Network Lab Report

Shuvayan Ghosh Dastidar
001810501044
JU-BCSE - III

Assignment 1: Design and implement an Error Detection module

Theory

Error detection is the detection of errors caused by noise or any other impairments during transmission from the transmitter to the receiver. All error detection schemes add some kind of redundancy (some extra data) so that the receiver can use to check consistency of the the delivered message and to recover the data.

For our purposed we focus on four kinds of error detection schemes:

- VRC: VRC is also called single parity check. In this type of scheme, a single parity bit is appended to the dataword to form the codeword. The parity bit is calculated on basis of the even parity of “1”s in the dataword. The codeword so formed is sent to the receiver which checks for errors. Single bit errors can be detected through VRC.

- LRC: Longitudinal redundancy check is a form of redundancy check that is applied independently to each of the parallel group of bit streams. The data is divided into transmission blocks, to which additional check data is added.

- Checksum: A checksum of a message is a modular arithmetic sum of message code words of a fixed word length (e.g., byte values). The sum may be negated by means of a ones'-complement operation prior to transmission to detect unintentional all-zero messages.

- Cyclic Redundancy Check (CRC): A cyclic redundancy check (CRC) is a non-secure hash function designed to detect accidental changes to digital data in computer networks. It is not suitable for detecting maliciously introduced errors. It is characterized by specification of a generator polynomial, which is used as the divisor in a polynomial long division over a finite field, taking the input data as the dividend. The remainder becomes the result.

Implementation

The language used for making this error detection module was Java because of it's vast support for IO-based applications and overall programming support.

File structure for the code :

```
src
    ├── ErrorDetection
    |   ├── CRC.java
    |   ├── Checksum.java
    |   ├── LRC.java
    |   ├── VRC.java
    |   └── detect.java
    ├── Receiver
    |   └── Receiver.java
    ├── Sender
    |   ├── Sender.java
    |   └── Utils.java
    ├── Test
    |   └── Gen.java
    ├── dataword.txt
    ├── gen.txt
    └── mapped.txt
```

Inside the src package, there are four packages - Error Detection, Receiver, Sender and Test.

Under the Error Detection package are the four files containing classes for each of the four error detection schemes. Each of the classes have certain functions such as prepare_codeword, extract_dataword, and detect_error.

1. `prepare_codeword(String)` : This function takes in the dataword(prefixed with it's length in binary) and calculates the redundant bits based on it and returns the formed codeword.

2. `detect_error(String)` : This takes in the codeword from the receiver side and applies the error detection algorithm with which it was encoded and if an error is detected, the codeword is rejected.
3. `extract_dataword(String)` : This takes in the codeword from the receiver and returns the dataword from it according to the length of the dataword in the codeword.

The sender package contains the sender class and an Utils class which contains helper functions for framing the dataword.

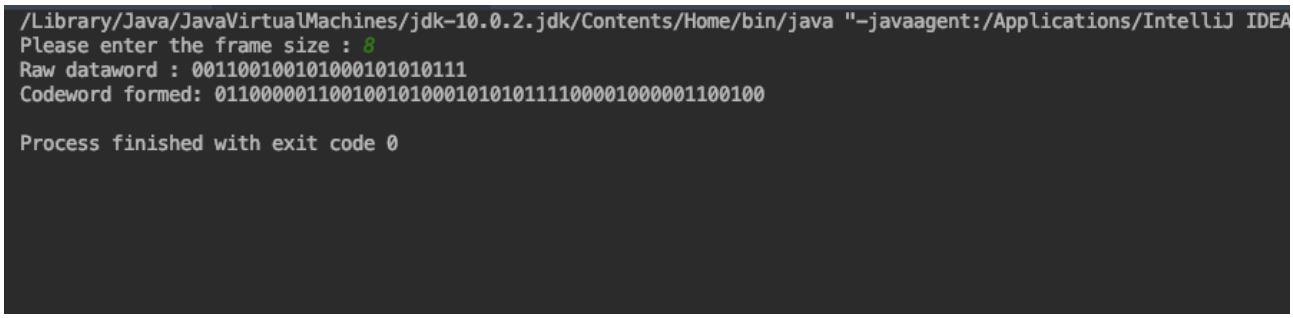
The process of framing the dataword:

The raw dataword is read from a file, whose path is given as an argument to the sender process. The length of the dataword is prefixed after converting it into a binary string. Then it is sent to the prepare_codeword function of the error_detection module to prepare the codeword along with redundant bits. The formed codeword is padded with “0”s if necessary according to the frame size which is taken as input in the sender process.

Inter process communication takes place between the sender and receiver process through a memory mapped file channel.

The receiver process is started and it waits for data …

Then the sender process is started which asks for the frame size from the input:



the above shows the codeword encoded with CRC-8 scheme

Then the receiver gets the data from the file channel and is checked for errors. Without errors, the dataword is extracted from the codeword and is shown in the console.

CASES

1. Now an error injecting function is made which injects errors in the formed codeword at random positions. For a 24 bit dataword and 3 random bits in the codeword flipped to simulate an error in the channel. All the four error detection schemes were able to detect the error.

2. To the codeword formed if we add the binary bits of the divisor of the CRC, it will not be able to detect the impairment whereas Checksum will successfully detect an error.

3. When an CRC polynomial is chosen with odd number of “1”s such as CRC-7, VRC will be able to detect an error when the divisor bits are added whereas the CRC algorithm will fail.