Computer Networks Lab Assessment 4

Suryakumar P 21MIS1146

(a) The generated polynomial for the CRC error detection scheme is X5+X3+1. The data to be transferred from the server is 111011010011001. Write the code to find the data transferred from the sender.

Problem Definition:

To implement a CRC (Cyclic Redundancy Check) error detection scheme for data transfer from a server to a sender. The generator polynomial for the CRC scheme is given as X^5 + X^3 + 1 and to determine the data transferred from the sender using this CRC error detection scheme.

Method:

1. Initialize the data to be transferred from the server, including the message bits and the CRC bits:
    - Message bits: 111011010011001
    - CRC bits: 00000 (initialized with zeros since they will be filled later)
2. Perform polynomial division using bitwise XOR operations:
    - Append zeros to the message bits to match the degree of the polynomial (5 zeros for a degree-5 polynomial).
    - Begin dividing the polynomial by the CRC polynomial:
        - Take the first 6 bits (degree of the polynomial + 1) from the message bits and perform a bitwise XOR with the polynomial.
        - The result of the XOR operation will give you the next bits to append to the CRC bits.
        - Shift the polynomial one bit to the right (divide by x) and repeat the process until you have divided the entire message bits.
3. The resulting CRC bits will be the remainder after polynomial division.
    - Append the resulting CRC bits to the original message bits.
4. The data transferred from the sender will be the combination of the original message bits and the CRC bits.

Code:

```
#CRC - 21MIS1146
def crc_error_detection(data, divisor):
    data = list(data)  # Convert the data string to a list of bits
    divisor_length = len(divisor)

    # Append zeros to the data string to match the divisor length
```

```
    data.extend(['0'] * (divisor_length - 1))

    # Perform polynomial division
    for i in range(len(data) - divisor_length + 1):
        if data[i] == '1':
            for j in range(divisor_length):
                data[i + j] = str(int(data[i + j]) ^ int(divisor[j]))

    # Return the remainder (CRC code)
    crc_code = ''.join(data[-divisor_length + 1:])

    return crc_code


def crc_data_transferred(data, polynomial):
    # Append zeros to the data to match the polynomial length
    data_bits = data + '0' * (len(polynomial) - 1)

    # Compute the CRC code
    crc_code = crc_error_detection(data_bits, polynomial)

    # Append the CRC code to the original data
    transmitted_data = data + crc_code

    return transmitted_data


data = '111011010011001'
polynomial = '101001'

transmitted_data = crc_data_transferred(data, polynomial)
print("Transmitted Data:", transmitted_data)
```

Output:

```
PS D:\VIT\Fall Inter Semester 23-24\SWE2002 - Computer Networks\Lab Assessments\Assessment4\Question1> python CRC.py
Transmitted Data: 1110110100110010100
PS D:\VIT\Fall Inter Semester 23-24\SWE2002 - Computer Networks\Lab Assessments\Assessment4\Question1>
```

(b) Write the code for error correction techniques.

(i) Hamming distance

Problem Definition:

To implement Hamming Code Error Correction in Python

Method:

1. Enter the Data to be transmitted.
2. Check if their lengths are equal.
3. If equal iterate each character and increment the count variable if they are unequal.
4. Print the value of count as Hamming Distance.

Code:

```
# Python3 program to find Hamming Distance between two strings - 21MIS1146

# Function to calculate Hamming distance
def hammingDist(str1, str2):
    i = 0
    count = 0
    #Exception to handle unequal string length
    if len(str1) != len(str2):raise ValueError ("Input strings must have equal
length.")

    while(i < len(str1)):
        if(str1[i] != str2[i]):
            count += 1
        i += 1
    return count

# Driver code
str1 = "10101011"
str2 = "11001010"

# function call
print("Hamming Distance : ",hammingDist(str1, str2))
```

Output:

```
PS D:\VIT\Fall Inter Semester 23-24\SWE2002 - Computer Networks\Lab Assessments\Assessment4\Question2> python Hamming_Distance.py
Hamming Distance :  3
PS D:\VIT\Fall Inter Semester 23-24\SWE2002 - Computer Networks\Lab Assessments\Assessment4\Question2>
```

(ii) Reed-Solomon code

Problem Definition:

To implement Reed-Solomon Code in Python

Method:

1. Define a class called ReedSolomon.
2. Declare a private constant integer variable GF_SIZE with the value 256.
3. Declare two private integer arrays: generator and parity, in the constructor of the ReedSolomon class and initialize them with values passed as arguments to the constructor.
4. Define an encode() method that takes an array of integers as input and returns an array of integers.
5. Inside the encode() method, create a new integer array called coefficients with a length equal to the sum of the lengths of the data and parity arrays.
6. Copy the elements of the input data array to the first part of the coefficients array using System.arraycopy() method.
7. Use two nested loops to calculate the remaining elements of the coefficients array.
8. For each element in the parity array, iterate over all the elements in the data array and multiply them with the corresponding generator element. Add up these products for each parity element and store the result in the coefficients array.
9. Return the calculated coefficients array.
10. Define a decode() method that takes an integer array and an integer numErrors as inputs, and returns an integer array.
11. Inside the decode() method, create a new integer array called data with a length equal to the length of the input coefficients array.
12. Use two nested loops to calculate the values of each element in the data array.
13. For each element in the data array, iterate over all the elements in the coefficients array and add them up.
14. Store the final values of the data array in the output array.
15. Return the calculated data array.
16. In the main method, create an instance of the ReedSolomon class by passing the generator and parity arrays.
17. Encode the input data array using the encode() method and store the results in encodedData array.
18. Corrupt some of the data in the encodedData array by modifying certain elements.
19. Decode the corrupted encodedData array using the decode() method with a numErrors argument of 2, and store the results in decodedData array.

20. Print the original data, encoded data, and decoded data arrays to the console using Arrays.toString() method.

Code:

```java
// Java Implementation for Reed Solomon Code - 21MIS1146

import java.util.Arrays;

public class ReedSolomon {

    private static final int GF_SIZE = 256;

    private final int[] generator;
    private final int[] parity;

    public ReedSolomon(int[] generator, int[] parity) {
        this.generator = generator;
        this.parity = parity;
    }

    public int[] encode(int[] data) {
        int[] coefficients = new int[data.length + parity.length];
        System.arraycopy(data, 0, coefficients, 0, data.length);
        for (int i = 0; i < parity.length; i++) {
            for (int j = 0; j < data.length; j++) {
                coefficients[i + data.length] += generator[j] * data[i];
            }
        }
        return coefficients;
    }

    public int[] decode(int[] coefficients, int numErrors) {
        int[] data = new int[coefficients.length];
        for (int i = 0; i < data.length; i++) {
            int value = 0;
            for (int j = 0; j < coefficients.length; j++) {
                value += coefficients[j] ;
            }
            data[i] = value;
        }
        return data;
    }

    public static void main(String[] args) {
        int[] data = {1, 2, 3, 4, 5, 6, 7, 8};
        int[] generator = {1, 0, 1, 0, 1, 0, 1, 0, 1};
        int[] parity = {1, 1, 1, 1};
```

```
        ReedSolomon reedSolomon = new ReedSolomon(generator, parity);
        int[] encodedData = reedSolomon.encode(data);

        // Corrupt some of the data
        encodedData[0] = 9;
        encodedData[1] = 10;

        int[] decodedData = reedSolomon.decode(encodedData, 2);

        System.out.println("Original data: " + Arrays.toString(data));
        System.out.println("Encoded data: " + Arrays.toString(encodedData));
        System.out.println("Decoded data: " + Arrays.toString(decodedData));
    }
}
```

Output:

```
PS D:\VIT\Fall Inter Semester 23-24\SWE2002 - Computer Networks\Lab Assessments\Assessment4\Question2> java ReedSolomon.java
Original data: [1, 2, 3, 4, 5, 6, 7, 8]
Encoded data: [9, 10, 3, 4, 5, 6, 7, 8, 4, 8, 12, 16]
Decoded data: [92, 92, 92, 92, 92, 92, 92, 92, 92, 92, 92, 92]
PS D:\VIT\Fall Inter Semester 23-24\SWE2002 - Computer Networks\Lab Assessments\Assessment4\Question2>
```

(c) An organization plans to send general instructions to the in charge of each department using a socket. Write the code to check the message for any data loss when transmitting using the checksum in the header.

Problem Definition:

To implement Checksum Algorithm in Python.

Method:

Sender

1. Set the server host and port to send the message.
2. Create a socket connection to the server.
3. Get the output stream to send data using ObjectOutputStream.
4. Define the message to be sent and convert it to byte array.
5. Calculate the checksum of the message using CRC32 class.
6. Create a MessageData object with the message and checksum.
7. Send the MessageData object over the socket using writeObject() method.
8. Flush the output stream.
9. Close the socket and streams.

Receiver

1. Set the server port to listen on.
2. Create a server socket.
3. Wait for a client connection using accept() method.
4. Get the input stream to receive data using ObjectInputStream.
5. Receive the MessageData object from the client using readObject() method.
6. Get the message and checksum from the received object.
7. Calculate the checksum of the received message using CRC32 class.
8. Check if the received checksum matches the calculated checksum.
9. If the received checksum matches the calculated checksum, print the received message and confirmation message; otherwise, print a message about data loss.
10. Close the socket and streams.
11. Close the server socket.

Message

1. Define the class with two private fields: message and checksum.
2. Create a constructor that takes both fields as parameters.

Create getter methods for both fields.

Code:

```
# Function to find the Checksum of Sent Message
def findChecksum(SentMessage, k):
//Server Checksum - 21MIS1146
import java.io.*;
import java.net.*;
import java.util.zip.CRC32;

public class Sender {
    public static void main(String[] args) {
        String message = "General instructions to the departments"; // Example
message

        try {
            // Create a socket and connect to the receiver
            Socket socket = new Socket("localhost", 12345);

            // Get the output stream to send data
            OutputStream outputStream = socket.getOutputStream();
            ObjectOutputStream objectOutputStream = new
ObjectOutputStream(outputStream);

            // Calculate checksum
            CRC32 crc32 = new CRC32();
            crc32.update(message.getBytes());
            long checksum = crc32.getValue();

            // Create a message object with the message and checksum
            Message transmittedMessage = new Message(message, checksum);

            // Send the message object
            objectOutputStream.writeObject(transmittedMessage);
            objectOutputStream.flush();

            // Close the streams and socket
            objectOutputStream.close();
            outputStream.close();
            socket.close();

            System.out.println("Message sent successfully.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```java
//Receiver - 21MIS1146
import java.io.*;
import java.net.*;
import java.util.zip.CRC32;

public class Receiver {
    public static void main(String[] args) {
        try {
            // Create a server socket and start listening for incoming
connections
            ServerSocket serverSocket = new ServerSocket(12345);
            System.out.println("Waiting for incoming connection...");

            // Accept the connection from the sender
            Socket socket = serverSocket.accept();
            System.out.println("Connection established.");

            // Get the input stream to receive data
            InputStream inputStream = socket.getInputStream();
            ObjectInputStream objectInputStream = new
ObjectInputStream(inputStream);

            // Receive the message object
            Message receivedMessage = (Message)
objectInputStream.readObject();

            // Get the message and checksum from the received message
            String message = receivedMessage.getMessage();
            long receivedChecksum = receivedMessage.getChecksum();

            // Calculate checksum on the receiver side
            CRC32 crc32 = new CRC32();
            crc32.update(message.getBytes());
            long calculatedChecksum = crc32.getValue();

            // Compare the received checksum with the calculated checksum
            if (receivedChecksum == calculatedChecksum) {
                System.out.println("Checksum verification successful. Message
received without data loss.");
                System.out.println("Received message: " + message);
            } else {
                System.out.println("Checksum verification failed. Data loss
detected during transmission.");
            }

            // Close the streams and socket
            objectInputStream.close();
            inputStream.close();
```

```
            socket.close();
            serverSocket.close();
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

```
//Message Class
import java.io.Serializable;

public class Message implements Serializable {
    private final String message;
    private final long checksum;

    public Message(String message, long checksum) {
        this.message = message;
        this.checksum = checksum;
    }

    public String getMessage() {
        return message;
    }

    public long getChecksum() {
        return checksum;
    }
}
```

Output: