EECS 3150

Data Communication

Project 1

Milestone 2

J Andrew McCormick

11/12/2015

**Table of Contents**

How to Run ……………………………………………………………………………………… 1

Results …………………………………………………………………………………………… 1

**How to Run**

In order to run this program, you must follow the following steps:

1. Using two terminal windows, navigate to the root directory in each window.

2. Run make from the root directory (where the makefile is located).

3. In one of the windows type ./rcvr and press enter (this will open up the receiver where it waits for something to be transmitted) (the receiver and transmitter operate on the localhost at port 5050).

4. In the other window, type ./xmtr and press enter.

5. In the window you ran ./xmtr, you will be prompted to enter the file name of the file you wish to transmit.

6. After that, you will be asked how you would like to transmit the file. 0 for no modification, 1 for CRC error detection, and 2 for Hamming error correction/detection

7. Finally, the transmitter will ask you the MAXIMUM of errors you would like to produce in each frame

**Results**

There is currently a module in place that forces a random number of errors in random places and random frames.

**Error Creation Module**

The user inputs the maximum number of errors they would like to produce in each frame. This means that if the user enters 10, there will either be 0,1,2,3,4,5,6,7,8,9, or 10 errors in each frame.

**CRC**

If you choose CRC as your error detection/correction scheme, the transmitter will output the location of the errors that are introduced into the message. The receiver, upon failing to pass the CRC will simply inform you that the CRC failed on that frame. If the CRC passes, the transmitted message will be printed out.

**Hamming**

If you chose Hamming as your error detection/correction scheme, the transmitter will once agin output the location of the errors that are introduced into the message. The receiver will take this message, calculate a syndrome, correct the errors, inform the user of the errors that are corrected including the frame, byte, and bit location of the error and output the message. If in the message there is the sequence of characters #PCF# output to the console, it implies that that particular byte of data had a double error. When the receiver corrected a bit that it thought was in error, it corrected the wrong one.

**Testing**

For testing, I used the Hamlet.txt file included. The receiver outputs to the console the results of each frames message as well as the errors that are corrected.