

Ex No : 1	NETWORKING COMMANDS
Date :	

AIM

To study the basic networking commands.

C:\>arp -a: ARP is short form of address resolution protocol. It will show the IP address of your computer along with the IP address and MAC address of your router.

C:\>hostname: This is the simplest of all TCP/IP commands. It simply displays the name of your computer.

C:\>ipconfig: The ipconfig command displays information about the host (the computer you're sitting at) computer TCP/IP configuration.

C:\>ipconfig /all: This command displays detailed configuration information about your TCP/IP connection including Router, Gateway, DNS, DHCP, and type of Ethernet adapter in your system.

C:\>Ipconfig /renew: Using this command will renew all your IP addresses that you are currently (leasing) borrowing from the DHCP server. This command is a quick problem solver if you are having connection issues, but does not work if you have been configured with a static IP address.

C:\>Ipconfig /release: This command allows you to drop the IP lease from the DHCP server.

C:\>ipconfig /flushdns: This command is only needed if you're having trouble with your network's DNS configuration. The best time to use this command is after network configuration frustration sets in, and you really need the computer to reply with flushed.

C:\>nbtstat -a: This command helps solve problems with NetBIOS name resolution. (Nbt stands for NetBIOS over TCP/IP)

C:\>netdiag: Netdiag is a network testing utility that performs a variety of network diagnostic tests, allowing you to pinpoint problems in your network. Netdiag isn't installed by default, but can be installed from the Windows XP CD after saying no to the install. Navigate to the CD ROM drive letter and open the support\tools folder on the XP CD and click the setup.exe icon in the support\tools folder.

C:\>netstat: Netstat displays a variety of statistics about a computer's active TCP/IP connections. This tool is most useful when you're having trouble with TCP/IP applications such as HTTP, and FTP.

C:\>nslookup: Nslookup is used for diagnosing DNS problems. If you can access a resource by specifying an IP address but not its DNS you have a DNS problem.

C:\>pathping: Pathping is unique to Windows, and is basically a combination of the Ping and Tracert commands. Pathping traces the route to the destination address then launches a 25 second test of each router along the way, gathering statistics on the rate of data loss along each hop.

C:\>ping: Ping is the most basic TCP/IP command, and it's the same as placing a phone call to your best friend. You pick up your telephone and dial a number, expecting your best friend to reply with "Hello" on the other end. Computers make phone calls to each other over a network by using a Ping command. The Ping command's main

purpose is to place a phone call to another computer on the network, and request an answer. Ping has 2 options it can use to place a phone call to another computer on the network. It can use the computers name or IP address.

C:\>route: The route command displays the computers routing table. A typical computer, with a single network interface, connected to a LAN, with a router is fairly simple and generally doesn't pose any network problems. But if you're having trouble accessing other computers on your network, you can use the route command to make sure the entries in the routing table are correct.

C:\>tracert: The tracert command displays a list of all the routers that a packet has to go through to get from the computer where tracert is run to any other computer on the internet.

C:\>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.

RESULT

Thus the above list of primitive has been studied.

Ex No : 2	Download a web page using TCP Sockets

AIM:

To write a http web client program to download a web page using TCP sockets in java

Problem Statement:

A client program to get the file name from the user.

A HTTP server program that resolves the given file and displays the contents of the file.

Concept: HTTP

1. The Hypertext Transfer Protocol (HTTP) is a protocol used mainly to access data on the WWW.
2. HTTP functions as a combination of FTP and SMTP.
3. SMTP messages are stored and forwarded, but HTTP messages are delivered immediately.
4. The commands from the client to the server are embedded in a request message. The contents of the requested file or other information are embedded in a response message.

Algorithm:

Step1: Set a server port as 80.

Step2: Using HTTP services create a Socket for server by specifying the server port.

Step3: Use HTTP socket for connecting the client to the URL.

Step4: Use BufferedReader to output stream to place the response from the server by the client.

Step5: Close the Connection as soon the request is been serviced. Use Malformed URL exception If any errors in grabbing the server

PROGRAM:

```
import java.io.*;

import java.net.*;

public class SocketHTTPClient {
```

```

public static void main(String[] args) {

    String hostName = "www.gmail.com";

    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.gmail.com\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

```
C:\Windows\System32\cmd.exe - java SocketHTTPClient
Usage: javac <options> <source files>
use -help for a list of possible options

D:\>javac SocketHTTPClient.java

D:\>java SocketHTTPClient
HTTP/1.1 301 Moved Permanently
Location: https://www.google.com/gmail/
Content-Type: text/html; charset=UTF-8
X-Content-Type-Options: nosniff
Date: Fri, 09 Aug 2019 02:27:33 GMT
Expires: Sat, 10 Aug 2019 02:27:33 GMT
Server: sffe
Content-Length: 226
X-XSS-Protection: 0
Cache-Control: public, max-age=86400
Age: 22319

<HTML><HEAD><meta http-equiv="content-type" content="text/html; charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="https://www.google.com/gmail/">here</A>.
</BODY></HTML>
```

Result:

Thus the program for creating sockets for HTTP web page upload and download was implemented

Ex No : 3	TCP ECHO CLIENT/SERVER
Date :	

AIM:

To write java program for TCP echo client server.

ALGORITHM:

SERVER:

- STEP 1: Start
- STEP 2: Declare the variables for the socket
- STEP 3: Specify the family, protocol, IP address and port number
- STEP 4: Create a socket using socket() function
- STEP 5: Bind the IP address and Port number
- STEP 6: Listen and accept the client's request for the connection
- STEP 7: Read the client's message
- STEP 8: Display the client's message
- STEP 9: Close the socket
- STEP 10: Stop

CLIENT:

- STEP 1: Start
- STEP 2: Declare the variables for the socket
- STEP 3: Specify the family, protocol, IP address and port number
- STEP 4: Create a socket using socket() function
- STEP 5: Call the connect() function
- STEP 6: Read the input message
- STEP 7: Send the input message to the server
- STEP 8: Display the server's echo
- STEP 9: Close the socket
- STEP 10: Stop

PROGRAM:

SERVER PROGRAM:

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

```

import java.net.*;
import java.rmi.*;
class echos
{
    public static void main(String args[])throws Exception
    {
        try
        {
            ServerSocket ss=new ServerSocket(2100);

            Socket s=ss.accept();

            DataInputStream in=new DataInputStream(s.getInputStream());

            DataOutputStream out=new DataOutputStream(s.getOutputStream());

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

            String msg=in.readUTF();

            System.out.println("The message from client is:"+msg);

            out.writeUTF(msg);

        }
        catch(Exception e)
        {
            e.printStackTrace();

        }

    }
}

```

CLIENT PROGRAM:

```

import java.io.*;
import java.net.*;
import java.rmi.*;
class echoc
{
    public static void main(String args[])throws Exception

```

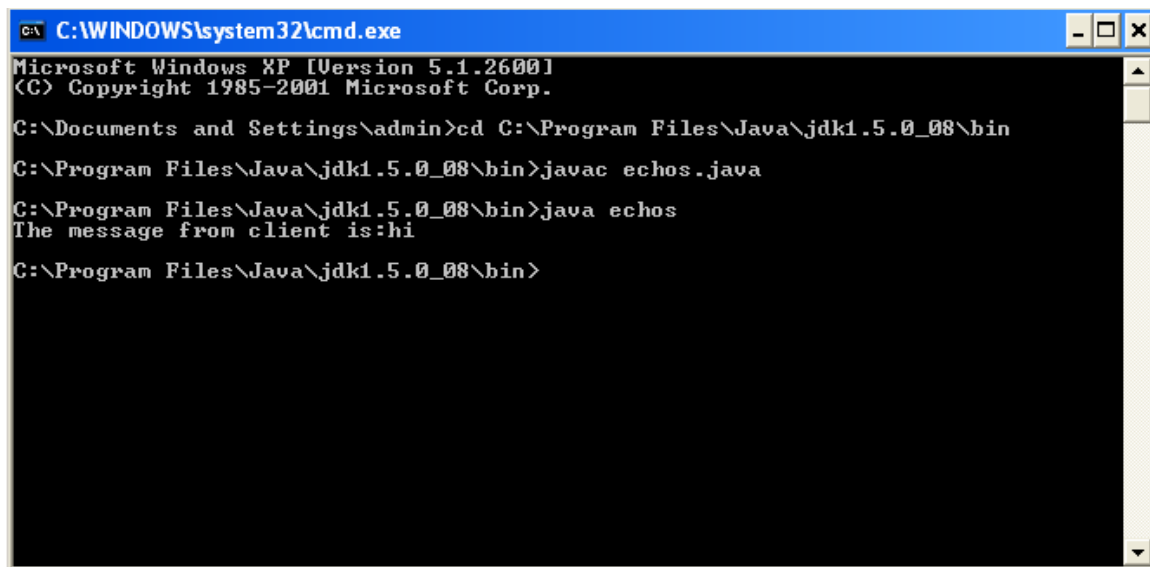
```

{
    try
    {
        Socket s=new Socket(InetAddress.getLocalHost(),2100);
        DataInputStream in=new DataInputStream(s.getInputStream());
        DataOutputStream out=new DataOutputStream(s.getOutputStream());
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the message:");
        String msg1=br.readLine();
        out.writeUTF(msg1);
        String msg2=in.readUTF();
        System.out.println("The message from client is:"+msg2);
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
}

```

OUTPUT:

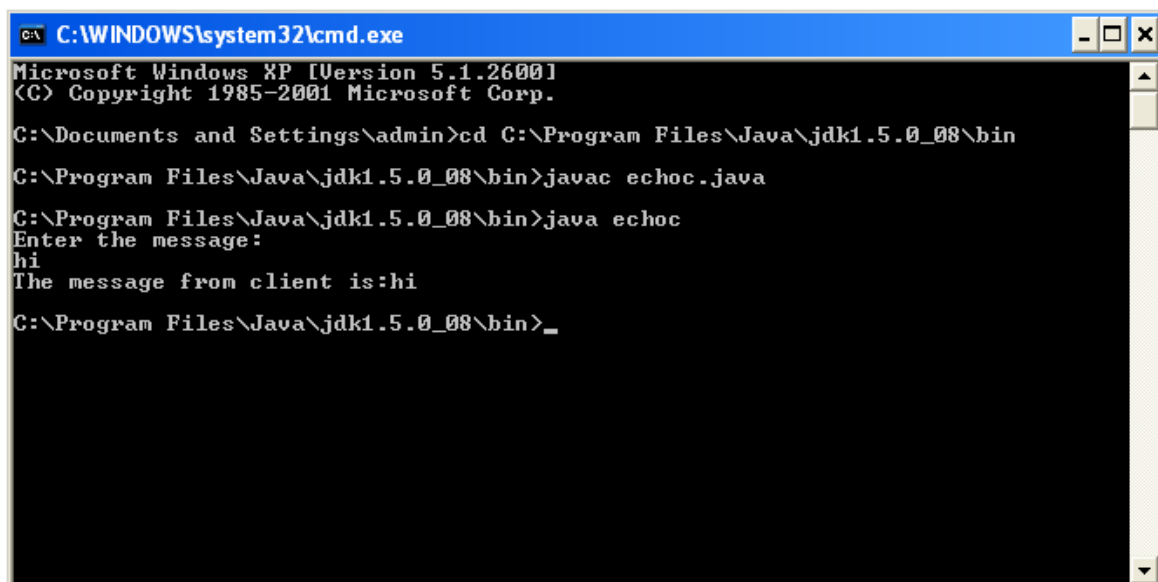
Server



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\admin>cd C:\Program Files\Java\jdk1.5.0_08\bin
C:\Program Files\Java\jdk1.5.0_08\bin>javac echos.java
C:\Program Files\Java\jdk1.5.0_08\bin>java echos
The message from client is:hi
C:\Program Files\Java\jdk1.5.0_08\bin>
```

Client



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\admin>cd C:\Program Files\Java\jdk1.5.0_08\bin
C:\Program Files\Java\jdk1.5.0_08\bin>javac echoc.java
C:\Program Files\Java\jdk1.5.0_08\bin>java echoc
Enter the message:
hi
The message from client is:hi
C:\Program Files\Java\jdk1.5.0_08\bin>_
```

RESULT:

Thus the java program to client/server using TCP Sockets was executed successfully

Ex No : 4	TCP CLIENT/SERVER CHAT
Date :	

AIM:

To implement a CHAT application, where the Client establishes a connection with the Server. The Client and Server can send as well as receive messages at the same time. Both the Client and Server exchange messages.

ALGORITHM:

Server

- Step1: Start the program and create server and client sockets.
- Step2: Use input streams to get the message from user.
- Step3: Use output streams to send message to the client.
- Step4: Wait for client to display this message and write a new one to be displayed by the server.
- Step5: Display message given at client using input streams read from socket.
- Step6: Stop the program.

Client

- Step1: Start the program and create a client socket that connects to the required host and port.
- Step2: Use input streams read message given by server and print it.
- Step3: Use input streams; get the message from user to be given to the server.
- Step4: Use output streams to write message to the server.
- Step5: Stop the program.

PROGRAM:

TCP Chat Server--tcpchatserver.java

```
import java.io.*;
import java.net.*;
class tcpchatserver
{
public static void main(String args[])throws Exception
{
    PrintWriter toClient;
    BufferedReader fromUser, fromClient;
    try
    {
        ServerSocket Srv = new ServerSocket(5555);
        System.out.print("\nServer started\n");
        Socket Clt = Srv.accept();
        System.out.println("Client connected");
        toClient = new PrintWriter(new BufferedWriter(new OutputStreamWriter(Clt.getOutputStream())), true);
        fromClient = new BufferedReader(new InputStreamReader(Clt.getInputStream()));
        fromUser = new BufferedReader(new InputStreamReader(System.in));
        String CltMsg, SrvMsg;
        while(true)
        {
            CltMsg= fromClient.readLine();
            if(CltMsg.equals("end"))
                break;
            else
            {
                System.out.println("\nServer <<< " +CltMsg);
                System.out.print("Message to Client : ");
                SrvMsg = fromUser.readLine();
                toClient.println(SrvMsg);
            }
        }
        System.out.println("\nClient Disconnected");
        fromClient.close();
        toClient.close();
        fromUser.close();
        Clt.close();
        Srv.close();}
    catch (Exception E)
    {
        System.out.println(E.getMessage());
    }
}}
```

TCP Chat Client--tcpchatclient.java

```
import java.io.*;
import java.net.*;
class tcpchatclient
{
public static void main(String args[])throws Exception
{
Socket Clt;
PrintWriter toServer;
BufferedReader fromUser, fromServer;
try
{
if (args.length > 1)
{
System.out.println("Usage: java hostipaddr");
System.exit(-1);
}
if (args.length == 0)
Clt = new Socket(InetAddress.getLocalHost(),5555);
else
Clt = new Socket(InetAddress.getByName(args[0]), 5555);
toServer = new PrintWriter(new BufferedWriter(new OutputStreamWriter(Clt.getOutputStream())), true);
fromServer = new BufferedReader(new InputStreamReader(Clt.getInputStream()));
fromUser = new BufferedReader(new InputStreamReader(System.in));
String CltMsg, SrvMsg;
System.out.println("Type \"end\" to Quit");
while (true)
{
System.out.print("\nMessage to Server : ");
CltMsg = fromUser.readLine();
toServer.println(CltMsg);
if (CltMsg.equals("end"))
break;
SrvMsg = fromServer.readLine();
System.out.println("Client <<< " + SrvMsg);
}
}
catch(Exception E)
{
System.out.println(E.getMessage());
}
}
```

OUTPUT:

Server:

```
$ javac tcpchatserver.java
$ java tcpchatserver
Server started
Client connected
Server <<< hi
Message to Client : hello
Server <<< how r u?
Message to Client : fine
Server <<< me too
Message to Client : bye
Client Disconnected
```

Client :

```
$ javac tcpchatclient.java
$ java tcpchatclient
Type "end" to Quit
Message to Server : hi
Client <<< hello
Message to Server : how r u?
Client <<< fine
Message to Server : me too
Client <<< bye
Message to Server : end
```

RESULT:

Thus the java program to chat using TCP Sockets was executed successfully

Ex No : 5	TCP FILE TRANSFER

AIM:

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

Problem Statement:

To transfer computer files between a client and server on a computer network using TCP Sockets.

Concept:

Our aim is to send a file from the client machine to the server machine using TCP Communication.

Algorithm

Server

Step1: Import java packages and create class file server.

Step2: Create a new server socket and bind it to the port.

Step3: Accept the client connection

Step4: Get the file name and stored into the BufferedReader.

Step5: Create a new object class file and realine.

Step6: If file is exists then FileReader read the content until EOF is reached.

Step7: Stop the program.

Client

Step1: Import java packages and create class file server.

Step2: Create a new server socket and bind it to the port.

Step3: Now connection is established.

Step4: The object of a BufferedReader class is used for storing data content which has been retrieved from socket object.

Step5: The content of file is displayed in the client window and the connection is closed.

Step6: Stop the program.

PROGRAM:

fs.java

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

public class fs
{
    public static void main(String args[])throws IOException
    {
        ServerSocket s1=null;

        try
        {
            s1=new ServerSocket(1187);
        }
        catch(IOException u1)
        {
            System.out.println("Could not found port 1187");
            System.exit(1);
        }

        Socket c=null;

        try
        {
            c=s1.accept();
            System.out.println("Connection Frame"+c);
        }
        catch(IOException e)
        {
            System.out.println("accept failed");
            System.exit(1);
        }

        PrintWriter out=new PrintWriter(c.getOutputStream(),true);
        BufferedReader sin=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the text file name:");
        String s=sin.readLine();
        File f=new File(s);
        if(f.exists())
        {
            BufferedReader in =new BufferedReader(new FileReader(s));
```

```

String v;
while((v=in.readLine())!=null)
{
out.write(v);
out.flush();
}
System.out.println("The file send successfully");
in.close();
c.close();
s1.close();
}
}
}

```

fc.java

```

import java.io.*;
import java.net.*;
public class fc
{
public static void main(String args[])throws IOException
{
Socket s=null;
BufferedReader b=null;
try
{
s=new Socket(InetAddress.getLocalHost(),1187);
b=new BufferedReader(new InputStreamReader(s.getInputStream()));
}
catch(Exception u)
{
System.out.println("The file is received");
System.out.println("Dont know host");
System.exit(1);
}
String inp;
while((inp=b.readLine())!=null)
{
System.out.println("The content of the file is");

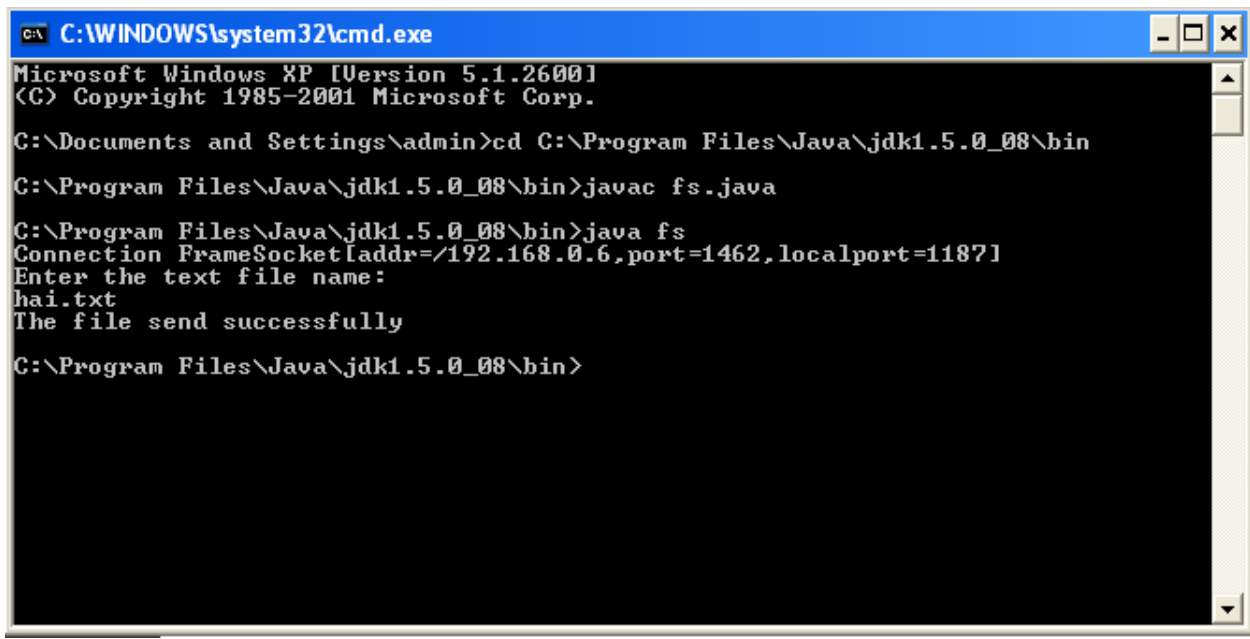
```



```
System.out.println(inp);
System.out.println("The file received successfully");
}
b.close();
s.close();
}
}
```

OUTPUT:

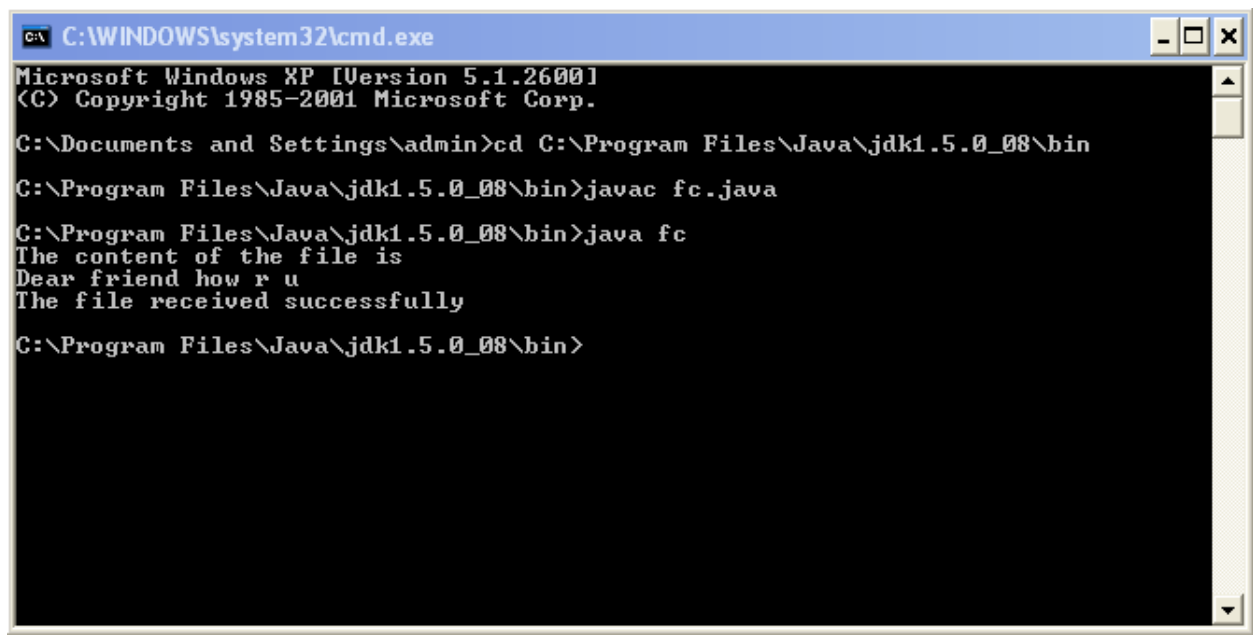
Server



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\admin>cd C:\Program Files\Java\jdk1.5.0_08\bin
C:\Program Files\Java\jdk1.5.0_08\bin>javac fs.java
C:\Program Files\Java\jdk1.5.0_08\bin>java fs
Connection FrameSocket[addr=/192.168.0.6,port=1462,localport=1187]
Enter the text file name:
hai.txt
The file send successfully
C:\Program Files\Java\jdk1.5.0_08\bin>
```

Client



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\admin>cd C:\Program Files\Java\jdk1.5.0_08\bin
C:\Program Files\Java\jdk1.5.0_08\bin>javac fc.java
C:\Program Files\Java\jdk1.5.0_08\bin>java fc
The content of the file is
Dear friend how r u
The file received successfully
C:\Program Files\Java\jdk1.5.0_08\bin>
```

RESULT:

Thus the java program file transfer application using TCP Sockets was executed

Ex No : 6	ADDRESS RESOLUTION PROTOCOL
Date :	

AIM:

To write a java program to implement Address Resolution Protocol.

.

Objective:

ARP is used to get the physical address (MAC address) of destination machine. Before sending the IP packet, the MAC address of destination must be known. If not so, then sender broadcasts the ARP-discovery packet requesting the MAC address of intended destination.

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:

Clientarp:

```
import java.io.*;
import java.net.*;
import java.util.*;
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",139);
            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);
        }
    }
}
```

Serverarp:

```
import java.io.*;
import java.net.*;
import java.util.*;
class Serverarp
{
    public static void main(String args[])
    {
        try
        {
            ServerSocket obj=new ServerSocket(139);
            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:

```

E:\networks>java Serverarp
E:\networks>java Clientarp
Enter the Logical address(IP):
165.165.80.80
The Physical Address is: 6A:08:AA:C2

```

RESULT:

Thus the MAC address was generated for IP address using ARP protocol

Ex No : 7	REVERSE ADDRESS RESOLUTION PROTOCOL
Date :	

AIM:

To write a java program to implement Address Resolution Protocol.

Objectives:

RARP (Reverse Address Resolution **Protocol**) is a **protocol** by which a physical machine in a local area **network** can request to learn its IP address from a gateway server's Address Resolution **Protocol** (ARP) table or cache.

ALGORITHM:

Client

1. Start the program
2. Using socket connection is established between client and server.
3. Get the MAC address to be converted into IP address.
4. Send this MAC 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 MAC address which is send by the client.
5. Map the MAC address with its IP address and return the IP address to client.

PROGRAM:

RARP Client -- rarpclient.java

```
import java.io.*;
import java.net.*;
import java.util.*;
class Clientrarp
{
    public static void main(String args[])
    {
        try
        {
            BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
            Socket clsct=new Socket("127.0.0.1",139);

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

RARP Server -- rarpserver.java

```
import java.io.*;
import java.net.*;
import java.util.*;
class Serverrarp
{
    public static void main(String args[])
    {
        try
        {
            ServerSocket obj=new ServerSocket(139);
            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<mac.length;i++)
        {
            if(str.equals(mac[i]))
            {
                dout.writeBytes(ip[i]+"\\n");
                break;
            }
        }
        obj.close();
    }
}
catch(Exception e)
{
    System.out.println(e);
}
}

```

OUTPUT:

Output:

```

E:\networks>java Serverrarp
E:\networks>java Clientrarp
Enter the Physical Address (MAC):
6A:08:AA:C2
The is Logical address(IP): 165.165.80.80

```

RESULT:

Thus the IP address was generated for MAC address using RARP protocol

Ex No : 8	<i>Domain Name System (DNS) using UDP</i>
Date :	

AIM:

To implement a DNS server and client in java using UDP sockets.

DESCRIPTION:

DNS stands for domain name system. Unique name of the host is identified with its IP address through server client communication.

ALGORITHM:

Server

1. Create an array of hosts and its ip address in another array
2. Create a datagram socket and bind it to a port
3. Create a datagram packet to receive client request
4. Read the domain name from client to be resolved
5. Lookup the host array for the domain name
6. If found then retrieve corresponding address
7. Create a datagram packet and send ip address to client
8. Repeat steps 3-7 to resolve further requests from clients
9. Close the server socket
10. Stop

Client

1. Create a datagram socket
2. Get domain name from user
3. Create a datagram packet and send domain name to the server
4. Create a datagram packet to receive server message
5. Read server's response
6. If ip address then display it else display "Domain does not exist"
7. Close the client socket
8. Stop

Program:**Client:**

```
import java.io.*;
import java.net.*;
import java.util.*;
class Clientdns12
{
    public static void main(String args[])
    {
        try
        {
            DatagramSocket client=new DatagramSocket();
            InetAddress addr=InetAddress.getByName("127.0.0.1");

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

Server:

```
import java.io.*;
import java.net.*;
import java.util.*;
class Serverdns12
{
    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[]={ "165.165.80.80","165.165.79.1"};
String name[]={ "www.apititudeguru.com","www.downloadcyclone.blogspot.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);
}
}
}

```

Output

```

I:\ex>java Serverdns12
I:\ex>java Clientdns12
Enter the DOMAIN NAME or IP adress:
165.165.80.80
IP address or DOMAIN NAME: www.apititudeguru.com

I:\ex>java Clientdns12
Enter the DOMAIN NAME or IP adress:
www.downloadcyclone.blogspot.com
IP address or DOMAIN NAME: 165.165.79.1

```

RESULT:

Thus domain name requests by the client are resolved into their respective logical address using lookup method.

Ex No : 9	STUDY OF NETWORK SIMULATOR (NS2)
Date :	

AIM:

To study of Network Simulator (NS2) tool.

INTRODUCTION

The network simulator is discrete event packet level simulator. The network simulator covers a very large number of applications of different kind of protocols of different network types consisting of different network elements and traffic models. Network simulator is a package of tools that simulates behavior of networks such as creating network topologies, log events that happen under any load, analyze the events and understand the network. Well the main aim of our first experiment is to learn how to use network simulator and to get acquainted with the simulated objects and understand the operations of network simulation and we also need to analyze the behavior of the simulation object using network simulation.

Platform required to run network simulator

- Unix and Unix like systems
- Linux (Use Fedora or Ubuntu versions)
- Free BSD
- SunOS/Solaris
- Windows 95/98/NT/2000/XP

Backend Environment of Network Simulator

Network Simulator is mainly based on two languages. They are C++ and OTcl. OTcl is the object oriented version of Tool Command language. The network simulator is a bank of different network and protocol objects. C++ helps in the following way:

- It helps to increase the efficiency of simulation.
- It is used to provide details of the protocols and their operation.
- It is used to reduce packet and event processing time.

OTcl helps in the following way:

- With the help of OTcl we can describe different network topologies
- It helps us to specify the protocols and their applications
- It allows fast development
- Tcl is compatible with many platforms and it is flexible for integration
- Tcl is very easy to use and it is available in free

Basics of Tcl Programming (w.r.t. ns2)

Before we get into the program we should consider the following things:

1. Initialization and termination aspects of network simulator.
2. Defining the network nodes, links, queues and topology as well.
3. Defining the agents and their applications
4. Network Animator(NAM)
5. Tracing

Initialization

To start a new simulator we write

1. `set ns [new Simulator]`

From the above command we get that a variable `ns` is being initialized by using the `set` command. Here the code `[new Simulator]` is a instantiation of the class `Simulator` which uses the reserved word 'new'. So we can call all the methods present inside the class `simulator` by using the variable `ns`.

Creating the output files

- 1 `#To create the trace files we write`
- 2 `set tracefile1 [open out.tr w]`
- 3 `$ns trace-all $tracefile1`
- 4 `#To create the nam files we write`
- 5 `set namfile1 [open out.nam w]`
- 6 `$ns namtrace-all $namfile1`

In the above we create a output trace file `out.tr` and a nam visualization file `out.nam`. But in the Tcl script they are not called by their names declared, while they are called by the pointers initialized for them such as `tracefile1` and `namfile1` respectively. The line which starts with '#' are commented. The next line opens the file 'out.tr' which is used for writing is declared 'w'.The next line uses a simulator method `trace-all` by which we will trace all the events in a particular format. The termination program is done by using a 'finish' procedure

- 1 `# Defining the 'finish' procedure'`
- 2 `proc finish {} {`
- 3 `global ns tracefile1 namfile1`
- 4 `$ns flush-trace`
- 5 `close $tracefile1`
- 6 `close $namfile1`
- 7 `exec nam out.nam &`
- 9 `exit 0`

10 }

In the above the word 'proc' is used to declare a procedure called 'finish'.The word 'global' is used to tell what variables are being used outside the procedure.

'flush-trace' is a simulator method that dumps the traces on the respective files.the command 'close' is used to close the trace files and the command 'exec' is used to execute the nam visualization.The command 'exit' closes the application and returns 0 as zero(0) is default for clean exit.

In ns we end the program by calling the 'finish' procedure

```
1  #end the program
2  $ns at 125.0 "finish"
```

Thus the entire operation ends at 125 seconds.To begin the simulation we will use the command

```
1  #start the the simulation process
2  $ns run
```

Defining nodes,links,queues and topology

Way to create a node:

view source

print?

```
1 set n0 [ns node]
```

In the above we created a node that is pointed by a variable n0.While referring the node in the script we use \$n0. Similarly we create another node n2.Now we will set a link between the two nodes.

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

So we are creating a bi-directional link between n0 and n2 with a capacity of 10Mb/sec and a propagation delay of 10ms.

In NS an output queue of a node is implemented as a part of a link whose input is that node to handle the overflow at the queue.But if the buffer capacity of the output queue is exceeded then the last packet arrived is dropped and here we will use a 'DropTail' option.Many other options such as RED(Random Early Discard) mechanism, FQ(Fair Queuing), DRR(Deficit Round Robin), SFQ(Stochastic Fair Queuing) are available.

So now we will define the buffer capacity of the queue related to the above link

```
1  #Set queue size of the link
2  $ns queue-limit $n0 $n2 20
```

so, if we summarize the above three things we get

```
1  #create nodes
```

```

2. set n0 [$ns node]
2   set n1 [$ns node]

3   set n2 [$ns node]
4   set n3 [$ns node]
5   set n4 [$ns node]
6   set n5 [$ns node]
7   #create links between the nodes
8   $ns duplex-link $n0 $n2 10Mb 10ms DropTail
9   $ns duplex-link $n1 $n2 10Mb 10ms DropTail
10  $ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
11  $ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
12  $ns duplex-link $n0 $n2 0.5Mb 40ms DropTail
13  $ns duplex-link $n0 $n2 0.5Mb 40ms DropTail
14  #set queue-size of the link (n2-n3) to 20
15  $ns queue-limit $n2 $n3 20

```

Agents and applications

TCP

TCP is a dynamic reliable congestion protocol which is used to provide reliable transport of packets from one host to another host by sending acknowledgements on proper transfer or loss of packets. Thus TCP requires bi-directional links in order for acknowledgements to return to the source.

Now we will show how to set up tcp connection between two nodes

```

1  #setting a tcp connection
2  set tcp [new Agent/TCP]
4  $ns attach-agent $n0 $tcp
5  set sink [new Agent/TCPSink]
6  $ns attach-agent $n4 $sink
7  $ns connect $tcp $sink
8  $tcp set fid_1
9  $tcp set packetSize_552

```

The command 'set tcp [new Agent/TCP]' gives a pointer called 'tcp' which indicates the tcp agent which is a object of ns. Then the command '\$ns attach-agent \$n0 \$tcp' defines the source node of tcp connection. Next the command 'set sink [new Agent/TCPSink]' defines the destination of tcp by a pointer called sink. The next command '\$ns attach-agent \$n4 \$sink' defines the destination node as n4. Next, the command '\$ns connect \$tcp \$sink' makes the TCP connection between the source and the destination. i.e n0 and n4. When we have several flows such as TCP, UDP etc in a network. So, to identify these flows we mark these flows by using the command '\$tcp set fid_1'. In the last line we set the packet size of tcp as 552 while the default packet size of tcp is 1000.

FTP over TCP

File Transfer Protocol(FTP) is a standard mechanism provided by the Internet for transferring files from one host to another. Well this is the most common task expected from a networking or a inter networking . FTP differs from other client server applications in that it establishes between the client and the server. One connection is used for data transfer and other one is used for providing control information. FTP uses the services of the TCP. It needs two connections. The well Known port 21 is used for control connections and the other port 20 is used for data transfer.

Well here we will learn in how to run a FTP connection over a TCP

- 1 #Initiating FTP over TCP
- 2 set ftp [new Application/FTP]
- 3 \$ftp attach-agent \$tcp

In above,the command 'set ftp [new Application/FTP]' gives a pointer called 'ftp' which indicates the FTP application.Next, we attach the ftp application with tcp agent as FTP uses the services of TCP.

UDP

The User datagram Protocol is one of the main protocols of the Internet protocol suite.UDP helps the host to send send messages in the form of datagrams to another host which is present in a Internet protocol network without any kind of requirement for channel transmission setup. UDP provides a unreliable service and the datagrams may arrive out of order,appear duplicated, or go missing without notice. UDP assumes that error checking and correction is either not necessary or performed in the application, avoiding the overhead of such processing at the network interface level. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for delayed packets, which may not be an option in a real-time system.

Now we will learn how to create a UDP connection in network simulator.

- 1 # setup a UDP connection
- 2 set udp [new Agent/UDP]
- 3 \$ns attach-agent \$n1 \$udp
- 4 \$set null [new Agent/Null]
- 5 \$ns attach-agent \$n5 \$null

```
6 $ns connect $udp $null
```

```
7 $udp set fid_2
```

Similarly, the command 'set udp [new Agent/UDP]' gives a pointer called 'udp' which indicates the udp agent which is a object of ns. Then the command '\$ns attach-agent \$n1 \$udp' defines the source node of udp connection. Next the command 'set null [new Agent/Null]' defines the destination of udp by a pointer called null. The next command '\$ns attach-agent \$n5 \$null' defines the destination node as n5. Next, the command '\$ns connect \$udp \$null' makes the UDP connection between the source and the destination. i.e n1 and n5. When we have several flows such as TCP, UDP etc in a network. So, to identify these flows we mark these flows by using the command '\$udp set fid_2

Constant Bit Rate(CBR)

Constant Bit Rate (CBR) is a term used in telecommunications, relating to the quality of service. When referring to codecs, constant bit rate encoding means that the rate at which a codec's output data should be consumed is constant. CBR is useful for streaming multimedia content on limited capacity channels since it is the maximum bit rate that matters, not the average, so CBR would be used to take advantage of all of the capacity. CBR would not be the optimal choice for storage as it would not allocate enough data for complex sections (resulting in degraded quality) while wasting data on simple sections.

CBR over UDP Connection

```
1 #setup cbr over udp
```

```
2 set cbr [new Application/Traffic/CBR]
```

```
3 $cbr attach-agent $udp
```

```
4 $cbr set packetSize_1000
```

```
5 $cbr set rate_0.01Mb
```

```
6 $cbr set random_false
```

In the above we define a CBR connection over a UDP one. Well we have already defined the UDP source and UDP agent as same as TCP. Instead of defining the rate we define the time interval between the transmission of packets in the command '\$cbr set rate_0.01Mb'. Next, with the help of the command '\$cbr set random_false' we can set random noise in cbr traffic. we can keep the noise by setting it to 'false' or we can set the noise on by the command '\$cbr set random_1'. We can set by packet size by using the command '\$cbr set packetSize_(packetsize)'. We can set the packet size up to sum value in bytes.

Scheduling Events

In ns the tcl script defines how to schedule the events or in other words at what time which event will occur and stop. This can be done using the command \$ns at .

So here in our program we will schedule the ftp and cbr.

```
1 # scheduling the events
2 $ns at 0.1 "cbr start"
3 $ns at 1.0 "ftp start"
4 $ns at 124.0 "ftp stop"
5 $ns at 124.5 "cbr stop"
```

Network Animator (NAM)

When we will run the above program in ns then we can visualize the network in the NAM. But instead of giving random positions to the nodes, we can give suitable initial positions to the nodes and can form a suitable topology. So, in our program we can give positions to the nodes in NAM in the following way

```
1 #Give position to the nodes in NAM
2 $ns duplex-link-op $n0 $n2 orient-right-down
3 $ns duplex-link-op $n1 $n2 orient-right-up
4 $ns simplex-link-op $n2 $n3 orient-right
5 $ns simplex-link-op $n3 $n2 orient-left
6 $ns duplex-link-op $n3 $n4 orient-right-up
7 $ns duplex-link-op $n3 $n5 orient-right-down
```

We can also define the color of cbr and tcp packets for identification in NAM. For this we use the following command

```
1 #Marking the flows
2 $ns color1 Blue
3 $ns color2 Red
```

To view the network animator we need to type the command: nam

Tracing

Tracing Objects

NS simulation can produce visualization trace as well as ASCII file corresponding to the events that are registered at the network. While tracing ns inserts four objects: EnqT, DeqT, RecvT & DrpT. EnqT registers information regarding the arrival of packet and is queued at the input queue of the link. When overflow of a packet occurs, then the information of the dropped packet is registered in DrpT. DeqT holds the information about the packet that is dequeued instantly. RecvT holds the information about the packet that has been received instantly.

Event	Time	From node	To node	Pkt type	Pkt size	Flags	Fid	Src addr	Dst addr	Seq num	Pkt id
-------	------	-----------	---------	----------	----------	-------	-----	----------	----------	---------	--------

Structure of Trace files

1. The first field is event. It gives you four possible symbols '+', '-', 'r', 'd'. These four symbols correspond respectively to enqueued, dequeued, received and dropped.
2. The second field gives the time at which the event occurs
3. The third field gives you the input node of the link at which the event occurs
4. The fourth field gives you the output node at which the event occurs
5. The fifth field shows the information about the packet type. i.e. whether the packet is UDP or TCP
6. The sixth field gives the packet size
7. The seventh field gives information about some flags
8. The eighth field is the flow id (fid) for IPv6 that a user can set for each flow in a tcl script. It is also used for specifying the color of flow in NAM display
9. The ninth field is the source address
10. The tenth field is the destination address
11. The eleventh field is the network layer protocol's packet sequence number
12. The last field shows the unique id of packet

Following are trace of two events:

```
r 1.84471 2 1 cbr 210 ----- 1 3.0 1.0 195 600
```

```
r 1.84566 2 0 ack 40 ----- 2 3.2 0.1 82 602
```

The trace file can be viewed with the cat command:

```
cat out.tr
```

RESULT

Thus the introduction of Network Simulator was studied.

Ex No : 10	SIMULATION OF CONGESTION CONTROL ALGORITHM
Date :	

AIM:

To simulate a link failure and observe the congestion control algorithm using NS2.

ALGORITHM:

1. Create a simulation object
2. Set routing protocol to routing
3. Trace packets and all links onto NAM trace and to trace file
4. Create right nodes
5. Describe their layout topology as octagon
6. Add a sink agent to node
7. Connect source and sink.

PROGRAM:

```

set ns [new Simulator]

set nr [open thro_red.tr w]

$ns trace-all $nr

set nf [open thro.nam w]

$ns namtrace-all $nf

proc finish { } {

    global ns nr nf

    $ns flush-trace

    close $nf

    close $nr

    exec nam thro.nam &

```

```
    exit 0 }

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]

set n7 [$ns node]

$ns duplex-link $n0 $n3 1Mb 10ms RED
$ns duplex-link $n1 $n3 1Mb 10ms RED
$ns duplex-link $n2 $n3 1Mb 10ms RED
$ns duplex-link $n3 $n4 1Mb 10ms RED
$ns duplex-link $n4 $n5 1Mb 10ms RED
$ns duplex-link $n4 $n6 1Mb 10ms RED
$ns duplex-link $n4 $n7 1Mb 10ms RED
$ns duplex-link-op $n0 $n3 orient right-up
$ns duplex-link-op $n3 $n4 orient middle
$ns duplex-link-op $n2 $n3 orient right-down
$ns duplex-link-op $n4 $n5 orient right-up
$ns duplex-link-op $n4 $n7 orient right-down
$ns duplex-link-op $n1 $n3 orient right
$ns duplex-link-op $n6 $n4 orient left set udp0 [new Agent/UDP] $ns attach-agent $n2 $udp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

```
set null0 [new Agent/Null]
$ns attach-agent $n5 $null0
$ns connect $udp0 $null0
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
set cbr1 [new Application/Traffic/CBR] $cbr1 set packetSize_ 500 $cbr1 set interval_ 0.005

$cbr1 attach-agent $udp1
set null0 [new Agent/Null]
$ns attach-agent $n6 $null0
$ns connect $udp1 $null0
set udp2 [new Agent/UDP]
$ns attach-agent $n0 $udp2
set cbr2 [new Application/Traffic/CBR] $cbr2 set packet size_ 500 $cbr2 set interval_ 0.005

$cbr2 attach-agent $udp2
set null0 [new Agent/Null]
$ns attach-agent $n7 $null0
$ns connect $udp2 $null0
$udp0 set fid_ 1
$udp1 set fid_ 2
$udp2 set fid_ 3
$ns color 1 Red
$ns color 2 Green
$ns color 2 Blue
$ns at 0.1 "$cbr0 start"
$ns at 0.2 "$cbr1 start"
$ns at 0.5 "$cbr2 start"
$ns at 4.0 "$cbr2 stop"
```

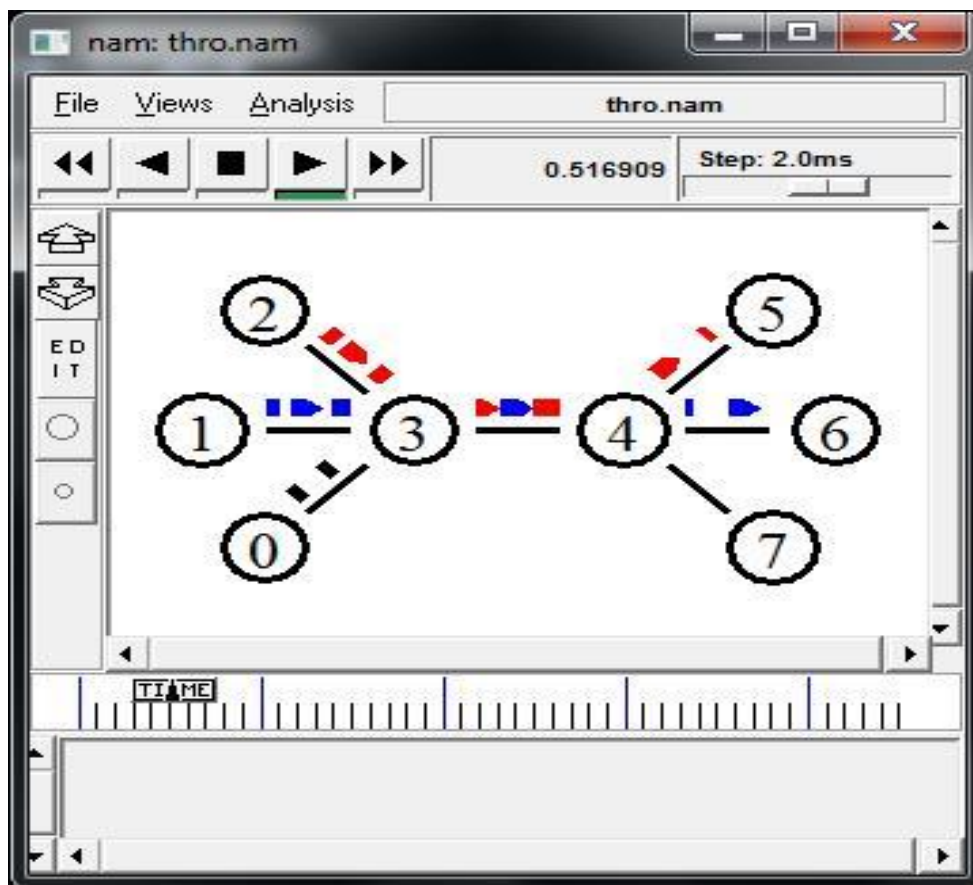

\$ns at 4.2 "\$cbr1 stop"

\$ns at 4.5 "\$cbr0 stop"

\$ns at 5.0 "finish"

\$ns run

OUTPUT:



RESULT:

Thus the congestion control algorithm is simulated by using NS2.

Ex No : 11	IMPLEMENTATION OF LINK STATE ROUTING ALGORITHM
Date :	

AIM:

To simulate and observe traffic route of a network using distance vector routing protocol.

ALGORITHM:

1. Create a simulator object
2. Set routing protocol to Distance vector routing
3. Trace packets on all links on to NAM trace and text trace file.
4. Define finish procedure to close files, flash tracing and run NAM
5. Create 5 nodes
6. Specify the link characteristics between the nodes
7. Describer their layout topology as a octagon
8. Add UDP agent for node n0
9. Create CBR traffic on the top of UDP and set traffic parameters
10. Add NULL agent to node n3
11. Connect source and sink
12. Schedule as follows
 - Start traffic flow at 1.0
 - Down the link n1 – n2 at 15.0
 - Up the link n1 – n2 at 25.0
 - Call finish procedure at 35.0
13. Start the scheduler
14. Observe the traffic route when the link is up and down
15. View the simulated events and trace file analyze it
16. Stop.

PROGRAM:

```
#Create a simulator object

set ns [new Simulator]
```

```

#Define different colors for data flows(for NAM)

$ns color 1 Blue

$ns color 2 Red

#Open the NAM trace file

set nf [open out.nam w]

$ns namtrace-all $nf

#Define a 'finish' procedure

proc finish { } {

    global ns nf

    $ns flush-trace

    #close the NAM trace file

    close $nf

    #Execute NAM on the trace file

    exec nam out.nam &

    exit 0

}

for {set i 0} {$i < 5} {incr i 1} {

    set n($i) [$ns node]}

for {set i 0} {$i < 4} {incr i} {

$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail}


$ns duplex-link $n(0) $n(1) 1Mb 10ms DropTail

$ns duplex-link $n(1) $n(2) 1Mb 10ms DropTail

$ns duplex-link $n(2) $n(3) 1Mb 10ms DropTail

$ns duplex-link $n(3) $n(4) 1Mb 10ms DropTail

$ns duplex-link $n(4) $n(1) 1Mb 10ms DropTail

set udp0 [new Agent/UDP]

$ns attach-agent $n(0) $udp0

```

```

set cbr0 [new Application/Traffic/CBR] $cbr0 set packetSize_ 500 $cbr0 set interval_ 0.005

$scbr0 attach-agent $sudp0

set null0 [new Agent/Null]

$ns attach-agent $n(3) $null0

$ns connect $sudp0 $null0

$ns rtproto DV

$ns rtmodel-at 15.0 down $n(1) $n(2)

$ns rtmodel-at 25.0 up $n(1) $n(2)

$sudp0 set fid_ 1

$ns color 1 Green

$ns at 1.0 "$cbr0 start"

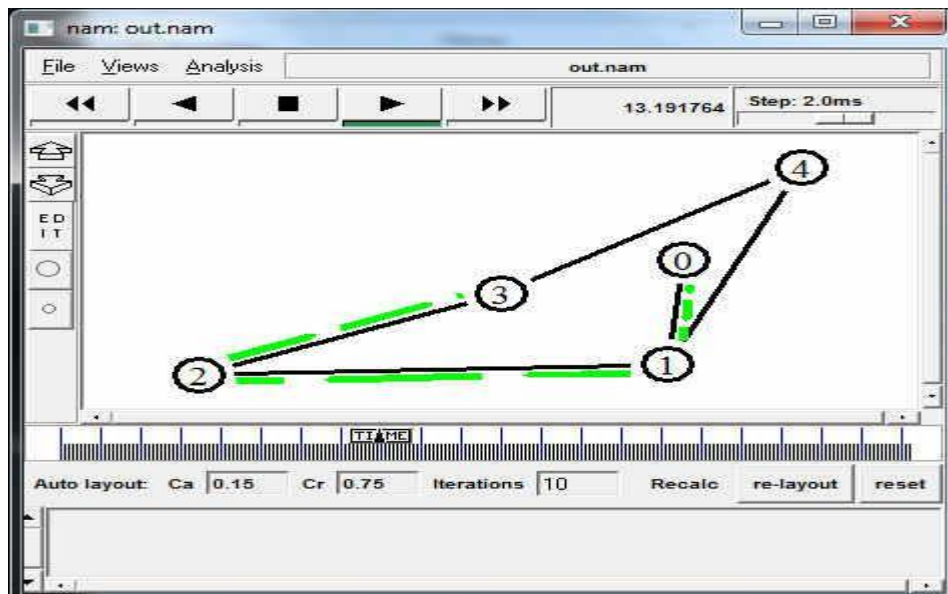
$ns at 35 "finish"

#Run the simulation

$ns run

```

OUTPUT



RESULT:

Thus the Link State Routing algorithm was simulated by using NS2

Ex No : 12	IMPLEMENTATION OF DISTANCE VECTOR ROUTING ALGORITHM
Date :	

AIM:

To simulate and observe traffic route of a network using distance vector routing protocol.

ALGORITHM:

1. Create a simulator object
2. Set routing protocol to Distance vector routing
3. Trace packets on all links on to NAM trace and text trace file.
4. Define finish procedure to close files, flash tracing and run NAM
5. Create 5 nodes
6. Specify the link characteristics between the nodes
7. Describer their layout topology as a octagon
8. Add UDP agent for node n0
9. Create CBR traffic on the top of UDP and set traffic parameters
10. Add NULL agent to node n3
11. Connect source and sink
12. Schedule as follows
 - Start traffic flow at 1.0
 - Down the link n1 – n2 at 15.0
 - Up the link n1 – n2 at 25.0
 - Call finish procedure at 35.0
13. Start the scheduler
14. Observe the traffic route when the link is up and down
15. View the simulated events and trace file analyze it
16. Stop.

PROGRAM:

#Distance vector routing protocol – distvect.tcl

```

#Create a simulator object

set ns [new Simulator]

#Use distance vector routing

$ns rtproto DV

#Open the nam trace file

set nf [open out.nam w]

$ns namtrace-all $nf

# Open tracefile

set nt [open trace.tr w]

$ns trace-all $nt

#Define 'finish' procedure

proc finish {} {

global ns nf

$ns flush-trace

#Close the trace file

close $nf

#Execute nam on the trace file exec nam -a out.nam & exit 0

}


#Create 8 nodes set n1 [$ns node] set n2 [$ns node] set n3 [$ns node] set n4
[$ns node] set n5 [$ns node] set n6 [$ns node] set n7 [$ns node] set n8 [$ns
node]

# Specify link characteristics

$ns duplex-link $n1 $n2 1Mb 10ms DropTail

$ns duplex-link $n2 $n3 1Mb 10ms DropTail

$ns duplex-link $n3 $n4 1Mb 10ms DropTail

$ns duplex-link $n4 $n5 1Mb 10ms DropTail

$ns duplex-link $n5 $n6 1Mb 10ms DropTail

```

```

$ns duplex-link $n6 $n7 1Mb 10ms DropTail
$ns duplex-link $n7 $n8 1Mb 10ms DropTail
$ns duplex-link $n8 $n1 1Mb 10ms DropTail
# specify layout as a octagon
$ns duplex-link-op $n1 $n2 orient left-up
$ns duplex-link-op $n2 $n3 orient up
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n4 $n5 orient right
$ns duplex-link-op $n5 $n6 orient right-down
$ns duplex-link-op $n6 $n7 orient down
$ns duplex-link-op $n7 $n8 orient left-down
$ns duplex-link-op $n8 $n1 orient left
Create a UDP agent and attach it to node n1
set udp0 [new Agent/UDP]
$ns attach-agent $n1 $udp0
#Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
#Create a Null agent (a traffic sink) and attach it to node n4
set null0 [new Agent/Null]
$ns attach-agent $n4 $null0
#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0
#Schedule events for the CBR agent and the network dynamics $ns at
0.0 "$n1 label Source"

$ns at 0.0 "$n4 label Destination"

```

\$ns at 0.5 "\$cbr0 start"

\$ns rtmodel-at 1.0 down \$n3 \$n4

\$ns rtmodel-at 2.0 up \$n3 \$n4

\$ns at 4.5 "\$cbr0 stop"

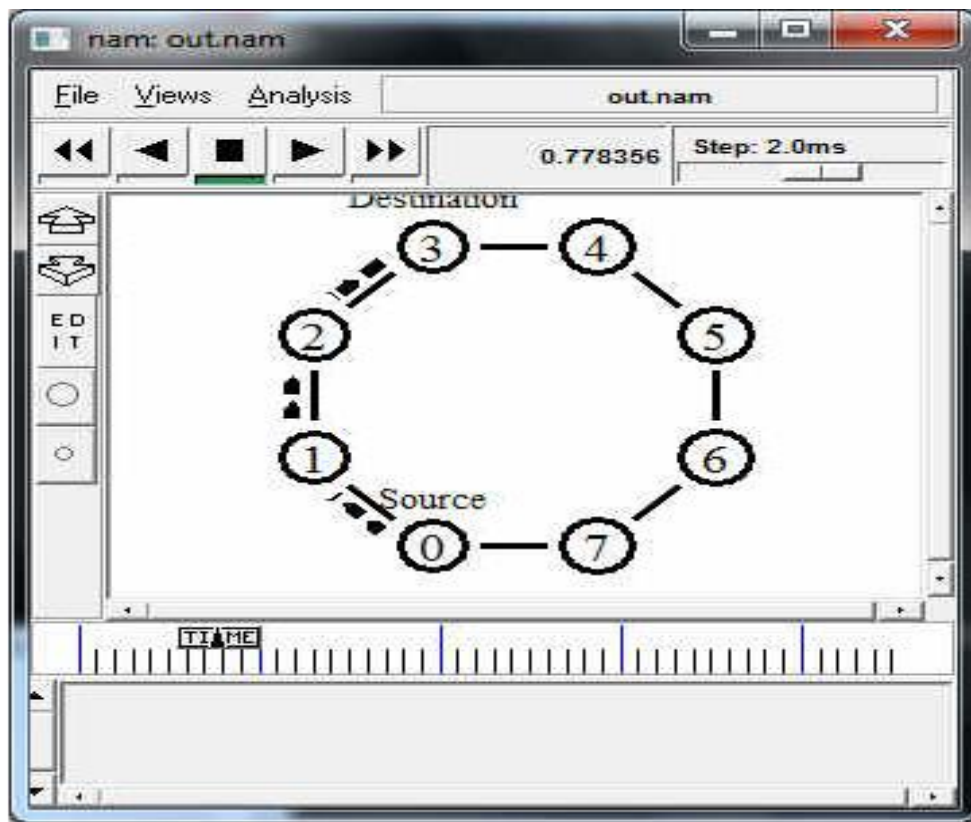
#Call the finish procedure after 5 seconds of simulation time

\$ns at 5.0 "finish"

#Run the simulation

\$ns run

Output:



RESULT:

Thus the Distance Vector Routing algorithm was simulated by using NS2

Ex No : 13	IMPLEMENTATION ERROR CORRECTION CODE ALGORITHM
Date :	

AIM:

To simulate Error Correction code algorithm using JAVA.

ALGORITHM:

1. Start the program
2. Enter the size of the input data.
3. Read the data bit by bit using for loop.
4. Enter the size of data devices.
5. Scan the data bit by bit.
6. Generate the CRC code
7. Find the error and report
8. Stop the program.

PROGRAM:

```
import java.util.*;

class CRC {

    public static void main(String args[]) {

        Scanner scan = new Scanner(System.in);

        int n;

        //Accept the input
        System.out.println("Enter the size of the data:");
        n = scan.nextInt();
        int data[] = new int[n];
        System.out.println("Enter the data, bit by bit:");
        for(int i=0 ; i < n ; i++) {
            System.out.println("Enter bit number " + (n-i) + ":");
            data[i] = scan.nextInt();
        }

        // Accept the divisor
        System.out.println("Enter the size of the divisor:");
        n = scan.nextInt();
        int divisor[] = new int[n];
        System.out.println("Enter the divisor, bit by bit:");
        for(int i=0 ; i < n ; i++) {
            System.out.println("Enter bit number " + (n-i) + ":");
            divisor[i] = scan.nextInt();
        }

        // Divide the inputted data by the inputted divisor
        // Store the remainder that is returned by the method
        int remainder[] = divide(data, divisor);
        for(int i=0 ; i < remainder.length-1 ; i++) {
            System.out.print(remainder[i]);
```

```

    }
    System.out.println("\nThe CRC code generated is:");

    for(int i=0 ; i < data.length ; i++) {
        System.out.print(data[i]);
    }
    for(int i=0 ; i < remainder.length-1 ; i++) {
        System.out.print(remainder[i]);
    }
    System.out.println();

    // Create a new array
    // It will have the remainder generated by the above method appended
    // to the inputted data
    int sent_data[] = new int[data.length + remainder.length - 1];
    System.out.println("Enter the data to be sent:");
    for(int i=0 ; i < sent_data.length ; i++) {
        System.out.println("Enter bit number " + (sent_data.length-i)
                           + ":");

        sent_data[i] = scan.nextInt();
    }
    receive(sent_data, divisor);
}

```

```

static int[] divide(int old_data[], int divisor[]) {
    int remainder[] , i;
    int data[] = new int[old_data.length + divisor.length];
    System.arraycopy(old_data, 0, data, 0, old_data.length);
    // Remainder array stores the remainder
    remainder = new int[divisor.length];
    // Initially, remainder's bits will be set to the data bits
    System.arraycopy(data, 0, remainder, 0, divisor.length);

    // Loop runs for same number of times as number of bits of data

```

```

// This loop will continuously exor the bits of the remainder and
// divisor
for(i=0 ; i < old_data.length ; i++) {
    System.out.println((i+1) + ".) First data bit is : "
                                + remainder[0]);

    System.out.print("Remainder : ");
    if(remainder[0] == 1) {
        // We have to exor the remainder bits with divisor bits
        for(int j=1 ; j < divisor.length ; j++) {
            remainder[j-1] = exor(remainder[j], divisor[j]);
            System.out.print(remainder[j-1]);
        }
    }
    else {
        // We have to exor the remainder bits with 0
        for(int j=1 ; j < divisor.length ; j++) {
            remainder[j-1] = exor(remainder[j], 0);
            System.out.print(remainder[j-1]);
        }
    }
    // The last bit of the remainder will be taken from the data
    // This is the 'carry' taken from the dividend after every step
    // of division
    remainder[divisor.length-1] = data[i+divisor.length];
    System.out.println(remainder[divisor.length-1]);
}
return remainder;
}

static int exor(int a, int b) {
    // This simple function returns the exor of two bits
    if(a == b) {
        return 0;
    }
}

```

```

        return 1;
    }

    static void receive(int data[], int divisor[]) {
        // This is the receiver method
        // It accepts the data and divisor (although the receiver already has
        // the divisor value stored, with no need for the sender to resend it)
        int remainder[] = divide(data, divisor);
        // Division is done
        for(int i=0 ; i < remainder.length ; i++) {
            if(remainder[i] != 0) {
                // If remainder is not zero then there is an error
                System.out.println("There is an error in received data...");
                return;
            }
        }
        //Otherwise there is no error in the received data
        System.out.println("Data was received without any error.");
    }
}

```

Output:

Enter the size of the data:

7

Enter the data, bit by bit:

Enter bit number 7:

1

Enter bit number 6:

0

Enter bit number 5:

0

Enter bit number 4:

1

Enter bit number 3:

1

Enter bit number 2:

0

Enter bit number 1:

1

Enter the size of the divisor:

4

Enter the divisor, bit by bit:

Enter bit number 4:

1

Enter bit number 3:

0

Enter bit number 2:

1

Enter bit number 1:

1

1.) First data bit is: 1

Remainder: 0101

2.) First data bit is: 0

Remainder: 1010

3.) First data bit is: 1

Remainder: 0011

4.) First data bit is: 0

Remainder: 0110

5.) First data bit is: 0

Remainder: 1100

6.) First data bit is: 1

Remainder: 1110

7.) First data bit is: 1

Remainder: 1010

101

The CRC code generated is:

1001101101

Enter the data to be sent:

Enter bit number 10:

1

Enter bit number 9:

0

Enter bit number 8:

0

Enter bit number 7:

1

Enter bit number 6:

1

Enter bit number 5:

0

Enter bit number 4:

1

Enter bit number 3:

1

Enter bit number 2:

0

Enter bit number 1:

1

1.) First data bit is: 1

Remainder: 0101

2.) First data bit is: 0

Remainder: 1010

3.) First data bit is: 1

Remainder: 0011

4.) First data bit is: 0

Remainder: 0111

5.) First data bit is: 0

Remainder: 1110

6.) First data bit is: 1

Remainder: 1011

7.) First data bit is: 1

Remainder: 0000

8.) First data bit is: 0

Remainder: 0000

9.) First data bit is: 0

Remainder: 0000

10.) First data bit is: 0

Remainder: 0000

Data was received without any error.

RESULT:

Thus the Error Correction algorithm (like CRC) was simulated by using JAVA.

Content Beyond Syllabus

EXPT.NO.1	DATA ENCRYPTION AND DECRYPTION
DATE:	

AIM:

To implement Data encryption and decryption

SOFTWARE REQUIREMENTS:

Turbo C software

THEORY:

In **encryption**, each letter position in the file text which is given in encrypt mode is changed according to the ascending order of the key text. In **decryption** each letter position in the encrypted file text is changed according to the ascending order of the key text.

ALGORITHM-ENCRYPTION

Get the text to be encrypted (plain text) and key text.

- a. Find the length of the plain text.
- b. For $i=1$ to length of plain text
- c. Find the binary equivalent of i^{th} character of plain text.
- d. Find the binary equivalent of i^{th} character of key text
- e. Find the XOR of above two values.

The resulting value will be the encrypted format (cipher text) of the plain text.

ALGORITHM-DECRYPTION

Get the text to be decrypted (cipher text) and key text. Find the length of the cipher text.

For $i=1$ to length of cipher text

- a. Find the binary equivalent of i^{th} character of cipher text.
- b. Find the binary equivalent of i^{th} character of key text
- c. Find the XOR of above two values.

The resulting value will be the decrypted format (original plain text) of the cipher plain text.

PROGRAM

```
#include<stdio.>
#include<conio.>
void main ( )
{ static int s, i, k,n,c[100];
printf("\n program 1: encryption and 2.
decryption"); scanf ("%d", &s);
switch (s)
{ case 1: printf("enter the key
            value:"); scanf("%d", &k);
printf("enter the length of
            text:"); scanf("%d",
            &n);
printf("enter the data to be
encrypted:"); for (i=0;i<=n;i++)
scanf("%c",
&c[i]); for
(i=0;i<=n;i++)
{ c[i]=c[i]+k
; if
(c[i]>90)
c[i]=c[i]-
26;}
printf("encrypted
data"); for
(i=1;i<=n;i++)
printf("%c", c[i]);
break;
case 2: printf("enter the key
            value:"); scanf("%d",
            &k);
printf("enter the length of
            text:"); scanf("%d",
```

```

        &n);
printf("enter the data to be
decrypted:"); for (i=0;i<=n;i++)
scanf("%c",
&c[i]); for
(i=0;i<=n;i++)
{ c[i]=c[i]-k;
if (c[i]<65)
c[i]=c[i]+26;
}
printf("decrypted
data"); for
(i=1;i<=n;i++)
printf("%c", c[i]);
break;
case 3:
break; getch
();
}
}

```

OUTPUT:

Program 1: encryption and 2. decryption

1. ENCRYPTIO

N enter the key value:

1 enter the length of

text: 5

enter the data to be encrypted:

HELLO encrypted data : IFMMP

2. DECRYPTIO

N enter the key value:

1 enter the length of

text: 5

enter the data to be decrypted:

IFMMP decrypted data : HELLO

VIVA QUESTIONS:

- 1.What is meant by Encryption?
- 2.What is Decryption?
- 3.Encrypt the word “HELLO” using Caesar cipher.
4. What are the different algorithms available for encryption?
- 5.What type of information can be secured with Cryptography?

RESULT:

Thus the Data Encryption and Decryption was studied.

