```
import java.io.*;
import java.net.*;
public class GossipServer
{
  public static void main(String[] args) throws Exception
{
     ServerSocket sersock = new ServerSocket(3000);
     System.out.println("Server  ready for chatting");
     Socket sock = sersock.accept( );
                    // reading from keyboard (keyRead object)
     BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));
                    // sending to client (pwrite object)
     OutputStream ostream = sock.getOutputStream();
     PrintWriter pwrite = new PrintWriter(ostream, true);
                    // receiving from server ( receiveRead  object)
     InputStream istream = sock.getInputStream();
     BufferedReader receiveRead = new BufferedReader(new InputStreamReader(istream));
     String receiveMessage, sendMessage;
     while(true)
     {
       if((receiveMessage = receiveRead.readLine()) != null)
       {
         System.out.println(receiveMessage);
       }
       sendMessage = keyRead.readLine();
       pwrite.println(sendMessage);
       pwrite.flush();
     }
   }
}
```

```
import java.io.*;
import java.net.*;
public class GossipClient
{
  public static void main(String[] args) throws Exception
  {
    Socket sock = new Socket("127.0.0.1", 3000);
                        // reading from keyboard (keyRead object)
    BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));
                        // sending to client (pwrite object)
    OutputStream ostream = sock.getOutputStream();
    PrintWriter pwrite = new PrintWriter(ostream, true);
                        // receiving from server ( receiveRead  object)
    InputStream istream = sock.getInputStream();
    BufferedReader receiveRead = new BufferedReader(new InputStreamReader(istream));
    System.out.println("Start the chitchat, type and press Enter key");
    String receiveMessage, sendMessage;
    while(true)
    {
      sendMessage = keyRead.readLine();  // keyboard reading
      pwrite.println(sendMessage);       // sending to server
      pwrite.flush();                    // flush the data
      if((receiveMessage = receiveRead.readLine()) != null) //receive from server
      {
         System.out.println(receiveMessage); // displaying at DOS prompt
      }
     }
   }
}
```

```
import java.io.*;
import java.net.*;
public class FileServer {
   public static void main(String[] args) {
      try {
         ServerSocket serverSocket = new ServerSocket(5000);
         System.out.println("Server is listening on port 5000");
         Socket socket = serverSocket.accept();
         System.out.println("Client connected");
         InputStream inputStream = socket.getInputStream();
         FileOutputStream fileOutputStream = new FileOutputStream("received_file.txt");
         byte[] buffer = new byte[1024];
         int bytesRead;
         while ((bytesRead = inputStream.read(buffer)) != -1) {
            for (int i = 0; i < bytesRead; i++) {
               byte cipherByte = buffer[i];
               byte plainByte = (byte) ((cipherByte - 3 + 256) % 256);
               System.out.println("Cipher Text: " + cipherByte + " (" + (char) cipherByte + ") -> Plain Text:
" + plainByte + " (" + (char) plainByte + ")");
               buffer[i] = plainByte;
            }
            fileOutputStream.write(buffer, 0, bytesRead);
         }
        fileOutputStream.close();
         inputStream.close();
         socket.close();
         serverSocket.close();
         System.out.println("File received and decrypted successfully");
      } catch (IOException ex) {
         ex.printStackTrace();
      } } }
```

```
import java.io.*;
import java.net.*;
public class FileClient {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("localhost", 5000);
            System.out.println("Connected to server");
            FileInputStream fileInputStream = new FileInputStream("file_to_send.txt");
            OutputStream outputStream = socket.getOutputStream();
            byte[] buffer = new byte[1024];
            int bytesRead;
            while ((bytesRead = fileInputStream.read(buffer)) != -1) {
                for (int i = 0; i < bytesRead; i++) {
                    byte plainByte = buffer[i];
                    byte cipherByte = (byte) ((plainByte + 3) % 256);
                    System.out.println("Plain Text: " + plainByte + " (" + (char) plainByte + ") -> Cipher Text: " +
cipherByte + " (" + (char) cipherByte + ")");
                    buffer[i] = cipherByte;
                }
                outputStream.write(buffer, 0, bytesRead);
            }
            fileInputStream.close();
            outputStream.close();
            socket.close();
            System.out.println("File encrypted and sent successfully");
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

```
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface Calculator extends Remote
{
  public long add(long a,long b)throws RemoteException;
}
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class CalculatorImpl extends UnicastRemoteObject implements Calculator
{
        protected CalculatorImpl()throws RemoteException
        {
        super();
        }
        public long add(long a,long b)throws RemoteException
        {
                return a+b;
        }
}
```

```
import java.rmi.Naming;
public class CalculatorServer
{
CalculatorServer()
{
try
{
Calculator c=new CalculatorImpl();
Naming.rebind("rmi://127.0.0.1:1099/CalculatorService",c);
}
catch(Exception e)
{
e.printStackTrace();
}
}
public static void main(String[] args)
{
new CalculatorServer();
}
}
```

```
import java.rmi.Naming;
public class CalculatorClient
{
	public static void main(String[] args)
	{
		try
		{
			Calculator
c=(Calculator)Naming.lookup("//127.0.0.1:1099/CalculatorService");
			System.out.println("addition:"+c.add(10,15));
		}
		catch(Exception e)
		{
			System.out.println(e);
		}
	}
}
```

```
#create a simulator object
set ns [new Simulator]
#create a trace file, this file is for logging purpose
set tracefile [open wired.tr w]
$ns trace-all $tracefile
#create a animation infomration or NAM file creation
set namfile [open wired.nam w]
$ns namtrace-all $namfile
#create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
#creation of link between nodes with DropTail Queue
#Droptail means Dropping the tail.
$ns duplex-link $n0 $n1 5Mb 2ms DropTail
$ns duplex-link $n2 $n1 10Mb 5ms DropTail
$ns duplex-link $n1 $n4 3Mb 10ms DropTail
$ns duplex-link $n4 $n3 100Mb 2ms DropTail
$ns duplex-link $n4 $n5 4Mb 10ms DropTail
#creation of Agents
#node 0 to Node 3
set udp [new Agent/UDP]
set null [new Agent/Null]
$ns attach-agent $n0 $udp
$ns attach-agent $n3 $null
$ns connect $udp $null
#creation of TCP Agent
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
```

```
$ns attach-agent $n2 $tcp
$ns attach-agent $n5 $sink
$ns connect $tcp $sink
#creation of Application CBR, FTP
#CBR - Constant Bit Rate (Example nmp3 files that have a CBR or 192kbps, 320kbps, etc.)
#FTP - File Transfer Protocol (Ex: To download a file from a network)
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
set ftp [new Application/FTP]
$ftp attach-agent $tcp
#Start the traffic
$ns at 1.0 "$cbr start"
$ns at 2.0 "$ftp start"
$ns at 10.0 "finish"
#the following procedure will be called at 10.0 seconds
proc finish {} {
 global ns tracefile namfile
 $ns flush-trace
exec nam wired.nam &
 close $tracefile
 close $namfile
 exit 0
}
puts "Simulation is starting..."
$ns run
```

```
BEGIN{
send=0;
received=0;
dropped=0;
start=1.0;
stop=3.0;
}
{
if($1=="/+/")
{
send++;
}
if($5=="tcp")
{
if($1=="r")
{
received++;
}
}
if($1=="d"){
dropped++;
}
}
END{
if(send=="0" && received=="0")
{
print "empty trace file\t"
}
print "Number of Packets Received  " received
print "Throughput =" (received*8)/(start-stop) "bits per second"
print "Number of Packets Dropped = " dropped
}
```