Computer Networks Lab

Assignment Solution Report

Submitted by,

Name :	Avraneel Pal
Roll:	002010501047
Assignment:	1
Group:	A2
Semester:	3rd Year Semester 1
Year:	2020-24

Problem Statement:

Design and implement an error detection module.

Assignment 1: Error detection

Design:

Solution Approach:

Two files, '*sender.txt*' and '*receiver.txt*' were used to put the input and output test cases separately. Another file, '*encode.py*' contained the encoding algorithms for all four schemes, Vertical Redundancy Check (VRC), Longitudinal Redundancy Check (LRC), Cyclic Redundancy Check (CRC) and Checksum.

For Vertical Redundancy Check and Cyclic Redundancy Check, we assume the entire file contents as 1 whole frame. For Longitudinal Redundancy Check and Checksum algorithms, we divide the file contents into frames of size four each.

For adding noise, we first choose how many bits get changed by noise by picking a random integer, and then change that many bits by choosing them at random using random.sample() method.

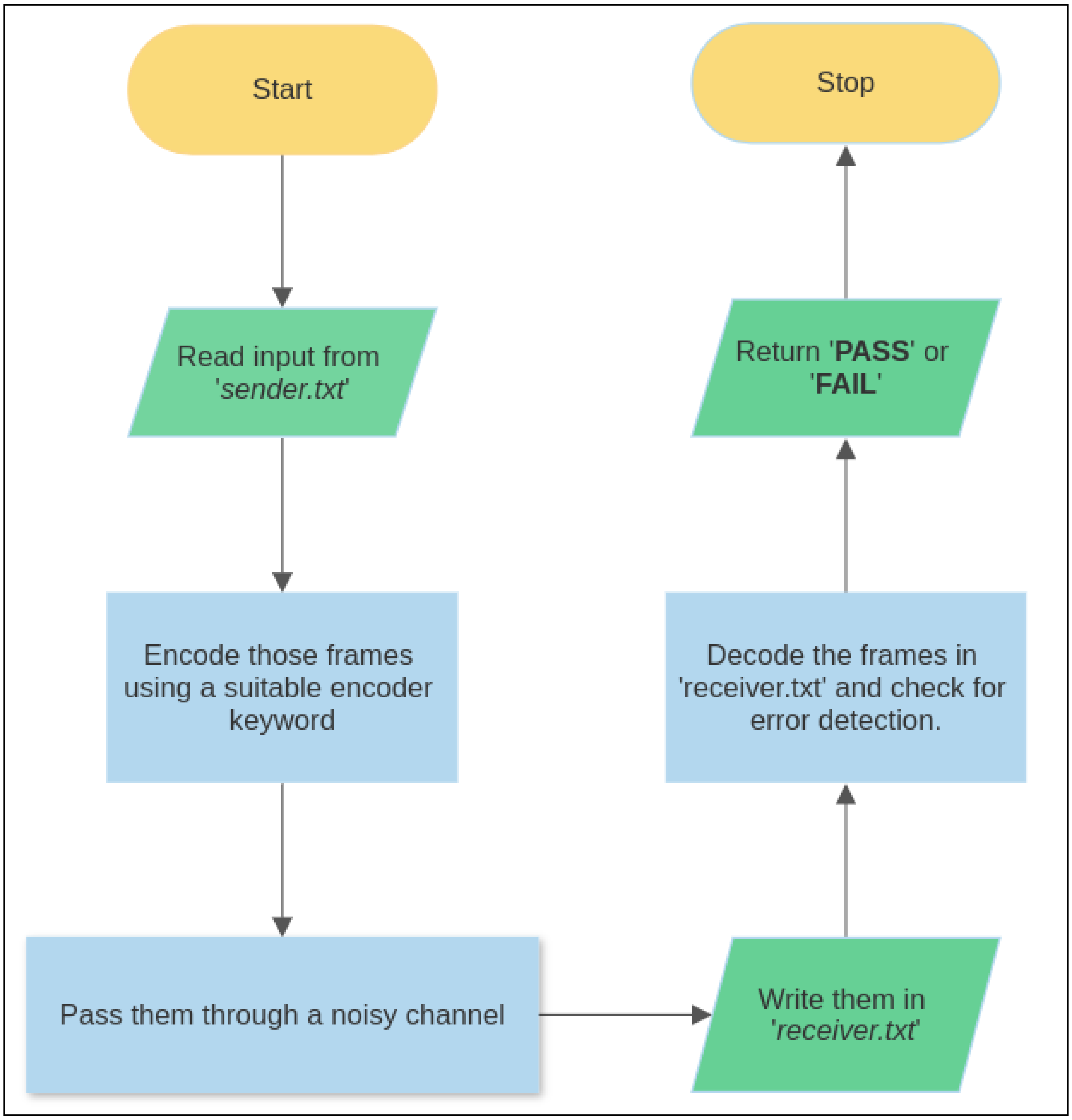
The file '*decoder.py*' decodes the messages from 'receiver.txt' and then uses the corresponding algorithms to detect any errors.

And finally the program is implemented in '*main.py*'

Methods for binary division and warped sum are also implemented.

CRC Polynomial used: CRC-10. That is, $x^{10} + x^{9} + x^{5} + x^{4} + x + 1$ or 11000110011.

Diagram for how one error detection algorithm is implemented:


Start
Read input from 'sender.txt'
Encode those frames using a suitable encoder keyword
Pass them through a noisy channel
Write them in 'receiver.txt'
Decode the frames in 'receiver.txt' and check for error detection.
Return 'PASS' or 'FAIL'
Stop

Input format:

A text file '*sender.txt*' containing binary bits.

Output format:

For each time the method *execute()* is called:

Testing case: <*case name*>

Result: <'*PASS*'/'*FAIL*'>

Example:

Testng case: "Error detected by all four schemes."
Result: PASS
Testing case: "Error detected by checksum but not by CRC."
Result: FAIL
Testing case: "Error detected by VRC but not by CRC."
Result: FAIL

Note: *Solely for evaluating performance over a large number of calls, we put execute() over a loop.*

Implementation:

Method description:

1. File: *encoder.py*
    a. **encode_vrc(frame):** Encodes a frame using the VRC algorithm.
    b. **encode_lrc(frame_list):** Encodes a list of frames using the LRC algorithm.
    c. **encode_crc(frame, divisor):** Encodes a frame using the CRC algorithm using a suitable divisor.
    d. **encode_checksum(frame_list):** Encodes a list of frames using the Checksum algorithm.

2. File: *decoder.py*
    a. **decode_vrc(frame):** Decodes a frame using the VRC algorithm.
    b. **decode_lrc(frame_list):** Decodes a list of frames using the LRC algorithm.
    c. **decode_crc(frame, divisor):** Decodes a frame using the CRC algorithm using a suitable divisor.
    d. **decode_checksum(frame_list):** Decodes a list of frames using the Checksum algorithm.

3. File: *operations.py*
    a. **noisy_channel(frame):** Emulates the passing of a frame through a noisy channel where error can occur randomly.
    b. **xor(a, b):** Finds out the XOR of two binary numbers a and b.

c. **binary_division(dividend, divisor):** Implements binary division.
d. **warped_sum(sum, frame_size):** Finds out the warped sum in checksum algorithm
e. **readfile(filename, frame_size):** Extracts frames from a file.
f. **writefile(filename, frame_list):** Writes a list of frames into a file.
g. **execute():** Executes the program

4. File: *main.py*
    a. **vrc():** Implements VRC algorithm.
    b. **lrc():** Implements LRC algorithm.
    c. **crc():** Implements CRC algorithm.
    d. **checksum():** Implements checksum algorithm.

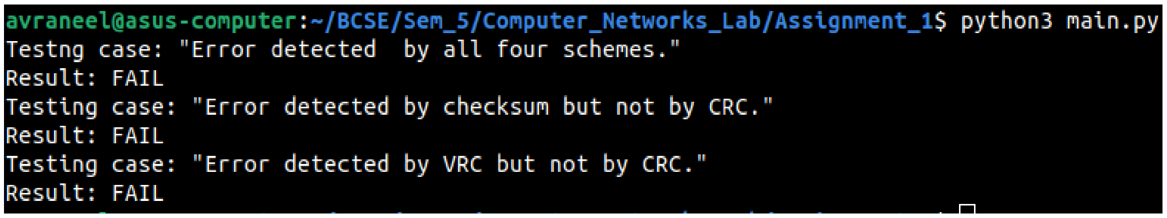
Test cases:

Contents of the sender file: 110101101011

1. *No. of bits changed = 0.*

   This means that the message will pass through a noiseless channel.

   Output:



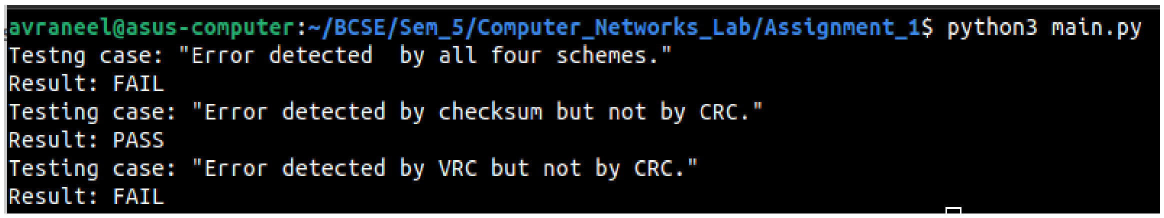
```
avraneel@asus-computer:~/BCSE/Sem_5/Computer_Networks_Lab/Assignment_1$ python3 main.py
Testng case: "Error detected  by all four schemes."
Result: FAIL
Testing case: "Error detected by checksum but not by CRC."
Result: FAIL
Testing case: "Error detected by VRC but not by CRC."
Result: FAIL
```

   Thus, no error is detected.

2. *No. of bits changed = random*

   Error will occur at random positions.

   Output for one of these cases:



```
avraneel@asus-computer:~/BCSE/Sem_5/Computer_Networks_Lab/Assignment_1$ python3 main.py
Testng case: "Error detected  by all four schemes."
Result: FAIL
Testing case: "Error detected by checksum but not by CRC."
Result: PASS
Testing case: "Error detected by VRC but not by CRC."
Result: FAIL
```

Results:

For evaluating performance, we call the execute() function 1000 times at once and count how many times each case gives a ‘PASS’ result.

The sum of all the counts is less than 1000 because sometimes errors are not detected at all.

Sl. No.	**Case 1 ‘PASS’ count**	**Case 2 ‘PASS’ count**	**Case 3 ‘PASS’ count**
1	401	71	478
2	379	76	513
3	384	75	518
4	382	67	498

Analysis:

Possible Improvements:

- ➢ Socket programming could be used to further emulate the actual process of real-time communication.
- ➢ The error generating algorithm could be improved further.
- ➢ A better CRC divisor could be chosen.

Comments:

This assignment helped to learn how to implement various error detecting algorithms and also helped me in understanding how to implement various mathematical functions like binary division and warped sum.