DATA COMMUNICATIONS AND SECURITY

Assignment: Stage 2
Unit Code: COS70007

Student Name: Lakshmi Saketh
Student Id: 101734216

Student Name: Mounika Somu
Student Id: 101791949

Contents

1. Introduction

Here in this assignment we are developing a file transfer protocol through which we are transferring the data from server to the client based on the request. The transferring of the data is utilising the TCP connections between the PC's in the network. In this we are calculating the checksum and then we have to attach that and send that in a frame. In the client we have to deserialize the data and calculate the checksum and validate it and provide acknowledgement to the server and enter the data in file. Here we are implementing the timer so that we can send the acknowledgements based on the timer if transfer of packet is not done before time then we can have to send NACK and if the transfer is done. Here we are waiting and sending the packets based on the timer and we are using the RDT stop and wait protocol. If all the lines are transmitted from the server to client, we have to pass the EOT and we have to terminate the transfer and has to save the file.

2. Structure of the Frame

Here in this we are following the format for sending the data to the client and it is as follows in the below table.

The format of the frame should be:

Start Flag	Frame no	Data block	FCS	End Flag
STX	1 or 0	100 bytes	1 byte	ETX

In this the start flag is STX which is an ASCII character and its decimal equivalent is 002 and here in this the next one is frame number. Then we must add the data to it and then we have to set the checksum value initially to zero and then we have to add the end flag as ETX. In this way we have to create the frame and has to send for the client.

Here we have to create a frame with zero checksum initially.

```
String checkSum="stx"+j+recieveChar+fcs+"etx";
String checkSum="stx"+j+recieveChar+fcs+"etx";
data=checkSum.getBytes("UTF-8");
Calculate_checksum c = new Calculate_checksum();
long s = c.checksum(true, data, data.length, null);
```

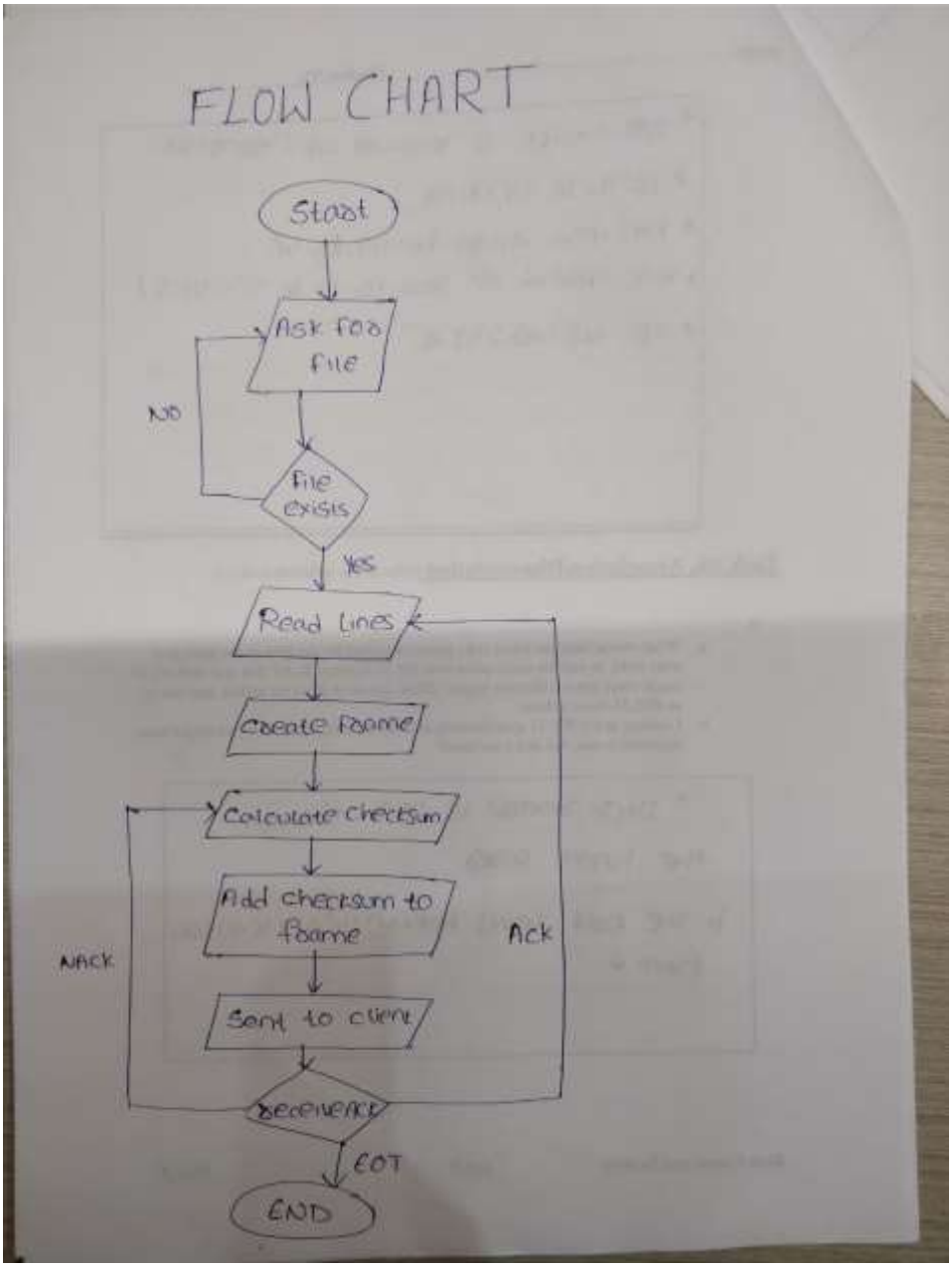
Here the receive char is the data and the fcs is zero in this frame now we have to convert this into 100 bytes of data and then we have to calculate the checksum. Here we are calculating the checksum using the CRC8(Cyclic redundancy check) here we are calculating for the 8 bits of data. Here the CRC is a error detecting technique which is used to calculate the positional checksum. It is used to detect the accidental changes for the raw data. CRC can be used for the error correction.

Calculation of checksum:

```
public class CRC8 implements Checksum {

	private final short init;

	private static final short[] crcTable = new short[256];

	private short value;

	private static final short CRC_POLYNOM = 0x8c;

	private static final short CRC_INITIAL = 0x00;

	static {
		for (int dividend = 0; dividend < 256; dividend++) {
			int remainder = dividend;// << 8;
			for (int bit = 0; bit < 8; ++bit)
				if ((remainder & 0x01) != 0)
					remainder = (remainder >>> 1) ^ CRC_POLYNOM;
				else
					remainder >>>= 1;
			crcTable[dividend] = (short) remainder;
		}
	}

	public  CRC8() {
		this.value = this.init = CRC_INITIAL;
	}

	@Override
	public  void update(byte[] buffer, int offset, int len) {
		for (int i = 0; i < len; i++) {
			int data = buffer[offset + i] ^ value;
			value = (short) (crcTable[data & 0xff] ^ (value << 8));
		}
	}

	/**
	 * Updates the current checksum with the specified array of bytes.
	 * Equivalent to calling <code>update(buffer, 0, buffer.length)</code>.
	 *
	 * @param buffer
	 *            the byte array to update the checksum with
	 */
	public  void update(byte[] buffer) {
		update(buffer, 0, buffer.length);
	}

	@Override
	public  void update(int b) {
		update(new byte[] { (byte) b }, 0, 1);
	}

	@Override
	public  long getValue() {
		return value & 0xff;
	}

	@Override
	public  void reset() {
		value = init;
	}

}
```

‘Here in the above we have written a checksum calculation. Through which we can calculate the checksum for any type of data. Here the Arguments for the functions are Boolean,byte[], data.length,null. In this we can represent the 8-bit data in 2^8 ways so we can do in 256 ways. So here we are dividing the bytes by 256 and we are converting that data and we are returning that in the for of long Format. After calculating the value we are adding this checksum value to the frame and now again we are reframing it to a new one and now using the writeString() we are sending the data to the client.

3. Modular structure of Program

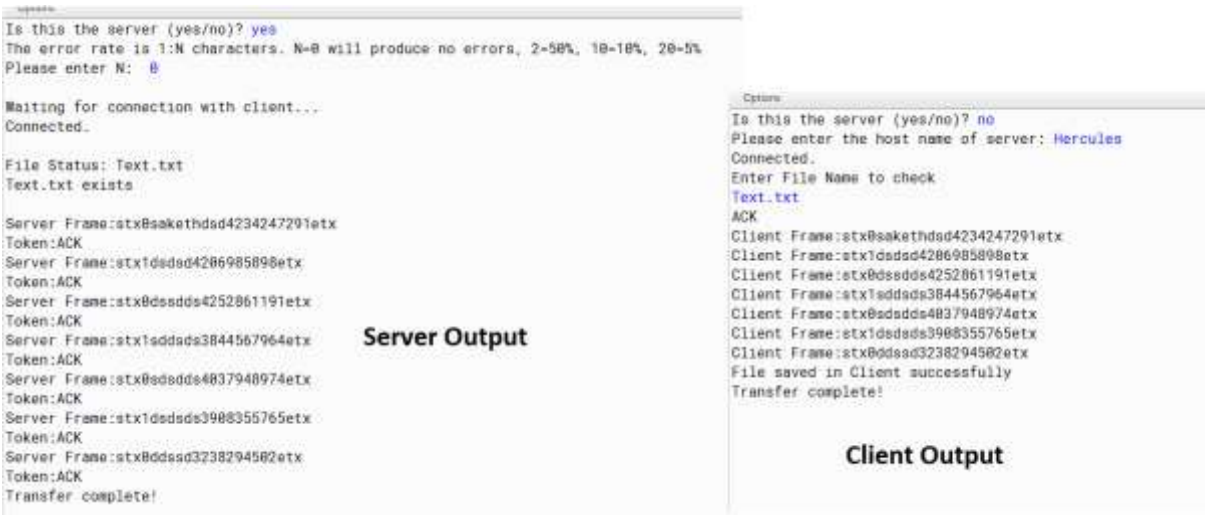


Flow Chart for the Program structure

4. *Protocols RDT stop and wait*

Here in this we have implemented the program in way such that it satisfies the protocol RDT stop and wait. The program is entirely based on the **error and timeout** values in the program. In this if the checksum value is wrong then the program will return the NACK token to the server and then it has to resend the data. If the data transferred is crossing the timeline set by us will also cause in failure and again the data is resent by the server and for this also the client will return the NACK token for the client. If the data transferred is in proper way and the data is satisfying the checksum values of server data and then the data is eligible for storing in the file. In the end of the transmission the server will send the token EOT based on that we will end the transmission process.

Output:



5. *Conclusion*

Here we have completed the tasks based on the requirements. We have calculated the checksum in the program. We have created the format for the frame. We have transferred the data to the client. We have prepared the error detecting and timeout option in the program, i.e. we have performed the RDT stop and wait protocol which is used in transferring the data between the computers connected to a network.

6. Contribution section

Lakshmi Saketh Chebrolu (101734216)	**Mounika Somu (101791949)**
• Assignment analyzation and planning based on the requirement. • Serializing data. • Calculate checksum. • Reading the file in the program. • Server program tasks. • Documentation of things done by me.	• Critical analyzation and understanding of the requirement. • Creating frame. • Deserializing the data. • Writing the file in the program. • Client program tasks • Documentation of things done by me