An Introduction to Block Error Correcting Codes and their Real World Applications

Thursday April 20th, 2023

Professor Rajesh Pereira

Michael Dombrovsky

Contents

Preface

This paper is meant to introduce the reader to the core mechanics of error correcting codes, in order to educate them about how a lot of the world around them functions. Effort was undertaken to make this simple to understand without advanced mathematics knowledge.

Introduction

Digital storage and transmission of data are not inherently stable, as they can be corrupted by external factors such as radiation rays or competing signals during transmission [4]. The field of correcting errors in digital communication is based on error correcting codes. Error correcting codes are a set of techniques that are used to detect and correct errors that happen in the transmission and storage of digital data.

There are two main types of error correcting codes; block codes that work on fixed size chunks of data and convolution codes that works on continuous streams of data. We will be focusing on how block codes work.

Data Corruption

Data corruption occurs when data is unknowingly changed. The simplest example of data corruption to visualize is when a single bit of data is changed, either a 0 to a 1 or a 1 to a 0. An infamous example of this happening is during 2003 Belgian elections, when a candidate received 4096 extra votes [4].

Review of binary representations

Lets review how digital computers store numbers as powers of 2:

00000001 corresponds to 1 as:

$$0 * 2^7 + 0 * 2^6 + 0 * 2^5 + 0 * 2^4 + 0 * 2^3 + 0 * 2^2 + 0 * 2^1 + 1 * 2^0 = 1$$

00001001 corresponds to 9 as:

$$0 * 2^7 + 0 * 2^6 + 0 * 2^5 + 0 * 2^4 + 1 * 2^3 + 0 * 2^2 + 0 * 2^1 + 1 * 2^0 = 9$$

Notice how the difference between these is 8, which is a power of 2 as $2^3 = 8$, in this example a single bit was flipped to go from the number 1 to the number 8.

As 4096 is equal to 2 to the power of 12 it is fair to conclude that this voting error was indeed a single bit error, most likely caused by a cosmic ray [4].

Algorithms

Triple-Repetition

Triple-repetition is an example of a simple error correction technique meant to familiarize the reader with error correction. Triple-repetition coding works by storing the data multiple times, and then seeing what the majority data is [3]. Lets take an example, 001001001 where the same data, 001 is stored three times, and then if an error occurs in one of those three chunks such as 011001001, you can easily correct it because the other two chunks all have 001 as the message.

Reed-Solomon

Reed-Solomon codes are some of the most widely used error correcting codes, however they are a bit complicated to understand, so only a general overview of how they work will be provided.

They are specified as $RS(n, k)$, where [2]:

- n is the total length of a block of an encoded block.
- k is the length of the message data to encode.
- $n - k$ is the length of the parity data.

Overview of Galois Fields

Galois fields are a branch of finite mathematics that are based on modular arithmetic, they will not be covered here as they are non-trivial, however it is important to keep the following points in mind:

- Every Galois field has a generator polynomial, $g(x)$, that contains a set of roots.
- In Galois fields a binary representation of some data has an equivalent polynomial. This means that you can freely convert from data to polynomials and from polynomials to data. The exact way how this occurs is not trivial, but it is enough to note that this process is defined.

Reed-Solomon codes are based on the mathematical concept of Galois fields, of the form of $GF(2^m)$ where m is the amount of binary bits per symbol [2]. Particularly $GF(2^8)$ is useful for communication as computers store data as groups of bytes, with each byte being composed of 8 bits.

Lets see how we can encode a message using Reed-Solomon code for a given Galois field [1]:

- Lets define a set of n points $a_1, \ldots, a_n$ that are the roots of $g(x)$, the generator polynomial.
- We convert our message into a polynomial, $m(x)$, this will have degree $k-1$.
- We calculate the encoded polynomial, $c(x)$, by evaluating $m(x)$ at $a_1, \ldots, a_n$. Note that $c(x)$ will have degree $n-1$.
- We convert the encoded polynomial to message data of size n.

Reed Solomon codes can tolerate errors up to half the amount of parity bits used, meaning that if $n-k=2t$, up to t errors can be corrected [2]. For example, if the parity size $n-k=10$, up to 5 errors can be corrected.

The decoding algorithm is a bit more involved than the encoding part [1]:

- We convert the received message into a polynomial, $r(x)$, this will have degree $n-1$.
- Lets define an error polynomial $e(x)$, with roots at error locations, this will have degree up to t. Note that: $r(x) = c(x) + e(x)$.
- Note that for every root a of $g(x)$, this will hold: $m(a) \cdot e(a) = r(a) \cdot e(a)$, which allows us to solve for $e(x)$.
  - This gives us a system of n equations as $g(x)$ has n roots.
  - This system of equations will have $k+2t$ unknowns.
    - left side, $m(a) \cdot e(a)$ will provide $k+t$ unknowns.
    - right side, $e(a)$ will provide t unknowns.
  - We know that $n-k \geq 2t \Rightarrow n \geq k+2t$, so that we can solve for $e(x)$.
- We can then solve for the original encoded message, $c(x)$ as $c(x) = r(x) - e(x)$.
- We then evaluate $c(x)$ at $a_1, \ldots, a_k$ to get the original $m(x)$.
- We then convert $m(x)$ into the original message of size k.

It is an interesting side note that the decoding algorithm above shows us why we can correct up to half the amount of errors as compared to the amount of parity data, $n - k \geq 2t$. This relies on the fact that in order to solve a system of equations there must be at least as many equations as there are unknowns.

Overall Reed Solomon codes are fairly complex to implement, however they make up for this complexity with their error correction power.

Applications

Computer memory and signals can degrade over time, leading to inaccuracies in information they hold. Error correcting codes often can mitigate these issues. Below are just two examples of applications of error correcting codes in the real world.

Space Shuttle Redundancy Management

It was paramount that space shuttle computers performed properly without errors in order to execute multiple highly important functions such as trajectory calculation. This requirement was compounded by the fact that higher amount of radiation that the computers would receive while in space would make bit flips more common [4]. This problem was overcome by NASA using a technique known as redundancy management, where multiple computers were fed the same input and performed calculations simultaneously, they would then vote on the correct course of action for any stage of flight [5]. This process is an example of a triple-repetition code used in real life.

QR Codes

QR (Quick Response) codes are two dimensional barcodes that are meant to store large amounts of data in a small space. They are meant as an improvement over conventional barcodes that only hold 20 alphanumeric characters, whereas QR codes can hold up to 4296 alphanumeric characters (when using low error recovery) [6]. QR codes employ Reed-Solomon error correction that allows them to work properly even if up to 30% of the QR code is blocked by exogenous factors [6]. It was this error correction capability that led to their development and continued use. This allows QR codes to still be readable from some blurry photographs, this ability is useful today but used to be more important in prior times as mobile computing devices cameras did not have the capability to capture high definition images in the past. The current QR code specifications (version 40) allow you to select between an error correction level of 7% and 30%, allowing you to store between 1852 and 4296 alphanumeric characters respectively [6].

Conclusion

Error correcting codes are used all around us, from something as complex as space travel all the way down to QR codes. The world relies on error correcting codes to make sure that computing devices work as desired.

References

[1] Ashish Choudhury. *Lec 19 reed-solomon error-correcting codes*. 2022. URL: https://www.youtube.com/watch?v=6X1OCX-iq9w.

[2] Yongmei Liu et al. "Reed-Solomon Codes for Satellite Communications". In: *2009 IITA International Conference on Control, Automation and Systems Engineering (case 2009)*. 2009, pp. 246–249. DOI: 10.1109/CASE.2009.30.

[3] Joy Morris. *Combinatorics: An Upper-level Introductory Course in Enumeration, Graph Theory, and Design Theory*. Joy Morris, 2017, p. 190.

[4] Antonio Nappa, Christopher Hobbs, and Andrea Lanzi. *Deja-Vu: A Glimpse on Radioactive Soft-Error Consequences on Classical and Quantum Computations*. 2021. arXiv: 2105.05103 [cs.CR].

[5] J. R. Sklaroff. "Redundancy Management Technique for Space Shuttle Computers". In: *IBM Journal of Research and Development* 20.1 (1976), pp. 20–28. DOI: 10.1147/rd.201.0020.

[6] Sumit Tiwari. "An Introduction to QR Code Technology". In: *2016 International Conference on Information Technology (ICIT)*. 2016, pp. 39–44. DOI: 10.1109/ICIT.2016.021.