A Novel Design of Reliable File Transfer Application based on UDP using JAVA Socket Programming

Submitted by

MD MUKIB MOSLEH (113-37-1097)

NIKHIL PRATHAPANI (113-66-0344)

(GROUP#7)

Project Report submitted to the Faculty of the Graduate School of the

University of Maryland, College Park in partial fulfilment

of the requirements for the course of

Networks & Protocols I
(ENTS 640)

Course Instructor:

Dr. Zoltan Safar,
Director, Master's in Telecommunications Program

Date of Submission: 1 Dec, 2014

TABLE OF CONTENTS

I. PROBLEM STATEMENT

The purpose of this project is to create a client-server communication link using Java's UDP sockets. Upon receiving a request from the client, the server will serve the requested text files to the client using UDP's unreliable data transfer services. With UDP's connectionless data transfer protocol, there is no handshaking between sender and receiver and hence there's no guarantee whether the data has arrived or is intact. However, due to a minimum of overhead in comparison with TCP, a much faster data transmission is possible over high quality physical links. In this project, the reliability of the communication link has to be ensured by adding extra features in the protocol, e.g. integrity check, timeout and, if needed, retransmission of packets according to the user selection.

The protocol requires a client and server set up using UDP socket parameters to facilitate a successful data transfer link. The sample texts files are saved in a defined directory on the server machine. The messages sent are all text-oriented and consist of human readable character sequences and hence are assembled as String Object before transmission. The required protocols are ensured in both server and client machine to ensure reliability and functionality of the link. While communicating, both server and client send a message of defined and fixed character sequence for server and client respectively.

The client request message consists of protocol name & version, type of message, name and format of the requested file in a given format. The whole message is then processed according to the given algorithm for deriving an integrity check number. The integrity check value is also a part of user's request message which is sent to the server. The integrity check field is then recalculated and matched to ensure theirs is no error or irregularity during transmission at server's end. In case of anomaly, the whole process restarts for and prompts user either to re-send the request or to request the file again.

Server message includes a response code which denotes the state of the message received so that the client can respond accordingly. Server checks for integrity, protocol name and version, correct file name and format etc. to generate the correct response code. If everything is okay, the server reads the requested text file from its given directory and includes it in the response message. The response message with a response code 0 including the requested content and the integrity check value is converted into a byte array and sent as a response back to the client. The server goes back to step 1 and wait for another session.

The design of the program is then described in this report with necessary flowcharts, and UML class diagrams. The solution design segments are discussed in details with various output scenarios and correct functionalities.

II. ALGORITHM

II.A. Algorithm for Client Application:

Step 1: The application prompts the user to select one of the available files that he needs from the server.

If there is an invalid selection (non-existent file is requested), response code 3 is displayed and the same step 1 is repeated.

Step 2: The request message is created with 3 fields, "ENTS/1.0 Request", "filename.extension", and the integrity check value of the aforementioned fields.(**ENTS/1.0 Request [CR+LF] filename.extension [CR+LF] integrity check [CR+LF]**)

Step 3: The client application send the request message to server after converting the request message into a byte array.

Step 4: After sending the request message, the application waits for response message from the server. The first timeout is scheduled after 1 second. After the first timeout event, the request is re-sent to the server. There are totally four timeout events, and the timeout value doubles after every timeout event (1sec, 2sec, 4sec, and 8sec) and request message is resent accordingly.

Step 4A: If there is no response from the server after the 4^{th} timeout, the application prompts back the user to select one of the available files that he needs from the server.

Step 4B: If there is response from the server before the timer expires, the application verifies the integrity check value of the response message.

Step 4B (i): If the Integrity check value doesn't match, it prompts the user to resend the request again. If the user denies to resend the request, the application prompts the user to select one of the available files that he needs from the server. (Here the Response code 1 is displayed)

Step 4B (ii):

a. If no errors have occurred and the Integrity check value matches, Response code 0 is displayed along with the characters of the requested file.
b. If there is problem with the "filename" or "extension" fields, Response code 2 is displayed.
c. If the requested file doesn't exist, Response code 3 is displayed.
d. If the protocol version is wrong (i.e. other than 1.0), Response code 4 is displayed.

Step 5: The application goes back to step 1 and runs continuously until it is terminated.

II.B. Algorithm for Server Application:

Step 1: The Server application waits for requests on the specified port.

Step 2: If a message is received, the server checks the integrity check value of request message (neglecting the appended integrity check string at the end of request message) and compares it with the appended integrity check string at the end of request message.

Step 3:

Step 3A: If the integrity check fails, the server sends a response message with response code 1 and without any file content. (**ENTS/1.0 Response [CR+LF] response code 1 [CR+LF] content length [CR+LF] integritycheck [CR+LF]**)

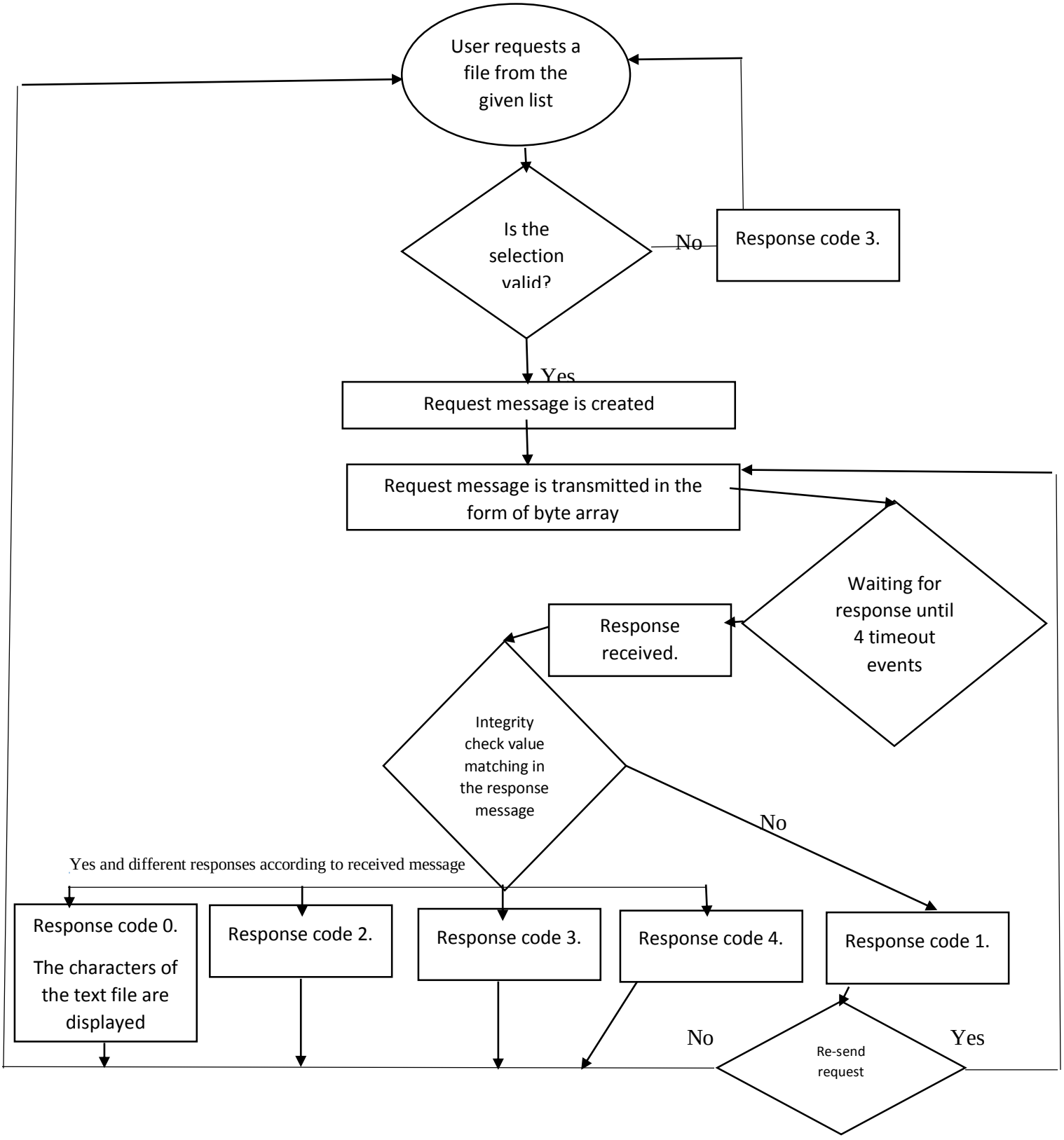
Step 3B: If the integrity check value matches, the following approach is adopted,

a. The server checks if the "filename" and "extension" are syntactically correct. If no, response code 2 is sent without any file content. (**ENTS/1.0 Response [CR+LF] response code 2 [CR+LF] content length [CR+LF] integritycheck [CR+LF]**)
   If yes, go to following step.
b. The server checks the version number. If it is not equal to 1.0, response code 4 is sent without any file content. (**ENTS/1.0 Response [CR+LF] response code 4 [CR+LF] content length [CR+LF] integritycheck [CR+LF]**)
   If it is equal to 1.0, it goes to following step.
c. The server checks the file in the given path. If the file does not exist, response code 3 is sent without any file content. (**ENTS/1.0 Response [CR+LF] response code 3 [CR+LF] content length [CR+LF] integritycheck [CR+LF]**)
   If the requested file exists, the server opens the requested text file and reads its content.
d. After reading the content, it prepares a response message (**ENTS/1.0 Response [CR+LF] response code 0 [CR+LF] content length [CR+LF] content integritycheck [CR+LF]**)

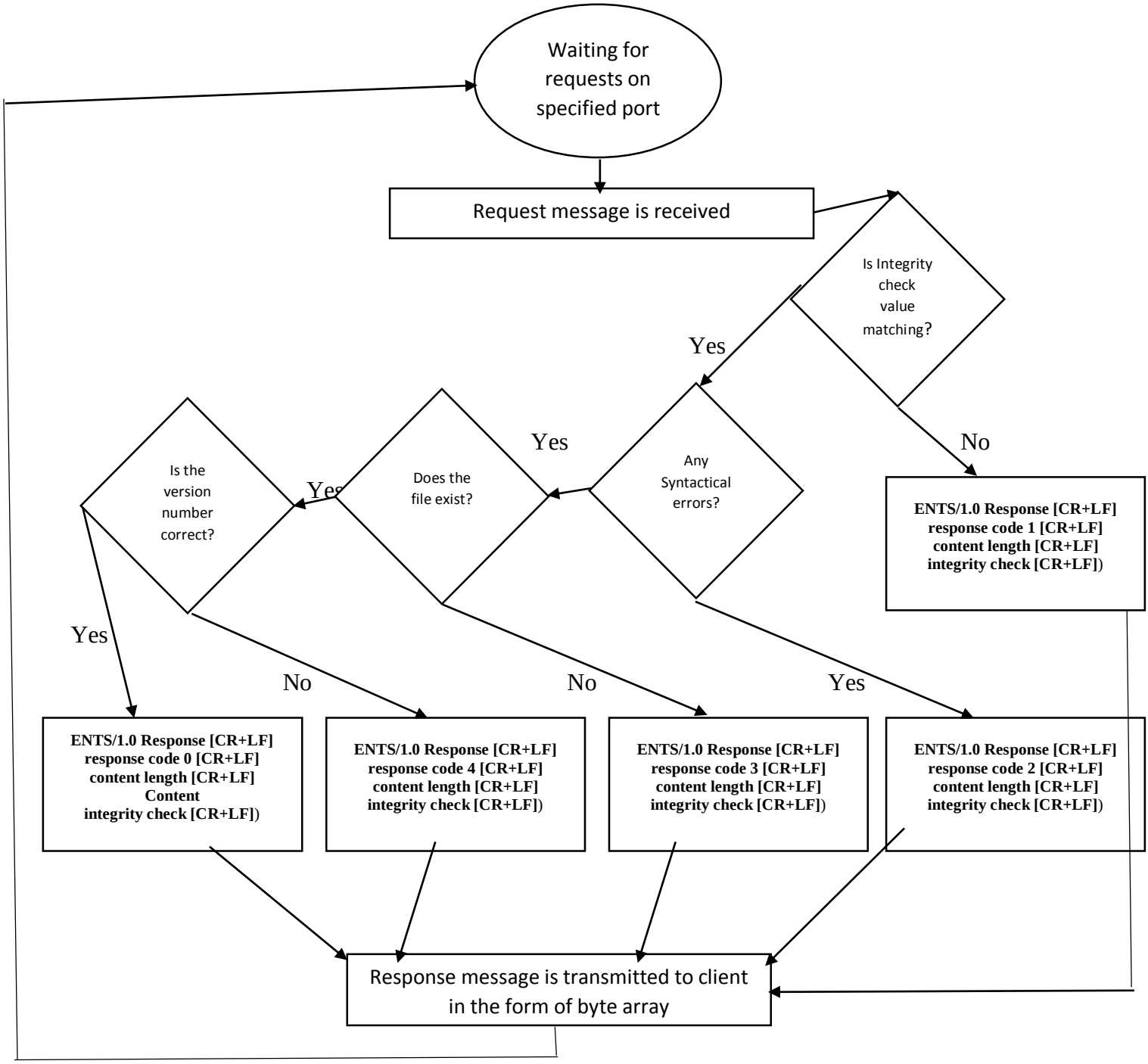
Step 4: The application goes back to step 1 and runs continuously until it is terminated.

III. FLOW CHARTS

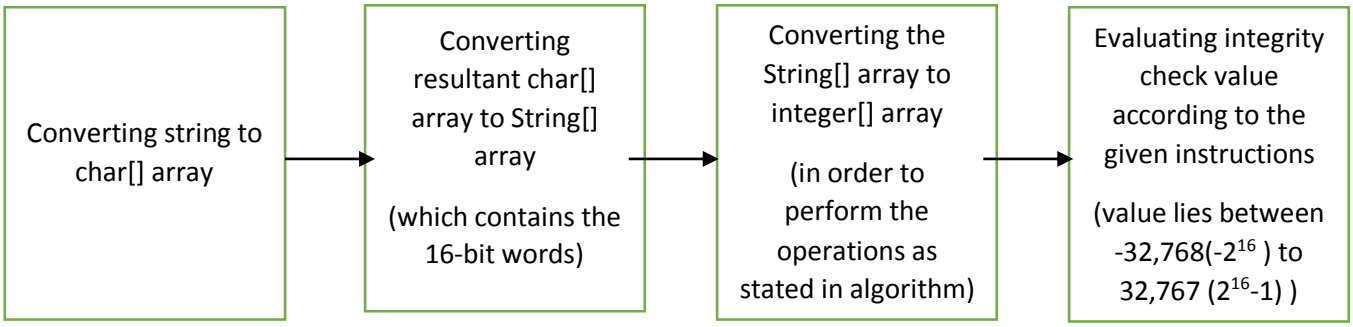
III.A. Flowchart for Client functionality



III.B. Flowchart for Server functionality



IV. APPROACH TO CALCULATE INTEGRITY CHECK



V. BLOCK DIAGRAMS

V.A. Block diagram depicting the algebraic approach adopted for Request Message

ENTS/1.0 Request[CR+LF]	--DELIM[0]
[filename].[extension][CR+LF]	--DELIM[1]
[integrity check][CR+LF]	--DELIM[2]

LENGTH(REQUEST MESSAGE) = LENGTH(DELIM[0])+LENGTH(DELIM[1])+LENGTH(DELIM[2])+ 6

NOTE: 6 CORRESPONDS TO THE 3 CR+LF PAIRS

V.B. Block diagram depicting the algebraic approach adopted for Response Message

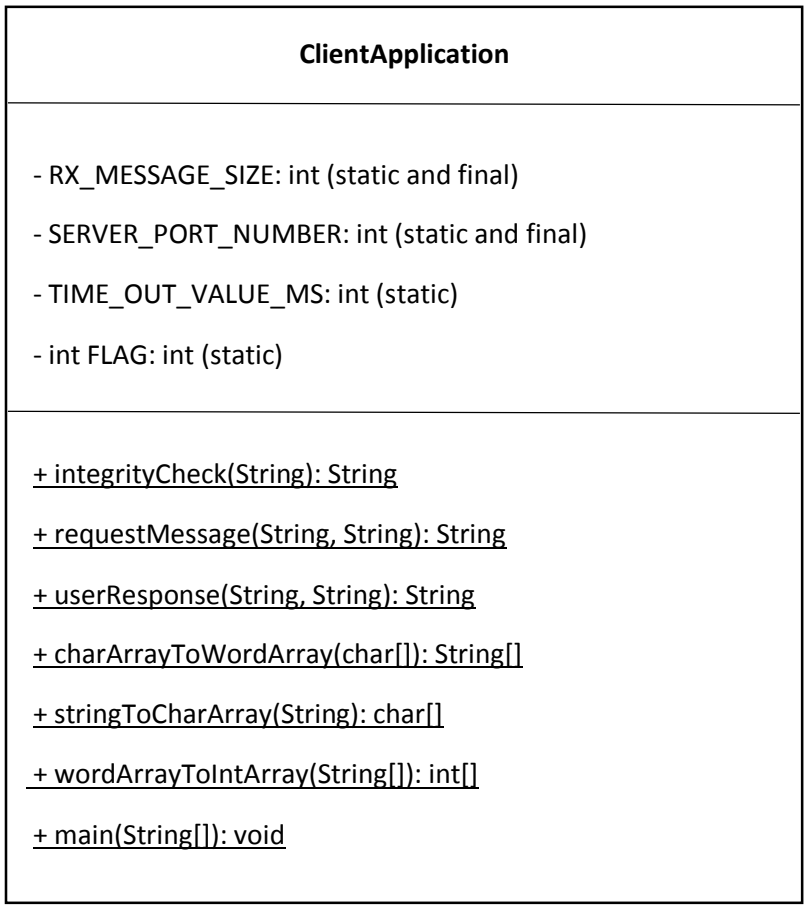
ENTS/1.0 Response[CR+LF]	-- DELIM[0]
[response code][CR+LF]	--DELIM[1]
[content length][CR+LF]	--DELIM[2]
[content – if any]	number of characters in this field=(int)DELIM[2]
[integrity check][CR+LF]	to extract this value, we have used the following approach: length(received text)-length(DELIM[0])-length(DELIM[1])-length(DELIM[2])-(int) DELIM[2] we have used substring() to extract this value

length(received text)= length(DELIM[0])+length(DELIM[1])+length(DELIM[2])+(INT) DELIM[2]+LENGTH(INTEGRITY CHECK FIELD) +8

NOTE: 8 CORRESPONDS TO THE 4 CR+LF PAIRS

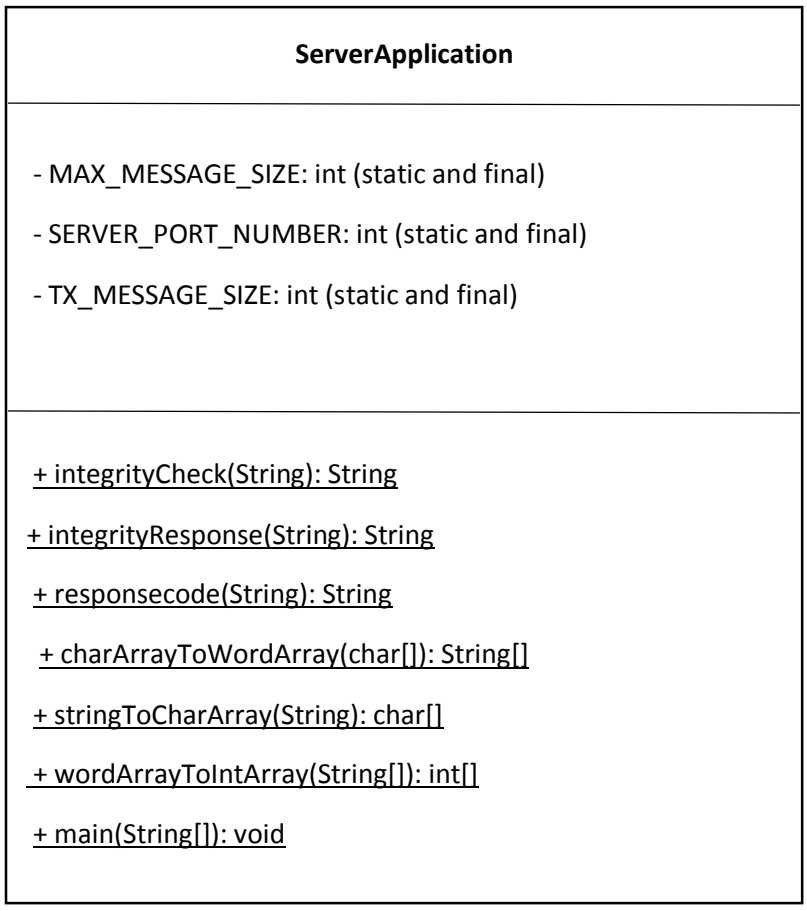
VI. UML CLASS DIAGRAMS

VI.A. UML diagram for client



Note: underlined attributes represent static methods. '+' denotes public and '-' denotes private attribute.

VI.B. UML diagram for server



Note: underlined attributes represent static methods. '+' denotes public and '-' denotes private attribute.

VII. OUTPUT

Case 1: When the server responds before the timer expires

Client Side:

```
The available files in the server are:
A. directors_message.txt
B. program_overview.txt
C. scholarly_paper.txt
Please enter the text file which you want to access from server: A/B/C
A
value of Integrity Check is: -904
The Request message sent to the server is:
ENTS/1.0 Request
directors_message.txt
-904

trial# 1: waiting for server's response

The received text received from the server is:
ENTS/1.0 Response
0
1562
Thank you for your interest in the Master’s in Telecommunications Program at the University of
Maryland!

Our program offers a unique opportunity to engage in cross-disciplinary coursework from the
Department of Electrical and Computer Engineering and the Robert H. Smith School of Business. This
exceptional combination allows students to enhance their technical background and gain the
necessary business and management skills to advance their careers.

Our mission is to provide industry-oriented graduate education in telecommunications that produces
graduates who are business-minded, possess the needed technical skills and are well versed in the
latest technology trends. We offer a wide range of interesting and relevant technical courses in
wireless communications, computer networking and cyber-security. Our thought-provoking business
and entrepreneurship courses expose our students to economics, finance, marketing, and management.

The Master of Science degree in Telecommunications is very well respected in the industry. Our
recent graduates have landed jobs at companies like Cisco, Juniper Networks, Amazon, Qualcomm,
Hughes Network Systems, and Intel. After working in the industry for a few years, many of our
alumni advance to engineering managers, product managers, and program managers. There are even
directors and vice presidents among our alumni.

I invite you to explore our website or contact us directly at telecomprogram@umd.edu with any
questions you may have. We look forward to hearing from you!

Zoltan Safar, Director-29652

value of Integrity Check is: -29652

OK. The response has been created according to the request.

The characters of the received file are:
Thank you for your interest in the Master’s in Telecommunications Program at the University of
Maryland!

Our program offers a unique opportunity to engage in cross-disciplinary coursework from the
Department of Electrical and Computer Engineering and the Robert H. Smith School of Business. This
```

exceptional combination allows students to enhance their technical background and gain the necessary business and management skills to advance their careers.

Our mission is to provide industry-oriented graduate education in telecommunications that produces graduates who are business-minded, possess the needed technical skills and are well versed in the latest technology trends. We offer a wide range of interesting and relevant technical courses in wireless communications, computer networking and cyber-security. Our thought-provoking business and entrepreneurship courses expose our students to economics, finance, marketing, and management.

The Master of Science degree in Telecommunications is very well respected in the industry. Our recent graduates have landed jobs at companies like Cisco, Juniper Networks, Amazon, Qualcomm, Hughes Network Systems, and Intel. After working in the industry for a few years, many of our alumni advance to engineering managers, product managers, and program managers. There are even directors and vice presidents among our alumni.

I invite you to explore our website or contact us directly at telecomprogram@umd.edu with any questions you may have. We look forward to hearing from you!

Zoltan Safar, Director

The available files in the server are:
A. directors_message.txt
B. program_overview.txt
C. scholarly_paper.txt
Please enter the text file which you want to access from server: A/B/C

Server Side:

Waiting for a message from the client.
Message received from the client.
The received text from the client is:
ENTS/1.0 Request
directors_message.txt
-904

Value of Integrity Check is: -904
Value of Integrity Check is: -29652
The response message that is being sent back to client is:
ENTS/1.0 Response
0
1562
Thank you for your interest in the Master’s in Telecommunications Program at the University of Maryland!

Our program offers a unique opportunity to engage in cross-disciplinary coursework from the Department of Electrical and Computer Engineering and the Robert H. Smith School of Business. This exceptional combination allows students to enhance their technical background and gain the necessary business and management skills to advance their careers.

Our mission is to provide industry-oriented graduate education in telecommunications that produces graduates who are business-minded, possess the needed technical skills and are well versed in the latest technology trends. We offer a wide range of interesting and relevant technical courses in wireless communications, computer networking and cyber-security. Our thought-provoking business and entrepreneurship courses expose our students to economics, finance, marketing, and management.

```
The Master of Science degree in Telecommunications is very well respected in the industry. Our
recent graduates have landed jobs at companies like Cisco, Juniper Networks, Amazon, Qualcomm,
Hughes Network Systems, and Intel. After working in the industry for a few years, many of our
alumni advance to engineering managers, product managers, and program managers. There are even
directors and vice presidents among our alumni.

I invite you to explore our website or contact us directly at telecomprogram@umd.edu with any
questions you may have. We look forward to hearing from you!

Zoltan Safar, Director-29652

Sending an OK response to the client.
-----------------------------------------


Waiting for a message from the client.
```

Case 2: When the server does not respond and the timer expires

Client side:

```
The available files in the server are:
A. directors_message.txt
B. program_overview.txt
C. scholarly_paper.txt
Please enter the text file which you want to access from server: A/B/C
A
value of Integrity Check is: -904
The Request message sent to the server is:
ENTS/1.0 Request
directors_message.txt
-904

trial# 1: waiting for server's response
trial# 2: waiting for server's response
trial# 3: waiting for server's response
trial# 4: waiting for server's response

 client socket timeout! Exception object e:java.net.SocketTimeoutException: Receive timed out
The available files in the server are:
A. directors_message.txt
B. program_overview.txt
C. scholarly_paper.txt
Please enter the text file which you want to access from server: A/B/C
```

Server side: --nothing to display, as the server is not running and the clock expires—

Case 3: When the user selects non-existent file

Client side:

```
The available files in the server are:
A. directors_message.txt
B. program_overview.txt
C. scholarly_paper.txt
Please enter the text file which you want to access from server: A/B/C
x
-----------------------------------------
value of Integrity Check is: 3968
The Request message sent to the server is:
ENTS/1.0 Request
default.txt
3968

trial# 1: waiting for server's response

The received text received from the server is:
ENTS/1.0 Response
3
0
31039


Error: non-existent file. The file with the requested name does not exist.

The available files in the server are:
A. directors_message.txt
B. program_overview.txt
C. scholarly_paper.txt
Please enter the text file which you want to access from server: A/B/C
```

Server side:

```
Waiting for a message from the client.
Message received from the client.
The received text from the client is:
ENTS/1.0 Request
default.txt
3968

Value of Integrity Check is: 3968
Value of Integrity Check is: 31039
ENTS/1.0 Response
3
0
31039

The length of sent string is: 32
Sending an OK response to the client.
-----------------------------------------


Waiting for a message from the client.
```

APPENDIX A: JAVA CODE FOR CLIENT APPLICATION

```
package trials;


import java.io.InterruptedIOException;// timeout exception type
import java.net.*;// networking classes
import java.util.*;//scanner and other classes in java.util package
import java.io.*;//java.io classes

public class ClientApplication
{
	// should be the same port as the one the
	// server is listening on!
	private final static int SERVER_PORT_NUMBER = 9999;
	// received message size
	private final static int RX_MESSAGE_SIZE=8192;
	//this variable will be used for 4 timeout events iteratively
	private static int FLAG=4;
	//initial timeout value in milliseconds
	private static int TIME_OUT_VALUE_MS = 1000;

	public static void main(String[] args) throws Exception
	{
		//this should run forever unless it is terminated
		while(true)
		{
			// for console input
			Scanner inp=new Scanner(System.in);

			System.out.println("The available files in the server are:");
			System.out.println("A. directors_message.txt");
			System.out.println("B. program_overview.txt");
			System.out.println("C. scholarly_paper.txt");
			System.out.println("Please enter the text file which you want to access from
server: A/B/C");


			String req_file=null;;
			String inp1=inp.nextLine();
			/*FORMING THE REQUEST MESSAGE BY CALLING THE METHOD requestMessage
			the request message contains protocol and its version, filename and its
			extension,           and integrity check value of its first two fields
			all the fields are separated by CR+LF */
			String req_msg=requestMessage(inp1,req_file);

			// converting the request message into a byte array
			byte[] req_msgBytes=req_msg.getBytes();

			/*System.out.println("The sent byte sequence is:");
			for(int ind=0;ind<req_msgBytes.length;ind++)
			{
				System.out.printf("%d,",req_msgBytes[ind]);
			}*/
```

```
                // creating the IP address object for the server machine
                // method of looping back the request to the same machine
                InetAddress serverIp= InetAddress.getLocalHost();
                // creating the UDP packed to be sent
                DatagramPacket sentPacket=new
DatagramPacket(req_msgBytes,req_msgBytes.length,serverIp,SERVER_PORT_NUMBER);

                // creating the UDP client socket
                DatagramSocket clientSocket=new DatagramSocket();

                // sending the UDP packet to the server
                clientSocket.send(sentPacket);

                while(FLAG>0)
                {
                        // setting the timeout for the socket
                        clientSocket.setSoTimeout(TIME_OUT_VALUE_MS);

                        // receiving the server's response
                        try
                        {
                        /*printing the trial number of timeout-we totally have 4 timeout events
                                * 1000ms,2000ms,4000ms and 8000ms */
                System.out.print("trial# "+(5-FLAG) +": waiting for server's response\n");

                                // creating the receive UDP packet
                                byte[] receivedBytes = new byte[RX_MESSAGE_SIZE];
        DatagramPacket receivedPacket=new DatagramPacket(receivedBytes, RX_MESSAGE_SIZE);
                                clientSocket.receive(receivedPacket);
                                // the timeout timer starts running here
                                // the receive() method blocks here (program execution stops)
                                // only two ways to continue:
                                // a) packet is received (normal execution after the catch block)
                                // b) timer expires after 4 timeout events (exception is thrown)

        InetAddress serverAddress = receivedPacket.getAddress(); // server's IP address
                int serverPort = receivedPacket.getPort(); // server's port number
                int dataLength = receivedPacket.getLength(); // the number of received bytes

                                /*printing the received byte sequence if needed
                                 * System.out.println("\nThe received byte sequence is: ");
                                for(int ind = 0; ind < dataLength; ind++)
                                {
                                        System.out.printf("%d", receivedBytes[ind]);
                                        if(ind < dataLength-1)
                                                System.out.print(", ");
                                }*/

                                // converting the received byte array into String
                                String receivedText = new String(receivedBytes, 0, dataLength);
                        System.out.println("\nThe received text received from the server is: ");
                                System.out.println(receivedText);

                                /*splitting the received string using CR+LF par
                                 * although individual lines are also broken in text file,
```

```
                                        * we are using the length of response code and response code
                                        later in the code
                                         *  to extract the contents of the file*/
                                        String[] delim = receivedText.split("\r\n");


                                        /*the response code is broken using CR+LF,
                                         *  the first part of string is response message,
                                         *  second part is response code either 0/1/2/3/4*/
                                        /*the later parts contain the contents of the file and integrity
                                        check value */
                                        if(Integer.parseInt(delim[1])==0)
                                        {
                                              /*the delim5 string only contains the integrity check
                                              value of response message.
                                               * because we are clipping the first 4 parts
                                              *  .i.e. ENTS/1.0 Response,response code,content length,
                                              contents of the text file.
                                               * additionally six is added to substring method because,
                                              it denotes 3 CR and LF pairs accompanied by the first 3
                                              parts.
                                               * "Integer.parseInt(delim[2],10)" denotes the content
                                              length of the text file*/
String
delim5=receivedText.substring(delim[0].length()+delim[1].length()+delim[2].length()+Integer.parseI
nt(delim[2],10)+6);

                                        /*the dlim6 contains the characters of the selected text file.
                                               *  here we are selecting the substring accordingly.
                                               *  ignoring the ENTS/1.0 Response,response code,content
                                              length, and selecting upto end of text file characters.
                                               *   and we are also ignoring the integrity check value
                                              which exists in the received text*/
String
delim6=receivedText.substring(delim[0].length()+delim[1].length()+delim[2].length()+6,delim[0].len
gth()+delim[1].length()+delim[2].length()+Integer.parseInt(delim[2],10)+6);

                                              /*calculating the integrity check value for received text,
                                               *  by clipping the numeric integrity check value added to
                                              received text
                                               *   and we are calculating the value only until the
                                               contents of file
                                               *   i.e. we are calculating integrity check value for
                                         *   ENTS/1.0 Response,response code,content length, and content*/
String
newChecksum1=integrityCheck(receivedText.substring(0,delim[0].length()+delim[1].length()+delim[2].
length()+Integer.parseInt(delim[2],10)+6));

                                              /* checking if the transmission was OK
                                               *  .i.e. integrity check value that we calculated is same
                                               *   as the value that we obtained at the end of received text*/
                                        if(newChecksum1.equals(delim5.substring(0, delim5.length()-2)))
                                              {
                                                    /*Response code 0: the characters of the received
                                                    file are displayed*/
                  System.out.println("\nOK. The response has been created according to the request.");
                  System.out.println("\nThe characters of the received file are: ");
```

```
            System.out.println(delim6+"\n");
            System.out.println("-----------------------------------------");

            break;
                                    }
                                    else
                                    {

                                          /*As integrity check failure occured,
                                           *  we are prompting the user whether he likes to
                                          resend the request */
System.out.print("\nError: integrity check failure. The request has one or more bit errors.");
                              System.out.println("Would you like to resend the request: Y/N");
                                    String sel=inp.nextLine();
                                          if(sel.equals("Y"))
                                          {
                                                /*if user likes to resend the request,
                                                * we are sending the packet again and the
                                                process is repeated accordingly*/
                                                clientSocket.send(sentPacket);
                                          }
                                          else
                                          {
                                                /*as the user is not willing to resend the
                                                request,
                                                 * we are prompting him to main menu to
                                                 select the file
                                                 *  that he needs to access from the server*/
      System.out.print("prompting to the main menu after integrity check failure\n\n");

                                                break;
                                          }
                                    }

                                }
                                else if(Integer.parseInt(delim[1])==2)
                                {
                                      /*Response code 2: syntax of the request message is incorrect*/
                                      System.out.print("\nError: malformed request. The syntax
of the request message is not correct.\n\n");

                                      break;
                                }
                                else if(Integer.parseInt(delim[1])==3)
                                {
                                      /*response code 3: non-existent file requested by the user*/
                                      System.out.print("\nError: non-existent file. The file
with the requested name does not exist.\n\n");

                                      break;
                                }
                                else if(Integer.parseInt(delim[1])==4)
                                {
                                /*wrong protocol version (anything other than 1.0) entered by user*/
                                      System.out.print("\nError: wrong protocol version. The
version number in the request is different from 1.0.\n\n");
```

```
                                    break;
                                }

                            }//try
                            catch(InterruptedIOException e)
                            {

                            /*doubling timer value for every timeout event until 4 timeout events*/
                                TIME_OUT_VALUE_MS=2*TIME_OUT_VALUE_MS;
                                FLAG--;
                            /*sending the packet to server again after every timeout event
                                until 4 timeouts*/
                                clientSocket.send(sentPacket);
                                if(FLAG==0)
                                {
// timeout - timer expired (after 4 timeout events) before receiving the response from the server
                                    System.out.println("\n client socket timeout! Exception
object e:"+e);

                                }
                            }//catch
                        }//while(flag)
                    }//while(true)
            }//main

            public static String requestMessage(String a,String b) throws Exception,IOException
            {

                    /*calling userResponse() method with user selection string :
                     * A/B/C and requested file name as parameters*/
                    String s1=userResponse(a,b);
                    String s2=(("ENTS/1.0 Request").concat("\r\n")+s1).concat("\r\n"));
                    String newChecksum=integrityCheck(s2);
                    String s3=s2+newChecksum+"\r\n";
                    System.out.println("The Request message sent to the server is: ");
                    System.out.println(s3);
                    return s3;
            }//requestMessage() method
            public static String userResponse(String a,String b)
            {
                    switch (a)
                    {
                            case "A":b="directors_message.txt";
                                    break;
                            case "B":b="program_overview.txt";
                                    break;
                            case "C":b="scholarly_paper.txt";
                                    break;
                            default:b="default.txt";
                            System.out.println("-----------------------------------------");
                                    break;
                    }
                    return b;
            }//userResponse() method
            public static String integrityCheck(String a)
            {
```

```
		/*converting string to char array*/
		char[] b=stringToCharArray(a);
		/*converting resultant char array to string/word array*/
		String[] c=charArrayToWordArray(b);
		/*converting the word array to integer array*/
		int[] d=wordArrayToIntArray(c);
		/*evaluating integrity check value according to the given instructions*/
		short S=0;
		for(int i=0;i<d.length-1;i++)
		{
			short index= (short)(S^d[i]);
			//System.out.println("index is"+index);
			S=(short)((7919*index)%65536);
			//System.out.println("S is"+S);
		}

		System.out.println("value of Integrity Check is: "+S);
		String newChecksum=String.valueOf(S);
		return newChecksum;
	}//integrityCheck() method
	public static char[] stringToCharArray(String a)
	{
		char[] charArrayOfS2 = new char[a.length()];
		for(int i=0;i<charArrayOfS2.length;i++)
		{
			charArrayOfS2[i]=a.charAt(i);
		}
		return charArrayOfS2;
	}//stringToCharArray() method
	public static String[] charArrayToWordArray(char[] charArrayOfS2)
	{
		String[] wordArray = new String[charArrayOfS2.length/2];
		for(int i=0;i<charArrayOfS2.length-1;i=i+2)
		{
			if(charArrayOfS2.length%2==0)
			{
				/*if there are even number of characters in char array,
				 * the word array has exactly half the length of char array
				 * and has appended consecutive elements as its elements*/
wordArray[i/2]=(String.format("%8s", Integer.toBinaryString((int) charArrayOfS2[i])).replace(' ',
'0').concat(String.format("%8s", Integer.toBinaryString((int) charArrayOfS2[i+1])).replace(' ',
'0')));
			}
			else
			{
				/*if there are odd number of characters in char array,
				 * the word array has exactly half the length of char array
				 * and has appended consecutive elements as its elements and the last
character has to be appended with 8-bit zeros "00000000"*/
wordArray[i/2]=String.format("%8s", Integer.toBinaryString((int) charArrayOfS2[i])).replace(' ',
'0')+String.format("%8s", Integer.toBinaryString((int) charArrayOfS2[i+1])).replace(' ', '0');

wordArray[((charArrayOfS2.length)/2)-1]=String.format("%8s", Integer.toBinaryString((int)
charArrayOfS2[charArrayOfS2.length-1])).replace(' ', '0')+"00000000";
			}
		}
```

```
        return wordArray;
    }//charArrayToWordArray() method
    public static int[] wordArrayToIntArray(String[] a)
    {
        /*the given wordarray has elements in string array,
         * so we need to convert it into integer array in order to perform calculations*/
        int[] convIntArray=new int[a.length];

        for(int i=0;i<a.length;i++)
        {
            convIntArray[i]=Integer.parseInt(a[i],2);
        }
        return convIntArray;
    }//wordArrayToIntArray() method

}//ClientApplication class
```

APPENDIX B: JAVA CODE FOR SERVER APPLICATION

```
package trials;

import java.nio.file.*;// "nio.file" package is imported to read data from file
import java.net.*;// networking classes
import java.util.*;//scanner and other classes in java.util package
import java.io.*;//java.io classes

public class ServerApplication
{
	/*should be the same port as the client is sending packets to*/
	private final static int SERVER_PORT_NUM = 9999;
	 /* UDP segments larger than this value are not supported by some OSs*/
	private final static int MAX_MESSAGE_SIZE = 8192;
	private final static int TX_MESSAGE_SIZE = 8192; // transmitted message size

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

		// send and receive data buffers
		byte[] receivedMessage = new byte[MAX_MESSAGE_SIZE];
		byte[] sentMessage = new byte[TX_MESSAGE_SIZE];

		// send and receive UDP packets
	DatagramPacket receivePacket = new DatagramPacket(receivedMessage,receivedMessage.length);

		// server socket, listening on port SERVER_PORT_NUM
		DatagramSocket serverSocket = new DatagramSocket(SERVER_PORT_NUM);

		// do this forever
		while(true)
		{
			System.out.print("\n\nWaiting for a message from the client.");

			// receiving client's request
			serverSocket.receive(receivePacket);
			System.out.print("\nMessage received from the client.");

			// getting the client info out of the received UDP packet object
			InetAddress clientAddress = receivePacket.getAddress(); // client's IP address
			int clientPort = receivePacket.getPort(); // client's port number
			int dataLength = receivePacket.getLength(); // the number of received bytes

			//print the received byte sequence if needed
			/*System.out.print("\nThe received byte sequence is: ");
			for(int ind = 0; ind < dataLength; ind++)
			{
				System.out.printf("%d", receivedMessage[ind]);
				if(ind < dataLength-1)
					System.out.print(", ");
			}*/

			// converting the received byte array into String
			String receivedText = new String(receivedMessage, 0, dataLength);
			System.out.println("\nThe received text from the client is: ");
```

```
                System.out.println(receivedText);

                // setting up the response UDP packet object
                //generating the response code for the received String
                //calling responseCode() method to find the specific response code
                String s1=responseCode(receivedText);
                System.out.println("Sending an OK response to the client.");
                System.out.println("-----------------------------------------");

                // sending the response to the client
                receivePacket.setLength(MAX_MESSAGE_SIZE);
                // the socket is not closed, as this program is supposed to run "forever"

                byte[] responsemsgtotx=s1.getBytes();
                int portclient=receivePacket.getPort();
                InetAddress clientAddress1=receivePacket.getAddress();
                DatagramPacket sentPacket=new
DatagramPacket(responsemsgtotx,responsemsgtotx.length,clientAddress1,portclient);
                serverSocket.send(sentPacket);

        } // while(true)

    }//main()
    public static String responseCode(String recText) throws Exception,IOException
    {
        /*splitting the string using CR+LF in order to extract different parts of request message*/
        String[] delim = recText.split("\r\n");

        String delim1 = delim[0];
        String delim2 = delim[1];
        String delim3 = delim[2];

        String S2=delim1+"\r\n"+delim2+"\r\n";
        /*calculating  integrity check value only for first two parts of request message
         *  thereby verifying integrity check value later*/
        String newChecksum=integrityCheck(S2);

        /*checking whether the calculated integrity check value
         *  matches the last part of request message*/
        if(newChecksum.equals(delim3))
        {//4if
            /*verifying syntax errors in different parts of request message as follows*/
            /*part11[] contains the request message "ENTS/1.0" and "Request" fields*/
            String[] part11=delim1.split(" ");
            /*part21[] contains "filename" and "extension" fields */
            String[] part21=delim2.split("\\.");
            /*part31[] contains integrity check value of request message*/
            String[] part31=delim3.split("");

            /*the filename field should only contain alphanumeric characters and underscore */
            /*the extension field can contain only alphanumeric characters*/
            /*the first character of filename filed should be a alphabet*/

            if(part21[0].matches("^[a-zA-Z0-9_]*") && part21[1].matches("^[a-zA-Z0-9]*")
        && part21[0].substring(1, 2).matches("^[a-zA-Z]*"))
            {//3if
                /*checking the version name*/
```

```
                        if(delim1.contains("1.0"))
                        {//2if


                        /*checking user request with existing files on server*/

      if(delim2.equals("scholarly_paper.txt")||delim2.equals("directors_message.txt")||delim2.equ
als("program_overview.txt"))
                                {//1if

                                      /*printing the specific response code if needed
                                       * System.out.println("The Response code is 0");
                                       */

                                      /*the characters from the file can also be read in this way
                                      String s = new Scanner(new
                                      File("D:\\pa\\"+part2)).useDelimiter("\\Z").next();*/
                                      /*reading the characters from the selected file*/
                                      /*we can use FileReader and BufferedReader in this context,
                                       * but to simplify the code, an easy method is adopted here*/
            String s = new String(Files.readAllBytes(Paths.get("D:\\pa\\"+delim2)));
            String s3="ENTS/1.0 Response"+"\r\n"+"0\r\n"+s.length()+"\r\n"+s;
            String newChecksumr0=integrityCheck(s3);
                                /*creating the response message according to the given
                                directions*/
                                /*as all syntaxes are correct and integrity check value is
                                also matching,
                                 *  response code 0 is sent along with file data
                                accordingly*/
      String sr0="ENTS/1.0 Response"+"\r\n"+"0\r\n"+s.length()+"\r\n"+s+newChecksumr0+"\r\n";
      System.out.println("The response message that is being sent back to client is:");
      System.out.println(sr0);
      return sr0;
                                }//1if
                                else
                                {
                                      /*As the file with given name does not exist,
                                       *  Response Code 3 is sent without any file data*/

                                      /*printing the specific response code if needed
                                       * System.out.println("response code 3");
                                       */
                                      String sr3=integrityResponse("3");
                                      return sr3;

                                }//1else
                        }//2if
                        else
                        {
                                /*As the protocol version does not match,
                                 * Response code 4 is sent without any file data*/
                                /*printing the specific response code if needed
                                 *System.out.println("response code 4");*/

                                String sr4=integrityResponse("4");
                                return sr4;
                        }//2else
```

```
			}//3if
			else
			{
				/*As the syntax of the request message fails,
				 *  Response code 2 is sent without any file data*/
				/*printing the specific response code if needed
				 *System.out.println("response code 2");*/
				String sr2=integrityResponse("2");
				return sr2;
			}//3else
		}//4if
		else
		{
			/*As the integrity check value does not match,
			 *  Response code 1 is sent without any file data*/
			/*printing the specific response code if needed
			 *System.out.println("response code 1");*/
			String sr1=integrityResponse("1");
			return sr1;
		}//4else
	}//end of main() method
	public static String integrityCheck(String a)
	{
		/*converting string to char array*/
		char[] b=stringToCharArray(a);
		/*converting resultant char array to string/word array*/
		String[] c=charArrayToWordArray(b);
		/*converting the word array to integer array*/
		int[] d=wordArrayToIntArray(c);
		/*evaluating integrity check value according to the given instructions*/
		short S=0;
		for(int i=0;i<d.length-1;i++)
		{
			short index= (short)(S^d[i]);
			S=(short)((7919*index)%65536);
		}

		System.out.println("Value of Integrity Check is: "+S);
		String newChecksum=String.valueOf(S);
		return newChecksum;
	}//integrityCheck() method
	public static char[] stringToCharArray(String a)
	{
		char[] charArrayOfS2 = new char[a.length()];
		for(int i=0;i<charArrayOfS2.length;i++)
		{
			charArrayOfS2[i]=a.charAt(i);
		}
		return charArrayOfS2;

	}//stringToCharArray() method
	public static String[] charArrayToWordArray(char[] charArrayOfS2)
	{
		String[] wordArray = new String[charArrayOfS2.length/2];
		for(int i=0;i<charArrayOfS2.length-1;i=i+2)
		{
```

```
                    if(charArrayOfS2.length%2==0)
                    {
                          /*if there are even number of characters in char array,
                          * the word array has exactly half the length of char array and has
                          appended consecutive elements as its elements*/
                          wordArray[i/2]=(String.format("%8s", Integer.toBinaryString((int)
charArrayOfS2[i])).replace(' ', '0').concat(String.format("%8s", Integer.toBinaryString((int)
charArrayOfS2[i+1])).replace(' ', '0')));
                    }
                    else
                    {
                          /*if there are odd number of characters in char array,
                          * the word array has exactly half the length of char array and has
                          appended consecutive elements as its elements and the last character has
                          to be appended with 8-bit zeros "00000000"*/
                          wordArray[i/2]=String.format("%8s", Integer.toBinaryString((int)
charArrayOfS2[i])).replace(' ', '0')+String.format("%8s", Integer.toBinaryString((int)
charArrayOfS2[i+1])).replace(' ', '0');
                          wordArray[((charArrayOfS2.length)/2)-1]=String.format("%8s",
Integer.toBinaryString((int) charArrayOfS2[charArrayOfS2.length-1])).replace(' ', '0')+"00000000";
                    }
              }
              return wordArray;
        }//charArrayToWordArray() method
        public static int[] wordArrayToIntArray(String[] a)
        {
              /*the given wordarray has elements in string array,
               *  so we need to convert it into integer array in order to perform calculations*/
              int[] convIntArray=new int[a.length];
              for(int i=0;i<a.length;i++)
              {
                    convIntArray[i]=Integer.parseInt(a[i],2);
              }
              return convIntArray;
        }//wordArrayToIntArray() method

        public static String integrityResponse(String a)
        {
              /*if the response code is anything other than 0, we should not send any file data.
               * the following code forms the response code for rest other codes .i.e. 1,2,3,4*/
              String s4="ENTS/1.0 Response"+"\r\n"+a+"\r\n"+"0\r\n";
              String newChecksumr0=integrityCheck(s4);
              String sr2="ENTS/1.0 Response"+"\r\n"+a+"\r\n"+"0\r\n"+newChecksumr0+"\r\n";
              System.out.println(sr2);
              System.out.println("The length of sent string is: "+sr2.length());
              return sr2;
        }//integrityResponse() method

}//ServerApplication class
```

---------- OOOOO ----------