Evaluation of the Effectiveness of Packet Erasure Codes in a Docker-Simulated Unreliable Network

Minghan Chan
Murong Ng
Pei Xin Yu

University of California, Berkeley

Apr 2024

Abstract

This report explores the efficacy of packet erasure coding in unreliable network environments. Packet erasure coding divides data into packets, encodes them with redundancy to preserve data integrity amidst packet loss, and is particularly vital when networks are unreliable. The study aims to find the optimal redundancy rate for successful packet decoding at end nodes based on the packet drop rates at the routers. Leveraging Docker for network simulation, controlled experiments vary redundancy factors and packet drop rates across diverse network topologies. The simulation results give insight into how we can adjust the redundancy factor in order to optimise decoding success rates based network reliability.

Keywords: Packet Erasure Codes, Network Simulation
https://github.com/minghancmh/network-emulation

1. Background

Erasure coding is a method of data protection by which data is broken into packets. These packets are then expanded and encoded with redundant data pieces so that the original data is still preserved in case of packet erasure or outages. In the context of communication networks, packet erasure coding [1] is used in the case of multicasting, where it is infeasible for the source node to track acknowledgements from all the destination nodes. As such, this scheme is designed to be able to recover from the "erasures", or dropping of packets as packets propagate through the network. This paper hopes to discuss and evaluate the effectiveness of packet erasure coding in the case of unicasting within an unreliable network.

2. Introduction

This project investigates the resilience and efficiency of network communication protocols under varying conditions of redundancy and packet loss. The primary goal is to examine the impact of adjusting the $redundancyFactor$ alongside the $packetDropRate$ on the probability of successful payload decoding at the receiver end. Such an investigation is crucial for understanding and improving the robustness of network systems in environments where data loss and transmission errors are prevalent.

To facilitate a controlled and replicable experiment setup, Docker is employed as the simulation environment. Docker provides a stable and isolated platform where different network configurations and conditions can be emulated with precision. By leveraging Docker containers, we can simulate a network of routers and endpoints, with the ability to finely tune parameters such as the $redundancyFactor$ and $packetDropRates$. This setup allows for a systematic analysis of how these variables influence the effectiveness of data transmission, specifically focusing on the decoding success rate at the network's receiver nodes.

Throughout this project, various scenarios will be constructed to reflect a range of realistic operating conditions. The outcomes will contribute valuable insights into optimal network configurations and strategies, potentially guiding future developments in network design and error correction methodologies. By the conclusion of this experiment, we aim to establish a clearer understanding of the interdependencies between redundancy, packet loss, and payload integrity, ultimately enhancing the reliability of network communications in adverse conditions.

3. Methodology

The idea of this project was to vary the values of $redundancyFactor$ and $packetDropRate$, and finally determine the probability that a receiver can accurately decode a message.

With these definitions, we have:

$$m = \rho_{rf} \cdot k \tag{1}$$

where ρ_{rf} is the $redundancyFactor$, k is the number of original packets after packetizing a stream, and m is the total number of packets to be sent through the network. This provides us $m - n$ redundant packets in the network. Based on the $zfec$ encoder, a receiver would be able to decode the message as long as it received any n of the m messasges in the network. The value of ρ_{rf} was then varied, $0 \leq \rho_{rf} \leq 1$. It should be noted that when $\rho_{rf} = 0$, this degenerates to sending the raw packets over the network, with no redundancy.
Next, we also varied the value of $packetDropRate$, where all routers in the network had a specific constant ω. This was key to emulating the idea of an unrealiable network.

$$P(routerForward) = 1 - \omega \tag{2}$$

3.1. Topology Generation

To setup the network in the docker environment, we decided to employ the use of an Erdos-Renyi graph. In the context of emulating a router network for a networks project, employing the Erdős-Rényi (E-R) model is particularly advantageous due to its fundamental characteristics, which mirror aspects of the real-world structure of the Internet[2]. The Erdős-Rényi graph is a type of random graph where edges between pairs of nodes are established with a constant probability, independent of other edges. This stochastic nature of the E-R model captures the inherent randomness and organic development observed in the topology of the World Wide Web. Unlike more deterministic models or regular graphs where the connections are fixed and predictable, the E-R model allows for the exploration of network behaviors under various random configurations, reflecting the unpredictable nature of real-world networks. This aspect is crucial for studying properties such as network robustness, connectivity, and path diversity, which are integral to understanding and designing efficient and resilient router networks. The flexibility and simplicity of the E-R graph make it a suitable choice for modeling networks that need to represent a wide array of potential connection scenarios, akin to the complex and diverse linkage seen in the global Internet infrastructure.

3.2. Simulation Parameters

To simulate the experiment, we created a network of 20 nodes using python's networkx, with $p = 0.2$, where p represents the probability of edge creation. We also attached a host to each of these routers, where a random selected host was to be our final $receiver$ node (acting as the unicast receiver), which decodes the transmitted packets. In each experiment, a total of 200 different messages were sent, each of 200 words long.

3.3. Encoding and Decoding

The encoding and decoding scheme is implemented by the zfec library [3]. The encoding is parameterized by two integers, k and m. m is the total number of blocks produced, and k is how many of those blocks are necessary to reconstruct the original data. Data is divided into "primary blocks" sized at $\frac{1}{k}$ of the original data. "Secondary blocks" are generated on primary blocks for redundancy. Metadata, including k, m, share numbers, and padding, is generated. The metadata and blocks are then compressed and appended to each share. The final encoded data comprises m blocks, each containing compressed metadata and data blocks. The decoding process is the reverse of encoding. Minimum k blocks are required for decoding. Compressed metadata is extracted and decompressed to retrieve necessary information. The original data is then reconstructed using metadata and selected blocks.

4. Results & discussion

Due to each run of the experiment needing to spin up 30 docker containers and run Link State Protocol to convergence to generate routing tables, each simulation took a lot of compute and ran for approximately 3 minutes. As such, the experimental surface plot was smoothened using a linear interpolating technique to generate more plot points. Also, each experiment was only run once per setting of redundancy factor and packet drop rate, and further, due to randomness in graph generation, there are some outliers in the graph.

Nonetheless, we see that our surface plot roughly resembles the theoretical surface plot. We find that further work can be done to analyse the maximum of these surface plots, and a unconstrained optimization technique could be used to find the most suitable redundancy factor based on the unreliability of the particular network that a payload is about to be transferred over.

4.1. Theoretical Analysis

Assuming that every router in the network has the same packet drop rate, we arrive at the following equations.

$$N = (1 + \rho_{rf}) * n \tag{3}$$

where N is the effective number of packets. Further, defining s as the survival per hop, we have

$$s = 1 - \omega \tag{4}$$

In order for a packet to survive all hops, we have

$$s_a = s^h \tag{5}$$

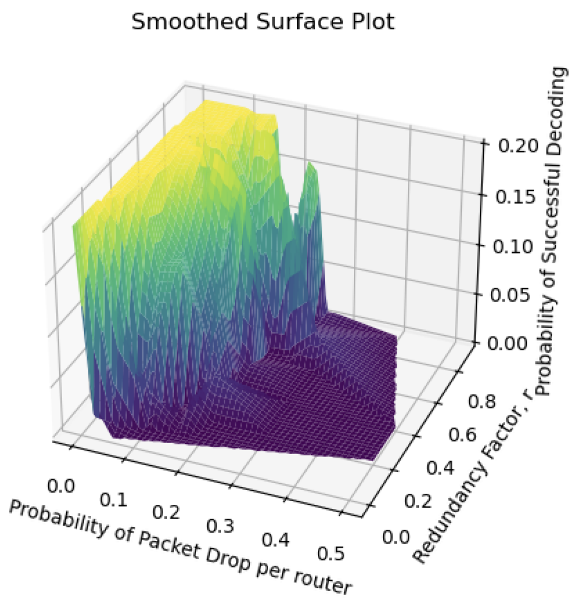

Figure 1: Experimental Surface Plot

where h is the average number of hops taken by the packet thorugh the network. As such, we arrive at a final probability p

$$p = \sum_{k=n}^{N} \binom{N}{k} s_a^k (1 - s_a)^{N-k} \tag{6}$$

Using this equation, we are able to plot the theoretical surface plot as shown.

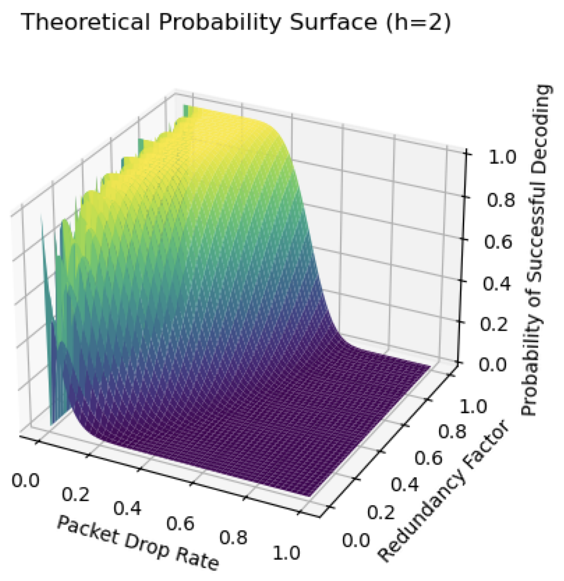

Figure 2: Theoretical Surface Plot with h-2

As we can see from the theoretical plot, as packet drop rate increases, the probability of a successful decoding decreases exponentially. This ”unreliability” in the network can be offset by sending redundant packets as seen by the yellow surface in the plot. However, this only works to a certain extent, and when packet drop rates beyond 0.5 still result in a very low probability of successful decoding. It should also be noted that with larger payloads, the probability of a successful decoding could be much lower.

5. Conclusions

In conclusion, our study aligns with theoretical expectations regarding the relationship between packet drop rates, redundancy factors, and decoding success rates in unreliable network environments. The findings underscore the importance of minimizing redundancy rates to reduce packet encoding overhead. However, our study encountered limitations due to computational constraints, resulting in simulations limited to a network of 20 routers with a restricted number of hops between each router, which may not fully reflect real-world scenarios. To address these limitations and further refine our understanding, we recommend expanding resources to simulate larger and more complex network topologies. We could also explore other types of packet erasure codes such as Tornado Encoding which might prove to be more robust. By addressing these recommendations, we can contribute to the development of more robust and efficient communication protocols for unreliable network environments.

References

[1] S. J. Walrand. Communication networks: A concise introduction, second edition, synthesis lectures on communication networks. 2, 2017.

[2] J. Li. A brief overview of graph theory: Erdos-renyi random graph model and small world phenomenon. 2021.

[3] L. Rizzo. zfec – efficient, portable erasure coding tool. 2013.