ARKOJYOTI SEN

Roll: 001810501037
BCSE UG III
Date: 02.11.2020

Computer Networks

Assignment-1

Overview

Design and implement an error detection module namely LRC, VRC, Checksum and CRC.

Goals

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 a sequence of 0,1) from the command line. Then it will prepare the data frame (decide the size of the frame) from the input. Based on the schemes, codeword will be prepared. Sender will send the codeword to the Receiver. Receiver will extract the data word from the codeword and show if there is any error detected. Test the same program to produce a PASS/FAIL result for following cases.

- Error is detected by all four schemes. Use a suitable CRC polynomial
- Error is detected by checksum but not by CRC.
- Error is detected by VRC but not by CRC.

Specifications

The code has been written in **python**. The input is being taken from a file, user has the freedom to select the frame size, the block size and even the CRC polynomial (by default it has kept **CRC 4 ITU**). We can create single bit errors and burst errors at **random**. The generated error is then detected by any of the error detection module chosen by the user.

Code

main.py

main.py will take the filename from which input is being taken. The frame size and the block size are also taken from user. Then encryption in all the four schemes are done. Then error is injected randomly, though the number of errors is taken as an input. Finally the errors are detected through any of the four error detection models.

decryption.py

It takes the input file, frame size, block size and generates the code for all the four schemes on the given input using some helper functions. It also writes the final frame to the output file that will be send to the receiver.

adderror.py

This function inserts a given number of errors in random locations throughout the packet. The number of erors is taken as input from user.

decryption.py

This function decodes the data word from the frame and does the error checking by any of the four schemes. It outputs whether the code is corrupted or not.

helper.py

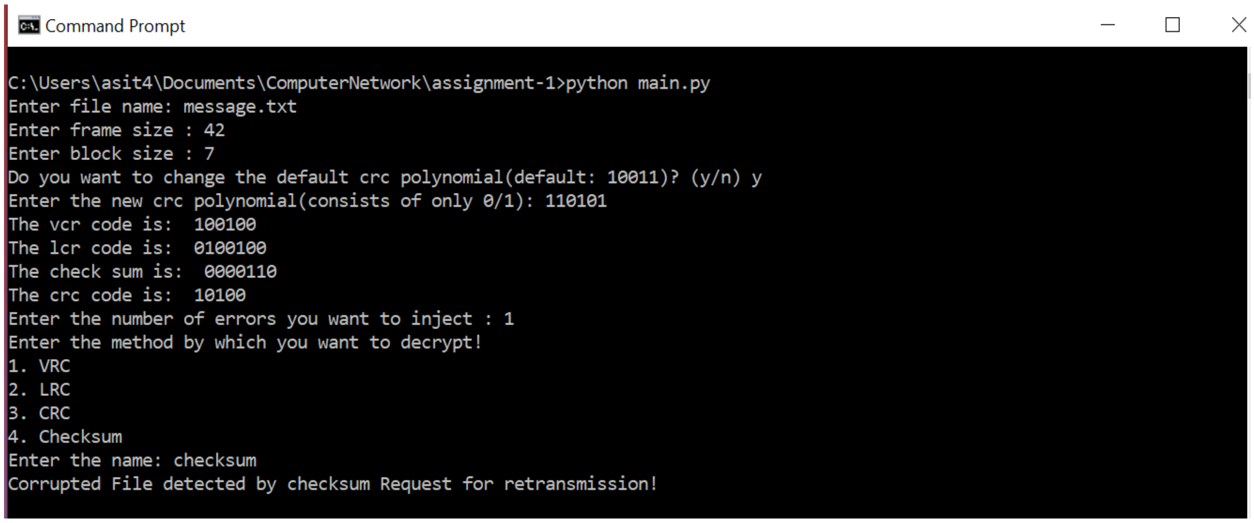
Contains all the code for implementing the different error detection schemes. Basically given the frame size, block size and the data word, it generates the code word for different schemes.

Other implementation details

If the input file has lesser bit values than frame size, we randomly generate the extra bits needed.

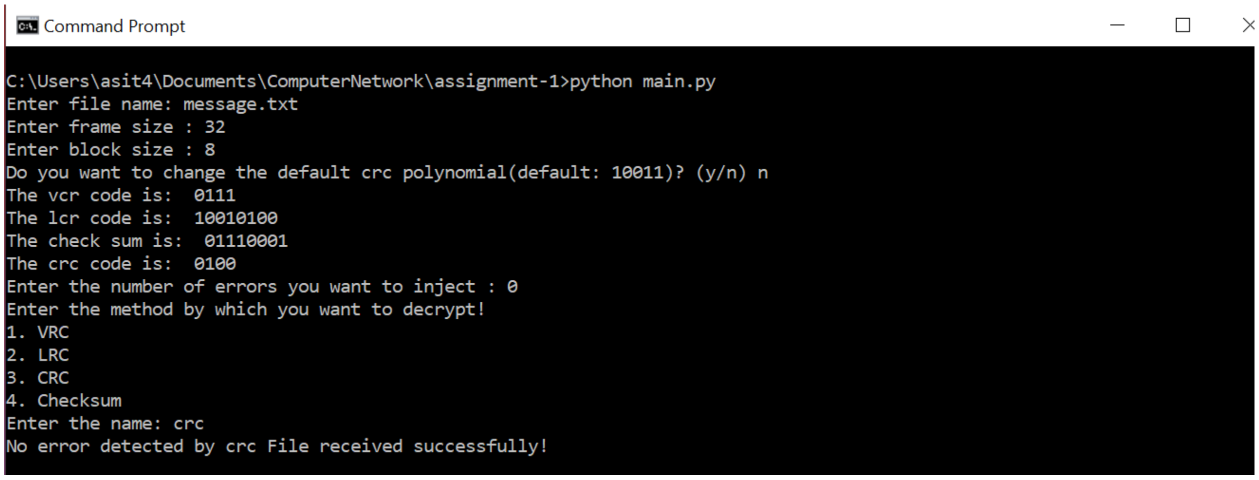
Procedure

Firstly, the filename is taken as an input. After entering the frame size, block size and CRC polynomial, the code for different schemes get generated. Errors are then added. Finally the user chooses a method by which the data word is then decrypted. The result of the error detection is printed out.



```
C:\Users\asit4\Documents\ComputerNetwork\assignment-1>python main.py
Enter file name: message.txt
Enter frame size : 42
Enter block size : 7
Do you want to change the default crc polynomial(default: 10011)? (y/n) y
Enter the new crc polynomial(consists of only 0/1): 110101
The vcr code is:  100100
The lcr code is:  0100100
The check sum is:  0000110
The crc code is:  10100
Enter the number of errors you want to inject : 1
Enter the method by which you want to decrypt!
1. VRC
2. LRC
3. CRC
4. Checksum
Enter the name: checksum
Corrupted File detected by checksum Request for retransmission!
```

Only one error was added and error was detected by the receiver.



```
C:\Users\asit4\Documents\ComputerNetwork\assignment-1>python main.py
Enter file name: message.txt
Enter frame size : 32
Enter block size : 8
Do you want to change the default crc polynomial(default: 10011)? (y/n) n
The vcr code is:  0111
The lcr code is:  10010100
The check sum is:  01110001
The crc code is:  0100
Enter the number of errors you want to inject : 0
Enter the method by which you want to decrypt!
1. VRC
2. LRC
3. CRC
4. Checksum
Enter the name: crc
No error detected by crc File received successfully!
```

No error was added and the message got successfully received.

Observation

- The error was generated at random, so I wasn’t able to find the cases where one detection technique failed and the other succeeded. But it can be done manually and can be proved that the question-mentioned scenarios are very possible. So, what I have done is encode the data word in all the four detection techniques.
- The checksum technique was very sensitive towards error.
- The CRC technique is also very much reliable. The efficiency of the CRC technique primarily depends on choosing the CRC polynomial. It also can be used to detect real life data transfer errors.

Results

Implementing all the four error detection schemes, it guarantees error are caught in almost 96-99% of the time. CRC 24 was better at checking errors than the default CRC ITU, the error rate goes down to almost 2-3% when using CRC 24.