Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

Comparing Various Locality Approaches for Codes Repairing Two Erasures

(Dutch title: Vergelijking van Verschillende Localiteitsvormen voor Codes die Twee Weggevallen Symbolen Repareren)

Thesis submitted to the
Delft Institute of Applied Mathematics
in partial fulfillment of the requirements

for the degree

BACHELOR OF SCIENCE
in
APPLIED MATHEMATICS

by

W.J.P. Speé

Delft, The Netherlands
June 2018

TU Delft

BSc thesis Applied Mathematics

"Comparing Various Locality Approaches for Codes Repairing Two Erasures"

(Dutch title: "Vergelijking van Verschillende Localiteitsvormen voor Codes die Twee Weggevallen Symbolen Repareren")

W.J.P. Speé

Delft University of Technology

Supervisor

Dr.ir. J.H. Weber

Thesis committee

Dr. D.C. Gijswijt

Dr. J.A.M. De Groot

June, 2018

Delft

Preface

This thesis is submitted as the final part of the Bachelor program in Applied Mathematics at Delft University of Technology. The choice for coding theory as a subject of this thesis is motivated by my special interest in this field of mathematics. My first introduction to the subject was at the pre-university course "Junior TU-Delft" in 2015 during which coding theory was studied at an introductory level. In the first year of the Bachelor program I wrote a "Star Module" in which I combined this subject with elements of the linear algebra theory. In addition to these preparations, in 2017 I followed the course "Applied Algebra; Codes and Cryptography Systems" which provided me with a mathematically more profound basis for writing this thesis.

The subject of this thesis, efficient erasure repair using coding, is a rapidly developing field of study. Up to this moment, it is not yet thoroughly understood. During the process of writing this thesis several publications have been issued presenting new contributions, such as [1] and [2]. It is specifically worthwhile to mention the latter thesis because it contains significant contributions on the subject of Section 4.4. Unfortunately, this publication arrived too late in order to be incorporated in this thesis.

Furthermore, I would like to use this opportunity to express my gratitude to my supervisor, J. Weber, for his guidance in the process of writing this thesis and for his positive feedback during the weekly meetings. Finally, I would like to thank the thesis committee for reviewing this thesis.

Ward Speé
Delft, June 2018

Abstract

For many IT applications information is stored digitally across multiple storage units and needs to be available continuously. Due to malfunctions data servers might be erased or temporarily inaccessible. Different approaches using linear coding theory have been proposed in order to retrieve the content of erased servers from repair sets containing a selection of the remaining servers. This thesis focuses on comparing various repair methods for two erasures; *disjoint parallel repair* which protects each erasure by multiple disjoint repair sets, *cooperative repair* which uses a single repair set to repair all erasures at once and *sequential repair* which repairs erasures individually by using already repaired erasures. Coding inevitably induces storage overhead measured by the information rate. The applied method creates transmission overhead measured by locality, which concerns the number of servers accessed to perform erasure repair.

The comparison mainly consists of appropriately computing minimal transmission overhead for these methods given a predetermined storage overhead. It is shown that Hamming codes have the best possible transmission overhead applying the cooperative method. The next best method is sequential repair, whereas disjoint parallel repair is too restrictive for two erasure repair with Hamming codes. For a generalized parity code it is demonstrated that the sequential method can repair more erasures than the disjoint parallel approach, and reach a better locality than the cooperative method. In general, the cooperative method efficiently uses access to servers in order to reduce transmission overhead. For the same purpose, the sequential method uses already repaired servers allowing smaller repair sets to be accessed. In any repair process it is key to find optimal combinations of a method and code which exploit these qualities. The cooperative method applied to Hamming codes and the sequential method combined with generalized parity codes prove to be high-ranking combinations in this regard.

Contents

1 Introduction

In a distributed storage setting data is stored across spatially distributed storage units called nodes. The data stored in these nodes needs to be constantly accessible and has to be protected against all kinds of failures and errors. These threats may originate from various causes and can range from a bit changing value because of a reading malfunction to the loss of a complete node due to a power cut. Companies dealing with the storage of large amounts of data need to find ways to cope with the temporary loss of information. For example, in [2] and [3] it was observed that the social media network *Facebook* has a daily median of 50 nodes which are unavailable for more than 15 minutes. Moreover this unavailability generates a daily median of 180TB data transmission for content retrieving purposes. Therefore it is in Facebook's best interest to come up with efficient ways to retrieve the content of these nodes during the inaccessibility periods. The media provider *Netflix* goes far in attempting to guarantee the resilience of its IT-infrastructure [4]. The company designed a tool which simulates various node failures and deliberately caused breakdowns in the systems to test the resilience. The tool operates from the assumption that malfunctions are certain instead of possible and proper resilience is an obligation rather than an option. These examples illustrate the necessity of node repair and explain the interest in research and development in this field of study.

Coding theory is a branch of mathematics studying the properties of codes which have proven to be a powerful instrument for the retrieval of the content of failed nodes. Originally codes have been designed for transmitting information flawlessly through a noisy medium, but they are currently used in a broad spectrum of applications in electrical engineering, computer science and information theory. For an introductory overview of the principles of coding theory we refer the reader to Section 2. The focus of this thesis will be on the usage of codes for restoring the content of failed nodes in a distributed storage setting. More specifically, we will compare the qualities of several methods using codes that have been proposed for this purpose.

Suppose that information has been stored digitally in multiple data servers called *information servers*. It is inevitable that occasionally malfunctions occur and some or even all servers fail. The failure of a server is called an *erasure* and the servers used for repair of this erasure form a so-called *repair set*. In this thesis and in the context of coding theory repairing an erasure means the retrieval of the content of the erased server using the repair set instead of the actual reparation. The protection of the information servers by coding relies fundamentally on setting up additional servers as back-up. The addition of *back-up servers* naturally induces *storage overhead* which is ideally minimized and measured by the *information rate*; the ratio of information servers to the total number of servers. The large quantities of data that are usually involved in erasure repair form an additional aspect of the recovery process. The repair has to be performed efficiently in a sense that the transmission of data and access of other servers is minimized. The transmission of repair data is called *transmission overhead* and measured in *locality*, which -loosely speaking-

indicates the size of the used repair set.

Possibly the most straightforward way of arranging back-up servers is by duplicating the information servers. In case one of the information servers fails, it can be restored by accessing its duplicate. However, the repair of multiple erasures depends on the position of the erased servers. The failure of all information servers can be solved, whereas the erasure of both an information server and its duplicate is fatal. Another simple approach of installing back-up servers depends on the content of the server. Suppose each information server consists of a single bit, then we design one back-up server containing a single bit which makes the sum of all bits even. In case one of the servers is erased its content can be determined by making sure the sum of all bits is even again. These two basic recovery approaches for one erasure are illustrated in Figure 1 and respectively known as the repetition code and parity code. Notice that the repetition code has optimal transmission overhead since it has to access only one server for the repair. The parity code on the other hand has optimal storage overhead because only one back-up server is added. For both codes this comes at the price of a worse overhead of the other type. Namely, the parity code needs to access all other servers to perform reparation and the repetition code has doubled the number of servers.

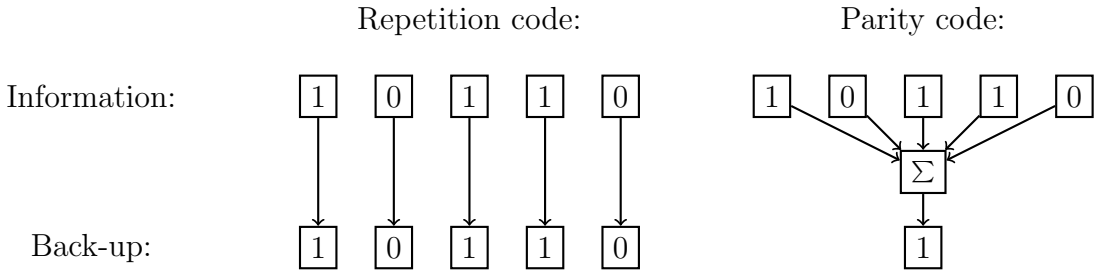

Figure 1: Installation of back-up servers using the repetition and parity codes given the content of five information servers.

Storage and transmission overhead form two important factors that should be taken into account for optimization of erasure repair design. The previously discussed codes perfectly illustrate the trade-off between these two factors. It will not come as a surprise that there exist many other ways of setting up back-up servers, some of which reach a better balance. As stated storage overhead is induced by the introduction of back-up servers and by the used code. Transmission overhead however is determined by both the code and a method of reparation. Namely, together with the used code a method should be specified which describes the way in which the code is used to perform reparation. Usually three main methods are distinguished; *parallel repair* which protects each erasure by parallel repair sets, *cooperative repair* which uses a single repair set to repair all erasures at once and *sequential repair* which repairs erasures individually by using already repaired erasures.

1.1 Problem Statement

In this thesis we initially study the previously mentioned repair methods separately. For each method we are interested in the maximum number of erasures that in any case can be repaired given a certain code. Furthermore, we discuss appropriate locality measures of transmission overhead for each repair method and determine the optimal trade-off between storage and transmission overhead. This means that for a predetermined storage overhead we determine the theoretically lowest achievable transmission overhead and then search for a combination of method and code which actually achieves this balance. We aim at appropriately comparing the performance of the methods using these results and intent to answer the following research question:

> " How do the parallel, cooperative and sequential repair methods compare in terms of storage and transmission overhead in the search for an optimal method in case of two erasures? "

1.2 Organisation

The sequel of this thesis is organized as follows. Chapter 2 contains an overview of the necessary concepts of coding theory. In Chapter 3 we discuss the repair of a single erasure. Next, multiple erasures are considered in Chapter 4 in which three different locality approaches are treated separately. These approaches are disjoint parallel repair, cooperative repair and sequential repair respectively in Sections 4.2, 4.3 and 4.4. Then, in Chapter 5 these approaches are compared based on their ability to reduce storage and transmission overhead. Finally, the thesis is concluded in Chapter 6.

1.3 Notation

Throughout this thesis we use the following notation and conventions. For an integer $n \in \mathbb{N}$ we write $[n] = \{1, 2, ..., n\}$. Let $\mathcal{E}$ be a subset of $[n]$ then its complement is denoted by $\overline{\mathcal{E}} = [n] \setminus \mathcal{E}$ and its size by $|\mathcal{E}|$. The support of a vector $\mathbf{x} = (x_1, x_2, ..., x_n) \in \mathbb{F}_2^n$ is the set of indices where it is not zero, i.e. $supp(\mathbf{x}) = \{i \in [n] : x_i \neq 0\}$. The support can thus be treated as a set of integers. By writing $\tilde{\mathbf{x}}$ in stead of $\mathbf{x}$ we distinguish respectively vectors which are (partially) unknown due to erasures and vectors which are fully known or already repaired. Furthermore, $\mathbf{x}$ will denote an element of a code and $\mathbf{y}$ an element of the dual code.

2 Prerequisites

In this chapter we introduce some concepts of coding theory and discuss the Parity code as an explanatory example. This chapter only contains results necessary for this thesis. For a rigorous treatment of coding theory and for proofs of the stated results we refer to [5] and [6].

2.1 Introduction to Coding Theory

The main principle of coding is adding symbols to data in order to detect and correct errors or to repair erasures in the data. Suppose that stored data consists of a sequence of *information symbols*. Extra symbols, so-called *parity symbols*, are added intelligently to the sequence and correspond to the back-up servers. The sequences containing both the information and parity symbols are called *codewords*. The set of all possible codewords is called a *code* and specifically a *block code* if all codewords have equal length. The process of adding parity symbols is called *encoding* and makes sure that not all possible symbol sequences are codewords. This distinction between codewords and non-codewords may give the code erasure repairing or error correcting properties.

The symbols are all elements of a predetermined *alphabet*. For instance, the alphabet might be our linguistic alphabet or the numbers zero until nine. By far the most frequently used alphabet in applications of coding theory is the binary alphabet, $\{0, 1\}$, and its elements are called *bits*. The binary alphabet is usually associated with the elements of the field $\mathbb{F}_2$ which enables computation with the symbols. In this setting the sequences are elements of the vector space $\mathbb{F}_2^n$ over $\mathbb{F}_2$. This means that a codeword can be seen as a n-dimensional row vector containing only zeros and ones. The focus in this thesis is solely on binary block codes and therefore we restrict our attention to these codes from this point forward. Let us discuss an easy example to briefly explain these concepts.

Example 2.1 (Parity code)**.** *In this example the Parity code introduced in Chapter 1 is analyzed further. During the encoding process one parity symbol is added in such a way that the sum of all symbols is even. This means that encoding is performed via the map;*

$$f : \mathbb{F}_2^{n-1} \longrightarrow \mathbb{F}_2^n, \ f((x_1, x_2, ..., x_{n-1})) = (x_1, x_2, ..., x_{n-1}, \sum_{i=1}^{n-1} x_i). \tag{1}$$

Note that the addition is performed in $\mathbb{F}_2$ and will result either 0 or 1. For $n = 4$ the corresponding parity code $\mathcal{C} \subset \mathbb{F}_2^4$ is given by: $\{0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111\}$. Since $\mathcal{C}$ solely contains words with an even number of 1's, it can be used for single erasure repair. In case one symbol is erased its value can be determined by all remaining symbols.

2.2 Linear Codes

A class of commonly adopted codes is linear codes. The linearity assumption enables the usage of powerful mathematical tools from linear algebra on codes.

Definition 2.2. *Let $n \geq k$ be integers. A subset $\mathcal{C} \subseteq \mathbb{F}_2^n$ is a linear $[n,k]_2$-code if it is a subspace of $\mathbb{F}_2^n$ of dimension k. The parameters n and k are respectively called the length and dimension of $\mathcal{C}$.*

In a linear $[n,k]_2$-code $\mathcal{C}$ every codeword contains k information symbols and $n-k$ parity symbols. These parity symbols induces storage overhead measured by the information rate; $\frac{k}{n}$. For a linear $[n,k]_2$-code $\mathcal{C}$ it suffices to specify k independent vectors - a basis - of $\mathcal{C}$ in order for all elements $\mathcal{C}$ to be fixed. A list of all codewords as in Example 2.1 is no longer needed, since $\mathcal{C}$ is formed by all linear combinations of the basis vectors. A common representation of the basis vectors is given by the rows of a $k \times n$ matrix, called a *generator matrix* of $\mathcal{C}$. This generator matrix induces an encoding map; $\mathbf{x} \in \mathbb{F}_2^k \to \mathbf{x}G \in \mathbb{F}_2^n$ and forms precisely all possible linear combinations of the basis vectors. In case the generator matrix is of the form $G = [I_k|A]$ for the $k \times k$ identity matrix I_k and arbitrary $k \times (n-k)$ matrix A then it is said to be in *standard form*. Every linear code has multiple generator matrices, but the one in standard form - in case it exists - is unique. A generator matrix in standard form corresponds an encoding map which adds parity symbols at the end of the sequence of information symbols. This is a desirable property of linear codes designed for erasure repair, since it implies that the content of the information servers does not change by the encoding map.

Creating a code with a basis and generator matrix can be seen as a construction by extension; each additional basis vector extends the code, since it allows more linear combinations to be formed.

Example 2.3 (Parity code, continuation)**.** *For the parity code linearity follows from the observation that any linear combination of codewords has an even number of 1's and is thus again a codeword. The parameters of the code relate as $n = k+1$ because the encoding process adds one parity symbol. The code discussed previously in this example has parameters $n = 4$ and $k = 3$ and can be fully determined by three independent codewords given by generating matrix G:*

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

Notice that G is in standard form and therefore $\boldsymbol{x} \to \boldsymbol{x}G$ yields the encoding map (1). Any other choice of independent codewords results in the same code, but a different map.

2.3 Dual Codes

The counterpart of constructing a code by extension, i.e. a generator matrix, is by restriction. Every linear $[n,k]_2$-code $\mathcal{C}$ corresponds to another code called the *dual code* of $\mathcal{C}$ and denoted by $\mathcal{C}^\perp$. It is defined as the null space of $\mathcal{C}$, in which the elements should be written as row vectors. We will use - without proof - some basic properties of the dual code, such as the fact that $\mathcal{C}^\perp$ is a linear $[n,n-k]_2$-code and that $\mathcal{C} = (\mathcal{C}^\perp)^\perp$. Intuitively,

the elements of the dual code form the restrictions for a vector to be an element of C. A generator matrix H of $C^\perp$ is called a parity check matrix for $\mathcal{C}$, because it is used to check whether a vector is a codeword. In fact, $\mathbf{x} \in \mathbb{F}_q^n$ is a codeword if and only if it has a dot product of zero with every element of $\mathcal{C}^\perp$, or equivalently $\mathbf{x} \in \mathcal{C}$ if and only if $\mathbf{x}H^T = \mathbf{0}$.

Example 2.4 (Parity code, continuation)**.** *Using standard linear algebra techniques to determine the null space of $\mathcal{C}$ we find that the dual code of the $[4,3]$-parity code is given by $\mathcal{C}^\perp = \{0000, 1111\}$ and a parity check matrix H of C:*

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}.$$

In general a parity check matrix of a $[k+1,k]$-parity code is given by the $1 \times k$ all-one matrix. Notice that we originally constructed $\mathcal{C}$ by the restriction that each codeword should satisfy that the sum of all symbols equals zero. This exactly corresponds to $\boldsymbol{x} \in \mathcal{C}$ if and only if $\boldsymbol{x}H^T = 0$.

2.4 Hamming Distance

The goal of adding parity symbols is to achieve a code with erasure repairing or error correcting properties. Hamming distance is a concept that is closely related to this and which plays a central role in coding theory.

Definition 2.5. *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_q^n$ then the Hamming distance $d(\boldsymbol{x}, \boldsymbol{y})$ between $\boldsymbol{x}$ and $\boldsymbol{y}$ is given by the number of code symbols in which $\boldsymbol{x}$ and $\boldsymbol{y}$ differ. Let $\mathcal{C}$ be a code then the minimum Hamming distance of C is defined as $d = \min\{d(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}\}$.*

Notice that the minimum Hamming distance is defined for any code. In particular, for a linear code we have $d = \min\{d(\mathbf{x}, 0) : \mathbf{x} \in \mathcal{C}\}$ and it is denoted as a $[n,k,d]_2$-code. For $\mathbf{x} \in \mathcal{C}, d(\mathbf{x}, 0)$ is usually known as the *Hamming weight* of $\mathbf{x}$ denoted by $wt(\mathbf{x})$.

A codeword from a code with Hamming distance d needs to be altered in at least d symbol positions before it can be another codeword. This means that up to $d-1$ erasures can be repaired using such a code since in that case there exists only a single codeword which fits all remaining symbols. Notice that the addition of parity symbols to a code and consequently not allowing every word to be a codeword is a necessary condition for a code to have a high minimum Hamming distance. This idea is encapsulated in an important inequality that connects the main parameters of a linear code.

Lemma 2.6 (Singleton bound)**.** *Let $\mathcal{C}$ be a linear $[n,k,d]_2$-code. Then the code parameters satisfy the following inequality;*

$$d \leq n - k + 1. \tag{2}$$

A code satisfying the Singleton bound with equality is called a *Maximum Distance Separable*, MDS code. The length n and dimension k of a code often follow directly from the definition of a code, whereas the minimum Hamming distance is sometimes hard to compute. MDS codes such as well known Reed-Solomon codes offer an advantage in this regard. For other linear codes the following theorem might help determining the Hamming distance.

Theorem 2.7. *Let $\mathcal{C}$ be a linear $[n,k,d]_2$-code and H any parity check matrix of $\mathcal{C}$. Then $\mathcal{C}$ has minimum Hamming distance d if and only if every (d-1) columns of H are independent, but there exist d dependent columns.*

Example 2.8 (Parity code, continuation)**.** *The parity check matrix H shows using Theorem 2.7 that the Parity code has minimum Hamming distance $d = 2$ and therefore it is a MDS code. The parameter value $d = 2$ results as well from the fact that $\mathcal{C}$ only contains codewords with an even numbers of 1's. It follows that the parity code can be used for reparation of a single erasure.*

2.5 (Shortened) Hamming Codes

It has already been stated that a linear $[n,k,d]_2$-code can resolve up to $d-1$ erasures. This thesis focuses on the repair of two (or more) erasures and therefore we finalize this section by a classification of linear codes with Hamming distance $d \geq 3$. Before we state this result in Lemma 2.10 we define the commonly used Hamming codes and introduce the concept of shortening.

Definition 2.9. *Let $m \geq 2$ be an integer, then $Ham(2,m)$ is a linear $[2^m-1, 2^m-m-1, 3]_2$-Hamming code with a parity check matrix containing all nonzero words of $\mathbb{F}_2^m$ exactly once as columns.*

It should be noted that the two parameters of $Ham(2,m)$ respectively denote that it is a binary code and that it has $m = n-k$ parity bits. Permutations of columns in this parity check matrix often result in different, but equivalent codes. Since the parity-check matrix of a binary Hamming code contains all nonzero words of $\mathbb{F}_2^m$ exactly once, it has the following property; any two columns are pairwise linearly independent, while there exist three dependent columns. This implies $d = 3$ according to Theorem 2.7.

There exist many ways of altering codes in order to improve its qualities. In Example 2.1 we discussed the addition of a parity bit in order to make the number of 1's in every codeword even. This alteration can be done to any code and is known as *extending*. Another alteration is called *shortening* and decreases the length and dimension of a code, while its Hamming distance does not decrease. Let $\mathcal{C}$ be a linear $[n,k,d]_2$-code with $k \geq 2$ then we select the set of codewords which contain a zero on a certain index. By deleting the bits on this index the resulting code is a linear $[n-1,k-1,d']_2$-code with $d' \geq d$. In terms of parity check matrices shortening corresponds to deleting a column. The process of shortening can be repeated.

Lemma 2.10. *Let $\mathcal{C}$ be a linear $[n,k,d]_2$-code with $d \geq 3$, then $\mathcal{C}$ is either a $Ham(2,n-k)$ code or a shortened $Ham(2,n-k)$.*

Proof. Let $\mathcal{C}$ be a linear $[n,k,d]$-code. By Theorem 2.7 Hamming distance $d \geq 3$ implies that the columns of any parity check matrix H of $\mathcal{C}$ are pairwise independent. It follows that H contains each word of $\mathbb{F}_2^{n-k}$ at most once as a column and does not contain the zero word. This implies that number of columns of H cannot exceed the number of

nonzero words in $\mathbb{F}_2^{n-k}$ or equivalently $n \leq 2^{n-k} - 1$. In case equality occurs H must contain all nonzero words of $\mathbb{F}_2^{n-k}$ exactly once and is thus the parity check matrix of a $Ham(2, n-k)$ code. Otherwise H lacks some nonzero words of $\mathbb{F}_2^{n-k}$ and it are exactly these missing words which have been deleted by shortening from a parity check matrix of a $Ham(2, n-k)$ code. □

Remark 2.11. *Hamming codes have Hamming distance $d = 3$ and therefore a linear $[n, k, d]_2$-code with $d \geq 4$ is always a shortened $Ham(2, n-k)$.*

3 Repairing a Single Erasure

In order to fully understand reparation methods for multiple erasures it is natural to address single erasure repair first [7]. Suppose a linear $[n, k, d]_2$ code $\mathcal{C}$ is used for reparation purposes and one of its codewords, say $\mathbf{x} = (x_1, x_2, ..., x_n) \in \mathcal{C}$, is stored symbol-wise over n servers. This means each codeword symbol is stored in a different server, which all have been indexed by an element of $[n]$ accordingly. In this context, the erasure of a single server $i \in [n]$ corresponds to not knowing the symbol value of $\tilde{x}_i$ in this codeword. The other symbol values are known and can hopefully be used to retrieve the value of $\tilde{x}_i$.

As a first method of repair, notice that any single erasure can be repaired if there exists only one codeword of $\mathcal{C}$ which matches the remaining symbols [8]. This is the case if $d \geq 2$, since it implies that two codewords differ at least in two positions. It would create a conflict in at least one position if the wrong codeword is chosen. This idea can easily be extended to multiple erasures in the form of cooperative repair, Section 4.3.

Another repair approach exploits the properties of the dual code of $\mathcal{C}$ [9]. In Section 2 we noted that dual codewords form the restrictions of codewords of $\mathcal{C}$. Since the partially erased $\tilde{\mathbf{x}}$ has to be a codeword it must hold that for any $\mathbf{y} \in \mathcal{C}^{\perp}$, $\tilde{\mathbf{x}} \cdot \mathbf{y}^T = 0$, or equivalently $\sum_{j \in supp(\mathbf{y})} x_j = 0$. In case the unknown symbol $\tilde{x}_i$ is a part of this summation the value of $\tilde{x}_i$ can be retrieved. This happens if $\mathcal{C}^{\perp}$ contains a $\mathbf{y}$ which is nonzero on index i, i.e. $y_i \neq 0$. Let us clarify this by an example.

Example 3.1. *Consider the $Ham(2, 3)$ code $\mathcal{C}$ with the parity check matrix H given by;*

$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \boldsymbol{y}_3 \end{pmatrix},$$

with $\boldsymbol{y}_1$, $\boldsymbol{y}_2$ and $\boldsymbol{y}_3$ rows of H. Let $\boldsymbol{x} = (0, 1, 1, 1, 1, 0, 0)$, then computing $\boldsymbol{x}H^T = 0$ implies that $\boldsymbol{x}$ is a codeword. Suppose that $\boldsymbol{x}$ has been stored symbol by symbol in seven different servers of which the fourth is erased. This means that $\tilde{x}_4$ is unknown and has to be retrieved. Instead we know $\tilde{\boldsymbol{x}} = (0, 1, 1, \tilde{x}_4, 1, 0, 0)$ and since it has to be a codeword it should for example satisfy the following restriction imposed by $\boldsymbol{y}_1$:

$$0 = \tilde{\boldsymbol{x}} \cdot \boldsymbol{y}_1^T = (0, 1, 1, \tilde{x}_4, 1, 0, 0) \cdot (1, 0, 0, 1, 1, 0, 1)^T = \tilde{x}_4 + 1,$$

which implies $\tilde{x}_4 = 1$. Notice that for instance taking $\boldsymbol{y}_3$ would not have resulted in reparation of $\tilde{x}_4$, because $\boldsymbol{y}_3$ does not contain a 1 in the fourth position.

The latter approach can be extended to handling multiple erasures in several ways. Concepts of both parallel and sequential recovery will be based on this approach. It is worthwhile to state that cooperative, parallel and sequential repair will lead to the same method when considering a single erasure.

In order to compare the performance of different reparation methods properly it is helpful to discuss some desirable qualities and properties of the methods on which the comparison can be based. As stated in the introduction it is favorable if a reparation

method does not induce much overhead both in terms of storage and transmission. Storage overhead can be measured by the fraction of servers that contains actual information, which is usually expressed by the *information rate*; $\frac{k}{n}$. This ratio does only depend on the parameters of the chosen code and not on the reparation method at use and the number of considered erasures. A common measure for transmission overhead is *locality*, which is the number of servers that need to be accessed in order to repair an erasure. In contrast to the information rate locality does depend on the used method and number of erasures. Let us formally define locality for one erasure.

Definition 3.2. *Let $\mathcal{C}$ be a linear $[n, k, d]_2$-code and $i \in [n]$. A subset $\mathcal{R}_i \subseteq [n] \setminus \{i\}$ is a repair set of i if for every codeword of $\mathcal{C}$ its i-th symbol can be repaired using the symbols indexed by $\mathcal{R}_i$. This is equivalent to the existence of a codeword $\boldsymbol{y} \in \mathcal{C}^\perp$ such that $supp(\boldsymbol{y}) = \mathcal{R}_i \cup \{i\}$.*

Definition 3.3. *Let $\mathcal{C}$ be a linear $[n, k, d]_2$-code, then $\mathcal{C}$ is said to have $(r, 1)$-locality if every $i \in [n]$ has a repair set $\mathcal{R}_i$ of size at most r.*

An improvement in locality comes down to reducing the size of the repair sets and finding the lowest r for given parameters (n, k). Similarly, an increase in the information rate results in a better storage overhead. The following lemma relates storage and transmission overhead and illustrates their trade-off [8].

Lemma 3.4. *Let $\mathcal{C}$ be a linear $[n, k, d]_2$-code with $(r, 1)$-locality, then it satisfies:*

$$\frac{k}{n} \leq \frac{r}{r+1} \Longrightarrow r \geq \left\lceil \frac{k}{n-k} \right\rceil. \tag{3}$$

Codes that satisfy Bound (3) with equality are referred to as optimal in this paragraph. From Lemma 3.4 it follows that for optimal codes an improvement in either locality or information rate must result in an deterioration of the other. Notice that both the duplication code from Chapter 1 and the parity code from Example 2.1 are optimal, while both have completely different code parameters. As an example of non-optimal codes we discuss next the $(r, 1)$-locality of Hamming codes.

Lemma 3.5. *Let $m \geq 2$ and $\mathcal{C}$ be a $Ham(2, m)$, then $\mathcal{C}$ has $(r, 1)$-locality for at best $r = 2^{m-1} - 1$.*

Proof. Let us prove the following assertion: let $m \geq 2$ and $\mathcal{C}$ be a $Ham(2, m)$, then $C^\perp$ contains only words of Hamming weight 2^{m-1} apart from the zero word. Let $\mathbf{y} \in \mathcal{C}^\perp$ nonzero vector, then it can be chosen as a row of some parity check matrix M of $\mathcal{C}$. $\mathcal{C}$ is a $Ham(2, m)$ code, so M contains every nonzero vector of $\mathbb{F}_2^m$ precisely once as a column. We can add a zero column to M without changing the weight of $\mathbf{y}$. Since M now contains each of the 2^m words of $\mathbb{F}_2^m$ precisely once as columns it follows easily that any row of M should contain an equal number of zeros and ones and thus $wt(\mathbf{y}) = 2^{m-1}$. By the assertion and Definition 3.2 we know that every repair set has size precisely $2^{m-1} - 1$. For any $i \in [n]$ we can find a $\mathbf{y} \in \mathcal{C}^\perp$ with $i \in supp(\mathbf{y})$ and thus every $i \in [n]$ has a repair set of size $2^{m-1} - 1$, which proves the lemma. □

4 Methods for Two Erasure Repair

Our goal is to use linear coding for retrieving the content of multiple, mainly two, erased servers. Suppose that we have k information servers within a total of n servers, which all have been indexed by a number $i \in [n]$. Due to malfunctions the content of a few servers is unavailable and needs to be retrieved. The erased servers correspond to a set $\mathcal{E} \subseteq [n]$ of size e called the erasure set. The remaining servers, i.e. $\overline{\mathcal{E}} = [n] \setminus \mathcal{E}$, contain hopefully enough information to retrieve the lost data.

In general three different approaches are distinguished when dealing with multiple erasures. In Section 4.2 presumably the most basic method, parallel recovery, is presented. For this method all erasures are repaired individually. Cooperative repair will be the topic of Section 4.3 and takes all erasures at once into account. Section 4.4 discusses sequential repair in which erasures are repaired subsequently. This method allows already repaired erasures to be used in the repair of other erasures. The focus of this chapter is to state for each method the main definitions and results necessary for the comparison of methods in the next chapter. It is our goal to collect and combine all existing results in a structured way rather than to derive new results.

4.1 General Results

Before treating every method separately let us state some results that hold regardless of the used method. Suppose a linear $[n, k, d]_2$-code $\mathcal{C}$ is used for reparation, then the Hamming distance of the code determines the number of erasures that can be recovered. In fact, if $|\mathcal{E}| \geq d$ then reparation cannot be performed in any case. Suppose a codeword $\mathbf{x}_1 \in \mathcal{C}$ has been stored and there exists another codeword $\mathbf{x}_2 \in \mathcal{C}$ which differs from $\mathbf{x}_1$ in precisely d symbols, then the erasure of precisely these symbols would be fatal. The content of the remaining servers could have come from both $\mathbf{x}_1$ and $\mathbf{x}_2$. Conversely, repair can be performed if the number of erasures is strictly smaller than the Hamming distance. Namely, it cannot happen that the remaining symbols come from two different codewords, since they must differ in at least one position which is still active. Since we require the code to be able to repair two (or more) erasures we have $d \geq 3$, which implies:

$$n - k \geq 2, \tag{4}$$

$$n \leq 2^{n-k} - 1, \tag{5}$$

respectively by the Singleton bound of Lemma 2.6 and by the proof of Lemma 2.10. The latter inequality defines implicitly all (n, k) pairs which allow a linear $[n, k, d]_2$-code to repair two erasures. Notice that Hamming codes satisfy with equality;

$$n = 2^m - 1 = 2^{(2^m-1)-(2^m-m-1)} - 1 = 2^{n-k} - 1.$$

This implies that Hamming codes have optimal storage overhead given a fixed number of parity symbols.

4.2 Parallel Repair: Multiple Disjoint Repair Sets

The first method we discuss for repairing multiple erasures is sub-method of parallel repair which considers every erasure individually. Let $\mathcal{C}$ be a linear $[n, k, d]_2$ code used for parallel recovery and $\mathcal{E} \subseteq [n]$ an arbitrary erasure set of size $1 \leq e < d$ of some codeword $\mathbf{x} = (x_1, x_2, ..., x_n)$. An erasure $i \in \mathcal{E}$ can be recovered if there exists a repair set $\mathcal{R}_i$ which is disjoint with $\mathcal{E}$, i.e. $\mathcal{R}_i \cap \mathcal{E} = \emptyset$. Obviously it is not known beforehand which symbols of $\mathbf{x}$ will erase and thus whether the repair set and erasure set will be disjoint. There exist various ways of designing repair sets for this individual erasure in order to cope with this problem in a parallel way [9]. A possible solution makes sure that each erasure has multiple disjoint repair sets. In fact, if every symbol has e disjoint repair sets, then at least one of these sets is disjoint of $\mathcal{E}$ and is able to repair the symbol. Let us formalize this in the next definition.

Definition 4.1. *A linear $[n, k, d]_2$-code $\mathcal{C}$ is said to have (r, e)-disjoint parallel locality if for every $i \in [n]$ there exist e disjoint repair sets of size at most r.*

Let us explain this method by an example which generalizes the ideas of the parity codes discussed in Example 2.1. This example has been deduced from a simple alteration to the first code construction in [1].

Example 4.2 (Generalized parity code)**.** *In this example we construct a code which builds upon the parity code of Example 2.1 and which is also known as a Product code in coding theory. For $p \in \mathbb{N}$ consider a $(p+1) \times (p+1)$ array P_p of which each row and each column is filled with codewords of the $[p, p+1]$ binary parity code. This means that the upper left $p \times p$ sub-array can be filled arbitrarily and corresponds to the information servers. The rightmost column is filled such that each row has an even number of 1's and the bottom row such that each column has an even number of 1's. This does not lead to a conflict in the lower right symbol, since it makes the total number of 1's in P_p even. The array P_2 is given by:*

$$P_2 = \left(\begin{array}{cc|c} p_{1,1} & p_{1,2} & \sum_j p_{1,j} \\ p_{2,1} & p_{2,2} & \sum_j p_{2,j} \\ \hline \sum_i p_{i,1} & \sum_i p_{i,2} & \sum_i \sum_j p_{i,j} \end{array}\right), \; p_{i,j} \in \mathbb{F}_2, \; \forall i, j \in [p].$$

Remark that this code consists of codewords of length $n = (p+1)^2$ with $k = p^2$ information symbols. A codeword can be seen as the concatenation of the rows of P_p into a vector. Linearity of the code follows from linearity of the parity code, which implies that its minimum Hamming distance equals minimum weight of a non-zero codeword being four. So far we have defined the codewords completely by putting restrictions upon the rows and columns. This means that a parity check matrix is easy to find. Each restriction forms a row; $n - k = (p+1)^2 - p^2 = 2p + 1$ in total. They follow from the upper p rows, the left p columns and the summation of all symbols. For P_2 this results respectively in the rows

of the following parity check matrix H_2:

$$H_2 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

The parity restrictions of the lowest row and rightmost column corresponding to rows;

$$\begin{aligned} \boldsymbol{y}_1 &= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}, \\ \boldsymbol{y}_2 &= \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

follow clearly from linear combinations of rows of H_2. Notice that each $p_{i,j} \in P_p$ is part of two different parity codes and that it has two disjoint repair sets of size p; the other symbols both in its row and in its column. In short, this is a construction of a linear $[(p+1)^2, p^2, 4]_2$-code with $(p, 2)$-disjoint parallel locality and information rate $(\frac{p}{p+1})^2$. Remark that $d = 4$ implies that this code is capable of repairing three erasures, whereas only locality for two erasures has been proven so far. In Section 4.4 we will revise this code using sequential repair and show it can sequentially repair three erasures.

Ideally the repair sets are as small as possible in order to reduce transmission overhead. Our goal is to compute the lowest possible r-value such that there exists a linear $[n, k, d]_2$ code with $(r, 2)$-disjoint parallel locality. This corresponds to finding optimal transmission overhead given a predetermined storage overhead. Suppose an information rate is given by the pair (n, k), then the following lemma states that the size of repair sets cannot be arbitrarily small [1].

Lemma 4.3. *Let $\mathcal{C}$ be a linear $[n, k, d]_2$-code with $(r, 2)$-disjoint parallel locality, then it satisfies:*

$$\frac{k}{n} \leq \frac{1}{(1 + \frac{1}{r})(1 + \frac{1}{2r})} = \frac{r^2}{(r+1)(r+\frac{1}{2})}. \tag{6}$$

Special interest goes out to those codes which satisfy Inequality (6) with equality, because their locality is optimal given an information rate. The information rate of code constructed in Example 2.1 is not optimal in the sense of Lemma 4.3, but comes arbitrarily close as p increases. Notice that the right hand side of the inequality is an increasing function of r. Therefore, the lemma induces a lower bound on the r-value such that there exists a linear $[n, k, d]_2$-code with $(r, 2)$-disjoint parallel locality. This lower bound is implicitly given by the inverse of the increasing function evaluated in $\frac{k}{n}$ and can be computed by solving the quadratic inequality for r. The total number of symbols that need to be accessed in order to resolve two erasures is given by at most $r \cdot e$.

The restriction on a code that each symbol should have multiple disjoint repair set is confining. The code should contain sufficiently many parity symbols and the information rate cannot be too large in order to satisfy this restriction. In Section 4.1 we showed

that Hamming codes have the lowest information rate given a number of parity symbols. Consequently, the condition of multiple repair sets is too restrictive for Hamming codes as we observe in the following lemma.

Lemma 4.4. *Let $m \geq 3$ and $\mathcal{C}$ be a $Ham(2, m)$, then $\mathcal{C}$ does not have $(r, 2)$-disjoint parallel locality.*

Proof. Let $\mathcal{E} = \{i, j\}$ be two erasures, then we show that we cannot find two disjoint repair sets for erasure i. Suppose this is in fact possible and they are $\mathcal{R}_1 \cap \mathcal{R}_2 = \emptyset$. These corresponds naturally to dual code words $\mathbf{y}_1, \mathbf{y}_2 \in \mathcal{C}^\perp$. The proof of Lemma 3.5 shows that every codeword of $\mathcal{C}^\perp$ has weight 2^{m-1}. This implies $|\mathcal{R}_1 \cup \{i\}| = |supp(\mathbf{y}_1)| = 2^{m-1}$ and $|\mathcal{R}_1| = 2^{m-1} - 1$. Similar results hold for $\mathbf{y}_2$. By linearity of (the dual of) the Hamming code $\mathbf{y}_1 + \mathbf{y}_2$ should be in $\mathcal{C}^\perp$. Notice that $wt(\mathbf{y}_1 + \mathbf{y}_2) = |supp(\mathbf{y}_1 + \mathbf{y}_2)| = |\mathcal{R}_1 \cup \mathcal{R}_2| = |\mathcal{R}_1| + |\mathcal{R}_2| = 2^m - 2$, because $\mathcal{R}_1$ and $\mathcal{R}_2$ are disjoint. This is in contraction with every codeword of $\mathcal{C}^\perp$ having weight 2^{m-1}, which completes the proof. □

It has already been demonstrated that Hamming codes have Hamming distance equal to three and should therefore be capable of repairing two erasures.

Remark 4.5. *The disjoint parallel method has turned out to be a quite restrictive method of repair. For instance with Hamming codes, we have proven that $d > e$ does not necessarily imply that e erasures can be repaired using this method. Similar results follow from Example 4.2. The cooperative and sequential repair methods discussed in the next sections will make sure that a code with Hamming distance d can always repair $e < d$ erasures.*

Remark 4.6. *So far we have only discussed the disjoint parallel method in which each symbol is protected by several disjoint repair sets. There exist more parallel approaches which will not be under discussion in this thesis. For example, each symbol can be protected in parallel by a repair single set of size $r + e - 1$ such that it remain a repair set even tough up to $e - 1$ symbols in this set might be erased.*

4.3 Cooperative Repair

Let $\mathcal{C}$ be a linear $[n,k,d]_2$-code and $\mathcal{E} \subseteq [n]$ an arbitrary erasure set of size $1 \leq e < d$ of some codeword $\mathbf{x} = (x_1, x_2, ..., x_n)$. This means that the symbols $x_i, i \in \mathcal{E}$ are erased and $x_i, i \in \overline{\mathcal{E}}$ are known. Notice that we can repair all erasures of this codeword at once if there exists only one codeword in $\mathcal{C}$ that matches all remaining symbols. While this might seem a plausible solution we need to check all remaining symbols which might be a tedious job in practice. In order to reduce the transmission overhead we are interested whether a subset of $\overline{\mathcal{E}}$ suffices as well. In fact, a subset $\mathcal{R} \subseteq \overline{\mathcal{E}}$ is able to cooperatively repair all erasures in $\mathcal{E}$ if all codewords that agree on $\mathcal{R}$ also agree on $\mathcal{E}$. Using the linearity of $\mathcal{C}$ we summarize this in the following formal definition of a cooperative repair set:

Definition 4.7. *Let $\mathcal{C}$ be a linear $[n,k,d]_2$ code. A subset $\mathcal{R} \subseteq \overline{\mathcal{E}}$ is a cooperative repair set for $\mathcal{E} \subseteq [n]$ if all codewords that are zero on $\mathcal{R}$ are also zero on $\mathcal{E}$*

There exist many different and equivalent definitions of a cooperative repair set and one of these equivalences is stated in the following lemma [8].

Lemma 4.8. *Let $\mathcal{C}$ be a linear $[n,k,d]_2$ code, then the following statements are equivalent:*

- *A subset $\mathcal{R} \subseteq \overline{\mathcal{E}}$ is a cooperative repair set for $\mathcal{E} \subseteq [n]$.*
- *Every codeword which is zero on $\mathcal{R}$ is also zero on $\mathcal{E}$.*
- *The subspace of the columns indexed by $\mathcal{E}$ of a generator matrix G of $\mathcal{C}$ is in the subspace of the columns indexed by $\mathcal{R}$.*

Proof. The first two statements are equivalent by definition and therefore the equivalence of the second and third statement remains to be proven. Let G be a generator matrix of $\mathcal{C}$ with columns $\{\mathbf{g}_1, \mathbf{g}_2, ..., \mathbf{g}_n\}$. Every codeword $\mathbf{x} = (x_1, ..., x_n)$ of $\mathcal{C}$ can be written as $\mathbf{x} = \mathbf{u}G$ for some $\mathbf{u} \in \mathbb{F}_2^k$ and we have $x_i = \mathbf{u} \cdot \mathbf{g}_i$. Let $\mathcal{E} \subset [n]$ and $\mathcal{R} \subset [n] \setminus \mathcal{E}$ and suppose that every codeword which is zero on $\mathcal{R}$ is also zero on $\mathcal{E}$. This is equivalent to $x_i = \mathbf{u} \cdot \mathbf{g}_i = 0$ for $i \in \mathcal{E}$ if $x_j = \mathbf{u} \cdot \mathbf{g}_j = 0$ for $j \in \mathcal{R}$. In other words, $\mathbf{u}$ is orthogonal to all columns indexed by $\mathcal{E}$ if $\mathbf{u}$ is orthogonal to all columns indexed by $\mathcal{R}$. This is the case if and only if the subspace of every column indexed by $\mathcal{E}$ of G is in the subspace of the columns indexed by $\mathcal{R}$. □

Obviously we do not know beforehand which servers will be erased and therefore ideally all possible combinations of erasures have to be reparable. Moreover, we wish to minimize the number of servers needed for repair or equivalently the size of a cooperative repair set. These two qualities of a code $\mathcal{C}$ have been combined in the definition of (r,e)-cooperative locality.

Definition 4.9. *Let $\mathcal{C}$ be a linear $[n,k,d]_2$-code. $\mathcal{C}$ is said to have (r,e)-cooperative locality if for every subset $\mathcal{E} \subseteq [n]$ of size $1 \leq e < d$ there exists a cooperative repair set $\mathcal{R}$ for $\mathcal{E}$ of size at most r.*

Intuitively a code has (r, e)-cooperative locality if it has the ability of repairing e erasures at once using at most r active servers. Next we discuss a linear code of which we determine (r, e)-cooperative locality. This example will be useful in the sequel for tightening bounds on the cooperative locality of linear $[n, k, d]_2$-codes.

Example 4.10. *Let us determine (r, e)-cooperative locality for a linear $[10, 5]_2$-code $\mathcal{C}$ with the following generator matrix G:*

$$G = \Big(I_5|A\Big) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Notice that G is in standard form and that a parity check matrix is given by: $H = \Big(A^T|I_5\Big)$. Notice furthermore that the columns of H are a permutation of the columns of G. Applying Lemma 2.10 to the columns of G results in $d = e - 1 = 3$. In order to compute r we randomly select two columns $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ of G as an erasure set $\mathcal{E}$ and apply Lemma 4.8. It can be easily observed that always four columns $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ and $\boldsymbol{v}_4$ can be found which coincide a repair set for $\mathcal{R}$. This implies that $\mathcal{C}$ is a linear $[10, 5, 3]_2$-code with $(4, 2)$-cooperative locality.

As stated in the introduction every repair method has a balance between transmission and storage overhead. For cooperative repair (r, e)-cooperative locality is a measure of transmission overhead. Storage overhead is measured as usual by the information rate and induced solely by the code. In the sequel of this section we discuss ways of optimizing transmission overhead when the storage overhead is fixed. More specifically, our goal is to find for every valid pair (n, k) the lowest possible r-value such that there exists a linear $[n, k, d]$-code with $(r, 2)$-cooperative locality.

Let us first focus on codes with high information rate. In Section 4.1 we determined that such codes include Hamming codes, since for a fixed number of parity symbols (≥ 2) Hamming codes have the highest information rate. In [10] a useful theorem has been proven regarding the cooperative locality of Hamming codes.

Theorem 4.11. *Let $m \geq 2$, then $Ham(2, m)$ has $(r, 2)$-cooperative locality for $r \geq 3 \cdot 2^{m-2} - 2$, but not for $r \leq 3 \cdot 2^{m-2} - 3$.*

Corollary 4.12. *For each pair $(n, k) = (2^m - 1, 2^m - m - 1)$, $m \geq 2$ the smallest r such that there exists a linear $[n, k, d]$-code with $(r, 2)$-cooperative locality is $r = 3 \cdot 2^{m-2} - 2$.*

Proof. Theorem 4.11 shows that $Ham(2, m)$ suffices these conditions. It remains to prove there does not exist another code with a better r. We will do so by showing that $Ham(2, m)$ is the only possible code with parameters $(n, k) = (2^m - 1, 2^m - m - 1)$, $m \geq 2$ and having $(r, 2)$-cooperative locality. It has already been established in Section 4.1 that any such code should have $d \geq 3$ and a parity check matrix of size $m \times (2^m - 1)$.

Theorem 2.7 implies that this is only possible if the parity check matrix contains every word of $\mathbb{F}_2^m$ exactly once as columns and thus by Definition 2.9 if such a code is a $Ham(2, m)$ code. □

By Corollary 4.12 we find the optimal r-value for the pairs of (n, k) for which there exists a Hamming code. For example, $Ham(2, 4)$ with $(n, k) = (15, 11)$ has $(10, 2)$-cooperative locality and there does not exist a linear code with these parameters and $(r, 2)$-cooperative locality for $r \leq 9$. In Table 1 we list the optimal r-value for some pairs of (n, k) that permit a Hamming code.

m	2	3	4	5	6	7	8
(n, k)	(3, 1)	(7, 4)	(15, 11)	(31, 26)	(63, 57)	(127, 120)	(255, 247)
Optimal r	1	4	10	22	46	94	190

Table 1: $(r, 2)$-cooperative locality achieved by Hamming codes. Moreover, any (n, k) pair which allows a Hamming code and which has $(r, 2)$-cooperative locality must have at least this r value.

Table 1 can be seen as an optimal trade off in which we favor storage overhead over transmission overhead. This comes at the price of a high transmission overhead. For example, for a $Ham(2, 7)$ code 94 out of 127 ($\approx 74.0\%$) servers need to be consulted in order to repair only two erasures, but 120 out of 127 ($\approx 94.5\%$) servers are information servers. Notice that when m increases storage overhead improves relatively, whereas transmission overhead weakens. Letting m tend to infinity we have:

$$\begin{aligned} \frac{k}{n} &= \frac{2^m - m - 1}{2^m - 1} \longrightarrow 1, \\ \frac{r}{n} &= \frac{3 \cdot 2^{m-2} - 2}{2^m - 1} \longrightarrow \frac{3}{4}. \end{aligned}$$

By far not all valid (n, k) pairs allow a Hamming code and therefore we still have many unknown optimal r-values. The following results have been proven in [10] and give some other optimal r-values.

Theorem 4.13. *Let $m \geq 3$, $1 \leq s \leq m$ and $\mathcal{C}$ be a linear $[2^m - 1 - s, 2^m - m - 1 - s, d]$-code which is achieved by shortening a $\mathcal{C}' = Ham(2, m)$ code in such a way that the removed columns of the parity check matrix of $\mathcal{C}'$ form an independent set of size s. Then $\mathcal{C}$ has $(r, 2)$-cooperative locality with $r = 3 \cdot 2^{m-2} - s - 2$.*

Theorem 4.14. *Let $m \geq 3$, $1 \leq s \leq 2^m - 2 - m$ and $\mathcal{C}$ be a linear $[2^m - 1 - s, 2^m - m - 1 - s, d]$-code which is achieved by shortening a $\mathcal{C}' = Ham(2, m)$ code precisely s times. Then $\mathcal{C}$ does not have $(r, 2)$-cooperative locality with $r \leq 3 \cdot 2^{m-2} - s - 3$.*

Corollary 4.15. *Let $m \geq 2$ and $1 \leq s \leq m$, then for each pair $(n, k) = (2^m - 1 - s, 2^m - m - 1 - s)$ the lowest r such that there exists a linear $[n, k, d]$-code with $(r, 2)$-cooperative locality is $r = 3 \cdot 2^{m-2} - s - 2$.*

Proof. Let $m \geq 3$ and $1 \leq s \leq m$, then there always exist s independent columns in a parity check matrix of $Ham(2, m)$; for example, take s unit vectors. This implies that a code $\mathcal{C}$ as described in Theorem 4.13 can always be found. Thus, for each pair $(n, k) = (2^m - 1 - s, 2^m - m - 1 - s)$ there exists a linear $[n, k, d]$-code having $(r, 2)$-cooperative locality for $r = 3 \cdot 2^{m-2} - s - 2$. It remains to prove that this r-value is indeed the lowest. Let $\mathcal{C}$ be a linear $[n, k, d]$-code with $(n, k) = (2^m - 1 - s, 2^m - m - 1 - s)$ for some $m \geq 3$, $1 \leq s \leq m$, then by Lemma 2.10 it must be a s times shortened Hamming code. Since $m \geq 3$ we have $s \leq m \leq 2^m - m - 2$ and therefore Theorem 4.14 implies that $\mathcal{C}$ does not have $(r, 2)$-cooperative locality for $r \leq 3 \cdot 2^{m-2} - s - 3$, which proves this corollary. □

Table 2 lists for some (n, k) pairs the optimal r-value such that there exists a linear $[n, k, d]$-code with $(r, 2)$-cooperative locality based on the findings in Corollaries 4.12 and 4.15. The lowest value in each column corresponds to the value in Table 1. For the other (n, k) pairs a range has been deduced where the optimal r-value has to be within. The code discussed in Example 4.10 forms a minor improvement on these ranges. The bounds follow from the following observations;

- Let k be fixed and $n-k$ increase, then the optimal r-value cannot increase. In other words, moving to the right in the table cannot increase the value.

- Let $n - k$ be fixed and k decrease, then the optimal r-value cannot increase. In other words, moving up in the table cannot increase the value.

- Let $n - k$ be fixed and k increase, then the optimal r-value cannot decrease. In other words, moving down in the table cannot decrease the value.

- Let $n - k$ be fixed and k decrease by one, then it follows from Corollary 4.15 that optimal r-value can not decrease by more than one. In other words, moving up by one step in the table cannot decrease the value by more than one.

- Let $n-k$ be fixed and k increase by one, then it follows from the previous observation that optimal r-value can not decrease by more than one. In other words, moving down by one step in the table cannot increase the value by more than one.

Remark 4.16. *So far we have only discussed linear codes with Hamming distance $d = 3$. In case we favor storage overhead over transmission overhead this does not pose a problem, since a higher Hamming distance induces more parity servers. Usually, this will not lead to a better storage overhead. However, if the trade-off is balanced more towards transmission overhead we might have to let go of the restriction $d = 3$. Loosening this restriction might result in a lower locality than can be achieved by setting $d = 3$. Notice that increasing the Hamming distance simultaneously allows more erasures to be repaired, which is not the primary focus of this thesis.*

$k, n-k$	2	3	4	5
1	**1**	**1**	1	1
2		**2**	1-2	1
3		**3**	2-3	1-2
4		**4**	3-4	1-3
5			4-5	1-4*
6			5-6	2-5
7			**6**	3-6
8			**7**	4-7
9			**8**	5-8
10			**9**	6-9
11			**10**	7-10
12				8-11
13				9-12
14				10-13
15				11-14
16				12-15
17				13-16
18				14-17
19				15-17
20				16-17
21				**17**
22				**18**
23				**19**
24				**20**
25				**21**
26				**22**

Table 2: Optimal r-values for some $(k, n-k)$ pairs from Corollaries 4.12 and 4.15 in bold. Upper and lower bounds on the optimal r-value for $(k, n-k)$ pairs deduced from the values in bold. Example 4.10 sharpens the upper bound from $(*)$.

4.4 Sequential Repair

Another approach for repairing erased servers which has been introduced recently is sequential repair [1]. As the name suggests, a code has the ability of sequentially repairing erasures if there exists an order in which the erasures are repaired. In a similar way as with parallel repair each erasure is considered individually, but sequential repair allows repaired erasures to be part of a repair set of yet to repair erasures. This makes the sequential method more general than the parallel methods and suggests that it can potentially repair more erasures than disjoint parallel repair with the same locality. Recall Definition 3.2 of a repair set, then we define (r, e)-sequential locality.

Definition 4.17. *Let $\mathcal{C}$ be a linear $[n, k, d]_2$-code, then $\mathcal{C}$ is said to have (r, e)-sequential locality if for every erasure set $\mathcal{E} \subseteq [n]$ of size $|\mathcal{E}| \leq e$ there exists an order $\mathcal{E} = (i_1, i_2, ..., i_{|\mathcal{E}|})$ such that each $i_j \in \mathcal{E}, 1 \leq j \leq |\mathcal{E}|$ has a repair set $\mathcal{R}_{i_j} \subseteq \overline{\mathcal{E}} \cup \{i_1, i_2, ..., i_{j-1}\}$ of size $|\mathcal{R}_{i_j}| \leq r$.*

Remark 4.18. *The r in (r, e)-sequential locality denotes the size of the largest repair set of an individual erasure. The total number of accessed servers during the reparation procedure can be overestimated by $r \cdot e$. This last number compares to the r of (r, e)-cooperative locality, since both involve the repair of e erasures. However, the overestimation $r \cdot e$ often results in an unfair comparison. This problem is addressed by the next lemma and corollary.*

Note that if in all stages of the reparation procedure there exists an erased server which can be repaired, then eventually all erasures can be repaired sequentially. In [1] the authors proved this equivalent definition of (r, e)-sequential locality in the form of the following lemma. It can be used to compute the total number of accessed servers during repair instead of the overestimation $r \cdot e$.

Lemma 4.19. *Let $\mathcal{C}$ be a linear $[n, k, d]_2$-code, then $\mathcal{C}$ has (r, e)-sequential locality if and only if for any nonempty $\mathcal{E} \subseteq [n]$ of size $|\mathcal{E}| \leq e$, there exists an $i \in \mathcal{E}$ such that i has a repair set $\mathcal{R} \subseteq \overline{\mathcal{E}}$ of size $|\mathcal{R}_i| \leq r$.*

Corollary 4.20. *Let $e \geq 1$ and $\mathcal{C}$ be a linear $[n, k, d]_2$-code with (r_t, t)-sequential locality for $1 \leq t \leq e$ and take r_t as low as possible. Then $\mathcal{C}$ can sequentially repair any e erasures by accessing in total $\sum_{t=1}^{e} r_t \leq r \cdot e$ symbols.*

Proof. If $1 \leq t \leq e$ erasures are still unresolved, then one of the remaining erasure can be repaired by accessing r_t servers. This results in a total of $\sum_{t=1}^{e} r_t$ accessed servers. Notice that r_t is a non-decreasing sequence and thus $\sum_{t=1}^{e} r_t \leq r \cdot e$. Namely, if $\mathcal{C}$ has (r_t, t)-sequential locality, then it most certainly has $(r_t, t-1)$-sequential locality. □

The power of Corollary 4.20 lies in the observation that for many codes r_t is a non-decreasing sequence, which shows that the actual number of accessed servers is a lot better than the overestimation $r \cdot e$.

Example 4.21 (Generalized parity code, continuation)**.** *Recall Example 4.2 with the construction of a linear* $[(p+1)^2, p^2, 4]_2$*-code for* $p \in \mathbb{N}$ *via an array filled with parity codes as rows and columns. Notice that any three erasures can be repaired, since there is always at least one erasure which is part of a column or row which does not contain one of the other erasures. This implies that this erasure can be repaired and thus all three erasures can sequentially be repaired. For example, suppose that in the following array* $p_{2,2}$*,* $p_{2,5}$ *and* $p_{3,2}$ *have been erased;*

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & \tilde{p_{2,2}} & 1 & 1 & \tilde{p_{2,5}} \\ 0 & \tilde{p_{3,2}} & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

The fifth column forms a repair set for $\tilde{p_{2,5}}$ *and we find* $\tilde{p_{2,5}} = 0$*. This allows* $\tilde{p_{2,2}}$ *to be recovered, since at first both its repair sets contained other erased symbols. The last two erasures are recovered as* $\tilde{p_{2,2}} = 1$ *and* $\tilde{p_{3,2}} = 1$ *by using the second and third rows respectively as repair sets. For the sequel it is important to note that the erasure of* $p_{3,5}$ *would have been fatal. The positions of the erasures form a square and make sure that for each erasure both repair sets contain an erasure as well.*

Next we discuss two ways in which we can generalize the ideas of this example. First of all, changing the sizes of the array introduces repair sets of different sizes. Let $1 \leq p_1 < p_2$ *and let* P *be a* $(p_1+1) \times (p_2+1)$ *array defined such that both rows and columns are parity codes of appropriate length. This means each symbol has a repair set of size* p_1 *and of size* p_2*. This has the advantage that we can use a smaller repair set in case only one erasure occurs. A second and third erasure can be recovered using a repair set of size either* p_1 *or* p_2 *depending on the erasure positions in the array. In case it is assumed that erasures happen independently of each other and with low probability, then most of the erasure can be recovered using the smaller repair set. This advantage in transmission overhead comes at the price of losing storage overhead. For these codes it is easy to see that the information rate is optimal in the square case.*

Secondly, let us generalize the concept to a higher dimension in order to repair more (than three) erasures. As a starting point one can define this code as a cube with parity codes in three directions instead of a square. This would result in a code with $(p, 3)$*-disjoint parallel locality and* $(p, 2^3 - 1 = 7)$*-sequential locality. In case seven or less erasures occur we can always find an erasure with a fully active repair set. Let us clarify this as follows; notice that the smallest number of erasures which cannot be resolved is eight, because they can be arranged in the form of a* $2 \times 2 \times 2$ *cube. This means that for each erasure its three repair sets contain an erasure as well and is similar to the squared array case. In general, let us solely state that expanding in an identical way to dimension* $s \in \mathbb{N}$ *would result in a linear* $[(p+1)^s, p^s, 2^s]_2$*-code. Hamming distance* $d = 2^s$ *implies that up to* $2^s - 1$ *erasures can be resolved.*

The goal for this section is similar to the cooperative approach; to find for every valid

pair (n, k) the lowest possible r-value such that there exists a linear $[n, k, d]_2$-code with $(r, 2)$-sequential locality. The following lemma gives an upper bound for the information rate and a lower bound on the sequential locality of such codes [1].

Lemma 4.22. *Let $\mathcal{C}$ be a linear $[n, k, d]_2$-code with $(r, 2)$-sequential locality, then it satisfies:*

$$\frac{k}{n} \leq \frac{r}{r+2} \Longrightarrow r \geq \left\lceil 2\frac{k}{n-k} \right\rceil. \tag{7}$$

Codes that satisfy Inequality (7) with equality are optimal in a way that for given storage overhead, i.e. $\frac{k}{n}$, their $(r, 2)$-sequential locality is as low as possible. Hamming codes have turned out to be optimal in the cooperative case and therefore we start by analyzing lower bounds on the sequential locality for (n, k) pairs which allow a Hamming code. Combining the results of Lemmas 3.4, 4.22 and Corollary 4.20 gives that the total number of accessed servers for the sequential repair of two erasures can be lower bounded by $\lceil \frac{2k}{n-k} \rceil + \lceil \frac{k}{n-k} \rceil$. In other words, any linear $[n, k, d]_2$ code needs to access at least this many symbols to sequentially repair two erasures.

$n-k$	2	3	4	5	6	7	8
(n, k)	(3, 1)	(7, 4)	(15, 11)	(31, 26)	(63, 57)	(127, 120)	(255, 247)
$r_1 + r_2$	2	4	9	17	29	53	93

Table 3: Lower bound on the total number of accessed servers using sequential repair of two erasures for (n, k) pairs which allow a Hamming code. $r_1 = \lceil \frac{k}{n-k} \rceil$, $r_2 = \lceil 2\frac{k}{n-k} \rceil$.

Table 3 shows a lower bound on the total number of accessed symbols for some (n, k)-pairs. This does not imply that there exists a code which actually reaches this locality. As long as this has not been proven, sequential repair has only the potential of this locality. Next we prove that this lower bound is not reached by Hamming codes.

Lemma 4.23. *Let $m \geq 2$, then $Ham(2, m)$ has $(r, 2)$-sequential locality for at best $r = 2^{m-1} - 1$.*

Proof. Let $m \geq 2$ and $\mathcal{C}$ be a $Ham(2, m)$, then by the proof of Lemma 3.5 we know that $C^\perp$ contains only words of Hamming weight 2^{m-1} apart from the zero word. This implies that every repair set has size precisely $2^{m-1} - 1$. It remains to prove that for every pair of two erasures we can find suitable repair sets. Let $\mathcal{E} = \{i, j\} \subset [n]$ be two erased symbols. It suffices to show that there exists a $\mathbf{y} \in \mathcal{C}^\perp$ such that $i \in supp(\mathbf{y})$ and $j \notin supp(\mathbf{y})$ or vice versa. Suppose that such a dual codeword cannot be found, then for all $\mathbf{y} = (y_1, ..., y_n) \in \mathcal{C}^\perp$ it must hold that $y_i = y_j$. This implies that columns i and j of a parity check matrix of $\mathcal{C}$ agree, which is obviously in contradiction with $\mathcal{C}$ being a Hamming code. This proves that we can find a $\mathbf{y} \in \mathcal{C}^\perp$ with $i \in supp(\mathbf{y})$ and $j \notin supp(\mathbf{y})$ and thus $\mathbf{y}$ forms a repair set of i of size $2^{m-1} - 1$. Lemma 3.5 proves that erasure j can be resolved as well using $2^{m-1} - 1$ symbols, which finalizes the proof of this lemma. □

By Corollary 4.20 it is known that a total number of at most $2^m - 2$ servers need to be consulted for the sequential repair of two erasures using a $Ham(2, m)$ code. For small values of m these have been listed in Table 4. Hamming codes are the only possible codes on these (n, k)-pairs and we conclude that Table 3 does not provide a useful lower bound.

m	2	3	4	5	6	7	8
(n, k)	(3, 1)	(7, 4)	(15, 11)	(31, 26)	(63, 57)	(127, 120)	(255, 247)
r	2	6	14	30	62	126	254

Table 4: Number of accessed symbols for the repair of two erasures using Hamming codes.

Remark 4.24. *The authors of [1] have proven the following bound on the information rate for linear $[n, k, d]_2$-code with $(r, 3)$-sequential locality:*

$$\frac{k}{n} \leq \left(\frac{r}{r+1}\right)^2. \tag{8}$$

Notice that the code constructed in Example 4.2 satisfies this bound with equality and has therefore an optimal balance between storage overhead and transmission overhead in the repair of three erasures.

For the repair of an arbitrary number of erasures it is worthwhile to state that the author of [2] proved an upper bound on the information rate for linear $[n, k, d]_2$-code with general (r, e)-sequential locality. This upper bound coincides for $e \in \{1, 2, 3\}$ respectively with Lemmas 3.4, 4.22 and Inequality (8). Furthermore it is shown to be an achievable bound. Unfortunately, this very recent publication arrived too late for the results to be incorporated in this thesis.

5 Comparison of Methods

In the previous chapter several locality approaches for erasure repair have been reviewed. In order to compare the methods appropriately it is first needed to clarify what properties of the methods we will be comparing. Of course, the number of erasures that a method is able to repair in any case is a basic quality. After this discussion we restrict our attention to the case of repairing two erasures. The main objective is to compare the methods in their ability to reduce storage and transmission overhead. It has already been demonstrated that the improvement of either storage or transmission overhead results in an deterioration of the other. The information rate as a measure of storage overhead only depends on the considered code and not on the used method. Transmission overhead, however, depends heavily on the used method and number of erasures. This makes it evident to fix storage overhead and optimize transmission overhead for the various methods. More precisely, for each (n, k) pair the lowest r-value is determined such that there exists a linear $[n, k, d]_2$ code with $(r, 2)$-locality. Obviously, this locality depends on the reparation method and is used to determine the total number of accessed symbols in order to repair the erasures.

5.1 Number of Erasures

It should first be recalled that for the repair of one erasure all methods coincide and that any difference between methods only occurs when considering multiple erasures. In Section 4.1 it has been determined that a code which is able to repair any e erasures should have $e < d$. It follows from the definition of cooperative locality that a (linear) code with minimum Hamming distance d can always repair $e = d - 1$ erasures using the cooperative method. This is directly the maximum number of erasures that can be recovered by any approach. The generalized parity code in Example 4.2 showed that this is not necessarily the case for the disjoint parallel approach; any s erased servers can be recovered with the disjoint parallel method, whereas for this code $d = 2^s$. The restriction of multiple disjoint repair sets turns out to be restrictive for the number erasures the method can always repair. The same example shows clearly the power of being able to use already repaired symbols, since with the sequential approach an optimal number of $2^s - 1$ erasures can be repaired. Notice that the cooperative method is able to repair the same number of erasures, but possibly with different locality.

5.2 Quantitative Comparison for Hamming Codes

Before comparing the performance of the methods specifically for Hamming codes let us recall what makes considering Hamming codes interesting in the context of erasure repair. For a fixed number of parity symbols Hamming codes have the highest storage overhead reachable for codes that are able to repair two erasures. This can be explained as follows. It has been demonstrated in Section 4.1 that a code with the ability of repairing two erasures should satisfy $d \geq 3$, which in turn implies $n \leq 2^{n-k} - 1$. Hamming codes satisfy this inequality with equality which gives them in the highest storage overhead

possible for codes which can repair two erasure and have a predetermined number of parity symbols. They are in fact the only codes with this property. Namely, by the proof of Corollary 4.12 we know that Hamming codes are the only codes with parameters $(n, k) = (2^m - 1, 2^m - m - 1)$ for $m \geq 2$ and thus the only codes with $n = 2^{n-k} - 1$.

In Lemma 4.4 it has been proven that the condition of two disjoint repair sets is too confining for Hamming codes and that they cannot be used for disjoint parallel repair of two erasures. On the other hand, the cooperative and sequential method can repair two erasures with respectively a minimal access of $r = 3 \cdot 2^{m-2} - 2$ versus $r = 2^m - 2$ symbols. For small values of m these values are listed in Table 5. Notice that the cooperative method needs access to approximately 25% less symbols than the sequential method. This can be explained by the fact that for the sequential method the sizes of the repair sets are added all together. This is done regardless of symbols which might be part of multiple repair sets and are thus counted double. Since we are dealing with Hamming codes all dual codewords have weight 2^{m-1}, which implies that for any two dual codewords there are precisely 2^{m-2} indices on which have both a 1. Otherwise the sum of these two dual codewords would not have weight 2^{m-1} as well which violates the linearity of the dual code. This means 2^{m-2} symbols are counted double, which is indeed approximately 25% of $r = 2^m - 2$. One might argue that it is not necessary to count these symbols double, since once the content of some symbol is known it can be used repeatedly. However, reusing the content of a server would imply that two erasures are recovered cooperatively. In case a strict distinction is made between cooperative and sequential repair the differences in Table 5 remain present. If, however, we allow the properties of both approaches to be combined into a single method, then the values in this table would all agree with the ones of the cooperative method.

$m = n - k$	2	3	4	5	6	7	8
(n, k)	(3, 1)	(7, 4)	(15, 11)	(31, 26)	(63, 57)	(127, 120)	(255, 247)
Seq, r	2	6	14	30	62	126	254
Coop, r	1	4	10	22	46	94	190

Table 5: Total number of accessed symbols r for the repair of two erasures using Hamming codes for cooperative and sequential method.

5.3 Qualitative Comparison of Cooperative and Sequential Method

Notice that the disjoint parallel approach is a special case of sequential repair and therefore this section solely considers the cooperative and sequential methods. Let us discuss and compare the strengths of both methods starting with the cooperative method. In doing so the focus will be on qualitative properties of the methods rather than quantitative properties.

The cooperative method efficiently uses the access to symbols. It considers the erasures all at once instead of individually and builds a single repair set for all erasures which rules

out the possibility of accessing the same symbols multiple times. This has the advantage that the size of the repair set does not necessarily grow linearly with the number of erasures. For example, Lemma 3.5 and Theorem 4.11 show that the access to symbols increases by approximately 50% when cooperatively repairing two erasures instead of one erasure; $3 \cdot 2^{m-2}$ versus $2^{m-1} - 1$ symbols. The quality of cooperative repair of not having to access symbols multiple times resulted in the good transmission overhead of Table 5 compared to sequential repair, which - strictly speaking - cannot reuse access.

The sequential repair method works at its full potential if already repaired erasures are used in the repair of other erasures. Example 4.2 showed that the generalized parity code was able to repair more erasures with sequential repair than in disjoint parallel case, because already repaired erasures were used. Moreover, it was argued that the total number of accessed symbols could be reduced compared to the cooperative case if the code was not designed with a squared array. Suppose the matrix P of this example has size $(p_1+1) \times (p_2+1)$ with $p_1 \gg p_2$, which means the columns are much bigger than the rows. Suppose furthermore that two erasures occur in the same row. Using sequential repair the first and second erasure can clearly be repaired by two repair sets of size respectively p_1 and p_2. However, using cooperative repair we have to access at least two columns or equivalently $2p_1$ symbols, since a cooperative repair set cannot contain already repaired erasures. For the not squared generalized parity code the sequential method performs much better than the cooperative method, because the sequential method is capable of using already repaired erasures.

6 Conclusions and Future Work

In this thesis, we examined various repair methods for two erasures using coding. We aimed at comparing the parallel, cooperative and sequential repair methods in terms of storage and transmission overhead in the search for an optimal method in case of two erasures. This means that we searched for combinations of method and code which achieve the balance between the two types of overhead.

Firstly, as a sub-method of parallel repair we discussed disjoint parallel method which protects each erasure by multiple disjoint repair sets. It has proven to be a quite restrictive approach both in terms of locality and the number of erasures that can be repaired in any case. As a more advanced method, cooperative repair uses a single repair set to repair all erasures at once. Its definition implies that any code with Hamming distance d can repair up to $d = e - 1$ erasures, which is shown to be maximal. Furthermore, the cooperative method reduces transmission overhead by efficiently using access to symbols. Hamming codes exploit this quality since their cooperative locality is given by $r = 3 \cdot 2^{n-k-2} - 2$. All other discussed methods do not reach this locality on the same (n, k) pairs. Therefore we can conclude that the combination of cooperative method and Hamming codes has an optimal trade-off between transmission overhead and these specific values of storage overhead. As a third discussed repair method, the sequential approach repairs erasures individually by using already repaired erasures. This has the advantage that other and possibly smaller repair sets can be accessed. Generalized parity codes use the sequential property to repair more erasure than with the disjoint parallel method. Moreover they use the smaller repair sets in the non-squared case to reach lower locality than cooperatively. This proves that the combination of this code and sequential method has a good transmission overhead for the (n, k) pairs on which they are defined, but it does not yet prove optimality.

In general, in the search of any optimal reparation process it is important to find combinations of a method and code which benefit from their qualities. The cooperative method applied to Hamming codes and the sequential method combined with generalized parity codes use these qualities to their advantage.

A remarkable observation from our findings is the poor performance of the disjoint parallel method. This can be explained because it is the most basic and restrictive sub-method of the parallel approach. As stated in Remark 4.6 there exist more advanced parallel approaches which are less confining on the used repair sets. It would be interesting to investigate these methods and incorporate the results in the comparison.

Secondly, a recently published contribution in [2] regarding an achievable upper bound on the information rate of codes with the sequential repair of an arbitrary number of erasures can be researched. Apart from this major contribution on the subject of Section 4.4, it contains much more information on the topics of this thesis and on other forms of data protection and erasure repair .

References

[1] Wentu Song, Kai Cai, Chau Yuen, *Senior Member, IEEE*, Kui Cai, Guangyue Han. "On Sequential Locally Repairable Codes". In: *IEEE Trans. Inf. Theory* 64.5 (May 2018), pp. 3513–3527.

[2] Balaji S.B. "Erasure Codes for Distributed Storage: Tight Bounds and Matching Constructions". In: *Electrical Communication Engineering, Indian Institute of Science Bangalore, M.Sc. Thesis* 257 (June 2018).

[3] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran. "Solution to the Network Challenges of Data Recovery in Erasure-coded Distributed Storage Systems: A Study on the Facebook Warehouse Cluster". In: *Proc. 5th USENIX Workshop on Hot Topics in Storage and File Systems, San Jose, CA, USA* (2013).

[4] Yury Izrailevsky, Director of Cloud Systems Infrastructure, Ariel Tseitlin, Director of Cloud Solutions. *The Netflix Simian Army.* URL: `https://medium.com/netflix-techblog/the-netflix-simian-army-16e57fbab116`. (accessed: 22.06.2018).

[5] Darrel R. Hankerson. *Coding Theory and Cryptography: The Essentials.* CRC Press, 2000. ISBN: 9780585421414.

[6] R. Hill. *A First Course in Coding Theory.* Clarendon Press, Oxford, 1986. ISBN: 0198538030.

[7] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin. "On the locality of codeword symbols". In: *IEEE Trans. Inf. Theory* 58.11 (Nov 2012), pp. 6925–6934.

[8] K.A.S. Abdel-Ghaffar and J.H. Weber. "Bounds for Cooperative Locality Using Generalized Hamming Weights". In: *Proceedings IEEE International Symposium on Information Theory (ISIT), Aachen, Germany* (June 2017), pp. 699–703.

[9] N. Prakash, V. Lalitha and P. Vijay Kumar. "Codes with Locality for Two Erasures". In: *Proceedings IEEE Int. Symp. Inf. Theory (ISIT), Honolulu, HI, USA,* (June/Jul, 2014), pp. 1962–1966.

[10] J. Bom. "Cooperative Locality of Shortened Hamming Codes". In: TU Delft, Applied Mathematics B.Sc. Thesis (June 2017).