Assignment 1

Name: Mu In Nasif
Class: BCSE III
Session: 2019-2023
Semester: V
Roll: 001910501036

Topic title: Design and implement an error detection module.

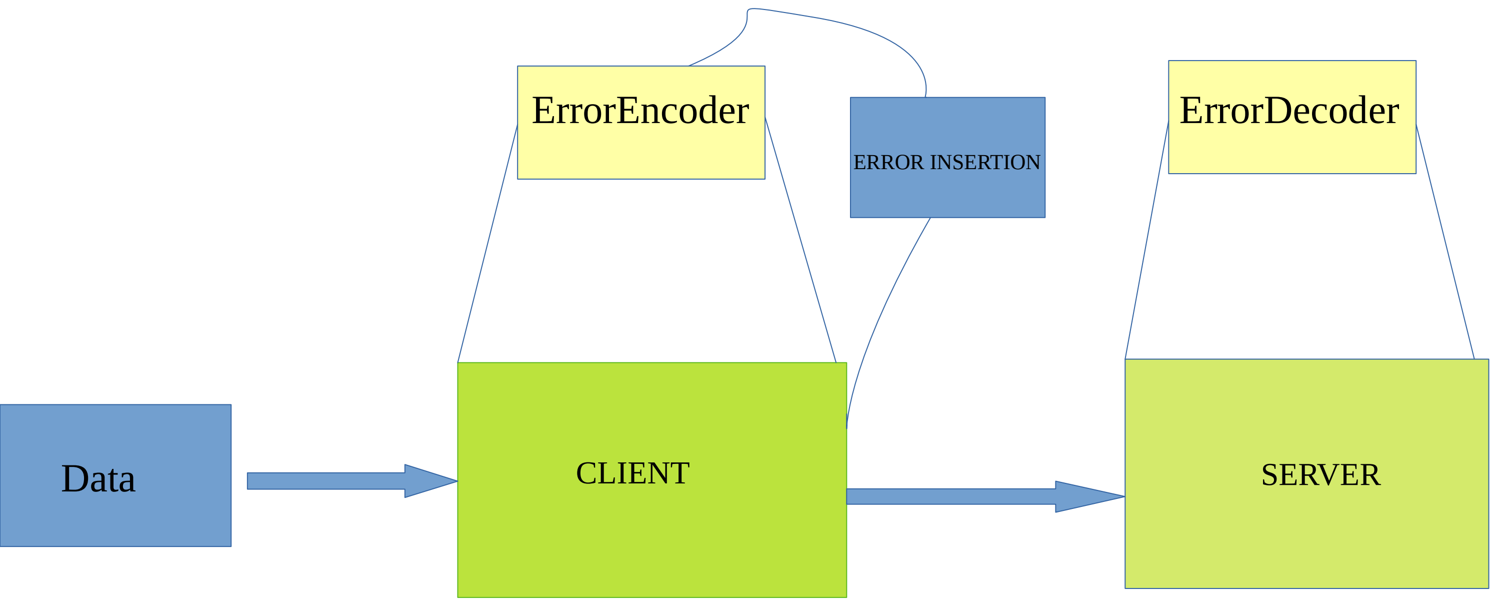
Design:

The complete project was implemented using Java 16. The key elements to the program is the Client-Socket module. The client and socket are made to run on the same system connected through a given port number.

The elaborate component list is listed below:

- Encode Data module
- Detect Error module
- Client (sender)
- Server (receiver)
- File(containing the data to be tested)

The diagrammatic representation of the complete setup is provided below:



Implementation:

The data is read from the file data.txt and stored in an array of String of size 1000. Substrings of size 16bits (2 bytes) are created and encoded using the Encode Data module. The encoded message is then send through the Client and received by the Server the data is then analysed using the Error Detection module

Individual methods

VRC: The given data of 16 bits is divided into 8bit substrings and the parity of the sliced strings are calculated and made to catenate with the 8bit substring.

Sent data : data/2 + VRC(bit) + data/2 + VRC(bit)

LRC: The given data of 16 bits is divided into 8bit substrings and the parity of each 8 columns and the new parity substring is catenated to the end of the data.

Sent data: data + LRC(bits)

Checksum: The given 16 bit data is complemented and we find the xor between the complemented with the binary of 2^n – 1 where n is the number of bits of the data. The sum of the complemented data is then catenated with the original 16 bit data.

Sent data: data + Checksum(bits)

CRC: We implement the CRC-10 and catenate the remainder to the 16bit data

Sent data: data + CRC(bits)

Test cases:

test cases have been generated by a random testcase generating program.

the code for the test case generating program is provided below

```
// *** java 16.0.1
import java.io.FileWriter;
import java.io.IOException;
```

```
public class GenerateDataWord {
   public static void main(String[] args) {
      // File dataFile = new File("data.txt");
      try {
         FileWriter fileWriter = new FileWriter("data.txt");

         for (int i = 0; i < 1000; i++) {
            String dat = "";
            for (int j = 0; j < 16; j++) {
               long z = Math.round((Math.random() * 10));
               if (z % 2 == 0) {
                  dat += "0";
               } else {
                  dat += "1";
               }
            }
            dat += "\n";
            fileWriter.write(dat);
         }

         fileWriter.close();
      } catch (IOException e) {
         e.printStackTrace();
      }

   }
}
```

Result:

No of error detected of 1000 test cases over various iterations

VRC:	52.5%
LRC:	87.0%
Checksum:	71.2%
CRC:	100.0%

Analysis:

From random generated test cases over various iterations we can come to the conclusion that implementation of the CRC(Cyclic Redundancy Check) proves to provide the most efficient in transferring data.