EC 4060 – COMPUTER AND DATA NETWORK

LAB 03

K.J.M.U.G.S. Eranda Jayasinghe

2021/E/075

GROUP CG8

EC4060

13.11.2023

OBJECTIVES:

At the end of this lab you should be able to do,

• Write a simple program to simulate CRC.

• Write a simple program to simulate the Hamming code.

• Write a simple program to simulate the Go-Back-N (Sliding window) Protocol.

Cyclic Redundancy Check (CRC) CRC codes are often used for error detection over frames or vectors of a certain length. To get a convenient mathematical notation of the positions in the frame it can be expressed as a polynomial in x, where the exponent of x is the place marker of the coefficient. The vector $a = a_{L-1}a_{L-2} \dots \dots \dots \dots a_1a_0$ length L is represented by the degree $L - 1$ polynomial

$$a(x) = \sum_{i=0}^{L-1} a_i x^i = a_{L-1}x^{L-1} + a_{L-2}x^{L-2} + \cdots + a_1 + a_0$$

Example: the vector $a = 1000011$ is transformed as

$$\begin{aligned} a(x) &= 1x^6 + 0x^5 + 0x^4 + 0x^3 + 0x^2 + 1x^2 + 1x^0 \\ &= x^6 + x + 1 \end{aligned}$$

Algorithm: -

1. Given a bit string, append 0S to the end of it (the number of 0s is the same as the degree of the generator polynomial) let $a(x)$ be the polynomial corresponding to a.

2. Divide $a(x)$ by some agreed on polynomial $G(x)$ (generator polynomial) and determine the remainder $R(x)$. This division is to be done using Modulo 2 Division.

3. Define $T(x) = a(x) - R(x)$

4. $(T(x)/G(x) \Rightarrow remainder\ 0)$

5. Transmit T, the bit string corresponding to $T(x)$.

6. Let T' represent the bit stream the receiver gets and $T'(x)$ the associated polynomial. The receiver divides $T'(x)$ by $G(x)$. If there is a 0 remainder, the receiver concludes $T = T'$ and no error occurred otherwise, the receiver concludes an error occurred and requires a retransmission

Task 1: Write a program for error detecting code using CRC-CCITT (16-bits).

```
import java.util.Scanner;


public class CRC {


   // Define polynomial
   final static String polynomial = "1010000001010110";


   // Perform XOR division
   public static int[] xordivider(int[] numerator, int[] denominator) {
      int[] remainder = new int[denominator.length - 1];
      System.arraycopy(numerator, 0, remainder, 0, denominator.length - 1);


      for (int i = denominator.length - 1; i < numerator.length; i++) {
         if (remainder[0] == 1) {
            for (int j = 1; j < denominator.length - 1; j++) {
               remainder[j - 1] = remainder[j] ^ denominator[j];
            }
            remainder[denominator.length - 2] = numerator[i] ^ denominator[denominator.length - 1];
         } else {
            for (int j = 1; j < denominator.length - 1; j++) {
               remainder[j - 1] = remainder[j];
            }
            remainder[denominator.length - 2] = numerator[i];
         }
      }
      return remainder;
   }


   public static void main(String[] args) {
      // Get user input for sender message
      System.out.print("Enter the send message (1 or 0 s): ");
Scanner sendMessageScanner = new Scanner(System.in);
      String sendMessage = sendMessageScanner.nextLine();
      int[] sendMessageArray = new int[sendMessage.length() + polynomial.length() - 1];


      // Convert sender message to array
```

```
        for (int i = 0; i < sendMessageArray.length; ++i) {
            if (i < sendMessage.length()) {
                sendMessageArray[i] = (sendMessage.charAt(i) == '1') ? 1 : 0;
            } else {
                sendMessageArray[i] = 0;
            }
        }

        // Convert polynomial to array
        int[] polynomialArray = new int[polynomial.length()];
        for (int i = 0; i < polynomial.length(); ++i) {
            polynomialArray[i] = (polynomial.charAt(i) == '1') ? 1 : 0;
        }

        // Calculate remainder using sender message and polynomial
        int[] remainder1 = xordivider(sendMessageArray, polynomialArray);

        // Get user input for received message
        System.out.print("Enter the received message (1 or 0 s): ");
Scanner receiveMessageScanner = new Scanner(System.in);
        String receiveMessage = receiveMessageScanner.nextLine();
        int[] receiveMessageArray = new int[receiveMessage.length() + remainder1.length];

        // Convert received message to array
        for (int i = 0; i < receiveMessageArray.length; ++i) {
            if (i < receiveMessage.length()) {
                receiveMessageArray[i] = (receiveMessage.charAt(i) == '1') ? 1 : 0;
            } else {
                receiveMessageArray[i] = remainder1[i - receiveMessage.length()];
            }
        }

        // Calculate remainder using received message and polynomial
        int[] remainder2 = xordivider(receiveMessageArray, polynomialArray);

        // Check if the remainder has only zeros
        int total = 0;
```

```
    for (int i = 0; i < remainder2.length; ++i) {
        total += remainder2[i];
    }

    // Print result based on the remainder
    if (total == 0) {
        System.out.println("Message arrived with no error");
    } else {
        System.out.println("Message corrupted");
    }
  }
}
```

```java
import java.util.Scanner;

public class CRC {

    // Define polynomial
    final static String polynomial = "1010000001010110";

    // Perform XOR division
    public static int[] xordivider(int[] numerator, int[] denominator) {
        int[] remainder = new int[denominator.length - 1];
        System.arraycopy(numerator, 0, remainder, 0, denominator.length - 1);

        for (int i = denominator.length - 1; i < numerator.length; i++) {
            if (remainder[0] == 1) {
                for (int j = 1; j < denominator.length - 1; j++) {
                    remainder[j - 1] = remainder[j] ^ denominator[j];
                }
                remainder[denominator.length - 2] = numerator[i] ^ denominator[denominator.length - 1];
            } else {
                for (int j = 1; j < denominator.length - 1; j++) {
                    remainder[j - 1] = remainder[j];
                }
                remainder[denominator.length - 2] = numerator[i];
            }
        }
        return remainder;
    }

    public static void main(String[] args) {
        // Get user input for sender message
        System.out.print("Enter the send message (1 or 0 s): ");
Scanner sendMessageScanner = new Scanner(System.in);
        String sendMessage = sendMessageScanner.nextLine();
        int[] sendMessageArray = new int[sendMessage.length() + polynomial.length() - 1];

        // Convert sender message to array
        for (int i = 0; i < sendMessageArray.length; ++i) {
            if (i < sendMessage.length()) {
                sendMessageArray[i] = (sendMessage.charAt(i) == '1') ? 1 : 0;
            } else {
                sendMessageArray[i] = 0;
            }
        }

        // Convert polynomial to array
        int[] polynomialArray = new int[polynomial.length()];
        for (int i = 0; i < polynomial.length(); ++i) {
            polynomialArray[i] = (polynomial.charAt(i) == '1') ? 1 : 0;
        }

        // Calculate remainder using sender message and polynomial
        int[] remainder1 = xordivider(sendMessageArray, polynomialArray);

        // Get user input for received message
        System.out.print("Enter the received message (1 or 0 s): ");
Scanner receiveMessageScanner = new Scanner(System.in);
```

```
        for (int i = 0; i < sendMessageArray.length; ++i) {
            if (i < sendMessage.length()) {
                sendMessageArray[i] = (sendMessage.charAt(i) == '1') ? 1 : 0;
            } else {
                sendMessageArray[i] = 0;
            }
        }

        // Convert polynomial to array
        int[] polynomialArray = new int[polynomial.length()];
        for (int i = 0; i < polynomial.length(); ++i) {
            polynomialArray[i] = (polynomial.charAt(i) == '1') ? 1 : 0;
        }

        // Calculate remainder using sender message and polynomial
        int[] remainder1 = xordivider(sendMessageArray, polynomialArray);

        // Get user input for received message
        System.out.print("Enter the received message (1 or 0 s): ");
Scanner receiveMessageScanner = new Scanner(System.in);
        String receiveMessage = receiveMessageScanner.nextLine();
        int[] receiveMessageArray = new int[receiveMessage.length() + remainder1.length];

        // Convert received message to array
        for (int i = 0; i < receiveMessageArray.length; ++i) {
            if (i < receiveMessage.length()) {
                receiveMessageArray[i] = (receiveMessage.charAt(i) == '1') ? 1 : 0;
            } else {
                receiveMessageArray[i] = remainder1[i - receiveMessage.length()];
            }
        }

        // Calculate remainder using received message and polynomial
        int[] remainder2 = xordivider(receiveMessageArray, polynomialArray);

        // Check if the remainder has only zeros
        int total = 0;
        for (int i = 0; i < remainder2.length; ++i) {
            total += remainder2[i];
        }

        // Print result based on the remainder
        if (total == 0) {
            System.out.println("Message arrived with no error");
        } else {
            System.out.println("Message corrupted");
        }
    }
}
```

No error

```
C:\Users\2021E075\OneDrive - University of Jaffna\netlab3\T1>java CRC
Enter the send message (1 or 0 s): 11001010
Enter the received message (1 or 0 s): 11001010
Message arrived with no error
```

With error

```
C:\Users\2021E075\OneDrive - University of Jaffna\netlab3\T1>java CRC
Enter the send message (1 or 0 s): 11100111
Enter the received message (1 or 0 s): 11101111
Message currepted
```

2. Hamming Codes

The key to the Hamming Code is the use of extra parity bits to allow the identification of a single error. Create the code word as follows:

1. Mark all bit positions that are powers of two as parity bits. (1, 2, 4, 8, 16, 32, 64, ….)

2. All other bit positions are for the data to be encoded. (3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, ….)

3. Each parity bit calculates the parity for some of the bits in the code word. The position of the parity bit determines the sequence of bits that it alternately checks and skips.

Position 1: check 1 bit, skip 1 bit. (1, 3, 5, 7, 9, 11, 13, 15,...)

Position 2: check 2 bits, skip 2 bits. (2, 3, 6, 7, 10, 11, 14, 15,...)

Position 4: check 4 bits, skip 4 bits. (4, 5, 6, 7, 12, 13, 14, 15, 20, 21, 22, 23,...)

Position 8: check 8 bits, skip 8 bits (8-15,24-31, 40-47, ...)

Position 16: check 16 bits, skip 16 bits (16-31, 48-63, 80-95, ...)

Position 32: check 32 bits, skip 32 bits (32-63, 96-127, 160-191, ...)

4. Set a parity bit to 1 if the total number of ones in the positions it checks is odd. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

Example: A byte of data: 10011010

• Create the data word, leaving spaces for the parity bits: _ _ 1 _ 0 0 1 _ 1 0 1 0 Calculate the parity for each parity bit (a ? represents the bit position being set):

• Position 1 checks bits 1,3,5,7,9,11: ? _ 1 _ 0 0 1 _ 1 0 1 0. Even parity so set position 1 to a 0: 0 _ 1 _ 0 0 1 _ 1 0 1 0

• Position 2 checks bits 2,3,6,7,10,11: 0 ? 1 _ 0 0 1 _ 1 0 1 0. Odd parity so set position 2 to a 1: 0 1 1 _ 0 0 1 _ 1 0 1 0

•

Position 4 checks bits 4,5,6,7,12: 0 1 1 ? 0 0 1 _ 1 0 1 0. Odd parity so set position 4 to a 1: 0 1 1 1 0 0 1 _ 1 0 1 0

• Position 8 checks bits 8,9,10,11,12: 0 1 1 1 0 0 1 ? 1 0 1 0. Even parity so set position 8 to a 0: 0 1 1 1 0 0 1 0 1 0 1 0 Code word: 011100101010.

Task 2: Write a program for Hamming Code generation for error detection and correction

```
import java.util.Scanner;
import java.util.Arrays;


public class HammingCode
{
	public static int[]sender_message_convert(int []sender_message_array)
	{
		int total=0;
		for(int i=0;i<3;++i)
		{
			int paritybit_position=(int)Math.pow(2,i)-1;
			if(i==0)
			{

	total=sender_message_array[2]+sender_message_array[4]+sender_message_array[6]+sender_message_array[8]+sender_message_array[10];
				if(total%2==0)
				{
					sender_message_array[paritybit_position]=0;
				}
				else
				{
					sender_message_array[paritybit_position]=1;
				}
			}
			else if(i==1)
			{
				total=0;

	total=sender_message_array[2]+sender_message_array[5]+sender_message_array[6]+sender_message_array[9]+sender_message_array[10];
				if(total%2==0)
				{
					sender_message_array[paritybit_position]=0;
				}
				else
				{
```

```
                                        sender_message_array[paritybit_position]=1;
                                }
                        }
                        else if(i==2)
                        {
                                total=0;

        total=sender_message_array[4]+sender_message_array[5]+sender_message_array[6]+sender_message_arra
y[11];
                                if(total%2==0)
                                {
                                        sender_message_array[paritybit_position]=0;
                                }
                                else
                                {
                                        sender_message_array[paritybit_position]=1;
                                }
                        }
                        else
                        {
                                total=0;

        total=sender_message_array[8]+sender_message_array[9]+sender_message_array[10]+sender_message_arr
ay[11];
                                if(total%2==0)
                                {
                                        sender_message_array[paritybit_position]=0;
                                }
                                else
                                {
                                        sender_message_array[paritybit_position]=1;
                                }
                        }
                }
                return   sender_message_array;
        }

        public static boolean statement(int []reciver_message_word)
```

```
        {
                int total=0,increment=0;
                for(int i=0;i<=3;++i)
                {
                        int paritybit_position=(int)Math.pow(2,i)-1;
                        if(i==0)
                        {

        total=reciver_message_word[2]+reciver_message_word[4]+reciver_message_word[6]+reciver_message_word[8]+reciver_message_word[10];
                                if(total%2==0)
                                {
                                        if(reciver_message_word[paritybit_position]==0)
                                        {
                                                ++increment;
                                        }
                                }
                                else
                                {
                                        if(reciver_message_word[paritybit_position]==1)
                                        {
                                                ++increment;
                                        }
                                }
                        }
                        else if(i==1)
                        {
                                total=0;

        total=reciver_message_word[2]+reciver_message_word[5]+reciver_message_word[6]+reciver_message_word[9]+reciver_message_word[10];
                                if(total%2==0)
                                {
                                        if(reciver_message_word[paritybit_position]==0)
                                        {
                                                ++increment;
                                        }
                                }
```

```
                else
                {
                        if(reciver_message_word[paritybit_position]==1)
                        {
                                ++increment;
                        }
                }
        }
        else if(i==2)
        {
                total=0;

        total=reciver_message_word[4]+reciver_message_word[5]+reciver_message_word[6]+reciver_message_word[11];
                if(total%2==0)
                {
                        if(reciver_message_word[paritybit_position]==0)
                        {
                                ++increment;
                        }
                }
                else
                {
                        if(reciver_message_word[paritybit_position]==0)
                        {
                                ++increment;
                        }
                }
        }
        else
        {
                total=0;

        total=reciver_message_word[8]+reciver_message_word[9]+reciver_message_word[10]+reciver_message_word[11];
                if(total%2==0)
                {
                        if(reciver_message_word[paritybit_position]==0)
```

```
                                        {
                                                ++increment;
                                        }
                                }
                                else
                                {
                                        if(reciver_message_word[paritybit_position]==1)
                                        {
                                                ++increment;
                                        }
                                }
                        }
                }
                if(increment==3)
                {
                        return true;
                }
                else
                {
                        return false;
                }
        }
        public static void correct(int [] reciver_message_array)
        {
                int total=0,correct_parity_bit=0;
                for(int i=0;i<=3;++i)
                {
                        int paritybit_position=(int)Math.pow(2,i)-1;
                        if(i==0)
                        {

        total=reciver_message_array[2]+reciver_message_array[4]+reciver_message_array[6]+reciver_message_arra
y[8]+reciver_message_array[10];
                                if(total%2==0)
                                {
                                        if(reciver_message_array[paritybit_position]!=0)
                                        {
```

```
                                        correct_parity_bit+=paritybit_position;
                                }
                        }
                        else
                        {
                                if(reciver_message_array[paritybit_position]!=1)
                                {
                                        correct_parity_bit+=paritybit_position;
                                }
                        }
                }
                else if(i==1)
                {
                        total=0;

        total=reciver_message_array[2]+reciver_message_array[5]+reciver_message_array[6]+reciver_message_arra
y[9]+reciver_message_array[10];
                        if(total%2==0)
                        {
                                if(reciver_message_array[paritybit_position]!=0)
                                {
                                        correct_parity_bit+=paritybit_position;
                                }
                        }
                        else
                        {
                                if(reciver_message_array[paritybit_position]!=1)
                                {
                                        correct_parity_bit+=paritybit_position;
                                }
                        }
                }
                else if(i==2)
                {
                        total=0;

        total=reciver_message_array[4]+reciver_message_array[5]+reciver_message_array[6]+reciver_message_arra
y[11];
```

```
                        if(total%2==0)
                        {
                                if(reciver_message_array[paritybit_position]!=0)
                                {
                                        correct_parity_bit+=paritybit_position;
                                }
                        }
                        else
                        {
                                if(reciver_message_array[paritybit_position]!=1)
                                {
                                        correct_parity_bit+=paritybit_position;
                                }
                        }
                }
                else
                {
                        total=0;

        total=reciver_message_array[8]+reciver_message_array[9]+reciver_message_array[10]+reciver_message_arr
ay[11];
                        if(total%2==0)
                        {
                                if(reciver_message_array[paritybit_position]!=0)
                                {
                                        correct_parity_bit+=paritybit_position;
                                }
                        }
                        else
                        {
                                if(reciver_message_array[paritybit_position]!=1)
                                {
                                        correct_parity_bit+=paritybit_position;
                                }
                        }
                }
        }
```

```
            System.out.println((correct_parity_bit+2) +" bit position is wrong ");
            System.out.print("After the correction of correct message is : ");
            if(reciver_message_array[correct_parity_bit+1]==0)
            {
                  reciver_message_array[correct_parity_bit+1]=1;
            }
            else if(reciver_message_array[correct_parity_bit+1]==1)
            {
                  reciver_message_array[correct_parity_bit+1]=0;
            }
            for(int i=0;i<reciver_message_array.length;++i)
            {
                  System.out.print(reciver_message_array[i]);
            }
      }

      public static void main(String args[])
      {
            int [] sender_message_array=new int[12];
            Arrays.fill(sender_message_array,0);
            System.out.print("Enter the sender message(for 8 bits only 0 or 1 s) : ");
            Scanner Sender_message=new Scanner(System.in);
            String sender_message=Sender_message.nextLine();
            int j=0;
            for(int i=0;i<sender_message_array.length;++i)
            {
                  if((i!=0)&&(i!=1)&&(i!=3)&&(i!=7))
                  {
                        if(sender_message.charAt(j)=='0')
                        {
                              sender_message_array[i]=0;
                              ++j;
                        }
                        else
                        {
                              sender_message_array[i]=1;
                              ++j;
```

```
                }
                if(j==sender_message.length())
                {
                        break;
                }
        }
}

int sender_message_word[]=sender_message_convert(sender_message_array);
System.out.print("Code word : ");
for(int i=0;i<sender_message_word.length;++i)
{
        System.out.print(sender_message_word[i]);
}

System.out.print("\nEnter the reciver message(for 12 bits only 0 or 1 s) : ");
Scanner Reciver_message=new Scanner(System.in);
String reciver_message=Reciver_message.nextLine();
int []reciver_message_array=new int[reciver_message.length()];
for(int i=0;i<reciver_message.length();++i)
{
        if(reciver_message.charAt(i)=='0')
        {
                reciver_message_array[i]=0;
        }
        else
        {
                reciver_message_array[i]=1;
        }
}
if(statement(reciver_message_array))
{
        System.out.println("Message send no error" );
}
else
{
        System.out.println("Message currepted" );
```

```
                correct(reciver_message_array);

            }

        }

}
```

```
import java.util.Scanner;
import java.util.Arrays;

public class HammingCode
{
    public static int[]sender_message_convert(int []sender_message_array)
    {
        int total=0;
        for(int i=0;i<3;++i)
        {
            int paritybit_position=(int)Math.pow(2,i)-1;
            if(i==0)
            {
                total=sender_message_array[2]+sender_message_array[4]+sender_message_array[6]+sender_message_array[8]+sender_message_array[10];
                if(total%2==0)
                {
                    sender_message_array[paritybit_position]=0;
                }
                else
                {
                    sender_message_array[paritybit_position]=1;
                }
            }
            else if(i==1)
            {
                total=0;
                total=sender_message_array[2]+sender_message_array[5]+sender_message_array[6]+sender_message_array[9]+sender_message_array[10];
                if(total%2==0)
                {
                    sender_message_array[paritybit_position]=0;
                }
                else
                {
                    sender_message_array[paritybit_position]=1;
                }
            }
            else if(i==2)
            {
                total=0;
                total=sender_message_array[4]+sender_message_array[5]+sender_message_array[6]+sender_message_array[11];
                if(total%2==0)
                {
                    sender_message_array[paritybit_position]=0;
                }
                else
                {
                    sender_message_array[paritybit_position]=1;
                }
            }
            else
            {
                total=0;
                total=sender_message_array[8]+sender_message_array[9]+sender_message_array[10]+sender_message_array[11];
                if(total%2==0)
                {
                    sender_message_array[paritybit_position]=0;
```

```
                {
                    sender_message_array[paritybit_position]=0;
                }
                else
                {
                    sender_message_array[paritybit_position]=1;
                }
            }
        }
        return  sender_message_array;
    }

    public static boolean statement(int []reciver_message_word)
    {
        int total=0,increment=0;
        for(int i=0;i<=3;++i)
        {
            int paritybit_position=(int)Math.pow(2,i)-1;
            if(i==0)
            {
                total=reciver_message_word[2]+reciver_message_word[4]+reciver_message_word[6]+reciver_message_word[8]+reciver_message_word[10];
                if(total%2==0)
                {
                    if(reciver_message_word[paritybit_position]==0)
                    {
                        ++increment;
                    }
                }
                else
                {
                    if(reciver_message_word[paritybit_position]==1)
                    {
                        ++increment;
                    }
                }
            }
            else if(i==1)
            {
                total=0;
                total=reciver_message_word[2]+reciver_message_word[5]+reciver_message_word[6]+reciver_message_word[9]+reciver_message_word[10];
                if(total%2==0)
                {
                    if(reciver_message_word[paritybit_position]==0)
                    {
                        ++increment;
                    }
                }
                else
                {
                    if(reciver_message_word[paritybit_position]==1)
                    {
                        ++increment;
                    }
                }
            }
            else if(i==2)
            {
                total=0;
                total=reciver_message_word[4]+reciver_message_word[5]+reciver_message_word[6]+reciver_message_word[11];
                if(total%2==0)
                {
                    if(reciver_message_word[paritybit_position]==0)
                    {
                        ++increment;
                    }
                }
                else
                {
                    if(reciver_message_word[paritybit_position]==0)
                    {
                        ++increment;
                    }
                }
            }
            else
            {
                total=0;
                total=reciver_message_word[8]+reciver_message_word[9]+reciver_message_word[10]+reciver_message_word[11];
                if(total%2==0)
                {
                    if(reciver_message_word[paritybit_position]==0)
                    {
                        ++increment;
                    }
                }
                else
                {
                    if(reciver_message_word[paritybit_position]==1)
                    {
                        ++increment;
                    }
                }
            }
        }
        if(increment==3)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    public static void correct(int [] reciver_message_array)
    {
        int total=0,correct_parity_bit=0;
        for(int i=0;i<=3;++i)
        {
            int paritybit_position=(int)Math.pow(2,i)-1;
            if(i==0)
```

```
            if(i==0)
            {
                total=reciver_message_array[2]+reciver_message_array[4]+reciver_message_array[6]+reciver_message_array[8]+reciver_message_array[10];
                if(total%2==0)
                {
                    if(reciver_message_array[paritybit_position]!=0)
                    {
                        correct_parity_bit+=paritybit_position;
                    }
                }
                else
                {
                    if(reciver_message_array[paritybit_position]!=1)
                    {
                        correct_parity_bit+=paritybit_position;
                    }
                }
            }
            else if(i==1)
            {
                total=0;
                total=reciver_message_array[2]+reciver_message_array[5]+reciver_message_array[6]+reciver_message_array[9]+reciver_message_array[10];
                if(total%2==0)
                {
                    if(reciver_message_array[paritybit_position]!=0)
                    {
                        correct_parity_bit+=paritybit_position;
                    }
                }
                else
                {
                    if(reciver_message_array[paritybit_position]!=1)
                    {
                        correct_parity_bit+=paritybit_position;
                    }
                }
            }
            else if(i==2)
            {
                total=0;
                total=reciver_message_array[4]+reciver_message_array[5]+reciver_message_array[6]+reciver_message_array[11];
                if(total%2==0)
                {
                    if(reciver_message_array[paritybit_position]!=0)
                    {
                        correct_parity_bit+=paritybit_position;
                    }
                }
                else
                {
                    if(reciver_message_array[paritybit_position]!=1)
                    {
                        correct_parity_bit+=paritybit_position;
                    }
                }
```

```
                }
            }
            else
            {
                total=0;
                total=reciver_message_array[8]+reciver_message_array[9]+reciver_message_array[10]+reciver_message_array[11];
                if(total%2==0)
                {
                    if(reciver_message_array[paritybit_position]!=0)
                    {
                        correct_parity_bit+=paritybit_position;
                    }
                }
                else
                {
                    if(reciver_message_array[paritybit_position]!=1)
                    {
                        correct_parity_bit+=paritybit_position;
                    }
                }
            }
        }
        System.out.println((correct_parity_bit+2) +" bit position is wrong ");
        System.out.print("After the correction of correct message is : ");
        if(reciver_message_array[correct_parity_bit+1]==0)
        {
            reciver_message_array[correct_parity_bit+1]=1;
        }
        else if(reciver_message_array[correct_parity_bit+1]==1)
        {
            reciver_message_array[correct_parity_bit+1]=0;
        }
        for(int i=0;i<reciver_message_array.length;++i)
        {
            System.out.print(reciver_message_array[i]);
        }
    }

    public static void main(String args[])
    {
        int [] sender_message_array=new int[12];
        Arrays.fill(sender_message_array,0);
        System.out.print("Enter the sender message(for 8 bits only 0 or 1 s) : ");
        Scanner Sender_message=new Scanner(System.in);
        String sender_message=Sender_message.nextLine();
        int j=0;
        for(int i=0;i<sender_message_array.length;++i)
        {
            if((i!=0)&&(i!=1)&&(i!=3)&&(i!=7))
            {
                if(sender_message.charAt(j)=='0')
                {
                    sender_message_array[i]=0;
                    ++j;
                }
```

```
                    ++j;
                }
                else
                {
                    sender_message_array[i]=1;
                    ++j;
                }
                if(j==sender_message.length())
                {
                    break;
                }
            }
        }

        int sender_message_word[]=sender_message_convert(sender_message_array);
        System.out.print("Code word : ");
        for(int i=0;i<sender_message_word.length;++i)
        {
            System.out.print(sender_message_word[i]);
        }

        System.out.print("\nEnter the reciver message(for 12 bits only 0 or 1 s) : ");
        Scanner Reciver_message=new Scanner(System.in);
        String reciver_message=Reciver_message.nextLine();
        int []reciver_message_array=new int[reciver_message.length()];
        for(int i=0;i<reciver_message.length();++i)
        {
            if(reciver_message.charAt(i)=='0')
            {
                reciver_message_array[i]=0;
            }
            else
            {
                reciver_message_array[i]=1;
            }
        }
        if(statement(reciver_message_array))
        {
            System.out.println("Message send no error" );
        }
        else
        {
            System.out.println("Message currepted" );
            correct(reciver_message_array);
        }
    }
}
```

With error

```
C:\Users\2021E075\OneDrive - University of Jaffna\netlab3\T2>java HammingCode
Enter the sender message(for 8 bits only 0 or 1 s) : 11001110
Code word : 011110001110
Enter the reciver message(for 12 bits only 0 or 1 s) : 011110001100
Message currepted
3 bit position is wrong
After the correction of correct message is : 010110001100
```

No error

```
C:\Users\2021E075\OneDrive - University of Jaffna\netlab3\T2>java HammingCode
Enter the sender message(for 8 bits only 0 or 1 s) : 11100010
Code word : 111011000010
Enter the reciver message(for 12 bits only 0 or 1 s) : 111011000010
Message send no error
```

3. Sliding window protocol (Go back N ARQ)

Sliding window protocols, each outbound frame contains a sequence number, ranging from 0 up to some maximum. The maximum is usually 2n - 1 so the sequence number fits exactly in an n-bit field. The essence of all sliding window protocols is that at any instant of time, the sender maintains a set of sequence numbers corresponding to frames it is permitted to send. These frames are said to fall within the sending window. Similarly, the receiver also maintains a receiving window corresponding to the set of frames it is permitted to accept. The sender's window and the receiver's window need not have the same lower and upper limits or even have the same size. In some protocols they are fixed in size, but in others they can grow or shrink over the course of time as frames are sent and received. Although these protocols give the data link layer more freedom about the order in which it may send and receive frames. In Go-Back-N Automatic Repeat Request, we can send several frames before receiving acknowledgments; we keep a copy of these frames until the acknowledgments arrive. If there is one frame k missing, the receiver simply discards all subsequent frames k+1, k+2, ..., sending no acknowledgments. So, the sender will retransmit frames from k onwards.

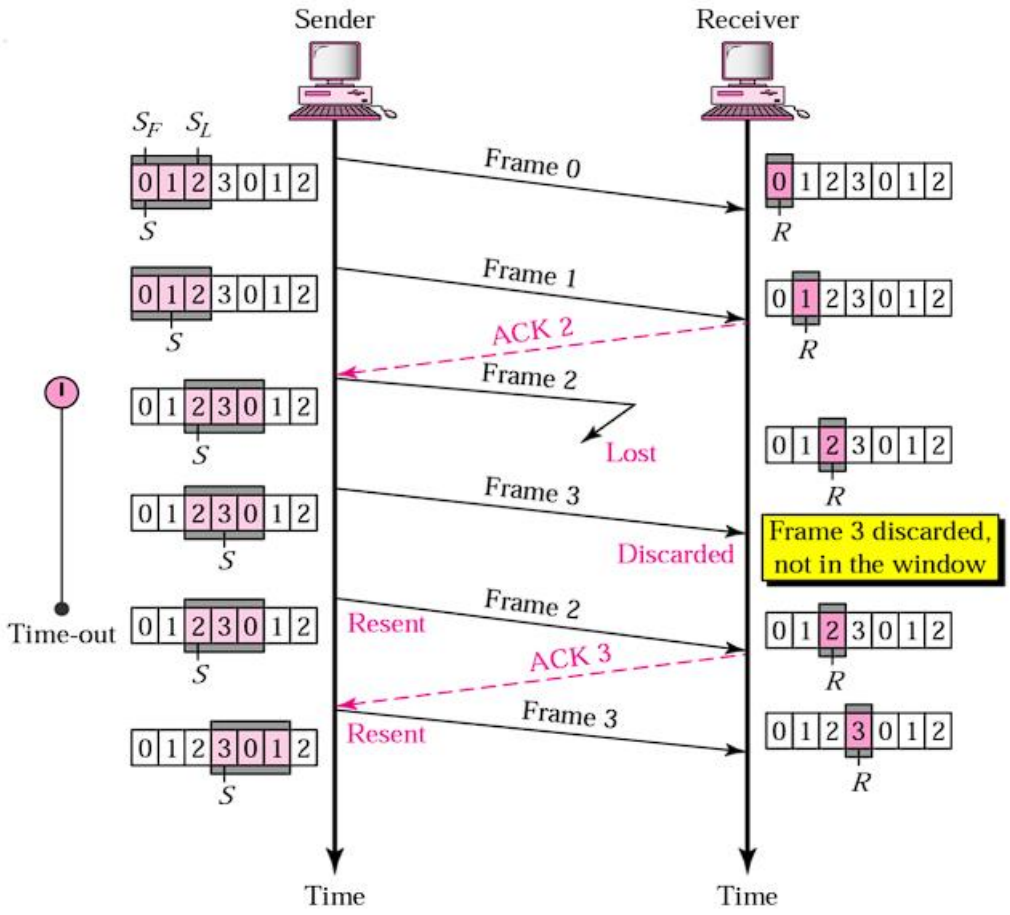

Figure 1: Go-Back-N ARQ

Algorithm:

1. Start the program.

2. Generate a random that gives the total number of frames to be transmitted.

3. Set the size of the window.

4. Generate a random number less than or equal to the size of the current window and identify the

number of frames to be transmitted at a given time.

5. Transmit the frames and receive the acknowledgement for the frames sent.

6. Find the remaining frames to be sent.

7. Find the current window size.

8. If an acknowledgement is not received for a particular frame retransmit the frames from that frame again.

9. Repeat the steps 4 to 8 till the number of remaining frames to be send becomes zero.

10. Stop the program.

Task 3: Write a program to perform simulation on sliding window (Go Back N) protocol.

```
import java.util.Scanner;
import java.util.Queue;
import java.util.LinkedList;

public class SWP
{
	// ACK - acknowlage
	public static void main(String args[])
	{
		System.out.print("Enter the number of frame you expect : "); //get a user input number of frame
		Scanner No_frame=new Scanner(System.in);
		int no_frame=No_frame.nextInt();

		Queue <Integer> frame_queue = new LinkedList <>();  //define as queue for the frame
		for(int i=0;i<no_frame;++i)
		{
			frame_queue.offer(i);
		}

		System.out.print("Enter the size of window : "); //get a user input size of window
		Scanner Windows_size=new Scanner(System.in);
		int windows_size=Windows_size.nextInt();

		int increment1=0;   //define increment
		for(;increment1<windows_size;++increment1)
		{
			System.out.println("Sending frame "+increment1 );
		}

		int increment2=0;
		while(!frame_queue.isEmpty()) //check whether all the frame are send or not
		{
			System.out.print("Enter the acknowlage number of reciver : "); //get a user input which Ack is recived
```

```
                    Scanner Acknowlage=new Scanner(System.in);
                    int acknowlage=Acknowlage.nextInt();
                    if(acknowlage==increment2)   //check whether correct acknowlage are recived
                    {
                        frame_queue.poll();
                        if(increment1<no_frame)
                        {
                            System.out.println("Sending frame "+increment1 );
                        }
                        ++increment1;
                        ++increment2;
                        continue;
                    }
                    else
                    {
                        int increment3=increment2; //if correct ACK not recive the agian previous frame send
                        for(int i=0;i<windows_size;++i)
                        {
                            System.out.println("Sending frame " + (increment3) );
                            ++increment3;
                        }
                    }
                }
                if(frame_queue.isEmpty()) //after the all frame send and all ACK recive then dispaly above message
                {
                    System.out.println("All the frams are transmited and all the ACK recived " );
                }
            }
}
```

```java
import java.util.Scanner;
import java.util.Queue;
import java.util.LinkedList;

public class SWP
{
    // ACK - acknowlage
    public static void main(String args[])
    {
        System.out.print("Enter the number of frame you expect : "); //get a user input number of frame
        Scanner No_frame=new Scanner(System.in);
        int no_frame=No_frame.nextInt();

        Queue <Integer> frame_queue = new LinkedList <>();  //define as queue for the frame
        for(int i=0;i<no_frame;++i)
        {
            frame_queue.offer(i);
        }

        System.out.print("Enter the size of window : "); //get a user input size of window
        Scanner Windows_size=new Scanner(System.in);
        int windows_size=Windows_size.nextInt();

        int increment1=0;   //define increment
        for(;increment1<windows_size;++increment1)
        {
            System.out.println("Sending frame "+increment1 );
        }

        int increment2=0;
        while(!frame_queue.isEmpty()) //check whether all the frame are send or not
        {
            System.out.print("Enter the acknowlage number of reciver : "); //get a user input which Ack is recived
            Scanner Acknowlage=new Scanner(System.in);
            int acknowlage=Acknowlage.nextInt();
            if(acknowlage==increment2)   //check whether correct acknowlage are recived
            {
                frame_queue.poll();
                if(increment1<no_frame)
                {
                    System.out.println("Sending frame "+increment1 );
                }
                ++increment1;
                ++increment2;
                continue;
            }
            else
            {
                int increment3=increment2; //if correct ACK not recive the agian previous frame send
                for(int i=0;i<windows_size;++i)
                {
                    System.out.println("Sending frame " + (increment3) );
                    ++increment3;
                }
            }
        }
        if(frame_queue.isEmpty()) //after the all frame send and all ACK recive then dispaly above message
        {
            System.out.println("All the frams are transmited and all the ACK recived " );
        }
    }
}
```

All the frams are transmitted and all the ACK received without any error

```
C:\Users\2021E075\OneDrive - University of Jaffna\netlab3\T3>javac SWP.java

C:\Users\2021E075\OneDrive - University of Jaffna\netlab3\T3>java SWP
Enter the number of frame you expect : 8
Enter the size of window : 4
Sending frame 0
Sending frame 1
Sending frame 2
Sending frame 3
Enter the acknowlage number of reciver : 0
Sending frame 4
Enter the acknowlage number of reciver : 1
Sending frame 5
Enter the acknowlage number of reciver : 2
Sending frame 6
Enter the acknowlage number of reciver : 3
Sending frame 7
Enter the acknowlage number of reciver : 4
Enter the acknowlage number of reciver : 5
Enter the acknowlage number of reciver : 6
Enter the acknowlage number of reciver : 7
All the frams are transmited and all the ACK recived
```

With some error occurred but after the correction error again All the frames are transmitted and all the ACK received

```
C:\Users\2021E075\OneDrive - University of Jaffna\netlab3\T3>java SWP
Enter the number of frame you expect : 8
Enter the size of window : 4
Sending frame 0
Sending frame 1
Sending frame 2
Sending frame 3
Enter the acknowlage number of reciver : 0
Sending frame 4
Enter the acknowlage number of reciver : 1
Sending frame 5
Enter the acknowlage number of reciver : 3
Sending frame 2
Sending frame 3
Sending frame 4
Sending frame 5
Enter the acknowlage number of reciver : 2
Sending frame 6
Enter the acknowlage number of reciver : 3
Sending frame 7
Enter the acknowlage number of reciver : 4
Enter the acknowlage number of reciver : 5
Enter the acknowlage number of reciver : 6
Enter the acknowlage number of reciver : 7
All the frams are transmited and all the ACK recived
```