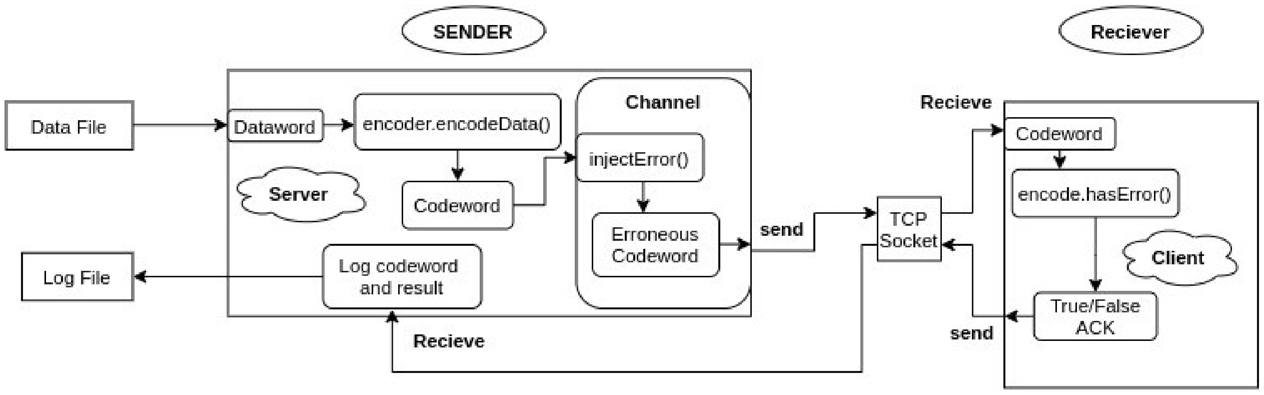
Mohiuddin Mondal
Roll- 001910501043 (A2)
Computer Networks
3rd year 1st Sem BCSE UG
Assignment 1

Problem statement: Design and implement an error detection module.

Description: Design and implement an error detection module which has four schemes namely LRC, VRC, Checksum and CRC. The Sender program should accept the name of a test file (contains binary strings) from the command line. Then it will prepare the data frame from the input. Based on the schemes, codeword will be prepared. Sender will send the codeword to the Receiver. Receiver will extract the dataword from codeword and show if there is any error detected.

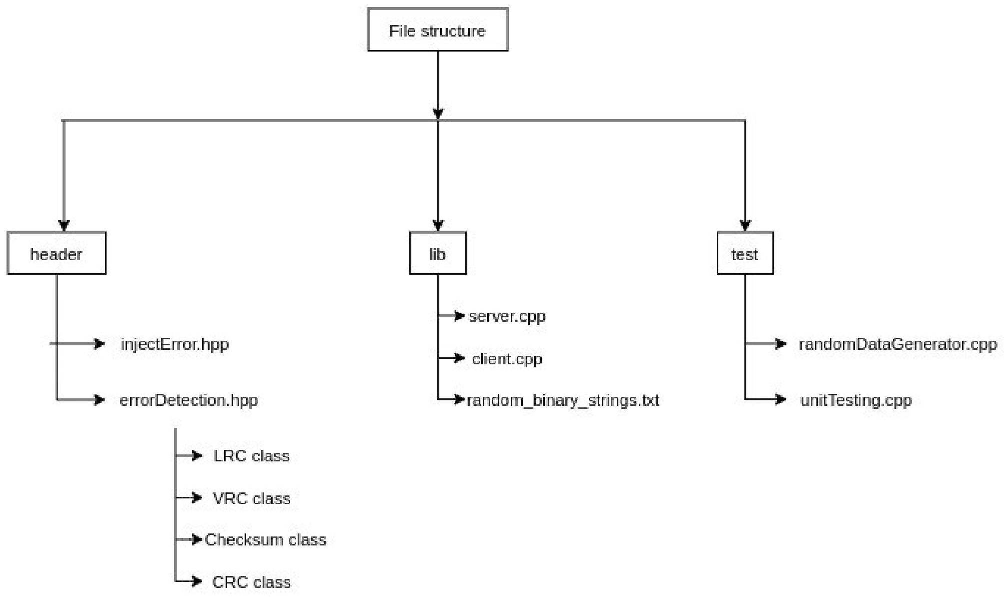
Design: Network communication is simulated through client-server . Programms are written using C++ . Random 64bit binary strings are generated using another program and stored in a text file. Server takes each string, encodes it with one scheme, injects error manually, and sends to client. Client checks for the error using the same scheme and returns true/false. Server logs the returned result in a log file “log_report” . Client and server exchanges these data through a socket.



Drawback: No separate channel program is used. Therefore injectError function is called within the server program.

Input/Output : 64 bit binary strings are used as data. All the schemes (LRC,VRC,CRC,Checksum) generates respective codewords. Client program returns true/false bsed on error is detected or not by those corresponding scheme.

Implementation:



Server program

main() : This function starts the server and starts sending enoded data. It waits for each data untill client replies with true/false. Two other function are defined as server task to accomplish it in two different manner.

server_task() : This function first randomly generates an error pattern (simply , at which positions of dataword, bitflip will occur). Then applies it on each codeword after encoding. Thus, only the dataword section changes but redundancy bits remains same. Also, we can compare between different schemes for same error.

server_task_randomError() : In real time scenario, error can occur in both dataword and redundancy bits, or the whole codeword can be erroneous. This is simulated here. After encoding with each scheme, whole codeword is infected with error randomly. Consecuently , accuracy is less than previous case.

Client program

main() : It recieves data and check if error occurs. Server sends 4 codeword with encoding scemes LRC,VRC,Checksum,CRC respectively. Client verifies data with respective schemes and returns true/false based on validation.

injectError.hpp
“random_device” of random library is used to generate the random numbers in this file.

injectErrorRandom() : This method injects error in whole codeword randomly. Both the dataword and redundancy bits are affected. Atmost 30% error is injected in the data. It also returns total number of   erroneous bit through argument reference.

newErrorPattern() : This method genearates a 64 bit binary string(same size as dataword) as error pattern. 1 in this string means bit flip will occur at that position in the dataword. i.e. “1001” means bit flip will be applied at positon 1 and 4 in dataword.

injectErrorDaaword() : This method takes codeword and a pattern string genearated by previous method. Then applies it on the dataword part and returns.

errorDetection.hpp

LRC class : It assumes that dataword length is multiple of 4. Then it divides dataword in 4 equal binary strings and genearates parity row. Adds this parity row at the end of data and returns. It has a hasError() method returns that takes a codeword and returns true if any error is detected.

VRC class: It checks for even parity and adds that parity bit at the end of dataword. It also contains hasError() method to check error in codeword.

Checksum class: It divides the binary data into 16 bit chunks and adds them to genearte checksum. Padds 1’s complement of the result at the end and returns the codeword.

CRC class: It uses CRC-16-CCITT polynomial ($x^{16}+x^{12}+x^{5}+1$). Adds the remainder at the end and returns the codeword. HasError method check for any error and returns the result.

Test cases:

50,000 binary string of size 64 bit is generated to test the program. Error is injected at runtime in the codeword using above mentioned methods. Also two type of test is done:

1. Error is injected in only the dataword, redundancy bits remains unaffected. Results in this case are logged in the file log_report.txt .
2. Error is injected in the whole codeword. Results are logged in the file detailed_report.txt .

In both cases, original data, encoded data, erroneous data and their result is logged. Final status conveys overall result on that data for LRC,VRC,Checksum,CRC respectively. Or, 1010 status means, lrc detected error, VRC couldn’t, Checksum detected, and CRC couldn’t.

Special cases:

Error detected by all four schemes:

```
DATA:                   0010011000111111100001010100110010010101010110101110100001010001
Bit Flips:              5
Error pattern:          0000010000000001000000000000000000000000000000000000010011000000
LRC:                    00100110001111111000010101001100100101010101101011101000010100011101111001111000
Erroneous LRC:          00100010001111101000010101001100100101010101101011101100100100011101111001111000
Error Detected:         True
VRC:                    00100110001111111000010101001100100101010101101011101000010100010
Erroneous VRC:          00100010001111101000010101001100100101010101101011101100100100010
Error Detected:         True
Checksum:               00100110001111111000010101001100100101010101101011101000010100011101011011000111
Erroneous Checksum:     00100010001111101000010101001100100101010101101011101100100100011101011011000111
Error Detected:         True
CRC:                    00100110001111111000010101001100100101010101101011101000010100010001010110111111
Erroneous CRC:          00100010001111101000010101001100100101010101101011101100100100010001010110111111
Error Detected:         True
```

Error is detected by checksum but not by CRC:

```
DATA:                   0011100110000101010101001101100101011011001110001010001010011101
Bit Flips:               4
Error Pattern:          0000000000000000000000010001000000100001000000000000000000000000
LRC:                    00111001100001010101010011011001010110110011100010100010100111011001010011111001
Erroneous LRC:          00111001100001010101010111001001011110100011100010100010100111011001010011111001
Error Detected:         True
VRC:                    00111001100001010101010011011001010110110011100010100010100111011
Erroneous VRC:          00111001100001010101010111001001011110100011100010100010100111011
Error Detected:         False
```

```
Checksum:               00111001100001010101010011011001010110110011100010100010100111010111001111001011
Erroneous Checksum:     00111001100001010101010111001001011110100011100010100010100111010111001111001011
Error Detected:         True
CRC:                    00111001100001010101010011011001010110110011100010100010100111010000101000001000
Erroneous CRC:          00111001100001010101010111001001011110100011100010100010100111010000101000001000
Error Detected:         False
```

Reuslts and Analysis:

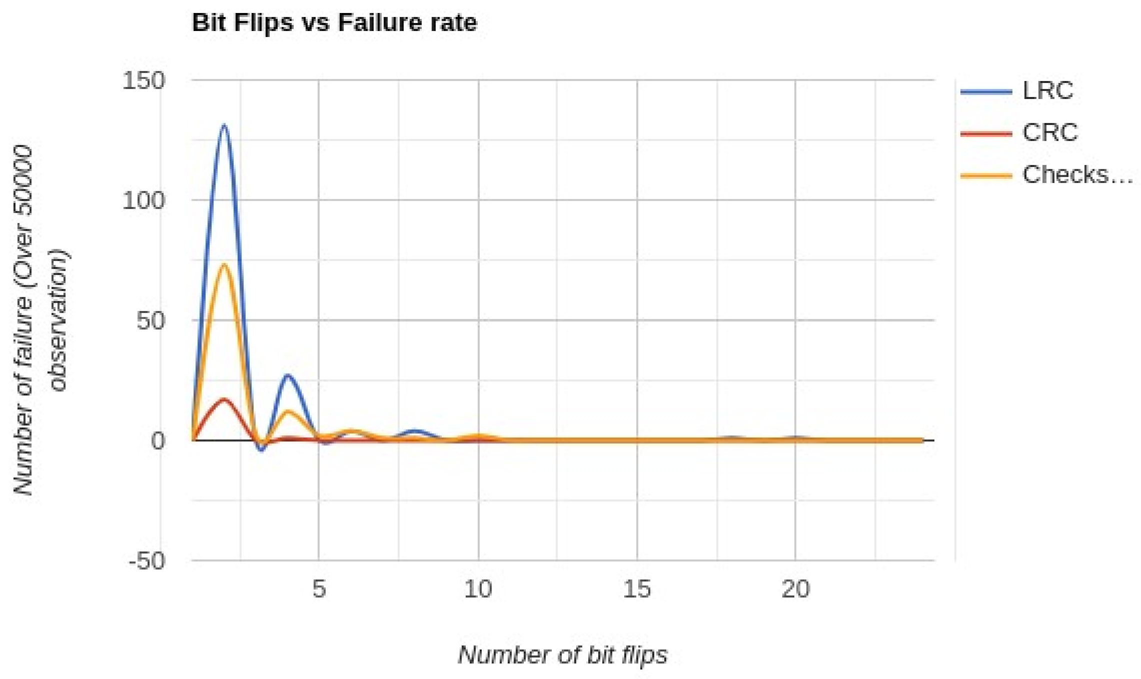
Results are based on 50,000 testcases in each category.

When only dataword in erroneous:

LRC accuracy : 99.724 % or failure rate: **0.276%**
VRC accuracy: 51.6111% or failure rate: **48.3889%**
Checksum accuracy: 99.858% or failure rate: **0.142%**
CRC accuracy: 99.996% or failure rate: **0.004%**

When whole codeword is erroneous:

LRC accuracy: 99.708% or failure rate: **0.292%**
VRC accuracy: 49.684% or failure rate: **50.316%**
Checksum accuracy: 99.824% or failure rate: **0.176%**
CRC accuracy: 99.938% or failure rate: **0.062%**



Conclusion:

From above results, we can see that CRC is more sensitive to error than rest of the schemes.