Name- Srestha Kanjilal

Class- BCSE- III

Group- A3

Roll- 001810501078

Subject- Computer Networks

Assignment- 1

CO- 1

Topic- Design and implementation of an error detection module.

Submission Data- 05.11.20

Problem Statement:

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 dataword from codeword and show if there is any error detected.

Test the same program to produce a PASS/FAIL result for following cases.

(a) Error is detected by all four schemes. Use a suitable CRC polynomial.

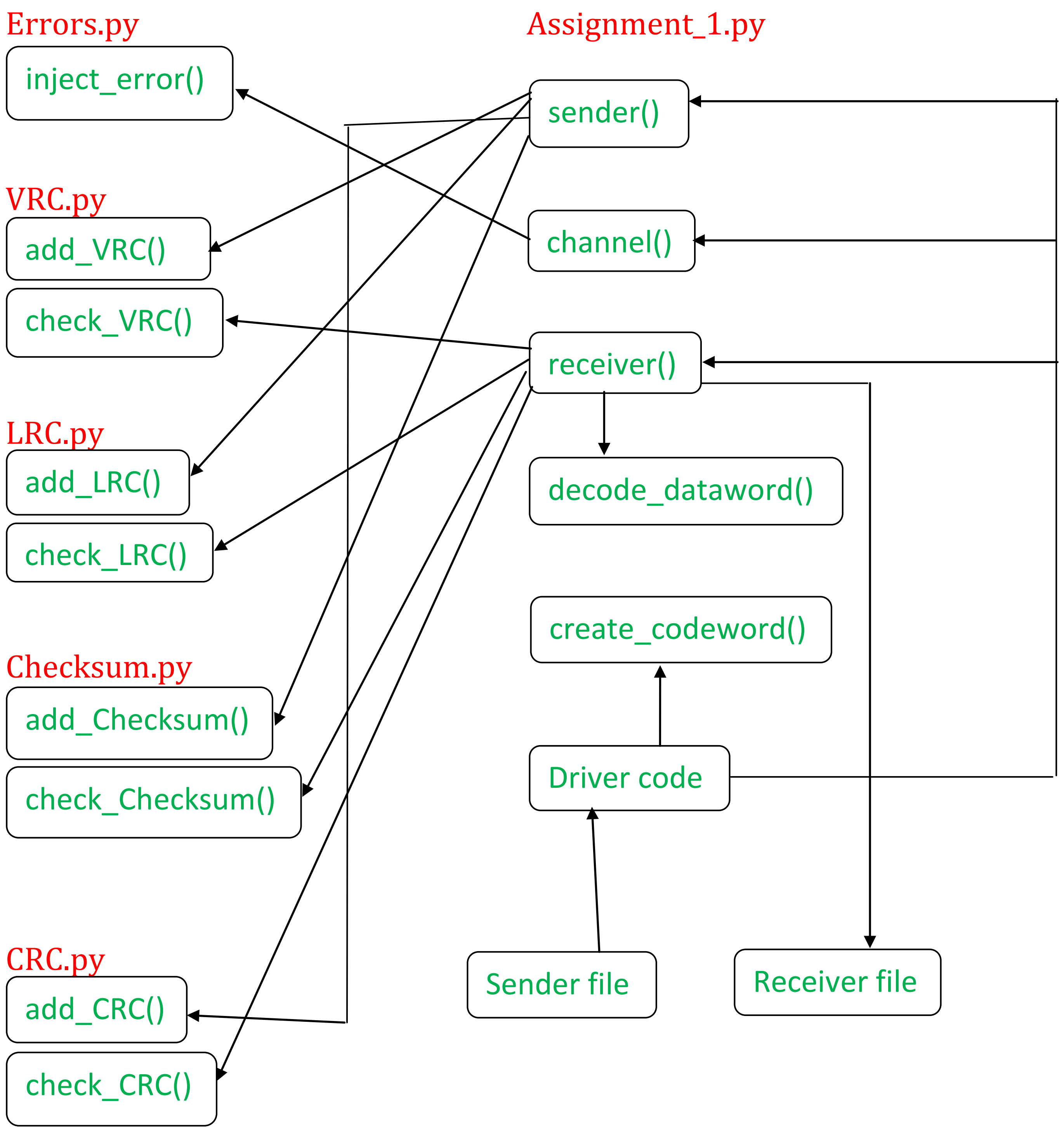
(b) Error is detected by checksum but not by CRC.

(c) Error is detected by VRC but not by CRC

Design:

Program Description - The purpose of the programs is to detect errors during transmission from sender to receiver using 4 different error detection schemes namely LRC (Longitudinal Redundancy Check), VRC (Vertical Redundancy Check), Checksum and CRC (Cyclic Redundancy Check). Initially a set of characters is read from the sender file and converted into binary format to form the dataword. In the sender program, redundant bits for all these schemes are added to the dataword to form the codeword. Consequently, in the channel program, random number of errors have been injected at random positions in the codeword after user's consent and the resulting codeword is passed on to the receiver. In the receiver program the codeword is checked using all 4 schemes to determine the presence of any error. In case of an error, an output message requesting retransmission is displayed, otherwise the codeword is decoded, resulting characters are written in the receiver file and then the next set of characters is read from the sender file. The program runs until file transmission is complete.

Program Structure – Given below is the procedural organisation of the program



Input - The user has to input a string which is written into the sender file. The program reads from the file and transmits the set of characters.

Output – Firstly all the schemes are checked to detect errors. If no error is detected, the decoded characters are written in the receiver file, else a message requesting retransmission is displayed.

Implementation:

In file Assignment_1.py-------

Driver code - User inputs a string which is written into sender file. Program then reads a set of characters from the file and prepares to transfer it to the receiver file by calling the functions sender, channel, receiver.

create_codeword – Set of characters passed to it is converted to a string of 0s and 1s (dataword)

sender – Inside the function, add_VRC( ), add_CRC( ), add_Checksum( ) and add_LRC( ) functions are called to add redundant characters to the dataword to form the codeword.

channel – User's consent is taken before function to inject errors into the codeword is called.

receiver – Inside the function, check_VRC( ), check_LRC( ), check_CRC( ) and check_Checksum( ) functions are called to check for errors in the codeword. If no error is detected, decode_dataword( ) is called, else error message requesting retransmission is displayed.

decode_dataword – The dataword is decoded into a set of characters which is then written into the receiver file.

In file Errors.py-------

inject_error - Random number of errors is injected at random positions in the codeword and the resulting codeword is returned.

In file LRC.py-------

add_LRC( ) – Redundant characters are added as per the LRC scheme.

check_LRC( ) – The codeword is checked for errors using the LRC scheme and message is displayed accordingly.

In file VRC.py-------

add_VRC( ) – Redundant characters are added as per the VRC scheme.

check_VRC( ) – The codeword is checked for errors using the VRC scheme and message is displayed accordingly.

In file CRC.py-------

add_CRC( ) – Redundant characters are added as per the CRC scheme.

check_CRC( ) – The codeword is checked for errors using the CRC scheme and message is displayed accordingly.

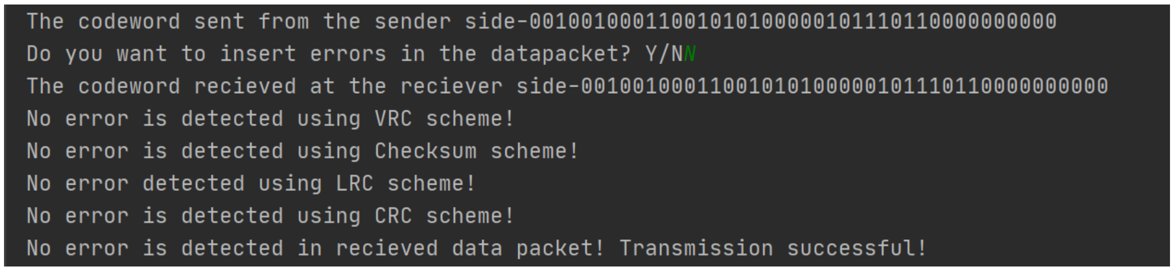
In file Checksum.py-------

add_Checksum( ) – Redundant characters are added as per the Checksum scheme.

check_Checksum( ) – The codeword is checked for errors using the Checksum scheme and message is displayed accordingly.

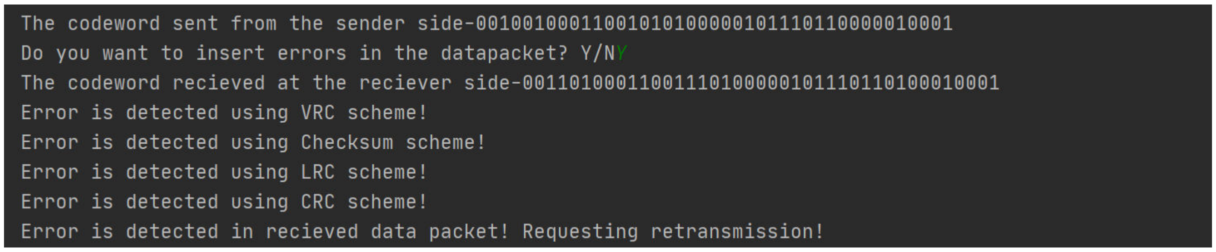
Test cases (using CRC-8):

1. No error is injected and no error is detected



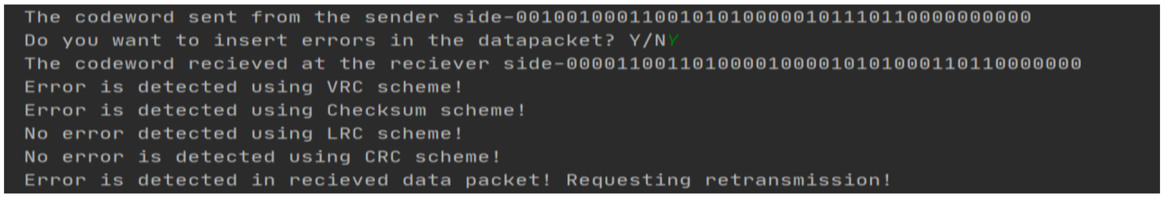
```
The codeword sent from the sender side-00100100011001010100000101110110000000000
Do you want to insert errors in the datapacket? Y/NN
The codeword recieved at the reciever side-00100100011001010100000101110110000000000
No error is detected using VRC scheme!
No error is detected using Checksum scheme!
No error detected using LRC scheme!
No error is detected using CRC scheme!
No error is detected in recieved data packet! Transmission successful!
```

2. Error is injected and detected by all 4 schemes, VRC, Checksum, LRC, CRC



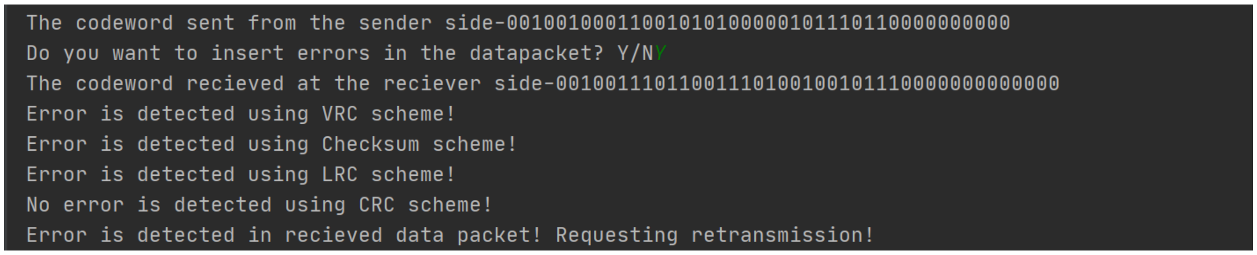
```
The codeword sent from the sender side-00100100011001010100000101110110000010001
Do you want to insert errors in the datapacket? Y/NY
The codeword recieved at the reciever side-00110100011001110100000101110110100010001
Error is detected using VRC scheme!
Error is detected using Checksum scheme!
Error is detected using LRC scheme!
Error is detected using CRC scheme!
Error is detected in recieved data packet! Requesting retransmission!
```

3. Error is injected and detected by VRC, Checksum but not LRC, CRC



```
The codeword sent from the sender side-00100100011001010100000101110110000000000
Do you want to insert errors in the datapacket? Y/NY
The codeword recieved at the reciever side-00001100110100001000010101000110110000000
Error is detected using VRC scheme!
Error is detected using Checksum scheme!
No error detected using LRC scheme!
No error is detected using CRC scheme!
Error is detected in recieved data packet! Requesting retransmission!
```

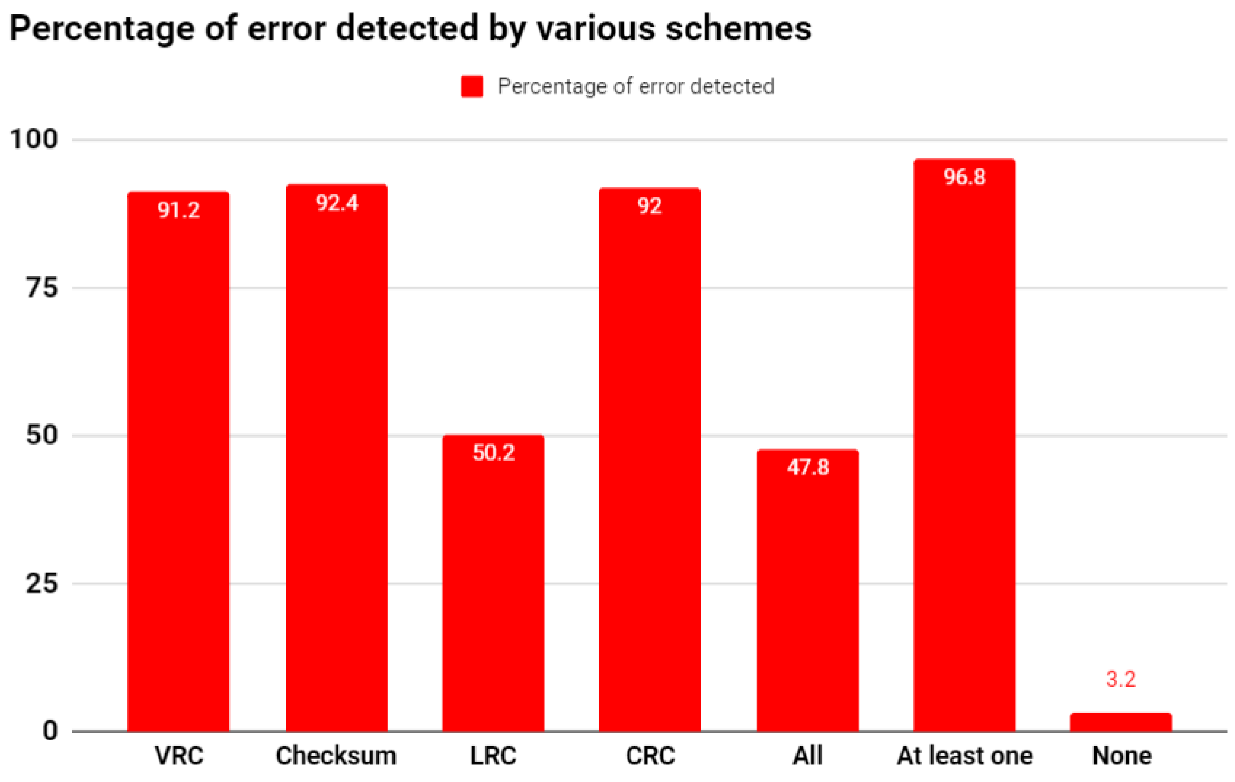
4. Error is injected and detected by VRC, Checksum, LRC but not CRC



```
The codeword sent from the sender side-00100100011001010100000101110110000000000
Do you want to insert errors in the datapacket? Y/NY
The codeword recieved at the reciever side-00100111011001110100100101110000000000000
Error is detected using VRC scheme!
Error is detected using Checksum scheme!
Error is detected using LRC scheme!
No error is detected using CRC scheme!
Error is detected in recieved data packet! Requesting retransmission!
```

Results:

Type of scheme	VRC	Checksum	LRC	CRC	By all schemes	By at least one scheme	By no schemes
Avg. No. of errors detected (out of 500 runs)	456	462	251	460	239	484	16
Detection %	91.2	92.4	50.2	92.0	47.8	96.8	3.2



Conclusion:

- VRC scheme is able to detect errors with efficiency 92.0%
- Checksum scheme is able to detect errors with efficiency 92.4%
- LRC scheme is able to detect errors with efficiency 50.2%
- CRC scheme is able to detect errors with efficiency 92.0%
- In 47.8% cases, all the schemes detected errors
- In 96.9% cases, at least one of the schemes detected errors.

Analysis:

Error detection capabilities of the code is increased significantly when all 4 schemes are used. Approximately 3.2% of the time, error is not detected by any of the schemes.

The errors introduced are random and erratic, so the results obtained are subject to experimental errors. There can be cases where a same bit is flipped an even number of times and hence no error is introduced by the inject_error( ) function. Such cases have been ignored and the program assumes that the function is indeed able to introduce errors in all codewords.

Comments:

The assignment has helped in understanding and implementing the various error detection schemes available. I have analysed the performance of each of the schemes after introducing random errors. The difficulty level was moderate.