

**GOVERNMENT COLLEGE OF ENGINEERING**  
**SRIRANGAM, TRICHY – 620 012.**



**B.E.COMPUTER SCIENCE AND ENGINEERING**

**V – SEMESTER**

**(R-2017 Regulations)**

**2021-2022**

***CS8581– Networks Laboratory***

**Name :**

**Reg.No :**

# **GOVERNMENT COLLEGE OF ENGINEERING SRIRANGAM, TRICHY – 620 012.**

**(Approved by AICTE and Affiliated to Anna University, Chennai)**



**Anna University Register No:**

## **CERTIFICATE**

Certified that this is the bonafide record of work done by  
Mr./Ms. \_\_\_\_\_ of V Semester B.E. Computer Science and  
Engineering in the CS8581 – NETWORKS LABORATORY during the academic  
year 2021 – 2022.

**Faculty in-charge**

**Head of the Department**

Submitted for the Anna University Practical Examination held on .....

**Internal Examiner**

**External Examiner**

## INDEX

EX.NO.	NAME OF THE EXERCISE	DATE	SIGN
1.	Study of basic Network commands and capturing of PDUs using network protocol analyser		
2.	HTTP Web client program to download web page using TCP Sockets		
3.(a)	Implementation of Echo server and client using TCP sockets		
3.(b)	Implementation of chat application using TCP sockets		
3.(c)	File transfer using TCP sockets		
4.	Simulation of DNS using UDP sockets		
5.	Simulation of ARP/RARP protocols		
6.(a)	Study of Network Simulator (NS) – Packet Tracer		
6.(b)	Simulation of congestion control algorithm using Network Simulator		
7.	Study of TCP/UDP performance using Network Simulator		
8.(a)	Simulation of Distance Vector Routing algorithm		
8.(b)	Simulation of Link State Routing algorithm		
9.	Performance evaluation of Routing protocols using Network Simulator		
10.	Simulation of Error Correction using CRC		

<b>EX.NO:1</b>	<b>Study of basic Network commands and capturing of PDUs using Network Protocol Analyzer</b>

### Aim

To Learn and use commands like tcpdump, netstat, ifconfig, nslookup and traceroute.

To Capture ping and traceroute PDUs using a network protocol analyzer and examine.

#### 1.tcpdump (windump for windows)

#### Description

tcpdump prints out a description of the contents of packets on a network interface that match the boolean expression. It can also be run with the -w flag, which causes it to save the packet data to a file for later analysis, and/or with the -r flag, which causes it to read from a saved packet file rather than to read packets from a network interface. In all cases, only packets that match expression will be processed by tcpdump.

install windump

#### C:\>windump -i

windump version 3.9.5, based on tcpdump version 3.9.5

WinPcap version 4.1.3 (packet.dll version 4.1.0.2980), based on libpcap version 1.0 branch

1\_0\_rel0b (20091008)

Usage: windump [-aAdDeflLnNOpqRStuUvxX] [-B size] [-c count] [-C file\_size]

[-E algo:secret] [-F file] [-i interface] [-M secret]

[-r file] [-s snaplen] [-T type] [-w file]

[-W filecount] [-y datalinktype] [-Z user]

[ expression ]

#### C:\>windump -D

1.\Device\NPF\_{F0DC9F45-9A13-4397-8076-2D07E7C315D0} (Realtek Ethernet Controller)

2.\Device\NPF\_{AA490167-5FD4-433F-B3AC-72DD4985CA8B} (Oracle)

#### C:\>windump -i 1 -w c:\test\sample

windump: listening on \Device\NPF\_{F0DC9F45-9A13-4397-8076-2D07E7C315D0}

278 packets captured

288 packets received by filter

0 packets dropped by kernel

#### C:\>windump -i 1 host google.com

## Options

### **-A**

Print each packet (minus its link level header) in ASCII. Handy for capturing web pages.

### **-c**

Exit after receiving count packets.

### **-C**

Before writing a raw packet to a savefile, check whether the file is currently larger than file\_size and, if so, close the current savefile and open a new one. Savefiles after the first savefile will have the name specified with the -w flag, with a number after it, starting at 1 and continuing upward. The units of file\_size are millions of bytes (1,000,000 bytes, not 1,048,576 bytes).

### **-d**

Dump the compiled packet-matching code in a human readable form to standard output and stop.

### **-dd**

Dump packet-matching code as a C program fragment.

### **-ddd**

Dump packet-matching code as decimal numbers (preceded with a count).

### **-D**

Print the list of the network interfaces available on the system and on which tcpdump can capture packets. For each network interface, a number and an interface name, possibly followed by a text description of the interface, is printed. The interface name or the number can be supplied to the -i flag to specify an interface on which to capture.

This can be useful on systems that don't have a command to list them (e.g., Windows systems, or UNIX systems lacking ifconfig -a); the number can be useful on Windows 2000 and later systems, where the interface name is a somewhat complex string.

The -D flag will not be supported if tcpdump was built with an older version of libpcap that lacks the pcap\_findalldevs() function.

### **-e**

Print the link-level header on each dump line.

### **-E**

Use spi@ipaddr:secret for decrypting IPsec ESP packets that are addressed to addr and contain Security Parameter Index value spi. This combination may be repeated with comma or newline separation.

### **-f**

Print 'foreign' IPv4 addresses numerically rather than symbolically (this option is intended to get around serious brain damage in Sun's NIS server --- usually it hangs forever translating non-local internet numbers).

**-F**

Use file as input for the filter expression. An additional expression given on the command line is ignored.

**-i**

Listen on interface. If unspecified, tcpdump searches the system interface list for the lowest numbered, configured up interface (excluding loopback). Ties are broken by choosing the earliest match.

**-l**

Make stdout line buffered. Useful if you want to see the data while capturing it. E.g.,  
``tcpdump -l | tee dat" or ``tcpdump -l >dat& tail -f dat".

**-L**

List the known data link types for the interface and exit.

**-m**

Load SMI MIB module definitions from file module. This option can be used several times to load several MIB modules into tcpdump.

**-M**

Use secret as a shared secret for validating the digests found in TCP segments with the TCP-MD5 option (RFC 2385), if present.

**-n**

Don't convert addresses (i.e., host addresses, port numbers, etc.) to names.

**-N**

Don't print domain name qualification of host names. E.g., if you give this flag then tcpdump will print ``nic" instead of ``nic.ddn.mil".

**-O**

Do not run the packet-matching code optimizer. This is useful only if you suspect a bug in the optimizer.

**-p**

Don't put the interface into promiscuous mode. Note that the interface might be in promiscuous mode for some other reason; hence, ``-p' cannot be used as an abbreviation for `ether host {local-hw-addr} or ether broadcast'.

**-q**

Quick (quiet?) output. Print less protocol information so output lines are shorter.

**-R**

Assume ESP/AH packets to be based on old specification (RFC1825 to RFC1829). If specified, tcpdump will not print replay prevention field. Since there is no protocol version field in ESP/AH specification, tcpdump cannot deduce the version of ESP/AH protocol.

**-r**

Read packets from file (which was created with the -w option). Standard input is used if file is ``-".

**-S**

Print absolute, rather than relative, TCP sequence numbers.

**-t**

Don't print a timestamp on each dump line.

**-tt**

Print an unformatted timestamp on each dump line.

**-ttt**

Print a delta (in micro-seconds) between current and previous line on each dump line.

**-tttt**

Print a timestamp in default format proceeded by date on each dump line.

**-u**

Print undecoded NFS handles.

**-U**

Make output saved via the -w option ``packet-buffered"; i.e., as each packet is saved, it will be written to the output file, rather than being written only when the output buffer fills.

The -U flag will not be supported if tcpdump was built with an older version of libpcap that lacks the pcap\_dump\_flush() function.

**-v**

When parsing and printing, produce (slightly more) verbose output. For example, the time to live, identification, total length and options in an IP packet are printed. Also enables additional packet integrity checks such as verifying the IP and ICMP header checksum.

When writing to a file with the -w option, report, every 10 seconds, the number of packets captured.

**-vv**

Even more verbose output. For example, additional fields are printed from NFS reply packets, and SMB packets are fully decoded.

**-vvv**

Even more verbose output. For example, telnet SB ... SE options are printed in full. With -X Telnet options are printed in hex as well.

**-w**

Write the raw packets to file rather than parsing and printing them out. They can later be printed with the -r option. Standard output is used if file is ``-".

**-W**

Used in conjunction with the -C option, this will limit the number of files created to the specified number, and begin overwriting files from the beginning, thus creating a 'rotating' buffer.

In addition, it will name the files with enough leading 0s to support the maximum number of files, allowing them to sort correctly.

#### **-x**

When parsing and printing, in addition to printing the headers of each packet, print the data of each packet (minus its link level header) in hex. The smaller of the entire packet or snaplen bytes will be printed. Note that this is the entire link-layer packet, so for link layers that pad (e.g. Ethernet), the padding bytes will also be printed when the higher layer packet is shorter than the required padding.

#### **-xx**

When parsing and printing, in addition to printing the headers of each packet, print the data of each packet, including its link level header, in hex.

#### **-X**

When parsing and printing, in addition to printing the headers of each packet, print the data of each packet (minus its link level header) in hex and ASCII. This is very handy for analysing new protocols.

#### **-XX**

When parsing and printing, in addition to printing the headers of each packet, print the data of each packet, including its link level header, in hex and ASCII.

#### **-y**

Set the data link type to use while capturing packets to datalinktype.

#### **-Z**

Drops privileges (if root) and changes user ID to user and the group ID to the primary group of user.

This behavior can also be enabled by default at compile time.

## **2. netstat**

<b>Displays protocol statistics and current TCP/IP network connections.</b>	
<b>NETSTAT [-a] [-e] [-n] [-s] [-p proto] [-r] [interval]</b>	
<b>-a</b>	Displays all connections and listening ports. (Server-side connections are normally not shown).
<b>-e</b>	Displays Ethernet statistics. This may be combined with the -s option.
<b>-n</b>	Displays addresses and port numbers in numerical form.
<b>-p proto</b>	Shows connections for the protocol specified by proto; proto may be tcp or udp. If used with the -s option to display per-protocol statistics, proto may be tcp, udp, or ip.
<b>-r</b>	Displays the contents of the routing table.
<b>-s</b>	Displays per-protocol statistics. By default, statistics are shown for TCP, UDP and IP; the -p option may be used to specify a subset of the default.
<b>interval</b>	Redisplays selected statistics, pausing interval seconds between each display. Press CTRL+C to stop redisplaying statistics. If omitted, netstat will print the current configuration information once



## **netstat**

netstat - Print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships

### **3. Ifconfig (ipconfig for windows)**

#### **Name**

ifconfig - configure a network interface

#### **Synopsis**

ifconfig [-v] [-a] [-s] [interface]

ifconfig [-v] interface [atype] options | address ...

#### **Description**

Ifconfig is used to configure the kernel-resident network interfaces. It is used at boot time to set up interfaces as necessary. After that, it is usually only needed when debugging or when system tuning is needed.

If no arguments are given, ifconfig displays the status of the currently active interfaces. If a single interface argument is given, it displays the status of the given interface only; if a single -a argument is given, it displays the status of all interfaces, even those that are down. Otherwise, it configures an interface.

Ipconfig displays all current TCP/IP network configuration values and refreshes Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) settings. Used without parameters, ipconfig displays the IP address, subnet mask, and default gateway for all adapters.

### **4. nslookup**

The nslookup (which stands for name server lookup) command is a network utility program used to obtain information about internet servers. It finds name server information for domains by querying the Domain Name System. Most computer operating systems include a built-in command line program with the same name.

```
hp@hp-HP-280-G3-MT:~$ nslookup www.google.com
```

```
Server:          127.0.0.53
```

Address: 127.0.0.53#53

Non-authoritative answer:

Name: www.google.com

Address: 172.217.160.132

Name: www.google.com

Address: 2404:6800:4007:811::2004

## 5. Traceroute

### tracert

The tracert command is a Command Prompt command that's used to show several details about the path that a packet takes from the computer or device you're on to whatever destination you specify. You might also sometimes see the tracert command referred to as the trace route command or traceroute command.

## 6. ping

### ping

The ping command is a Command Prompt command used to test the ability of the source computer to reach a specified destination computer. The ping command is usually used as a simple way to verify that a computer can communicate over the network with another computer or network device. The ping command operates by sending Internet Control Message Protocol (ICMP) Echo Request messages to the destination computer and waiting for a response. How many of those responses are returned, and how long it takes for them to return, are the two major pieces of information that the ping command provides.

### Output: ipconfig

```
C:\Users\god>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

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

Wireless LAN adapter Local Area Connection* 1:

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

Wireless LAN adapter Local Area Connection* 2:

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

Wireless LAN adapter Wi-Fi:

    Connection-specific DNS Suffix . : 
    IPv6 Address. . . . . : 2409:4072:e9f:4a92:4141:aaa9:ad69:551c
    Temporary IPv6 Address. . . . . : 2409:4072:e9f:4a92:3468:b97c:5357:545d
    Link-local IPv6 Address . . . . . : fe80::4141:aaa9:ad69:551c%7
    IPv4 Address. . . . . : 192.168.43.185
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : fe80::9414:7aff:fe16:495a%7
                                192.168.43.1

Ethernet adapter Bluetooth Network Connection:

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

## Output: ipconfig/all

```
C:\Users\god>ipconfig/all

Windows IP Configuration

Host Name . . . . . : LAPTOP-6IENH4BK
Primary Dns Suffix . . . . . : 
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No

Ethernet adapter Ethernet:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . : 
Description . . . . . : Realtek PCIe GbE Family Controller
Physical Address. . . . . : B4-A9-FC-69-54-8B
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Local Area Connection* 1:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . : 
Description . . . . . : Microsoft Wi-Fi Direct Virtual Adapter
Physical Address. . . . . : C8-09-A8-EF-9E-27
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Local Area Connection* 2:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . : 
Description . . . . . : Microsoft Wi-Fi Direct Virtual Adapter #2
Physical Address. . . . . : CA-09-A8-EF-9E-26
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . : 
Physical Address. . . . . : 74-16-01-00-00-00
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
```

## Output: nslookup

cmd Command Prompt - nslookup

```
C:\Users\god>nslookup
Default Server:  dns.google
Address:  2001:4860:4860::8888

> www.facebook.com
Server:  dns.google
Address:  2001:4860:4860::8888

Non-authoritative answer:
Name:     star-mini.c10r.facebook.com
Addresses: 2a03:2880:f137:83:face:b00c:0:25de
           157.240.23.35
Aliases:  www.facebook.com

> _
```

## Output: ping (pinging a computer which is available in the local network)

```
C:\Users\god>ping 199.34.228.69

Pinging 199.34.228.69 with 32 bytes of data:
Reply from 199.34.228.69: bytes=32 time=433ms TTL=45
Reply from 199.34.228.69: bytes=32 time=653ms TTL=45
Reply from 199.34.228.69: bytes=32 time=669ms TTL=45
Reply from 199.34.228.69: bytes=32 time=683ms TTL=45

Ping statistics for 199.34.228.69:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 433ms, Maximum = 683ms, Average = 609ms

C:\Users\god>_
```

## Output: ping (pinging a computer which is not available in the local network)

```
C:\Users\god>ping 10.154.32.32

Pinging 10.154.32.32 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 10.154.32.32:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\Users\god>_
```

## Output: tracert

```
CA Command Prompt
Non-authoritative answer:
Name: www.nasoacademy.org
Address: 199.34.228.69

>
C:\Users\god>tracert 199.34.228.69

Tracing route to pages-custom-21.weebly.com [199.34.228.69]
over a maximum of 30 hops:

  0  *          *          *          Request timed out.
  1  *          *          *          Request timed out.
  2  59 ms     53 ms     59 ms     56.8.110.157
  3  478 ms    478 ms    421 ms    172.25.106.122
  4  320 ms    45 ms     53 ms    172.25.106.121
  5  57 ms     38 ms     52 ms    172.25.8.9
  6  *          *          *          Request timed out.
  7  *          *          *          Request timed out.
  8  *          *          *          Request timed out.
  9  60 ms     50 ms     77 ms    103.198.140.184
 10  401 ms    612 ms    611 ms    49.45.4.103
 11  613 ms    407 ms    610 ms    103.198.140.15
 12  *          *          *          Request timed out.
 13  *          1659 ms  1471 ms   ae-1-3501.edge7.LosAngeles1.Level3.net [4.69.219.41]
 14  2027 ms   *          *          64.125.15.88
 15  *          *          *          Request timed out.
 16  1848 ms   1759 ms   1847 ms   ae2.cs2.sjc2.us.eth.zayo.com [64.125.28.196]
 17  1741 ms   1592 ms   1548 ms   ae27.cr2.sjc2.us.zip.zayo.com [64.125.30.233]
 18  2189 ms   *          *          ae11.mpr4.sfo3.us.zip.zayo.com [64.125.24.226]
 19  *          1817 ms   3089 ms   64.124.5.2.IPYX-128160-025-ZY0.zip.zayo.com [64.124.5.2]
 20  *          1829 ms   1808 ms   core1-a-edge1-a.sf2p.weebly.net [74.115.50.158]
 21  1827 ms   1696 ms   1761 ms   pages-custom-21.weebly.com [199.34.228.69]
 22

Trace complete.

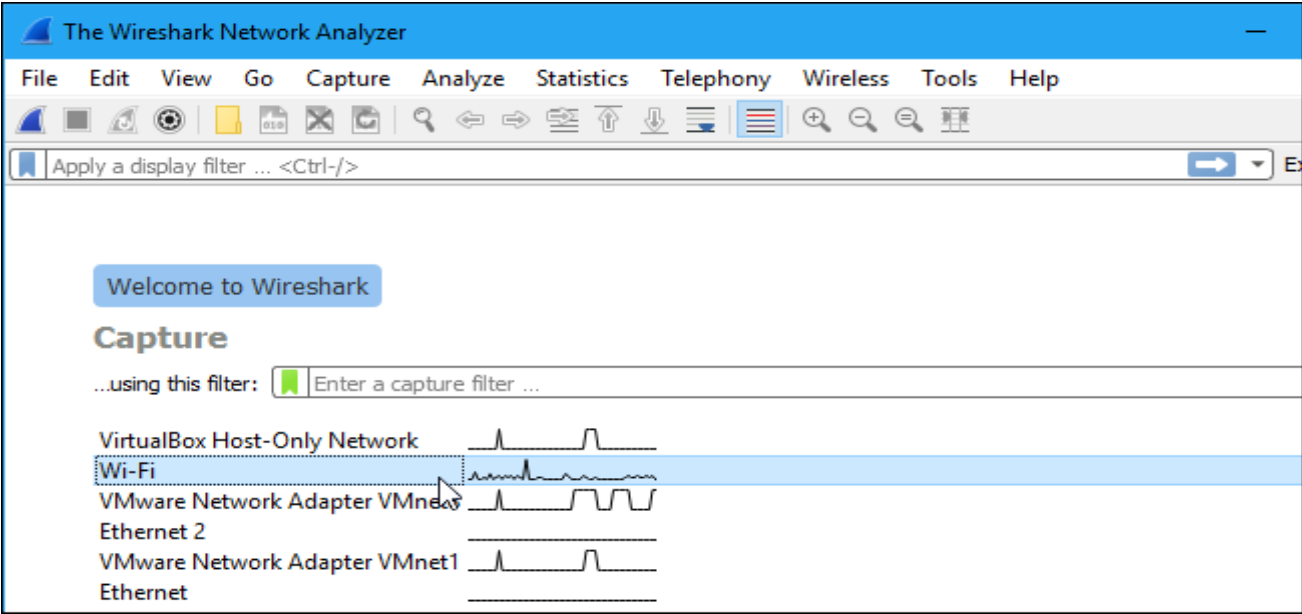
C:\Users\god>
```

## Wireshark

Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and displays them in human-readable format. Wireshark includes filters, color coding, and other features that let you dig deep into network traffic and inspect individual packets.

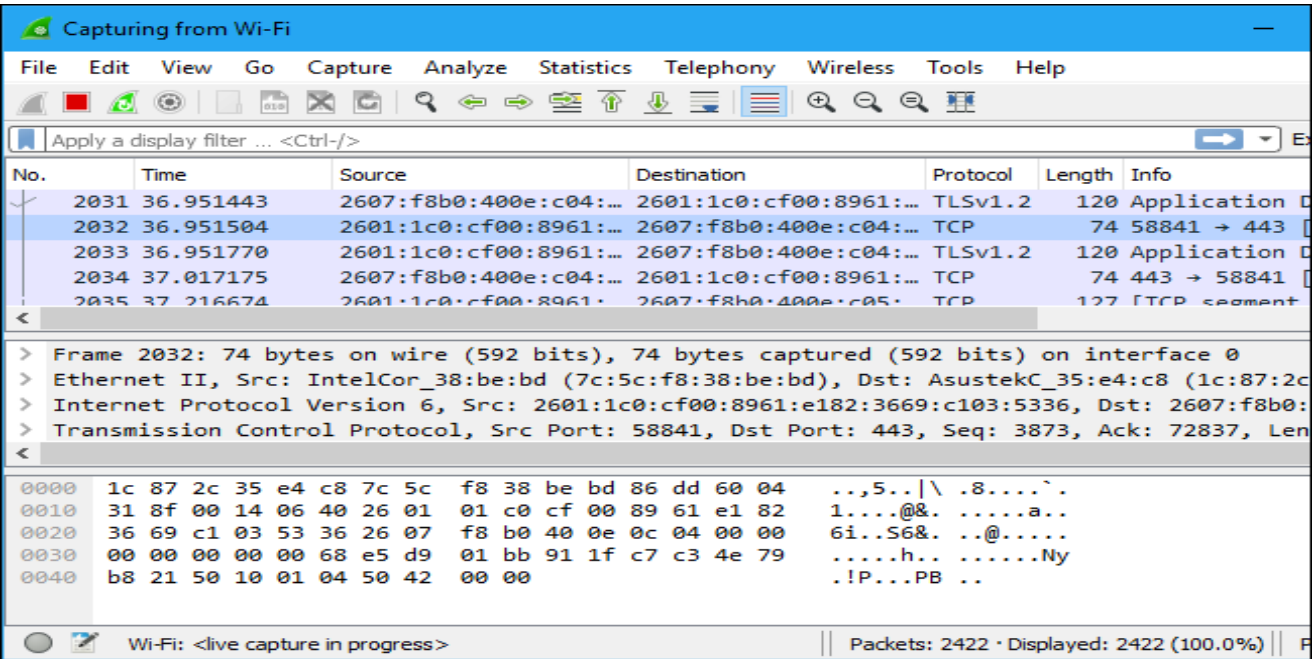
### Capturing Packets

As soon as you click the interface's name, you'll see the packets start to appear in real time.

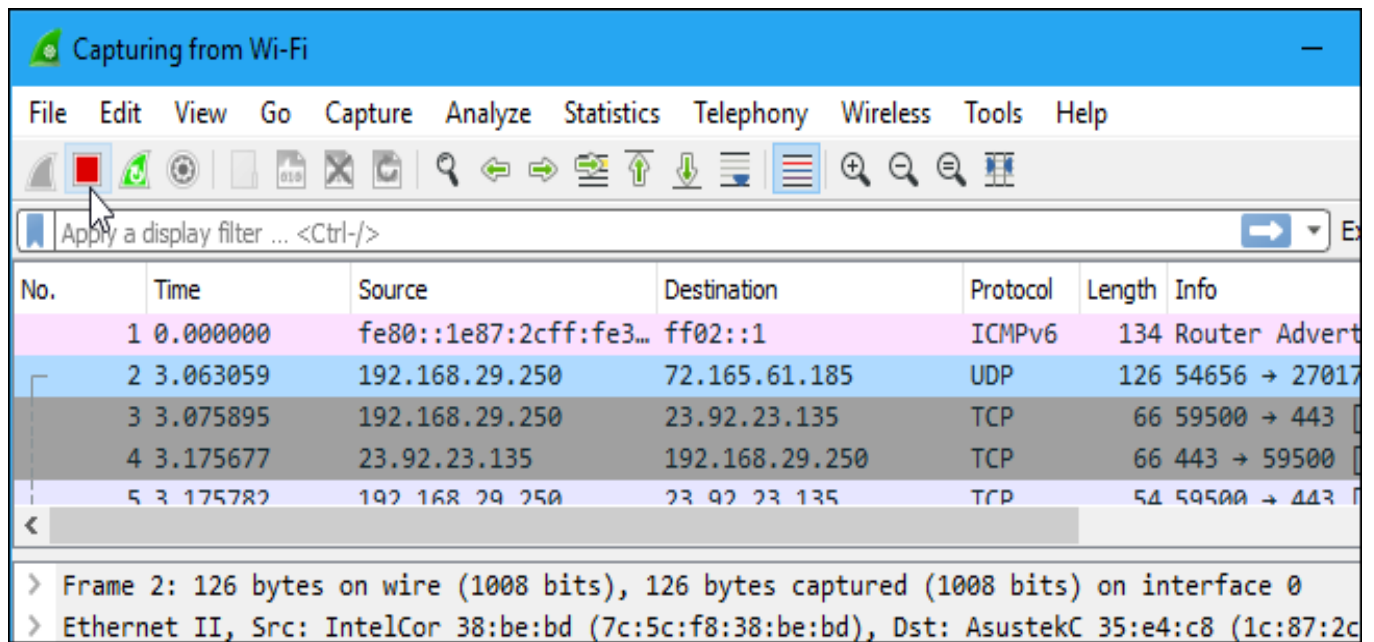


Wireshark captures each packet sent to or from your system.

If you have promiscuous mode enabled—it's enabled by default—you'll also see all the other packets on the network instead of only packets addressed to your network adapter. To check if promiscuous mode is enabled, click **Capture > Options** and verify the **—Enable promiscuous mode on all interfaces** checkbox is activated at the bottom of this window.



Click the red —Stop— button near the top left corner of the window when you want to stop capturing traffic.



## Result

Thus, the basic network commands were studied in detail and the outputs were verified. The PDUs were also captured and examined using a Network Protocol Analyzer (Wireshark).

<b>EX.NO:2</b>	<b>HTTP Web client program to download web page using TCP Sockets</b>

**Aim**

To write a HTTP web client program in java for downloading a web page using TCP sockets.

**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

```
import java.io.*;

import java.net.*;

public class HTTPwebclient

{

public static void main(String[] args)

{

String hostName = "www.sunnetwork.in";

int portNumber = 80;

try

{

Socket socket = new Socket(hostName, portNumber);

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

BufferedReader in =new BufferedReader(new

InputStreamReader(socket.getInputStream()));

out.println("GET / HTTP/1.1\nHost: www.sunnetwork.in\n\n");

String inputLine;

while ((inputLine = in.readLine()) != null)

{

System.out.println(inputLine);

}

}

catch (UnknownHostException e)

{

System.err.println("Don't know about host " + hostName);

System.exit(1);

}

catch (IOException e)

{

System.err.println("Couldn't get I/O for the connection to " + hostName);

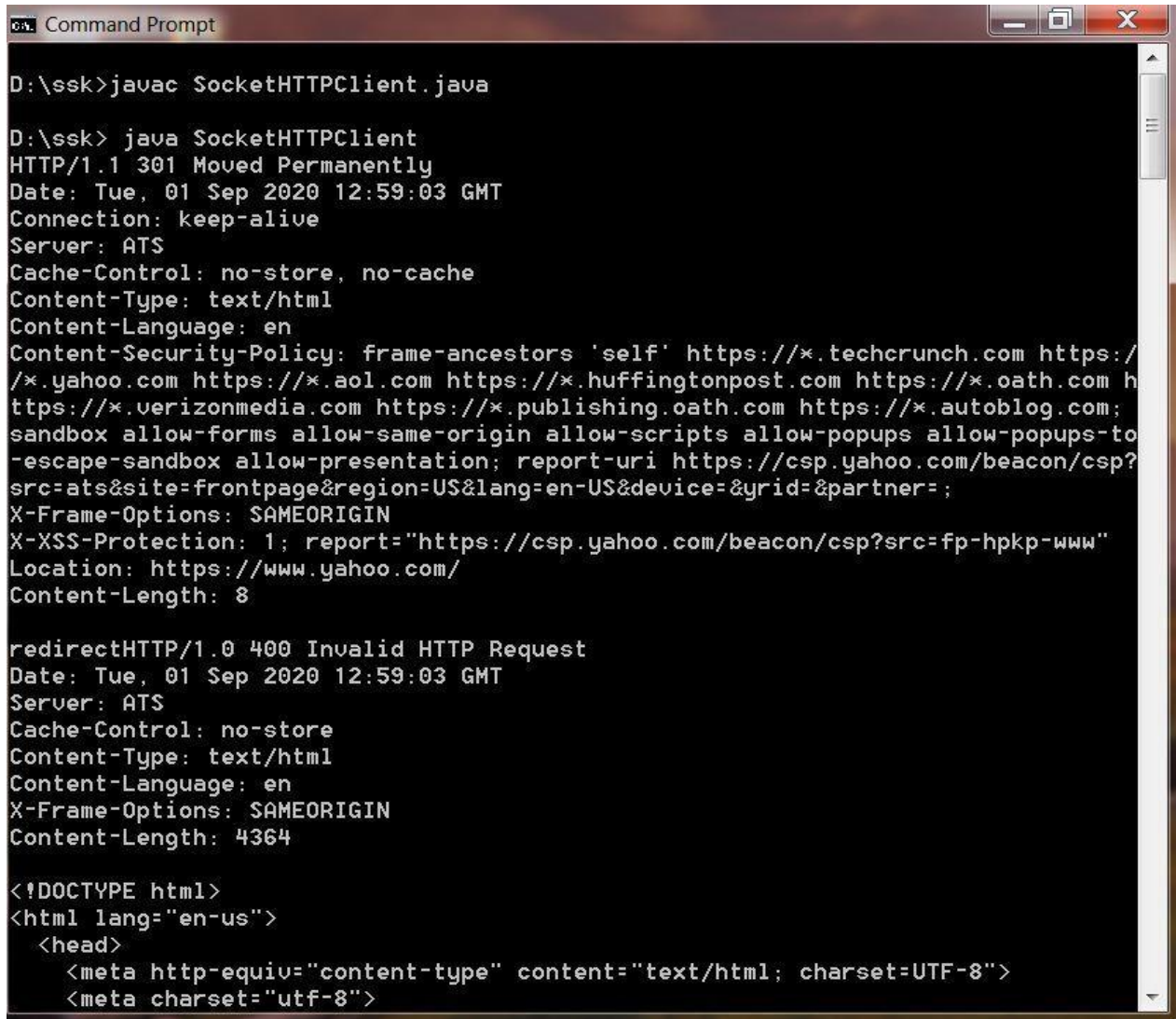
System.exit(1);

}

}}
```



## Output



```
D:\ssk>javac SocketHTTPClient.java

D:\ssk> java SocketHTTPClient
HTTP/1.1 301 Moved Permanently
Date: Tue, 01 Sep 2020 12:59:03 GMT
Connection: keep-alive
Server: ATS
Cache-Control: no-store, no-cache
Content-Type: text/html
Content-Language: en
Content-Security-Policy: frame-ancestors 'self' https://*.techcrunch.com https://
/*.yahoo.com https://*.aol.com https://*.huffingtonpost.com https://*.oath.com h
https://*.verizonmedia.com https://*.publishing.oath.com https://*.autoblog.com;
sandbox allow-forms allow-same-origin allow-scripts allow-popups allow-popups-to
-escape-sandbox allow-presentation; report-uri https://csp.yahoo.com/beacon/csp?
src=ats&site=frontpage&region=US&lang=en-US&device=&yrid=&partner=;
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; report="https://csp.yahoo.com/beacon/csp?src=fp-hpkip-www"
Location: https://www.yahoo.com/
Content-Length: 8

redirectHTTP/1.0 400 Invalid HTTP Request
Date: Tue, 01 Sep 2020 12:59:03 GMT
Server: ATS
Cache-Control: no-store
Content-Type: text/html
Content-Language: en
X-Frame-Options: SAMEORIGIN
Content-Length: 4364

<!DOCTYPE html>
<html lang="en-us">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <meta charset="utf-8">
```

## Result

Thus, a HTTP web client program was written in java to download a web page using TCP sockets.

**Aim**

To implement echo server and client using TCP sockets.

**Algorithm****Client Side**

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

**Server Side**

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

## Program

### Echo Server

```
import java.net.*;

import java.io.*;

class EchoServer1 {

    public static void main(String args[]) throws IOException {

        ServerSocket ss = null;

        try {

            ss = new ServerSocket(95);

        } catch (IOExceptionioe) {

            System.out.println("Error finding port");

            System.exit(1);

        }

        Socket soc = null;

        try {

            soc = ss.accept();

            System.out.println("Connection accepted at :" + soc);

        } catch (IOExceptionioe) {

            System.out.println("Server failed to accept");

            System.exit(1);

        }

        DataOutputStream dos = new DataOutputStream(soc.getOutputStream());

        BufferedReader br = new BufferedReader(new InputStreamReader(soc.getInputStream()));

        String s;

        System.out.println("Server waiting for message from the client");

        boolean quit = false;

        do {

            String msg = "";

            s = br.readLine();

            int len = s.length();
```

```

        if (s.equals("exit")) {
            quit = true;
        }

        for (int i = 0; i<len; i++) {
            msg = msg + s.charAt(i);
        }

        dos.write((byte) s.charAt(i));

    }

    System.out.println("From client :" + msg);

    dos.write(13);

    dos.write(10);

    dos.flush();

    } while (!quit);

    dos.close();

    soc.close();

    }

}

```

### **Echo Client:**

```

import java.net.*;

import java.io.*;

class EchoClient1 {

    public static void main(String args[]) throws IOException {

        Socket soc = null;

        String str = null;

        BufferedReader

        br = null;

        DataOutputStream dos = null;

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

        try {

            soc = new Socket(InetAddress.getLocalHost(), 95);

            br = new BufferedReader(new InputStreamReader(soc.getInputStream()));

```

```
        dos = new DataOutputStream(soc.getOutputStream());

    } catch (UnknownHostException e) {
System.out.println("Unknown Host");
System.exit(0);

    }

System.out.println("To start the dialog type the message in this client window \n Type exit to end");

boolean more = true;

    while (more) {

        str = kyrd.readLine();

dos.writeBytes(str);
dos.write(13);
dos.write(10);
dos.flush();

        String s, s1;

        s = br.readLine();

System.out.println("From server :" + s);

        if (s.equals("exit")) {

            break;

        }

    }

br.close();

dos.close();

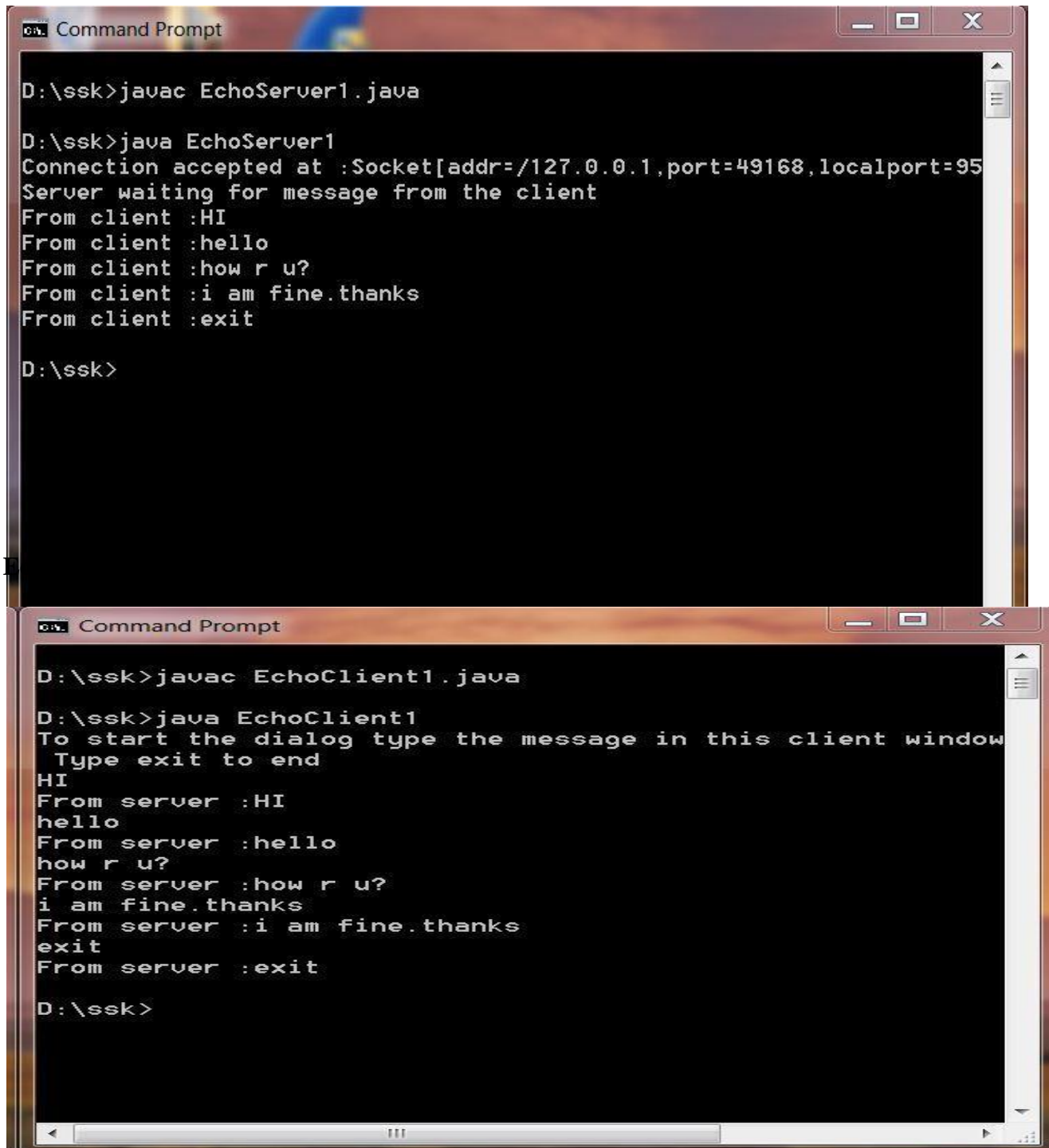
soc.close();

    }

}
```

## Sample Input / output

### Echo Server



The image displays two separate Windows Command Prompt windows. The top window shows the compilation and execution of an EchoServer1.java program. It receives client messages and echoes them back. The bottom window shows the compilation and execution of an EchoClient1.java program. It sends messages to the server and receives the echoed responses back.

```
D:\ssk>javac EchoServer1.java

D:\ssk>java EchoServer1
Connection accepted at :Socket[addr=/127.0.0.1,port=49168,localport=95
Server waiting for message from the client
From client :HI
From client :hello
From client :how r u?
From client :i am fine.thanks
From client :exit

D:\ssk>
```

```
D:\ssk>javac EchoClient1.java

D:\ssk>java EchoClient1
To start the dialog type the message in this client window
Type exit to end
HI
From server :HI
hello
From server :hello
how r u?
From server :how r u?
i am fine.thanks
From server :i am fine.thanks
exit
From server :exit

D:\ssk>
```

### Result

Thus, data from client to server is echoed back to the client.

**Aim**

To implement the chat application between client and server using TCP sockets.

**Algorithm****Server**

1. Start the program and create server and client sockets.
2. Use input streams to get the message from user.
3. Use output streams to send message to the client.
4. Wait for client to display this message and write a new one to be displayed by the server.
5. Display message given at client using input streams read from socket.
6. Stop the program.

**Client**

1. Start the program and create a client socket that connects to the required host and port.
2. Use input streams read message given by server and print it.
3. Use input streams; get the message from user to be given to the server.
4. Use output streams to write message to the server.
5. Stop the program.

## Program

### Chat Server

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

public class chatserver
{
    public static void main(String args[]) throws Exception
    {
        ServerSocket ss=new ServerSocket(2000);

        Socket sk=ss.accept();

        BufferedReadercin=new BufferedReader(new
        InputStreamReader(sk.getInputStream()));

        PrintStreamcout=new PrintStream(sk.getOutputStream());

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

        String s;

        while ( true )
        {
            s=cin.readLine();

            if (s.equalsIgnoreCase("END"))
            {
                cout.println("BYE");
                break;
            }

            System.out.print("Client : "+s+"\n");
            System.out.print("Server :");

            s=stdin.readLine();

            cout.println(s);
        }

        ss.close();
        sk.close();
        cin.close();
        cout.close();
    }
}
```



```
stdin.close();
```

```
}
```

```
}
```

### **Chat Client**

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

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

```
public class chatclient
```

```
{
```

```
    public static void main(String args[]) throws Exception
```

```
    {
```

```
        Socket sk=new Socket("127.0.0.1",2000);
```

```
        BufferedReader sin=new BufferedReader(new
```

```
        InputStreamReader(sk.getInputStream()));
```

```
        PrintStream sout=new PrintStream(sk.getOutputStream());
```

```
        BufferedReader stdin=new BufferedReader(new InputStreamReader(System.in));
```

```
        String s;
```

```
        while ( true )
```

```
        {
```

```
            System.out.print("Client :");
```

```
            s=stdin.readLine();
```

```
            sout.println(s);
```

```
            s=sin.readLine();
```

```
            System.out.print("Server :"+s+"\n");
```

```
            if ( s.equalsIgnoreCase("BYE") )
```

```
                break;
```

```
        }
```

```
        sk.close();
```

```
        sin.close();
```

```
        sout.close();
```

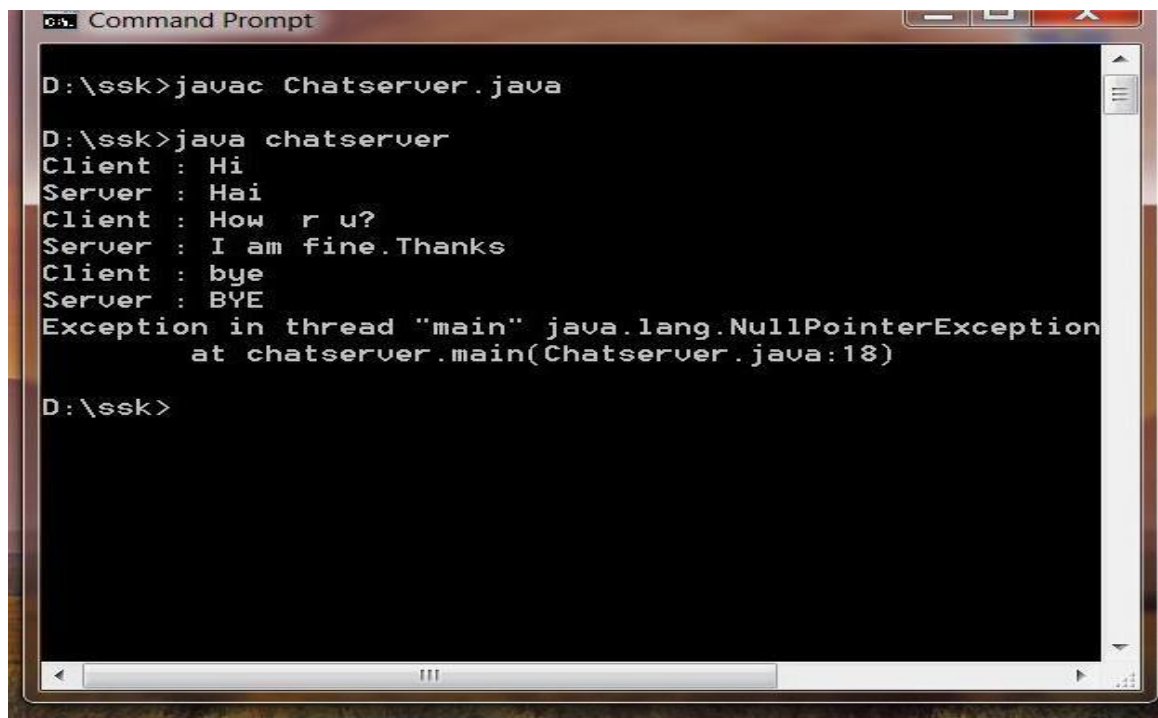
```
        stdin.close();
```

```
    }
```

```
}
```

## Output

### Chat Server

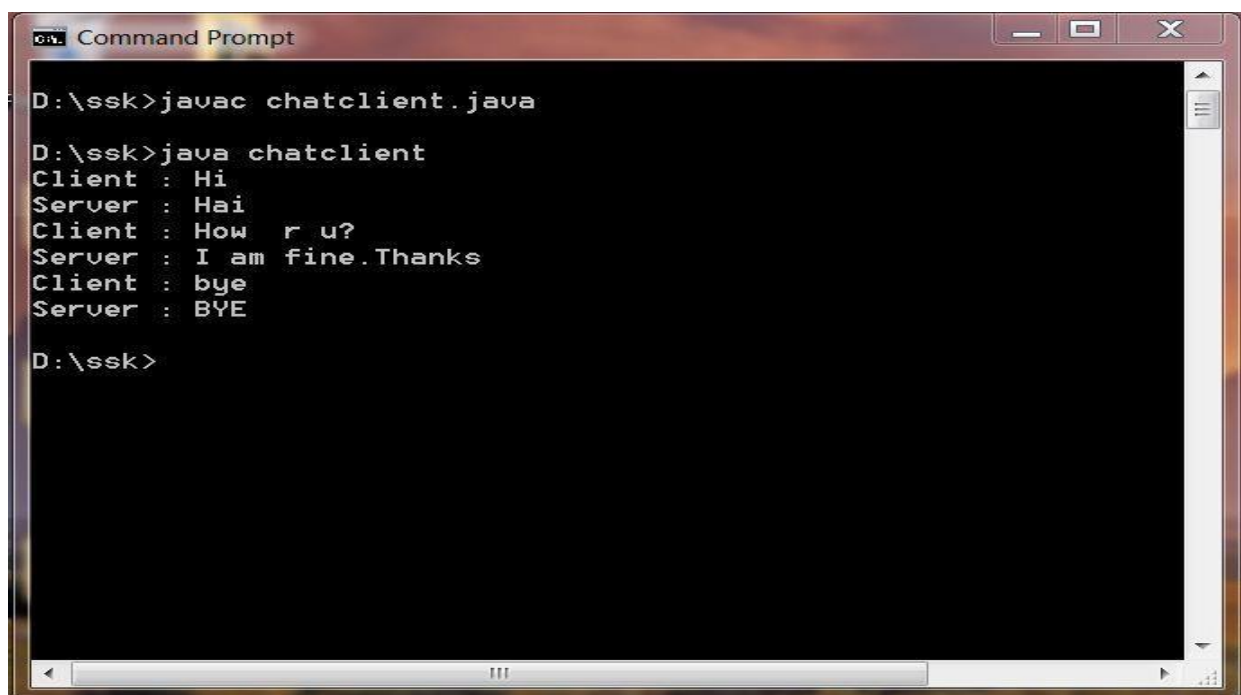


```
D:\ssk>javac Chatserver.java

D:\ssk>java chatserver
Client : Hi
Server : Hai
Client : How r u?
Server : I am fine.Thanks
Client : bye
Server : BYE
Exception in thread "main" java.lang.NullPointerException
        at chatserver.main(Chatserver.java:18)

D:\ssk>
```

### Chat Client



```
D:\ssk>javac chatclient.java

D:\ssk>java chatclient
Client : Hi
Server : Hai
Client : How r u?
Server : I am fine.Thanks
Client : bye
Server : BYE

D:\ssk>
```

## Result

Thus the java program to write applications using TCP sockets- Chat is executed successfully.

**Aim**

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

**Algorithm****Server**

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 is exists then FileReader read the content until EOF is reached.
7. Stop the program.

**Client**

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

### File Server

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

public class SimpleFileServer{ public static void main(String[] args) throws Exception
{
//Initialize Sockets
ServerSocket sock = new ServerSocket(5000);

Socket socket = sock.accept();

//The InetAddress specification
InetAddress IA = InetAddress.getByName("localhost");

//Specify the file
File file = new File("D:\\s1.txt");
FileInputStream fis = new FileInputStream(file);
BufferedReader bis = new BufferedReader(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(); sock.close();

System.out.println("File sent succesfully!");

}

}

```

### **File Client**

```

import java.io.BufferedOutputStream;

import java.io.FileOutputStream;

import java.io.InputStream;

import java.net.InetAddress;

import java.net.Socket;

public class SimpleFileClient

{

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("E:\\Source-downloaded.txt");

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!");
}
}
```

## Output

### File Server

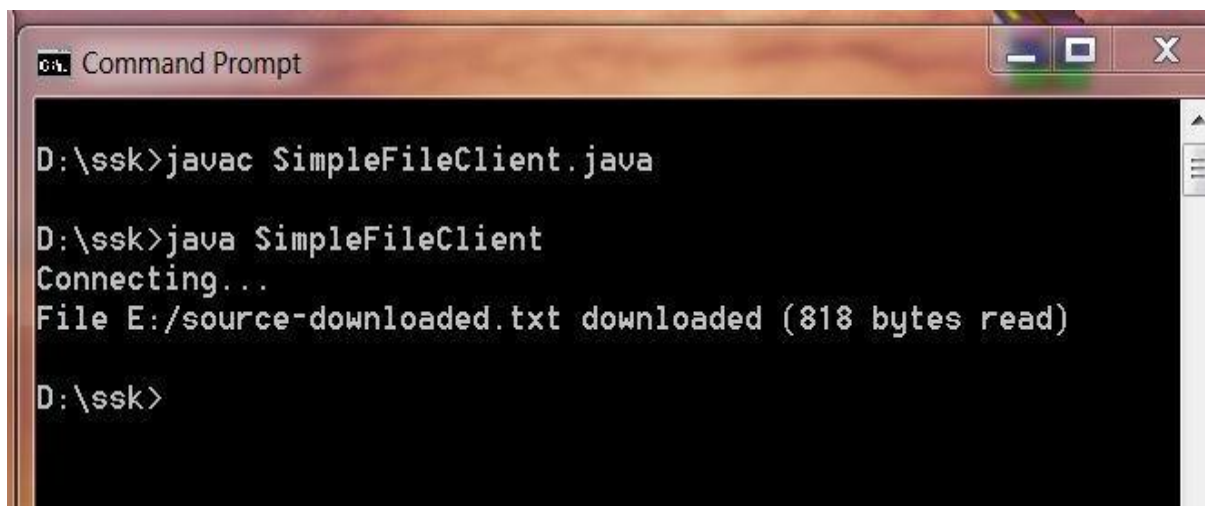


```
Command Prompt - java SimpleFileServer

D:\ssk>javac SimpleFileServer.java

D:\ssk>java SimpleFileServer
Waiting...
Accepted connection : Socket[addr=/127.0.0.1,port=49427,localport=
Sending D:/s1.txt(818 bytes)
Done.
Waiting...
```

### File Client



```
Command Prompt

D:\ssk>javac SimpleFileClient.java

D:\ssk>java SimpleFileClient
Connecting...
File E:/source-downloaded.txt downloaded (818 bytes read)

D:\ssk>
```

## Result

Thus, the file was transferred from server to client successfully using TCP sockets.

<b>EX.NO:4</b>	<b>Simulation of DNS using UDP sockets</b>

**Aim**

To write a java program for simulating DNS using UDP Sockets.

**Algorithm**

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



## Program

### UDP DNS Server

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

class Serverdns
{
    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();
                //System.out.println(s);

                InetAddress addr=receiver.getAddress();
                int port=receiver.getPort();

                String ip[]={"163.53.78.87","176.13.69.63", "172.28.251.59", "172.217.11.5","172.217.11.14"};

                String name[]={ "www.flipkart.com","www.facebook.com","www.zoho.com",
                "www.gmail.com","www.google.com"};

                for(int i=0;i<ip.length;i++)
                {
                    if(s.equals(ip[i]))
                    {
                        sendbyte=name[i].getBytes();

                        DatagramPacket sender=new DatagramPacket(sendbyte,sendbyte.length,addr,port);

                        server.send(sender);
```

```

                break;
            }
            else if(s.equals(name[i]))
            {
                sendbyte=ip[i].getBytes();
                DatagramPacket sender=new DatagramPacket(sendbyte,sendbyte.length,addr,port);
                server.send(sender);
                break;
            }
        }
        break;
    }
}
catch(Exception e)
{
    System.out.println(e);
}
}

```

### **UDP DNS Client**

```

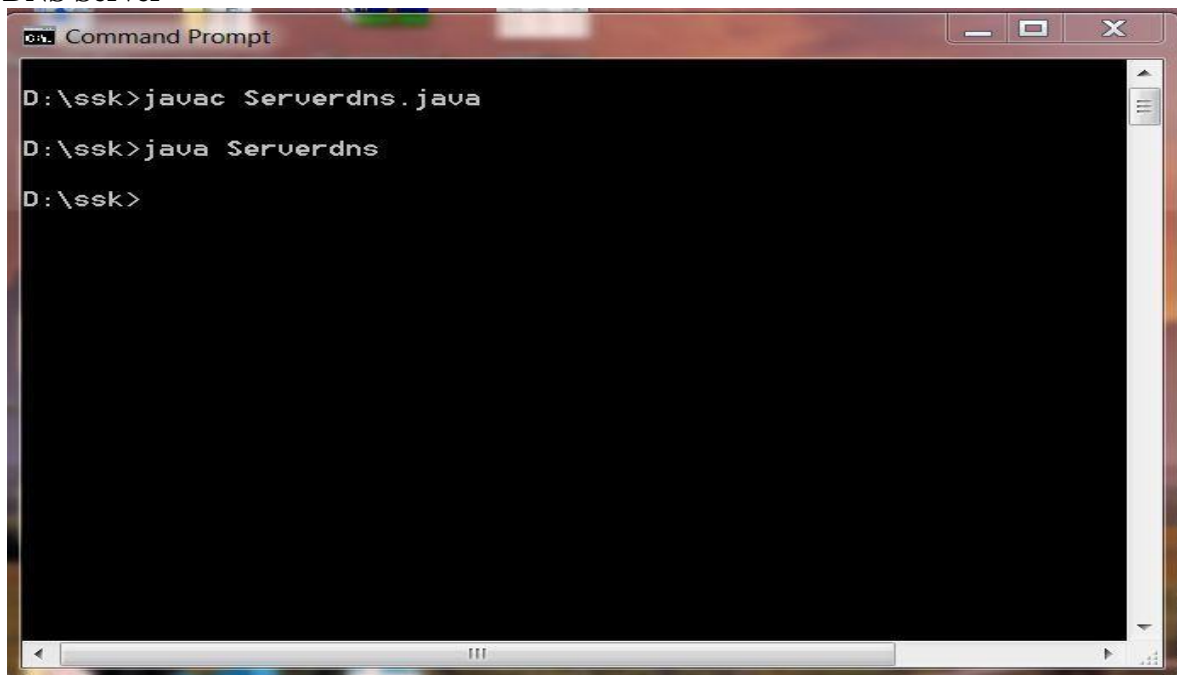
import java.io.*;
import java.net.*;
import java.util.*;
class Clientdns
{
    public static void main(String args[])
    {
        try
        {
            DatagramSocket client=new DatagramSocket();
            InetAddress addr=InetAddress.getByName("127.0.0.1");
            byte[] sendbyte=new byte[1024];
            byte[] receivebyte=new byte[1024];

```

```
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the DOMAIN NAME or IP adress:");
        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);
    }
}
```

## Output

### DNS Server



```
Command Prompt

D:\ssk>javac Serverdns.java

D:\ssk>java Serverdns

D:\ssk>
```

### DNS Client



```
Command Prompt

D:\ssk>javac Clientdns.java

D:\ssk>java Clientdns
Enter the DOMAIN NAME or IP address:
www.facebook.com
IP address or DOMAIN NAME: 176.13.69.63

D:\ssk>
```

## Result

Thus a java program to simulate DNS using UDP sockets is executed successfully.

<b>EX.NO:5</b>	<b>Simulation of ARP/RARP protocols</b>

**Aim**

To write a java program for simulating ARP and RARP protocols using Socket programming.

**Algorithm****Client**

- 1.Start the program
- 2.Using socket connection is established between client and server.
- 3.Get the IP address to be converted into MAC address.
- 4.Send this IP address to server.
- 5.Server returns the MAC address to client.

**Server**

- 1.Start the program
- 2.Accept the socket which is created by the client.
- 3.Server maintains the table in which IP and corresponding MAC addresses are stored.
- 4.Read the IP address which is send by the client.
- 5.Map the IP address with its MAC address and return the MAC address to client.

## Program

// Socket Programming

// ARP and RARP Protocol

// arp\_rarp.java

/\*

This program automatically creates a file with the IP address of machines, their MAC address and type.

ARP protocol is simulated by reading an IP address and returning the MAC address. RARP protocol is simulated by reading an MAC address and returning the IP address

The program can be extended to read an IP Address / Mac Address and a message and send a packet to the specified machine using TCP / IP or Datagram sockets \*/

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

```
import java.util.*;
```

```
public class arp_rarp
```

```
{
```

```
    private static final String Command = "arp -a";
```

```
        public static void getARPTable(String cmd) throws Exception
```

```
    {
```

```
        File fp = new File("ARPTable.txt");
```

```
        FileWriter fw = new FileWriter(fp);
```

```
        BufferedWriter bw = new BufferedWriter(fw);
```

```
        Process P = Runtime.getRuntime().exec(cmd);
```

```
        Scanner S = new Scanner(P.getInputStream()).useDelimiter("\\A");
```

```
            while(S.hasNext())
```

```
                bw.write(S.next());
```

```
        bw.close();
```

```
        fw.close();
```

```
    }
```

```
    public static void findMAC(String ip) throws Exception
```

```
    {
```

```
        File fp = new File("ARPTable.txt");
```

```
        FileReader fr = new FileReader(fp);
```

```
        BufferedReader br = new BufferedReader(fr);
```

```

String line;

    while ((line = br.readLine()) != null)
{
    if (line.contains(ip))
    {
        System.out.println("Internet Address    Physical Address    Type");
        System.out.println(line);

        break;
    }
}

if((line == null))

    System.out.println("Not found");

fr.close();

br.close();

}

public static void findIP(String mac) throws Exception
{
    File fp = new File("ARPTable.txt");
    FileReader fr = new FileReader(fp);
    BufferedReader br = new BufferedReader(fr);
    String line;

    while ((line = br.readLine()) != null)
    {
        if (line.contains(mac))
        {
            System.out.println("Internet Address    Physical Address    Type");
            System.out.println(line);

            break;
        }
    }

    if((line == null))

```

```
        System.out.println("Not found");

        fr.close();

    br.close();
}

public static void main(String as[]) throws Exception
{
    getARPTable(Command);

    Scanner S = new Scanner(System.in);

    System.out.println("ARP Protocol.");

    System.out.print("Enter IP Address: ");

    String IP = S.nextLine();

    findMAC(IP);

    System.out.println("RARP Protocol.");

    System.out.print("Enter MAC Address: ");

    String MAC = S.nextLine();

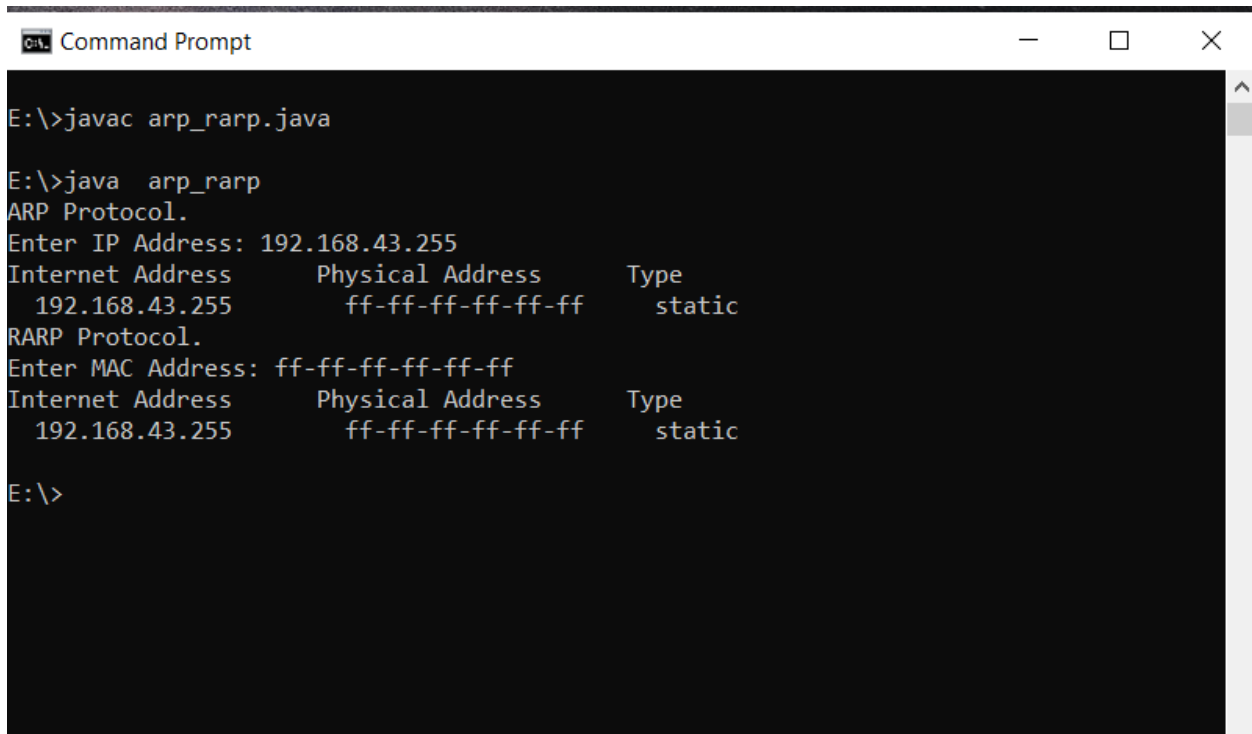
    findIP(MAC);

}

}
```



## Output



```
C:\> javac arp_rarp.java

C:\> java arp_rarp
ARP Protocol.
Enter IP Address: 192.168.43.255
Internet Address      Physical Address      Type
192.168.43.255       ff-ff-ff-ff-ff-ff     static
RARP Protocol.
Enter MAC Address: ff-ff-ff-ff-ff-ff
Internet Address      Physical Address      Type
192.168.43.255       ff-ff-ff-ff-ff-ff     static

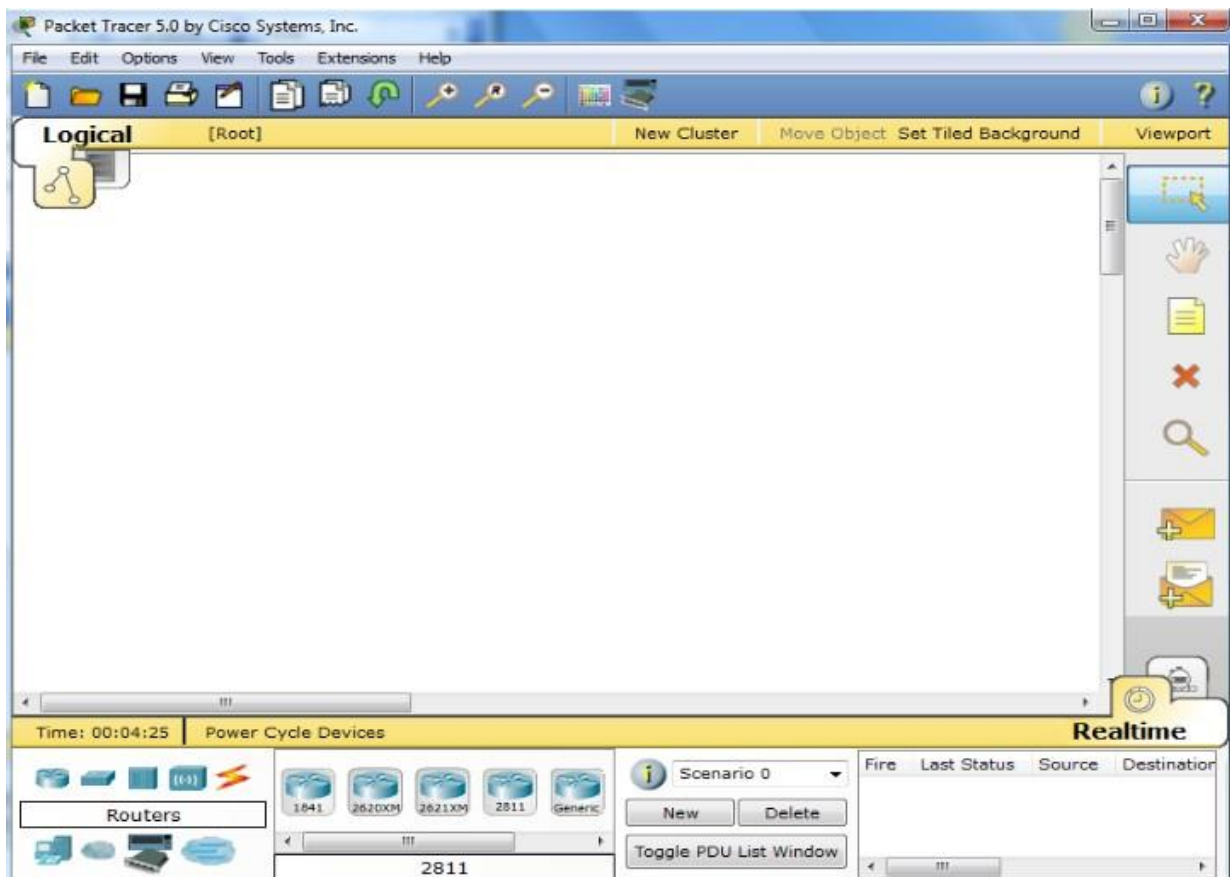
C:\>
```

## Result

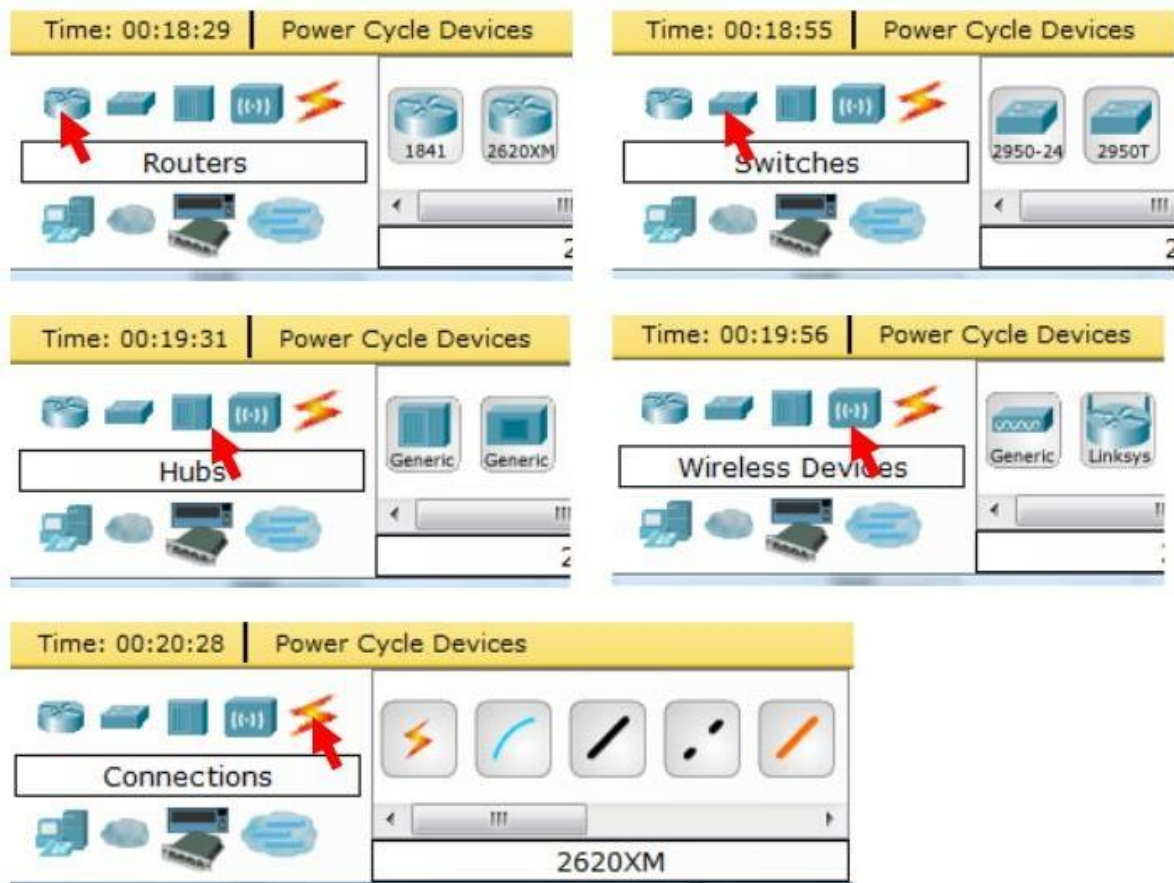
Thus, the ARP and RARP Protocols were implemented using socket programming and the output was verified.

**Aim**

To study network simulator-Packet Tracer. Packet Tracer is a protocol simulator developed by Dennis Frezzo and his team at Cisco Systems. Packet Tracer (PT) is a powerful and dynamic tool that displays the various protocols used in networking, in either Real Time or Simulation mode. This includes layer 2 protocols such as Ethernet and PPP, layer 3 protocols such as IP, ICMP, and ARP, and layer 4 protocols such as TCP and UDP. Routing protocols can also be traced.

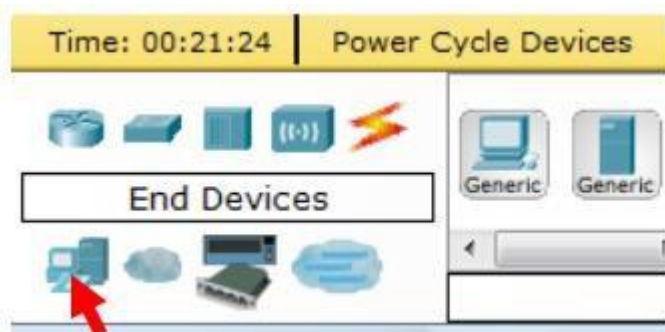
**Step 1: Start Packet Tracer****Step 2: Choosing Devices and Connections**

We will begin building our network topology by selecting devices and the media in which to connect them. Several types of devices and network connections can be used. For this lab we will keep it simple by using End Devices, Switches, Hubs, and Connections. Single click on each group of devices and connections to display the various choices. The devices you see may differ slightly.

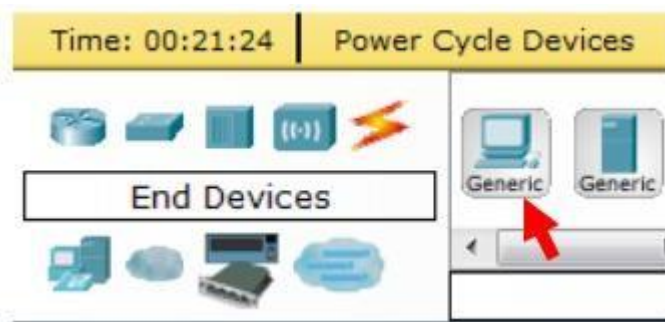


### Step 3: Building the Topology –Adding Hosts

Single click on the End Devices.

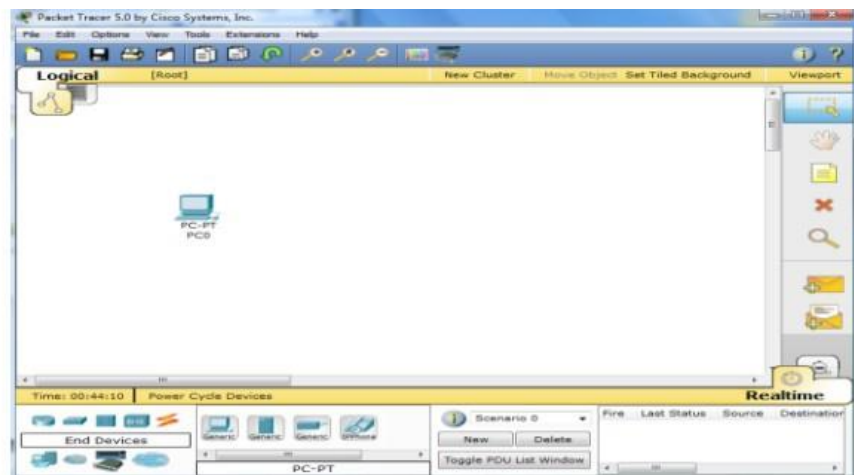


Single click on the Generic host.



Move the cursor into topology area. You will notice it turns into a plus “+” sign.

Single click in the topology area and it copies the device.



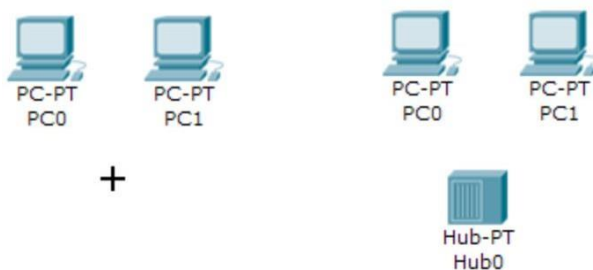
Add three more hosts.



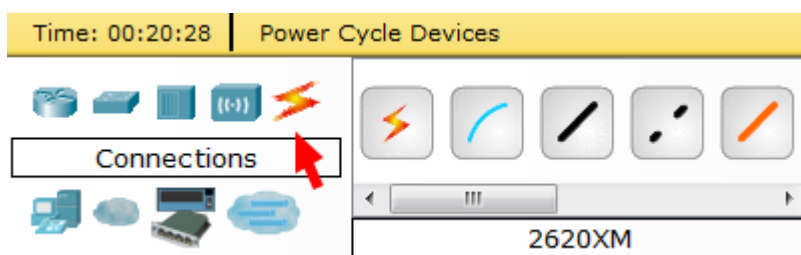
#### Step 4: Building the Topology – Connecting the Hosts to Hubs and Switches

##### Adding a Hub

Select a hub, by clicking once on **Hubs** and once on a **Generic** hub.



Connect PC0 to Hub0 by first choosing **Connections**.

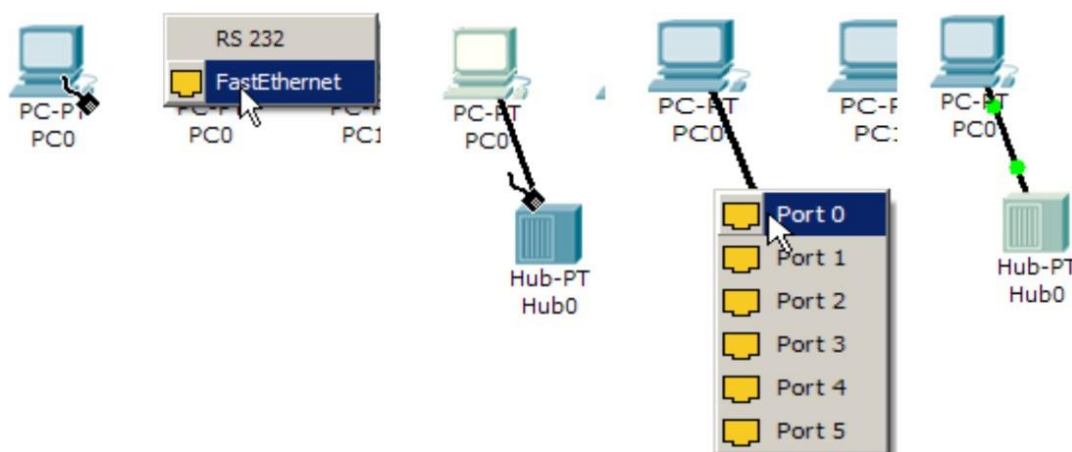


Click once on the **Copper Straight-through** cable.

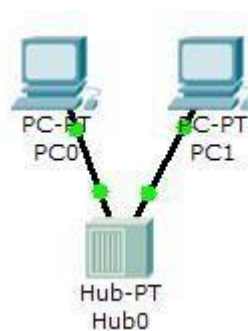


Perform the following steps to connect PC0 to Hub0:

1. Click once on PC0
2. Choose FastEthernet
3. Drag the cursor to Hub0
4. Click once on Hub0 and choose Port 0
5. Notice the green link lights on both the PC0 Ethernet NIC and the Hub0 Port 0 showing that the link is active.

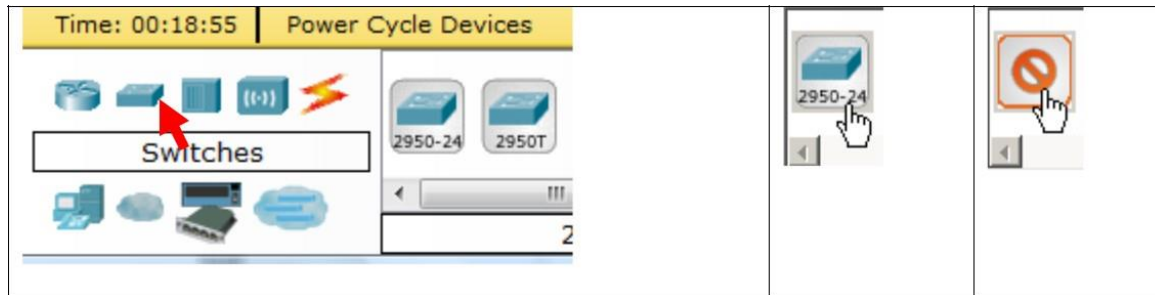


Repeat the steps above for PC1 connecting it to Port 1 on Hub0. (The actual hub port you choose does not matter.)

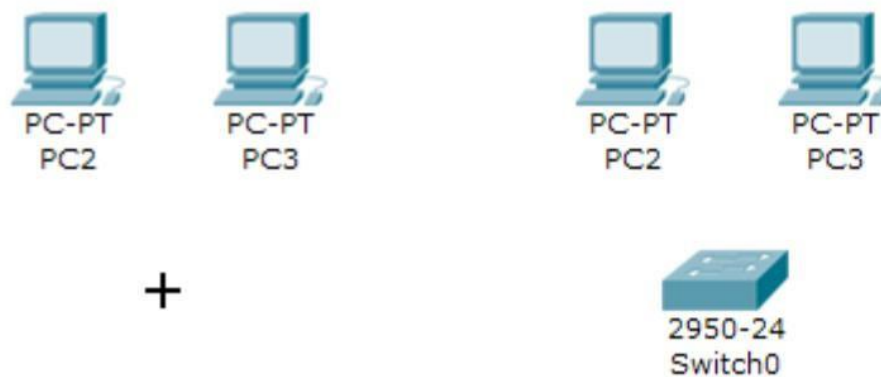


## Adding a Switch

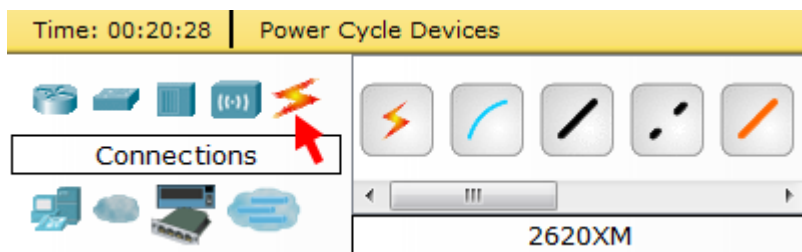
Select a switch, by clicking once on Switches and once on a 2950-24 switch.



Add the switch by moving the plus sign “+” below PC2 and PC3 and click once.



Connect PC2 to Hub0 by first choosing **Connections**.



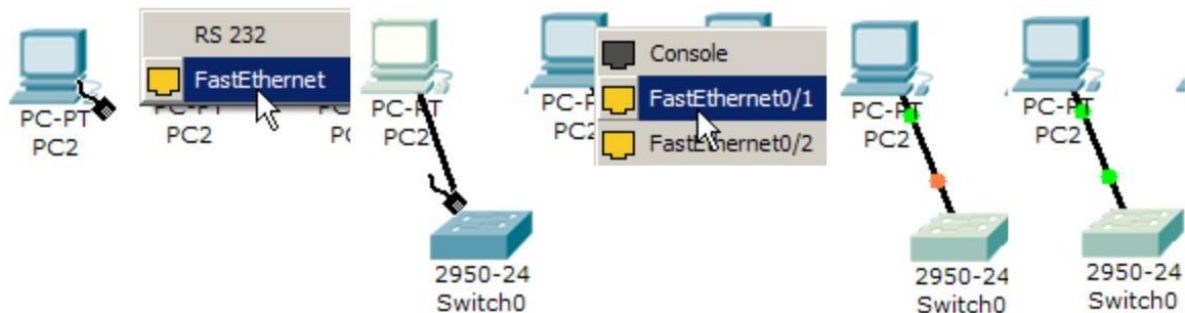
Click once on the **Copper Straight-through** cable.



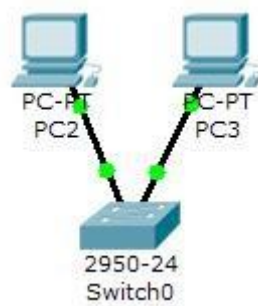
Perform the following steps to connect **PC2** to **Switch0**:

1. Click once on **PC2**
2. Choose **Fast Ethernet**
3. Drag the cursor to **Switch0**

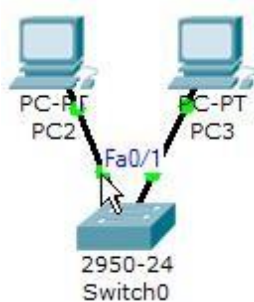
4. Click once on **Switch0** and choose **FastEthernet0/1**
5. Notice the green link lights on **PC2** Ethernet NIC and amber light **Switch0FastEthernet0/1 port**. The switch port is temporarily not forwarding frames, while it goes through the stages for the Spanning Tree Protocol (STP) process.
6. After a about 30 seconds the amber light will change to green indicating that the port has entered the forwarding stage. Frames can now forwarded out the switch port.



Repeat the steps above for **PC3** connecting it to **Port 3** on **Switch0** on port **FastEtherent0/2**. (The actual switch port you choose does not matter.)



Move the cursor over the link light to view the port number. **Fa** means FastEthernet, 100 Mbps Ethernet.

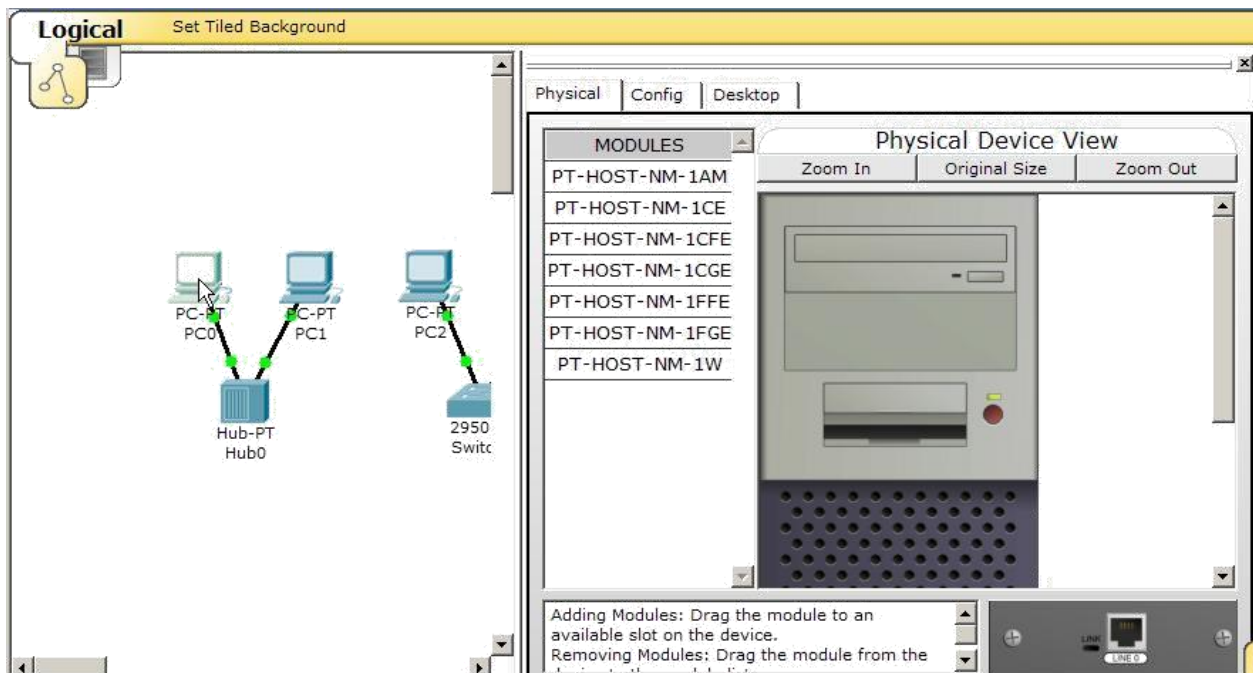




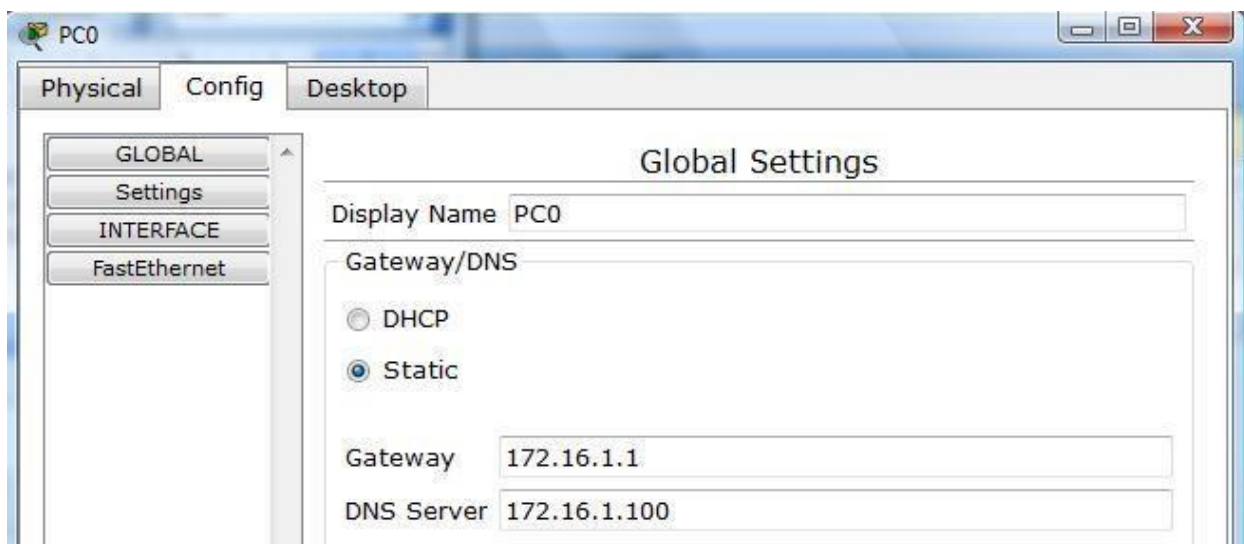
## Step 5: Configuring IP Addresses and Subnet Masks on the Hosts

Before we can communicate between the hosts we need to configure IP Addresses and Subnet Masks on the devices.

Click once on PC0.

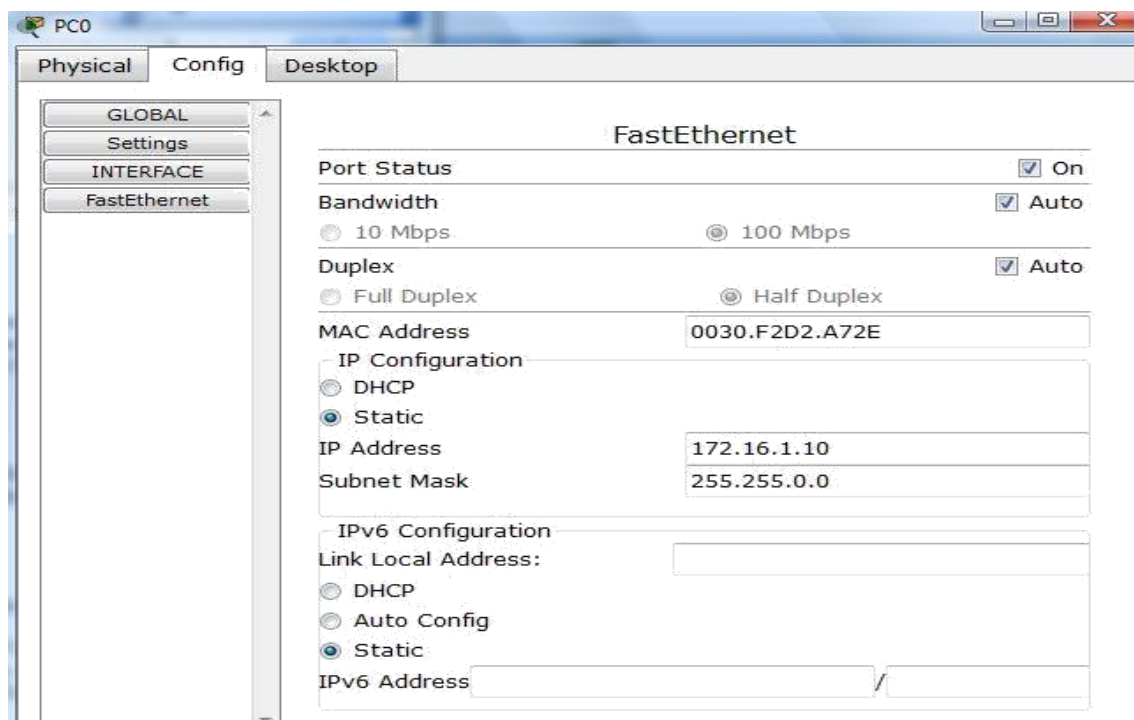


Choose the **Config** tab and click on **Settings**. It is here that you can change the name of PC0. It is also here where you would enter a **Gateway IP Address**, also known as the default gateway and the **DNS Server IP Address**.



Click on **Interface** and then **Fast Ethernet**. Although we have not yet discussed IP Addresses, add the IP Address to 172.16.1.10. Click once in the Subnet Mask field to enter the default Subnet Mask. You can leave this at 255.255.0.0.





Also, notice this is where you can change the Bandwidth (speed) and Duplex of the Ethernet NIC (Network Interface Card). The default is Auto (auto negotiation), which means the NIC will negotiate with the hub or switch. The bandwidth and/or duplex can be manually set by removing the check from the **Auto** box and choosing the specific option.

### Bandwidth - Auto

If the host is connected to a hub or switch port which can do 100 Mbps, then the Ethernet NIC on the host will choose 100 Mbps (Fast Ethernet). Otherwise, if the hub or switch port can only do 10 Mbps, then the Ethernet NIC on the host will choose 10 Mbps (Ethernet).

### Duplex - Auto

**Hub:** If the host is connected to a hub, then the Ethernet NIC on the host will choose Half Duplex.

**Switch:** If the host is connected to a switch, and the switch port is configured as Full Duplex (or Auto negotiation), then the Ethernet NIC on the host will choose Full Duplex. If the switch port is configured as Half Duplex, then the Ethernet NIC on the host will choose Half Duplex. (Full Duplex is a much more efficient option.)

The information is automatically saved when entered.

To close this dialog box, click the “X” in the upper right.

Repeat these steps for the other hosts.

### Deleting a Device or Link

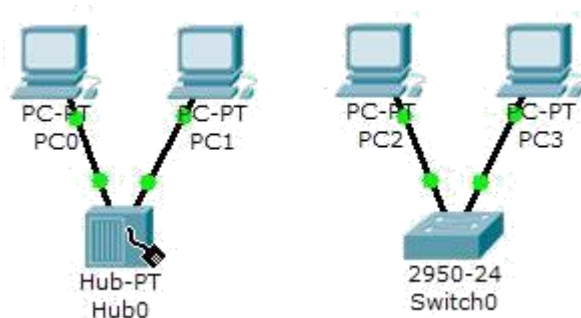
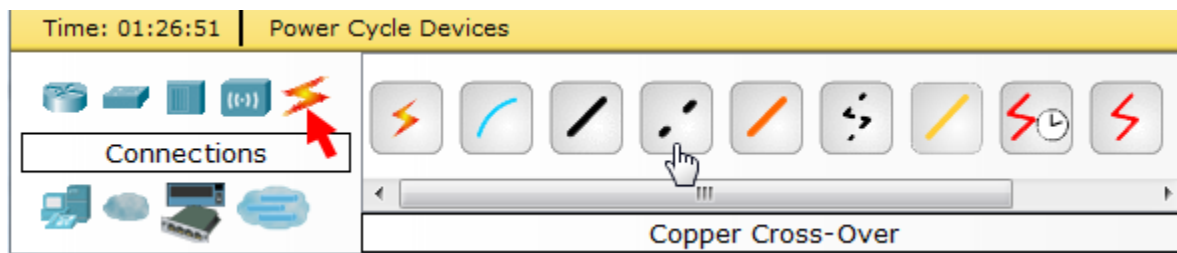
To delete a device or link, choose the **Delete** tool and click on the item you wish to delete.



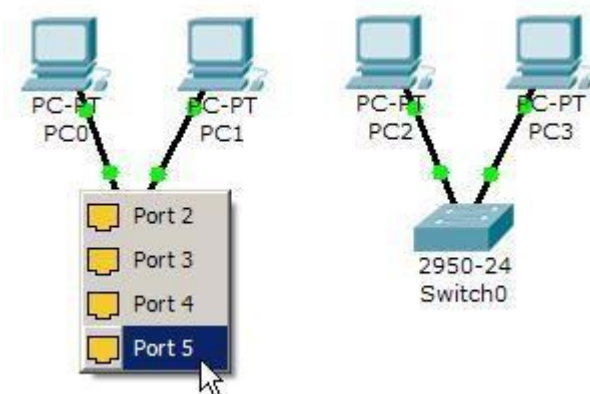
## Step 6: Connecting Hub0 to Switch0

To connect like-devices, like a Hub and a Switch, we will use a Cross-over cable.

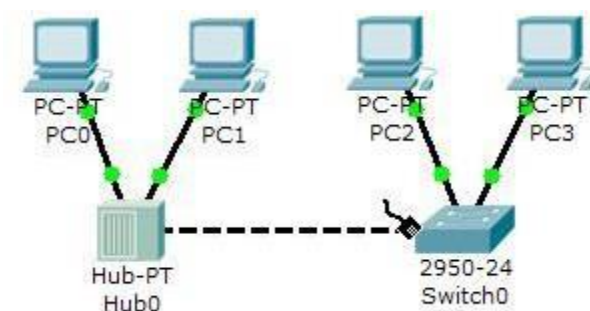
Click once the **Cross-over** Cable from the **Connections** options.



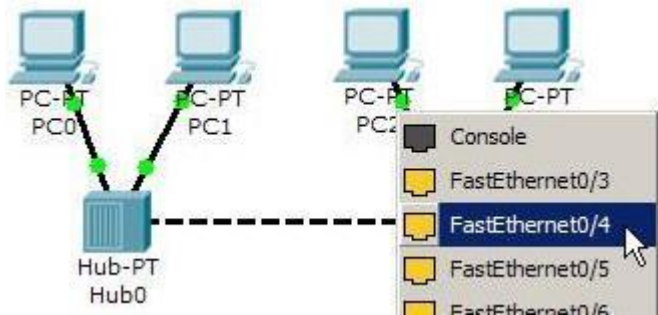
Select **Port 5** (actual port does not matter).



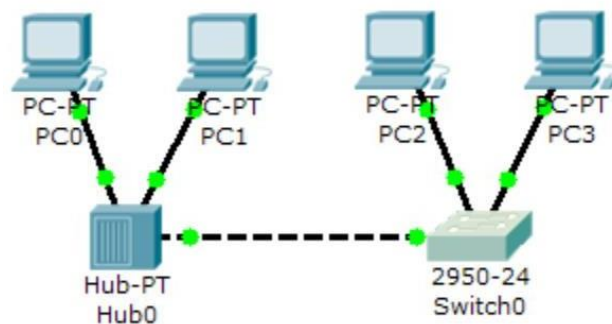
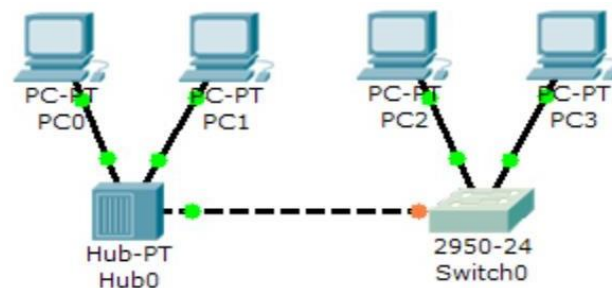
Move the Connections cursor to **Switch0**.



Click once on **Switch0** and choose **FastEthernet0/4** (actual port does not matter).



The link light for switch port **FastEthernet0/4** will begin as amber and eventually change to green as the Spanning Tree Protocol transitions the port to forwarding.



### Step 7: Verifying Connectivity in Real-time Mode

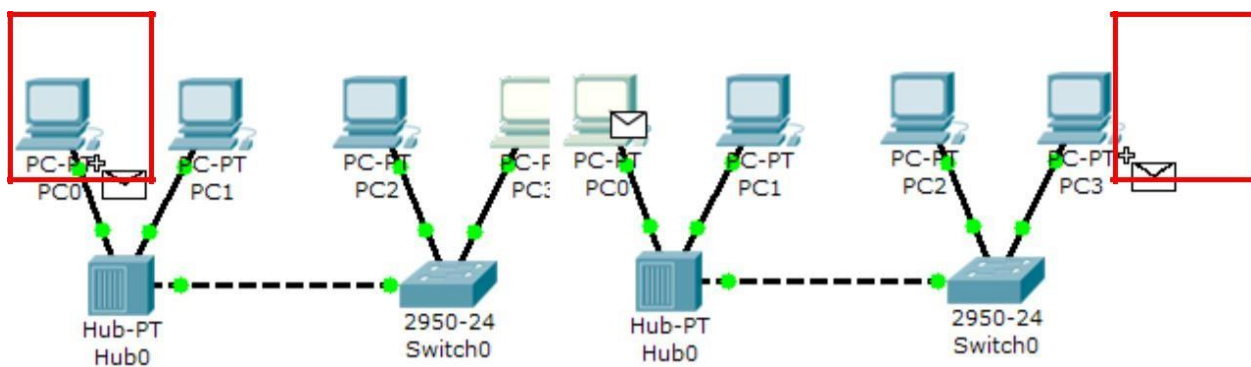
Be sure you are in **Real-time** mode.



Select the **Add Simple PDU** tool used to ping devices.



Click once on PC0, then once on PC3.



The PDU **Last Status** should show as **Successful**.

Fire	Last Status	Source	Destination	Type
	Successful	PC0	PC3	ICMP

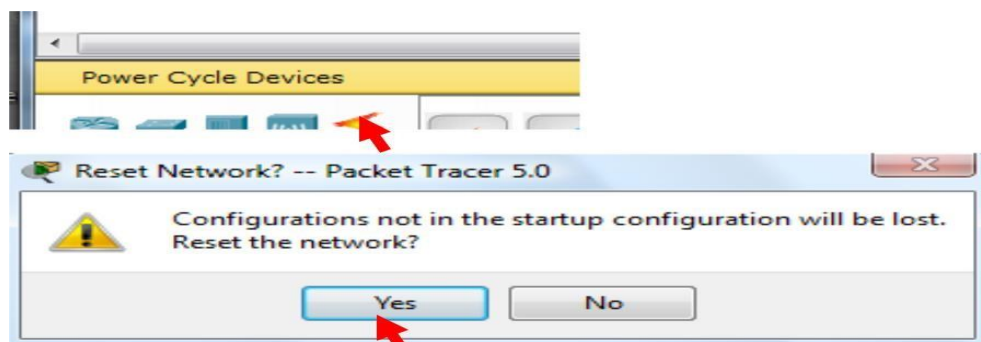
### Resetting the Network

At this point we will want to reset the network, whenever you want to reset the network and begin the simulation again, perform the following tasks:

Click **Delete** in the PDU area.

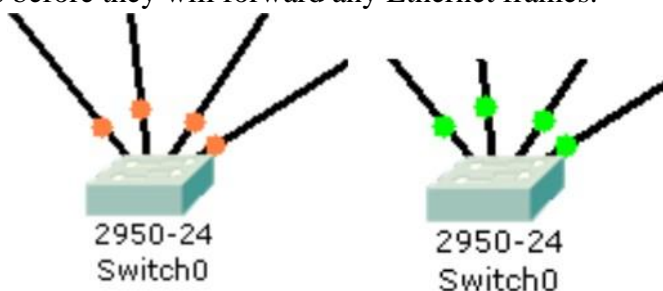


Now, Power Cycle Devices and confirm the action.



## Waiting for Spanning Tree Protocol (STP)

**Note:** Because Packet Tracer also simulates the Spanning Tree Protocol (later), attimes the switch may show amber lights on its interfaces. You will need to wait for the lights to turn green on the switches before they will forward any Ethernet frames.

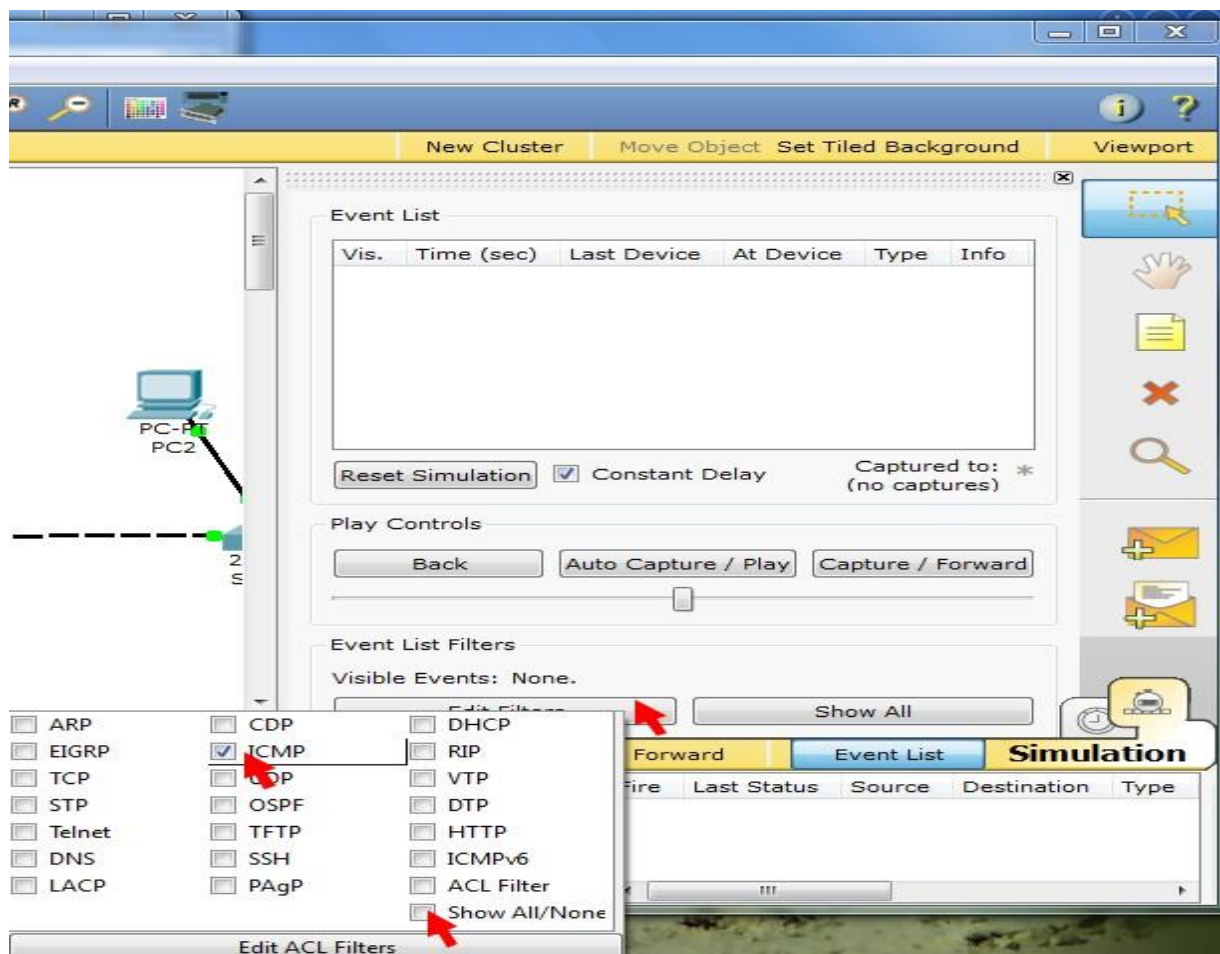


## Step 8: Verifying Connectivity in Simulation Mode

Be sure you are in **Simulation** mode.



Deselect all filters (All/None) and select only **ICMP**.

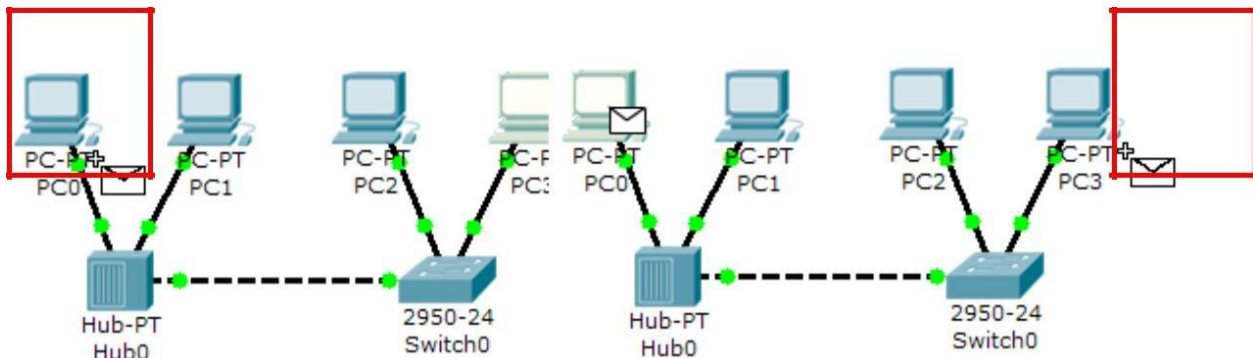


Select the **Add Simple PDU** tool used to ping devices..





Click once on PC0, then once on PC3.



Continue clicking **Capture/Forward** button until the ICMP ping is completed. You should see the ICMP messages move between the hosts, hub and switch. The PDU **Last Status** should show as **Successful**. Click on **Clear Event List** if you do not want to look at the events or click **Preview Previous Events** if you do. For this exercise it does not matter.

Buffer Full -- Packet Tracer 5.0

The maximum number of events has been reached. You may clear the event list and continue from where you left off or adjust the filters to view previous events.

Clear Event List View Previous Events

Vis.	Time (sec)	Last Device	At Device	Type	Infc
	0.009	Switch0	PC3	ICMP	
	0.010	PC3	Switch0	ICMP	
	0.011	Switch0	Hub0	ICMP	
			PC0	ICMP	
			PC1	ICMP	

Event List Filters

Visible Events: ICMP

Edit Filters Show All

Time: 01:45:00.969 Power Cycle Devices PLAY Back Auto Capture / Play Capture / Forward Event List Simulation

Connections

Copper Cross-Over

Scenario 0

New Delete

Toggle PDU List Window

Fire Last Status Source Destination Type

Successful PC0 PC3 ICMP

## Step 9: Saving the Topology

Perform the following steps to save the topology (uses .pkt file extension).

## Result

Thus, the network simulator –Packet Tracer were studied in detail.

**Aim**

To write a program to simulate congestion control algorithm-stop and wait protocol.

**Algorithm**

1. Read the number of frames to be sent.
2. Sender establish connection with the receiver.
3. After the connection is established, the sender send the frames one by one.
4. The next frame is sent only when the acknowledgement for the previous frame is received.
5. Disconnect the connection after all the frames are received by the receiver.

## Program

### Sender

```
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("receiver>" +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**

```
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();

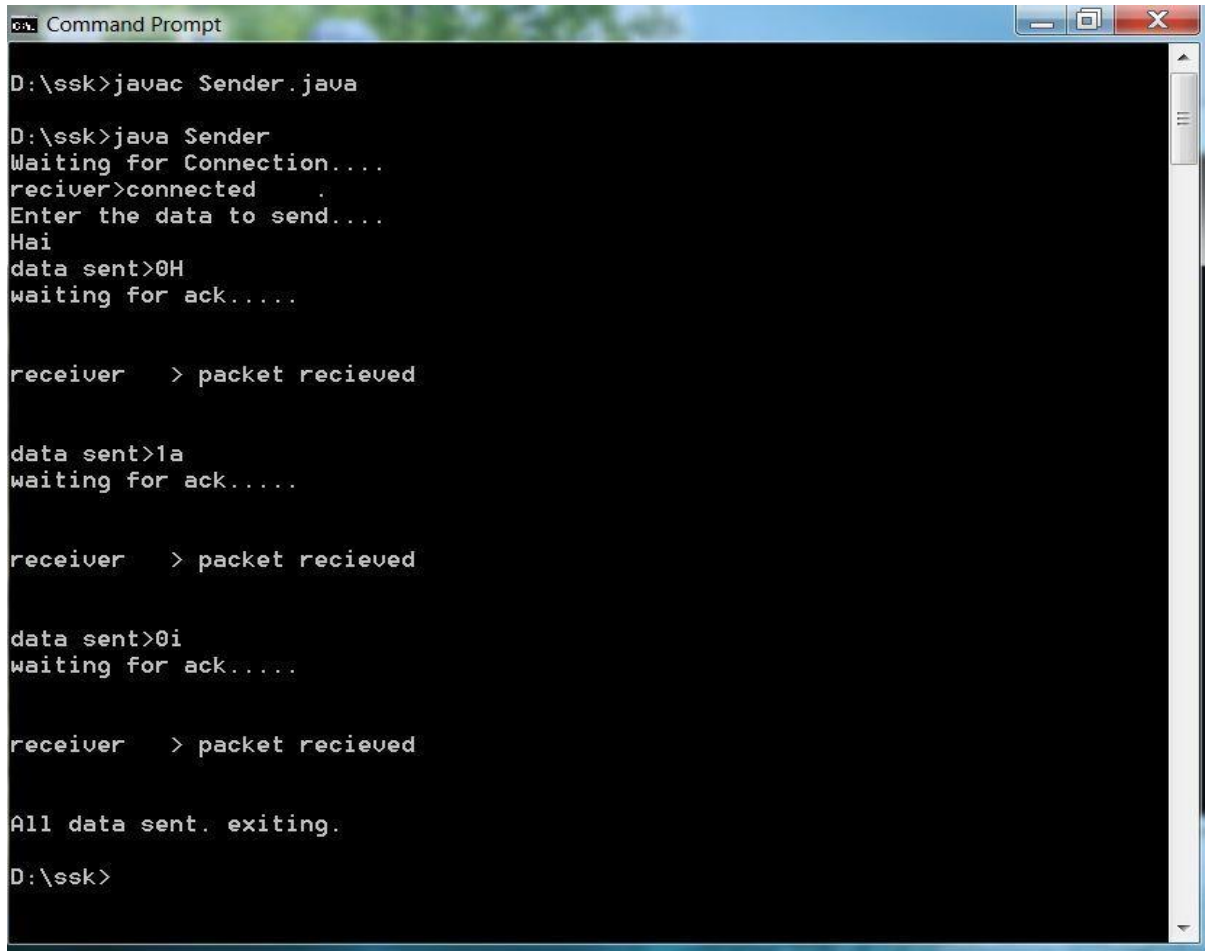
reciever.close();

```

```
}  
  
catch(Exception e){ }  
  
}  
  
}  
  
public static void main(String args[]){  
  
    Reciever s=new Reciever();  
  
    while(true){  
  
        s.run();  
  
    }  
  
}  
  
}
```

## Output

### Sender



```
D:\ssk>javac Sender.java
D:\ssk>java Sender
Waiting for Connection....
receiver>connected
Enter the data to send....
Hai
data sent>0H
waiting for ack.....

receiver > packet recieved

data sent>1a
waiting for ack.....

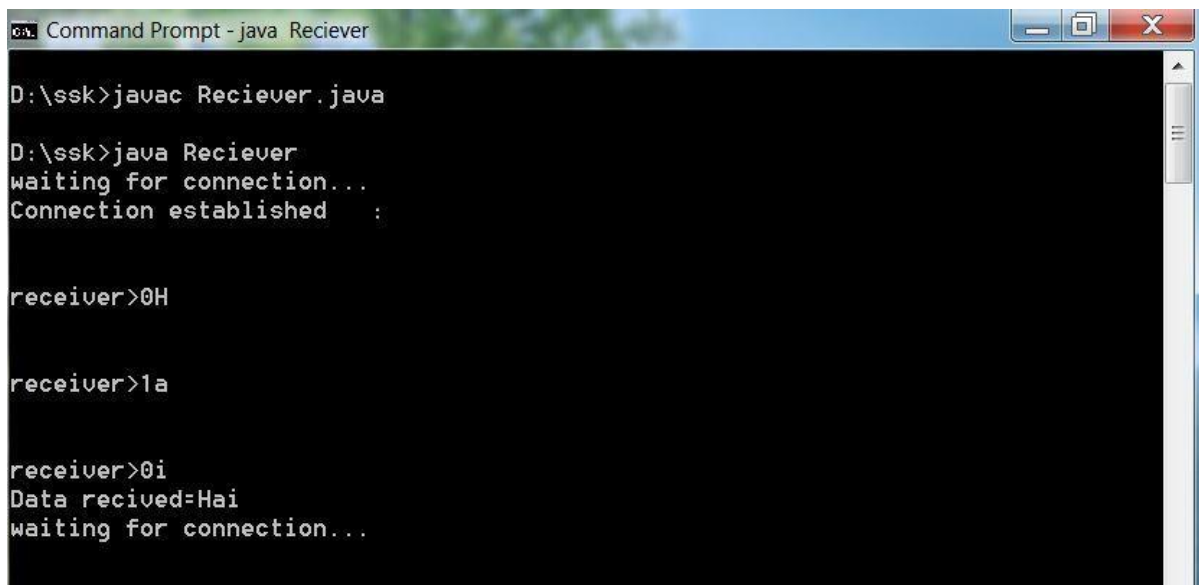
receiver > packet recieved

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

receiver > packet recieved

All data sent. exiting.
D:\ssk>
```

### Receiver



```
D:\ssk>javac Reciever.java
D:\ssk>java Reciever
waiting for connection...
Connection established :

receiver>0H

receiver>1a

receiver>0i
Data recieved=Hai
waiting for connection...
```

### Result

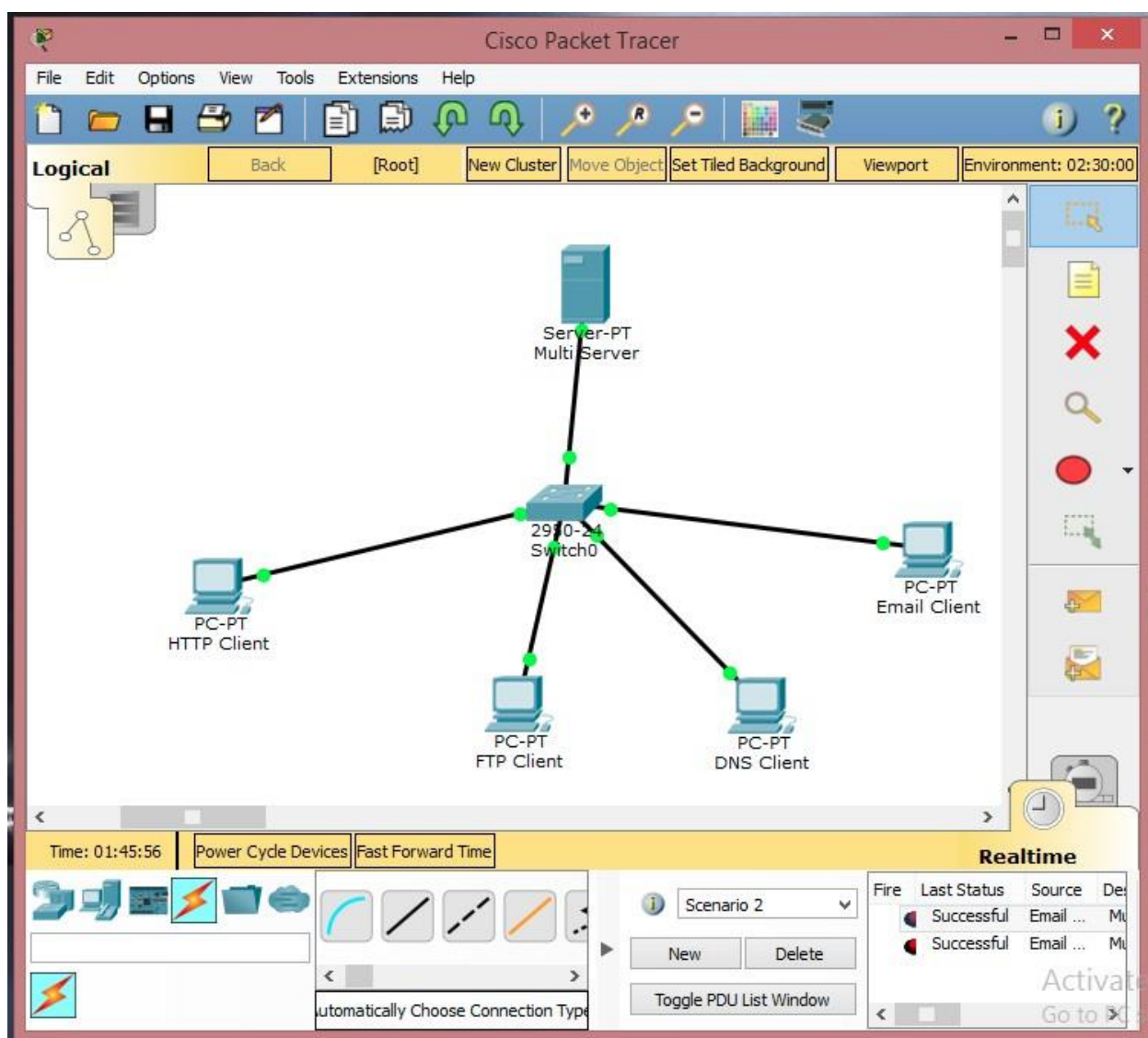
Thus the program to simulate congestion control algorithm-stop and wait protocol is executed successfully.

**Aim**

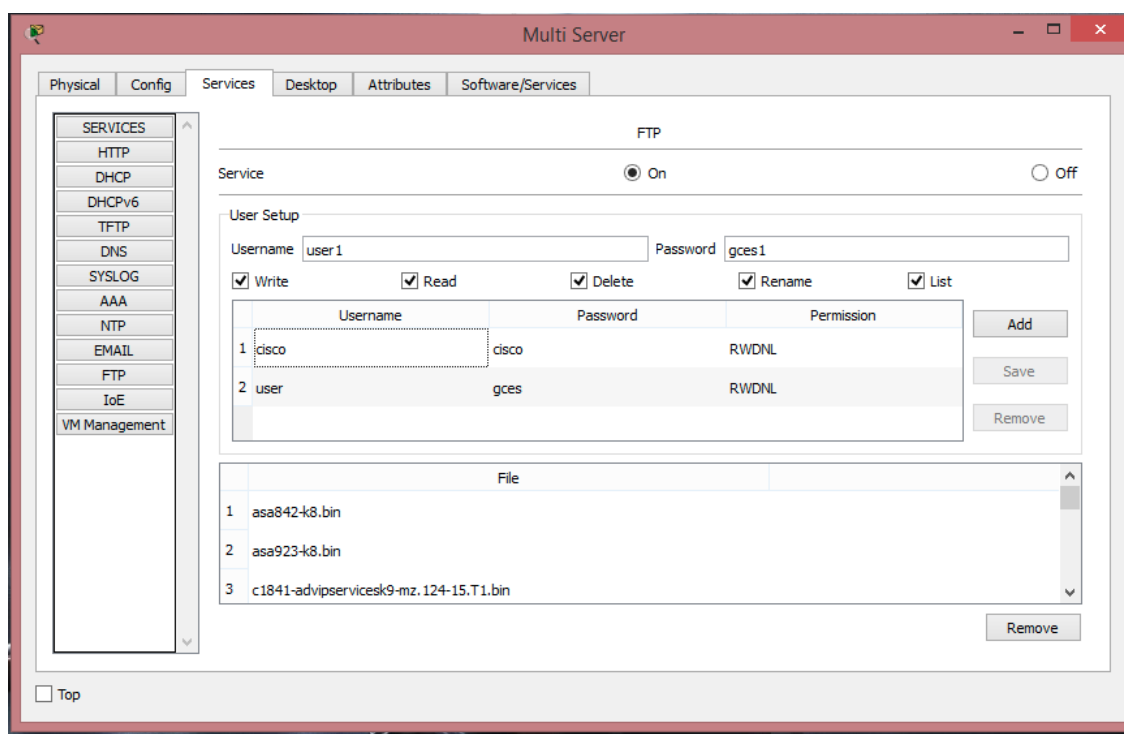
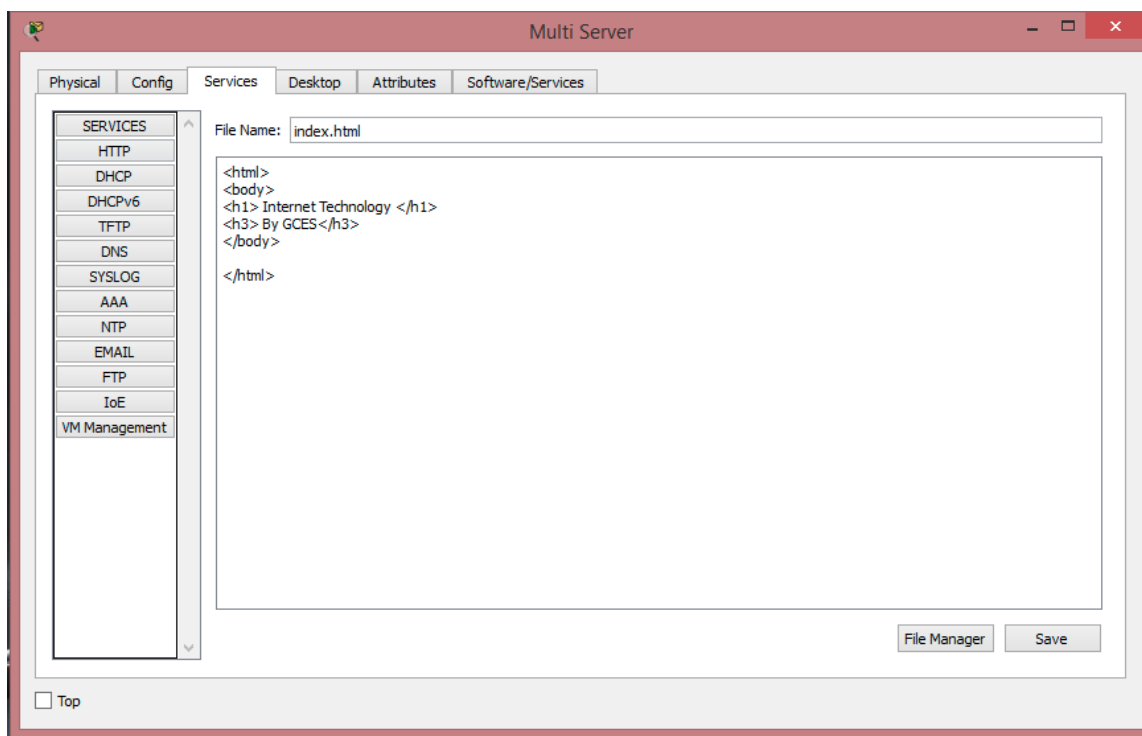
To study the performance of TCP/UDP using Packet Tracer.

**Procedure**

1. Develop a Topology shown in figure given below.
2. Add HTTP Client, FTP Client, DNS Client, and Email Client connected to a multi-server through the switch.
3. Configure the network.
4. Add users to the multi-server and create domain name.



5. Edit index.html file using the code shown in figure. Run the Code by giving the domain in multi-server and HTTP client.
6. Connect the pvg.edu.in domain using ftp server.
7. Compose, send and receive mail between the clients connected to the server.



Multi Server

Physical

Config

Services

Desktop

Attributes

Software/Services

SERVICES

HTTP

DHCP

DHCPv6

TFTP

DNS

SYSLOG

AAA

NTP

EMAIL

FTP

IoE

VM Management

DNS

DNS Service ☒ On ☐ Off

Resource Records

Name

Type

A Record

Address

Add

Save

Remove

No.	Name	Type	Detail
0	pvg.edu.in	A Record	192.168.0.2

DNS Cache

☐ Top

Multi Server

Physical

Config

Services

Desktop

Attributes

Software/Services

SERVICES

HTTP

DHCP

DHCPv6

TFTP

DNS

SYSLOG

AAA

NTP

EMAIL

FTP

IoE

VM Management

EMAIL

SMTP Service

☒ ON
☐ OFF

POP3 Service

☒ ON
☐ OFF

Domain Name:

pvg.edu.in

Set

User Setup

User

user2

Password

gces2

user

user 1

+

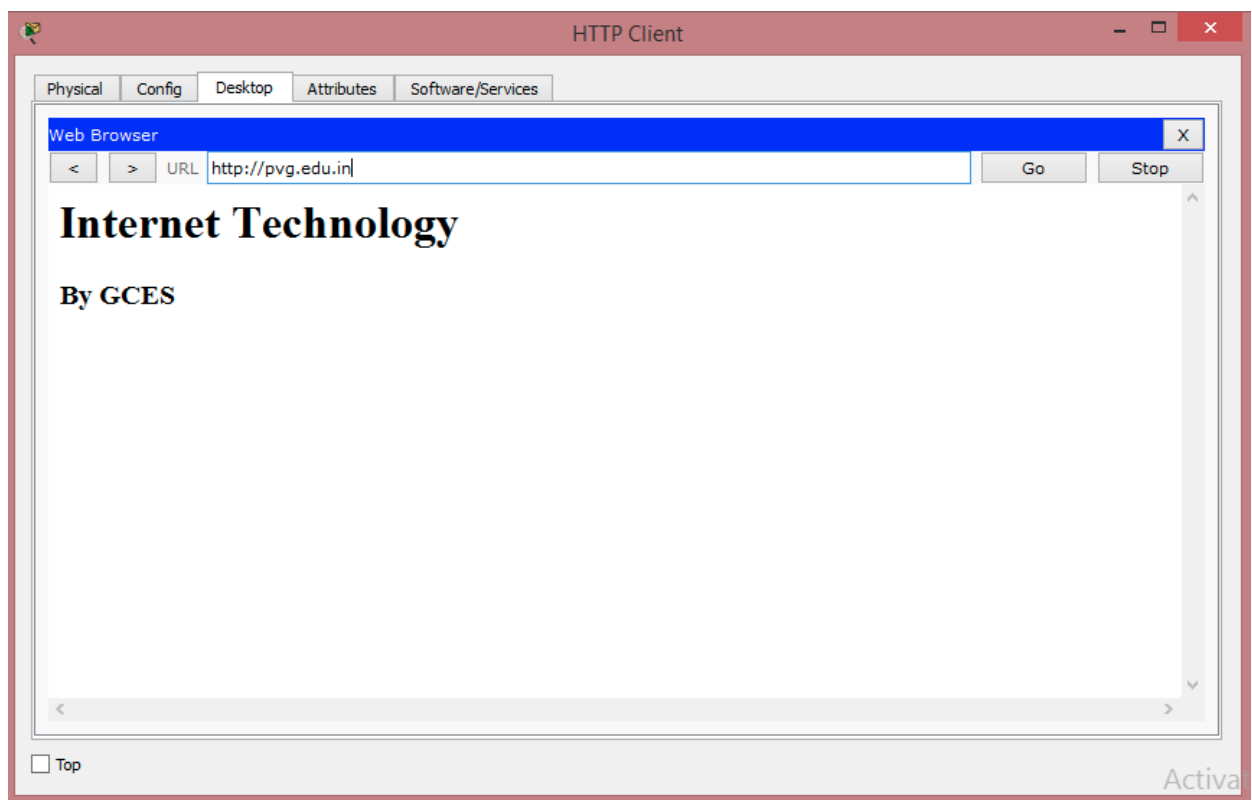
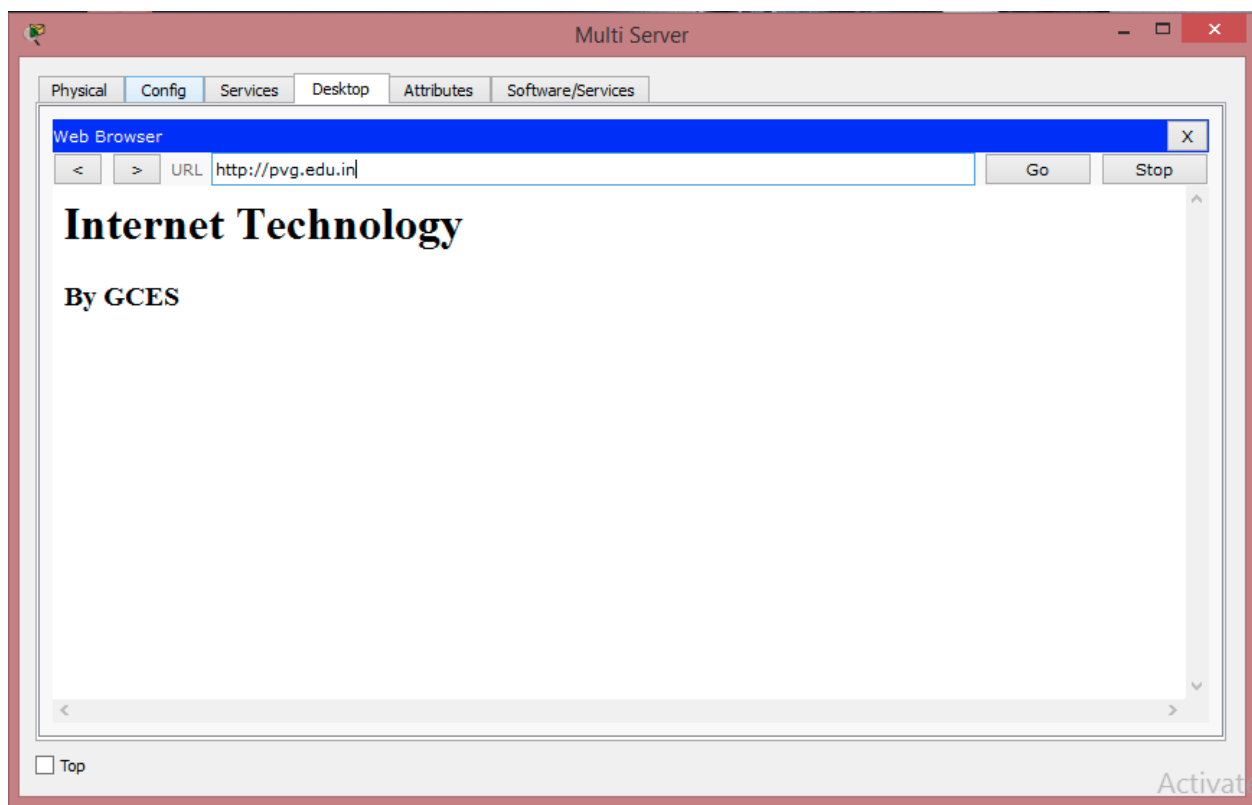
-

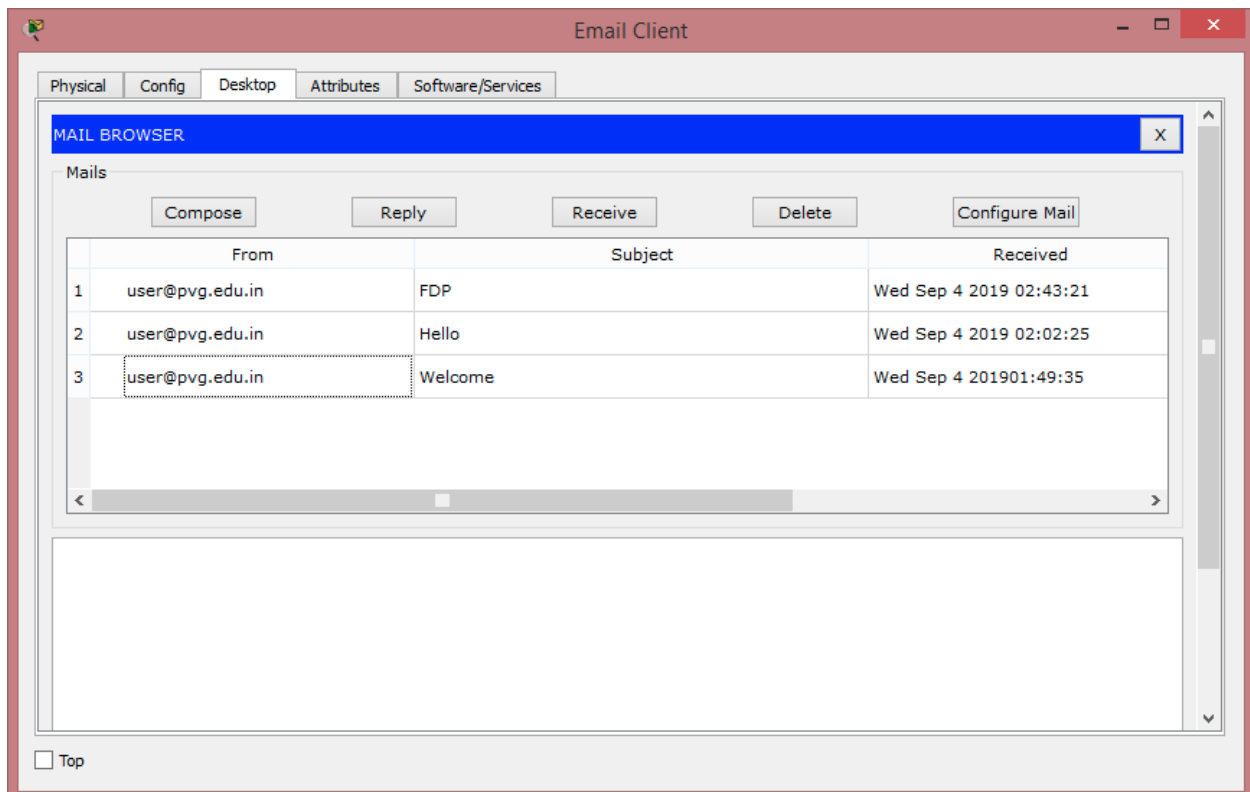
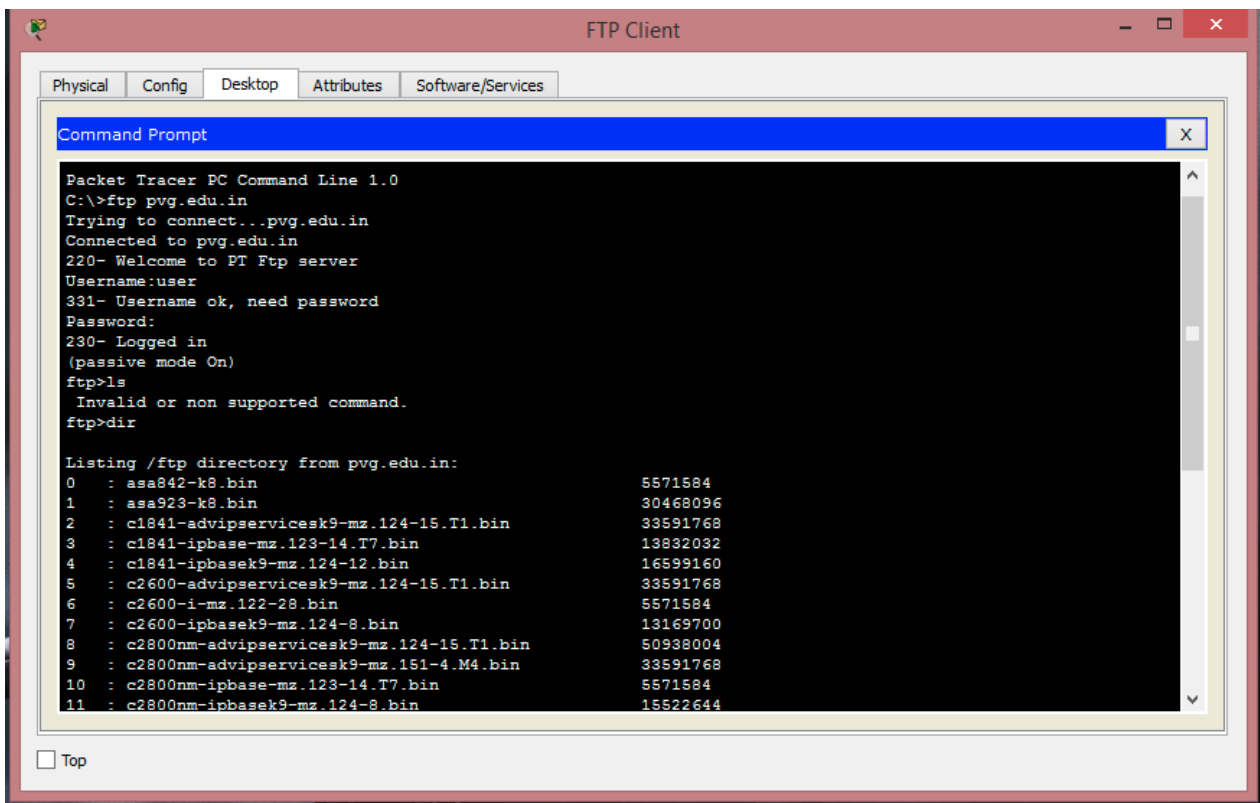
Change

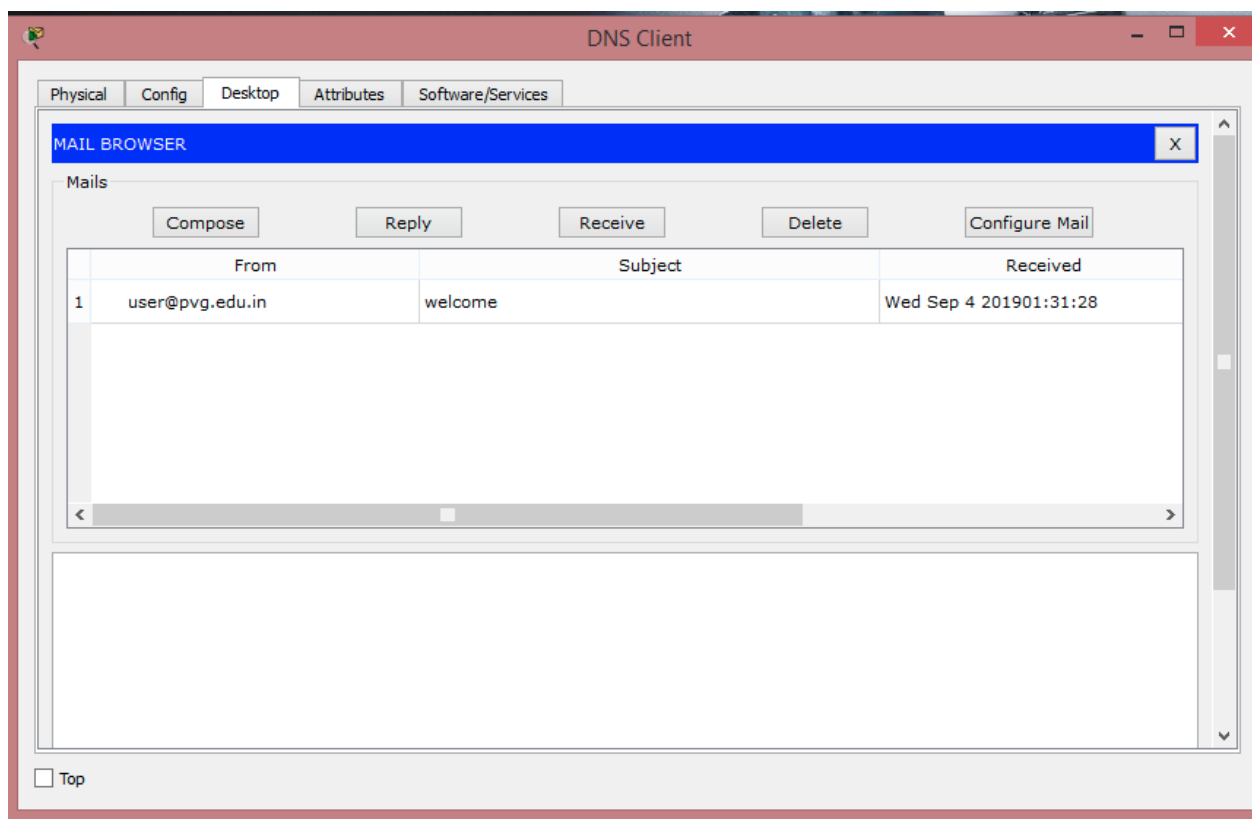
Password

☐ Top









## Result

Thus the TCP/UDP implementation using Packet Tracer is performed successfully.

**Aim**

To write a program to simulate Distance Vector Routing algorithm.

**Algorithm**

1. Create a network.
2. Assign cost to the edges in the network.
3. Initialize the distance of the node itself as zero.
4. Assign the edge cost as the cost for the immediate neighbours and for other nodes assign the value as infinity.
5. From time-to-time, each node sends its own distance vector estimate to neighbors. When a node x receives new DV estimate from any neighbor v, it saves v's distance vector and it updates its own DV using Bellman Ford equation:

$$D_x(y) = \min \{ C(x,v) + D_v(y) \} \text{ for each node } y \in N$$

6. Print the routing table.

## Program

```
import java.io.BufferedReader;
import java.io.*;
public class DVR {
static int graph[][];
static int via[][];
static int rt[][];
static int v;
static int e;
public static void main(String[] args) throws IOException{
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Please enter the number of Vertices: ");
    v = Integer.parseInt(br.readLine());
    System.out.println("Please enter the number of Edges: ");
    e = Integer.parseInt(br.readLine());
    graph = new int[v][v];
    via = new int[v][v];
    rt = new int[v][v];
    for(int i = 0; i < v; i++)
    for(int j = 0; j < v; j++)
    {
        if(i == j)
            graph[i][j] = 0;
        else
            graph[i][j] = 9999;
    }

    for(int i = 0; i < e; i++)
    {
        System.out.println("Please enter data for Edge " + (i + 1) + ":");
        System.out.print("Source: ");
        int s = Integer.parseInt(br.readLine());
        s--;
        System.out.print("Destination: ");
        int d = Integer.parseInt(br.readLine());
        d--;
        System.out.print("Cost: ");
        int c = Integer.parseInt(br.readLine());
        graph[s][d] = c;
        graph[d][s] = c;
    }

    dvr_calc_disp("The initial Routing Tables are: ");

    System.out.print("Please enter the Source Node for the edge whose cost has changed: ");
    int s = Integer.parseInt(br.readLine());
    s--;
    System.out.print("Please enter the Destination Node for the edge whose cost has changed: ");
```

```

    int d = Integer.parseInt(br.readLine());
    d--;
    System.out.print("Please enter the new cost: ");
    int c = Integer.parseInt(br.readLine());
    graph[s][d] = c;
    graph[d][s] = c;
    dvr_calc_disp("The new Routing Tables are: ");
}

```

```

static void dvr_calc_disp(String message)
{
    System.out.println();
    init_tables();
    update_tables();
    System.out.println(message);
    print_tables();
    System.out.println();
}

```

```

static void update_table(int source)
{
    for(int i = 0; i < v; i++)
    {
        if(graph[source][i] != 9999)
        {
            int dist = graph[source][i];
            for(int j = 0; j < v; j++)
            {
                int inter_dist = rt[i][j];
                if(via[i][j] == source)
                inter_dist = 9999;
                if(dist + inter_dist < rt[source][j])
                {
                    rt[source][j] = dist + inter_dist;
                    via[source][j] = i;
                }
            }
        }
    }
}

```

```

static void update_tables()
{
    int k = 0;
    for(int i = 0; i < 4*v; i++)
    {
        update_table(k);
        k++;
        if(k == v)

```

```
        k = 0;
    }
}

static void init_tables()
{
for(int i = 0; i < v; i++)
{
for(int j = 0; j < v; j++)
{
if(i == j)
{
    rt[i][j] = 0;
    via[i][j] = i;
}
else
{
    rt[i][j] = 9999;
    via[i][j] = 100;
}
}
}
}

static void print_tables()
{
for(int i = 0; i < v; i++)
{
for(int j = 0; j < v; j++)
{
System.out.print("Dist: " + rt[i][j] + "");
}
System.out.println();
}
}

}
```

## Output

```
Command Prompt

D:\ssk>javac DUR.java

D:\ssk>java DUR
Please enter the number of Vertices:
4
Please enter the number of Edges:
5
Please enter data for Edge 1:
Source: 1
Destination: 2
Cost: 1
Please enter data for Edge 2:
Source: 1
Destination: 3
Cost: 3
Please enter data for Edge 3:
Source: 2
Destination: 3
Cost: 1
Please enter data for Edge 4:
Source: 2
Destination: 4
Cost: 1
Please enter data for Edge 5:
Source: 3
Destination: 4
Cost: 4

The initial Routing Tables are:
Dist: 0Dist: 1Dist: 2Dist: 2
Dist: 1Dist: 0Dist: 1Dist: 1
Dist: 2Dist: 1Dist: 0Dist: 2
Dist: 2Dist: 1Dist: 2Dist: 0

Please enter the Source Node for the edge whose cost has changed: 2
```

```
Command Prompt

Please enter the Source Node for the edge whose cost has changed: 2
Please enter the Destination Node for the edge whose cost has changed: 4
Please enter the new cost: 10

The new Routing Tables are:
Dist: 0Dist: 1Dist: 2Dist: 6
Dist: 1Dist: 0Dist: 1Dist: 5
Dist: 2Dist: 1Dist: 0Dist: 4
Dist: 6Dist: 5Dist: 4Dist: 0

D:\ssk>
```

## Result

Thus a program to simulate Distance Vector Routing algorithm is executed successfully.



**Aim**

To write a program to simulate Link State Routing algorithm.

**Algorithm**

1. Create a network.
2. Assign cost to the edges in the network.
3. Initialize the distance of the node itself as zero.
4. The node is taken and chosen as a root node of the tree, this creates the tree with a single node, and now set the total cost of each node to some value based on the information in Link State Database.
5. Now the node selects one node, among all the nodes not in the tree like structure, which is nearest to the root, and adds this to the tree. The shape of the tree gets changed .
6. After this node is added to the tree, the cost of all the nodes not in the tree needs to be updated because the paths may have been changed.
7. The node repeats the Step 5 and Step 6 until all the nodes are added in the tree.
8. Print the shortest path from source to destination.

## Program

```
import java.util.HashSet;
import java.util.InputMismatchException;
import java.util.Iterator;
import java.util.Scanner;
import java.util.Set;
public class Dijkstra
{
    private int distances[];
    private Set<Integer> settled;
    private Set<Integer> unsettled;
    private int number_of_nodes;
    private int adjacencyMatrix[][];
    public Dijkstra(int number_of_nodes)
    {
        this.number_of_nodes = number_of_nodes;
        distances = new int[number_of_nodes + 1];
        settled = new HashSet<Integer>();
        unsettled = new HashSet<Integer>();
        adjacencyMatrix = new int[number_of_nodes + 1][number_of_nodes + 1];
    }
    public void dijkstra_algorithm(int adjacency_matrix[], int source)
    {
        int evaluationNode;
        for (int i = 1; i <= number_of_nodes; i++)
            for (int j = 1; j <= number_of_nodes; j++)
                adjacencyMatrix[i][j] = adjacency_matrix[i][j];

        for (int i = 1; i <= number_of_nodes; i++)
        {
            distances[i] = Integer.MAX_VALUE;
        }

        unsettled.add(source);

        distances[source] = 0;

        while (!unsettled.isEmpty())
        {
            evaluationNode = getNodeWithMinimumDistanceFromUnsettled();

            unsettled.remove(evaluationNode);

            settled.add(evaluationNode);

            evaluateNeighbours(evaluationNode);
        }
    }
}
```

```

    }
private int getNodeWithMinimumDistanceFromUnsettled()

{

    int min;
    int node = 0;
    Iterator<Integer> iterator = unsettled.iterator();
    node = iterator.next();
    min = distances[node];
    for (int i = 1; i<= distances.length; i++)

        {

            if (unsettled.contains(i))

                {

                    if (distances[i] <= min)

                        {

                            min = distances[i];

                            node = i;

                        }

                }

        }

    return node;

}
private void evaluateNeighbours(int evaluationNode)

{

    int edgeDistance = -1;

    int newDistance = -1;

    for (int destinationNode = 1; destinationNode<= number_of_nodes; destinationNode++)
    {

        if (!settled.contains(destinationNode))

            {

                if (adjacencyMatrix[evaluationNode][destinationNode] != Integer.MAX_VALUE)

                    {

                        edgeDistance = adjacencyMatrix[evaluationNode][destinationNode];
                        newDistance = distances[evaluationNode] + edgeDistance;

```

```

        if (newDistance < distances[destinationNode])
        {
            distances[destinationNode] = newDistance;
        }
    }
    unsettled.add(destinationNode);
}

}

}

}

}

public static void main(String[] arg)
{
    int adjacency_matrix[][];

    int number_of_vertices;

    int source = 0, destination = 0;

    Scanner scan = new Scanner(System.in);

    try
    {
        System.out.println("Enter the number of vertices");

        number_of_vertices = scan.nextInt();

        adjacency_matrix = new int[number_of_vertices + 1][number_of_vertices + 1];
        System.out.println("Enter the Weighted Matrix for the graph");

        for (int i = 1; i <= number_of_vertices; i++)
        {
            for (int j = 1; j <= number_of_vertices; j++)
            {
                adjacency_matrix[i][j] = scan.nextInt();

                if (i == j)
                {
                    adjacency_matrix[i][j] = 0;
                }
            }
        }
    }
    catch (Exception e)
    {
        System.out.println("Invalid Input");
    }
}

```

```

        continue;

    }

    if (adjacency_matrix[i][j] == 0)

    {
adjacency_matrix[i][j] = Integer.MAX_VALUE;

    }

    }

    }

System.out.println("Enter the source ");

    source = scan.nextInt();

System.out.println("Enter the destination ");

    destination = scan.nextInt();

Dijkstra dijkstrasAlgorithm = new Dijkstra(number_of_vertices);

dijkstrasAlgorithm.dijkstra_algorithm(adjacency_matrix, source);
System.out.println("The Shorted Path from " + source + " to " + destination + " is: ");

    for (int i = 1; i<= dijkstrasAlgorithm.distances.length - 1; i++)

    {

        if (i == destination)

System.out.println(dijkstrasAlgorithm.distances[i]);

    }

    } catch (InputMismatchException inputMismatch)

    {

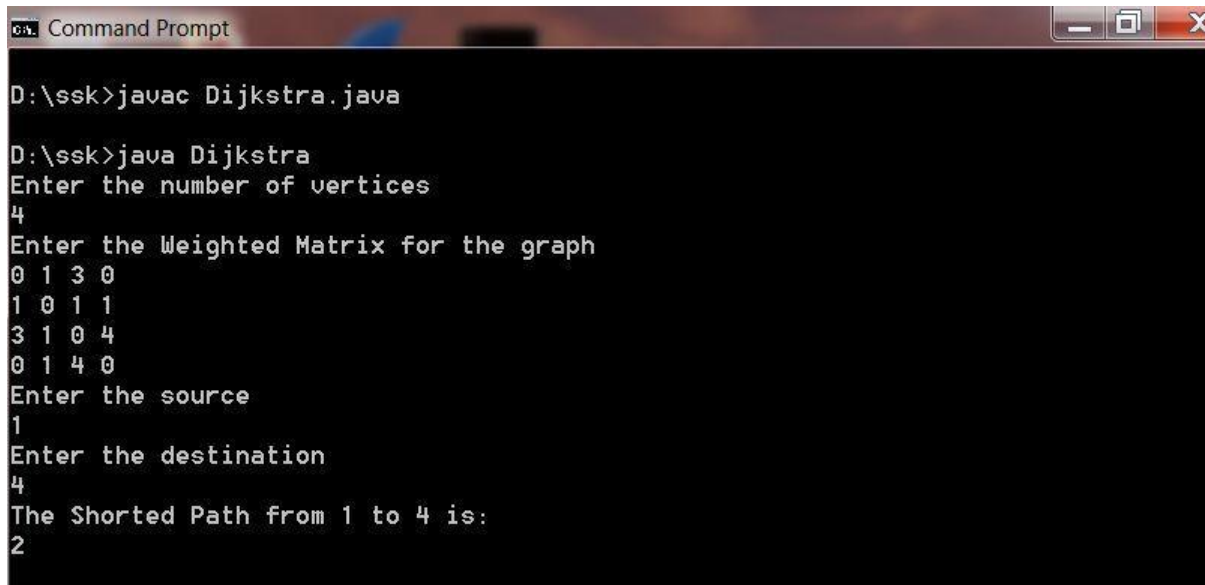
System.out.println("Wrong Input Format");
    }
scan.close();

    }

}

```

## Output



```
GA. Command Prompt
D:\ssk>javac Dijkstra.java
D:\ssk>java Dijkstra
Enter the number of vertices
4
Enter the Weighted Matrix for the graph
0 1 3 0
1 0 1 1
3 1 0 4
0 1 4 0
Enter the source
1
Enter the destination
4
The Shorted Path from 1 to 4 is:
2
```

## Result

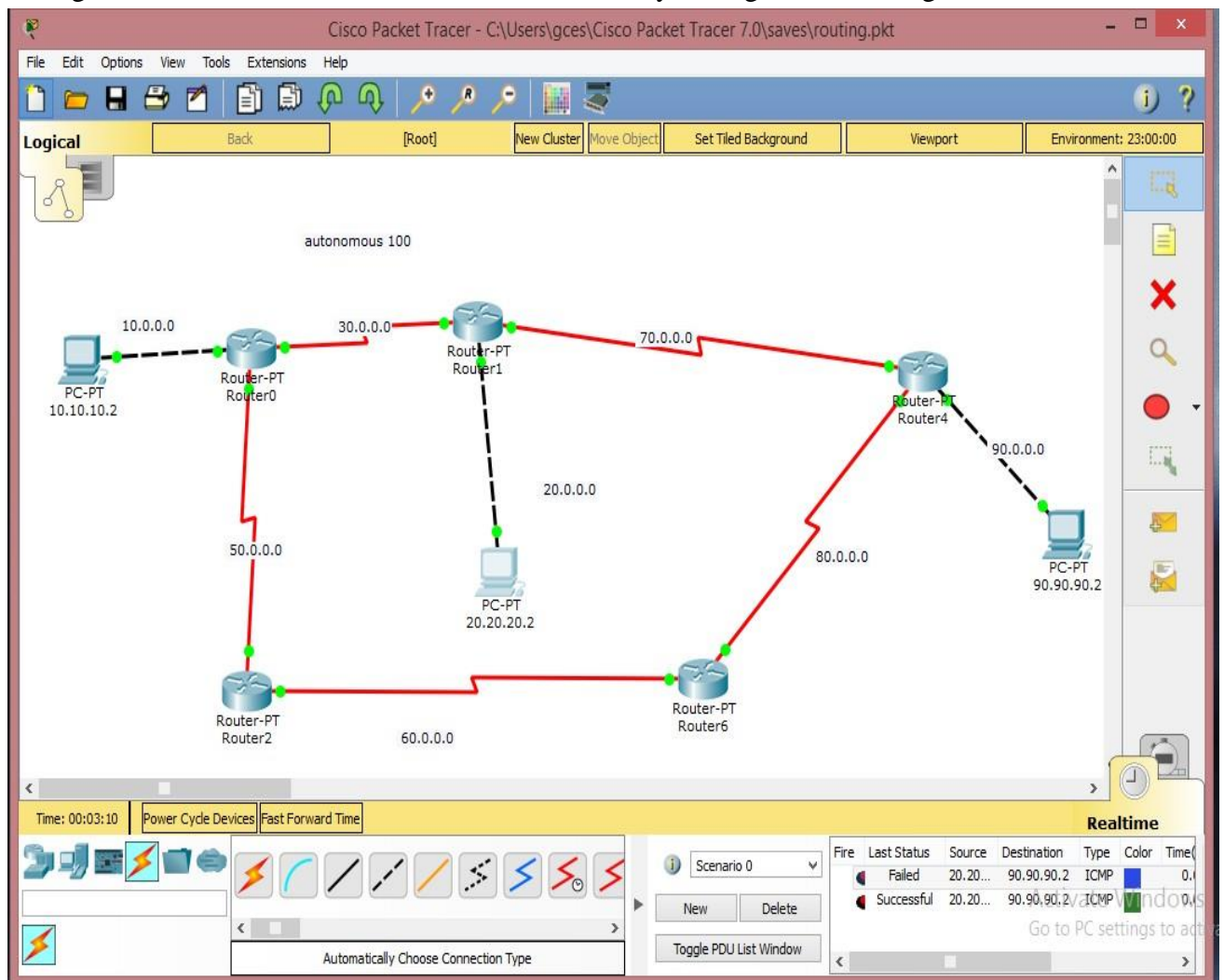
Thus a program to simulate Link State Routing algorithm is executed successfully.

**Aim**

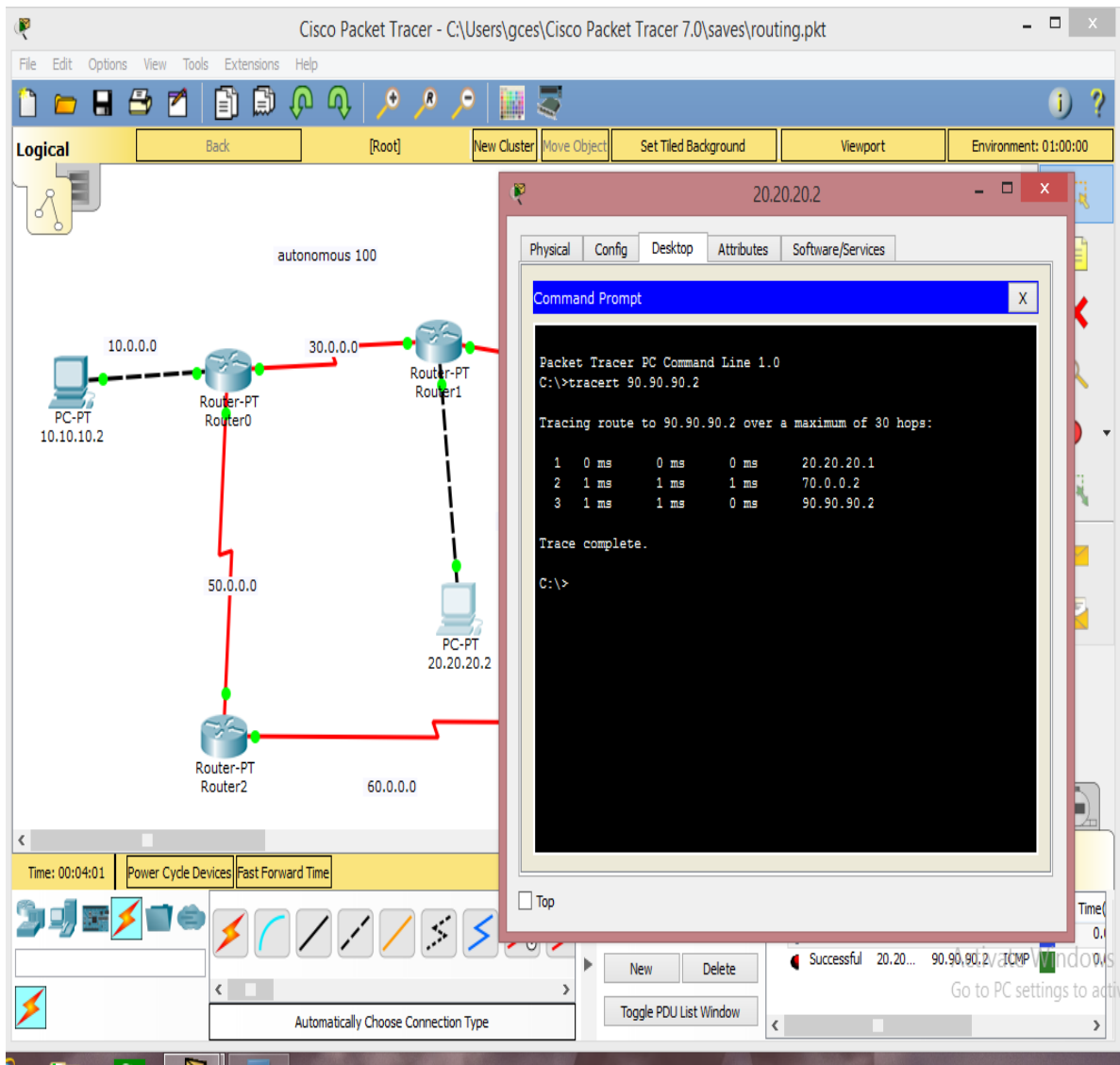
To evaluate the performance of routing protocols RIP and OSPF using Packet Tracer.

**Procedure: Routing Information Protocol (RIP)**

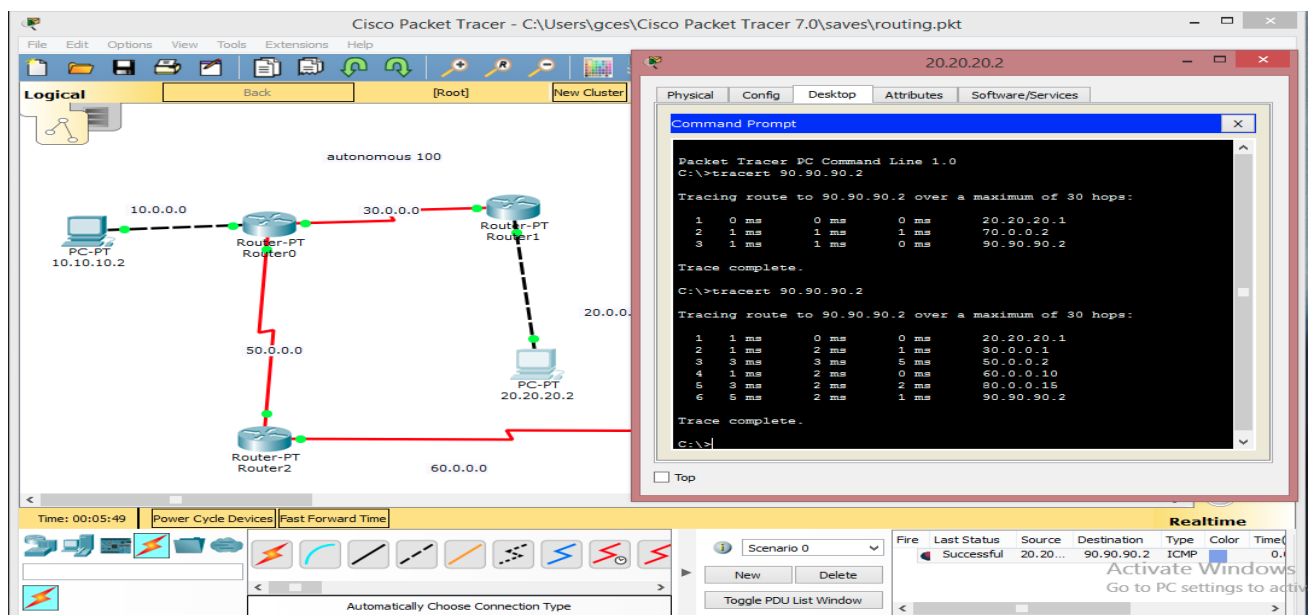
1. Develop a Topology shown in figure given below.
2. Configure all Routers.
3. Implement RIP protocols in Router to configure Network.
4. Assign IP address for all the devices as shown in the figure.
5. Configure all the routers with the network connected by adding network using RIP.



6. Send a packet from source host with IP address 20.20.20.2 to destination host 90.90.90.2.
7. Trace the path of the packet using tracert command and the shortest path is shown in figure below.



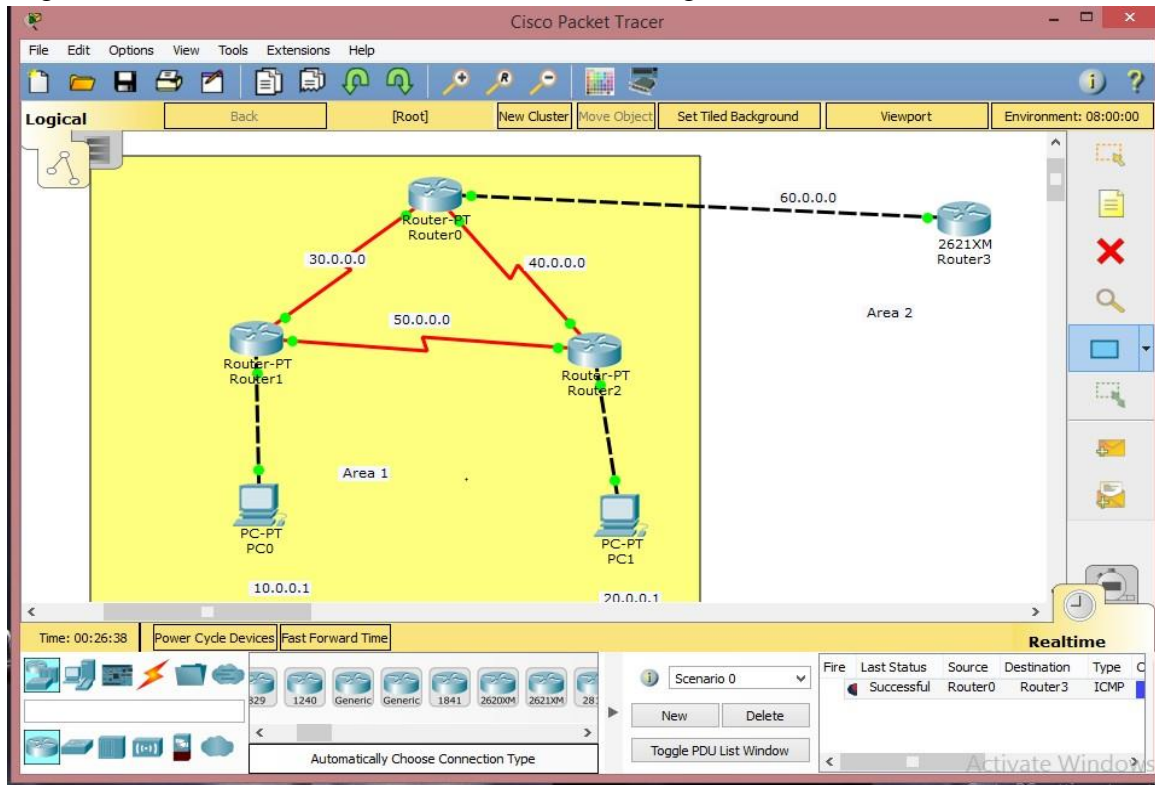
8. Remove the network with network address 70.0.0.0.
9. Send a packet from source host with IP address 20.20.20.2 to destination host 90.90.90.2.
10. Trace the path of the packet using `tracert` command and the shortest path is shown in figure below.



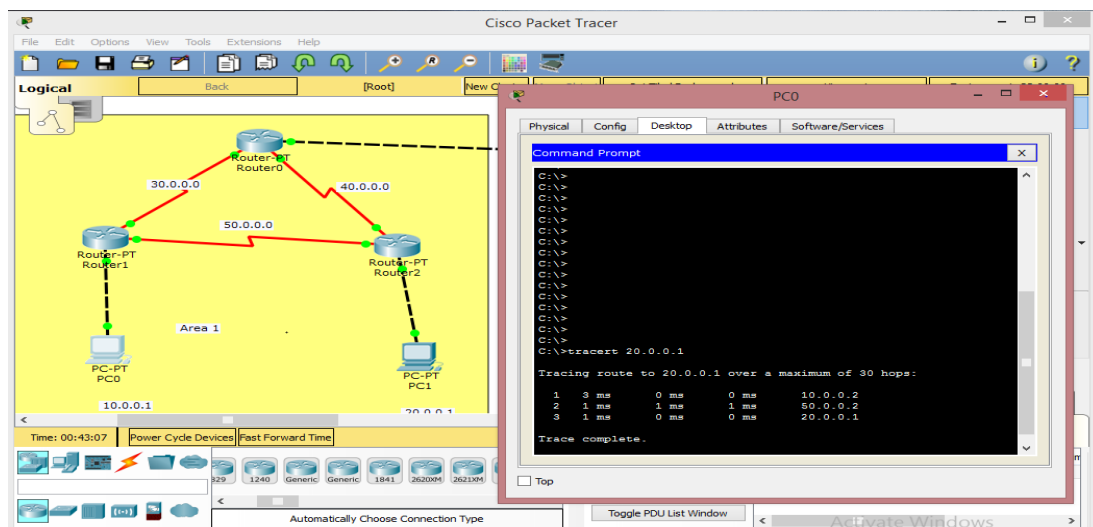


## Procedure: Open Source Shortest Path (OSPF)

1. Develop a Topology shown in figure given below.
2. Configure all Routers.
3. Implement OSPF protocols in Router to configure Network.
4. Perform OSPF area configuration to configure the route.
5. Router 0, router 1, and router 2 belongs to area 1 and Router 3 belongs to area 2.
6. Assign IP address for all the devices as shown in the figure.



7. Send a packet from source host with IP address 10.0.0.1 to destination host 20.0.0.1.
8. Trace the path of the packet using tracert command and the shortest path is shown in figure below.



## Configuring OSPF in Routers:

### Router 0

Router#

Router#conf t

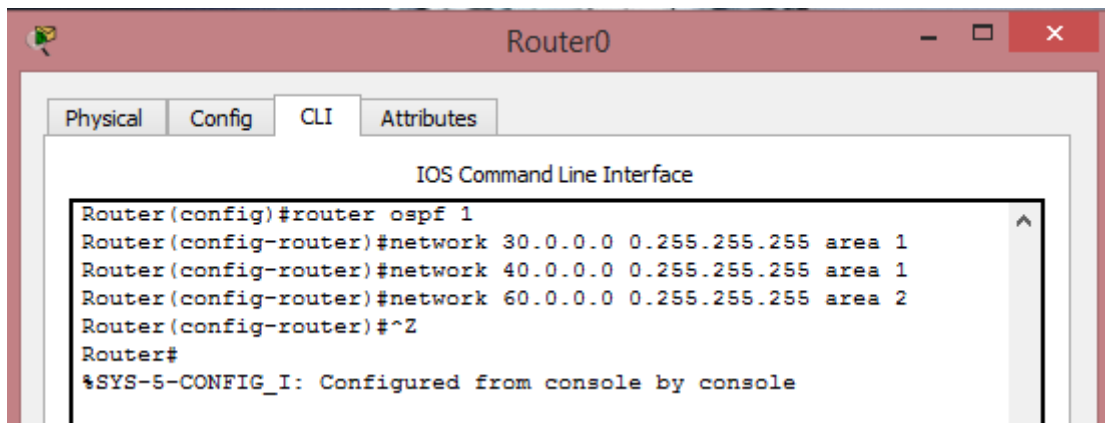
Router(config)#router ospf 1

Router(config-router)#network 30.0.0.0 0.255.255.255 area 1

Router(config-router)#network 40.0.0.0 0.255.255.255 area 1

Router(config-router)#network 60.0.0.0 0.255.255.255 area 2

Router(config-router)#^Z (Use Ctrl + Z to save settings)



### Router 1

Router#

Router#conf t

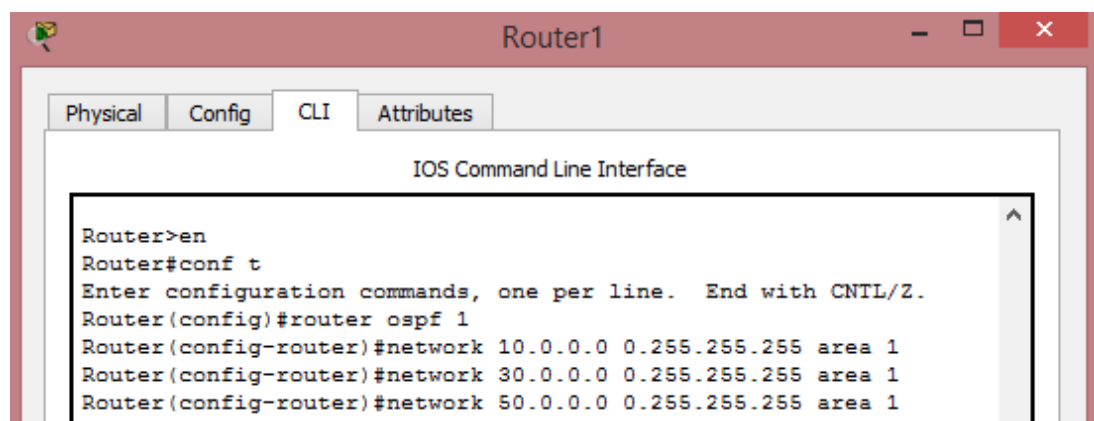
Router(config)#router ospf 1

Router(config-router)#network 10.0.0.0 0.255.255.255 area 1

Router(config-router)#network 30.0.0.0 0.255.255.255 area 1

Router(config-router)#network 50.0.0.0 0.255.255.255 area 1

Router(config-router)#^Z



## Router 2

Router#

Router#conf t

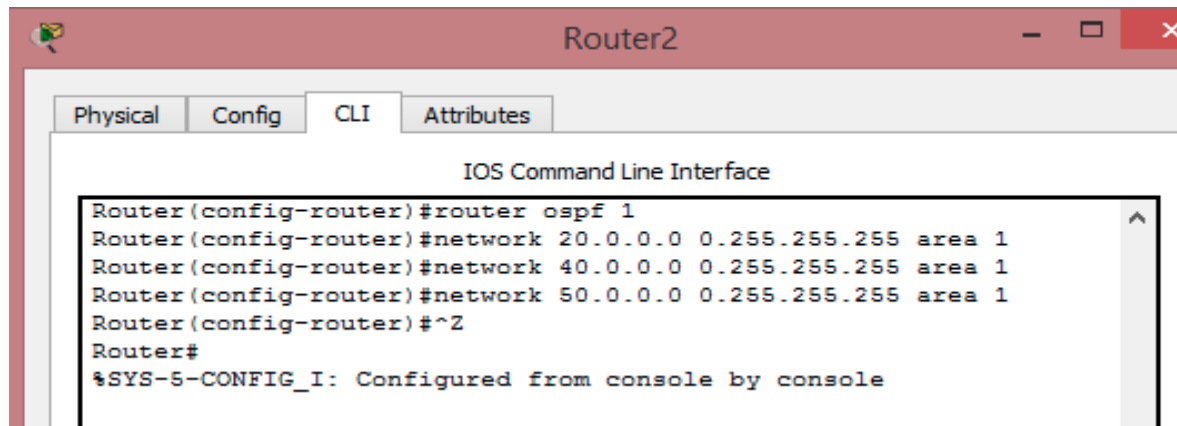
Router(config)#router ospf 1

Router(config-router)#network 20.0.0.0 0.255.255.255 area 1

Router(config-router)#network 40.0.0.0 0.255.255.255 area 1

Router(config-router)#network 50.0.0.0 0.255.255.255 area 1

Router(config-router)#^Z



## Router 3

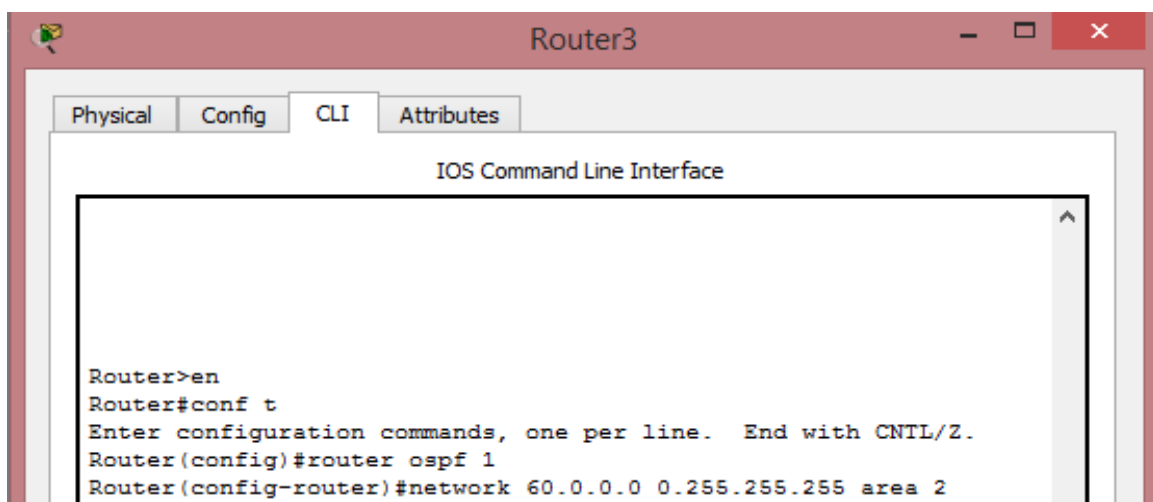
Router#

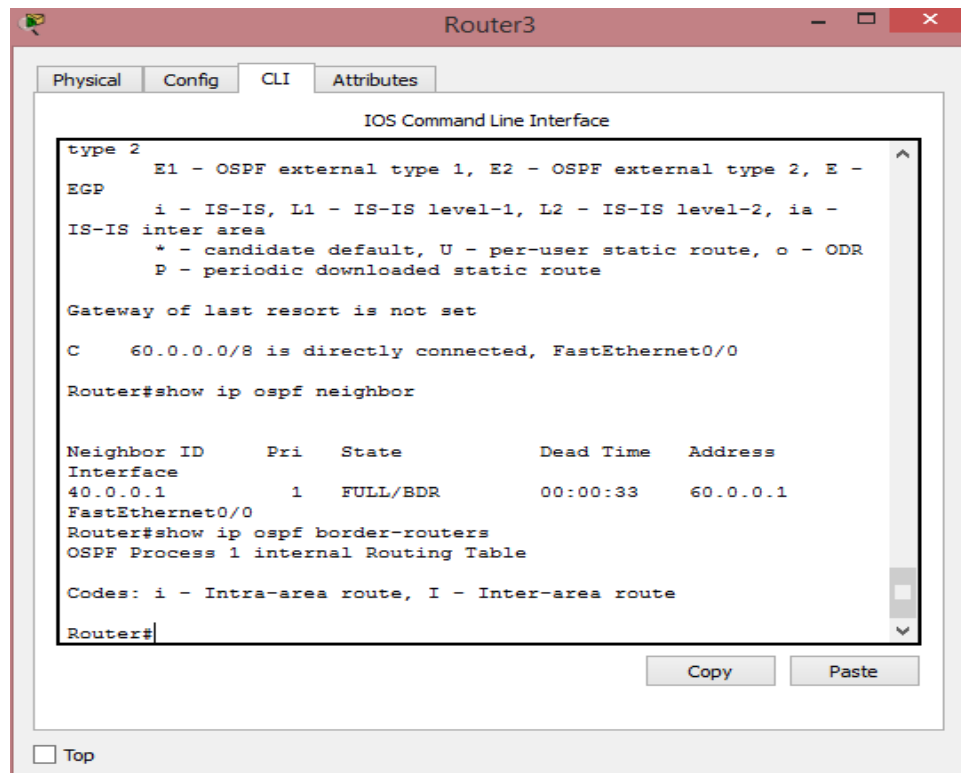
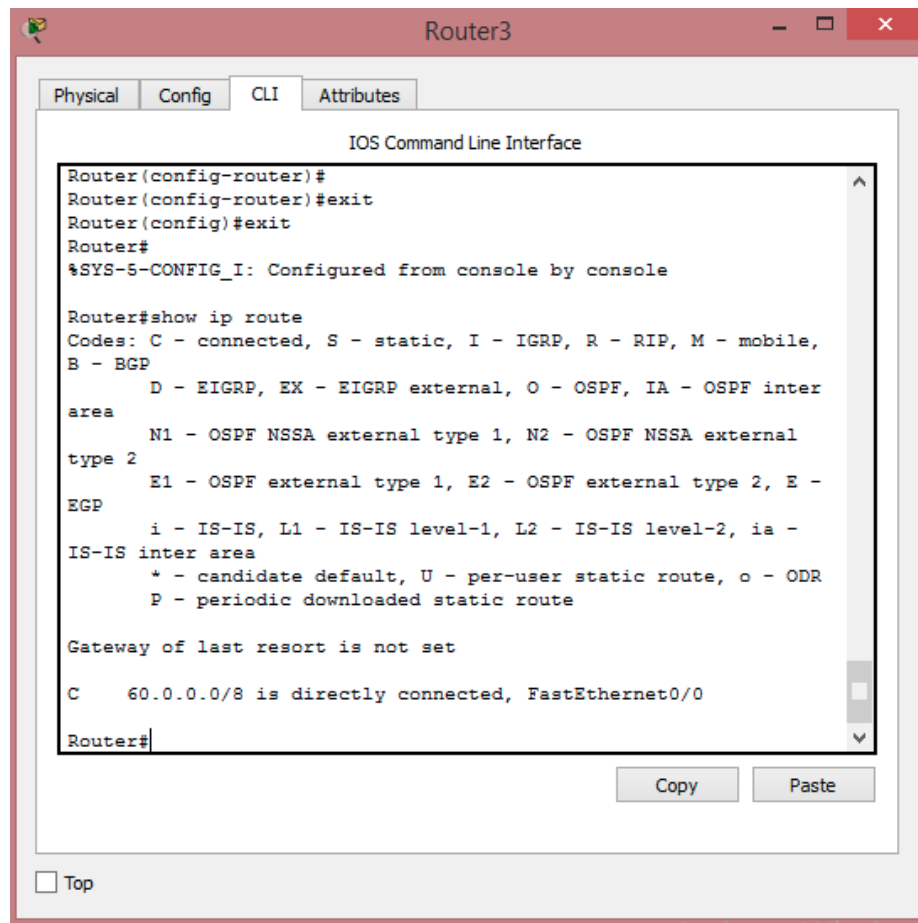
Router#conf t

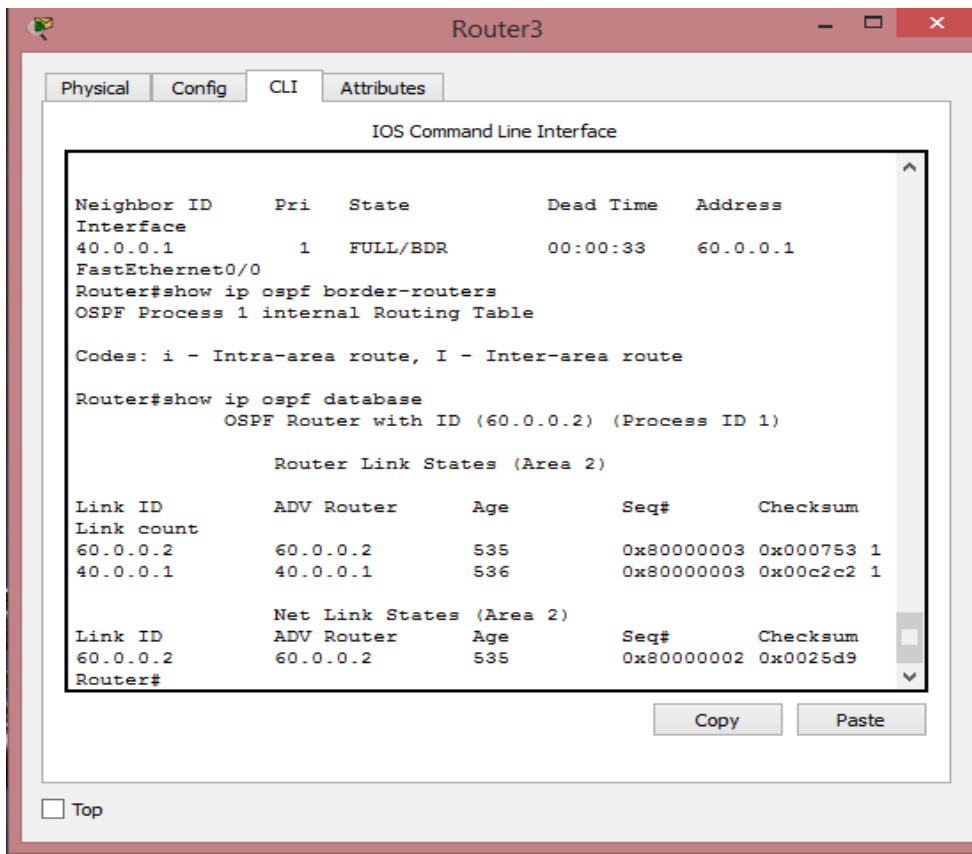
Router(config)#router ospf 1

Router(config-router)#network 60.0.0.0 0.255.255.255 area 2

Router(config-router)#^Z







## Result

Thus ,To evaluate the performance of routing protocols RIP and OSPF using Packet Tracer is performed successfully.

**Aim**

To write a program to simulate error correction code-Cyclic Redundancy Check (CRC).

**Algorithm**

1. Read a polynomial generator in bits. the value of is mutually agreed upon by the sending and the receiving parties.
2. Read the number of bits to be sent.

**Sender Side**

3. The binary data is first augmented by adding k-1 zeros in the end of the data
4. Use modulo-2 binary division to divide binary data by the key and store remainder of division.
5. Append the remainder at the end of the data to form the encoded data and send the same

**Receiver Side**

6. Perform modulo-2 division again and if remainder is 0, then there are no errors. Otherwise print error.

## Program

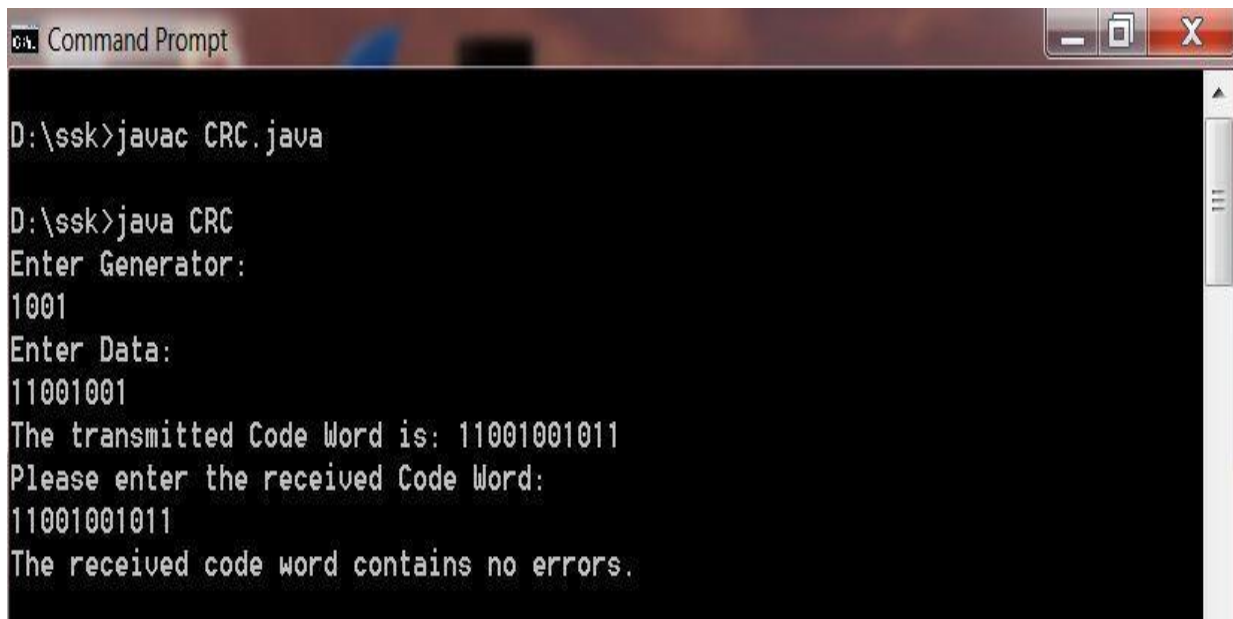
```
import java .io.*;
import java.io.BufferedReader;
public class CRC {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter Generator:");
        String gen = br.readLine();
        System.out.println("Enter Data:");
        String data = br.readLine();
        String code = data;
        while(code.length() < (data.length() + gen.length() - 1))
            code = code + "0";
        code = data + div(code,gen);
        System.out.println("The transmitted Code Word is: " + code);
        System.out.println("Please enter the received Code Word: ");
        String rec = br.readLine();
        if(Integer.parseInt(div(rec,gen)) == 0)
            System.out.println("The received code word contains no errors.");
        else
            System.out.println("The received code word contains errors.");
    }
    static String div(String num1,String num2)
    {
        int pointer = num2.length();
        String result = num1.substring(0, pointer);
        String remainder = "";
        for(int i = 0; i < num2.length(); i++)
        {
            if(result.charAt(i) == num2.charAt(i))
                remainder += "0";
            else
                remainder += "1";
        }
        while(pointer < num1.length())
        {
            if(remainder.charAt(0) == '0')
```

```
{
    remainder = remainder.substring(1, remainder.length());
    remainder = remainder + String.valueOf(num1.charAt(pointer));
    pointer++;
}
result = remainder;
remainder = "";
for(int i = 0; i < num2.length(); i++)
{
    if(result.charAt(i) == num2.charAt(i))
        remainder += "0";
    else
        remainder += "1";
}
}
return remainder.substring(1,remainder.length());

}
}
```



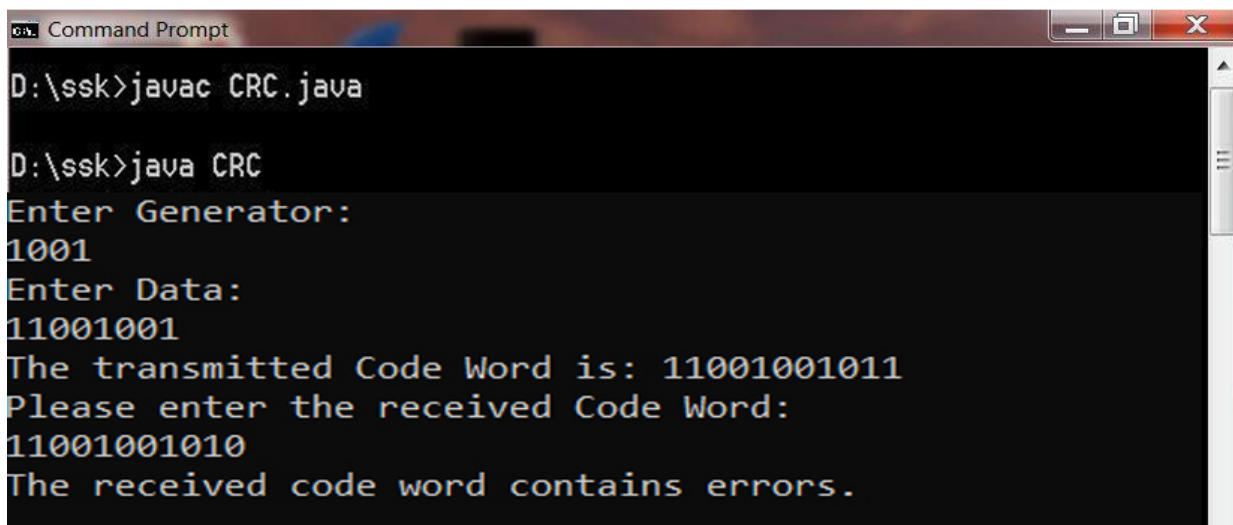
## Output



```
Command Prompt

D:\ssk>javac CRC.java

D:\ssk>java CRC
Enter Generator:
1001
Enter Data:
11001001
The transmitted Code Word is: 11001001011
Please enter the received Code Word:
11001001011
The received code word contains no errors.
```



```
Command Prompt

D:\ssk>javac CRC.java

D:\ssk>java CRC
Enter Generator:
1001
Enter Data:
11001001
The transmitted Code Word is: 11001001011
Please enter the received Code Word:
11001001010
The received code word contains errors.
```

## Result

Thus a program to simulate error correction code-Cyclic Redundancy Check (CRC) is executed successfully.