Ex No: 1 A	IMPLEMENTATION OF BUS TOPOLOGY USING CISCO PACKET TRACER
Date:	

AIM:

To implementation the bus topology using cisco packet tracer.

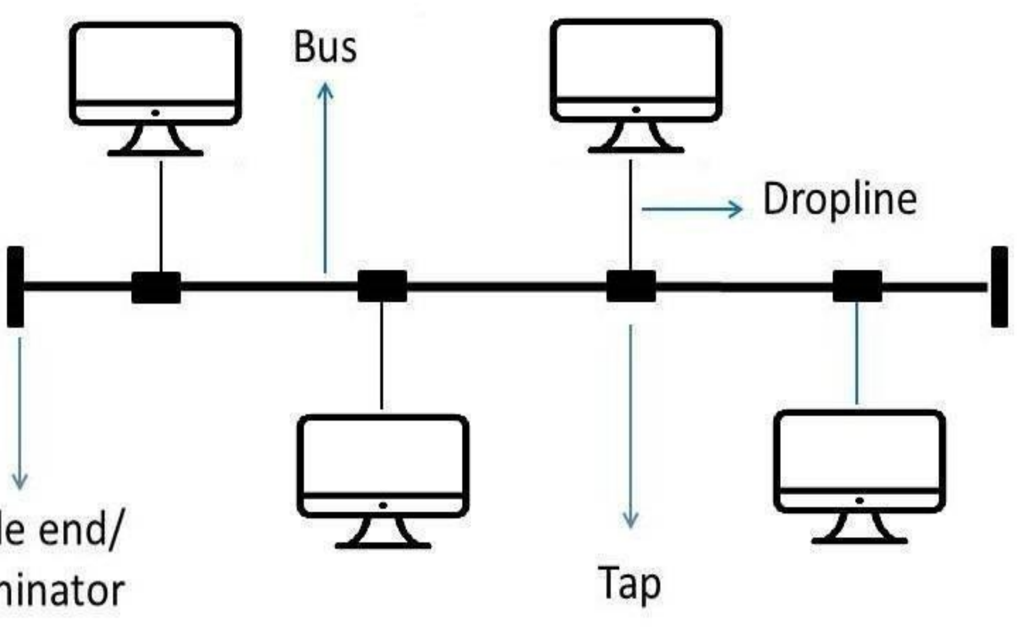
DISCRIPTION:

A bus topology is a network in which nodes are directly linked with a common half-duplex link. A host on a bus topology is called a station. In a bus network, every station will accept all network packets, and these packets generated by each station have equal information priority. A bus network includes a single network segment and collision domain.

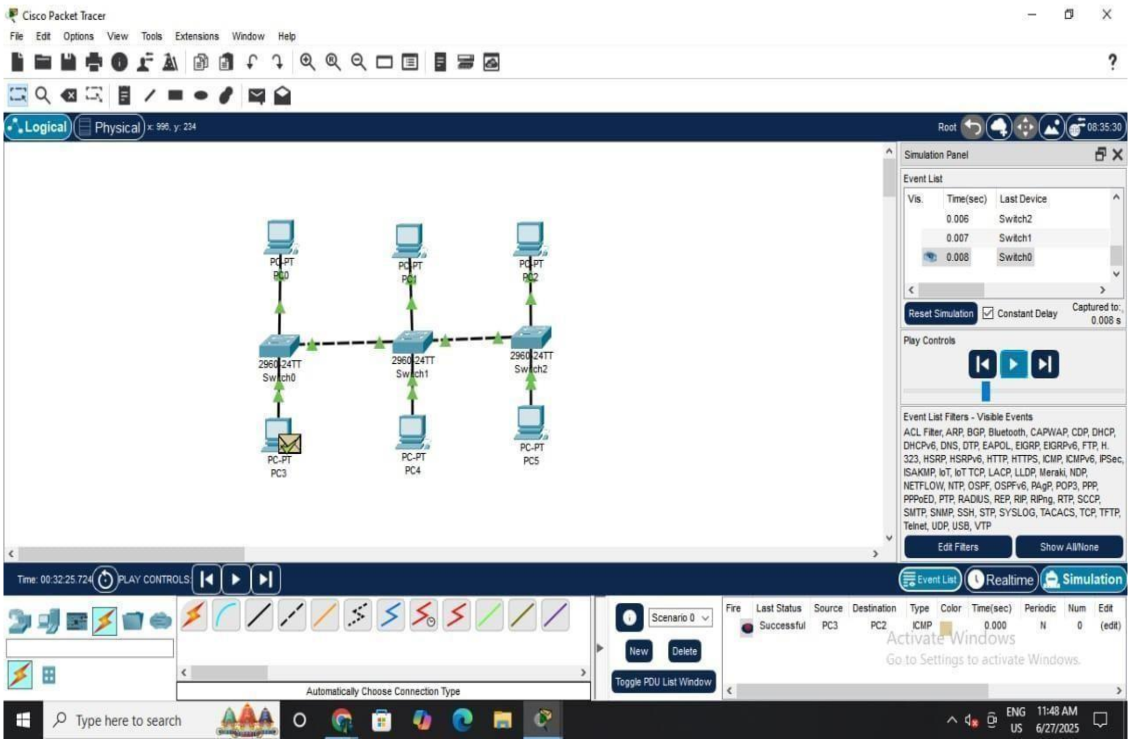
PROCEDURE:

1. Open cisco packet tracer.
2. Add 6PCs and 3 switches.
3. Connect each PC to the switch using copper straight-through cables.
4. Assign each PC and IP address in the same network(ex:192.168.1.10,192.168.1.11)
5. Test by pinging PCs from each other.

DIAGRAM:



OUTPUT:



Event List

Vis.	Time(sec)	Last Device
	0.006	Switch2
	0.007	Switch1
	0.008	Switch0

Captured to: 0.008 s

Event List Filters - Visible Events
ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EAPOL, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, IoT, IoT TCP, LACP, LLDP, Meraki, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit
●	Successful	PC3	PC2	ICMP	🟨	0.000	N	0	(edit)

Activate Windows
Go to Settings to activate Windows.

CLASS PERFORMANCE	
RECORD	
VIVA	
TOTAL	

RESULT:

Thus the implementation of bus topology using cisco packet tracer has been executed successfully.

Ex No: 1 B	IMPLEMENTATION OF RING TOPOLOGY USING CISCO PACKET TRACER
Date:	

AIM:

To implementation the ring topology using cisco packet tracer.

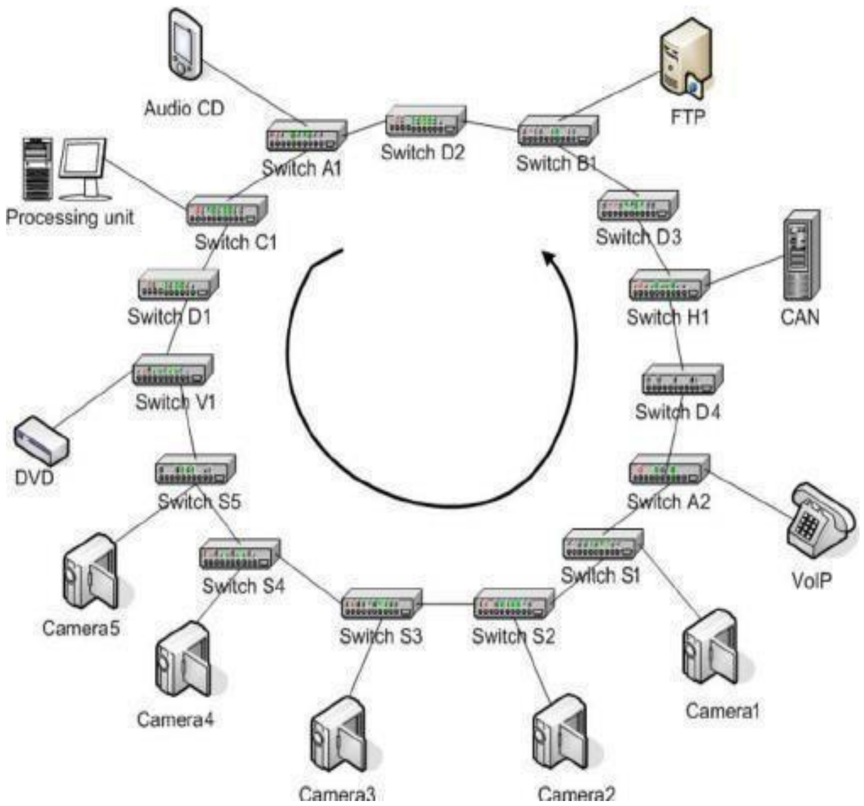
DISCRIPTION:

Ring topology is a network setup where each device is connected to two others, forming a circular path for data to travel. Data moves in one direction (or both in a dual ring) and passes through each device until it reaches its destination. It helps prevent data collisions and works well under heavy load, but a single device failure can affect the entire network.

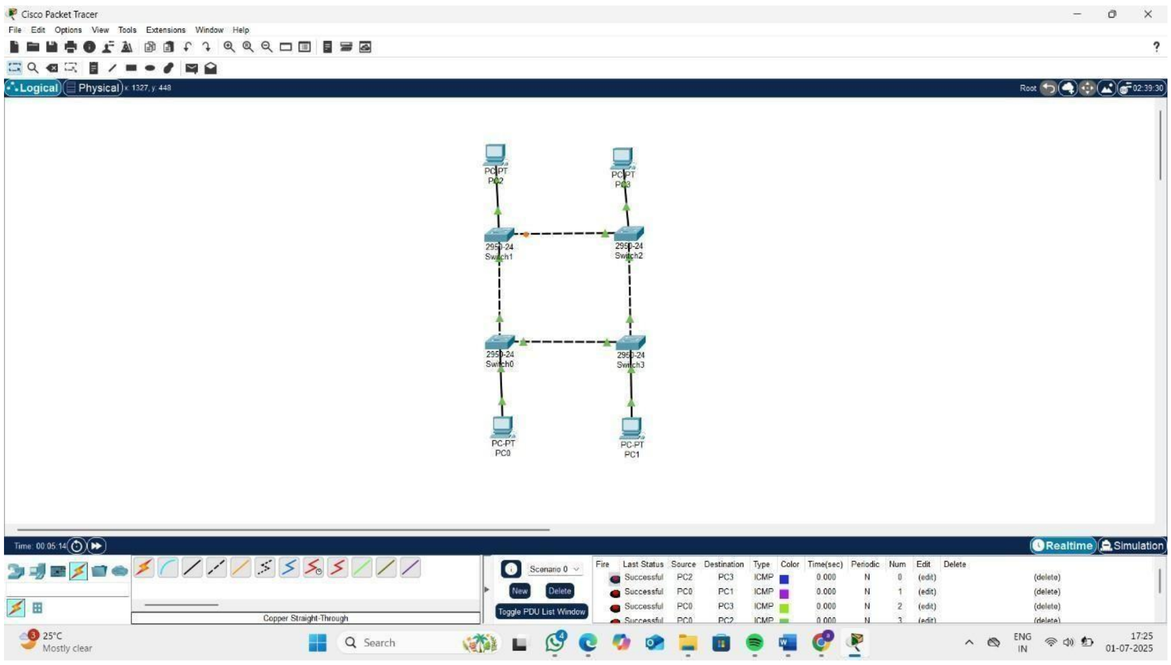
PROCEDURE:

1. Connect all devices in a circular loop.
2. Give each device a unique address.
3. Send data from the source device.
4. Data passes through each device in the ring.
5. Data reaches the destination and gets acknowledged.

DIAGRAM:



OUTPUT:



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC2	PC3	ICMP	🟦	0.000	N	0	(edit)	(delete)
●	Successful	PC0	PC1	ICMP	🟪	0.000	N	1	(edit)	(delete)
●	Successful	PC0	PC3	ICMP	🟩	0.000	N	2	(edit)	(delete)
●	Successful	PC0	PC2	ICMP	🟩	0.000	N	3	(edit)	(delete)

PROGRAM &EXECUTION	
CLASS PERFORMNCE	
VIVA	
TOTAL	

RESULT:

Thus the implementation of ring topology using cisco packet tracer has been executed successfully.

Ex No: 1 C	**IMPLEMENTATION OF STAR TOPOLOGY USING CISCO PACKET TRACER**
Date:	

AIM:

To implementation the star topology using cisco packet tracer.

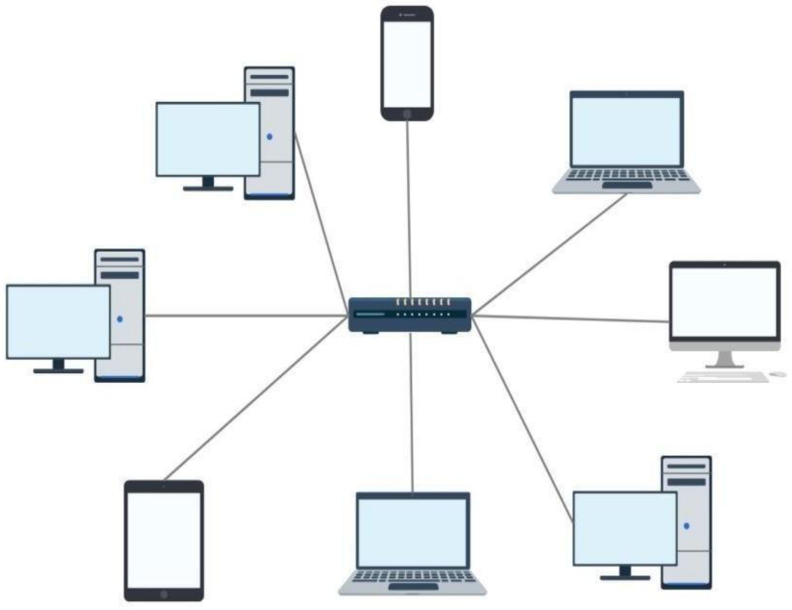
DISCRIPTION:

Star topology is a network layout where all devices are connected to a central hub or switch. The hub acts as a controller to manage data transmission between devices. It is easy to install and troubleshoot, as a failure in one device does not affect the rest of the network. However, if the central hub fails, the whole network stops working. This topology is commonly used in modern LAN setups.

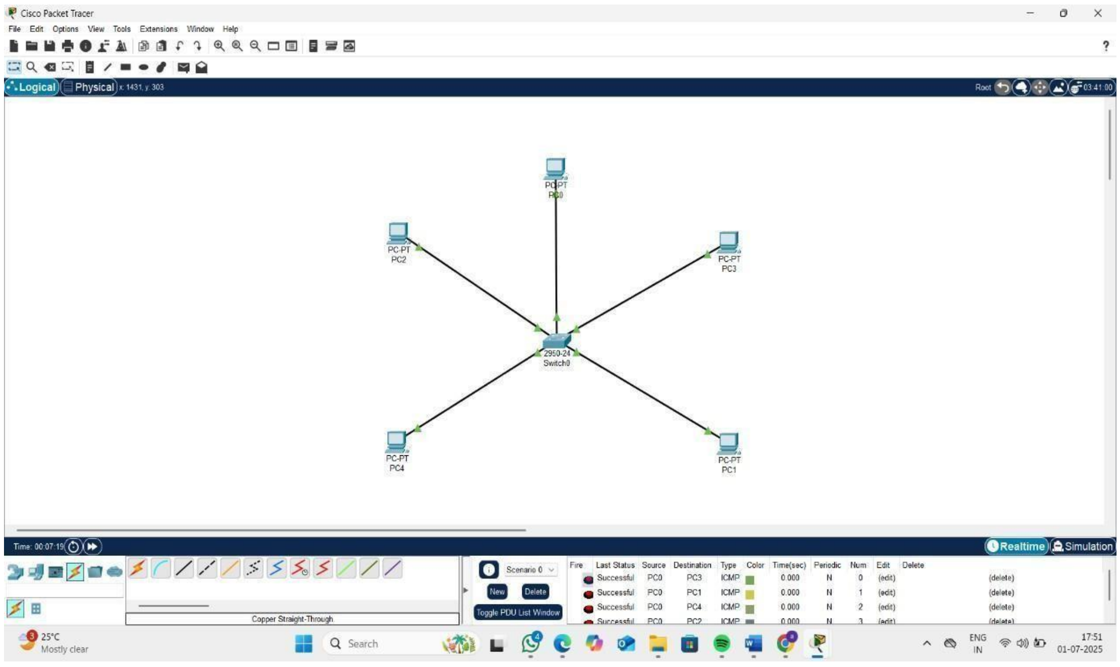
PROCEDURE:

1. Choose a central device like a hub or switch.
2. Connect each computer/device to the hub using network cables.
3. Power on all devices including the hub/switch.
4. Configure network settings (like IP addresses) on each device if needed.
5. Test the network by sharing files or pinging between devices.

DIAGRAM:



OUTPUT:



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC0	PC3	ICMP	🟩	0.000	N	0	(edit)	(delete)
●	Successful	PC0	PC1	ICMP	🟨	0.000	N	1	(edit)	(delete)
●	Successful	PC0	PC4	ICMP	🟩	0.000	N	2	(edit)	(delete)
●	Successful	PC0	PC2	ICMP	▨	0.000	N	3	(edit)	(delete)

PROGRAM &EXECUTION	
CLASS PERFORMNCE	
VIVA	
TOTAL	

RESULT:

Thus the implementation of star topology using cisco packet tracer has been executed successfully.

Ex No: 1 D	**IMPLEMENTATION OF MESH TOPOLOGY USING CISCO PACKET TRACER**
Date:	

AIM:

To implementation the mesh topology using cisco packet tracer.

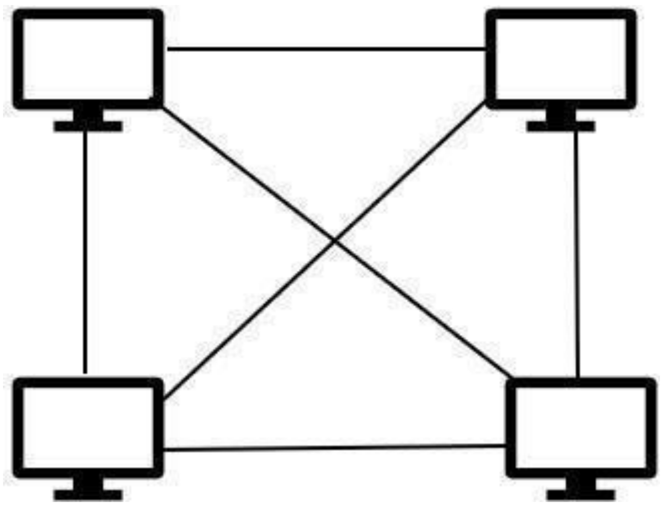
DISCRIPTION:

Mesh topology is a network setup where each device is connected to every other device in the network. This provides multiple paths for data to travel, making the network highly reliable and fault-tolerant. If one link fails, data can still be routed through other paths. However, it requires more cables and is more complex to install and maintain, so it is mostly used in critical systems like military or data centers.

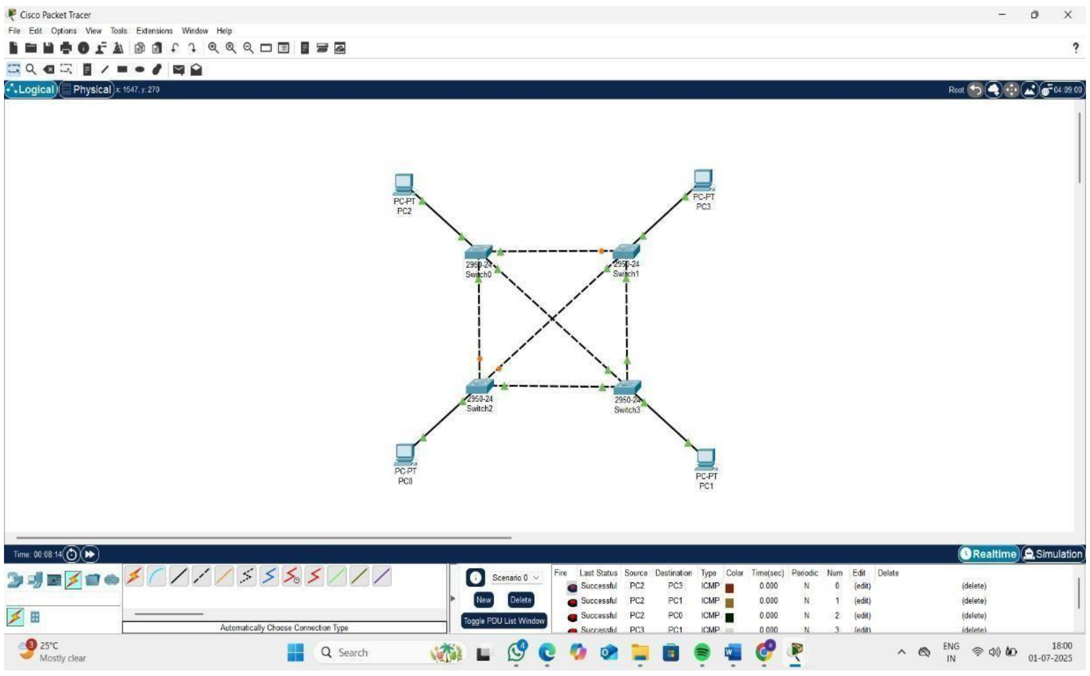
PROCEDURE:

1. Identify all devices that need to be connected in the network.
2. Connect each device to every other device using separate cables.
3. Ensure proper configuration of IP addresses or routing settings on each device.
4. Power on all devices and verify physical connections.
5. Test connectivity between all pairs of devices to ensure full communication.

DIAGRAM:



OUTPUT:



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC2	PC3	ICMP	🟥	0.000	N	0	(edit)	(delete)
●	Successful	PC2	PC1	ICMP	🟫	0.000	N	1	(edit)	(delete)
●	Successful	PC2	PC0	ICMP	⬛	0.000	N	2	(edit)	(delete)
●	Successful	PC3	PC1	ICMP	⬜	0.000	N	3	(edit)	(delete)

PROGRAM &EXECUTION	
CLASS PERFORMNCE	
VIVA	
TOTAL	

RESULT:

Thus the implementation of mesh topology using cisco packet tracer has been executed successfully.

Ex No: 2	**Learn to use commands like tcpdump, netstat, ifconfig, nslookup and traceroute.**
Date:	**Capture ping and traceroute PDUs using a network protocol analyzer and examine**

AIM:

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

1. Tcpdump

The tcpdump utility allows you to capture packets that flow within your network to assist in network troubleshooting. The following are several examples of using tcpdump with different options. Traffic is captured based on a specified filter.

Options	**Description**
-D	Print a list of Network interfaces
-i	Specify an interface on which to capture
-c	Specify the number of packets to receive
-v, -vv, -vvv	Increase the level of detail (verbosity)
-w	Write captured data to a file.
-r	Read captured data from a file.

Many other options and arguments can be used with tcpdump. The following are some specific examples of the power of the tcpdump utility.

1. Display traffic between 2 hosts

To display all traffic between two hosts (represented by variables host1 and host2):

```
# tcpdump host host1 and host2
```

2. Display traffic from a source or destination host only

To display traffic from only a source (src) or destination (dst) host:

```
# tcpdump src host
# tcpdump dst host
```

3. Display traffic for a specific protocol

Provide the protocol as an argument to display only traffic for a specific protocol, for example tcp, udp, icmp, arp:

```
# tcpdump protocol
```

For example to display traffic only for the tcp traffic :

```
# tcpdump tcp
```

4. Filtering based on source or destination port

To filter based on a source or destination port:

```
# tcpdump src port ftp
# tcpdump dst port http
```

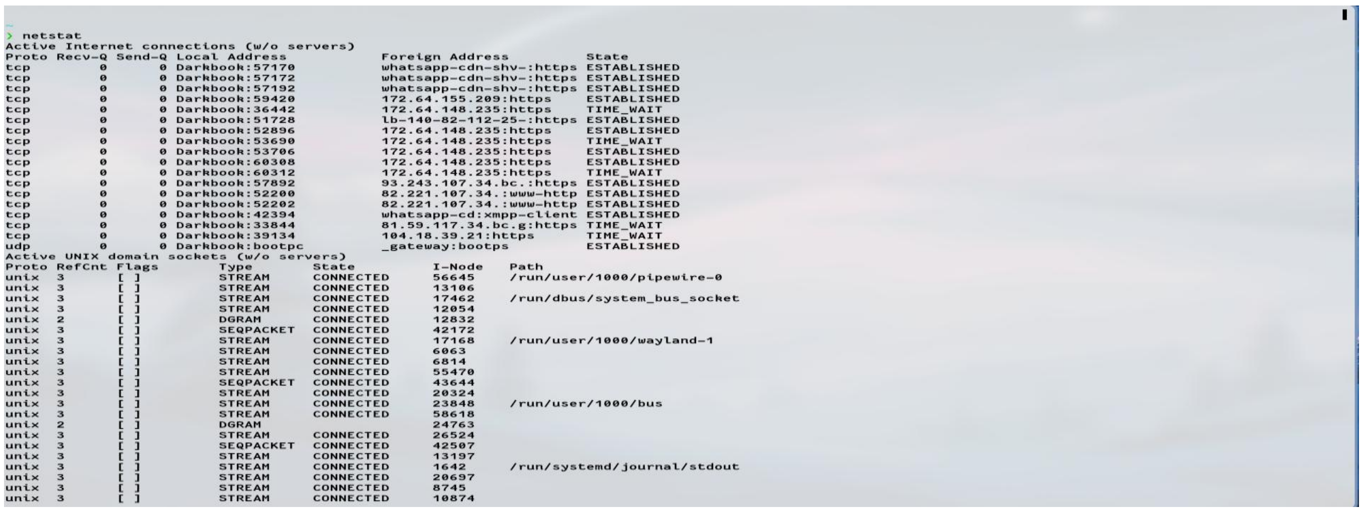
2. Netstat

Netstat is a common command line TCP/IP networking available in most versions of Windows, Linux, UNIX and other operating systems. Netstat provides information and statistics about protocols in use and current TCP/IP network connections. The Windows help screen (analogous to a Linux or UNIX for netstat reads as follows:

Displays protocol statistics and current TCP/IP network connections.

NETSTAT -a -b -e -n -o -p proto -r -s -v interval

-a	Displays all connections and listening ports.
-b	Displays the executable involved in creating each connection or listening port. In some cases well-known executables host multiple independent components, and in these cases the sequence of components involved in creating the connection or listening port is displayed. In this case the executable name is in [] at the bottom, on top is the component it called, and so forth until TCP/IP was reached. Note that this option can be time-consuming and will fail unless you have sufficient permissions
-e	Displays Ethernet statistics. This may be combined with the -s option
-n	Displays addresses and port numbers in numerical form
-o	Displays the owning process ID associated with each connection
-p proto	Shows connections for the protocol specified by proto; proto may be any of: TCP, UDP, TCPv6, or UDPv6. If used with the -s option to display per-protocol statistics, proto may be any of: IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP, or UDPv6
-r	Displays the routing table.
-s	Displays per-protocol statistics. By default, statistics are shown for IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP, and UDPv6; the -p option may be used to specify a subset of the default.
-v	When used in conjunction with -b, will display sequence of components involved in creating the connection or listening port for all executables
Interval	When used in conjunction with -b, will display sequence of components involved in creating the connection or listening port for all executables



```
> netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 Darkbook:57170          whatsapp-cdn-shv-:https ESTABLISHED
tcp        0      0 Darkbook:57172          whatsapp-cdn-shv-:https ESTABLISHED
tcp        0      0 Darkbook:57192          whatsapp-cdn-shv-:https ESTABLISHED
tcp        0      0 Darkbook:59420          172.64.155.209:https    ESTABLISHED
tcp        0      0 Darkbook:36442          172.64.148.235:https    TIME_WAIT
tcp        0      0 Darkbook:51728          lb-140-82-112-25-:https ESTABLISHED
tcp        0      0 Darkbook:52896          172.64.148.235:https    ESTABLISHED
tcp        0      0 Darkbook:53690          172.64.148.235:https    TIME_WAIT
tcp        0      0 Darkbook:53706          172.64.148.235:https    ESTABLISHED
tcp        0      0 Darkbook:60308          172.64.148.235:https    ESTABLISHED
tcp        0      0 Darkbook:60312          172.64.148.235:https    TIME_WAIT
tcp        0      0 Darkbook:57892          93.243.107.34.bc.:https ESTABLISHED
tcp        0      0 Darkbook:52200          82.221.107.34.:www-http ESTABLISHED
tcp        0      0 Darkbook:52202          82.221.107.34.:www-http ESTABLISHED
tcp        0      0 Darkbook:42394          whatsapp-cd:xmpp-client ESTABLISHED
tcp        0      0 Darkbook:33844          81.59.117.34.bc.g:https TIME_WAIT
tcp        0      0 Darkbook:39134          104.18.39.21:https      TIME_WAIT
udp        0      0 Darkbook:bootpc         _gateway:bootps         ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  3      [ ]         STREAM     CONNECTED     56645    /run/user/1000/pipewire-0
unix  3      [ ]         STREAM     CONNECTED     13106
unix  3      [ ]         STREAM     CONNECTED     17462    /run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     12054
unix  2      [ ]         DGRAM      CONNECTED     12832
unix  3      [ ]         SEQPACKET  CONNECTED     42172
unix  3      [ ]         STREAM     CONNECTED     17168    /run/user/1000/wayland-1
unix  3      [ ]         STREAM     CONNECTED     6063
unix  3      [ ]         STREAM     CONNECTED     6814
unix  3      [ ]         STREAM     CONNECTED     55470
unix  3      [ ]         SEQPACKET  CONNECTED     43644
unix  3      [ ]         STREAM     CONNECTED     20324
unix  3      [ ]         STREAM     CONNECTED     23848    /run/user/1000/bus
unix  3      [ ]         STREAM     CONNECTED     58618
unix  2      [ ]         DGRAM                    24763
unix  3      [ ]         STREAM     CONNECTED     26524
unix  3      [ ]         SEQPACKET  CONNECTED     42507
unix  3      [ ]         STREAM     CONNECTED     13197
unix  3      [ ]         STREAM     CONNECTED     1642     /run/systemd/journal/stdout
unix  3      [ ]         STREAM     CONNECTED     20697
unix  3      [ ]         STREAM     CONNECTED     8745
unix  3      [ ]         STREAM     CONNECTED     10874
```

Syntax:

netstat [-a] [-e] [-n] [-o] [-p Protocol] [-r] [-s] [Interval]

Parameters:

Used without parameters	Displays active TCP connections
-a	Displays all active TCP connections and the TCP and UDP ports on which the computer is listening
-e	Displays Ethernet statistics, such as the number of bytes and packets sent and received. This parameter can be combined with -s.
-n	Displays Ethernet statistics, such as the number of bytes and packets sent and received. This parameter can be combined with -s.
-o	Displays active TCP connections and includes the process ID (PID) for each connection. You can find the application based on the PID on the Processes tab in Windows Task Manager. This parameter can be combined with -a, -n, and -p.
-p	Shows connections for the protocol specified by Protocol. In this case, the Protocol can be tcp, udp, tcpv6, or udpv6. If this parameter is used with -s to display statistics by protocol, Protocol can be tcp, udp, icmp, ip, tcpv6, udpv6, icmpv6, or ipv6
-s	Displays statistics by protocol. By default, statistics are shown for the TCP, UDP, ICMP, and IP protocols. If the IPv6 protocol for Windows XP is installed, statistics are shown for the TCP over IPv6, UDP over IPv6, ICMPv6, and IPv6 protocols. The -p parameter can be used to specify a set of protocols.
-r	Displays the contents of the IP routing table. This is equivalent to the route print command.
interval	Redisplays the selected information every Interval seconds. Press CTRL+C to stop the redisplay. If this parameter is omitted, netstat prints the selected information only once
?/	- Displays help at the command prompt.

3. Ifconfig

In Windows, ipconfig is a console application designed to run from the Windows command prompt. This utility allows you to get the IP address information of a Windows computer. It also allows some control over active TCP/IP connections. Ipconfig replaced the older winipcfg utility.

Using ipconfig

From the command prompt, type ipconfig to run the utility with default options. The output of the default command contains the IP address, network mask, and gateway for all physical and virtual network adapter

Syntax

ipconfig [/all] [/renew [Adapter]] [/release [Adapter]] [/flushdns] [/displaydns] [/registerdns] [/showclassid Adapter] [/setclassid Adapter [ClassID]]

Parameters

Used without parameters	displays the IP address, subnet mask, and default gateway for all adapters
/all	Displays the full TCP/IP configuration for all adapters. Without this parameter, ipconfig displays only the IP address, subnet mask, and default gateway values for each adapter. Adapters can represent physical interfaces, such as installed network adapters, or logical interfaces, such as dial-up connections.
/renew [Adapter]	Renews DHCP configuration for all adapters (if an adapter is not specified) or for a specific adapter if the Adapter parameter is included. This parameter is available only on computers with adapters that are configured to obtain an IP address automatically. To specify an adapter name, type the adapter name that appears when you use ipconfig without parameters.
/renew [Adapter]	Sends a DHCPRELEASE message to the DHCP server to release the current DHCP configuration and discard the IP address configuration for either all adapters (if an adapter is not specified) or for a specific adapter if the Adapter parameter is included. This parameter disables TCP/IP for adapters configured to obtain an IP address automatically. To specify an adapter name, type the adapter name that appears when you use ipconfig without parameters
/flushdns	Flushes and resets the contents of the DNS client resolver cache. During DNS troubleshooting, you can use this procedure to discard negative cache entries from the cache, as well as any other entries that have been added dynamically.
/displaydns	Displays the contents of the DNS client resolver cache, which includes both entries preloaded from the local Hosts file and any recently obtained resource records for name queries resolved by the computer. The DNS Client service uses this information to resolve frequently queried names quickly, before querying its configured DNS servers.

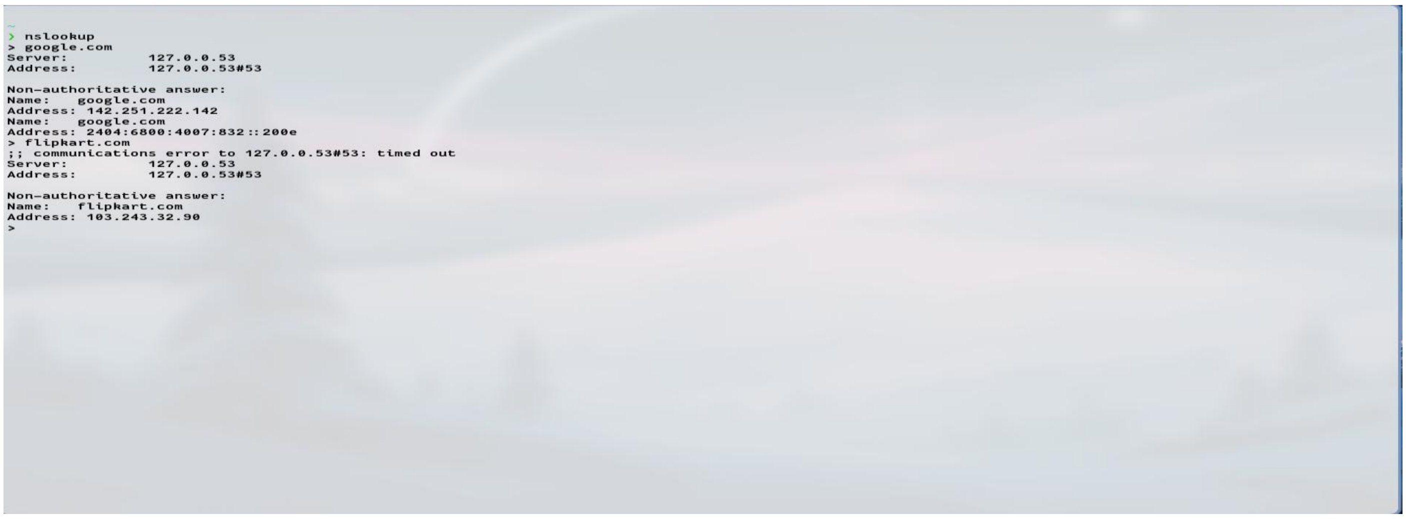
/registerdns	Initiates manual dynamic registration for the DNS names and IP addresses that are configured at a computer. You can use this parameter to troubleshoot a failed DNS name registration or resolve a dynamic update problem between a client and the DNS server without rebooting the client computer. The DNS settings in the advanced properties of the TCP/IP protocol determine which names are registered in DNS
/showclassid	Adapter Displays the DHCP class ID for a specified adapter. To see the DHCP class ID for all adapters, use the asterisk (*) wildcard character in place of Adapter. This parameter is available only on computers with adapters that are configured to obtain an IP address automatically
/setclassid	Adapter [ClassID] Configures the DHCP class ID for a specified adapter. To set the DHCP class ID for all adapters, use the asterisk (*) wildcard character in place of Adapter. This parameter is available only on computers with adapters that are configured to obtain an IP address automatically. If a DHCP class ID is not specified, the current class ID is removed

EXAMPLES:

Ipconfig	To display the basic TCP/IP configuration for all adapters
ipconfig /all	To display the full TCP/IP configuration for all adapters
ipconfig /renew "Local Area Connection"	To renew a DHCP-assigned IP address configuration for only the Local Area Connection adapter
ipconfig /flushdns	To flush the DNS resolver cache when troubleshooting DNS name resolution problems
ipconfig /showclassid Local	To display the DHCP class ID for all adapters with names that start with Local
ipconfig /setclassid "Local Area Connection" TEST	To set the DHCP class ID for the Local Area Connection adapter to TEST

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



```
~
> nslookup
> google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 142.251.222.142
Name:   google.com
Address: 2404:6800:4007:832::200e
> flipkart.com
;; communications error to 127.0.0.53#53: timed out
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   flipkart.com
Address: 103.243.32.90
>
```

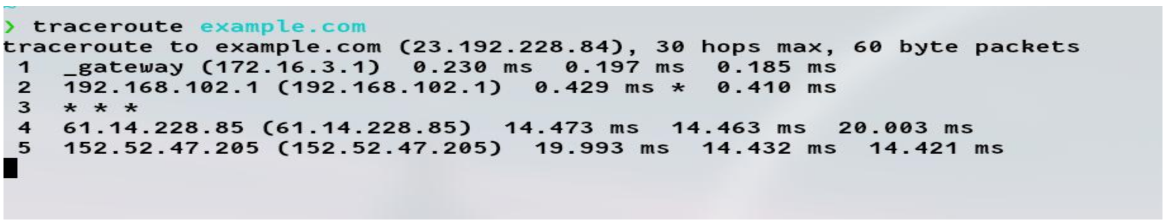
5. traceroute

Traceroute is a network diagnostic tool used to track the pathway taken by a packet on an IP network from source to destination. Traceroute also records the time taken for each hop the packet makes during its route to the destination.

Traceroute uses Internet Control Message Protocol (ICMP) echo packets with variable time to live (TTL) values. The response time of each hop is calculated. To guarantee accuracy, each hop is queried multiple times (usually three times) to better measure the response of that particular hop.

traceart www.google.com

With the tracert command shown above, we're asking tracert to show us the path from the local computer all the way to the network device with the hostname
www.google.com.
Tracing route to www.l.google.com [209.85.225.104] over a maximum of 30 hops:
1 <1 ms <1 ms <1 ms 10.1.0.1
2 35 ms 19 ms 29 ms 98.245.140.1
3 11 ms 27 ms 9 ms te-0-3.dnv.comcast.net [68.85.088.201]
... 13 81 ms 76 ms 75 ms 209.85.241.37
14 84 ms 91 ms 87 ms 209.85.248.102
15 76 ms 112 ms 76 ms iy-f104.1e100.net [209.85.225.104]
Trace complete.
tracert -j 10.12.0.1 10.29.3.1 10.1.44.1 www.google.com



```
~
> traceroute example.com
traceroute to example.com (23.192.228.84), 30 hops max, 60 byte packets
 1  _gateway (172.16.3.1)  0.230 ms  0.197 ms  0.185 ms
 2  192.168.102.1 (192.168.102.1)  0.429 ms *  0.410 ms
 3  * * *
 4  61.14.228.85 (61.14.228.85)  14.473 ms  14.463 ms  20.003 ms
 5  152.52.47.205 (152.52.47.205)  19.993 ms  14.432 ms  14.421 ms
```

PROGRAM &EXECUTION	
CLASS PERFORMNCE	
VIVA	
TOTAL	

RESULT:

Hence, we learned to use commands like tcpdump, netstat, ifconfig, nslookup and traceroute

Ex No: 3	**Write a HTTP web client program to download a web page using TCP sockets**
Date:	

AIM:

To write a java program for socket for HTTP for web page upload and download.

ALGORITHM:

Client:

Step-1: Start.

Step-2: Create socket and establish the connection with the server.

Step-3: Read the image to be uploaded from the disk

Step-4: Send the image read to the server

Step-5: Terminate the connection

Step-6: Stop

Server:

Step-1: Start

Step-2: Create socket, bind IP address and port number with the created socket and make server a listening server.

Step-3: Accept the connection request from the client

Step-4: Receive the image sent by the client.

Step-5: Display the image.

Step-6: Close the connection.

Step-7: Stop.

PROGRAM:

Client:

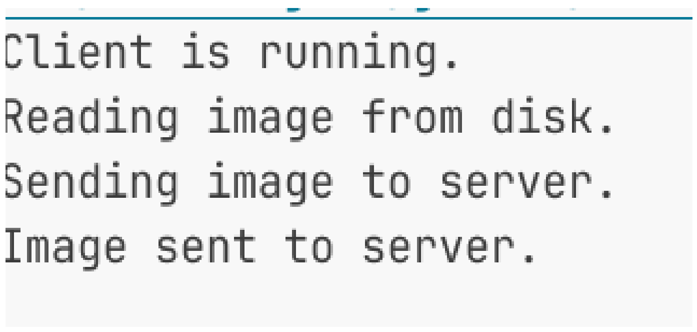
```
import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.OutputStream;
import java.net.Socket;
```

```
import javax.imageio.ImageIO;

public class Client {
    public static void main(String args[]) throws Exception {
        Socket soc;
        BufferedImage img = null;
        soc = new
            Socket("localhost", 4000);
        System.out.println("Client is running.");
        try {
            System.out.println("Reading image from disk. ");
            img = ImageIO.read(new File("digital_image_processing.jpg"));
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            ImageIO.write(img, "jpg", baos);
            baos.flush();
            byte[] bytes = baos.toByteArray();
            baos.close();
            System.out.println("Sending image to server.");
            OutputStream out = soc.getOutputStream();
            DataOutputStream dos = new DataOutputStream(out);
            dos.writeInt(bytes.length);
            dos.write(bytes, 0, bytes.length);
            System.out.println("Image sent to server. ");
            dos.close();
            out.close();
        } catch (Exception e) {
            System.out.println("Exception: " + e.getMessage());
            soc.close();
        }
        soc.close();
    }
}
```

Server:

```
import java.net.*;
import java.io.*;
import java.awt.image.*;
import javax.imageio.*;
import javax.swing.*;
class Server
{
   public static void main(String args[]) throws Exception
   {
      ServerSocket server=null;
      Socket socket;
      server=new ServerSocket(4000);
      System.out.println("Server Waiting for image");
      socket=server.accept(); System.out.println("Client connected.");
      InputStream in = socket.getInputStream();
      DataInputStream dis = new DataInputStream(in);
      int len = dis.readInt();
      System.out.println("Image Size: " + len/1024 + "KB"); byte[] data = new byte[len];
      dis.readFully(data);
      dis.close();
      in.close();
      InputStream ian = new ByteArrayInputStream(data);
      BufferedImage bImage = ImageIO.read(ian);
      JFrame f = new JFrame("Server");
      ImageIcon icon = new ImageIcon(bImage);
      JLabel l = new JLabel();
      l.setIcon(icon);
      f.add(l);
      f.pack();
      f.setVisible(true);
   }
}
```

OUTPUT:



```
Client is running.
Reading image from disk.
Sending image to server.
Image sent to server.
```

PROGRAM &EXECUTION	
CLASS PERFORMNCE	
VIVA	
TOTAL	

RESULT:

Hence, the java program was implemented and executed successfully

Ex No: 4	**Applications using TCP sockets like Echo client and Echo server**
Date:	

AIM:

To write a java program for applications using TCP sockets like Echo client and Echo server.

ALGORITHM:

Step-1 : Start the program.

Step-2 : Get the frame size from the user.

Step-3 : To create the frame based on the user request.

Step-4 : To send frames to server from the client side.

Step-5 : If your frames reach the server it will send ACK signal to client otherwise, it will send NACK signal to the client.

Step-6 : Stop the program.

PROGRAM:

EchoServer.java

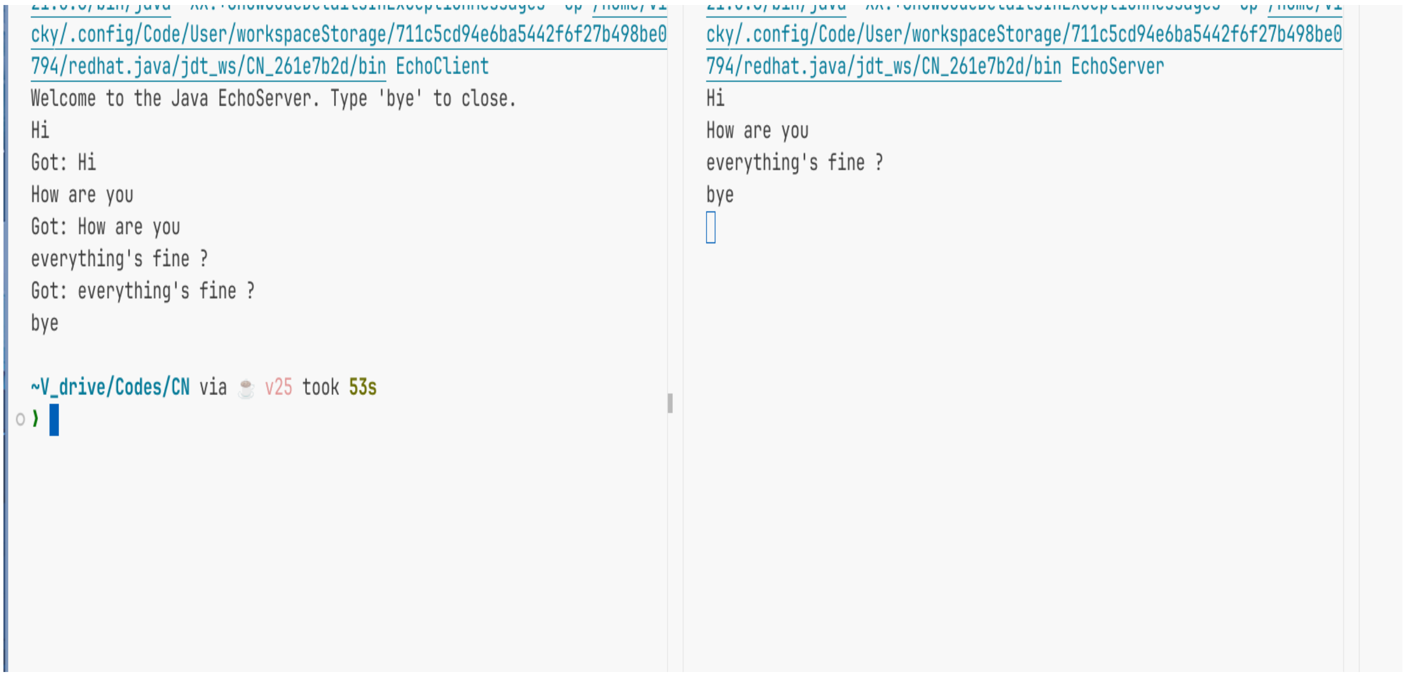
```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class EchoServer {
   private ServerSocket server;

   public EchoServer(int portnum) {
      try {
         server = new ServerSocket(portnum);
      } catch (Exception err) {
         System.out.println(err);
      }
   }
```

```
    public static void main(String[] args) {
        EchoServer s = new EchoServer(9999);
        s.serve();
    }

    public void serve() {
        try {
            while (true) {
                Socket client = server.accept();
                BufferedReader r = new BufferedReader(new
                        InputStreamReader(client.getInputStream()));
                PrintWriter w = new
                        PrintWriter(client.getOutputStream(),
                        true);
                w.println("Welcome to the Java EchoServer. Type 'bye' to close.");
                String line;
                do {
                    line = r.readLine();
                    if (line != null)
                        w.println("Got: " + line);
                    System.out.println(line);
                }
                while (!line.trim().equals("bye"));
                client.close();
            }
        } catch (Exception err) {
            System.err.println(err);
        }
    }
}
```

EchoClient.java

```
import java.io.*;
import java.net.*;
public class EchoClient
{
   public static void main(String[] args)
   {
      try
      {
         Socket s = new Socket("127.0.0.1", 9999);
         BufferedReader r = new BufferedReader(new
               InputStreamReader(s.getInputStream()));
         PrintWriter w = new PrintWriter(s.getOutputStream(), true);
         BufferedReader con = new BufferedReader(new InputStreamReader(System.in));
         String line;
         do
         {
            line = r.readLine();
            if ( line != null )
               System.out.println(line);
            line = con.readLine();
            w.println(line);
         }
         while ( !line.trim().equals("bye") );
      }
      catch (Exception err)
      {
         System.err.println(err);
      }
   }
}
```

OUTPUT:



```
cky/.config/Code/User/workspaceStorage/711c5cd94e6ba5442f6f27b498be0
794/redhat.java/jdt_ws/CN_261e7b2d/bin EchoClient
Welcome to the Java EchoServer. Type 'bye' to close.
Hi
Got: Hi
How are you
Got: How are you
everything's fine ?
Got: everything's fine ?
bye

~V_drive/Codes/CN via ☕ v25 took 53s
○ ❯
```

```
cky/.config/Code/User/workspaceStorage/711c5cd94e6ba5442f6f27b498be0
794/redhat.java/jdt_ws/CN_261e7b2d/bin EchoServer
Hi
How are you
everything's fine ?
bye
```

SIDDHARTH T 714023247088

PROGRAM &EXECUTION	
CLASS PERFORMNCE	
VIVA	
TOTAL	

RESULT:

Hence, the java program to concurrently communicate using TCP Sockets was executed successfully.

Ex No: 5	**Applications using TCP sockets like Chat.**
Date:	

AIM:

To implement a CHAT application using TCP Socket.

CONCEPT:

1. It uses TCP socket communication .We have a server as well as a client.
2. Both can be run in the same machine or different machines. If both are running in the machine, the address to be given at the client side is local host address.
3. If both are running in different machines, then in the client side we need to specify the ip address of machine in which server application is running

PROGRAM:

ChatServer.java

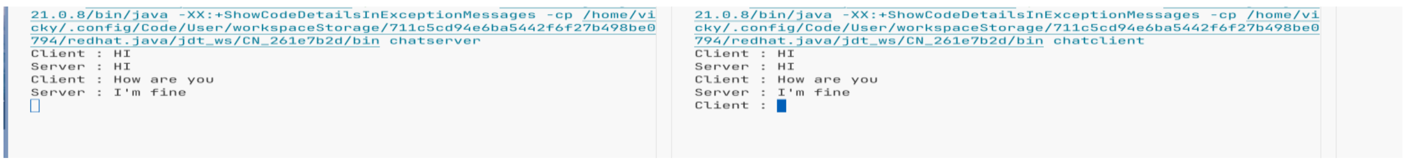
```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;

public class chatserver {
   public static void main(String args[]) throws Exception {
      ServerSocket ss = new ServerSocket(2000);
      Socket sk = ss.accept();
      BufferedReader cin = new BufferedReader(new
            InputStreamReader(sk.getInputStream()));
      PrintStream cout = 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();
    }
}
```

ChatClient.java

```
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);
```

OUTPUT:



```
21.0.8/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /home/vi
cky/.config/Code/User/workspaceStorage/711c5cd94e6ba5442f6f27b498be0
794/redhat.java/jdt_ws/CN_261e7b2d/bin chatserver
Client : HI
Server : HI
Client : How are you
Server : I'm fine
```

```
21.0.8/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /home/vi
cky/.config/Code/User/workspaceStorage/711c5cd94e6ba5442f6f27b498be0
794/redhat.java/jdt_ws/CN_261e7b2d/bin chatclient
Client : HI
Server : HI
Client : How are you
Server : I'm fine
Client :
        s=sin.readLine();
            System.out.print("Server : "+s+"\n");
             if ( s.equalsIgnoreCase("BYE") )
                 break;
        }
        sk.close();
        sin.close();
        sout.close();
        stdin.close();
    }
}
```

PROGRAM &EXECUTION	
CLASS PERFORMNCE	
VIVA	
TOTAL	

RESULT:

Hence, the java program to concurrently communicate using TCP Sockets was executed successfully.

Ex No: 6	**Applications using TCP sockets like File Transfer.**
Date:	

AIM:

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

ALGORITHM:

Server:
Step-1: Import java packages and create class file server.

Step-2: Create a new server socket and bind it to the port.

Step-3: Accept the client connection

Step-4: Get the file name and stored into the BufferedReader.

Step-5: Create a new object class file and realine.

Step-6: If file is exists then FileReader read the content until EOF is reached.

Step-7: Stop the program.

Client
Step-1: Import java packages and create class file server.

Step-2: Create a new server socket and bind it to the port.

Step-3: Now connection is established.

Step-4: The object of a BufferReader class is used for storing data content which has been retrieve from socket object.

Step-5: The connection is closed.

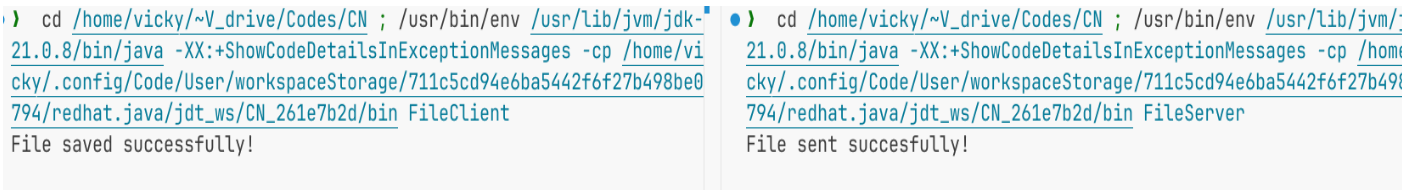
Step-6: Stop the program.

PROGRAM:

File Server

```
import java.io.BufferedInputStream;
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 FileServer {
```

OUTPUT:



```
❯ cd /home/vicky/~V_drive/Codes/CN ; /usr/bin/env /usr/lib/jvm/jdk-
21.0.8/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /home/vi
cky/.config/Code/User/workspaceStorage/711c5cd94e6ba5442f6f27b498be0
794/redhat.java/jdt_ws/CN_261e7b2d/bin FileClient
File saved successfully!
```

```
● ❯ cd /home/vicky/~V_drive/Codes/CN ; /usr/bin/env /usr/lib/jvm/j
21.0.8/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /home
cky/.config/Code/User/workspaceStorage/711c5cd94e6ba5442f6f27b498
794/redhat.java/jdt_ws/CN_261e7b2d/bin FileServer
File sent succesfully!
```

```
public static void main(String[] args) throws Exception {
      ServerSocket ssock = new ServerSocket(5000);
      Socket socket = ssock.accept();
      InetAddress IA = InetAddress.getByName("localhost");
      File file = new File("/home/vicky/~V_drive/Codes/CN/e:\\Bookmarks1.html");
      FileInputStream fis = new FileInputStream(file);
      BufferedInputStream bis = new BufferedInputStream(fis);
      OutputStream os = socket.getOutputStream();
      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();
      socket.close();
      ssock.close();
      System.out.println("File sent succesfully!");
   }
}
```

PROGRAM &EXECUTION	
CLASS PERFORMNCE	
VIVA	
TOTAL	

RESULT:

Hence, the java program to concurrently communicate using TCP Sockets was executed successfully

Ex No: 7	**Simulation of DNS using UDP sockets**
Date:	

AIM:

To Simulation of DNS using UDP sockets.

ALGORITHM:

Step-1 : Start the program.

Step-2 : Get the frame size from the user.

Step-3 : To create the frame based on the user request.

Step-4 : To send frames to server from the client side.

Step-5 : If your frames reach the server it will send ACK signal to client otherwise, it will send NACK signal to the client.

Step-6 : Stop the program

PROGRAM:

dnsserver.java

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

public class dnsserver {
   private static int indexOf(String[] array, String str) {
      str = str.trim();
      for (int i = 0; i < array.length; i++) {
         if (array[i].equals(str))
            return i;
      }
      return -1;
   }
```

```
public static void main(String arg[]) throws IOException {
    String[] hosts = { "zoho.com", "gmail.com", "google.com", "facebook.com" };
    String[] ip = { "172.28.251.59", "172.217.11.5", "172.217.11.14",
        "31.13.71.36" };
    System.out.println("Press Ctrl + C to Quit");
    while (true) {
      DatagramSocket serversocket = new DatagramSocket(1362);
      byte[] senddata = new byte[1021];
      byte[] receivedata = new byte[1021];
      DatagramPacket recvpack = new DatagramPacket(receivedata,
          receivedata.length);
      serversocket.receive(recvpack);
      String sen = new String(recvpack.getData());
      InetAddress ipaddress = recvpack.getAddress();
      int port = recvpack.getPort();
      String capsent;
      System.out.println("Request for host " + sen);
      if (indexOf(hosts, sen) != -1)
        capsent = ip[indexOf(hosts, sen)];
      else
        capsent = "Host Not Found";
      senddata = capsent.getBytes();
      DatagramPacket pack = new DatagramPacket(senddata,
          senddata.length, ipaddress, port);
      serversocket.send(pack);
      serversocket.close();
    }
  }
}
```

dnsclient.java

```
import java.io.*;

import java.net.*;


public class dnsclient {

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

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

        DatagramSocket clientsocket = new DatagramSocket();

        InetAddress ipaddress;

        if (args.length == 0)

            ipaddress = InetAddress.getLocalHost();

        else

            ipaddress = InetAddress.getByName(args[0]);

        byte[] senddata = new byte[1024];

        byte[] receivedata = new byte[1024];

        int portaddr = 1362;

        System.out.print("Enter the hostname : ");

        String sentence = br.readLine();

        senddata = sentence.getBytes();

        DatagramPacket pack = new DatagramPacket(senddata, senddata.length,

                ipaddress, portaddr);

        clientsocket.send(pack);

        DatagramPacket recvpack = new DatagramPacket(receivedata, receivedata.length);

        clientsocket.receive(recvpack);

        String modified = new String(recvpack.getData());

        System.out.println("IP Address: " + modified);

        clientsocket.close();

    }

}
```

OUTPUT:

Server:

```
Press Ctrl + C to Quit

Request for host google.com

Request for host flipkart.com
```

Client

```
Enter the hostname : google.com

IP Address: 172.217.11.14

Enter the hostname : flipkart.com

IP Address: Host Not Found
```

PROGRAM &EXECUTION	
CLASS PERFORMNCE	
VIVA	
TOTAL	

RESULT:

Hence, the DNS application program was executed successfully.

Ex No: 8	**Program for Address Resolution Protocol (ARP) using TCP**
Date:	

AIM:

To write a java program for simulating ARP protocols using TCP.

ALGORITHM:

Client:

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

Server:

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

PROGRAM:

Client:

```
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.Socket;

class Clientarp {
   public static void main(String args[]) {
      try {

         BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
         Socket clsct = new Socket("127.0.0.1", 5604);
         DataInputStream din = new DataInputStream(clsct.getInputStream());
         DataOutputStream dout = new
               DataOutputStream(clsct.getOutputStream());
```

```
         System.out.println("Enter the Logical address(IP):");
            String str1 = in.readLine();
            dout.writeBytes(str1 + '\n');
            String str = din.readLine();
            System.out.println("The Physical Address is: " + str);
            clsct.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

server:

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

class Serverarp {
    public static void main(String args[]) {
        try {
            ServerSocket obj = new ServerSocket(5604);
            Socket obj1 = obj.accept();
            while (true) {
                DataInputStream din = new DataInputStream(obj1.getInputStream());
                DataOutputStream dout = new DataOutputStream(obj1.getOutputStream());
                String str = din.readLine();
                String ip[] = { "165.165.80.80", "165.165.79.1" };
                String mac[] = { "6A:08:AA:C2", "8A:BC:E3:FA" };
                for (int i = 0; i < ip.length; i++) {
                    if (str.equals(ip[i])) {
                        dout.writeBytes(mac[i] + '\n');
                        break;
                    }
                }
                obj.close();
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

OUTPUT:

```
Enter the Logical address(IP):
165.165.80.80
The Physical Address is: 6A:08:AA:C2
```

PROGRAM &EXECUTION	
CLASS PERFORMNCE	
VIVA	
TOTAL	

RESULT:

Hence, the ARP protocol using TCP Sockets program was executed

Ex No: 9	**Program for Reverse Address Resolution Protocol (RARP) using UDP**
Date:	

AIM:

To write a java program for simulating RARP protocols using UDP.

ALGORITHM:

Client:

Step-1 : Start the program.

Step-2 : Using socket connection is established between client and server.

Step-3 : Get the IP address to be converted into MAC address.

Step-4 : Send this IP address to server.

Step-5 : Server returns the MAC address to client.

Server:

Step-1 : Start the program.

Step-2 : Accept the socket which is created by the client.

Step-3 : Server maintains the table in which IP and corresponding MAC address are stored.

Step-4 : Read the IP address which is send by the client.

Step-5 : Map the IP address with its MAC address and return the MAC address to client.

PROGRAM:

Client:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
class Clientrarp {
   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 Physical address (MAC):");
        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("The Logical Address is(IP): " + s.trim());
        client.close();
      } catch (Exception e) {
        System.out.println(e);
      }
    }
}
```

Server:

```
import java.io.*;
import java.net.*;
import java.util.*;
class Serverrarp {
    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);
```

OUTPUT:

```
Enter the Physical address (MAC):
6A:08:AA:C2
The Logical Address is(IP): 165.165.80.80
```

```
            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" };
                String mac[] = { "6A:08:AA:C2", "8A:BC:E3:FA" };
                for (int i = 0; i < ip.length; i++) {
                    if (s.equals(mac[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);
        }
    }
}
```

PROGRAM &EXECUTION	
CLASS PERFORMNCE	
VIVA	
TOTAL	

RESULT:

Hence, the RARP protocol using UDP was executed.

Ex No: 10	**Study of Network simulator (NS) and Simulation of Congestion Control Algorithms using NS.**
Date:	

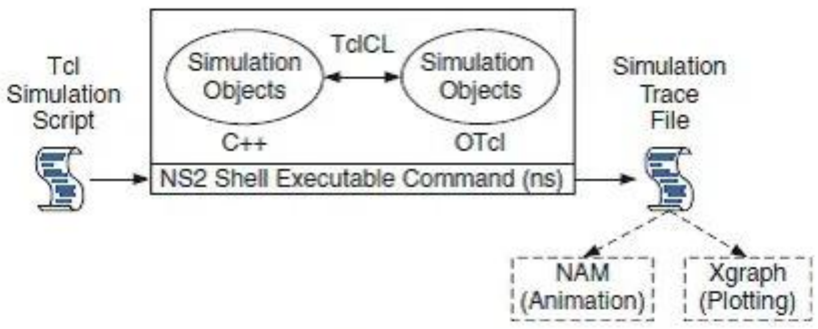
AIM:

To study the network simulator (NS) and simulation of congestion control algorithms using NS.

THEORY:

Network Simulator (Version 2), widely known as NS2, is simply an event driven simulationtool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way ofspecifying such network protocols and simulating their corresponding behaviors. Due to its flexibility and modular nature, NS2 has gained constant popularity in the networking researchcommunity since its birth in 1989.

Basic Architecture of NS2:



Basic architecture of NS.

The above figure shows the basic architecture of NS2. NS2 provides users with an executable command ns which takes on input argument, the name of a Tcl simulation scripting file. Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command ns. In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation. NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objectsas well as scheduling discrete events (i.e., a frontend).
The C++ and the OTcl are linked together using TclCL. Mapped to a C++ object, variables in the OTcl domains are sometimes referred to as handles. Conceptually, a handle (e.g., n as a Node handle) is just a string (e.g.,_o10) in the OTcl domain, and does not contain any functionality. instead, the functionality (e.g., receiving a packet) is defined in the mapped C++ object (e.g., of class Connector). In the OTcl domain, a handle acts as a frontend which interacts with users and other OTcl objects. It may defines its own procedures and variables to facilitate the interaction. Note that the member procedures andvariables in
Expt.No: 9 Study of Network simulator (NS) and Simulation of
Date: Congestion Control Algorithms using NS.

the OTcl domain are called instance procedures (instprocs) and instance variables (instvars), respectively

Tcl scripting:

- Tcl is a general purpose scripting language. [Interpreter]
- Tcl runs on most of the platforms such as Unix, Windows, and Mac.
- The strength of Tcl is its simplicity.
- It is not necessary to declare a data type for variable prior to the usage

Basics of TCL
Syntax: command arg1 arg2 arg3

Hello World!
puts stdout{Hello, World!}Hello,
World!

Variables

Command Substitution
set a 5 set len [string length foobar]
set b $a set len [expr [string length foobar] + 9]

Simple Arithmetic
expr 7.2 / 4

Procedures
proc Diag {a b} {
set c [expr sqrt($a * $a + $b * $b)]return $c }
puts ―Diagonal of a 3, 4 right triangle is [Diag 3 4]‖Output:
Diagonal of a 3, 4 right triangle is 5.0

Loops:

while{$i < $n} {
 	
....
 	
}
	for {set i 0} {$i < $n} {incr i} {
	
	....
	
	}

NS Simulator Preliminaries.
1. Initialization and termiantion aspects of the ns simulator.
2. Definition of network nodes, links, queues and topology.
3. Definition of agents and of applications.
4. The nam visualization tool.

5. Tracing and random variables.

Initialization and Termination of TCL Script in NS-2

An ns simulation starts with the command

set ns [new simulator]

Which is thus the first line in the tcl script? This line declares a new variable as using the setcommand, you can call this variable as you wish, In general people declares it as ns becauseit is an instance of the Simulator class, so an object the code[new Simulator] is indeed the installation of the class Simulator using the reserved word new.

In order to have output files with data on the simulation (trace files) or files used forvisualization (nam files), we need to create the files using —"open" command:

```
#Open the Trace file
        set tracefile1 [open out.tr w]
        $ns trace-all $tracefile1#Open the
NAM trace file
        set namfile [open out.nam w]
        $ns namtrace-all $namfile
```

The above creates a dta trace file called —out.tr‖ and a nam visualization trace file called —out.nam‖.Within the tcl script,these files are not called explicitly by their names, but instead by pointers that are declared above and called —tracefile1 and —nam file respectively. Remark that they begins with a # symbol.The second line open the file —out.tr‖to be used for writing, declared with the letter —w‖.The third line uses a simulator method called trace-all that have as parameter the name of the file where the traces will go.

The last line tells the simulator to record all simulation traces in NAM input format. It also gives the file name that the trace will be written to later by the command $ns flush-trace.In our case, this will be the file pointed at by the pointer —$namfile ,i.e the file —out.tr‖.
The termination of the program is done using a —finish‖ procedure

```
#Define a 'finish' procedure
        Proc finish { } {
        global ns tracefile1 namfile
```

$ns flush-trace Close

$tracefile1Close $namfile

Exec nam out.nam &Exit 0

The word proc declares a procedure in this case called finish and without arguments. The word global is used to tell that we are using variables declared outside the procedure. The simulator method ―flush-trace" will dump the traces on the respective files. The tcl command
―close" closes the trace files defined before and exec executes the nam program forvisualization.
The command exit will ends the application and return the number 0 as status to the system.
Zero
is the default for a clean exit. Other values can be used to say that is a exit because somethingfails.
At the end of ns program we should call the procedure ―finish‖ and specify at what timethe termination should occur. For example,

$ns at 125.0 "finish"

will be used to call **―finish‖** at time 125sec.Indeed,the at method of the simulator allows usto schedule events explicitly.
The simulation can then begin using the command

$ns runDefinition

of a network of links and nodes
The way to define a node is

set n0 [$ns node]

The node is created which is printed by the variable n0. When we shall refer to that node inthe script we shall thus write $n0.
Once we define several nodes, we can define the links that connect them. An example ofa definition of a link is:

$ns duplex-link $n0 $n2 10Mb 10ms DropTail

Which means that $n0 and $n2 are connected using a bi-directional link that has 10msof propagation delay and a capacity of 10Mb per sec for each direction. To define a directional link instead of a bi-directional one, we should replace "duplexlink" by "simplex- link".
In NS, an output queue of a node is implemented as a part of each link whose input isthat node. The definition of the link then includes the way to handle overflow at that queue.In our case, if the buffer capacity of the output queue is exceeded then the last packet to arrive is dropped. Many alternative options exist, such as the RED (Random Early Discard)mechanism, the FQ (Fair Queuing), the DRR (Deficit Round Robin), the stochastic Fair Queuing (SFQ) and the CBQ (which including a priority and

a round-robin scheduler).

In ns, an output queue of a node is implemented as a part of each link whose input is that node. We should also define the buffer capacity of the queue related to each link. An examplewould be

```
#set Queue Size of link (n0-n2) to 20
$ns queue-limit $n0 $n2 20
```

Agents and Applications

We need to define routing (sources, dsetinations) the agents (protocols) the applications that use them

FTP over TCP

TCP is a dynamic reliable congestion control protocol. It uses Acknowledgements created by the destination to know whether packets are well received. There are number variants of the TCP protocol, such as Tahoe, Reno, NewReno, Vegas. The type of agents appears in the first line:

set tcp [new Agent/TCP]

The command **$ns attach-agent $n0 $tcp** defines the source node of the tcp connection.

The command

set sink [new Agent/TCPSink]

Defines the behavior of the destination node of TCP and assigns to it a pointer called sink

```
#Setup a UDP connection
        set udp [new Agent/UDP]
        $ns attach-agent $n1 $udpset null
        [new Agent/Null]
        $ns attach-agent $n5 $null
        $ns connect $udp $null
        $udp set fid_2
#setup a CBR over UDP connection
        set cbr [new Application/Traffic/CBR]
        $cbr attach-agent $udp
        $cbr set packetsize_100
        $cbr set rate_0.01Mb
        $cbr set random_false
```

Above shows the definition of a CBR application using a UDP agent. The command **$ns attach-agent $n4 $sink** defines the destination node. The command **$ns connect $tcp $sink** Finally makes the TCP connection between the source and destination nodes.

TCP has many parameters with initial fixed defaults values that can be changed if mentioned explicitly. For example, the default TCP packet size has a size of 1000bytes. This can be changed to another value, say 552 bytes, using the command **$tcp set fid_1** that assigns to the TCP connection a flow identification of "1". We shall later give the flow identification of "2" to the UDP connection.

CBR over UDP

A UDP source and destination is defined in a similar way as in the case of TCP. Instead of defining the rate in the command **$cbr set rate_0.01 Mb,** one can define the time interval transmission of packets using the command

$cbr set interval_0.005

The packet size can be set to some value using

$cbr set packetSize_<packet size>

Scheduling Events

NS is a discrete event based simulation, The tcp script defines when event should occur. The initializing command **set ns [new Simulator]** creates an evenr scheduler, and events are then scheduled using the format:

$ns at<time> <event>

The scheduler is started when running ns that is through the following command

$ns at 0.1 "$cbr start"

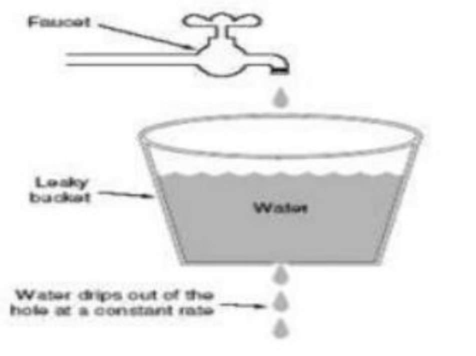
$ns at 0.1 "$ftp stop"

$ns at 124.0 "$ftp stop"

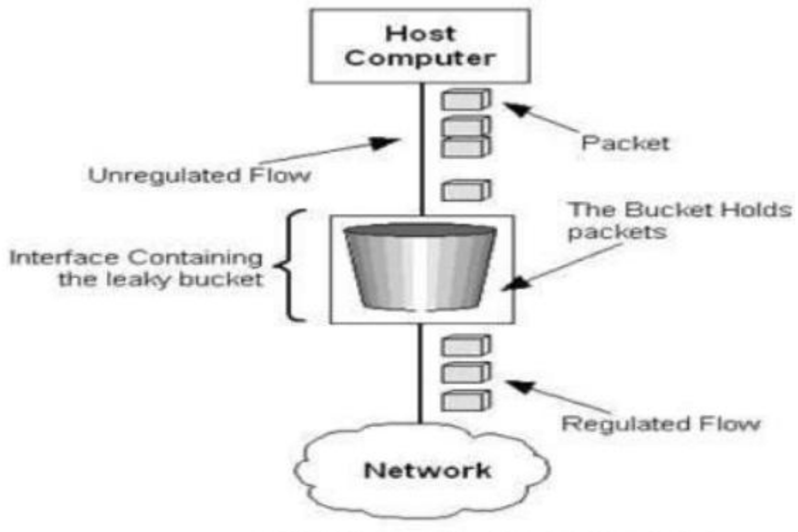
$ns at 124.5 "$cbr stop"

The main concept of the leaky bucket algorithm is that the output data flow remains constant despite teh variant input traffic, such as the water flow in a bucket with a small hole at the bottom. In case the bucket contains water(or packets) then the output flow follows a constant rate, while if the bucket is full any additional load will be lost because of spillover. In a similar way if the bucket is empty the output will be zero. From network perspective, leaky bucket consists of a finite queue (bucket) where all the incoming packets are stored in case there is space in the queue, otherwise the packets are discarded. In order to regulate the output flow, eaky bucket transmits one packet from the queue in a fixed time (e.g at every clock

tick). In the following figure we can notice the main rationale of leaky bucket algorithm, for both the two appraches (e.g leaky bucket with water (a) and with packets(b))



(a) A leaky bucket with water



(b) A leaky bucket with packets

While leaky bucket eliminates bursty traffic by regulating the incoming data flow its main drawbacks is that it drops packets if the bucket is full. Also, it doesn't take into acoount the idle process of the sender which means that if teh host doesn't transmit data for some time the bucket becomes empty without permitting the transmission of any packet.

PROGRAM:

```
import java.util.Scanner;

class Queue {

   int q[], f = 0, r = 0, size;

   void insert(int n) {

      Scanner in = new Scanner(System.in);
      q = new int[10];

      for (int i = 0; i < n; i++) {

         System.out.print("\nEnter" + i + " element: ");
```

```
            int ele = in.nextInt();
            if (r + 1 > 10) {

                System.out.println("\nQueue is full \n Lost Packet: " + ele);
                break;

            } else {

                r++;
                q[i] = ele;
            }

        }

    }

    void delete() {

        Scanner in = new Scanner(System.in);
        Thread t = new Thread();
        if (r == 0)
            System.out.print("\nQueue empty");
        else {
            for (int i = f; i < r; i++) {

                try {
                    t.sleep(1000);
                } catch (Exception e) {
                    System.out.print("\nLeaked Packet: " + q[i]);
                    f++;
                }
            }
            System.out.println();
        }
    }
```

OUTPUT:

```
Enter the packets to be sent: 12
Enter 0 element: 23
Enter 1 element: 34
Enter 2 element: 34
Enter 3 element: 334
Enter 4 element: 334
Enter 5 element: 34
Enter 6 element: 34
Enter 7 element: 34
Enter 8 element: 34
Enter 9 element: 34
Enter 10 element: 4
Queue is full Lost Packet: 4
Leaked Packet: 23
Leaked Packet: 34
Leaked Packet: 34
Leaked Packet: 334
Leaked Packet: 334
Leaked Packet: 34
Leaked Packet: 34
Leaked Packet: 34
Leaked Packet: 34
Leaked Packet: 34
```

```
    class Leaky extends Thread {

        public static void main(String ar[]) throws Exception {
            Queue q = new Queue();
            Scanner src = new Scanner(System.in);
            System.out.println("Enter the packets to be sent: ");
            int size = src.nextInt();
            q.insert(size);
            q.delete();
        }
    }
}
```

PROGRAM &EXECUTION	
CLASS PERFORMNCE	
VIVA	
TOTAL	

RESULT:

Hence, the study of network simulator (NS) and simulation of congestion control algorithms using NS was completed.

Ex No: 11	**Study of TCP/UDP Performance**
Date:	

AIM:

To study the prformance of TCP and UDP protocols.

THEORY:

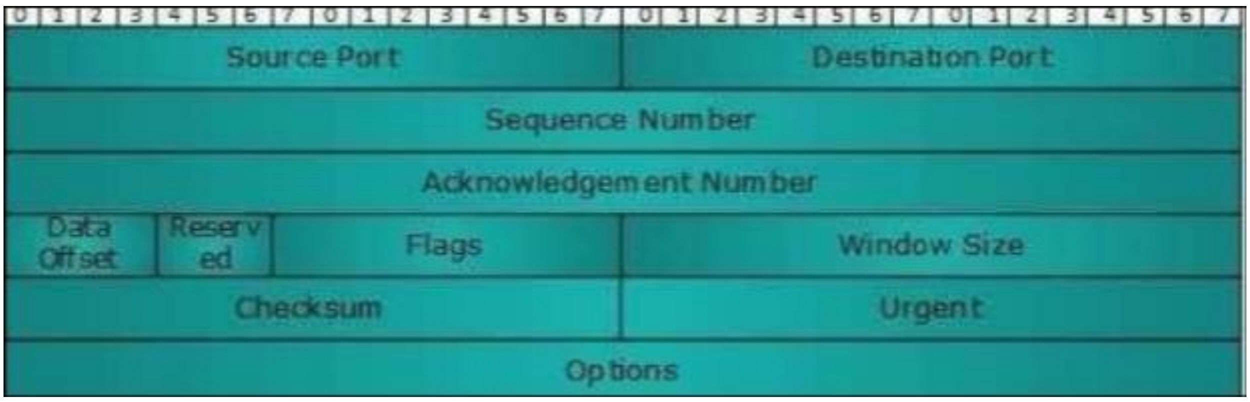
The transmission Control Protocol (TCP) is one of the most important protocols of Internet Protocols suite. It is most widely used protocol for data transmission in communication network such as internet.

Features:

- TCP is reliable protocol. That is, the receiver always sends either positive or negative acknowledgement about the data packet to the sender, so that the sender always has bright clue about whether the data packet is reached the destination or it needs to resend it.
- TCP ensures that the data reaches intended destination in the same order it was sent.
- TCP is connection oriented. TCP requires that connection between two remote points be established before sending actual data.
- TCP provides error-checking and recvery mechanism.
- TCP provides end-to-end communication.
- TCP provieds flow control and quality of service.
- TCP operated in Client/Server point-to-point mode.
- TCP provides full duplex server, i.e., it can perform roles of both receiver and sender.

Header:

The length of TCP header is minimum 20 bytes long and maximum 60 bytes.



- **Source Port (16-bits) -** It identifies source port of the application process on the sending device.
- **Destination Port (16-bits) –** It identifies destination port of the application porcess on the receiving device.
- **Sequence Number (32-bits) –** Sequence number of data bytes of a segment in a session.
- **Acknowledgement Number (32-bits) –** When ACK flag is set, this number contians the next sequnce number of the data byte expected and works as acknowledgement of the previous data received.
- **Data Offset (4-bits) –** This field implies both, the size of TCP header (32-bit words) and the offset of data in current packet in the whole TCP segment.
- **Reserved (3-bits) –** Reserved for future use and all are set zero by default.
- **Flags (1-bit each)**
  - **NS –** None sum bit is used by Explicit Congestion Notificatio signaling process.
  - **CWR –** When a host receives packet with ECE bit set, it sets Congestion Windows Reduced to acknowledge that ECE received.
  - **ECE –** It has two meanings:
    - IF SYN bit is clear to 0, then ECE means that the IP packet has its CE (Congestion experience) bit set.
    - If SYN bit is set to 1, ECE means that the device is ECT capable.
  - **URG –** It indicates that Urgent Pointer field has significant data and should be processed.
  - **ACK –** It indicates that Acknowledgement field has significance. If ACK is cleared to it indicates that packet does not contian any acknowledgement.
  - **PSH –** When set, it is a request to the receiving stattion to PUSH data (as soon as it comes) to the receiving application without buffering it.
  - **RST –** Reset flag has the following featuers:
    - It is used to refuse an incoming connection.
    - It is used to reject a segment.
    - It is used to restart a connection.
  - **SYN –** This flag is used to set up a connection between hosts.
  - **FIN –** This flag is used to release a connection and no more data is exchanged there after. Becuase packets with SYN and FIN flags have sequence numbers, they are processed in correct order.
- **Windows Size –** This field is used for flow control between two stations and indicates the amount of buffer (in bytes) the receiver has allocated for a segment, i.e., how much data is the receiver expecting.

- **Checksum –** This field contains the checksum of Header, Data and Pseudo Headers.
- **Urgent Pointer –** It points to teh urgent data byte if URG flag is set to 1.
- **Options –** It facilitates additional options which are not covered by the regular header. Option field is always described in 32-bit words. If this field contains data less than 32-bit, padding is used to cover the remaining bits to reach 32-bit boundary.

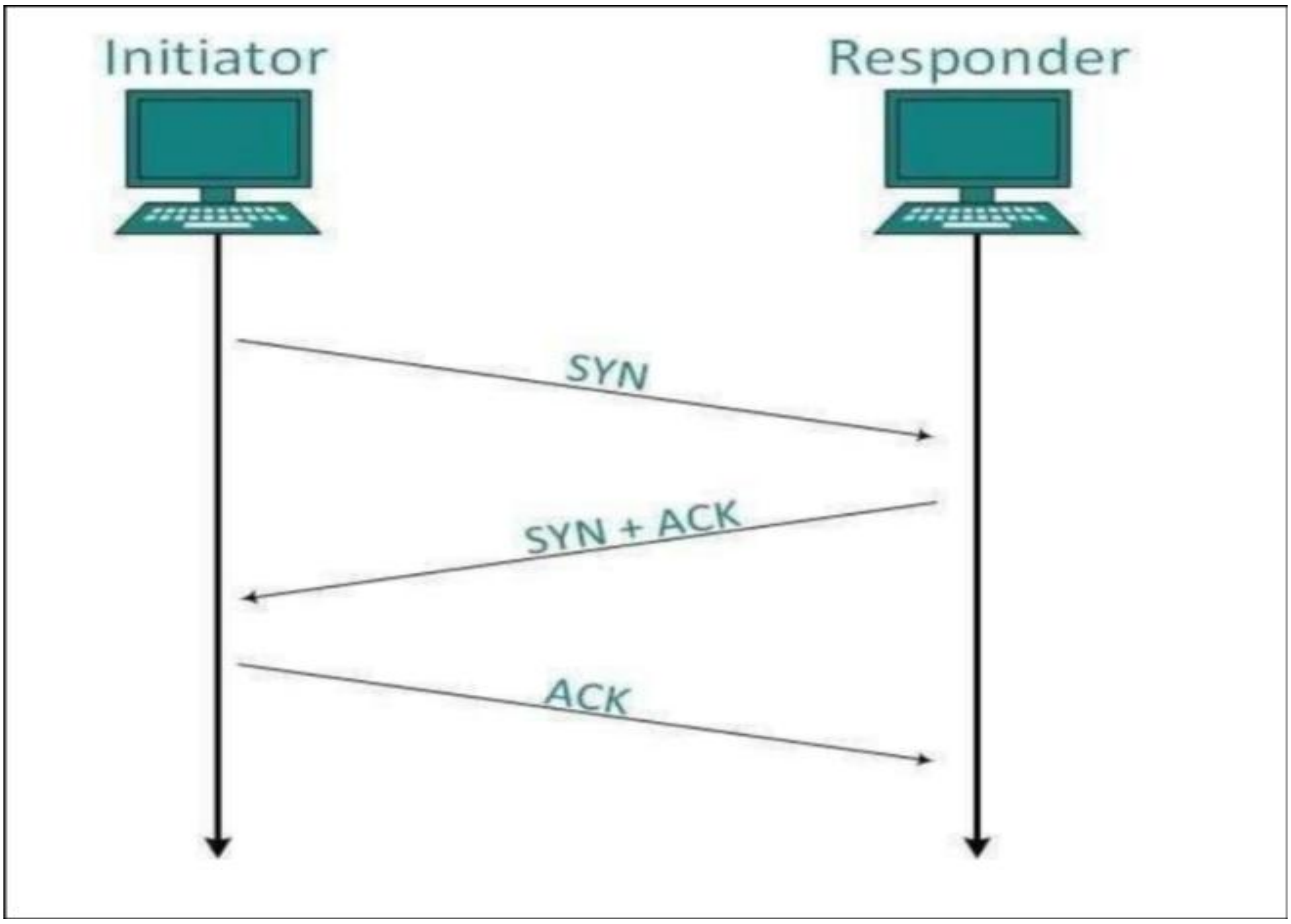
Addressing

TCP communication between two remote hosts is done by means of port numbers (TSAPs). Ports numbers can range from 0 – 65535 which are divided as

- System Ports (0 – 1023)
- User Ports (1024 – 49151)
- Private/Dynamic Ports (49152 – 65535)

Connection Management

TCP communication works in Server/Client model. The client initiates the conenction and the server wither accepts or rejects it. Three-way handshaking is used for connection management.


Initiator
Responder
SYN
SYN + ACK
ACK

Establishment

Clent initiates the connection and sends the segment with a sequence number. Server acknowledgems it back with its own Sequence number and ACK of client's segment which is one more than client's Sequence number. Client after receivign ACK of its sefment sends an acknowledgement of Server's response.

Release

Either of server and client can send TCP segment with FIN flag set to 1. When the receiving end responds it back by Acknowledging FIN, that direction of TCP communication is closed and connection is released.

Bandwidth Management

TCP uses the concept of window size to accommodate the need of Bandwidth management. Window size tells the sender at the remote end, the number of data byte segmentsthe receiver at this end can receive. TCP uses slow start phase by using window size 1 and increases the window size exponentially after each successful communication.

For example, the client uses windows size 2 and sends 2 bytes of data. When the acknowledgement of this segment received the windows size is doubled to 4 and next sent the segment sent will be 4 data bytes long. When the acknowledgement of 4-byte data segment is received, the client sets windows size to 8 and so on.

If an acknowledgement is missed, i.e. data lost in transit network or it received NACK, then the window size is reduced to half and slow start phase starts again.

Error Control &and Flow Control

TCP uses port numbers to know what application process it needs to handover the data segment. Along with that, it uses sequence numbers to synchronize itself with the remote host.All data segments are sent and received with sequence numbers. The Sender knows which lastdata segment was received by the Receiver when it gets ACK. The Receiver knows about the last segment sent by the Sender by referring to the sequence number of recently received packet.

If the sequence number of a segment recently received does not match with the sequence number the receiver was expecting, then it is discarded and NACK is sent back. If two segments arrive with the same sequence number, the TCP timestamp value is compared tomake a decision.

Multiplexing

The technique to combine two or more data streams in one session is called Multiplexing. When a TCP client initializes a connection with Server, it always refers to a well- defined port number which indicates the application process. The client itself uses a randomlygenerated port number from private port number pools.

Using TCP Multiplexing, a client can communicate with a number of different application process in a single session. For example, a client requests a web page which in turncontains different types of data (HTTP, SMTP, FTP etc.) the TCP session timeout is increasedand the session is kept open for longer time so that the three-way handshake overhead can be avoided

This enables the client system to receive multiple connection over single virtualconnection. These virtual connections are not good for Servers if the timeout is too long.

Congestion Control

When large amount of data is fed to system which is not capable of handling it, congestionoccurs. TCP controls congestion by means of Window mechanism. TCP sets a window size telling the other end how much data segment to send. TCP may use three algorithms for congestion control:

• Additive increase, Multiplicative Decrease

• Slow Start

• Timeout React

Timer Management

TCP uses different types of timer to control and management various tasks:

Keep-alive timer:

- This timer is used to check the integrity and validity of a connection.
- When keep-alive time expires, the host sends a probe to check if the connection still exists.

Retransmission timer:

- This timer maintains stateful session of data sent.
- If the acknowledgement of sent data does not receive within the Retransmission time, the data segment is sent again.

Persist timer:

- TCP session can be paused by either host by sending Window Size 0.
- To resume the session a host needs to send Window Size with some larger value.
- If this segment never reaches the other end, both ends may wait for each other for infinite time.

- When the Persist timer expires, the host re-sends its window size to let the other endknow.
- Persist Timer helps avoid deadlocks in communication.

Timed-Wait:

- After releasing a connection, either of the hosts waits for a Timed-Wait time toterminate the connection completely.
- This is in order to make sure that the other end has received the acknowledgement ofitsconnection termination request.
- Timed-out can be a maximum of 240 seconds (4 minutes).

Crash Recovery

TCP is very reliable protocol. It provides sequence number to each of byte sent in segment. Itprovides the feedback mechanism i.e. when a host receives a packet, it is bound to ACK that packet having the next sequence number expected (if it is not the last segment).

When a TCP Server crashes mid-way communication and re-starts its process it sends TPDU broadcast to all its hosts. The hosts can then send the last data segment which was never unacknowledged and carry onwards.

STUDY – 2 UDP

The User Datagram Protocol (UDP) is simplest Transport Layer communication protocol available of the TCP/IP protocol suite. It involves minimum amount of communicationmechanism. UDP is said to be an unreliable transport protocol but it uses IP services which provides best effort delivery mechanism.

In UDP, the receiver does not generate an acknowledgement of packet received and in turn, the sender does not wait for any acknowledgement of packet sent. This shortcoming makes thisprotocol unreliable as well as easier on processing.

Requirement of UDP

A question may arise, why do we need an unreliable protocol to transport the data? Wedeploy UDP where the acknowledgement packets share significant amount of bandwidth alongwith the actual data. For example, in case of video streaming, thousands of packets are forwarded towards its users.

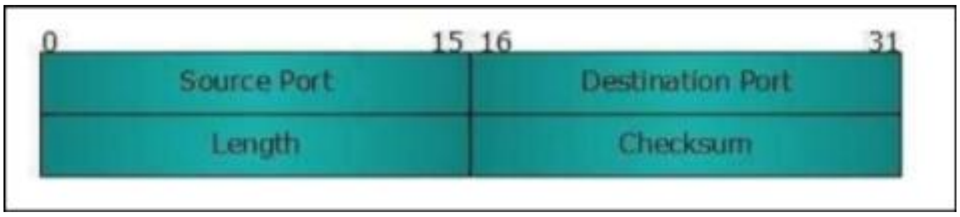
Acknowledging all the packets is troublesome and may contain huge amount of bandwidth wastage. The best delivery mechanism of underlying IP protocol ensures best efforts to deliver its packets, but even if some packets in video streaming get lost, the impact is not calamitous and can be ignored easily. Loss of few packets in video and voicetraffic sometimes goes unnoticed.

Features

- UDP is used when acknowledgement of data does not hold any significance.
- UDP is good protocol for data flowing in one direction.
- UDP is simple and suitable for query based communications.
- UDP is not connection oriented.
- UDP does not provide congestion control mechanism.
- UDP does not guarantee ordered delivery of data.
- UDP is stateless.
- UDP is suitable protocol for streaming applications such as VoIP, multimediastreaming

UDP Header

UDP header is as simple as its function.



UDP header contains four main parameters:

- **Source Port -** This 16 bits information is used to identify the source port of the packet.
- **Destination Port -** This 16 bits information, is used identify application level service on destination machine.
- **Length -** Length field specifies the entire length of UDP packet (including header). It is 16-bits field and minimum value is 8-byte, i.e. the size of UDP header itself.
- **Checksum -** This field stores the checksum value generated by the sender before sending. IPv4 has this field as optional so when checksum field does not contain any value it is made 0 and all its bits are set to zero

UDP application

Here are few applications where UDP is used to transmit data:

- Domain Name Services
- Simple Network Management Protocol
- Trivial File Transfer Protocol
- Routing Information Protocol
- Kerberos

PROGRAM &EXECUTION	
CLASS PERFORMNCE	
VIVA	
TOTAL	

RESULT:

Hence, we studied in detail about the Performance of TCP and UDP protocols.

Ex No: 12	**Simulation of Distance Vector Routing.**
Date:	

AIM:

To simulate and study the distance vector routing algorithm.

Distance Vector Routing Protocol

Routing is the process of selecting best paths in a network. In the past, the term routing was also used to mean forwarding network traffic among networks. However this latter function is muchbetter described as simply forwarding. Routing is performed for many kinds of networks, including the telephone network (circuit switching), electronic data networks (such as the Internet), and transportation networks. This article is concerned primarily with routing in electronic data networks using packet switching technology

In packet switching networks, routing directs packet forwarding (the transit of logically addressed network packets from their source toward their ultimate destination) through intermediate nodes. Intermediate nodes are typically network hardware devices such as routers, bridges, gateways, firewalls, or switches. General-purpose computers can also forward packets and perform routing, though they are not specialized hardware and may suffer from limited performance. The routing process usually directs forwarding on the basis of routing tables which maintain a record of the routes to various network destinations. Thus, constructing routing tables, which are held in the router's memory, is very important for efficient routing. Most routing algorithms use only one network path at a time. Multipath routing techniques enable the use of multiple alternative paths. In case of overlapping/equal routes, the following elements are considered in order to decide which routes get installed into the routing table (sorted by priority):

Prefix-Length: where longer subnet masks are preferred (independent of whether it is within a routing protocol or over different routing protocol)

Metric: where a lower metric/cost is preferred (only valid within one and the same routing protocol)

Administrative distance: where a lower distance is preferred (only valid between different routing protocols) Routing, in a more narrow sense of the term, is often contrasted with bridging in its assumption that network addresses are structured and that similar addresses imply proximity within the network. Structured addresses allow a single routing table entry to represent the route to a group of devices. In large

networks, structured addressing (routing, in the narrow sense) outperforms unstructured addressing (bridging). Routing has
become the dominant form of addressing on the Internet. Bridging is still widely used within localized environments.

ALGORITHM:

There are several variants of flooding algorithm. Most work roughly as follows:

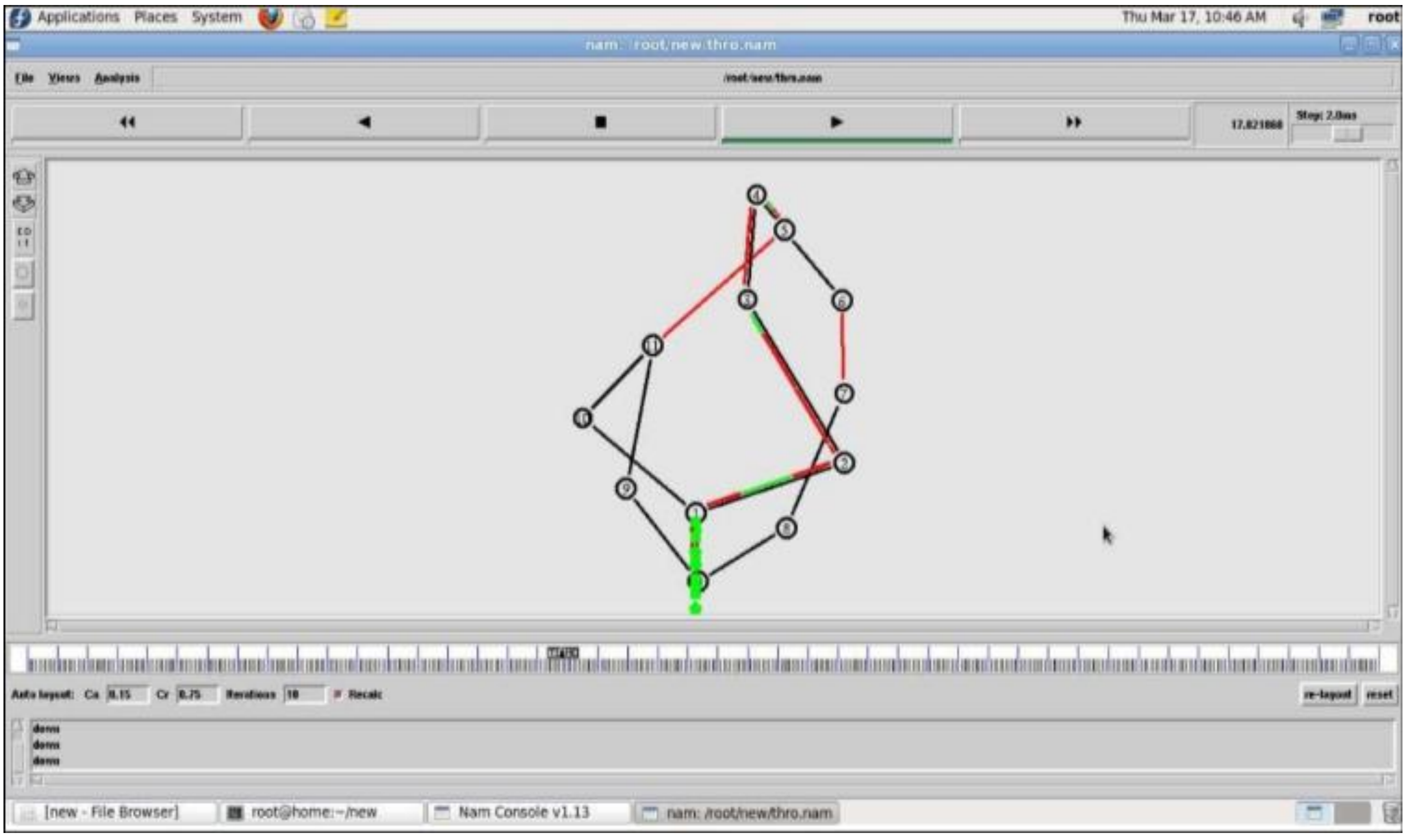
1. Each node acts as both a transmitter and a receiver.
2. Each node tries to forward every message to every one of its neighbours except the source node.

This results in every message eventually being delivered to all reachable parts of the network. Algorithms may need to be more complex than this, since, in some case, precautions have to be taken to avoid wasted duplicate deliveries and infinite loops, and to allow messages to eventually expire from the system. A duplicate deliveries and infinite loops, and to allow messages to eventually expire from the system. A variant of flooding called selective flooding partially addresses these issues by only sending packets to routers in the same direction. In selective flooding partially addresses these issues by only sending packets to routers in the same direction. In selective flooding the routers don’t send every incoming packet on every line but only on those lines which are going apporximately in the right direction.

PROGRAM:

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
set tr [open out.tr w]
$ns trace-all $tr
proc finish {} {
global nf ns tr
$ns flush-trace
close $tr
exit 0
exec nam out.nam &
}
set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns node]
$ns duplex-link $n0 $n1 10Mb 10ms DropTail
```

OUTPUT:


Applications Places System
Thu Mar 17, 10:46 AM
root
nam: /root/new/thro.nam
File Views Analysis
/root/new/thro.nam
17.821868
Step: 2.0ms
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout reset
down
down
down
[new - File Browser]
root@home:~/new
Nam Console v1.13
nam: /root/new/thro.nam

```
$ns duplex-link $n1 $n3 10Mb 10ms DropTail
$ns duplex-link $n2 $n1 10Mb 10ms DropTail
$ns duplex-link-op $n0 $n1 orient right-dow
$ns duplex-link-op $n1 $n3 orient right
$ns duplex-link-op $n2 $n1 orient right-up
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set ftp [new Application/FTP]
$ftp attach-agent $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
set udp [new Agent/UDP]
$ns attach-agent $n2 $udp
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $tcp $sink
$ns connect $udp $null
$ns rtmodel-at 1.0 down $n1 $n3
$ns rtmodel-at 2.0 up $n1 $n3
$ns rtproto DV
$ns at 0.0 "$ftp start"
$ns at 0.0 "$cbr start"
$ns at 5.0 "finish"
$ns run
```

PROGRAM &EXECUTION	
CLASS PERFORMNCE	
VIVA	
TOTAL	

RESULT:

Hence, the program for creating simulation of distance vector/link state routing was implemented.

Ex No: 13	Simulation of Link State Routing
Date:	

AIM:

To simulate and study the link state routing algorithm using simulation using NS2.

Link State Routing protocol

In link state routing, each router shares its knowledge of its neighborhood with every other router in the internet work. (i) Knowledge about Neighborhood: Instead of sending its entire routing table a router sends info about its neighborhood only. (ii) To all Routers: each router sends this information to every other router on the internet work not just to its neighbor .It does so by a process called flooding. (iii)Information sharing when there is a change: Each router sends out information about the neighbors when there is change.

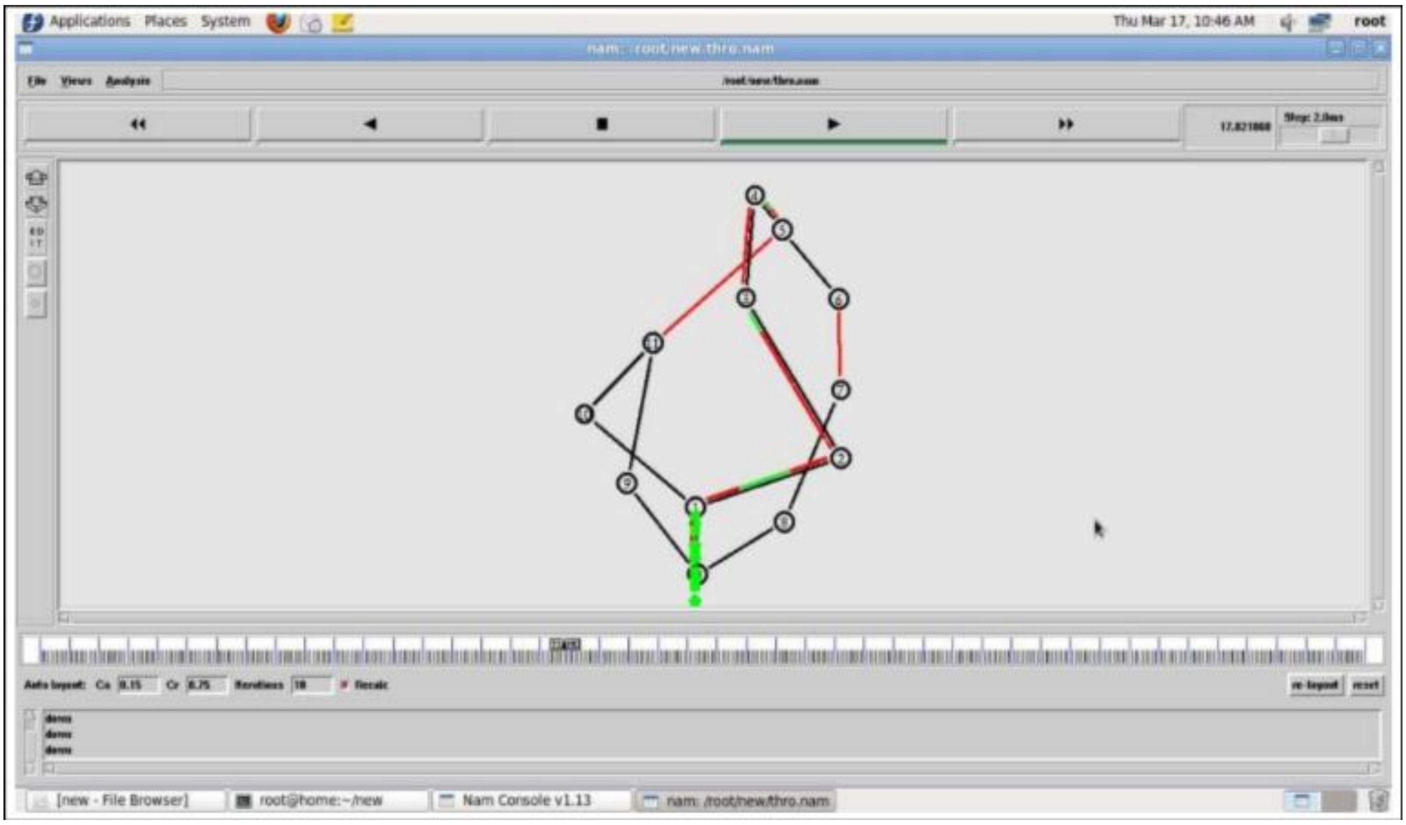
ALGORITHM:

Step-1 : Create a simulator object
Step-2 : Define different colors for different data flows
Step-3 : Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
Step-4 : Create n number of nodes using for loop
Step-5 : Create duplex links between the nodes
Step-6 : Setup UDP Connection between n(0) and n(5)
Step-7 : Setup another UDP connection between n(1) and n(5)
Step-8 : Apply CBR Traffic over both UDP connections
Step-9 : Choose Link state routing protocol to transmit data from sender to receiver.
Step-10 : Schedule events and run the program.

PROGRAM:

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
set tr [open out.tr w]
$ns trace-all $tr
proc finish {} {
global nf ns tr
$ns flush-trace
close $tr
exec nam out.nam &
exit 0
}
set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns node]
$ns duplex-link $n0 $n1 10Mb 10ms DropTail
$ns duplex-link $n1 $n3 10Mb 10ms DropTail
$ns duplex-link $n2 $n1 10Mb 10ms DropTail
$ns duplex-link-op $n0 $n1 orient right-down
$ns duplex-link-op $n1 $n3 orient right
$ns duplex-link-op $n2 $n1 orient right-up
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set ftp [new Application/FTP]
$ftp attach-agent $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
set udp [new Agent/UDP]
$ns attach-agent $n2 $udp
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $tcp $sink
$ns connect $udp $null
$ns rtmodel-at 1.0 down $n1 $n3
```

OUTPUT:


Applications Places System
Thu Mar 17, 10:46 AM root
nam: /root/new/thro.nam
File Views Analysis
/root/new/thro.nam
17.821868
Step: 2.0ms
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout reset
demo
demo
demo
[new - File Browser]
root@home:~/new
Nam Console v1.13
nam: /root/new/thro.nam

```
$ns rtmodel-at 2.0 up $n1 $n3
$ns rtproto LS
$ns at 0.0 "$ftp start"
$ns at 0.0 "$cbr start"
$ns at 5.0 "finish"
$ns run
```

PROGRAM &EXECUTION	
CLASS PERFORMNCE	
VIVA	
TOTAL	

RESULT:

Hence, the program for creating simulation of distance vector/link state routing was implemented.

Ex No: 14	**Performance Evaluation of Routing Protocols**
Date:	

AIM:

To study the performance evaluation of routing protocols.

ROUTING PROTOCOLS

There are many routing protocols available. Among them all we are working with AODV and DSR for performance analysis.

1. Ad-hoc On demand Distance Vector (AODV)

It is purely On-Demand route acquisition routing protocol. It is better protocol than DSDV network as the size of network may increase depending on the number of vehicle nodes.

Path Discovery Process: In order to discover the path between source and destination, a Route Request

message (RREQ) is broadcasted to all the neighbours in radio range who again continue to send the same to their neighbours in there radio range, until the destination is reached. Every node maintains two counters: sequence number and broadcast-id in order to maintain loop-free and most recent route information. The broadcast-id is incremented for every RREQ the source node initiates. If an intermediate node receives the same copy of request, it discards it without routing it further. When a node forwards the RREQ message, it records the address of the neighbour from which it received the first copy of the broadcast packet, in order to maintain a reverse path to the source node. The RREQ packet contains: the source sequence number and the last destination sequence number know to the source. The source sequence number is used to maintain information about reverse route and destination sequence number tells about the actual distance to the final node.

Route Maintenance: A source node sends a new moving request packet RREQ to find a new route to the

destination. But, if an intermediate node moves from its place, its upstream neighbor noticed the move and sends a message notification failure of the link to each of its active upstream neighbors to inform them about the move to source nodes is achieved. After the detection process is again initiated.

2. Dynamic Source Routing (DSR)

It is an On-Demand routing protocol in which the sequence of nodes through which a packet needs to travel is calculated and maintained as an information in packet header. Every mobile node in the network needs to maintain a route cache where it caches source routes that it has learned. When a packet is sent, the route-cache inside the node is compared with the actual route needs to be covered.

Route Discovery: The source node broadcasts request-packets to all the neighbours in the network containing the address of the destination node, and a reply is sent back to the source node with the list of network-nodes through which it should propagate in the process. Sender initiates the route record as a list with a single element containing itself followed by the linking of its neighbour in that route. A request packet also contains an identification number called request-id, which is counter increased only when a new route request packet is being sent by the source node. To make sure that no loops occur during broadcast, the request is processed in the given order. A route reply is obtained in DSR by two ways: Symmetric-links (bidirectional), in which the backward route is followed again to catch the source node. Asymmetric-links (unidirectional) needs to discover the route up to the source node in the same manner as the forward route is discovered.

Route Maintenance: In the hop by hop acknowledgement at data link layer allows the early detection and retransmission of lost or corrupt packets in the data-link layer. If a transmission error occurs, a route error packet containing the address of node detecting the error and the host address is sent back to the sender. Whenever a node receives a route error packet, the hop in error is removed from the route cache and all routes containing this hop are truncated at that point. When the wireless transmission between two nodes does not work equally well in both directions, and then end-to-end replies on the application or transport layer may be used to indicate the status of the route from one host to the other.

PROGRAM &EXECUTION	
CLASS PERFORMNCE	
VIVA	
TOTAL	

RESULT:

Hence, the performance evaluation protocol was studied

Ex No: 15	**Simulation of Error Correction Code (ERC).**
Date:	

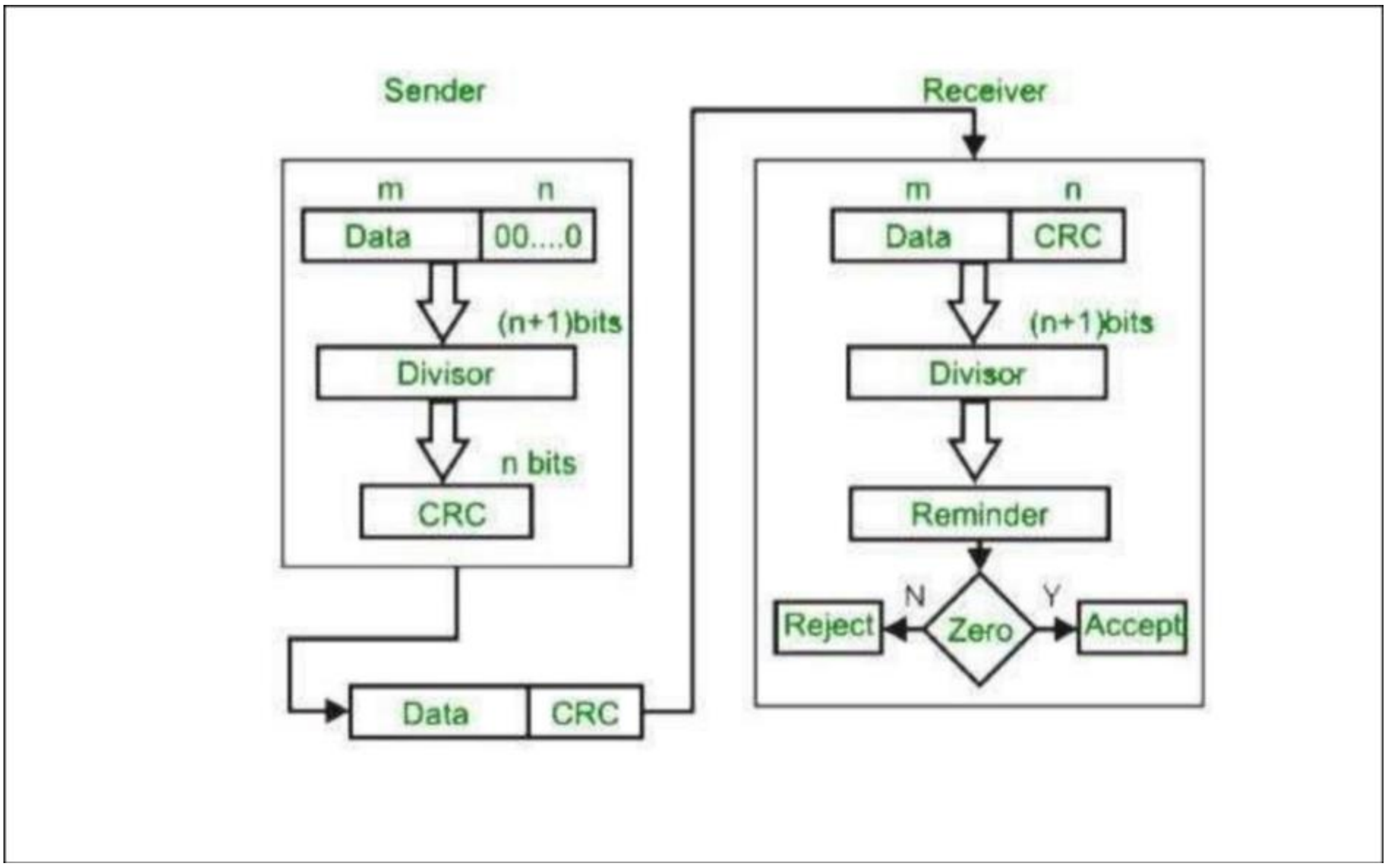
AIM:

To create simulation of Error Correction Code using java.

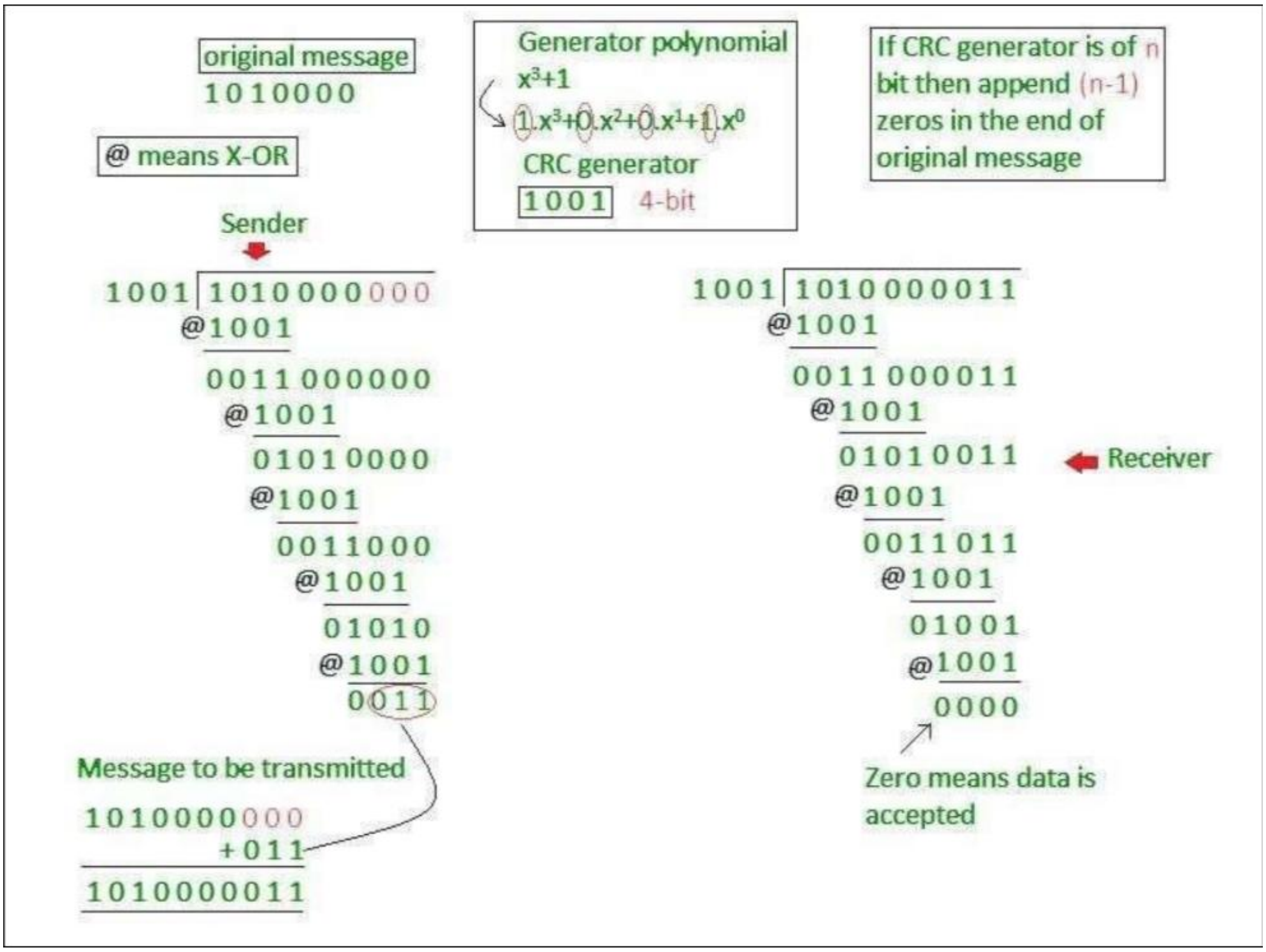
THEORY:

Cyclic redundancy check (CRC)

- Unlike checksum scheme, which is based on addition, CRC is based on binary division.
- In CRC, a sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number.
- At the destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted.
- A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.



Example:



ALGORITHM:

Step-1 : Open the editor and type the program for error detection

Step-2 : sGet the input in the form of bits.

Step-3 : Append the redundancy bits.

Step-4 : Divide the appended data using a divisor polynomial.

Step-5 : The resulting data should be transmitted to the receiver.

Step-6 : At the receiver the received data is entered.

Step-7 : The same process is repeated at the receiver.

Step-8 : If the remainder is zero there is no error otherwise there is some error in the received bits

Step-9 : Run the program.

PROGRAM:

```
import java.io.*;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())
```

OUTPUT:

```
Enter Generator:
1011
Enter Data:
1001010
The transmitted Code Word is: 1001010111
Please enter the received Code Word:
1110110111
The received code word contains errors.
```

```
{
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());
}
}
```

PROGRAM &EXECUTION	
CLASS PERFORMNCE	
VIVA	
TOTAL	

RESULT:

Hence, the error detection and error correction was implemented successfully.