Comparative Study of Efficient Raptor Codes Implementation using FFmpeg

Dhyey Thummar, Neeraj Jadhav, Pratyush Sahu

October 2024

Abstract

Fountain Codes are a class of erasure codes theorized for communication over Binary Erasure Channels (BEC). By using a rateless (potentially infinite) sequence of symbols to encode messages that can be, in reasonable time, decoded by a receiver, they provide guarentee of error correction. LT and Raptor codes are some of the most well-known practical classes of fountain codes that provide guarentees on the number of encoded symbols required to be generated for some message, and the time complexity of encoding and decoding. Raptor codes provide an advantage over LT codes by providing a linear time complexity for the process of encoding and decoding. There are multiple open-source projects that provide implementation for raptor codes over various languages. However, to the best of our knowledge, there doesn't exist an extensive study on the efficiency of these implementations on widely used video streaming applications where such erasure codes can vastly help.

In our project, we propose an extensive study of three implementations on application. The application of choice is FFmpeg, an open source library and tool which provides support for processing audio and video files, and can be operated from the command-line. The implementations we look to focus on are (1) a Raptor code implementation using Rust, (2) gofountain, an open source library from Google that provides implementation for various erasure codes, and (3) OpenRQ, a RaptorQ implementation in Java. We aim to develop a pipeline that

The eventual goal of our project is to *compare the efficiency* of different Raptor code implementations and provide a *benchmark* for their performance in video streaming applications. Additionally, we aim to explore potential avenues for *improving the performance* of these implementations under various network conditions.

To achieve this, we will:

1. Developing a video streaming pipeline using FFmpeg with simulated packet loss.
2. Implement error correction using different code implementations (Rust, gofountain, OpenRQ).
3. Measure performance metrics like recovery rate, latency, and overhead (with packet loss).
4. Analyze and benchmark each implementation and identify potential optimizations.