

Rohit Kabadi

Professor Varghese

CSE 123

6 May 2010

Homework Assignment #2

1. HDLC Framing

- a. If we change the stuffing rule to stuff a zero after six consecutive ones, the framing rules will NOT work. For example, if the data frame is 010111111, the stuffing rule will change it to 0101111110. When the receiver gets the message with two flags, it will recognize the second flag within the data frame and strip the data as 01 without entering the actual second flag. The data is now completely different from the original.
- b. If we change the stuffing rule to stuff a 0 after a zero followed by five consecutive ones, the stuffing rule will NOT work. For example, if the data frame is 011111111110 (eleven ones), the stuffing rule will change it to 0111110111110. Again, the receiver will recognize the flag within the data frame and strip the data as 01111. The data is now completely different from the original.
- c. The protocol WILL work if we change the stuffing rule to stuff a 1 after a zero followed by six consecutive ones. This rule ensures that we will never have six consecutive ones. The closest we can get is having a data frame containing ...0111110... or ...01111110...
- d. The protocol WILL work if we change the stuffing rule to stuff a zero after 4 consecutive ones. It prevents us from ever having six or even five consecutive ones. This rule is inferior to the HDLC rule because we stuff more zeros than we need to. For example, if the data frame is 011110, the stuffing rule will change it to 0111100, adding an extra zero when we don't really need it.

2. CRC Polynomial View

- a. By multiplying $(x^8 + x^6 + x^2 + x + 1) * (x^3 + 1)$, we get $x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^2 + x + 1$ which results in 101101110111 in binary. Multiplying $(x^8 + x^6 + x^2 + x + 1)$ by the following polynomials result in undetected burst errors. There are 4 undetected burst errors.

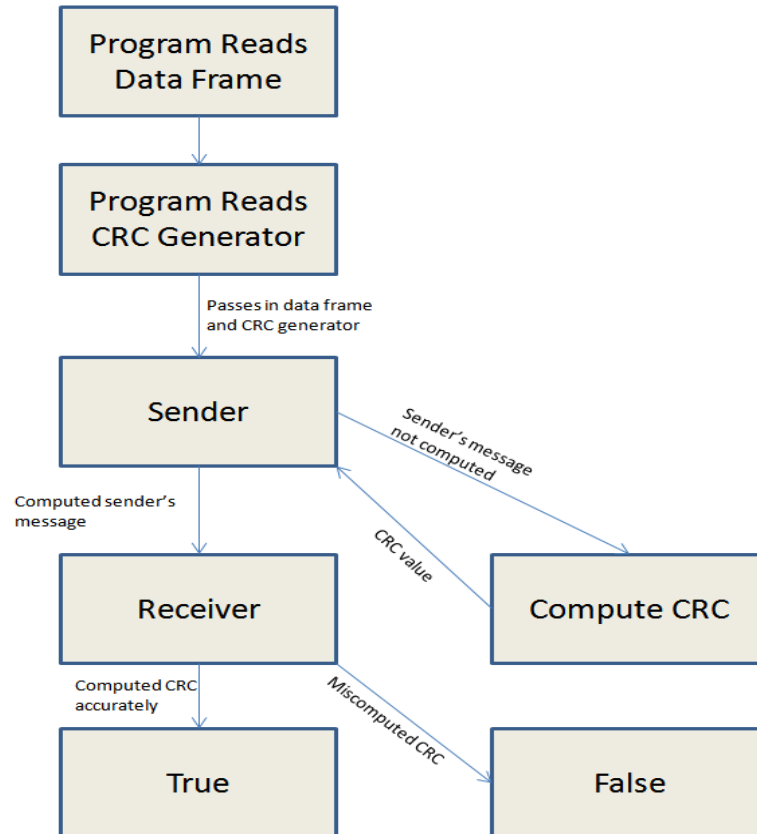
Polynomial	Binary
$x^3 + 1$	1001
$x^3 + x + 1$	1011
$x^3 + x^2 + 1$	1101
$x^3 + x^2 + x + 1$	1111

- b. Multiplying $x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^2 + x + 1$ by a polynomial whose highest power is x^2 and lowest power is 1 gives us all undetected burst errors at length 11. Two undetected burst errors formed by multiplying the original by $x^2 + 1$ or $x^2 + x + 1$

- c. With a burst length of n and CRC generator of power k , there are 2^{n-k-2} undetectable burst errors. Burst length n has optional terms for powers $n-2$ through 1 . The total number of possible burst length errors is 2^{n-2} . Therefore the probability of detecting a burst of arbitrary length is $\frac{2^{n-k-2}}{2^{n-2}} = 1/2^k$

3. CRC Program

- a. Program Data flow and description



The program written in Java reads a data frame from file and CRC generator from user input, shifts the data frame according to the length of the generator to form the groundwork of the message, and passes both values into the sender. The sender then calls the compute CRC method on the message and generator to calculate the CRC and adds the value to the message. The CRC is computed using java's primitive long to continuously XOR the data frame with the generator's value or zero similar to long division. I decided to use java long values so I could XOR all bits at once and shift them accordingly using built in bitwise operations. The CRC ends up as the remainder of all the operations. The receiver then takes the sender's message and generator and computes the CRC. If the value is zero, then the receiver outputs true that the message was sent correctly. It will output false if the opposite.


```

/*
 * CRC Program
 */

import java.io.*;
import java.util.Scanner;

public class CRC {

    public static int find_message_length(long input) {
        int count = 0;
        while(input != 0) {
            input = input >> 1;
            count ++;
        }
        return count;
    }

    public static long compute_crc (long message, long gen) {
        int r = find_message_length(gen);
        int message_length = find_message_length(message);
        int shift = message_length - r;
        long shifted_message = message >> shift;
        long temp_message = shifted_message;
        long crc = temp_message ^ gen;
        while(shift > 0) {
            shift = shift - 1;
            shifted_message = message >> shift;
            temp_message = crc << 1;
            temp_message = temp_message | (shifted_message & 1);
            if(temp_message >> (r-1) < gen >> (r-1)) {
                crc = temp_message ^ 0;
            }
            else
                crc = temp_message ^ gen;
        }
        return crc;
    }

    public static long string_to_long(String input) {
        char[] arr = input.toCharArray();
        int len = arr.length - 1;
        long long_val = 0;
        for(int i = len; i >= 0; i--) {
            if(arr[i] == '1')
                long_val += Math.pow(2, len-i);
        }
        return long_val;
    }

    public static long char_array_to_long(char[] input) {
        int len = input.length - 1;
        long long_val = 0;
        for(int i = len; i >= 0; i--) {
            if(input[i] == '1')
                long_val += Math.pow(2, len-i);
        }
        return long_val;
    }

    public static long sender(long message, long gen) {
        return message + compute_crc(message, gen);
    }
}

```

```

public static boolean receiver(long message, long gen) {
    if(compute_crc(sender(message, gen), gen) == 0)
        return true;
    return false;
}

public static void error_check(long message, long gen) {
    char[] sender_message =
        Long.toBinaryString(sender(message, gen)).toCharArray();
    char[] temp = sender_message.clone();
    long temp_long;
    int true_count = 0;
    int false_count = 0;
    int b1, c1, d1;
    for(int a = 36; a < 53; a++) {
        for(int b = a+1; b < 54; b++) {
            for(int c = b+1; c < 55; c++) {
                for(int d = c+1; d < 56; d++) {
                    if(temp[a] == '1')
                        temp[a] = '0';
                    else temp[a] = '1';
                    if(temp[b] == '1')
                        temp[b] = '0';
                    else temp[b] = '1';
                    if(temp[c] == '1')
                        temp[c] = '0';
                    else temp[c] = '1';
                    if(temp[d] == '1')
                        temp[d] = '0';
                    else temp[d] = '1';

                    temp_long = char_array_to_long(temp);
                    if(receiver(temp_long,gen)) {
                        true_count++;
                    }
                    else false_count++;

                    temp = sender_message.clone();
                }
            }
        }
    }
    System.out.println("TRUE: " + true_count);
    System.out.println("FALSE: " + false_count);
}

public static char[] flip_data_bits(char[] data) {
    char[] flipped_data = new char[data.length];
    for(int i = 0; i < data.length; i++) {
        if(data[i] == '1')
            flipped_data[i] = '0';
        else flipped_data[i] = '1';
    }
    return flipped_data;
}

public static void main (String[] args) {
    long data = 0;
    long gen = 0;
    long senders_message = 0;
    Scanner scanner = new Scanner(System.in);
}

```

```

System.out.print("Enter message input file name: ");
String filename = scanner.nextLine();

try {
    FileReader fr = new FileReader(filename);
    BufferedReader br = new BufferedReader(fr);
    data = string_to_long(br.readLine());
}
catch (IOException ex) {
    System.out.println("File does not exist");
}
System.out.print("Enter value for generator: ");
String gen_input = scanner.nextLine();
gen = string_to_long(gen_input);
int r = find_message_length(gen);
long message = data << (r-1);
long crc = compute_crc(message, gen);
senders_message = sender(message, gen);
System.out.println("CRC Value = " + Long.toBinaryString(crc));
System.out.println("Sender's Message = " +
    Long.toBinaryString(senders_message));
System.out.println("Receiver says CRC is " + receiver(message, gen));
System.out.println("The results of with the simulated bit errors:");
error_check(senders_message, gen);
long flipped_data =

char_array_to_long(flip_data_bits(Long.toBinaryString(data).toCharArray()));
System.out.println("Flipping the data bits gives us " +
    Long.toBinaryString(flipped_data));
System.out.println("Receiver says CRC of flipped bits is " +
    receiver(message,gen));
}

```