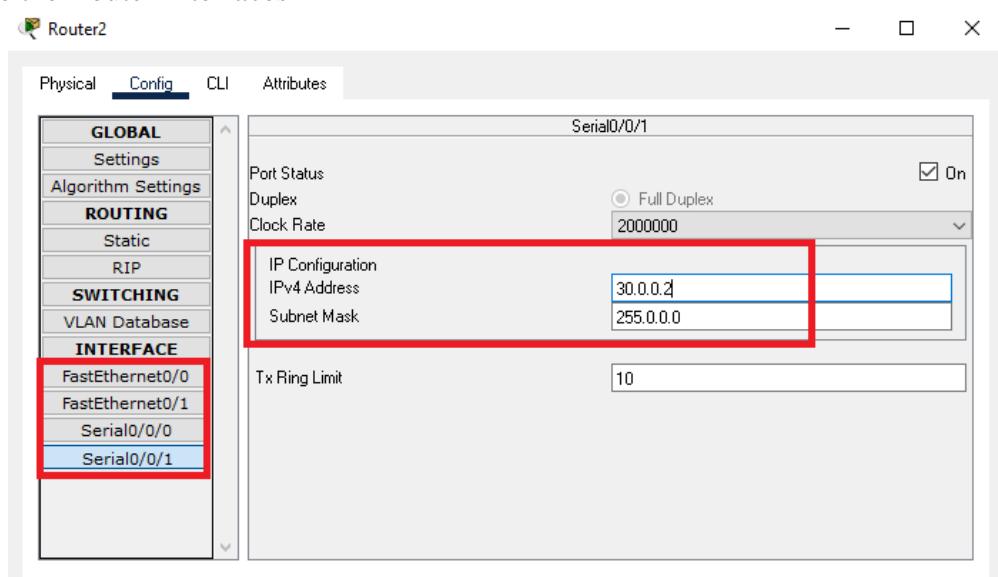
**Network B [155.165.1.0]**  
 PC1 → IP Address → 155.165.1.11  
 2950 T Switch0  
 1841 Router1

**Step: 3: Assign Default Gateway for PC1 in a Network**  
**For Network B → PC1 → Default Gateway → 155.165.1.1**

**Step: 4: Configure All the Router and Router Interfaces**  
 To configure the Routers



To configure the Router Interfaces



**Router 0:**  
 Fast Ethernet 0/0 → 192.168.1.1  
 Serial 0/0/0 → 20.0.0.1  
 Serial 0/0/1 → 30.0.0.2  
**Router 1:**  
 Fast Ethernet 0/0 → 155.165.1.1

Serial 0/0/0 → 30.0.0.1

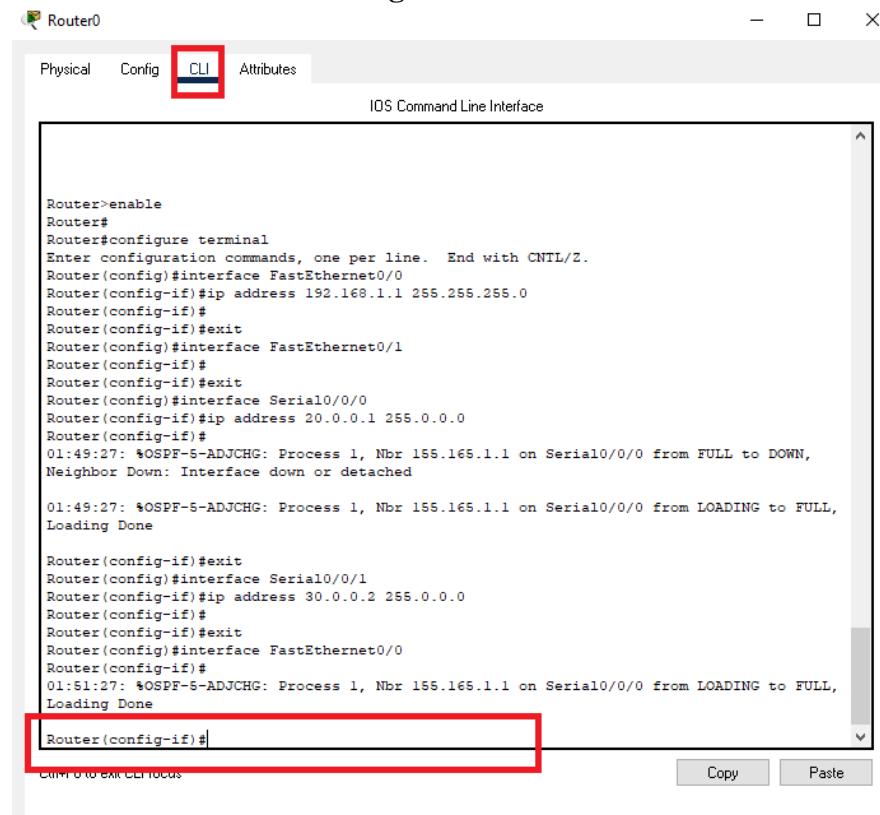
Serial 0/0/1 → 20.0.0.2

**Router 2:**

Serial 0/0/0 → 10.0.0.1

Serial 0/0/1 → 30.0.0.2

**Step: 5: Configure OSPF in all routers Using CLI**



The screenshot shows the Router0 CLI interface. The title bar says "Router0". Below it are tabs: "Physical", "Config" (which is selected and highlighted with a red box), and "Attributes". The main window is titled "IOS Command Line Interface". It contains the following configuration commands for Router 1:

```
Router>enable
Router#
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface FastEthernet0/0
Router(config-if)#ip address 192.168.1.1 255.255.255.0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/1
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial0/0/0
Router(config-if)#ip address 20.0.0.1 255.0.0.0
Router(config-if)#
01:49:27: *OSPF-5-ADJCHG: Process 1, Nbr 155.165.1.1 on Serial0/0/0 from FULL to DOWN,
Neighbor Down: Interface down or detached

01:49:27: *OSPF-5-ADJCHG: Process 1, Nbr 155.165.1.1 on Serial0/0/0 from LOADING to FULL,
Loading Done

Router(config-if)#exit
Router(config)#interface Serial0/0/1
Router(config-if)#ip address 30.0.0.2 255.0.0.0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/0
Router(config-if)#
01:51:27: *OSPF-5-ADJCHG: Process 1, Nbr 155.165.1.1 on Serial0/0/0 from LOADING to FULL,
Loading Done
```

The command line input field at the bottom is highlighted with a red box and contains "Router(config-if) #".

**Go to Router CLI and type the following**

**Router0:CLI**

```
#enable
#configure terminal
#router ospf 1
#network 192.168.1.0 0.0.0.255 area 0
#network 10.0.0.0 0.255.255.255 area 0
#network 20.0.0.0 0.255.255.255 area 0
#exit
```

**Router1:CLI**

```
#enable
#configure terminal
#router ospf 1
#network 10.0.0.0 0.255.255.255 area 0
#network 30.0.0.0 0.255.255.255 area 0
#exit
```

**Router2:CLI**

```
#enable
#configure terminal
#router ospf 1
#network 20.0.0.0 0.255.255.255 area 0
#network 30.0.0.0 0.255.255.255 area 0
#network 155.165.1.0 0.0.0.255.255 area 0
```

```
#exit
```

### Step: 6: Transfer the packets from PC0 to PC1

It will select the shortest path to reach the destination

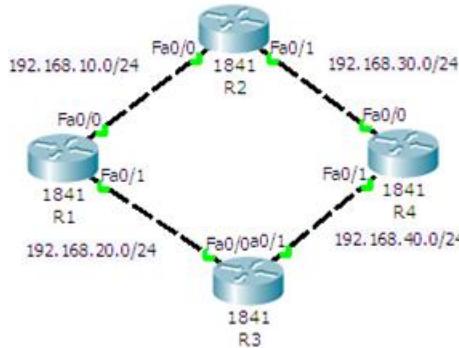
## Configuring Routing with the CLI

### Static routing

Static routing is the no-brainer method for configuring routing even though it requires more work. With Packet Tracer, static routing can be configured using the GUI alone. In this method, we configure a router with a destination and a gateway to reach it. So, each router in a topology should know the means to reach all destinations in the network, which requires manual work. Similarly, if a router is added or removed from the topology, all routers must be manually updated to reflect this.

### Static routing with GUI

Even if you do not know Cisco commands, this feature of Packet Tracer comes in handy. For this exercise, we will be using the topology shown in the following screenshot:

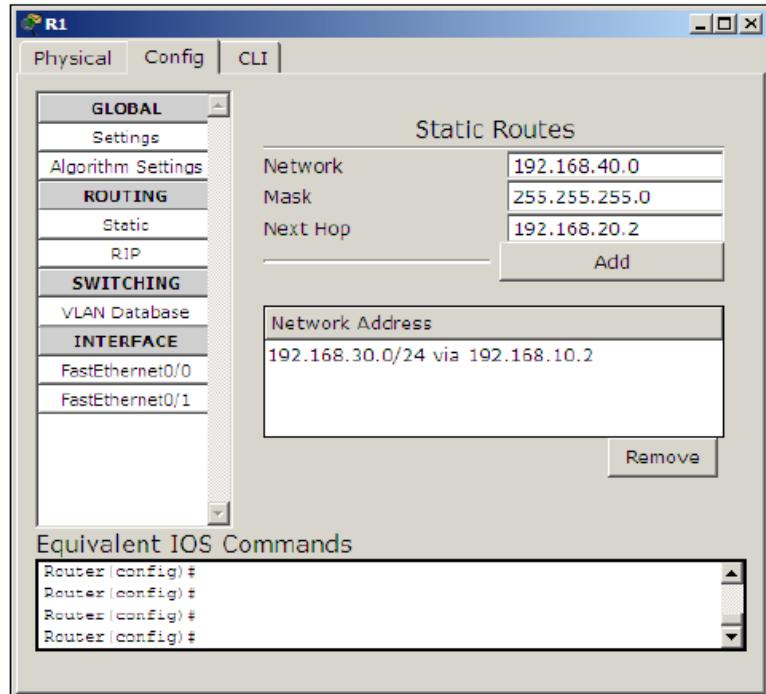


This network has four routers in a ring topology, with no PCs or loopback interfaces. Because we will be using only the GUI here, configuration will be kept to a minimum. The topology can be configured by performing the following steps:

1. Click on a router icon, go to the **Config** tab, select an interface, and configure the IP address. Make sure that you select the **On** checkbox in this section to bring the port state up. For this example, we'll be using the following IP addresses:

Router	Interface	IP Address
R1	FastEthernet0/0	192.168.10.1
	FastEthernet0/1	192.168.20.1
R2	FastEthernet0/0	192.168.10.2
	FastEthernet0/1	192.168.30.1
R3	FastEthernet0/0	192.168.20.2
	FastEthernet0/1	192.168.40.1
R4	FastEthernet0/0	192.168.30.2
	FastEthernet0/1	192.168.40.2

2. Under the **ROUTING** section, click on **Static**. The following screenshot is displayed:



- The following settings will be used for configuring static routing using the GUI. The concept here is to enter all routes that are not directly connected to a router and a gateway IP that belongs to a network that is directly connected.

Device	Network/Mask	Next Hop
R1	192.168.30.0 / 255.255.255.0	192.168.10.2
	192.168.40.0 / 255.255.255.0	192.168.20.2
R2	192.168.20.0 / 255.255.255.0	192.168.10.1
	192.168.40.0 / 255.255.255.0	192.168.30.2
R3	192.168.10.0 / 255.255.255.0	192.168.20.1
	192.168.30.0 / 255.255.255.0	192.168.40.2
R4	192.168.10.0 / 255.255.255.0	192.168.30.1
	192.168.20.0 / 255.255.255.0	192.168.40.1

- Now use simple PDU and test the connectivity between all of the routers. Then use the simulation mode to find the route taken by the packets.
- How about taking a look at the routing table? For this, too, the GUI has an option; click on the inspect icon or press *I* and select a router. A table containing four routes will appear for each router, as shown in the following screenshot:

Routing Table for R1					
Type	Network	Port	Next Hop IP	Metric	
C	192.168.10.0/24	FastEthernet0/0	---	0/0	
C	192.168.20.0/24	FastEthernet0/1	---	0/0	
S	192.168.30.0/24	---	192.168.10.2	1/0	
S	192.168.40.0/24	---	192.168.20.2	1/0	

But we configured only two routes, so why four? The extra two routes are the subnets of the directly-connected links.

In this topology, even though there is an alternate route to each network, only one route is used because this is how we have configured it. We'll learn more about having more than one route in the *Load Sharing* section.

## Static routing with the CLI

The configuration and the topology will be same in this section. We'll only see the commands required for one device. The topology can be configured by performing the following steps:

1. Assign IP addresses to the interfaces on each router using the following commands:

```
R1(config)#interface FastEthernet0/0
R1(config-if)#ip address 192.168.10.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#interface FastEthernet0/1
R1(config-if)#ip address 192.168.20.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#exit
```

2. Configure static routing with the ip route command, using the following syntax:

**R1(config)#ip route <Destination Prefix> <Destination prefix mask> <Gateway IP>**

- a. For router **R1**, the following commands are used:

```
R1(config)#ip route 192.168.30.0 255.255.255.0 192.168.10.2
R1(config)#ip route 192.168.40.0 255.255.255.0 192.168.20.2
```

Use simple PDU to test the connectivity. If you get message indicating a failure, switch to simulation mode and see which router is incorrectly configured.

## Dynamic routing protocols

When we learned about static routing we found that a lot of manual configuration was involved and a change to the topology also required manual configuration changes. Dynamic protocols work by advertising routes to each other.

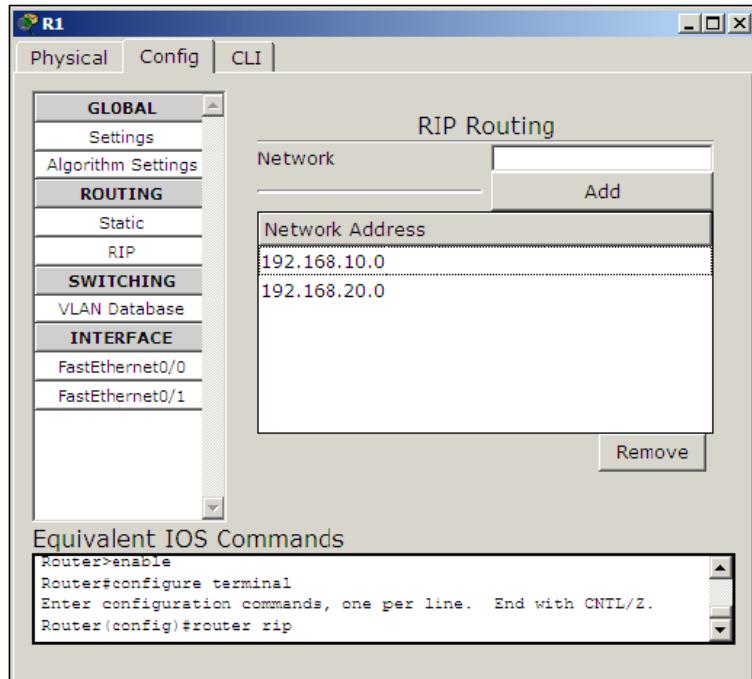
The configuration is the opposite of static routing; here, we enable dynamic routing on the required interfaces. The routing protocol then forms "neighborship" with other routers and sends them the directly-connected routes and other received routes. In this way, all routers exchange updates with one another. When a topology change occurs, those updates are also sent out by routers that learn about this loss of connectivity.

## Configuring RIP with the GUI

Packet Tracer offers a GUI to configure a dynamic routing protocol called **RIP (Routing Information Protocol)**. This GUI section is similar to the static routing section. It has only one textbox for entering the network address of the directly connected network.

You may think that the rest of the configuration is similar to the **Static** configuration, but it isn't. Whereas in the static configuration we entered routes of other routers, in RIP, we enter the network IP addresses of the router's interfaces. By doing this, you are enabling that routing protocol on a particular interface. To configure dynamic routing with the GUI, perform the following steps:

1. Create the same four-router topology we used previously and assign the same IP addresses through the **Config** tab.
2. Click on **RIP**—now, configuring this is very easy, with each router requiring only the **Network IP** of its own interfaces, as shown in the following screenshot:



3. Enter the following network IP addresses:

Device	RIP Network
R1	192.168.10.0
	192.168.20.0
R2	192.168.10.0
	192.168.30.0
R3	192.168.20.0
	192.168.40.0
R4	192.168.30.0
	192.168.40.0

4. Once the topology is configured, use the simple PDU to check for connectivity. Let's check for two indirectly connected routers (**R1** and **R4** or **R2** and **R3**). Once the connection is successful, let's see how dynamic routing works on topology changes.
5. Use the delete tool and remove either the link between **R1** and **R2** or the link between **R1** and **R3**. Use the simulation mode and test connectivity with the simple PDU. You'll find that the packet takes the alternate, longer route and succeeds in reaching the destination.

If you have tried step 5 of the static routing topology, the packet would've failed as we did not enter any alternate gateway to each destination network. This is the biggest advantage of using a dynamic routing protocol.

### Configuring RIP with the CLI

Let's do the same thing using the CLI tab. The commands are very simple and if you have noticed the **Equivalent IOS Commands** section under the **Config** tab, you'll know them already. To configure dynamic routing by using the **CLI** tab, perform the following steps:

1. Use the same commands used in the **Static** section to assign IP addresses to the interfaces.
2. Then, from the global configuration mode, enter into the config mode of RIP by issuing the following command:

**R1(config)#router rip**

3. Use the network command, followed by the network IP address. For the device **R1**, use the following commands:  
**R1(config-router)#network 192.168.10.0**  
**R1(config-router)#network 192.168.20.0**
4. Configure all the other routers in the same way. Use the simple PDU to test the connectivity.

**Result**

Thus the distance vector and link state routing algorithm has been simulation.

## Ex. No: 11

### IMPLEMENTATION OF IPV4 AND IPV6

#### Aim

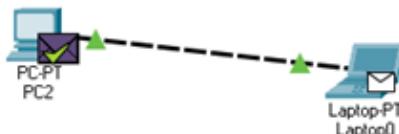
To implement IPv4 and IPv6.

#### Procedure

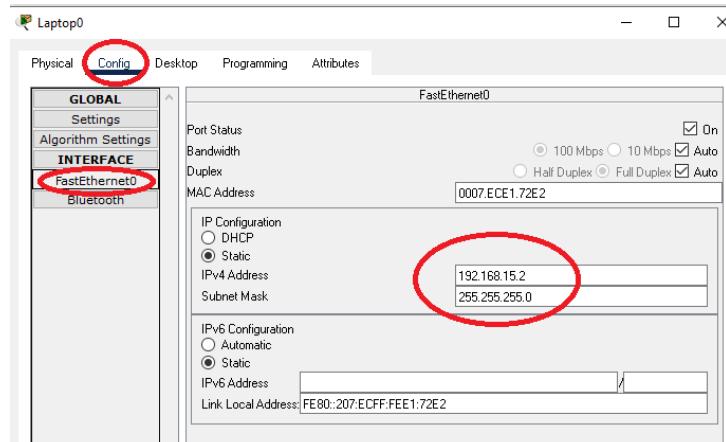
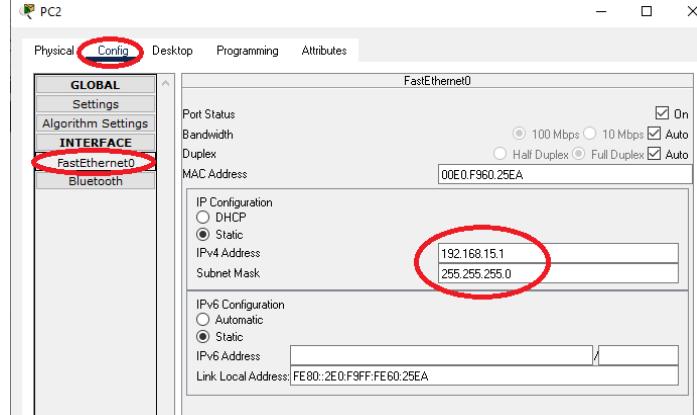
##### Creating a simple topology using IP

Now that you're familiar with the GUI of Packet Tracer, you can create your first network topology by carrying out the following steps:

1. From the network component box, click on **End Devices** and drag-and-drop a **Generic PC** icon and a **Generic laptop** icon into the Workspace.
2. Click on **Connections**, then click on **Copper Cross-Over**, then on **PC2**, and select **FastEthernet**. After this, click on **Laptop0** and select **FastEthernet**. The link status LED should show up in green, indicating that the link is up.



3. Click on the PC, go to the **Desktop** tab, click on **IP Configuration**, and enter an IP address and subnet mask. In this topology, the default gateway and DNS server information is not needed as there are only two end devices in the network.
4. Close the window, open the laptop, and assign an IP address to it in the same way. Make sure that both of the IP addresses are in the same subnet. We'll be learning



5. Close the IP Configuration box, open the command prompt, and ping the IP address of the device at the end to check connectivity.

```

Command Prompt

Packet Tracer PC Command Line 1.0
C:\>PING 192.168.15.2

Pinging 192.168.15.2 with 32 bytes of data:

Reply from 192.168.15.2: bytes=32 time=4ms TTL=128
Reply from 192.168.15.2: bytes=32 time=2ms TTL=128
Reply from 192.168.15.2: bytes=32 time=2ms TTL=128
Reply from 192.168.15.2: bytes=32 time=2ms TTL=128

Ping statistics for 192.168.15.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 4ms, Average = 2ms

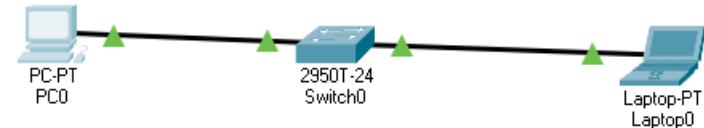
C:\>

```

### Creating a simple topology With Switch

Add an Ethernet switch to this topology so that more than two end devices can be connected, by performing the following steps:

1. Click on **Switches** from the device-type selection box and insert any switch (except **Switch-PT Empty**) into the workspace.
2. Remove the link between the PC and the laptop using the delete tool from the common tools bar.
3. Choose the **Copper Straight-Through** cable and connect the PC and laptop with the switch. At this point, the link indicators on the switch are orange in color because the switchports are undergoing the listening and learning states of the **Spanning Tree Protocol (STP)**.



```

C:\>ping 192.168.15.2

Pinging 192.168.15.2 with 32 bytes of data:

Reply from 192.168.15.2: bytes=32 time=1ms TTL=128
Reply from 192.168.15.2: bytes=32 time<1ms TTL=128
Reply from 192.168.15.2: bytes=32 time<1ms TTL=128
Reply from 192.168.15.2: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.15.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

```

4. Once the link turns green, as shown in the previous screenshot, ping again to check the connectivity.
5. To save this topology, navigate to **File | Save As** and choose a location. The topology will be saved with a .pkt extension, with the devices in the same state.

### Result

Thus the IPv4 and IPv6 has been implemented.

## Ex. No: 12

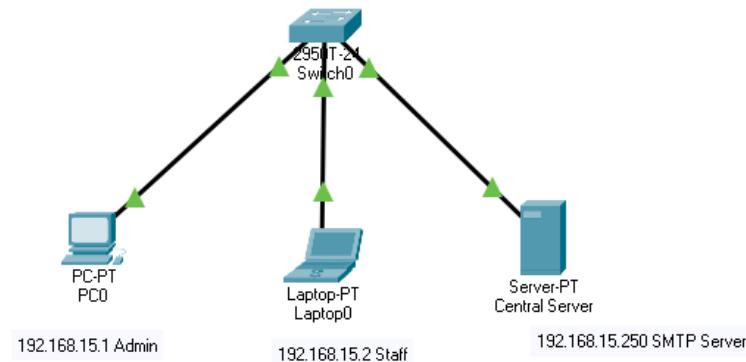
### IMPLEMENTATION OF SMTP

#### Aim

To implement SMTP.

#### Procedure

##### SMTP Configuration



**Step: 1:** Create a Networks with following devices

PC0 → IP Address → 192.168.15.1

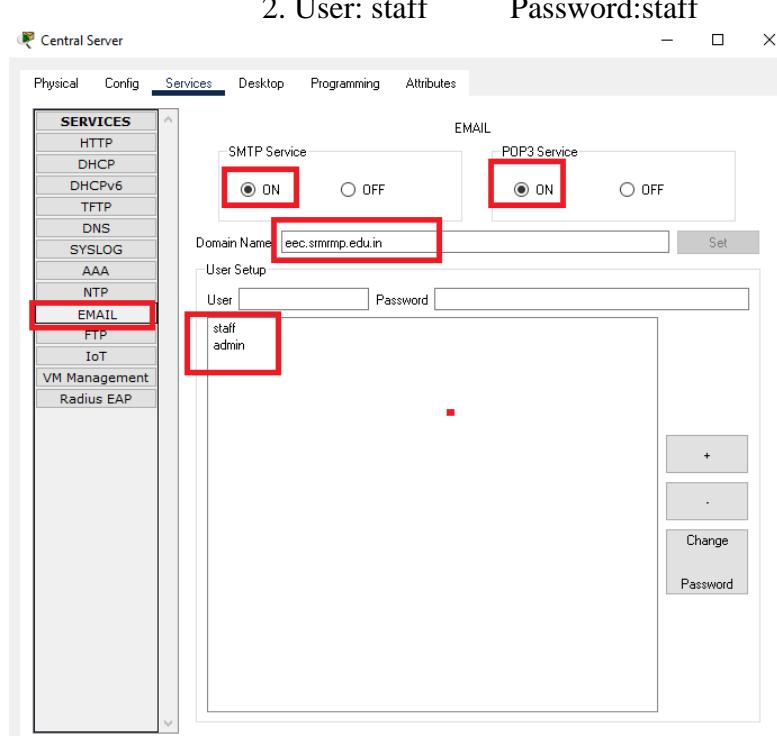
Laptop0 → IP Address → 192.168.15.2

Server-PT (Central Server) → IP Address → 192.168.15.250

**Step: 2:** Connect all the devices with 2950-T24 Switch0 to create a network.

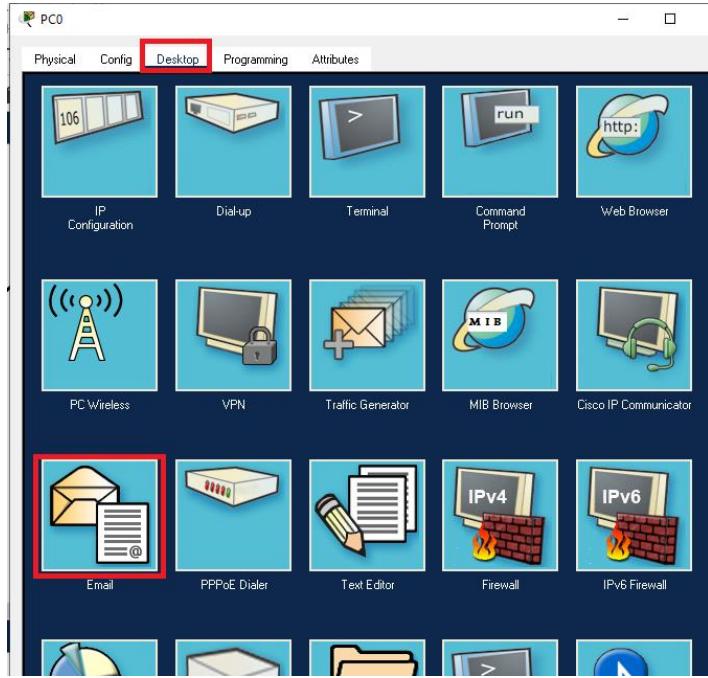
**Step: 3:** Configure SMTP in Central Server

In Central Server → Services → Email → Set Your Domain Name Eg: eec.srmrmp.edu.in → Add Multiple Users with their password, EG: 1. User: Admin      Password:Admin  
2. User: staff      Password:staff



**Step: 4:** Configure Email Settings in PC0

PC0 → Desktop → Email

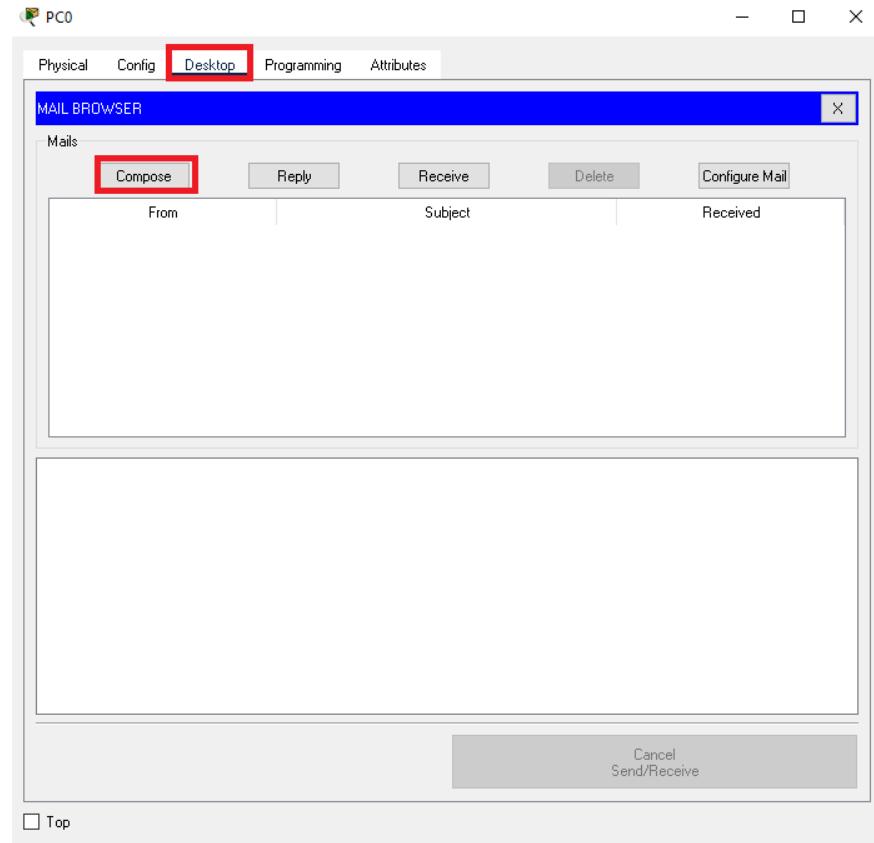


### All Credentials are related to Admin

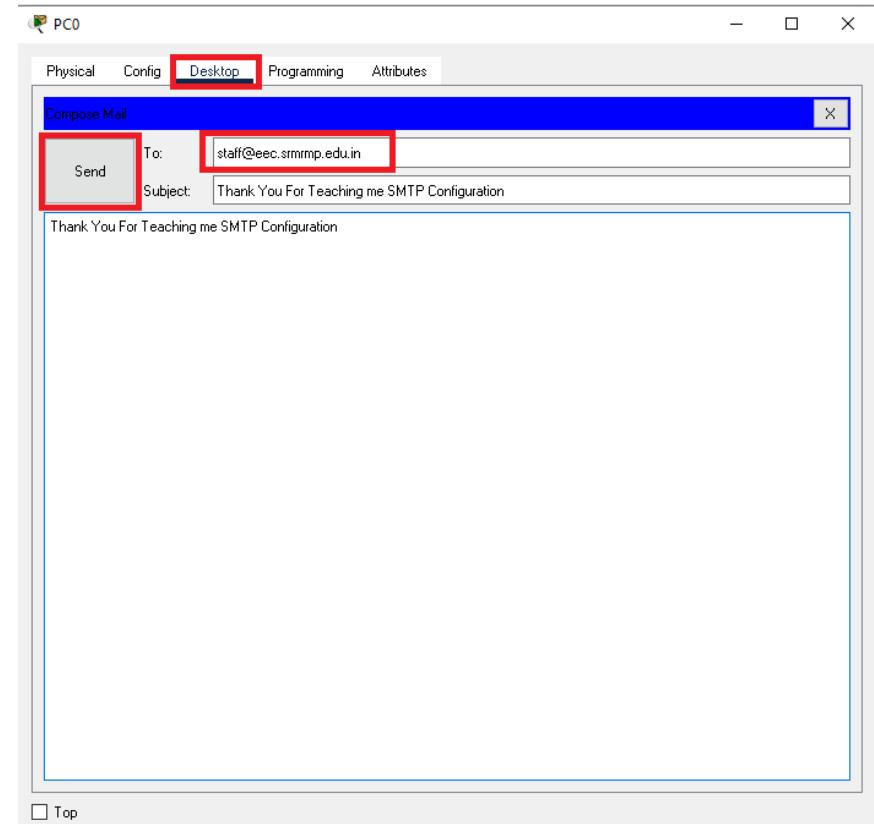
Your Name : admin  
Email Address : [admin@eec.srmrmp.edu.in](mailto:admin@eec.srmrmp.edu.in)  
Incoming Mail Server : 192.168.15.250  
Outgoing Mail Server : 192.168.15.250  
User Name : admin  
Password : admin

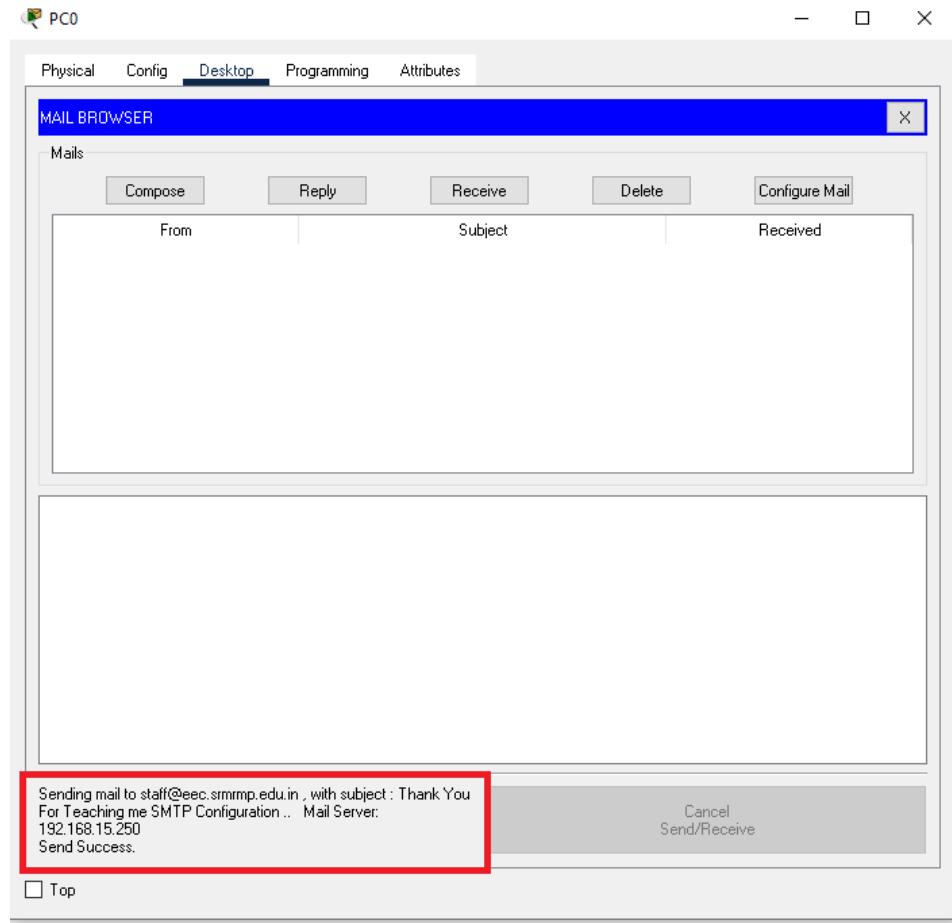
A screenshot of the 'Configure Mail' dialog box. The title bar says 'Configure Mail'. The top menu bar has tabs: Physical, Config, Desktop (highlighted with a red box), Programming, and Attributes. The dialog is divided into sections: User Information (Your Name, Email Address), Server Information (Incoming Mail Server, Outgoing Mail Server), and Logon Information (User Name, Password). A large rectangular area containing all these input fields is highlighted with a red box. At the bottom left is a 'Save' button, which is also highlighted with a red box. At the bottom right are 'Clear' and 'Reset' buttons.

Compose the mail with subject and content.



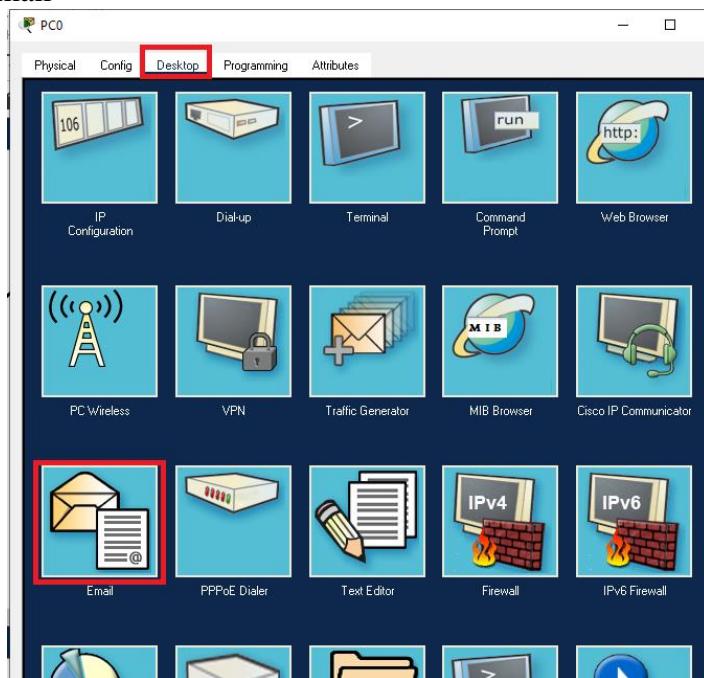
Type the address to send an Email





### Step 5: Configure Email Settings in Labtop0

Labtop0→Desktop→Email

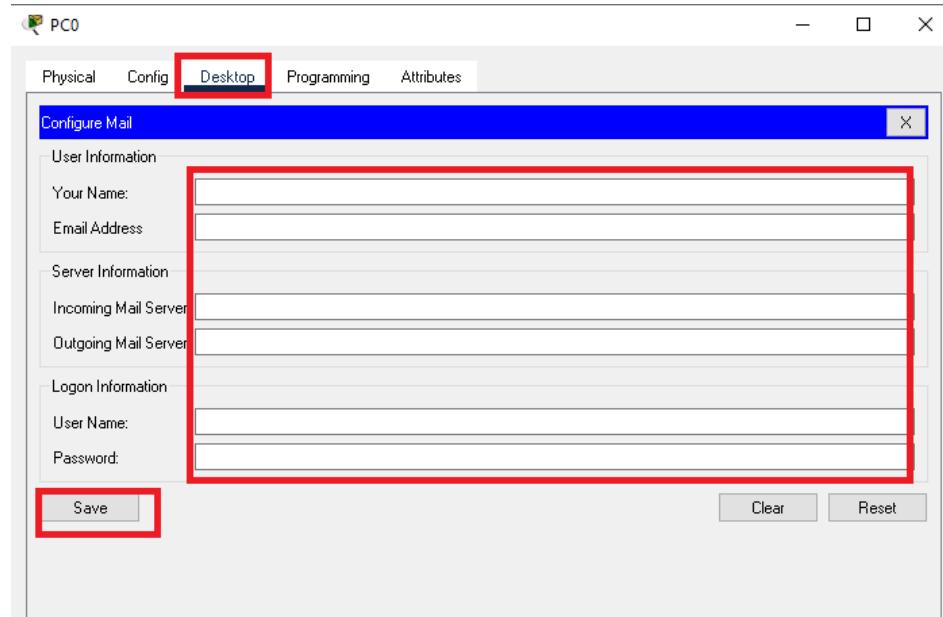


#### All Credentials are related to Staff

Your Name : Staff  
 Email Address : [staff@eec.srmrmp.edu.in](mailto:staff@eec.srmrmp.edu.in)  
 Incoming Mail Server : 192.168.15.250  
 Outgoing Mail Server : 192.168.15.250  
 User Name : staff

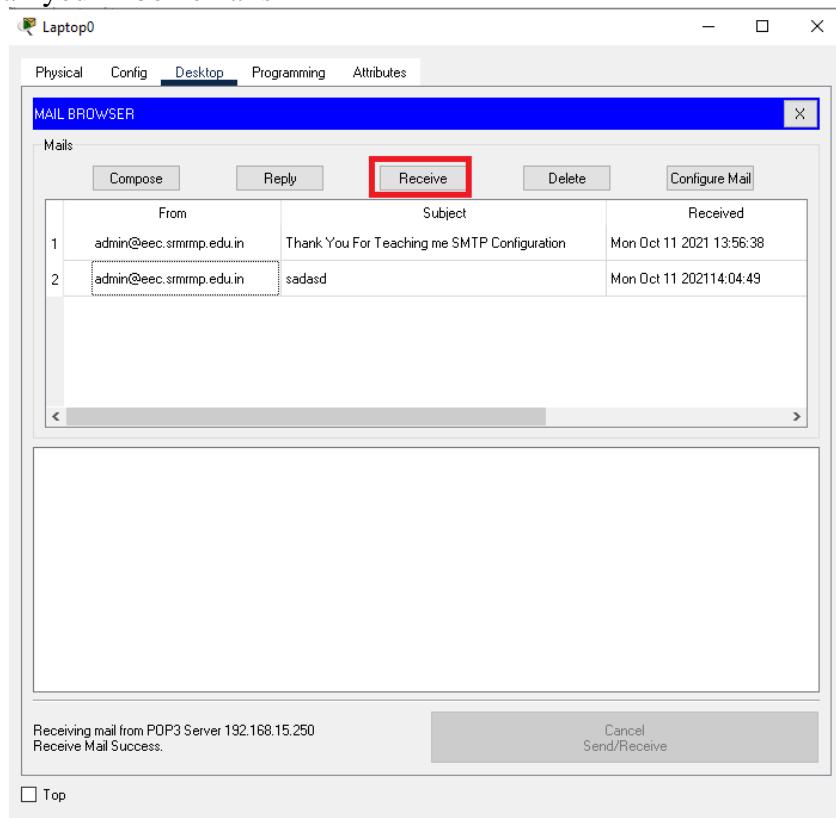
Password

: staff

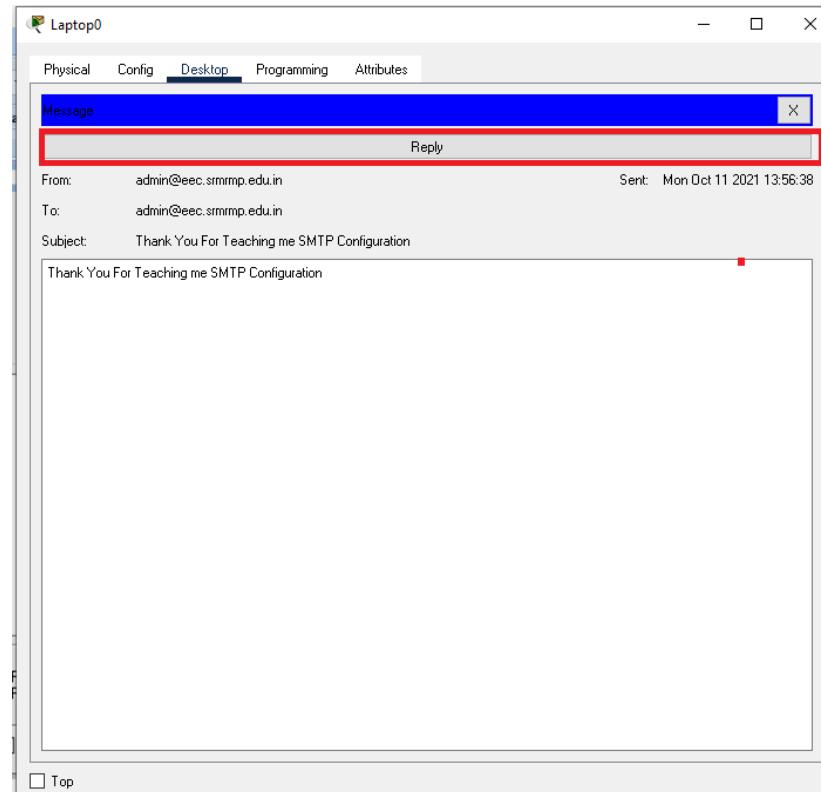


Laptop0 → Desktop → Email → Receive

Now You can see all your inbox emails



If you want to send a reply, Click Reply



## Result

Thus the SMTP has been implemented.