Locally Repairable Linear Block Codes for Distributed Storage Systems

by

Mostafa Shahabinejad

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Communications

Department of Electrical and Computer Engineering

University of Alberta

Abstract

Distributed storage systems (DSSs) traditionally replicate data blocks to achieve storage reliability. Considering the rapid growth of data volume as well as costly maintenance of storage components in DSSs, the replication method is becoming unattractive because of its very large storage overhead. Recently, locally repairable codes (LRCs) have been proposed and used in practice e.g., in Facebook HDFS-RAID and Windows Azure storage. LRCs are attractive because they *i*) significantly decrease the storage overhead of the replication method and *ii*) considerably decrease communications traffic for data recovery compared to traditional coding methods—such as Reed-Solomon codes. An LRC can reconstruct any coded block by accessing a small number of other coded blocks. The minimum number of blocks required to reconstruct a missing block is defined as the block's locality. The maximum locality of all blocks is defined as code locality. Similarly, the average locality of a code is defined as the average locality of its blocks. The main focus of this dissertation is on studying and designing LRCs with low computational complexity and on LRCs with small average locality.

Because of immense size of modern energy-hungry DSSs, reducing the computational complexity of coding methods is of great importance. In that regard, binary LRCs are attractive because they eliminate the need for multiplication in operations such as encoding, decoding, and reconstruction. In the first part of this dissertation, we propose a class of binary LRCs. Using storage overhead and reliability of the code as design metrics, we show that some instances of our proposed binary codes are optimal, while others sacrifice the storage overhead marginally to gain a low coding complexity. Also, by analyzing mean-time to data-loss as a reliability metric, we verify that the reliability of our proposed binary LRC is more than sufficient from a practical point of view.

In the second part of the dissertation, we study average locality of LRCs. We derive lower bounds on average locality and design three classes of LRCs with minimum average locality. We also establish an achievable lower bound on the average locality of information blocks of LRCs, and design LRCs that achieve the obtained bound. Minimizing the average locality of information blocks is important as only information blocks are needed to be recovered during a temporal node unavailability, which accounts for 90% of all block recoveries triggered in DSSs.

Preface

The results presented in Section 3.2 were published in IEEE Communications Letters [1] and IEEE Transactions on Communications [2] in July 2014 and August 2016, respectively. Some of the results of Section 4.2 were published in IEEE Transactions on Communications in June 2017 [3]. The results of Section 4.2.3 were presented in the IEEE International Conference on Computing, Networking and Communications (ICNC 2017) [4], which won the best paper award of the conference. Finally, the results presented in Chapter 5 were presented in the IEEE International Symposium on Information Theory (ISIT 2017) [5].

To my beloved parents and wife...

Acknowledgements

First of all, I would like to thank my supervisors Dr. Masoud Ardakani and Dr. Majid Khabbazian, who continuously supported my research by their display motivation, patience and vast knowledge. Undoubtedly, accomplishing this research was impossible without their friendly guidance.

Besides my supervisors, I would like to thank the rest of my thesis committee: Dr. Shahram Yousefi, Dr. Chintha Tellambura, and Dr. Ivan Fair. I would like to specially thank Dr. Shahram Yousefi for his journey to Edmonton to take part in my defense. I also wish to thank Afshin Arefi who assisted me in writing the computer programs of this research with his valuable help and guidance.

Last, but not the least, I must express my deep gratitude to my lovely family and wife. Their endless love and ever-lasting support were the main encouragement for me to do my thesis. Thanks to all of my friends, for their support and help when I was new in Edmonton. They always make my life wonderful, so that I think I have a family here. Thank you for helping me to follow my dreams and believe in myself.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Cloud storage provides a reliable, flexible, and cost-effective way to store digital data. Using the cloud, one may avoid data loss (caused by device failures, fire, theft, etc.), access data anywhere/anytime, and easily share data. Google Drive, Dropbox, Apple iCloud, and OneDrive are among the widely used cloud storage systems, and Amazon, Google, and Microsoft are among hosting companies providing a variety of cloud storage services.

A cloud storage system is implemented in the form of a distributed storage system (DSS). A DSS is a computer network, typically built using inexpensive commodity hardware, where information is stored on several storage devices called data nodes (DNs). A DN may become temporarily unavailable or permanently fail rendering data unavailable or lost[6]. When a DN is temporally unavailable, for example because of electricity/network resource outage, the DSS is not able to respond to a client data request immediately due to data unavailability. In the case of failure, for example due to a physical damage to the hard disk drive, the DSS may lose a client's data.

A typical approach to achieving availability and reliability in a DSS is to store a number of (typically, three) replicas of data in distinct DNs. The 3-

replication method, in which every original data block along with two replicas of it are stored in distinct storage nodes, has been traditionally used in DSSs, for example, in Facebook HDFS-RAID[6]. Replication is the most efficient solution in terms of bandwidth and the number of I/O operations required in the reconstruction process. However, replication results in a high storage overhead (e.g., %200 storage overhead in the case of the widely used 3-replication scheme). Considering the tremendous cost of construction and maintenance of DSSs as well as the fast growth of the world's data, employing new storage methods with less storage overhead is of interest.

Erasure coding is considered as an attractive solution for decreasing the storage overhead in DSS, e.g., see [8, 6, 9]. An (n, k) erasure code with minimum distance d maps k information symbols to n encoded symbols such that any $n - d + 1$ out of the n encoded symbols suffice to recover all the k information symbols, where $d \leq n - k + 1$. In other words, an (n, k, d) erasure code can tolerate any $d - 1$ encoded symbol failures. Hence, it is possible to obtain the desirable level of reliability by tuning d. If $d = n - k + 1$, the erasure code is called maximum-distance separable (MDS) code as it achieves the maximum possible minimum distance for a given n and k. The code rate, and the storage overhead are defined as $\frac{k}{n}$, and $\frac{n}{k} - 1$, respectively.

Erasure codes are being used in large-scale storage systems because they significantly reduce the storage overhead of the 3-replication method without sacrificing its reliability. For example, in Facebook HDFS-RIAD [6], Google File System [10], and Window Azure Storage [7], erasure codes with correspondingly storage overhead of %60, %50, and %33 are used. These codes provide at least the same level of reliability as that achieved in the 3-replication method with the storage overhead of 200%.

This significant storage overhead reduction can come at the price of high bandwidth, I/O, and computation for reconstructing unavailable/lost data blocks, if conventional erasure codes are used. For example, when an (n, k)

Reed-Solomon code is used, reconstruction of a missing block will require reading and downloading k blocks from k active nodes. This amount of read and download can lead to the underlying network saturation in the DSS [6]. Because of such high reconstruction overheads of conventional erasure codes, researchers have been recently working on designing new erasure codes carefully crafted for DSSs. Three main metrics considered in designing codes for DSSs are:

i) *Reconstruction bandwidth:* Reconstruction bandwidth is one of the targets to be minimized in a reconstruction process. The theoretical bound between the required reconstruction bandwidth and storage overhead is established in [11]. Codes that achieve this bound are called regenerating codes (RGCs). In RGCs, each block of data is partitioned into a few number of sub-blocks. In the case that a data node containing a packet fails, some sub-blocks of active nodes are downloaded such that the reconstruction bandwidth is minimized for a given storage overhead [11]. To see examples of RGCs, please refer to [12, 13, 14, 15] and references therein.

The main advantage of RGCs is that they have the minimum reconstruction bandwidth for a given storage overhead. This advantage is achieved by connecting to more than k DNs for a reconstruction process, which is not desirable from a practical point of view [16]. Therefore, a drawback of regenerating codes is their overhead associated with accessing a large number of DNs during a reconstruction process.

ii) *Disk I/O:* The disk I/O overhead (the required number of reads during a block recovery) can be a bottleneck during recovering a missing block of data. Another design goal therefore is disk I/O reduction (e.g. see [17, 18, 19] and references therein). In [17], a framework is introduced which is added to MDS codes to decrease their corresponding disk I/O. In [18], rotated RS codes are introduced to decrease disk I/O. In [19], a class of RGCs with small disk I/O has been introduced.

iii) *Code locality:* The locality of an (n, k) locally reparable code (LRC), de-

noted r, is defined as the maximum number of coded symbols required to reconstruct any missing coded symbol. Since reducing r decreases the number of storage nodes to be accessed during recovering a missing data block, it results in an agile recovery method [20, 16]. In the case of repairing permanent failures or retiring defective nodes, LRCs with small r are desired as they require less costly network bandwidth and disk I/Os. In the case of temporal unavailability, reducing r can significantly improve data availability, and expedite the process of distributing data among various data centers [6]. Hence, reducing the locality r of erasure codes is important.

Although RGCs have a smaller storage overhead in comparison with LRCs for a given d, in practice, LRCs are preferred to RGCs because of high reconstruction locality of RGCs [16]. In real-world DSSs, LRCs have been used [6, 7]. The focus of this dissertation is twofold: (i) binary LRCs (BLRCs) which are an important class of LRCs because of their low computational complexity; and (ii) LRCs with small average locality.

1.2 Organization and Overview of the dissertation

In the following chapter, we review the needed preliminary about erasure codes and LRCs.

Our results are categorized in two general groups. In the first group, we design LRCs in binary finite field (or finite fields with small order) to reduce the computational complexity of encoding, decoding and reconstruction. In the second group, we study average locality, prove bounds, and design codes that achieve our bounds. In the following, we discuss the contributions of this dissertation in more details.

In Chapter 3, we first introduce an $(n, k, d, r) = (15, 10, 4, 6)$ binary LRC. The proposed BLRC has a minimum distance of four, and provide a better

reliability than the widely used 3-replication method. Since our code is binary, encoding, decoding and repairing do not require finite field multiplication, resulting in significant computational saving. We prove that, among all $(n, k) = (15, 10)$ binary codes with minimum distance of four, our code has the minimum locality, thus imposes minimum repair traffic.

In Chapter 3, we also present a class of efficient binary LRCs with minimum distance of four, a distance shown by our analysis to provide sufficient reliability for a wide range of practical code and system parameters. We show that our proposed binary LRCs are optimal for locality $r \in \{1, 3\}$. For larger r, we show that the rate of our proposed binary codes is near optimal (with a rate gap of $\mathcal{O}(\log r)$).

In Chapter 4, we present our results on LRCs with small average locality. The average locality of LRCs is directly translated to the costly repair bandwidth of DSSs. Analyzing code's average locality ($\bar{r}$) is more involving than analyzing locality, which has been the focus of research in the literature. In Section 4.2, we use a novel approach to derive the first lower bound on average locality, $\bar{r}$, of erasure codes with arbitrary parameters. We also present the construction of $\bar{r}$-optimal LRCs for a broad range of codes' parameters offering improvement on $\bar{r}$ over the existing LRCs. Comparing with the LRC used in Facebook HDFS-RAID, a sample of our proposed $\bar{r}$-optimal LRCs improves the average locality by 22.5% without sacrificing the rate or minimum distance of the code.

In Chapter 5, we obtain an achievable lower bound on the average locality of the information blocks ($\bar{r}_{inf}$) and design a class of LRCs which always achieve the obtained bound. The average locality of the information blocks in DSSs is translated to the average data required to reconstruct an unavailable information block. The importance of the reconstruction cost as well as the frequent unavailability of information blocks in the real-world DSSs make designing codes with the minimum $\bar{r}_{inf}$ of interest.

The conclusion of this dissertation along with the future research works are presented in the last Chapter. Proofs of the theorems are presented in the appendix to improve the readability.

Notations: We denote matrices and vectors by capital boldface letters and boldface letters, respectively. Notations $(\cdot)^{-1}$ and $(\cdot)^T$ represent matrix inverse and matrix transpose operations, respectively. $\mathbf{I}_a$ represents an identity matrix of size a and the matrix transpose operation, respectively. $\mathbf{A}(\mathcal{I})$ is a sub-matrix of $\mathbf{A}$ obtained by maintaining columns of $\mathbf{A}$ indexed by set $\mathcal{I}$. $\mathbf{v}(i)$ represents the i-th element of vector $\mathbf{v}$. $\mathbf{0}_a$ represents a row vector of all zeros of size a. Similarly, $\mathbf{1}_a$ represents a row vector of all ones of size a. $wt(\mathbf{v})$ stands for Hamming weight of the vector $\mathbf{v}$.

For integers a and b, if $b \geq a$, $[a, b] = \{a, a+1, ..., b\}$ otherwise, $[a, b] = \{\}$; also, $[a] = \{1, \cdots, a\}$. $\overline{\mathcal{A}}$ and $|\mathcal{A}|$ stand for the complement and cardinality of set $\mathcal{A}$, respectively. For two sets $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{A} \backslash \mathcal{B}$ stands for the relative complement of $\mathcal{B}$ in $\mathcal{A}$, i.e., $\mathcal{A} \backslash \mathcal{B} = \{a \in \mathcal{A} | a \notin \mathcal{B}\}$. $\mathcal{P}(\mathcal{A})$ represent the power set of set $\mathcal{A}$. For set $\mathcal{B}$ and integer b, $\mathit{comb}(\mathcal{B}, b)$ represents the set of all combination of $\mathcal{B}$ taken b at a time. Therefore, $|\mathit{comb}(\mathcal{B}, b)| = \binom{|\mathcal{B}|}{b}$.

For integers a and b, $a \mod b$ represents the remainder of devision a by b. $\otimes$ stands for Kronecker product. $\mathbb{F}_q$ stands for a finite field with cardinality q. Finally, $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ represent the floor and ceiling operators, respectively.

Chapter 2

Background

This section contains the required definitions and assumptions. Also, this section reviews needed background on data storage, DSSs, linear block codes and their applications in DSSs.

2.1 Data Storage

Here, we briefly review popular types of data storage, describe hard disk drive (HDD), solid state drive (SSD), and redundant array of independent disks (RAID). Furthermore, briefly describe the concept of DSSs, which are used to store massive amount of data

2.1.1 HDD, SSD, and RAID

In this section, we first give a high-level description of HDDs and SSD as two commonly used storage devices. Then, we explain the popular data storage techniques, namely data striping and data mirroring. Finally, we briefly review different types of RAIDs as important parts of data storage world.

HDD, head, and track: An HDD is a storage component constructed out of a set of stacked disks. In a HDD, data is stored in concentric circles, called

tracks, on each disk. Mechanical arms with two heads, each on one side of a disk, perform the read/write operations.

SSD: An SSD is a storage component constructed out of microchips. Unlike HDD, an SSD has no moving parts. In a SSD, an embedded controller processor does the read/write operations.

In comparison with HDDs, SSDs have less power consumption and failure rate; have a higher read/write speed; and are safe from magnetism effects.

Data striping: The process of partitioning a stripe of data into blocks and storing data blocks on various HDDs is called data striping. Data striping can speed up throughput (I/O operations per second) by a factor of the number of HDDs. The downside of using data striping is the low resiliency. If even one disk fails, all of data becomes unavailable.

Example 2.1. *Assume that disk striping is performed on two HDDs and that each HDD runs at 180 IOPS (Input/Output Operations Per Second). Then, up to 360 IOPS can be achieved using data striping.*

Data mirroring (replication): The process of replicating data to at least two disks is called data mirroring. Data mirroring improves data availability and fault tolerance.

RAID: RAID uses an array of HDDs to improve I/O operations and/or data fault tolerance. Following is a list of different types of RAID currently in use with a brief explanation.

- RAID 0 stores block-level data stripes uniformly on two or more disks with no redundancy. An n-drive RAID 0 can improve I/O performance by a factor of n. RAID 0 is used when high I/O performance is required.

- RAID 1 uses disk mirroring to improve the read performance. A read operation can be performed by any disk. RAID 1 is used when high reliability and read performance is required.

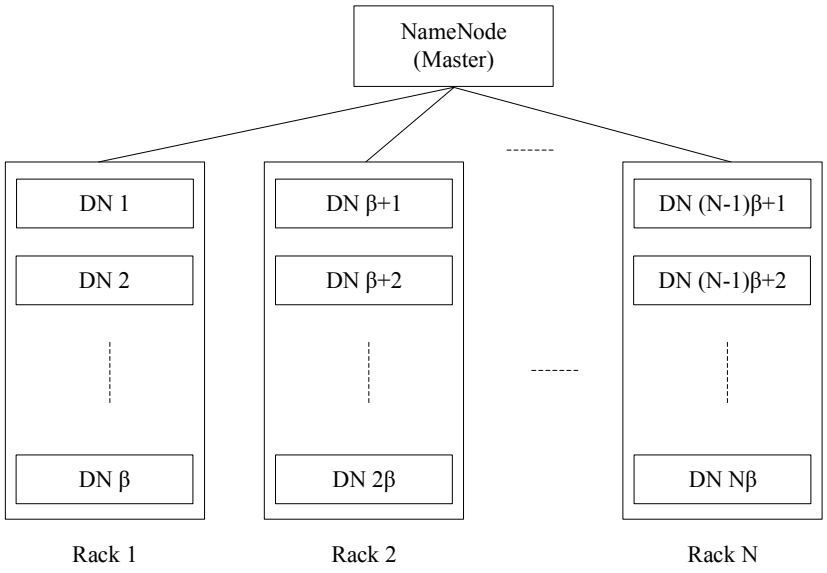


Figure 2.1: A simplified structure of HDFS with N racks and $N\beta$ DNs.

- RAID 5 uses block-level striping of data and stores the parity checksum across various disks. RAID 5 improves read data transaction and it can tolerate a single disk failure in the array.

- RAID 6 uses block-level striping of data and stores two parity blocks across various disks. RAID 6 can tolerate up to two disk failures in the array.

- RAID 10 is a hybrid configuration of RAID 1 and RAID 0. In other words, it uses both data mirroring and striping which improve read data transaction and I/O performance, respectively.

2.1.2 Hadoop Distributed File System (HDFS)

HDFS has a master/slave configuration used to store large-scale data. Fig. 2.1 depicts a simplified configuration of HDFS with one master node and $N\beta$ data nodes (DNs), where N is the total number of racks, and β is the number of DNs in each rack. Each rack has its own network and power cable.

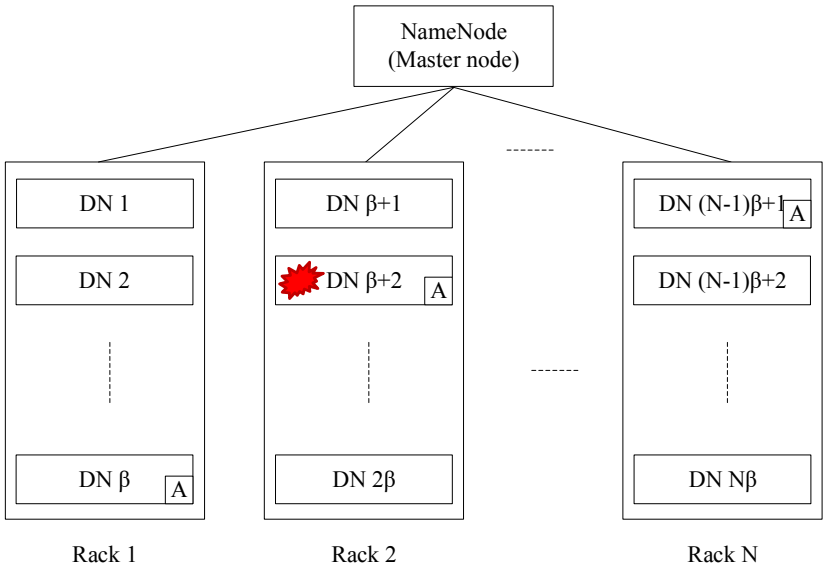


Figure 2.2: DN failure and redundancy through the three-replication method.

Example 2.2. *In a typical Facebook HDFS-RAID,* $N = 100$*,* $\beta = 30$*, and each DN has* 20 *TB storage capacity [6]. Thus, the whole HDFS has the storage capacity equal to* $100 \times 30 \times 20$ *TB* $= 60$ *PB.*

The name node is an administer computer that contain metadata, e.g., file directory of the stored data. To provide reliability, some backups of the name node are provided. Data resides in DNs. Each DN sends heartbeat reports to the name node frequently. DNs can communicate together, for example, in the case of replicating data.

Due to hardware failures, the data stored in a DN may be lost. In order to make HDFS reliable, redundant data is stored. A very common method to provide redundancy is through data replication. For example, in Fig. 2.2, data block A is replicated in three DNs β, $\beta + 2$, and $(N - 1)\beta + 1$.

Now, assume that $(\beta+2)$-th DN fails as shown in Fig. 2.2. Then, block A can be reconstructed by downloading data from either β-th DN or $(N - 1)\beta + 1$-th DN.

The three-replication method described above results in a very high storage

overhead of 200%. Very recently, erasure coding is proposed to store data to decrease the storage overhead. In Section 2.3, we describe the recent erasure coding methods proposed to alleviate the problem of large storage overhead of the replication.

2.2 Finite Field $\mathbb{F}_q$

Definition 2.1. *Abelian group: Let a, b and c be three arbitrary elements in a set $\mathcal{A}$. The set $\mathcal{A}$ and a binary operation $(\cdot)$ constitute an abelian group, denoted $(\mathcal{A}, \cdot)$, if the following five axioms are satisfied.*

(i) Closure: $a \cdot b \in \mathcal{A}$,

(ii) Associativity: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$,

(iii) Identity element: There exists an element $e \in \mathcal{A}$, such that $e \cdot a = a \cdot e = a$,

(iv) Inverse element: For each $a \in \mathcal{A}$, there exists an element $a^{-1} \in \mathcal{A}$ such that $a \cdot a^{-1} = a^{-1} \cdot a = e$ (e is the identity element), and

(v) Commutativity: $a \cdot b = b \cdot a$.

Example 2.3. *The set of integers $\mathbb{Z}$ with the addition operation "+" constitute the abelian group $(\mathbb{Z}, +)$ with the identity element $e = 0$.*

Definition 2.2. *Field: A set $\mathbb{F}$ with an addition operation "+" and a multiplication operation "$*$" represents a finite field, denoted $(\mathbb{F}, +, *)$, if*

(i) $(\mathbb{F}, +)$ is an abelian group with additive identity "0",

(ii) $(\mathbb{F} \setminus \{0\}, *)$ is an abelian group with multiplicative identity "1",

(iii) For all $a, b, c \in \mathbb{F}$, $(a + b) * c = a * c + b * c$.

If the set $\mathbb{F}$ has a finite q number of elements, then the field is called a finite field, denoted $(\mathbb{F}_q, +, *)$ or simply $\mathbb{F}_q$, where q is called the order of field.

Symbol, block: We call each element of the finite field a symbol. Also, we call a vector of symbols a block.

Example 2.4. *Finite field* $(\mathbb{F}_{2^2}, +, *)$ *has four elements* $\mathbb{F}_{2^2} = \{0, 1, 2, 3\}$*, where addition and multiplication are defined based on the following tables:*

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Remark 2.1. *Each element of a finite field* $\mathbb{F}_{2^m}$ *can be represented in the form of* m *bits. Hence, addition and multiplication can be performed at the bit level. The addition is equivalent to XOR and multiplication can be done using a lookup table.*

Example 2.5. *For* $\mathbb{F}_{2^2}$ *associated with Example 2.4, each element can be represented in the form of* $m = 2$ *bits as:* $0 = [0, 0], 1 = [0, 1], 2 = [1, 0]$*, and* $3 = [1, 1]$*. Addition and multiplication can be done based on tables presented in Example 2.4. Observe that the addition is equivalent to XOR. For example,* $2 + 3 = [1, 0] \oplus [1, 1] = [1 \oplus 1, 0 \oplus 1] = [0, 1] = 1$.

2.3 Linear Block Codes

Erasure codes provide protection against data block erasures by introducing redundant data blocks. In an erasure code, redundant data blocks are constructed from the information blocks using a coding algorithm. In the case of block erasures (i.e., when some blocks are missing/unavailable), available data blocks can be used to construct and retrieve the erased ones. More particularly, an (n, k) erasure code transforms k information data blocks into n

encoded data blocks such that up to $d-1$ erasures can be recovered, where d is the minimum distance of the code.

The replication method, which is an $(n, 1)$ erasure code, is the most straightforward erasure coding method. In an n-replication method, a data block is replicated $n-1$ times. Therefore, there are n replicas of data blocks in the storage system. Hence, in the case of up to $n-1$ block erasures, the original data block can be retrieved.

If the transformation from k information blocks to n encoded blocks involves only linear operations over the information blocks, the erasure code is called a linear block code. Also, if the n code blocks include the k information blocks, the code is called systematic.

Systematic linear block codes are a class of erasure codes used in storage systems. In the following, we explain this class of codes in more details.

2.3.1 Systematic linear block codes

An (n, k) systematic linear block code $\mathcal{C}$ in $\mathbb{F}_q$ takes k information symbols in $\mathbb{F}_q$ and converts them to $n > k$ coded symbols in $\mathbb{F}_q$ using the linear operations $\mathbf{y} = \mathbf{xG}$, where $\mathbf{x} = [x_1, x_2, \cdots, x_k] \in \mathbb{F}_q^{1\times k}$ and $\mathbf{y} = [y_1, y_2, \cdots, y_n] \in \mathbb{F}_q^{1\times n}$ represent the information and coded vectors with symbols x_i's and y_i's, respectively; and $\mathbf{G}$ is the generator matrix with the following form: $\mathbf{G} = [\mathbf{I}_k, \mathbf{P}] \in \mathbb{F}_q^{k\times n}$, where $\mathbf{P} \in \mathbb{F}_q^{k\times m}$ and $m = n-k$. The parity check matrix $\mathbf{H}$ of the systematic code $\mathcal{C}$ can be formed as $\mathbf{H} = [-\mathbf{P}^T, \mathbf{I}_m] \in \mathbb{F}_q^{m\times n}$. The coded symbol vector $\mathbf{y}$ of a linear block code satisfies $\mathbf{yH}^T = \mathbf{0}_m$.

2.3.2 Hamming weight and Hamming distance

The number of non-zero elements of vector $\mathbf{a}$ is called the Hamming weight of $\mathbf{a}$, denoted $wt(\mathbf{a})$. For two vectors $\mathbf{a}$ and $\mathbf{b}$, $wt(\mathbf{a}-\mathbf{b})$ is called the Hamming distance between $\mathbf{a}$ and $\mathbf{b}$, denoted $d(\mathbf{a}, \mathbf{b})$.

2.3.3 Minimum distance of code (d)

The minimum distance of a code is defined as the minimum number of differences between any two codewords. In other words, $d = \min\{d(\mathbf{u}, \mathbf{v})\}$, for any arbitrary codewords $\mathbf{u}$ and $\mathbf{v}$ of code $\mathcal{C}$. For an (n, k) code with minimum distance d, any $d - 1$ missing symbols can be recovered.

2.3.4 Maximum distance separable (MDS) codes

By the Singleton bound associated with an (n, k, d) linear block code, we have $d \leq n - k + 1$ [21]. Linear block codes that achieve this bound on d are called MDS. For MDS codes, any $n-d+1 = k$ encoded symbols suffice to reconstruct the k information symbols. Reed-Solomon (RS) codes are a well-known class of MDS codes [22].

2.3.5 n-replication method

In n-replication coding method, n replicas of data are stored in n distinct storage nodes. Hence, the storage overhead of n-replication method is $n - 1$. The three-replication method with the storage overhead 200% has been widely used in practice due to its simple, reliable implementation [6, 7].

2.3.6 Tanner graph

An $\mathbf{H}$-based Tanner graph of an (n, k) linear block code, denoted $\mathcal{T}$, is a bipartite graph with two sets of vertices: a set of n variable nodes (VNs) y_1 to y_n, denoted $\mathcal{Y}$, and a set of $n - k$ check nodes (CNs), denoted $\mathcal{P}$. The i-th variable node for $i \in [1, n]$ is adjacent with the j-th CN for $j \in [1, n - k]$ iff $\mathbf{h}_j(i) \neq 0$, where $\mathbf{h}_j$ represents the j-th row of the parity check $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$. Therefore, variable nodes linked to a CN are linearly dependent. In the case of binary codes, XOR of variable nodes linked to a CN is zero.

Remark 2.2. *A code does not have a unique Tanner graph representation.*

Example 2.6. *A systematic* $(n, k, d) = (4, 2, 3)$ *linear block code: Consider information symbol vector* $\mathbf{x} = [x_1, x_2] \in \mathbb{F}_{2^2}^{1\times 2}$ *with* $k = 2$ *information symbols* x_1 *and* x_2 *over field* $\mathbb{F}_{2^2}$*. Then, by using a* $(4, 2)$ *linear block code with the generator matrix* $\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix} \in \mathbb{F}_{2^2}^{2\times 4}$*,* $\mathbf{x}$ *is encoded as follows.*

$$\mathbf{y} = \mathbf{x} \times \mathbf{G} = [x_1, x_2, x_1 + x_2, x_1 + 2x_2],$$

where $\mathbf{y} = [y_1, y_2, y_3, y_4] = [x_1, x_2, x_1+x_2, x_1+2x_2] \in \mathbb{F}_{2^2}^{1\times 4}$ *is the encoded symbol vector. Observe that every* $n-(d-1) = 2$ *encoded symbol* y_i*'s suffice to recover the information symbols* x_1 *and* x_2*. Now, assume that information bit* 0111 *has to be encoded using the above* $(4, 2)$ *linear block code. First,* 0111 *is partitioned into two fractions* 01 *and* 11*. Then, as seen in Examples 2.4 and 2.5,* $x_1 = 1$ *and* $x_2 = 3$ *and*

$$[y_1, y_2, y_3, y_4] = [1, 3, 1 + 3, 1 + (2 * 3)] = [1, 3, 2, 0] = [01, 11, 10, 00]$$

The codeword associated with $x_1 = 1$ *and* $x_2 = 3$ *is* $[1, 3, 2, 0]$*. The above* $(4, 2)$ *linear block code is systematic because the original information symbols* x_1 *and* x_2 *are embedded in the encoded symbols.*

2.4 Erasure Coding in Distributed Storage Systems

Traditionally, erasure codes have been proposed for the binary erasure channel in which a bit is either received perfectly or erased. A similar situation exists in distributed storage systems (DSSs), where a previously recorded data is either available perfectly, or erased (due to, for example, a disk failure). This motivates the application of erasure codes in DSSs [23].

In DSSs, systematic erasure codes—in which information symbols are embedded in the encoded symbols—are preferred to non-systematic ones. This is because one can access the original data (information symbols) directly without performing a decoding process. We first explain how erasure codes are used in real-world DSSs.

Assume that a stripe of data of size L_B bits is to be stored in a DSS. The stripe is first broken into k data blocks each with size $l_B = \frac{L_B}{k}$ bits. Hence, each block has $\frac{l_B}{m}$ symbols in $\mathbb{F}_{2^m}$ (for simplicity, let $k \mid L_B$ and $m \mid l_B$). We denote by $x_{i,j}$ the i-th symbol of the j-th block. In linear block erasure codes, the encoded vector $\mathbf{y}_i$ is then obtained from $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,k}) \in \mathbb{F}_{2^m}^{1\times k}$ by multiplying $\mathbf{x}_i$ by a generator matrix $\mathbf{G} \in \mathbb{F}_{2^m}^{k\times n}$, i.e. $\mathbf{y}_i = (y_{i,1}, y_{i,2}, \cdots, y_{i,n}) = \mathbf{x}_i\mathbf{G}$. The coded vectors $\mathbf{y}_i$, $i \in [1, \frac{l_B}{m}]$, are then stacked to constitute $\mathbf{Y}$ whose columns are n coded blocks stored in n distinct storage nodes. From now on, we assume that $\frac{l_B}{m} = 1$, and use symbol and block interchangeably.

Advantages of using erasure codes in a DSS are:

(i) Low storage overhead: The storage overheads of the n-replication and an (n, k, d) erasure code are $n - 1$ and $\frac{n-k}{k}$, respectively. Hence, by adjusting k and n, it is possible to obtain a desirable level of storage overhead. Google File System (GFS), Windows Azure Storage (WAS), and Facebook HDFS-RAID use erasure codes with the storage overheads of 50%, 33%, and 60%, respectively.

(ii) Desirable level of reliability by adjusting d: In an (n, k, d) erasure code, each coded symbol can be computed from any $n - (d + 1)$ other coded symbols. In other words, an (n, k, d) erasure code can tolerate any $d - 1$ symbol erasures. Hence, by changing the value of d, it is possible to adjust the storage system reliability. GFS, WAS, and Facebook HDFS-RAID use erasure codes with the minimum distance 4, 4, and 5, respectively.

When reconstruction of a single block of data (the dominant reconstruction

scenario in practice) is considered, erasure codes can pose several challenges. For example, when an MDS code is used, reconstructing a block requires reading and downloading k blocks from k active nodes. Considering the limited reconstruction bandwidth resource as well as the frequent DNs unavailability and failure, this amount of read and download may lead to network saturation in DSSs[6]. Comparing with the conventional replication methods, disadvantages of using erasure codes in a DSS are: (i) high reconstruction bandwidth, (i.e., the amount of data downloaded from active DNs to reconstruct a single block) (ii) high disk I/O (i.e., the amount of data read from active DNs to reconstruct a single block), and (iii) high reconstruction locality (i.e., the number of active nodes from which data is downloaded to reconstruct a single block). To address these disadvantages, new classes of erasure codes have been proposed in the literature. These classes are: (i) Regenerating codes (RGCs), (ii) Codes with the improved disk I/O, and (iii) Locally repairable codes (LRCs).

RGCs have been proposed to alleviate the problem of the high reconstruction bandwidth of traditional erasure codes. The reconstruction bandwidth-storage trade-off associated with erasure codes was studied and a bound was established in [11]. Erasure codes that satisfy reconstruction bandwidth-storage trade-off bound derived in [11] are called RGCs. In other words, RGCs result in the minimum reconstruction bandwidth for a given storage overhead.

Codes whose aim is to alleviate the problem of the high disk I/O of erasure codes have been recently proposed [19, 17, 24]. However, bounds on the I/O-storage trade-off are yet to be found [11].

LRCs have been proposed to alleviate the problem of the high reconstruction locality. The trade-off between the code locality and its minimum distance was studied, and a bound was established in [16, 25]. Erasure codes that satisfy this bound are called optimal LRCs.

In the following, we first review RS codes due to their important role in theory and practice. Then, we briefly review RGCs, LRCs and the codes with

improved disk I/O.

2.5 Reed-Solomon (RS) Codes

RS codes are a class of MDS linear block codes proposed in [22]. The parity check matrix of an (n, k) RS code is $\mathbf{H}_{RS} = [\alpha^{(i-1)(j-1)}] \in \mathbb{F}_q^{(n-k)\times n}$, where $i \in [1, n-k]$, $j \in [1, n]$, and $\alpha \in \mathbb{F}_q$ is a primitive element of $\mathbb{F}_q$. Since any $(n-k)\times(n-k)$ sub-matrix of $\mathbf{H}_{RS}$ is full-rank, every $d-1 = n-k$ columns of $\mathbf{H}_{RS}$ are independent. Hence, an RS code is MDS with the minimum distance $d = n-k+1$. The systematic form of $\mathbf{H}_{RS}$ can be obtained by multiplying the inverse of an $(n-k)\times(n-k)$ sub-matrix of $\mathbf{H}_{RS}$ by $\mathbf{H}_{RS}$ from the left side.

Example 2.7. *In Google File System, an $(n, k, d) = (9, 6, 4)$ RS code with the storage overhead (9-6)/6=50% has been used since 2011 [10]. The $(9, 6)$-RS code is MDS with $d = n - k + 1 = 4$. Therefore, the code can handle any $d-1 = 3$ block failures. Also, any information block can be reconstructed using any six other blocks.*

Example 2.8. *In Facebook HDFS-RAID, an $(n, k, d) = (14, 10, 5)$ RS code with the storage overhead (14-10)/10=40% was used in 2012 [6]. The $(14, 10)$-RS code is MDS with $d = n - k + 1 = 5$. When one block is missing, ten other blocks are downloaded from ten active DNs and the reconstruction process is performed.*

The main disadvantages of RS codes are their high reconstruction cost (also known as reconstruction bandwidth) and high locality. In order to reconstruct a block of a (n, k) RS code, k data blocks have to be downloaded from k active DNs. In other words, the reconstruction cost is k. This results in a high data communication between DNs when a reconstruction process is performed, which can be very costly [6]. Furthermore, an RS code requires accessing k DNs in order to reconstruct a data block. This is not desirable in real-world

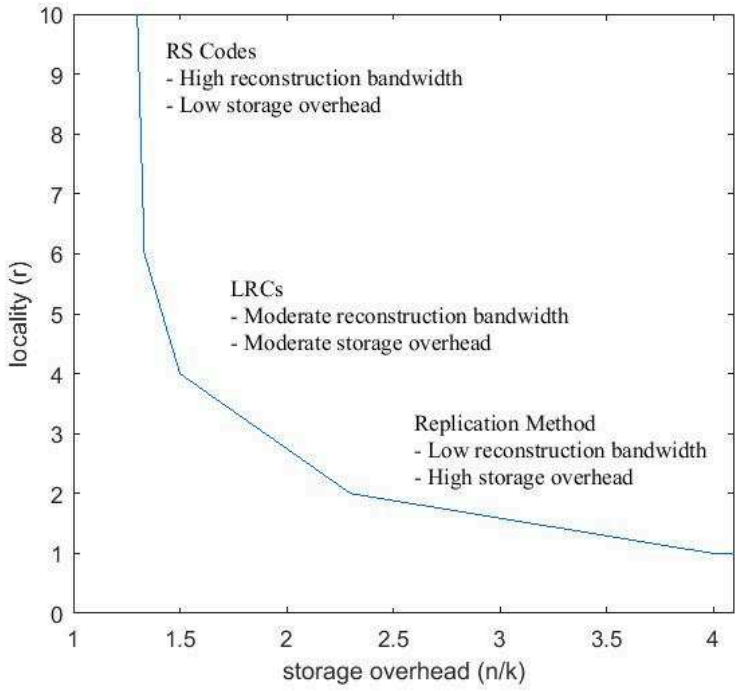


Figure 2.3: Storage overhead-locality trade-off for different coding methods with $d = 4$.

DSSs [16]. That is why new coding methods have been considered in DSSs such as Google File System, Windows Azure Storage, and Facebook HDFS.

2.6 Locally repairable codes (LRCs)

LRCs have smaller code locality than RS codes. The smaller code locality reduces the bandwidth and I/O required to reconstruct a missing block. LRCs can be considered as coding methods with moderate reconstruction bandwidth and moderate storage overhead (Fig. 2.3). LRCs have been recently used in practice. In Facebook HDFS-RAID, a $(n, k, r) = (16, 10, 5)$ LRC with $d = 5$ is used [6]. Also, in Windows Azure Storage a $(n, k, r) = (16, 12, 6)$ LRC with $d = 4$ is used [7]. In the following, we present definitions required in discussing our results on LRCs.

Definition 2.3. *(Symbol locality, code locality, average locality). Locality of*

i-th coded symbol of a code $\mathcal{C}$ is denoted by $\mathsf{Loc}(y_i)$, and is defined as the minimum number of other coded symbols required to reconstruct y_i. In other words, $\mathsf{Loc}(y_i)$ is the size of the smallest set $\mathcal{I}_i \subset [1,n]\backslash\{i\}$ that satisfies

$$\sum_{l \in \mathcal{A}_i} \alpha_l y_l = 0 \tag{2.1}$$

for a fixed set of coefficients $\alpha_l \in \mathbb{F}_q \setminus \{0\}$ and every codeword $(y_1, \ldots, y_n) \in \mathcal{C}$, where $\mathcal{A}_i = \{i\} \cup \mathcal{I}_i$. The average and maximum locality of the code are defined as $\overline{r} = \frac{\sum_{i=1}^{n} \mathsf{Loc}(y_i)}{n}$ and $r = \max_{i \in [1,n]} \{\mathsf{Loc}(y_i)\}$, respectively. Also, the average locality of the information symbols in a systematic code is defined as $\overline{r}_{inf} = \frac{\sum_{i=1}^{k} \mathsf{Loc}(y_i)}{k}$, where $y_i = x_i$ is the ith information symbol for $i \in [1,k]$.

Definition 2.4. *(Optimal LRCs). The minimum distance d of an (n,k,r) LRC is bounded as [25, 16]*

$$d \leq n - k + 1 - \left(\left\lceil \frac{k}{r} \right\rceil - 1 \right) = n - k - \left\lceil \frac{k}{r} \right\rceil + 2. \tag{2.2}$$

We say an LRC is d-optimal *if it satisfies (2.2) with equality for given (n,k,r). Similarly, we say an LRC is* r-optimal *if r is the smallest integer satisfying (2.2) for given (n,k,d). An LRC is optimal*[1] *if it is both d-optimal and r-optimal. The code rate of LRC with locality r is bounded as $\frac{k}{n} \leq \frac{r}{r+1}$ [26].*

Some existing results on LRCs: Several d-optimal LRCs have been proposed when $r+1$ divides n, i.e., $(r+1) \mid n$ (e.g. see [27, 28, 29, 30, 31, 26, 32, 33, 34]). The alphabet size q for the proposed codes in [28] and [30] is an exponential function of k and $\frac{nr}{r+1}$ respectively, which is improved to q being equal to or slightly larger than n in [26]. The choice of q can affect the coding complexity.

[1]Existing literature usually refers to d-optimal codes as optimal codes. In this dissertation, we distinguish between d-optimal and r-optimal codes and reserve the term "optimal" for codes that are both d-optimal and r-optimal.

In [35, 36] binary LRCs are proposed. In [37], a bound on k is obtained in terms of n, k, and r as well as finite field order q. In [38], LRCs for small values of r have been proposed some of which achieve the bound in [37].

In [39, 40], considering a linear block code with length n, dimension k, and locality profile $\mathbf{n} = \{n_1, \cdots, n_r\}$, where n_i is the number of encoded symbols with locality i, an upper bound on the code's minimum distance (d) has been found. In [41], the locality constant is relaxed by allowing some symbols to have a locality of $r + 1$ instead of r in order to gain flexibility in code construction. In [42], the problem of secure vector-LRCs is considered taking the maximum locality of the code into account. In [43], an upper bound on d of regenerating LRCs is established taking the maximum locality of the code into account and codes that achieve the obtained bound are designed.

In some other schemes of LRCs, a missing symbol can be reconstructed by accessing any group from a set of multiple disjoint groups of other symbols, e.g. see [31, 44, 45, 46, 47, 48, 49] and references therein.

2.6.1 Tanner Graph Representation of LRCs

As mentioned earlier, a code does not have a unique Tanner graph representation. Here, we define a representation of a Tanner graph, called locality Tanner graph, which reflects the locality of all the encoded symbols. First, we need to define locality-defining set.

Definition 2.5. *(Locality-defining set). Let $\mathcal{Y}_e \subseteq \mathcal{Y}$ be a set of variable nodes. A set of CNs $\mathcal{P}_s$ is called a locality-defining set for $\mathcal{Y}_e$ if every variable node $y_i \in \mathcal{Y}_e$ with locality $\mathsf{Loc}(y_i)$ is adjacent to at least one CN in $\mathcal{P}_s$ with degree $\mathsf{Loc}(y_i) + 1$. Let $\Phi(\mathcal{Y}_e)$ be a function from a set of variable nodes $\mathcal{Y}_e$ to a set of CNs that returns a minimal locality-defining set for $\mathcal{Y}_e$. That is, $\Phi(\mathcal{Y}_e)$ is a locality-defining set for $\mathcal{Y}_e$, and there is no proper subset of $\Phi(\mathcal{Y}_e)$ which is a locality-defining set for $\mathcal{Y}_e$.*

Definition 2.6. *(Locality Tanner graph). An LRC can be determined by different parity check matrices, which can yield different Tanner graphs. Among such Tanner graphs, we call a Tanner graph locality Tanner graph if the set of CNs of the graph includes $\Phi(\mathcal{Y})$, which is a locality-defining set for the set of all variable nodes.*

Lemma 2.1. *(Lemma 4.1). Every LRC has a locality Tanner graph.*

By fixing the locality Tanner graph of an LRC, the zeros of its corresponding parity check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ are determined. For the non-zero element of $\mathbf{H}$, we assume that they are randomly picked from a sufficiently large finite field $\mathbb{F}_q$. This ensures that the code achieves its maximum theoretical minimum distance. From now on, we ignore coefficients in the locality Tanner graph, and with a bit abuse of terminology, we use the terms LRC and locality Tanner graph interchangeably.

Working with locality Tanner graph of a code makes the locality analysis easier. In this dissertation, we restrict ourselves to locality Tanner graphs.

Definition 2.7. *(Local CN, global CN, and local group). In a locality Tanner graph, we call CNs of $\Phi(\mathcal{Y})$ local CNs. We denote by m the total number of local CNs of the locality Tanner graph, i.e., $m := |\Phi(\mathcal{Y})|$. Non-local CNs are called global CNs. The set of variable nodes adjacent to a local CN is called a local group. We call the local group corresponding with local CN i by $\mathcal{Y}_i$. We denote locality of local group $\mathcal{Y}_i$ by r_i, where $i \in [1, m]$.*

Definition 2.8. *(Non-overlapping local groups). We say that some local groups are non-overlapping if they have no variable nodes in common.*

Definition 2.9. *(Information node and parity node). We call variable nodes corresponding with the information blocks and parity blocks information nodes and parity nodes, respectively.*

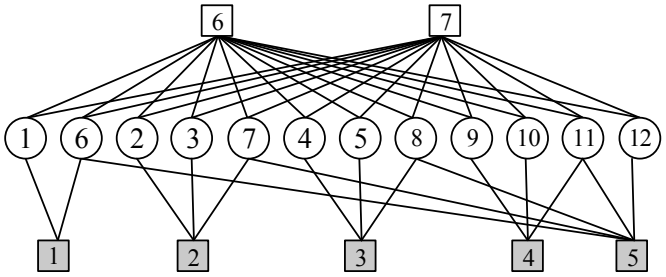


Figure 2.4: Locality Tanner graph of an $(n, k, d) = (12, 5, 6)$ LRC. Variable nodes indexed by 1 to 5 represent the information blocks, and the rest of variable nodes represent the parity blocks.

Example 2.9. *Fig. 2.4 shows the locality Tanner graph of an* $(n, k, d) = (12, 5, 6)$ *LRC. In this figure, variable nodes (circles) indexed by 1 to 5 are information nodes, and variable nodes indexed by 6 to 12 are parity nodes. Local CNs 1 to 5 (gray squares) define the locality of all the variable nodes. Local groups 1 to 4 are non-overlapping while they are overlapping with the local group corresponding with local CN 5. There are also two global CNs (white squares indexed by 6 and 7) linked to all the variable nodes. The parity check matrix corresponding with the Tanner graph depicted in Fig. 2.4, denoted* $\mathbf{H}_0 \in \mathbb{F}_q^{7\times 12}$*, is*

$$\mathbf{H}_0 = \begin{pmatrix} h_{1,1} & 0 & 0 & 0 & 0 & h_{1,6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & h_{2,2} & h_{2,3} & 0 & 0 & 0 & h_{2,7} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & h_{3,4} & h_{3,5} & 0 & 0 & h_{3,8} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_{4,9} & h_{4,10} & h_{4,11} & 0 \\ 0 & 0 & 0 & 0 & 0 & h_{5,6} & h_{5,7} & h_{5,8} & 0 & 0 & h_{5,11} & h_{5,12} \\ h_{6,1} & h_{6,2} & h_{6,3} & h_{6,4} & h_{6,5} & h_{6,6} & h_{6,7} & h_{6,8} & h_{6,9} & h_{6,10} & h_{6,11} & h_{6,12} \\ h_{7,1} & h_{7,2} & h_{7,3} & h_{7,4} & h_{7,5} & h_{7,6} & h_{7,7} & h_{7,8} & h_{7,9} & h_{7,10} & h_{7,11} & h_{7,12} \end{pmatrix},$$

where $h_{i,j}$*, for* $i \in [1, 7]$ *and* $j \in [1, 12]$*, are picked from a sufficiently large finite field.*

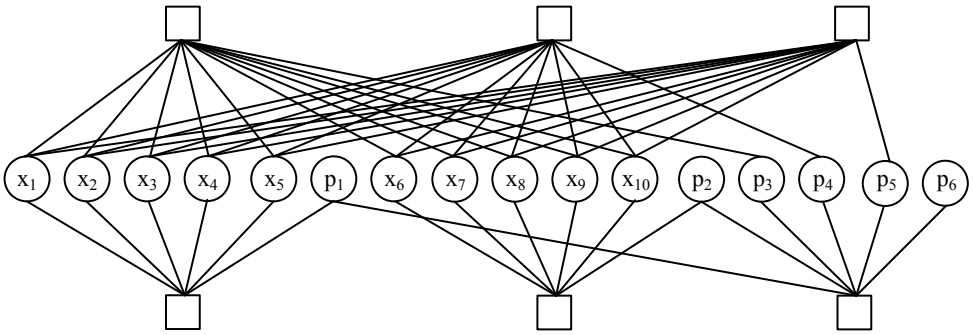


Figure 2.5: The Tanner graph of the $(n, k, d, \overline{r}) = (16, 10, 5, 5)$ LRC with three global and three local parity blocks implemented in Facebook HDFS-RAID [6].

2.6.2 LRCs in Use

LRCs have been recently used in practice, e.g., in Windows Azure Storage [7] and Facebook HDFS-RAID [6].

Example 2.10. *In 2011, as mentioned in example 2.8, an* $(n, k, d, r) = (14, 10, 5, 10)$ *RS was used in Facebook HDFS. In 2013, this code was improved by adding two other parity blocks resulting in an* $(n, k, d, r) = (16, 10, 5, 5)$ *LRC. In comparison with the* $(14, 10)$ *RS code, the modified* $(16, 10)$ *LRC increases the storage overhead from 40% to 60%, but decreases the locality from* 10 *to* 5*. Fig. 2.5 shows the Tanner graph associated with this coding method. In this figure,* x_1 *to* x_{10} *represent information symbols;* p_1 *to* p_3 *represent local parity symbols; and* p_4 *to* p_6 *represent global parity symbols. Since* $d = 5$*, any* $d - 1 = 4$ *failures (symbol erasures) can be tolerated by this coding method. As Fig. 2.5 depicts, locality of all* 16 *symbols is* $r = 5$*.*

Example 2.11. *An* $(n, k, d, \overline{r}) = (16, 12, 4, 6.75)$ *LRC with the storage overhead 33% has been used in Windows Azure Storage since 2012. That is, for every 12 information symbols (systematic blocks), four parity symbols (parity blocks) are formed. These 16 symbols are stored in 16 distinct DNs. Fig. 2.6 shows the Tanner graph associated with this coding method. In this figure,* x_1

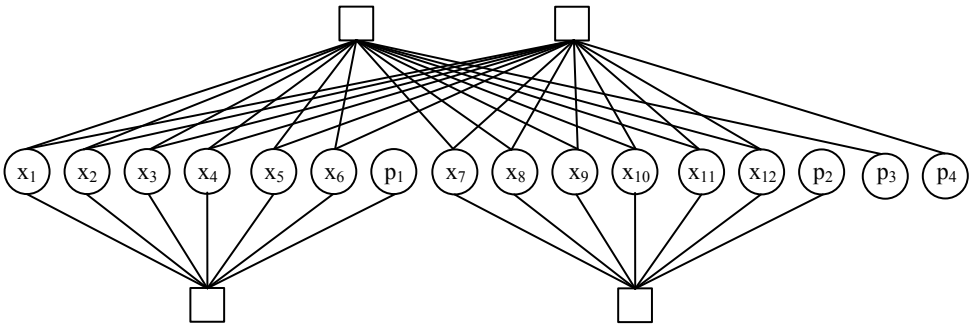


Figure 2.6: The Tanner graph of the $(n, k, d, \overline{r}) = (16, 12, 4, 6.75)$ LRC with 2 global and 2 local parity blocks implemented in Windows Azure Storage [7].

to x_{12} represent information symbols; p_1 and p_2 represent local parity symbols; and p_3 and p_4 represent global parity symbols. p_1 is a linear combination of information symbols x_1 to x_6; and p_2 is a linear combination of information symbols x_7 to x_{12}. Also, p_3 and p_4 are linear combinations of information symbols x_1 to x_{12}. Since $d = 4$, any $d - 1 = 3$ failures associated with the encoded symbols can be tolerated by this coding method. The locality of the following 14 encoded symbols $\{x_1, x_2, \cdots, x_{10}, p_1, p_2\}$ is six; also, the locality of p_3 and p_4 is 12. Hence, the average locality is $\overline{r} = \frac{14\times6+2\times12}{16} = 6.75$.

Chapter 3

On Binary Locally Repairable Codes

Considering the immense size of DSSs, reducing the computational complexity of their coding scheme is desirable. Coding complexity reduction can be beneficial for other operations such as data encryption, data compression, and redundant data removal performed in DSS [50]. LRCs that are currently in use (e.g. in Facebook HDFS-RAID [6], Windows Azure storage [7], and Google file system [10]) work based on additions and multiplications in binary extension fields $\mathbb{F}_{2^m}$. Multiplication is more expensive than addition because its implementation requires algorithms that work based on more hardware and computation [51]. Considering the immense size of data centers, such computations are non-negligible. For example, the LRC introduced in [6] is constructed based on a $(n, k) = (14, 10)$ RS code. Assuming that 30 PB of data is needed to be stored, 1.2×10^{17} multiplications have to be performed to store the total data in the warehouse cluster[1].

Binary LRCs (BLRCs) are of interest in practice, as they eliminate the need for multiplication in operations such as encoding, decoding, and repair. In [35],

[1]Four multiplications per byte are required for encoding of the code in [6].

BLRCs with $r = 2$, and $d \in \{2, 6, 10\}$ are proposed. In [37], a bound on k is obtained which depends on q, r, n, and d. In [38], by using the concept of anticode, BLRCs with $r \in \{2, 3\}$ have been proposed some of which reach the bound in [37]. In [36], using cyclic codes, optimal BLRCs with small locality are proposed. Following is the organization and overview of our main results that will be presented in this chapter.

i) In Section 3.1, we present a mean-time to data-loss (MTTDL) analysis, as it will be needed for reliability evaluation of our codes.

ii) In Section 3.2, we propose a class of BLRCs—called *spanning BLRCs*—whose code rate is only $(\lceil \log_2(r+1) \rceil - 2 + \lfloor \frac{2}{r+1} \rfloor)/n$ less than that of the r-optimal non-binary LRCs satisfying (2.2). Hence, for practical values of r, spanning BLRCs have either identical or slightly lower code rate compared to that of r-optimal non-binary LRCs. More specifically, for $r \in \{1, 2, 3\}$, there is no gap between the rates of our spanning BLRC and the r-optimal non-binary LRC, and for $r \in \{4, 5, 6, 7\}$ the gap is less than $\frac{1}{n}$. Hence, spanning BLRCs are r-optimal for $r \in \{1, 2, 3\}$.

iii) We also verify that spanning BLRCs are d-optimal for $r \in \{1, 3\}$. In particular, we propose optimal binary LRCs for the following set of parameters: $(n, k, r) = (2\iota, \iota - 1, 1)$ and $(n, k, r) = (4\iota, 3\iota - 2, 3)$ for integer ι.

iv) Using the idea behind our design, we propose optimal LRCs over finite fields of size $q \geq r + 2$ for $d = 4$. Comparing with the most recent results in [26], we decrease the required field size from n to $r + 2$.

v) Using the construction of spanning BLRC with $d = 4$ as a backbone, we design LRCs with minimum distance $d \geq 6$. We do this by adding only *one* non-binary parity block to the ones associated with spanning BLRC.

3.1 MTTDL Analysis

The main objective of this section is to argue that for a wide range of practical codes and system parameters, a code minimum distance of four is sufficient and it offers a higher reliability than that of the 3-replication which is widely used in practice [7, 6]. For our argument, we offer an MTTDL analysis. MTTDL of a coding method is an estimate measure of its reliability in DSSs. MTTDL is defined as the average time taken for DSS to miss information of a stripe of data. MTTDL of a storage system can be calculated via dividing MTTDL of a stripe by the total number of stripes in DSS [6]. In the following, we consider calculation of MTTDL of a stripe of data.

Although MTTDL is not perfect in assessing reliability of an LRC in practice, it can be used to estimate reliability (e.g. see [6, 7, 52]). Also, as indicated in [9], MTTDL is more advanced than other existing reliability criteria in the literature as it takes into account not only parameters of the erasure codes (such as n, k, r, d) but also parameters of DSS (such as failure rate, capacity, and repair bandwidth associated with storage nodes as well as total number of storage nodes).

Here, we obtain MTTDL of an (n, k, r) coding method with minimum distance d for a homogeneous DSS, i.e. for a DSS whose storage nodes all have the same characteristics. More specifically, we assume the repair bandwidth, storage capacity, and average failure rate are the same for all storage nodes. We also assume all blocks associated with one stripe of data are distributed among distinct storage nodes. In other words, a storage node failure does not influence more than one block associated with a stripe.

The Markov chain model is considered as a proper tool to evaluate the reliability of DSSs (e.g. see [6, 9, 52]). By the aforementioned assumptions, the Markov chain model of an $(n,\ k)$ erasure code with minimum distance d can be expressed as Fig. 3.1. An exponential distribution with parameter λ

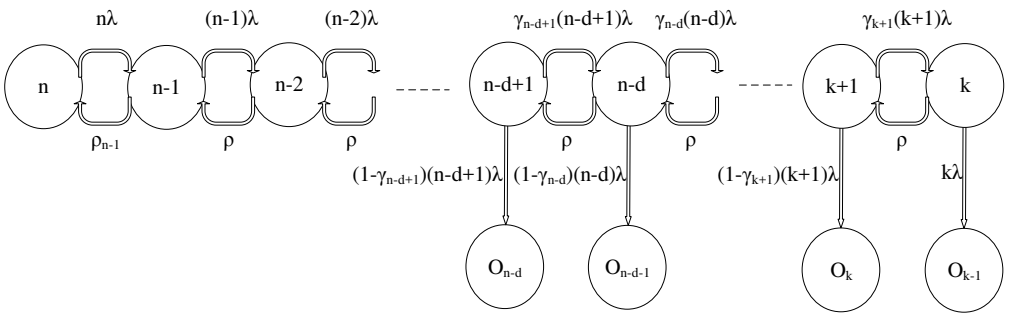


Figure 3.1: Morkov chain model associated with an (n, k) erasure code with minimum distance d.

is considered for storage node failure [52]. In Fig. 3.1, the state label shows the number of surviving nodes. Thus, for $n - i$ surviving nodes each with failure rate of λ, the transition rate from state $n - i$ to state $n - (i - 1)$ is $(n - i)\lambda$. In the case of failure of one block of a stripe, we assume that all surviving nodes participate in the repair process uniformly. Thus, the average repair rate associated with a single block $\rho_{n-1} \simeq \frac{B_{rep}M}{\bar{r}C}$, where B_{rep} is the repair bandwidth of each node, M is the total number of storage nodes, and C is the capacity of each storage node [7]. Furthermore, we assume that T_d—the time elapsed to recognize failures associated with more than one block of a stripe—is significantly larger than the repair time, hence the parameter ρ in Fig. 3.1 is $\rho = \frac{1}{T_d}$ [7]. State O_{n-j} for $j \in [d, n - k + 1]$ is the state where j block failures associated with one stripe of data cannot be recovered, i.e. the state where discarding j columns of $\mathbf{G}$ results in $k \times (n - j)$ sub-matrices of rank less than k. Therefore, γ_{n-j} is defined as the number of $k \times (n - j)$ sub-matrices of $\mathbf{G}$ with rank less than k divided by number of all $k \times (n - j)$ sub-matrices of $\mathbf{G}$, i.e. $\binom{n-j}{k}$. In brief, a code with d can recover any $d - 1$ block failures of a stripe, and $\gamma_{n-j} \times 100$ percent of j block failures of a stripe. In the following, in order to obtain a lower bound for MTTDL, we consider the worst case where $\gamma_{n-j} = 0$ for $j \in [d, n - k + 1]$. We have the following proposition.

Proposition 3.1. *MTTDL of an (n, k) LRC with the average locality $\overline{r} = \big(\sum_{i=1}^{n} \mathsf{Loc}(y_i)\big)/n$ is at least*

$$\frac{\gamma_{n-1}\gamma^{(d-2)}}{n(n-1)...(n-d+1)\lambda},$$

where $\gamma_{n-1} = \frac{\rho_{n-1}}{\lambda} = \frac{B_{rep}M}{\overline{r}C\lambda}$ and $\gamma = \frac{\rho}{\lambda}$.

Proof. Please see Appendix A.1. □

The above proposition shows that MTTDL is proportional to the repair bandwidth, and the total number of storage nodes. Furthermore, MTTDL is inversely proportional to the average locality of LRC and the capacity of each storage node. Proposition 3.1 also shows that MTTDL is exponentially proportional to d. Also, increasing the code length decreases the value of MTTDL.

The 3-replication can be considered as an LRC with $n = 3$, $d = 3$, and $\overline{r} = 1$. Thus, its MTTDL is obtained as $\frac{B_{rep}M\gamma}{6C\lambda^2}$ using Proposition 3.1. Comparing this with the general case of Proposition 3.1, we have the following conclusion: an $(n,\ k)$ LRC with minimum distance d and average locality $\overline{r}$ has a MTTDL not worse than that of the 3-replication scheme if it satisfies the following equation.

$$\zeta = \frac{6\gamma^{d-1}}{\overline{r}n(n-1)...(n-d+1)} \geq 1. \tag{3.1}$$

Therefore, assuming that for given λ and T_d, MTTDL of the 3-replication method is acceptable for a DSS, any LRC satisfying (3.1) can be considered reliable.

Fig. 3.2 shows ζ versus n for $d = 4$ and various values of $\overline{r}$. This figure is plotted for $\lambda = 0.25$ storage node failures per year and $T_d = 30$ minutes [7, 6]. As this figure shows, high values of $\overline{r}$ force the code length n to become smaller. Fig. 3.3 shows ζ versus n for $d = 4$, $\overline{r} = 3$ and different values of T_d. As expected, increasing T_d forces the code length n to become smaller. Both

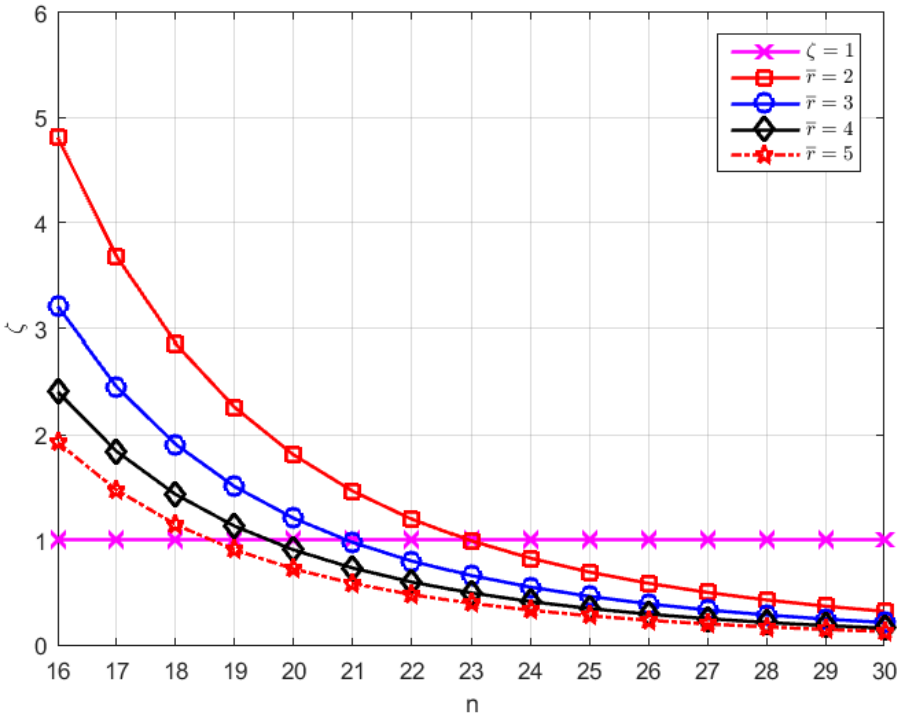


Figure 3.2: ζ versus n for $T_d = 30$ minutes, $d = 4$, and various values of $\overline{r}$.

figures show that with code length as large as 20, $d = 4$ provides reliability more than that of the widely used 3-replication method.

3.2 Spanning BLRCs: A Class of Binary LRCs

Here, we propose a family of BLRCs, called spanning BLRCs, with the minimum distance $d = 4$ (the choice $d = 4$ is justified in Section 3.1). In our coding scheme, similar to [28] and [26], we assume that $(r + 1) \mid n$. Our proposed binary codes are r-optimal or asymptotically r-optimal. In particular, for $r = \{1, 2, 3\}$, our proposed codes are indeed r-optimal. In addition, for $r = \{1, 3\}$, the proposed binary codes are both r-optimal and d-optimal, implying that neither their locality nor their minimum distance can be improved by non-binary codes of the same length and rate. For $r \geq 4$, the rate of our codes asymptotically approaches that of the non-binary r-optimal codes of the

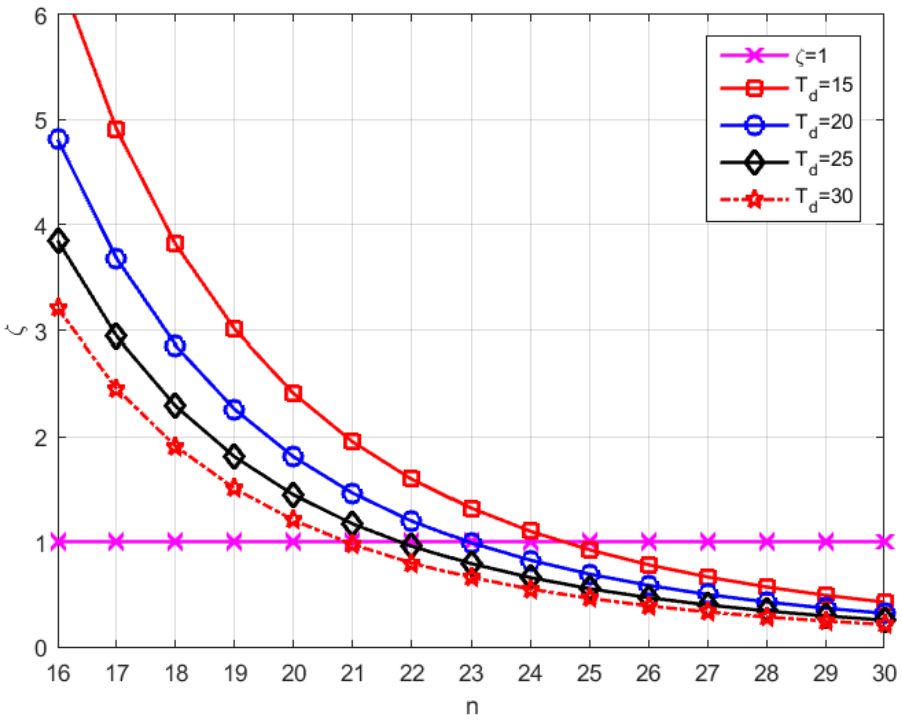


Figure 3.3: ζ versus n for $\overline{r} = 3$, $d = 4$, and various values of T_d.

same code length as $n \to \infty$.

Later, in Section 3.3, by allowing two non-binary rows of field size $q \geq r + 2$ in the parity check matrix $\mathbf{H}$, we construct r-optimal codes for all r and n, $(r+1)|n$. These codes also enjoy low computational complexity as most operations on them can be performed in binary. For example, any single failure (the dominant failure scenario) can be fully repaired only using the binary part of the code.

3.2.1 BLRC Construction

We describe our code construction using its Tanner graph. In our code design, first, n variable nodes are divided into $\iota = \frac{n}{r+1}$ groups. Then, as shown in Fig. 3.4, all variable nodes associated with each group are linked to a unique check node. Let us call such check nodes *local* check nodes, and the remaining ones *global* check nodes. This construction results in the maximum locality r

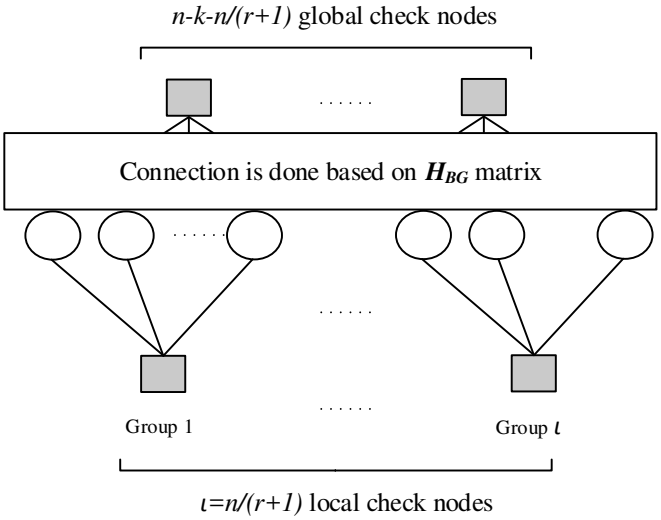


Figure 3.4: Tanner graph of spanning BLRCs.

for all symbols. Equivalently, we can specify the parity check matrix of our proposed BLRC as follows:

$$\mathbf{H}_{BLRC} = \begin{pmatrix} \mathbf{H}_{BL} \\ \mathbf{H}_{BG} \end{pmatrix} \in \mathbb{F}_2^{(n-k)\times n}, \tag{3.2}$$

where $\mathbf{H}_{BL} \in \mathbb{F}_2^{\frac{n}{r+1}\times n}$ and $\mathbf{H}_{BG} \in \mathbb{F}_2^{(n-k-\frac{n}{r+1})\times n}$ represent the connection associated with local and global check nodes, respectively. According to the above explanation,

$$\mathbf{H}_{BL} = \mathbf{I}_{\frac{n}{r+1}} \otimes \mathbf{1}_{r+1} \in \mathbb{F}_2^{\frac{n}{r+1}\times n}. \tag{3.3}$$

$\mathbf{H}_{BL} \in \mathbb{F}_2^{\frac{n}{r+1}\times n}$ constitute $\frac{n}{r+1}$ rows of $\mathbf{H}_{BLRC}$ in (3.2). We now specify $\mathbf{H}_{BG} \in \mathbb{F}_2^{(n-k-\frac{n}{r+1})\times n}$ as follows.

$$\mathbf{H}_{BG} = \mathbf{1}_{\frac{n}{r+1}} \otimes \mathbf{H}_0^{(r)} \in \mathbb{F}_2^{\lceil \log_2(r+1)\rceil \times n}. \tag{3.4}$$

In (3.4), $\mathbf{H}_0^{(r)}$ is the parity check matrix of an $(r+1, r+1-\lceil \log_2(r+1) \rceil)$ Hamming code. In other words,

$$\mathbf{H}_0^{(r)} = \Big(\ \mathbf{0}, \ \ \mathbf{1}, \ \ \cdots, \ \ \mathbf{r} \ \Big) \in \mathbb{F}_2^{\lceil \log_2(r+1) \rceil \times (r+1)}, \tag{3.5}$$

where each vector $\mathbf{m} \in \{\mathbf{0}, \cdots, \mathbf{r}\}$ is the transpose of the binary representation of $m \in [0, r]$ with $\lceil \log_2(r+1) \rceil$ bits. For example, if $r = 5$ and $m = 3$, then $\mathbf{m} = (0, 1, 1)^T$. Hence,

$$k = \frac{nr}{r+1} - \lceil \log_2(r+1) \rceil. \tag{3.6}$$

Considering (3.2), (3.3), and (3.4), $\mathbf{H}_{BLRC}$ has the following form

$$\begin{pmatrix} \overbrace{\begin{matrix} 1 & \cdots & 1 \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{matrix}}^{\mathbf{H}_{Group}^{(1)}} & \overbrace{\begin{matrix} 0 & \cdots & 0 \\ 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{matrix}}^{\mathbf{H}_{Group}^{(2)}} & \cdots & \overbrace{\begin{matrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \\ 1 & \cdots & 1 \end{matrix}}^{\mathbf{H}_{Group}^{(\iota)}} \\ \mathbf{H}_0^{(r)} & \mathbf{H}_0^{(r)} & \cdots & \mathbf{H}_0^{(r)} \end{pmatrix}. \tag{3.7}$$

Example 3.1. *Assume that $r = 2$ and $n = 9$. Hence, the number of local check nodes is $\iota = \frac{n}{r+1} = 3$, and the number of global check nodes is $n - k - \iota = 2$. From (3.5), we have*

$$\mathbf{H}_0^{(2)} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \in \mathbb{F}_2^{2 \times 3}. \tag{3.8}$$

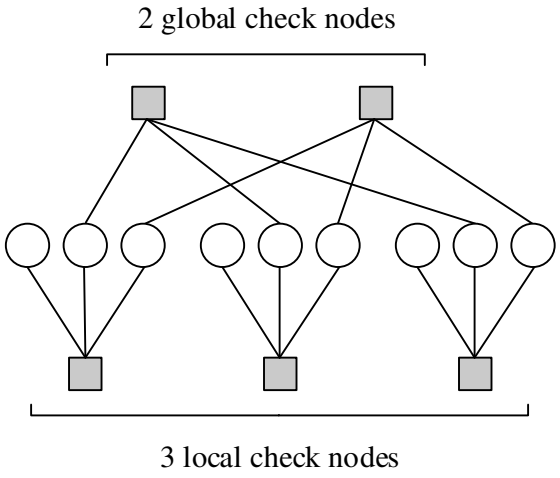


Figure 3.5: Tanner graph related to the $(9, 4, 2)$ spanning BLRC.

From (3.7), $\mathbf{H}_{BLRC}$ *is constructed as*

$$\mathbf{H}_{BLRC} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \in \mathbb{F}_2^{5\times 9}.$$

Fig. 3.5 shows the Tanner graph associated with the $(9, 4, 2)$ *spanning BLRC.*

Proposition 3.2. *The minimum distance* d *associated with the parity check matrix* $\mathbf{H}_{BLRC}$ *defined in (3.2) is four.*

Proof. For a linear block code with $d = 4$, every $d - 1 = 3$ columns of parity check matrix must be independent. No column of $\mathbf{H}_{BLRC}$ is a vector of all zeros. Now consider, sub-matrices of $\mathbf{H}_{BLRC}$ associated with l-th group as

$$\mathbf{H}^{(l)}_{Group} = \mathbf{H}_{BLRC}([(l-1)(r+1)+1, l(r+1)]), \text{ for } l \in [\frac{n}{r+1}]. \tag{3.9}$$

Observe that no two columns of $\mathbf{H}^{(r)}_0$ are identical. Therefore, sum of any

two columns picked from $\mathbf{H}_{Group}^{(l)}$ cannot result in a vector of all zeros. Also, sum of any three columns of $\mathbf{H}_{Group}^{(l)}$ has at least one non-zero element because $\mathbf{H}_{Group}^{(l)}$ has a row of all ones. This shows that $\mathbf{H}_{Group}^{(l)}$'s for $l \in [\frac{n}{r+1}]$ have $d = 4$. Furthermore, considering the structure of $\mathbf{H}_{BL}$, observe that sum of one column picked from $\mathbf{H}_{Group}^{(j)}$ and arbitrary number of columns picked from $\mathbf{H}_{Group}^{(l)}, l \neq j$ results in a vector with at least one non-zero element. Hence, no linear combinations of less than four columns of $\mathbf{H}_{BLRC}$ can result in a vector of all zeros. Also, observe that XOR of columns 1, 2, $r+2$, and $r+3$ of $\mathbf{H}_{BLRC}$ results in a vector of all zeros. This implies that $d = 4$ for $\mathbf{H}_{BLRC}$. □

To sum up, $\mathbf{H}_{BLRC}$ is the parity check matrix of an (n, k, r) binary LRC with minimum distance $d = 4$ whose number of information symbols (k_B) is

$$k_B = \frac{nr}{r+1} - \lceil \log_2(r+1) \rceil. \tag{3.10}$$

Remark 3.1. *Assuming that $(r+1)|n$ and $d = 4$, the code rate of spanning BLRCs is $(\lceil \log_2(r+1) \rceil - 2 + \lfloor \frac{2}{r+1} \rfloor)/n$ less than that of r-optimal non-binary LRCs. To show this, observe that for $d = 4$ by (2.2), we have*

$$k + \frac{k}{r} \leq n - 2. \tag{3.11}$$

Multiplying both sides of (3.11) by $\frac{r}{r+1}$, and keeping in mind that $(r+1) \mid n$, we get

$$k \leq \frac{nr}{r+1} - \left\lceil \frac{2r}{r+1} \right\rceil. \tag{3.12}$$

Noting that $r/(r+1) = 1 - 1/(r+1)$ and $\lceil -x \rceil = -\lfloor x \rfloor$, the number of information symbols of r-optimal non-binary LRCs (k_{r-opt}) is

$$k_{r-opt} = \frac{nr}{r+1} - 2 + \left\lfloor \frac{2}{r+1} \right\rfloor. \tag{3.13}$$

Let us denote the code rate of spanning BLRCs and r-optimal non-binary LRCs

by R_B and R_{r-opt}, respectively. Considering (3.10) and (3.13), we have

$$R_B - R_{r-opt} = \frac{k_B - k_{r-opt}}{n} = -\frac{\lceil \log_2(r+1) \rceil - 2 + \lfloor \frac{2}{r+1} \rfloor}{n}.$$

This implies that the binary construction of our proposed LRCs does not sacrifice the code rate significantly for practical values of r. This can be of interest because the computation cost decreases significantly when all multiplication operations are removed by using binary coding.

Remark 3.2. *Observe that*

$$\frac{\lceil \log_2(r+1) \rceil - 2 + \left\lfloor \frac{2}{r+1} \right\rfloor}{n} = \begin{cases} 0, & r \in \{1,2,3\} \\ \mathcal{O}(\frac{\log r}{n}) & r \geq 4 \end{cases}$$

This means that spanning BLRC are r-optimal for $r \in \{1,2,3\}$.

An interesting question is what is the best minimum distance d for given n, r, and k_{r-opt} in (3.13)? In other words, are r-optimal LRCs satisfying (3.13) d-optimal LRCs?

Remark 3.3. *For an LRC with given (n, k_{r-opt}, r), the maximum value of minimum distance d, denoted d_{opt}, is*

$$d_{opt} = \begin{cases} 5, & r = 2, \\ 4, & r \neq 2, \end{cases} \tag{3.14}$$

To show this, observe that by replacing (3.13) in (2.2), we have

$$\begin{aligned} d &\leq \iota + 2 - \left\lfloor \frac{2}{r+1} \right\rfloor - \left\lceil \frac{r\iota - 2 + \lfloor \frac{2}{r+1} \rfloor}{r} \right\rceil + 2 \\ &= 4 - \left\lfloor \frac{2}{r+1} \right\rfloor + \left\lfloor \frac{2 - \lfloor \frac{2}{r+1} \rfloor}{r} \right\rfloor = \begin{cases} 5, & r = 2 \\ 4, & r \neq 2 \end{cases} \end{aligned}$$

Remark 3.4. *Considering Remarks 3.2 and 3.3, spanning BLRCs are optimal LRCs for* $r \in \{1, 3\}$. *More specifically, spanning BLRC with* $(n, k_{r-opt}, r) = (2\iota, \iota-1, 1)$ *and* $(n, k_{r-opt}, r) = (4\iota, 3\iota-2, 3)$ *have the optimal minimum distance* $d_{opt} = 4$. *Hence, considering the three key parameters of an LRC—namely the code rate* $\frac{k}{n}$*, the code locality* r*, and the code minimum distance* d *when* $r \in \{1, 3\}$*—non-binary LRCs have no advantage over the spanning BLRCs.*

It is notable that spanning BLRC with $(n, k_{r-opt}, r) = (3\iota, 2\iota - 2, 2)$ satisfy the bound in (2.2) with equality (i.e. they are r-optimal), but their minimum distance can be increased by non-binary codes to $d = 5$ according to Remark 3.3.

In the following, we present an example of optimal spanning BLRC for the following parameters: $(r, \iota, d_{opt}) = (3, 4, 4)$. Hence, $(n, k_{r-opt}) = ((3 + 1)\iota, 3 \times \iota - 2) = (16, 10)$. Therefore, we have $\iota = \frac{n}{r+1} = 4$ and $n - k_{r-opt} - \iota = 2$ local and global check nodes, respectively. From (3.5), we have

$$\mathbf{H}_0^{(3)} = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} \in \mathbb{F}_2^{2\times 4}. \tag{3.15}$$

Then, from (3.7), $\mathbf{H}_{BLRC} \in \mathbb{F}_2^{6\times 16}$ is constructed as

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

Fig. 3.6 shows the Tanner graph associated with the $(16, 10)$ spanning BLRC. The $(16, 10)$ LRC used in Facebook HDFS-RAID does not achieve the

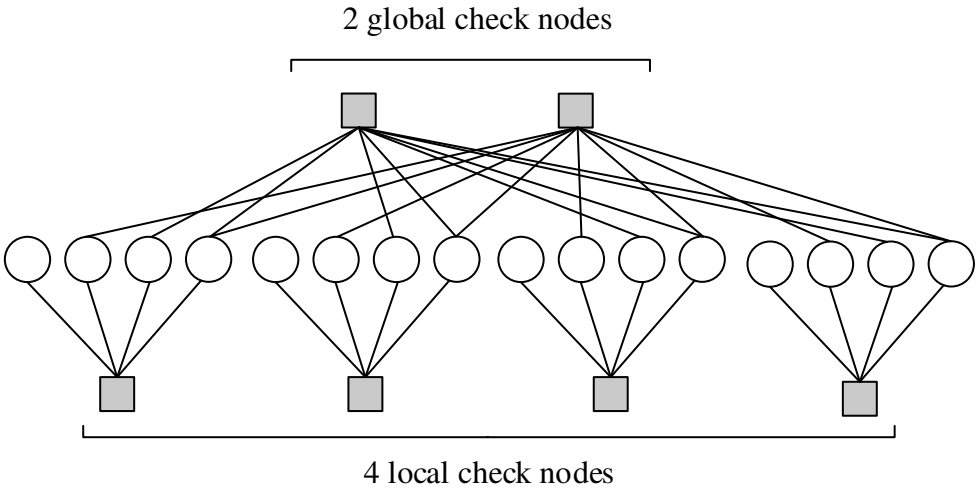


Figure 3.6: Tanner graph related to the $(16, 10)$ spanning BLRC.

bound in 2.2. However, it is shown in [6] that given $(n, k, r) = (16, 10, 5)$, the best possible minimum distance is 5 (observe that in this case $(r + 1) = 6 \nmid 16 = n$). Comparing with the $(n, k, r, d) = (16, 10, 5, 5)$ LRC currently in use in Facebook HDFS-RAID, our proposed $(n, k, r, d) = (16, 10, 3, 4)$ spanning BLRC results in the same storage overhead, decreases r from 5 to 3, eliminates the need of multiplication due to its binary essence, and it decreases d by one. While the minimum distance of our spanning BLRC is one less than that of the LRC used in Facebook HDFS-RAID, as will be discussed in Section 3.1, we still provide a reliability much better than that of the benchmark 3-replication scheme. Also note that in this case, increasing the field size does not improve the code locality, storage overhead, and reliability (which are the key parameters of an LRC used in a DSS).

In order to compare the coding speed associated with binary and non-binary LRCs, we simulated our $(16, 10)$ spanning BLRC and the $(16, 10)$ LRC used in Facebok HDFS-RAID using advanced vector extensions (AVX) instructions [34] and Intel® intelligent storage acceleration library (Intel® ISA-L). Using AVX makes it possible to process multiple pieces of data in a single step. ISA-

L uses a number of low-level functions to optimize the coding speed in storage applications. In ISA-L, all operations are performed over $\mathbb{F}_{2^8}$ (byte level) with primitive polynomial $x^8+x^4+x^3+x^2+1$ (please find more details in [34, 53, 54]). Since the only comparison parameter was the coding pace, our simulations are performed on a single machine. We used data blocks of size 1 kB, 2 kB, 4 kB, and 8 kB. Our simulation verifies that the encoding and decoding processes of spanning BLRC are on average ten times faster than those of the code used in Facebook HDFS-RAID. For the repair process associated with one missing block, both methods offer the same pace. This is because in both cases, the code structures allow performing the repair process of a single failure by XOR operations.

It is also notable that since the minimum distance of our code is four, the code can recover any pattern of less than four block failures associated with one stripe. By numerically inspecting all 6×4, 6×5, and 6×6 sub-matrices of $\mathbf{H}_{BLRC} \in \mathbb{F}_2^{6\times 16}$, the following results are also verified: 4% of all 6×4, 21% of all 6×5, and 58% of all 6×6 sub-matrices of $\mathbf{H}_{BLRC} \in \mathbf{F}_2^{6\times 16}$ have rank less than 4, 5, and 6, respectively. In other words, respectively 96%, 79%, and 42% of four, five, and six block failures associated with one stripe of data can be recovered by using our proposed $(16, 10, 3)$ binary LRC [2].

3.2.2 Optimal Binary Codes

We discussed earlier that for $d = 4$ our BLRCs are optimal in some cases. Also, when they are not optimal, we showed that they offer near optimal code rate with a rate gap of $\mathcal{O}\left(\frac{\log r}{n}\right)$. An interesting question is whether binary codes with higher code rate than our proposed binary codes exist. In the following, we show that our binary codes offer the maximum code rate among codes preserving the general structure of local check nodes (shown in Fig. 3.4).

[2]Recently, erasure codes whose goal is to optimize recovering failures beyond code minimum distance have been proposed (e.g. see [55, 56] and references therein).

Consider an (n, k, r) BLRC with $\iota = \frac{n}{r+1}$ local check nodes as shown in Fig. 3.4. Therefore, the matrix $\mathbf{H}_{BLRC}$ has the following form:

$$\begin{pmatrix} \overbrace{\begin{matrix} 1 & \cdots & 1 \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{matrix}}^{\mathbf{H}^{(1)}_{Group}} & \overbrace{\begin{matrix} 0 & \cdots & 0 \\ 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{matrix}}^{\mathbf{H}^{(2)}_{Group}} & \cdots & \overbrace{\begin{matrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \\ 1 & \cdots & 1 \end{matrix}}^{\mathbf{H}^{(\iota)}_{Group}} \\ \mathbf{H}_1 & \mathbf{H}_2 & \cdots & \mathbf{H}_\iota \end{pmatrix}.$$

Consider the submatrix $\mathbf{H}_1$. The number of columns of $\mathbf{H}_1$ is $r+1$. Therefore, the binary $\mathbf{H}_1$ will have two identical columns c_1 and c_2 if the number of rows of $\mathbf{H}_1$ is less than $\lceil \log_2(r+1) \rceil$. In this case, columns c_1 and c_2 of the matrix $\mathbf{H}_{BLRC}$ will be identical, implying that three failures in Group 1, two of which corresponding to columns c_1 and c_2 are not repairable. This contradicts $d = 4$.

In our construction, the columns of submatrix $\mathbf{H}_1$ (similarly $\mathbf{H}_2, \dots$) span binary representations of numbers 0 to r, hence the name spanning BLRC.

3.2.3 Systematic Construction of Spanning BLRC

From a practical point of view, it is more desirable to access the stored data blocks without decoding. Thus, systematic codes are preferred in DSSs. In order to obtain the systematic form of generator matrix associated with spanning BLRC, we first form a square full-rank matrix $\mathbf{H}_{FR}$ as

$$\mathbf{H}_{FR} = \mathbf{H}(\mathcal{I}_{FR}) \in \mathbb{F}_2^{(\frac{n}{r+1}+\lceil \log_2(r+1) \rceil)\times(\frac{n}{r+1}+\lceil \log_2(r+1) \rceil)},$$

where $\mathcal{I}_{FR}$ is a subset of set $[n]$ with cardinality $\left(\frac{n}{r+1}+\lceil \log_2(r+1) \rceil\right)$ resulting in a full-rank matrix $\mathbf{H}(\mathcal{I}_{FR})$. Then, from $\mathbf{H}' = \mathbf{H}_{FR}^{-1}\mathbf{H}$, the systematic generator matrix of spanning BLRC is obtained.

3.3 Optimal LRCs Over Finite Field of Size $r+2$

In [26], optimal LRCs have been proposed over $\mathbb{F}_q$ for q slightly larger than n, and $(r+1) \mid n$. In Section 3.2.1, we proposed optimal BLRCs for $r \in \{1,3\}$ (please see Remark 3.2). In this section, we construct LRCs over $\mathbb{F}_q$—which is a binary extension with a size greater than or equal to $(r+2)$—when $d=4$ and $(r+1) \mid n$. We show that the constructed codes are d-optimal. In addition, we prove that those codes are also r-optimal (hence are optimal) when $n \geq 2r+3$. We construct the parity check matrix as

$$\mathbf{H}_{LRC} = \begin{pmatrix} \mathbf{H}_L \\ \mathbf{H}_G \end{pmatrix} \in \mathbb{F}_q^{(\frac{n}{r+1}+2)\times n}, \tag{3.16}$$

where

$$\mathbf{H}_L = \mathbf{I}_{\frac{n}{r+1}} \otimes \mathbf{1}_{r+1} \in \mathbb{F}_q^{\frac{n}{r+1}\times n} \tag{3.17}$$

and

$$\mathbf{H}_G = \mathbf{1}_{\frac{n}{r+1}} \otimes \mathbf{H}_1^{(r)} \in \mathbb{F}_q^{2\times n}. \tag{3.18}$$

In (3.18),

$$\mathbf{H}_1^{(r)} = \begin{pmatrix} 1 & \alpha & \cdots & \alpha^r \\ 1 & \alpha^2 & \cdots & \alpha^{2r} \end{pmatrix} \in \mathbb{F}_q^{2\times(r+1)}, \tag{3.19}$$

where α is a primitive element of field $\mathbb{F}_q$. Using the fact that every $3\times(r+1)$ sub-matrix of

$$\mathbf{H}_1'^{(r)} = \begin{pmatrix} \mathbf{1}_{r+1} \\ \mathbf{H}_1^{(r)} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \alpha & \cdots & \alpha^r \\ 1 & \alpha^2 & \cdots & \alpha^{2r} \end{pmatrix} \in \mathbb{F}_q^{3\times(r+1)}$$

has full-rank, it is not difficult to show that the minimum distance of the code is four.

Proposition 3.3. *The proposed LRCs are d-optimal.*

Proof. For the proposed code with code length n, and locality r, we have

$$k = n - \underbrace{\left(\frac{n}{r+1} + 2\right)}_{\text{No. of parities}} = \frac{nr}{r+1} - 2.$$

Therefore, by (2.2), we have

$$\begin{aligned} d &\leq n - k - \left\lceil \frac{k}{r} \right\rceil + 2 \\ &= \frac{n}{r+1} + 2 - \left\lceil \frac{\frac{nr}{r+1} - 2}{r} \right\rceil + 2 \\ &= 4 - \left\lceil -\frac{2}{r} \right\rceil = 4 + \left\lfloor \frac{2}{r} \right\rfloor, \end{aligned}$$

which implies that $d \leq 4$ when $r \geq 3$. □

Proposition 3.4. *The proposed LRCs are optimal if $n \geq 2r + 3$.*

Proof. Suppose $n \geq 2r + 3$. Since it was previously shown that the proposed codes are d-optimal, it suffices to show that they are r-optimal. By contradiction, assume that there is a code with code length n, $k = \frac{nr}{r+1} - 2$, $d = 4$, and locality $r' < r$. Therefore, by (2.2), we have

$$d \leq n - k - \left\lceil \frac{k}{r'} \right\rceil + 2 =$$
$$\frac{n}{r+1} + 2 - \left\lceil \frac{\frac{nr}{r+1} - 2}{r'} \right\rceil + 2 = 4 - \left\lceil \frac{\frac{n(r-r')}{r+1} - 2}{r'} \right\rceil$$

Therefore, since $d = 4$, we must have $\left\lceil \frac{\frac{n(r-r')}{r+1} - 2}{r'} \right\rceil \leq 0$, hence $\frac{n(r-r')}{r+1} \leq 2$, or equivalently

$$n \leq \frac{2(r+1)}{r-r'} \leq 2(r+1),$$

which contradicts the assumption that $n \geq 2r + 3$. □

Note that for typical choices of n and r, $n \geq 2r + 3$, for which our design codes are optimal. In comparison to the existing optimal codes, our codes use smaller field sizes ($r + 2$ vs. n). Furthermore, their local parity check nodes are binary, therefore one failure (the dominant case) can be repaired without using costly multiplications.

3.4 Increasing the Minimum Distance

Motivated by the practical application of erasure codes with $d = 4$, we focused on designing BLRCs with minimum distance four in the previous sections. We showed that our proposed spanning BLRCs have minimum distance of 4, hence they can recover any patterns of one to three missing blocks of a stripe with *binary* operations. As numerically inspected in Fig. 3.7, a high percentage of up to four block failures associated with a stripe can also be recovered by the spanning BLRCs.

Note that 98.8% of stripes with missing blocks have precisely one missing block, 1.87% of them have two missing blocks, and 0.05% of them have three or more missing blocks [24]. Therefore, in real-world data centers, it is very unlikely that four block failures in one stripe occur. In case of such a rare occurrence, it is very unlikely that spanning BLRCs cannot fix it because spanning BLRCs can recover a high percentage of four block failures. For example, our proposed $(16, 10)$ BLRC can recover 96% of four failures associated with a stripe of data.

Nevertheless, in some rare cases, catastrophic failures may occur rendering data missing/unavailable. In such cases, erasure codes with minimum distance more than four are required. In this section, we use our solution for $d = 4$ as a base and construct LRCs with $d \geq 6$ by adding a *single* non-binary check node. Interestingly and somewhat surprisingly, adding this single parity block increases the code minimum distance by two. Note that for up to 3 block

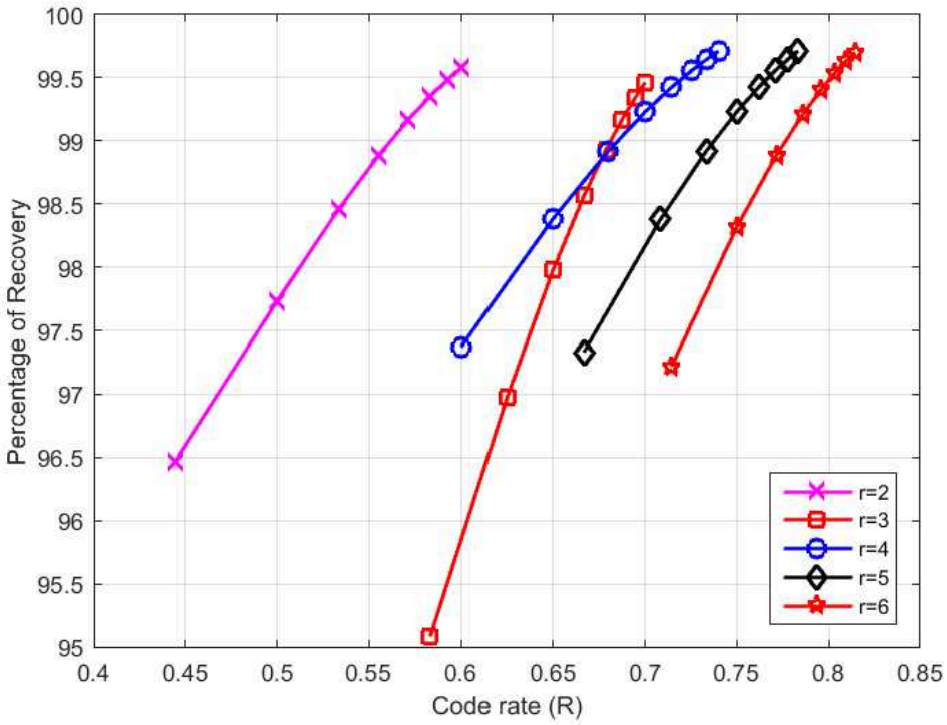


Figure 3.7: The percentage of up to four missing blocks recoverable by our proposed spanning BLRCs for different values of code locality (r) and code rate ($R = \frac{k}{n}$). Here, $n = \iota(r+1)$ and $k = \iota r - \lceil \log_2(r+1) \rceil$, where $\iota \in [3, 10]$.

failures, all recovery operations are performed in binary field as before and this single non-binary parity block is only used in rare cases that more than 3 failures occur.

For the encoding process, non-binary operations are required to generate a single non-binary parity block. Also, for the decoding process, operation over non-binary field is required only if the single non-binary check node has to be used. In the following, the construction of our proposed LRC is presented.

Consider the expanded form of $\mathbf{H}_{BLRC}$ presented in (3.7). Each column of $\mathbf{H}_{BLRC}$ stands for the transpose of the binary representation of a distinct number in binary extension field $\mathbb{F}_{2^{\Gamma_{r,n}}}$, where $\Gamma_{r,n} = \frac{n}{r+1} + \lceil \log_2(r+1) \rceil$. In

other words, $\mathbf{H}_{BLRC}$ can be presented as follows

$$\mathbf{H}_{BLRC} = (\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_n) \in \mathbb{F}_2^{\Gamma_{r,n} \times n}, \tag{3.20}$$

where each column vector $\mathbf{h}_i \in \mathbb{F}_2^{\Gamma_{r,n} \times 1}$ for $i \in [n]$ is the transpose of binary representation of $h_i \in \mathbb{F}_{2^{\Gamma_{r,n}}}$ with $\Gamma_{r,n}$ bits. The parity check matrix associated with our proposed LRC with $d \geq 6$ can be represented as

$$\mathbf{H}_{LRC} = \begin{pmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \cdots & \mathbf{h}_n \\ h_1^3 & h_2^3 & \cdots & h_n^3 \end{pmatrix} \in \mathbb{F}_{2^{\Gamma_{r,n}}}^{(1+\Gamma_{r,n}) \times n}. \tag{3.21}$$

Example 3.2. *In Example 3.1,* $\Gamma_{2,9} = 5$. *Hence,* $\mathbf{H}_{BLRC}$ *can be represented as:*

$$\begin{aligned} \mathbf{H}_{BLRC} &= (\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3, \mathbf{h}_4, \mathbf{h}_5, \mathbf{h}_6, \mathbf{h}_7, \mathbf{h}_8, \mathbf{h}_9) \\ &= (\mathbf{16}, \mathbf{17}, \mathbf{18}, \mathbf{8}, \mathbf{9}, \mathbf{10}, \mathbf{4}, \mathbf{5}, \mathbf{6}) \in \mathbb{F}_2^{5 \times 9} \end{aligned}$$

Considering $h_i \in \mathbb{F}_{2^5}$*, we have:*

$$\begin{aligned} &\left(h_1^3,\ h_2^3,\ h_3^3,\ h_4^3,\ h_5^3,\ h_6^3,\ h_7^3,\ h_8^3,\ h_9^3\right) \\ &= \left(16^3, 17^3, 18^3, 8^3, 9^3, 10^3, 4^3, 5^3, 6^3\right) \\ &= (14, 18, 22, 26, 25, 3, 10, 31, 23) \in \mathbb{F}_{2^5}^{1 \times 9} \end{aligned}$$

Finally, we have

$$\mathbf{H}_{LRC} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 14 & 18 & 22 & 26 & 25 & 3 & 10 & 31 & 23 \end{pmatrix} \in \mathbb{F}_{2^5}^{6 \times 9}$$

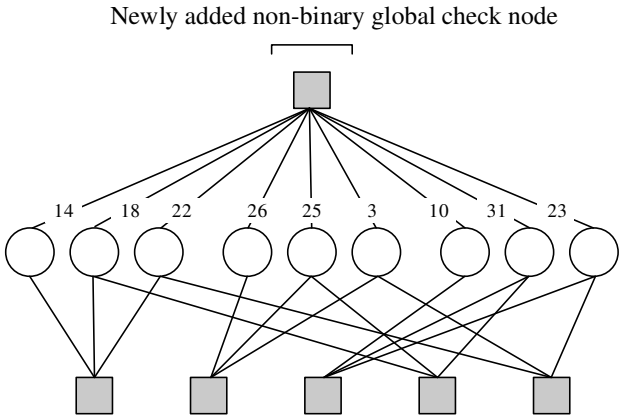


Figure 3.8: Tanner graph related to our proposed $(9, 3, 2)$ LRC with $d = 6$. Five lower check nodes represent five binary equations. The single upper check node represents the non-binary equation.

Fig. 3.8 shows the Tanner graph associated with the $(9, 3, 2)$ *spanning BLRC with* $d = 6$.

Proposition 3.5. *The minimum distance* d *associated with the parity check matrix* $\mathbf{H}_{LRC}^{(6)}$ *defined in (3.21) is bounded as follows:*

$$6 \leq d \leq \left\lceil \log_2(r+1) \right\rceil + \left\lfloor \frac{1}{r} \Big(\left\lceil \log_2(r+1) \right\rceil + 1 \Big) \right\rfloor + 3.$$

Proof. Please see Appendix A.2. □

Remark 3.5. *By adding only one non-binary parity block to the BLRC, our proposed LRC with* $d \geq 6$ *is obtained. In the case that LRCs with* $d > 6$ *are required, it might be possible to follow a similar solution. In other words, it might be possible to increase* d *by carefully adding more rows to* $\mathbf{H}_{BLRC}$ *presented in (3.2). Determining the coefficients of the newly added rows explicitly requires further research.*

3.5 Conclusion

In distributed storage systems, BLRCs are of interest because they eliminate the need for costly multiplications in encoding, decoding, and repair processes.

In this chapter, we proposed binary LRCs with minimum distance of four, a distance shown by our MTTDL analysis to provide sufficient reliability for a wide range of practical codes and system parameters. We showed that our proposed binary LRCs are optimal when $r \in \{1, 3\}$. For larger r, we discussed that the rate of our proposed binary codes are near optimal (with a rate gap of $\mathcal{O}\left(\frac{\log r}{n}\right)$). We further showed that with a field size of as small as $r + 2$, and only two non-binary parity blocks, optimal codes with optimal rates can be constructed when the code length $n \geq 2r + 3$. Also, by adding a single non-binary parity block, we designed LRCs with minimum distance at least six.

Chapter 4

On the Average Locality of Locally Repairable Codes

In the previous chapter, we studied the problem of LRCs with low coding computational complexity. Here we focus on average locality of LRCs. The average locality of an LRC determines the average bandwidth usage and the average number of disk I/O operations needed in reconstructing a block. For example, assume that S_{DN} and S_{Blk} represent size of a DN and size of a data block in bytes, respectively. Then, each DN can store $\frac{S_{DN}}{S_{Blk}}$ blocks of data. When a DN fails, on average, $\overline{r}\frac{S_{DN}}{S_{Blk}}$ blocks each of size S_{Blk} bytes have to be downloaded from active DNs in order to perform the repair process. Therefore, the repair bandwidth is equal to $\overline{r}\frac{S_{DN}}{S_{Blk}} \times S_{Blk} = \overline{r}S_{DN}$ bytes. Because S_{DN} (i.e. the capacity of a DN) is typically several TB, and since DN failure and unavailability are norm in large DSSs, even a small reduction in $\overline{r}$ can result in a significant reduction in the networking load [6].

Furthermore, mean-time to data-loss (MTTDL) associated with an LRC is inversely proportional to the average locality of the code as shown in Proposition 3.1. In other words, for two LRCs with the same values of (n, k, d) but different values of $\overline{r}$, the one with a smaller value of $\overline{r}$ has a higher MTTDL, hence more reliable.

In this chapter, we present our results on the average locality of all symbols. As we will show in (4.3), the bound $d \leq n-k-\lceil \frac{k}{r} \rceil + 2$ is simply translated to $r \geq \lceil \frac{k}{n-k-d+2} \rceil$, which is a lower bound on the maximum locality. The aim of this chapter is to *i*) obtain lower bounds on $\overline{r}$ in terms of n, k, and d; and *ii*) design LRCs that achieve these bounds. In the following section, we present preliminaries required to establish our main results on the average locality.

4.1 Preliminary Results

In this section, we present definitions and lemmas required in discussing and proving our main results. First, we show that there exists a Tanner graph, called locality Tanner graph, with at most $n-k$ CNs which determines the locality of all the encoded symbols. Then, after defining and constructing local groups, we establish lower bounds on the average locality $\overline{r}$ in Section 4.2.1.

In Definition 2.6, we called code's Tanner graph a locality Tanner graph, if it reflects the locality of all the encoded symbols, and has at most $n-k$ CNs.

Lemma 4.1. *Every LRC has a locality Tanner graph.*

Proof. Our proof is by construction. Let $\mathcal{Y}$ be the set of n variable nodes of an LRC code, i.e., $\mathcal{Y} = \{y_1, \cdots, y_n\}$ and $\mathcal{P}_L$ be the set of local check nodes of the code's locality Tanner graph. In the first step, we set $\mathcal{U}_1 = \{\}$, and select a local group Ψ_1 with minimum locality. In step j, $j \geq 2$, we set $\mathcal{U}_j = \mathcal{U}_{j-1} \cup \Psi_{j-1}$, and set Ψ_j to a local group with minimum locality, which is not a subset of $\mathcal{U}_j$. We stop this recursive construction process when $\mathcal{U}_j$ includes all the variable nodes. Note that each local group represents a CN. Since there are at most $n-k$ linearly independent equations of form (2.1), the algorithm terminates at most after $n-k$ local groups are selected, i.e., $|\mathcal{P}_L| \leq n-k$. □

A code can have many different Tanner graph representations. However, since all these Tanner graphs represent the same code, the code properties are

not affected when using different Tanner graph representations.

Local group construction: By using a greedy algorithm, we partition the set of the encoded symbols $\mathcal{Y} = \{y_1, ..., y_n\}$ into local groups $\mathcal{Y}_j$'s for $j \in [1, m]$, such that all elements of $\mathcal{Y}_j$ have the same locality r_j, where $r_1 \leq r_2 \leq \cdots \leq r_m$, and $\mathcal{Y}_p \cap \mathcal{Y}_q = \{\}$ for any distinct p and q in $[1, m]$. To this end, at the first step, assume $\mathcal{U}_1 = \{\}$, y_p is a VN with the minimum locality in set $\overline{\mathcal{U}_1} := \mathcal{Y} \backslash \mathcal{U}_1 = \mathcal{Y}$, and Ψ_1 is the locality-defining set for y_p. We set $r_1 = \mathsf{Loc}(y_p)$, $\mathcal{Y}_1 = \Psi_1 \backslash \mathcal{U}_1 = \Psi_1$. At the second step, assume $\mathcal{U}_2 = \mathcal{U}_1 \cup \mathcal{Y}_1 = \mathcal{Y}_1$, an updated y_p is a VN with the minimum locality in set $\overline{\mathcal{U}_2} := \mathcal{Y} \backslash \mathcal{U}_2 = \mathcal{Y} \backslash \mathcal{Y}_1$, and Ψ_2 is the locality-defining set for y_p. We set $r_2 = \mathsf{Loc}(y_p)$, $\mathcal{Y}_2 = \Psi_2 \backslash \mathcal{U}_2 = \Psi_2 \backslash \mathcal{Y}_1$. We continue this procedure m times until all VNs are placed in their corresponding set $\mathcal{Y}_i$, where $i \in [1, m]$ and $m \leq n - k$. The detailed procedure is described in the following algorithm.

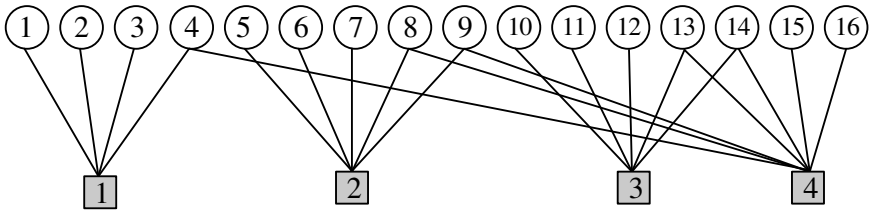


Figure 4.1: For the above locality Tanner graph, there are four local groups $\mathcal{Y}_1 = \{y_1, y_2, y_3, y_4\}$, $\mathcal{Y}_2 = \{y_5, y_6, y_7, y_8, y_9\}$, $\mathcal{Y}_3 = \{y_{10}, y_{11}, y_{12}, y_{13}, y_{14}\}$, $\mathcal{Y}_4 = \{y_{15}, y_{16}\}$. Also, $r = 6$ and $\overline{r} = (4 \times 3 + 10 \times 4 + 2 \times 6)/16 = 4$. Two global CNs are not shown in this figure.

Algorithm 1 Local groups construction

Input: $\mathcal{Y} = \{y_1, ..., y_n\}$

Output: $m, \mathcal{Y}_j, r_j$, for $j \in [1, m]$

Initialization: $j = 1, \mathcal{U}_1 = \{\}$

while $|\mathcal{U}_j| < n$ **do**

- Let Ψ_j be the locality-defining set for an element in $\arg \min_{y_p \in \overline{\mathcal{U}}_j} \{\mathsf{Loc}(y_p)\}$, where $\overline{\mathcal{U}}_j := \mathcal{Y} \setminus \mathcal{U}_j$
- Set $r_j = \mathsf{Loc}(y_p)$
- $\mathcal{Y}_j \leftarrow \Psi_j \setminus \mathcal{U}_j$
- $\mathcal{U}_{j+1} \leftarrow \mathcal{U}_j \bigcup \mathcal{Y}_j$
- $j \leftarrow j + 1$

end

$m \leftarrow j - 1$

Example 4.1. *Here, we show how Algorithm 1 works for the locality Tanner graph shown in Fig. 4.1 which corresponds with an* $(n, k, d) = (16, 10, 5)$ *LRC.*

Observe that the locality Tanner graph of this Figure has 4 VNs 1 to 4 with locality 3, 10 VNs 5 to 14 with locality 4, and 2 VNs 15 to 16 with locality 6. Observe that y_1 *is a VN with the minimum locality in set* $\overline{\mathcal{U}_1} := \mathcal{Y} \backslash \mathcal{U}_1 = \mathcal{Y}$[1]*. At the first step,* $\Psi_1 = \{y_1, y_2, y_3, y_4\}$*, locality-defining set for* y_1*, is selected. We set* $r_1 = \mathsf{Loc}(y_1) = 3$*,* $\mathcal{Y}_1 = \Psi_1 \backslash \mathcal{U}_1 = \{y_1, y_2, y_3, y_4\}$*.*

At the second step, we set $\mathcal{U}_2 = \mathcal{U}_1 \cup \mathcal{Y}_1 = \{y_1, y_2, y_3, y_4\}$*, choose* y_5 *as a VN with the minimum locality in set* $\overline{\mathcal{U}_2} := \mathcal{Y} \backslash \mathcal{U}_2 = \mathcal{Y} \backslash \mathcal{Y}_1 = \{y_5, \cdots, y_{16}\}$*, and define* $\Psi_2 = \{y_5, y_6, y_7, y_8, y_9\}$ *as the locality-defining set for* y_5*. We set* $r_2 = \mathsf{Loc}(y_5) = 4$*,* $\mathcal{Y}_2 = \Psi_2 \backslash \mathcal{U}_2 = \{y_5, y_6, y_7, y_8, y_9\}$*.*

At the third step, we set $\mathcal{U}_3 = \mathcal{U}_2 \cup \mathcal{Y}_2 = \{y_1, \cdots, y_9\}$*, choose* y_{10} *as a VN with the minimum locality in set* $\overline{\mathcal{U}_3} := \mathcal{Y} \backslash \mathcal{U}_3 = \mathcal{Y} \backslash \{y_1, \cdots, y_9\} = \{y_{10}, \cdots, y_{16}\}$*, and define* $\Psi_3 = \{y_{10}, y_{11}, y_{12}, y_{13}, y_{14}\}$ *as the local group corresponding with* y_{10}*. We set* $r_3 = \mathsf{Loc}(y_{10}) = 4$*,* $\mathcal{Y}_3 = \Psi_3 \backslash \mathcal{U}_3 = \{y_{10}, y_{11}, y_{12}, y_{13}, y_{14}\}$*.*

At the forth step, we set $\mathcal{U}_4 = \mathcal{U}_3 \cup \mathcal{Y}_3 = \{y_1, \cdots, y_{14}\}$*, choose* y_{15} *as a VN with the minimum locality in set* $\overline{\mathcal{U}_4} := \mathcal{Y} \backslash \mathcal{U}_4 = \mathcal{Y} \backslash \{y_1, \cdots, y_{14}\} = \{y_{15}, y_{16}\}$*, and define* $\Psi_4 = \{y_4, y_8, y_9, y_{13}, y_{14}, y_{15}, y_{16}\}$ *as the local group corresponding with* y_{15}*. We set* $r_4 = \mathsf{Loc}(y_{15}) = 6$*,* $\mathcal{Y}_4 = \Psi_4 \backslash \mathcal{U}_4 = \{y_{15}, y_{16}\}$*.*

Here, the algorithm terminates after $m = 4$ *steps, and all the 16 VNs are placed in their corresponding local groups* $\mathcal{Y}_i$*, where* $i \in [1, 4]$*.*

Definition 4.1. *(Non-overlapping local CNs). We say that local CNs one to* ζ *are non-overlapping if the set of VNs connected to check nodes do not overlap.*

Example 4.2. *In the locality Tanner graph of Fig. 4.1, local CNs* 1 *to* 3 *are non-overlapping; while local CNs* 1 *to* 4 *are overlapping.*

The following Lemma shows how Tanner graph of an (n, k) linear block code is related to the minimum distance of the code (d).

[1]Noting that y_2, y_3, and y_4 have the minimum locality as well, choosing either of y_1 to y_4 does not affect the performance of the Algorithm.

Lemma 4.2. *For an (n, k) linear block code $\mathcal{C}$ with Tanner graph $\mathcal{T}$, let the non-zero elements of parity-check matrix $\mathbf{H}$ corresponding with $\mathcal{T}$ be randomly selected from a sufficiently large finite field. Then, $\mathcal{C}$ has minimum distance d iff every γ CNs of $\mathcal{T}$ cover at least $\gamma + k$ VNs for every $\gamma \in [J,\ n-k]$, where $J := n - k - d + 2$.*

Proof. By fixing the Tanner graph of an (n, k, d) code, the zero elements of its parity check matrix $\mathbf{H}_{\mathbb{F}_q}^{(n-k)\times n}$ are fixed. Here, we assume that the non-zero elements of $\mathbf{H}$ are randomly selected from a sufficiently large finite field. This will assure that the code will have the maximum possible minimum distance among all the codes whose zero elements of their parity check matrix are fixed by the given Tanner graph.

Necessary condition (assumption: the minimum distance is d): By contradiction, assume that there is a subset of CNs of $\mathcal{T}$ with cardinality γ covering at most $\gamma + k - 1$ VNs, where $\gamma \geq n - k - d + 2$. Suppose that the $n-(\gamma+k-1) \in [1, d-1]$ VNs not covered by the mentioned γ CNs are erased. Observe that among all $\gamma + k - 1$ VNs covered by the mentioned γ CNs, there are at most $(\gamma + k - 1) - \gamma = k - 1$ independent VNs. This implies that $\mathcal{C}$ cannot recover k information symbols for the assumed $n-(\gamma+k-1) \in [1, d-1]$ erasures. This contradicts the fact that an erasure code with minimum distance d can recover k information symbols for any up to $d - 1$ symbol erasures.

Sufficient condition (assumption: every γ CNs of $\mathcal{T}$ cover at least $(\gamma + k)$ VNs for every $\gamma \in [J,\ n-k]$): An erasure code with minimum distance d can recover any up to $d - 1$ erasures. Let us define β as $\beta := (n - k) - \gamma + 1$. Since $\gamma \in [n-k-d+2,\ n-k]$, we have $\beta \in [1, d-1]$. Now, we show that any β VNs are covered by at least β CNs. This implies that any $\beta \in [1, d-1]$ erasures can be recovered using β independent equations associated with their corresponding CNs. By contradiction, assume that there is a subset of VNs with cardinality β covered by at most $\beta - 1$ CNs. Then, the remaining $(n - k) - (\beta - 1) = \gamma$ CNs cover at most $n - \beta = k + \gamma - 1$ VNs. This contradicts the assumption

that any γ CNs cover at least $k+\gamma$ VNs. $\square$

Remark 4.1. *As we will see, all LRCs discussed in this chapter have the following two properties: i) each local CN is connected to at least one VN which is not connected to other local CNs, and ii) each global CN is linked to all n VNs. When these conditions are satisfied, Lemma 4.2 results in the following corollary which allows us to verify that the minimum distance of a code is d by showing that every J local CNs cover at least $J+k$ VNs.*

Corollary 4.1. *For an LRC with Tanner graph $\mathcal{T}$, assume that i) each local CN is connected to at least one VN which is not connected to other local CNs, and ii) each global CN is linked to all n VNs. Then, the sufficient condition of Lemma 4.2 is equivalent to the following one: If every J local CNs cover $J+k$ VNs, then the minimum distance of the code is d.*

Proof. Without loss of generality, consider the first J local CNs that cover at least $J+k$ VNs. By adding $\gamma-J$ arbitrary local CNs to these J local CNs, at least $\gamma-J$ new VNs are covered, where $\gamma \in [J, n-k]$. This is true because each local CN is connected to at least one VN which is not connected to other local CNs. Also, each global CN is connected to all VNs. Hence, every γ CNs cover at least $\gamma+k$ VNs. $\square$

As we will see later, Lemma 4.2 is a strong tool which will help us obtain a bound on $\overline{r}$. In fact, the famous bound of (2.2) on the maximum locality also can easily be obtained by Lemma 4.2.

Remark 4.2. *(A simple proof for the well-known bound (2.2) using Lemma 4.2) For an (n,k,d,r) LRC, assume that the number of local CNs is m, where $m \in \{2,\cdots,J,J+1,\cdots,n-k\}$. Now, we consider two cases as follows.*
Case (i) $m \in [J,n]$: In this case, by Lemma 4.2, every γ local CNs cover at least $\gamma+k$ VNs. We have

$$r+1 \geq \frac{\gamma+k}{\gamma} \geq \frac{J+k}{J}, \quad m \in [J, n-k]. \tag{4.1}$$

Case (ii) $m \in [2, J-1]$: In this case, all the m local CNs cover all the n VNs. We have

$$r+1 \geq \frac{n}{m} \geq \frac{n}{J-1} > \frac{J+k}{J}, \quad m \in [2, J-1], \tag{4.2}$$

where the last equality holds because $n \geq J+k = n-(d-2)$. From (4.1), we have

$$r \geq \frac{k}{J} \Rightarrow n-k-d+2 \geq \frac{k}{r} \Rightarrow d \leq n-k-\frac{k}{r}+2.$$

Since d is an integer, $d \leq n-k-\left\lceil \frac{k}{r} \right\rceil + 2$.

The following Lemma is used to partition a subset of the encoded symbols, say $\mathcal{A}$, with cardinality A into ζ non-overlapping local groups such that the average locality of the encoded symbols in $\mathcal{A}$ is minimized.

Lemma 4.3. *Let z_j, $j \in [1, \zeta]$ be integers, and $\sum_{j=1}^{\zeta} z_j = A$. Then,*

$$\sum_{j=1}^{\zeta} z_j^2 \geq (\zeta - a)\left\lfloor \frac{A}{\zeta} \right\rfloor^2 + a\left\lceil \frac{A}{\zeta} \right\rceil^2,$$

where $a = A + \zeta - \zeta\lceil \frac{A}{\zeta} \rceil$.

Proof. Subject to $\sum_{j=1}^{\zeta} z_j = A$ and z_j's are integer, the sum $\sum_{j=1}^{\zeta} z_j^2$ is minimized if $|z_l - z_m| \leq 1$ for every pair z_l and z_m. Because otherwise if $z_l > z_m + 1$, then $\sum_{j=1}^{\zeta} z_j^2$ can be reduced by setting z_l to $z_l - 1$ and z_m to $z_m + 1$. This is true since

$$z_l > z_m + 1 \Leftrightarrow z_l^2 + z_m^2 \geq (z_l - 1)^2 + (z_m + 1)^2.$$

Consequently, $\sum_{j=1}^{\zeta} z_j^2$ is minimized if for every j, $z_j = z$ or $z_j = z - 1$ for some integer z. The number z is unique and it is a function of A and ζ. Now, assume that among ζ integers z_j, a integers are z and the rest $\zeta - a$ integers are $z-1$. Therefore, $az + (\zeta - a)(z-1) = A$; equivalently, $a = A + \zeta(1-z)$. Hence, $\lceil \frac{a}{\zeta} \rceil = \lceil \frac{A}{\zeta} \rceil + 1 - z$. Therefore, $z = \lceil \frac{A}{\zeta} \rceil$ because $a \leq \zeta$. □

Lemma 4.3 implies that the minimum of sum squares of some integers ($\min \sum_{j=1}^{\zeta} z_j^2$) with a constant summation ($\sum_{j=1}^{\zeta} z_j = A$) is obtained when

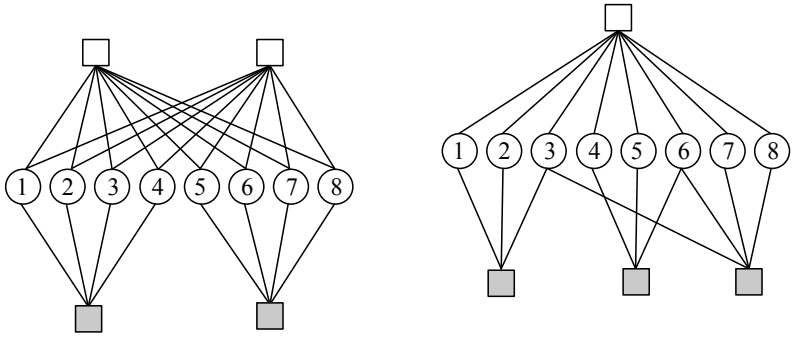
the integers (z_j's) are distributed as uniformly as possible. We will use this Lemma later to distribute VNs in distinct local groups in order to minimize the average locality of all VNs.

4.2 Results on the Average Locality of All Symbols

In [57], as the first step towards establishing a bound on $\overline{r}$ and motivated by the $(n, k, d) = (16, 10, 5)$ LRC used in Facebook HDFS-RAID [6], we proved that the average locality $\overline{r}$ of any $(n, k) = (16, 10)$ LRC with minimum distance $d = 5$ is at least 3.875. We also gave the construction of an LRC that achieves the bound $\overline{r} = 3.875$. In this section, we generalize our work in [57]. More specifically, we establish a lower bound on $\overline{r}$ of arbitrary (n, k, d) LRCs (Theorem 4.1). Furthermore, we obtain a tight lower bound on $\overline{r}$ for a practical case where $R = \frac{k}{n} > (1 - \frac{1}{\sqrt{n}})^2$ (Theorem 4.2). Finally, we design three classes of LRCs that achieve the obtained bounds on $\overline{r}$ (Section 4.2.2). Note that the improvement achieved by our proposed LRCs comes without sacrificing important code parameters such as rate ($\frac{k}{n}$) and minimum distance d. The following simple example shows the effectiveness of our solution.

Example 4.3. *The existing solution: For an $(n, k, r) = (8, 4, 3)$ LRC, bound in (2.2) results in $d \leq 8 - 4 - \lceil \frac{4}{3} \rceil + 2 = 4$. In this case, since $(r + 1) \mid n$, an optimal LRC with $d = 4$ can be constructed whose Tanner graph is depicted in Fig. 4.2a. Here, $\mathsf{Loc}(y_i) = r = \overline{r} = 3$ for $i \in [1, 8]$.*

Our proposed solution: Fig. 4.2b shows Tanner graph associated with our proposed LRC in this section. For this LRC, $\mathsf{Loc}(y_i) = 2$ for $i \in [1, 6]$, and $\mathsf{Loc}(y_i) = 3$ otherwise. Hence, $r = 3$ and $\overline{r} = (6 \times 2 + 2 \times 6)/8 = 2.25$. In other words, the average locality is improved by 25% without changing d and the code rate k/n.



(a) Tanner graph of an optimal $(n, k, d, r, \overline{r}) = (8, 4, 4, 3, 3)$ LRC.

(b) Tanner graph of our proposed $(n, k, d, r, \overline{r}) = (8, 4, 4, 3, 2.25)$ LRC.

Figure 4.2: Tanner graph associated with two $(n, k, d) = (8, 4, 4)$ LRCs. Here, local check nodes, which depicts the locality of all the encoded symbols, are shown by gray squares; and the non-local check nodes are shown by white squares. The average locality of the optimal LRC is improved by 25% using our proposed LRC.

In this section, by using a novel approach, we design LRCs with improved average locality. In other words, the amount of costly repair bandwidth, disk I/O, number of participating nodes during repair process of a lost/erased block of data, and MTTDL are all improved by using our proposed coding schemes. Considering the recent interest in using efficient erasure codes in real-world data centers (e.g. in Google File System [10], Windows Azure Storage [7], and Facebook HDFS-RAID [6]), our results can be of great importance from a practical point of view.

4.2.1 Lower Bounds on $\overline{r}$

First, in Section 4.2.1, we derive a lower bound on $\overline{r}$ that holds for any (n, k, d) LRCs. We compare this bound to the one on maximum locality r. In Section 4.2.1, we obtain a tight lower bound on $\overline{r}$ for (n, k, d) LRCs which satisfy the following constraint on the code rate $R = \frac{k}{n} > \left(1 - \frac{1}{\sqrt{n}}\right)^2$. In Section 4.2.2, we present three classes of LRCs that achieve the obtained bounds presented in this section.

A Lower Bound on $\overline{r}$ for Arbitrary (n, k, d) LRC

First, let us reform the bound in (2.2) to obtain a lower bound on r. From (2.2), we have

$$d \leq n - k - \left\lceil \frac{k}{r} \right\rceil + 2 \leq n - k - \frac{k}{r} + 2 \Rightarrow r \geq \frac{k}{n-k-d+2} = \frac{k}{J}.$$

Since r is an integer, the lower bound of the maximum locality (r) can be presented as

$$r \geq \left\lceil \frac{k}{J} \right\rceil. \tag{4.3}$$

Theorem 4.1. *For any (n, k, d) LRC with $J = n - k - d + 2$, the average locality of all symbols ($\overline{r}$) is bounded as follows*

$$\overline{r} \geq \left\lceil \frac{k}{J} \right\rceil \Big(1 - \frac{J\left\lceil \frac{k}{J} \right\rceil - k}{n}\Big). \tag{4.4}$$

Proof. Please see Appendix B.1. □

Remark 4.3. *Considering (4.4), the bound on $\overline{r}$ can be represented as follows*

$$\overline{r} \geq \alpha \left\lfloor \frac{k}{J} \right\rfloor + (1-\alpha)\left\lceil \frac{k}{J} \right\rceil, \tag{4.5}$$

where $\alpha := \frac{m_1}{n} = \frac{1}{n}(J\lceil \frac{k}{J} \rceil - k)(\lfloor \frac{k}{J} \rfloor + 1)$.

Remark 4.4. *By subtracting the right hand side of (4.4) from that of (4.3), the gap between the bounds on r and $\overline{r}$ is obtained as $\alpha = \frac{m_1}{n}$ which is less than one because $m_1 < n$. Therefore, compared to codes with optimum maximum locality, the saving in the average locality is at most one symbol. This gap is maximized when $J \mid (k-1)$. To see this, observe that*

$$\alpha = \frac{1}{n}\left\lceil \frac{k}{J} \right\rceil \Big(J\left\lceil \frac{k}{J} \right\rceil - k\Big) = \begin{cases} \frac{1}{n}(J-\beta)\lceil \frac{k}{J} \rceil, & J \nmid k \\ 0, & J \mid k \end{cases} \tag{4.6}$$

where $\beta = k \mod J$, *and* $\beta \neq 0$. *Hence, (4.6) achieves its maximum when* $\beta = 1$, *equivalently, when* $J \mid (k-1)$. *In Section 4.2.2, we show that for sufficiently large values of d, the bound in Theorem 4.1 is achievable when* $J \mid (k-1)$.

An Achievable Lower Bound on $\overline{r}$ of LRCs with $R > \left(1 - \frac{1}{\sqrt{n}}\right)^2$

In the previous section, in order to obtain a lower bound on $\overline{r}$, we assumed that the first J local groups of (n, k, d) LRCs cover $J + k$ VNs. Also, we assumed that locality of the remaining $n - (J + k) = d - 2$ is equal to maximum locality of the first $J + k$ VNs. The obtained bound in Theorem 4.1 is not always tight. In this section, we obtain a tight lower bound on $\overline{r}$ assuming that

$$R = \frac{k}{n} > \left(1 - \frac{1}{\sqrt{n}}\right)^2. \tag{4.7}$$

It is notable that linear block codes currently in use—e.g. the $(16, 10, 5)$ code used in Facebook HDFS-RAID, the $(16, 12, 4)$ code used in Windows Azure Storage, and the $(9, 6, 4)$ code in Google File System—satisfy the condition presented in (4.7) as depicted in Fig. 4.3. We also note that for a typical code length $n < 25$ the condition (4.7) on rate is $R > 0.64$ which is very practical.

Theorem 4.2. *For any* (n, k, d) *LRC with* $R = \frac{k}{n} > \left(1 - \frac{1}{\sqrt{n}}\right)^2$, *the average locality of all symbols* ($\overline{r}$) *is bounded as follows*

$$\overline{r} \geq \frac{\min\limits_{\theta \in [0, d-2]} \left\{ (J - a_\theta) \left\lfloor \frac{n-\theta}{J} \right\rfloor^2 + a_\theta \left\lceil \frac{n-\theta}{J} \right\rceil^2 + (n - dJ + 2J)\theta \right\}}{n} - 1, \tag{4.8}$$

where $J = n - k - d + 2$ *and* $a_\theta = n - \theta + J - J\left\lceil \frac{n-\theta}{J} \right\rceil$.

Proof. Here, we present the sketch of the proof. Please find the detailed proof in Appendix B.2.

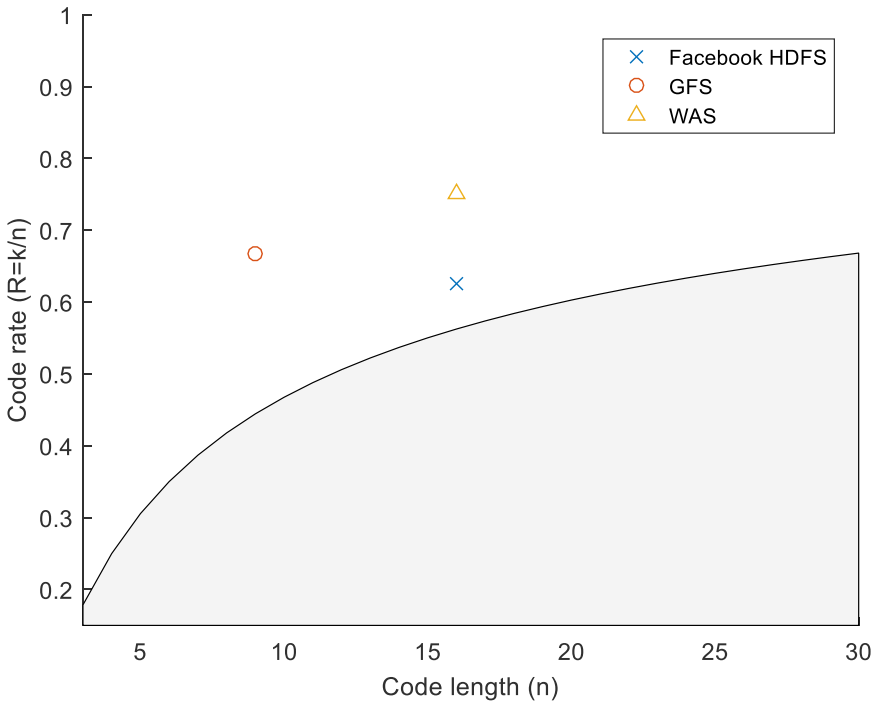


Figure 4.3: Graph of the function $R = (1 - \frac{1}{\sqrt{n}})^2$ over the interval $n \in [3, 30]$. The bound in Theorem 4.2 and codes designed in Section 4.2.2 are valid in the white area over the curve. As this figure shows all the practical cases currently in use are properly placed in this white area.

To begin with, we construct m local groups using Algorithm 1. Note that the maximum locality of an (n, k) linear block code is k, hence, $m > 1$. Furthermore, by Lemma 4.1, $m \leq n - k$. Therefore,

$$n\,\overline{r} = \sum_{i=1}^{n} \mathit{Loc}(y_i) = \sum_{j=1}^{m} |\mathcal{Y}_j| r_j, \quad m \in [2, n-k]. \tag{4.9}$$

Then, we verify that the minimum average locality is achieved when m is either J or $J+1$. By Lemma 4.2, in order to achieve minimum distance d, the first J local groups must cover at least $J+k = n-(d-2)$ VNs and at most $n-1$ VNs [2]. Among all possible $J+1$ local groups, we assume that the first J local groups

[2]Note that the case that the first J local groups cover n VNs is equivalent to the first

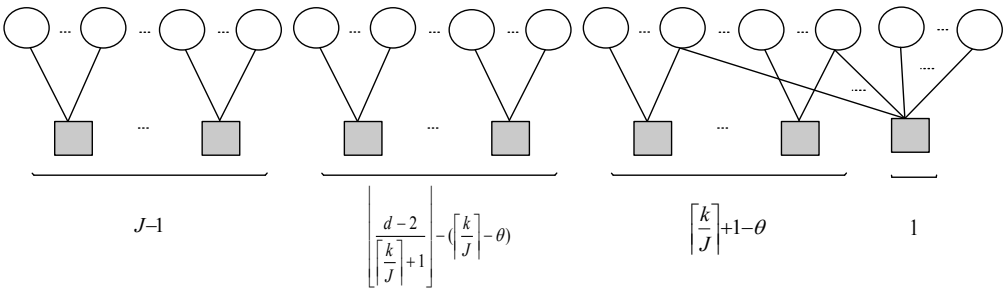


Figure 4.4: Locality Tanner graph of an (n, k, d) LRC with $J \mid (k-1)$ and $\left\lfloor \frac{d-2}{\lceil \frac{k}{J} \rceil + 1} \right\rfloor \geq \left\lceil \frac{k}{J} \right\rceil - \theta$. There are $(J-1)$ non-overlapping local groups with cardinality $(\lfloor \frac{k}{J} \rfloor + 1)$; $\left\lfloor \frac{d-2}{\lceil \frac{k}{J} \rceil + 1} \right\rfloor - (\left\lceil \frac{k}{J} \right\rceil - \theta)$ non-overlapping local groups with cardinality $(\left\lceil \frac{k}{J} \right\rceil + 1)$; $(\left\lceil \frac{k}{J} \right\rceil + 1 - \theta)$ overlapping local groups with cardinality $(\left\lceil \frac{k}{J} \right\rceil + 1)$; and one overlapping local group which has $(\left\lceil \frac{k}{J} \right\rceil + 1 - \theta)$ VNs in common with $(\left\lceil \frac{k}{J} \right\rceil + 1 - \theta)$ distinct local groups. Global CNs are not shown in this figure.

and the last local group cover $n-\theta$ and θ VNs, respectively, where $\theta \in [1, d-2]$. Then, we obtain a lower bound on $\overline{r}$ according to this assumption. Finally, the lower bound on $\overline{r}$ is obtained by taking the minimum of two bounds associated with the considered cases. □

The bound in Theorem 4.1 is not always achievable; while the bound in Theorem 4.2 is always achievable. Sometimes the two bound can be equal. For example, when $J \mid (k-1)$ and $\theta = d - 2 = \lceil \frac{k}{J} \rceil$, the bounds obtained in Theorems 4.1 and 4.2 become the same.

In Section 4.2.2, we design a class of LRCs that achieve the bound on $\overline{r}$ obtained in Theorem 4.2.

case with $m = J$. This case is considered in Theorem 4.2 by setting $\theta = 0$.

4.2.2 Achieving the Bounds: $\overline{r}$-Optimal LRCs

In this section, we design LRCs that achieve the lower bounds on $\overline{r}$ obtained in Theorems 4.1 and 4.2. We call such codes $\overline{r}$-optimal LRCs. In Sections 4.2.2 and 4.2.2, by following the equality conditions in the bound of Theorem 4.1, we construct two classes of $\overline{r}$-optimal LRCs that achieve the general bound in Theorem 4.1. Furthermore, in Section 4.2.2, we construct a class of LRCs that achieve the bound in Theorem 4.2.

The First Class of $\overline{r}$-Optimal LRCs

Suppose the parameters n, k and d are such that $(\lceil \frac{k}{J} \rceil + 1) \mid (d-2)$. The first $k+J$ VNs are partitioned into $J\lceil \frac{k}{J} \rceil - k$ and $J + k - J\lceil \frac{k}{J} \rceil$ local groups with cardinality $\lfloor \frac{k}{J} \rfloor + 1$ and $\lceil \frac{k}{J} \rceil + 1$, respectively. Then, $n - (J+k) = d - 2$ VNs not covered by the first J local CNs are partitioned into $\frac{d-2}{\lceil \frac{k}{J} \rceil + 1}$ local groups with cardinality $\lceil \frac{k}{J} \rceil + 1$. Hence, there are exactly $\frac{m_1}{\lfloor \frac{k}{J} \rfloor + 1} = J\lceil \frac{k}{J} \rceil - k$ and $\frac{n-m_1}{\lceil \frac{k}{J} \rceil + 1}$ local groups with cardinality $\lfloor \frac{k}{J} \rfloor + 1$ and $\lceil \frac{k}{J} \rceil + 1$, respectively. Note that for this class of $\overline{r}$-optimal LRCs, the constraint on d expressed in Corollary 4.1 is satisfied and the bound in Theorem 4.1 is achieved.

Example 4.4. *Considering Fig. 4.5, we present a numerical example of this class of LRCs. Let* $(n,k,d) = (14,5,8)$*, hence* $J = n-k-d+2 = 3$*. Observe that* $3 = (\lceil \frac{k}{J} \rceil + 1) \mid (d-2) = 6$ *is satisfied. The first* $k+J = 8$ *VNs are partitioned into* $J\lceil \frac{k}{J} \rceil - k = 1$ *and* $J+k-J\lceil \frac{k}{J} \rceil = 2$ *local groups with cardinality* $\lfloor \frac{k}{J} \rfloor + 1 = 2$ *and* $\lceil \frac{k}{J} \rceil + 1 = 3$*, respectively. Then,* $n-(J+k) = d-2 = 6$ *VNs not covered by the first* $J = 3$ *local CNs are partitioned into* $\frac{d-2}{\lceil \frac{k}{J} \rceil + 1} = 2$ *local groups with cardinality* $\lceil \frac{k}{J} \rceil + 1 = 3$*. Hence, there are exactly* $\frac{m_1}{\lfloor \frac{k}{J} \rfloor + 1} = J\lceil \frac{k}{J} \rceil - k = 1$ *and* $\frac{n-m_1}{\lceil \frac{k}{J} \rceil + 1} = 4$ *local groups with cardinality* $\lfloor \frac{k}{J} \rfloor + 1 = 2$ *and* $\lceil \frac{k}{J} \rceil + 1 = 3$*, respectively. Observe that for this* $\overline{r}$*-optimal LRC, the constraint on* d *expressed in Corollary 4.1 is satisfied and the bound in Theorem 4.1 is achieved as* $\overline{r} = \frac{2\times 1 + 12 \times 2}{14}$ *and* $\lceil \frac{k}{J} \rceil (1 - \frac{J\lceil \frac{k}{J} \rceil - k}{n}) = \frac{26}{14}$.

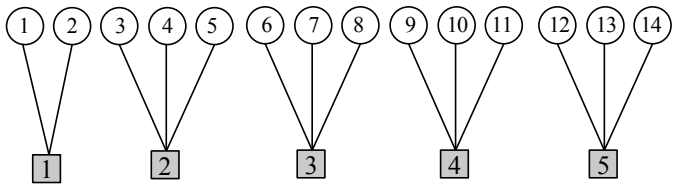


Figure 4.5: The proposed $(n, k, d) = (14, 5, 8)$ $\overline{r}$-optimal LRC. Here, the four global CNs are not depicted.

The Second Class of $\overline{r}$-Optimal LRCs

Suppose the parameters n, k and d are such that $J \mid (k-1)$ and

$$\left\lfloor \frac{d-2}{\lceil \frac{k}{J} \rceil + 1} \right\rfloor \geq \left\lceil \frac{k}{J} \right\rceil - \theta, \tag{4.10}$$

where

$$\theta = (d-2) \mod \left(\left\lceil \frac{k}{J} \right\rceil + 1 \right). \tag{4.11}$$

Fig. 4.4 shows the Tanner graph corresponding to our second class of $\overline{r}$-optimal LRCs ($\mathcal{C}_{\overline{r}LRC_2}$). In this graph, all the local groups, except $\lceil \frac{k}{J} \rceil + 2 - \theta$, are non-overlapping. Among the non-overlapping local groups, $J-1$ groups have cardinality $\lfloor \frac{k}{J} \rfloor + 1$, while the remaining have cardinality $\lceil \frac{k}{J} \rceil + 1$. Among the overlapping local groups, $\left\lceil \frac{k}{J} \right\rceil + 1 - \theta$ groups have cardinality $\lceil \frac{k}{J} \rceil + 1$, while the remaining one has cardinality θ. This last overlapping local group shares exactly one VN with $(\left\lceil \frac{k}{J} \right\rceil + 1 - \theta)$ overlapping local groups, which is possible by (4.10).

Note that each VN has locality of $\lfloor \frac{k}{J} \rfloor$ or $\lceil \frac{k}{J} \rceil$, and the average locality is optimum by Theorem 4.1. Also, it can be verified that every J local CNs cover at least $J + k$ VNs, thus, by Corollary 4.1, the minimum distance of $\mathcal{C}_{\overline{r}LRC_2}$ is at least d.

Example 4.5. *Here, we present a numerical example of this class of LRCs. Let* $(n, k, d) = (11, 4, 6)$ *and* $J = n - k - d + 2 = 3$. *Observe that* $3 = J \mid (k-1) = 6$

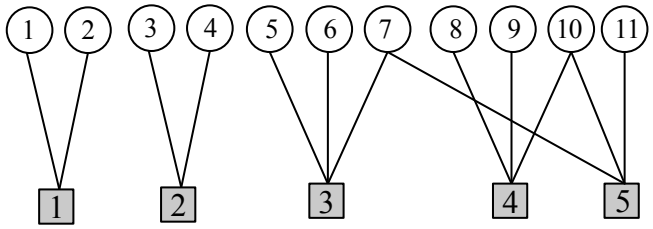


Figure 4.6: The proposed $(n, k, d) = (11, 4, 6)$ $\overline{r}$-optimal LRC. Here, the two global CNs are not depicted.

and with $\theta = (d-2) \mod (\lceil \frac{k}{J} \rceil + 1) = 1$, *condition* $1 = \lfloor \frac{d-2}{\lceil \frac{k}{J} \rceil + 1} \rfloor \geq \lceil \frac{k}{J} \rceil - \theta = 1$ *is satisfied. Fig. 4.6 represent the locality Tanner graph of this* $\overline{r}$*-optimal code. In this graph, there are* $(J-1) = 2$ *non-overlapping local groups with cardinality* $(\lfloor \frac{k}{J} \rfloor + 1) = 2$*;* $\lfloor \frac{d-2}{\lceil \frac{k}{J} \rceil + 1} \rfloor - (\lceil \frac{k}{J} \rceil - \theta) = 1 - 1 = 0$ *non-overlapping local groups with cardinality* $(\lceil \frac{k}{J} \rceil + 1) = 3$*;* $(\lceil \frac{k}{J} \rceil + 1 - \theta) = 2$ *overlapping local groups with cardinality* $(\lceil \frac{k}{J} \rceil + 1) = 3$*; and one overlapping local group which has* $(\lceil \frac{k}{J} \rceil + 1 - \theta) = 2$ *VNs in common with* $(\lceil \frac{k}{J} \rceil + 1 - \theta) = 2$ *distinct local groups. Note that each VN has locality of* $\lfloor \frac{k}{J} \rfloor = 2$ *or* $\lceil \frac{k}{J} \rceil = 3$*, and the average locality is optimum by Theorem 4.1:* $\frac{4 \times 1 + 7 \times 2}{11} = \overline{r} \geq \lceil \frac{k}{J} \rceil (1 - \frac{J \lceil \frac{k}{J} \rceil - k}{n}) = \frac{18}{11}$*. Also, observe that every* $J = 3$ *local CNs cover at least* $J + k = 7$ *VNs, thus, by Corollary 4.1, the minimum distance of* $\mathcal{C}_{\overline{r}LRC_2}$ *is at least* $d = 6$*.*

The Third Class of $\overline{r}$-Optimal LRCs

The first two classes of LRCs presented in Sections 4.2.2 and 4.2.2 achieve the bound in Theorem 4.1, which is not always achievable. In Theorem 4.2, we establish another bound on $\overline{r}$ corresponding with practical cases—where the code rate satisfies $R > (1 - \frac{1}{\sqrt{n}})^2$—which is always achievable. Here, we present a class of codes that achieve this bound.

Suppose the parameters n and k are such that $R > (1 - \frac{1}{\sqrt{n}})^2$. Then, the third class of our proposed $\overline{r}$-optimal LRCs ($\mathcal{C}_{\overline{r}LRC_3}$) achieving the bound in Theorem 4.2 is constructed through the following steps (Fig. 4.7).

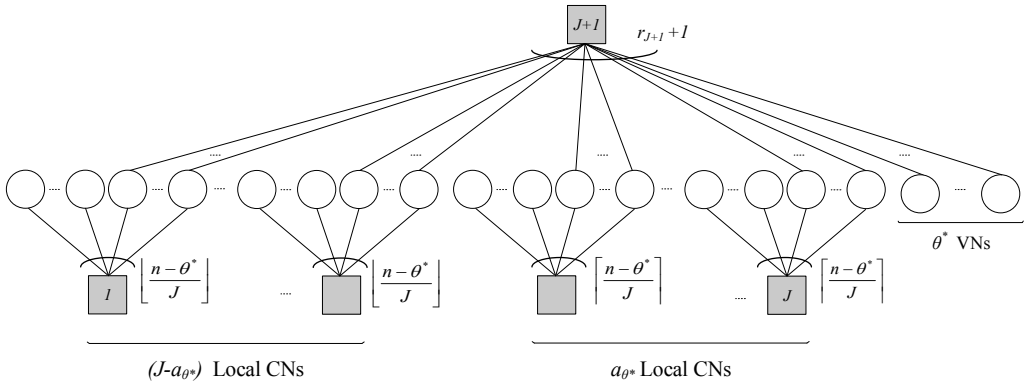


Figure 4.7: Locality Tanner graph of $\mathcal{C}_{\overline{r}LRC_3}$ with $J+1$ local CNs. Global CNs are not shown in this figure.

Step 1. Obtain $\theta \in [0, d-2]$, denoted θ^*, that minimizes (4.8) in Theorem 4.2.

Step 2. Partition the set of n VNs $\mathcal{Y} = \{y_1, \cdots, y_n\}$ into two subsets $\mathcal{Y}_A$ and $\mathcal{Y}_B$ with cardinality $n-\theta^*$ and θ^*, respectively.

Step 3. Partition $n-\theta^*$ VNs associated with $\mathcal{Y}_A$ into $J = n-k-d+2$ local groups as follows: $J - a_{\theta^*}$ local groups with cardinality $\lfloor \frac{n-\theta^*}{J} \rfloor$; and a_{θ^*} local groups with cardinality $\lceil \frac{n-\theta^*}{J} \rceil$, where $a_{\theta^*} = n - \theta^* + J - J\left\lceil \frac{n-\theta^*}{J} \right\rceil$. Therefore, there are $(J-a_{\theta^*})\lfloor \frac{n-\theta^*}{J} \rfloor$ VNs with locality $\left\lfloor \frac{n-\theta^*}{J} \right\rfloor - 1$ and $a_{\theta^*}\lceil \frac{n-\theta^*}{J} \rceil$ VNs with locality $\left\lceil \frac{n-\theta^*}{J} \right\rceil - 1$.

Step 4. If $\theta^* \neq 0$, construct $(J+1)$-th local group with the following VNs: *i)* $\lfloor \frac{n-\theta^*}{J} \rfloor - d + 2$ VNs from each of local groups 1 to $J - a_{\theta^*}$; *ii)* $\lceil \frac{n-\theta^*}{J} \rceil - d + 2$ VNs from each of local groups $J - a_{\theta^*} + 1$ to J; and *iii)* all the θ^* VNs associated with $\mathcal{Y}_B$. Hence, locality of the last local group is

$$r_{J+1} = (J - a_{\theta^*})\Big(\Big\lfloor \frac{n-\theta^*}{J} \Big\rfloor - d + 2\Big) + a_{\theta^*}\Big(\Big\lceil \frac{n-\theta^*}{J} \Big\rceil - d + 2\Big) + \theta^* - 1.$$

By manipulation, we have

$$r_{J+1} = n - J(d-2) - 1. \tag{4.12}$$

Step 5. Observe that there are $J + b_{\theta^*}$ local CNs and $n - k - (J + b_{\theta^*})$ global CNs, where $b_{\theta^*} = 0$ if $\theta^* = 0$ and $b_{\theta^*} = 1$ otherwise. Connect each of the $n - k - J - b_{\theta^*} = d - 2 - b_{\theta^*}$ to all n VNs.

Remark 4.5. *It can be verified that every J local CNs cover at least $J + k$ VNs, thus, by Corollary 4.1, the minimum distance of $\mathcal{C}_{\overline{r}LRC_3}$ is at least d.*

Example 4.6. *In this example, we construct an $(n, k, d) = (8, 4, 4)$ LRC by following the aforementioned steps. Here, $J = n - k - d + 2 = 2$ and θ minimizing (4.8) is $\theta^* = 2$. All the eight VNs are partitioned into two sets $\mathcal{Y}_A = \{y_1, \cdots, y_6\}$ and $\mathcal{Y}_B = \{y_7, y_8\}$. Since $J \mid (n - \theta^*)$, there are $J = 2$ local groups with cardinality $\frac{n-\theta^*}{J} = 3$. Since $\theta^* \neq 0$, the third local group has to be constructed by $\lceil \frac{n-\theta^*}{J} \rceil - d + 2 = \lceil \frac{8-2}{2} \rceil - 4 + 2 = 1$ VN from each of the first two local groups and $\theta^* = 2$ VNs associated with $\mathcal{Y}_B$. Let us form the third local group by VNs y_3, y_6, y_7, and y_8. There is $n - k - J - b_{\theta^*} = 8 - 4 - 2 - 1 = 1$ global check node which is connected to all 8 VNs. Fig. 4.2b depicts Tanner graph associated with LRC of this example.*

Proposition 4.1. *$\mathcal{C}_{\overline{r}LRC_3}$ satisfies the bound on $\overline{r}$ in Theorem 4.2 with equality.*

Proof. The first $n - \theta^*$ VNs of the code are partitioned into $(J - a_{\theta^*})\lfloor \frac{n-\theta^*}{J} \rfloor$ and $a_{\theta^*}\lceil \frac{n-\theta^*}{J} \rceil$ VNs with locality $\left\lfloor \frac{n-\theta^*}{J} \right\rfloor - 1$ and $\left\lceil \frac{n-\theta^*}{J} \right\rceil - 1$, respectively. Also, locality of the last θ^* VNs is $n - J(d-2) - 1$. Hence, for the average locality of $\mathcal{C}_{\overline{r}LRC_3}$, denoted $\overline{r}_{\mathcal{C}_3}$, we have

$$\begin{aligned} n\overline{r}_{\mathcal{C}_3} &= (J - a_{\theta^*})\left\lfloor \frac{n-\theta^*}{J} \right\rfloor \left(\left\lfloor \frac{n-\theta^*}{J} \right\rfloor - 1 \right) \\ &\quad + a_{\theta^*}\left\lceil \frac{n-\theta^*}{J} \right\rceil \left(\left\lceil \frac{n-\theta^*}{J} \right\rceil - 1 \right) + \theta^*(n - J(d-2) - 1) \\ &= (J - a_\theta^*)\left\lfloor \frac{n-\theta^*}{J} \right\rfloor^2 + a_\theta^* \left\lceil \frac{n-\theta^*}{J} \right\rceil^2 \\ &\quad - \left(a_{\theta^*}\left\lceil \frac{n-\theta^*}{J} \right\rceil + (J - a_{\theta^*})\left\lfloor \frac{n-\theta^*}{J} \right\rfloor \right) \\ &\quad + \theta^*(n - J(d-2) - 1). \end{aligned}$$

By manipulation, we have

$$a_{\theta^*}\left\lceil\frac{n-\theta^*}{J}\right\rceil+(J-a_{\theta^*})\left\lfloor\frac{n-\theta^*}{J}\right\rfloor=n-\theta^*.$$

Thus,

$$n\overline{r}_{\mathcal{C}_3}=(J-a_{\theta^*})\left\lfloor\frac{n-\theta^*}{J}\right\rfloor^2+a_{\theta^*}\left\lceil\frac{n-\theta^*}{J}\right\rceil^2+\theta^*(n-J(d-2))-n, \quad (4.13)$$

which is equivalent to the minimum of $\overline{r}$ in Theorem 4.2. □

4.2.3 Improvement of the LRC Used in Facebook HDFS-RAID

In this section, we show how our proposed approach can improve the locality of the $(n,k,d,\overline{r})=(16,10,5,5)$ LRC used in Facebook HDFS-RAID [6], denoted $\mathcal{C}_F$, without sacrificing its crucial parameters, namely the code rate (R) and the code minimum distance (d). We show that by using our proposed LRC, the average locality of $\mathcal{C}_F$ is improved by 22.5%.

For $\mathcal{C}_F$, we have $R=0.625>(1-\frac{1}{\sqrt{n}})^2=0.56$. Hence, the code construction described in Section 4.2.2 can be used. For an $(n,k,d)=(16,10,5)$ LRC, it is verified by Theorem 4.2 that $\overline{r}\geq 3.875$, which is obtained for $\theta=d-2=3$. The Tanner graph of our proposed LRC, denoted $\mathcal{C}_0$, is presented in Fig 4.8. By using the Tanner graph, the parity check matrix of $\mathcal{C}_0$, denoted $\mathbf{H}_0$, is constructed. The non-zero elements of $\mathbf{H}_0\in\mathbb{F}_{2^8}^{6\times 16}$ are numerically selected from the finite field $\mathbb{F}_{2^8}$ formed by the primitive polynomial $x^8+x^4+x^3+x^2+1$ such that every $d-1=4$ columns of $\mathbf{H}_0$ are linearly independent. Then, by multiplying the inverse of a full-rank 6×6 sub-matrix of $\mathbf{H}_0$ from the left by $\mathbf{H}_0$, the systematic form of $\mathbf{H}_0$ is obtained. By using the obtained systematic form, the systematic form of the generator matrix of our proposed $(n,k,d,\overline{r})=(16,10,5,3.875)$ LRC, denoted $\mathbf{G}_0\in\mathbb{F}_{2^8}^{10\times 16}$, can be explicitly

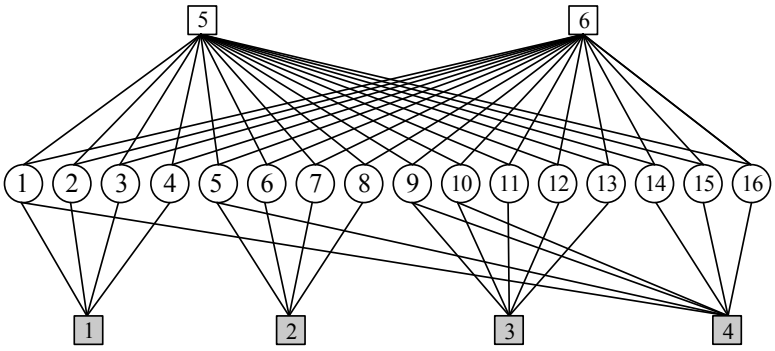


Figure 4.8: Tanner graph of our proposed $(n, k, d, \overline{r}) = (16, 10, 5, 3.875)$ LRC with four local CNs (gray squares) and two global CNs (white squares).

represented as follows.

$$\mathbf{G}_0^T = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
35 & 134 & 39 & 29 & 15 & 191 & 187 & 3 & 102 & 38 \\
34 & 135 & 39 & 29 & 15 & 191 & 187 & 3 & 102 & 38 \\
234 & 137 & 29 & 254 & 245 & 110 & 153 & 9 & 223 & 2 \\
243 & 249 & 60 & 11 & 59 & 234 & 48 & 37 & 217 & 104 \\
25 & 112 & 32 & 245 & 206 & 132 & 169 & 44 & 6 & 106
\end{pmatrix},$$

where each element of $\mathbf{G}_0$, denoted $g_{i,j}$ for $i \in [1, 16]$ and $j \in [1, 10]$, is the

decimal representation of a byte of the form $[a_7, \cdots, a_0] \in \mathbb{F}_2^{1\times 8}$, i.e. $g_{i,j} = \sum_{l=0}^{7} a_l 2^l$ where $a_l \in \mathbb{F}_2$, $l \in [0, 7]$. For example, $35 \equiv [0, 0, 1, 0, 0, 0, 1, 1]$.

4.3 Conclusion

The average locality of locally repairable codes (LRCs) determines how much bandwidth and how many I/O operations on average are required to recover a failed data block. The focus of the literature has been mainly on the maximum locality (simply called locality) which determines the maximum (as opposed to average) bandwidth and I/O required for a data block recovery. In the first part of this chapter, we used a novel approach to find the first general lower bound on $\overline{r}$ of erasure codes with arbitrary parameters. We also presented the graphical construction of $\bar{r}$-optimal LRCs for a broad range of codes' parameters offering improvement on $\overline{r}$ over the existing LRCs. Designing fully-explicit $\bar{r}$-optimal LRCs can be done by selecting the non-zero elements of the parity check matrix randomly from a sufficiently large finite field.

Chapter 5

On the Average Information Locality of Locally Repairable Codes

In the previous chapter, we addressed the problem of average locality of *all* symbols. In this chapter, we focus on the average locality of *information* symbols of LRCs.

Data node failures may be temporal or permanent where 90% of failures correspond to temporal cases. The work presented in this chapter is motivated by the fact that in temporal unavailabilities there is no need for reconstructing parity blocks, thus recovering information blocks is more frequently needed than recovering parity blocks. Considering the significant amount of block recovery performed daily in a DSS [6], even a small improvement of the average locality of information blocks ($\bar{r}_{inf}$) can result in significant savings.

In this chapter, we obtain an achievable lower bound on $\bar{r}_{inf}$ of any (n, k, d) LRC. We also design a class of LRCs that achieve the established bound.

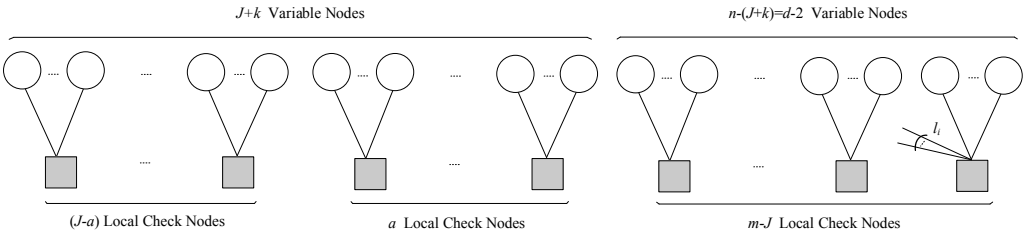


Figure 5.1: Locality Tanner graph of our proposed (n, k, d) $\bar{r}_{inf}$-optimal LRCs. Global check nodes are not shown in this figure.

5.1 A Lower Bound on $\bar{r}_{inf}$

Theorem 5.1. *For any (n, k, d) LRC, the average information locality ($\bar{r}_{inf}$) is bounded as follows*

$$\bar{r}_{inf} \geq \max\{1, \left\lceil \frac{k}{J} \right\rceil \left(2 - \frac{\lceil k/J \rceil - 1}{k/J}\right) - 1\},$$

where $J = n - k - d + 2$.

Proof. Please see Appendix C.1. □

Definition 5.1. *($\bar{r}_{inf}$-optimal LRCs). We call LRCs that achieve the bound obtained in Theorem 5.1 $\bar{r}_{inf}$-optimal LRCs.*

5.2 A Class of $\bar{r}_{inf}$-optimal LRCs

Here, we construct LRCs, denoted $\mathcal{C}_p$, with the minimum $\bar{r}_{inf}$ for any given (n, k, d). We use locality Tanner graph in order to represent our proposed $\bar{r}_{inf}$-optimal LRCs. Considering Fig. 5.1, $\mathcal{C}_p$ is constructed through the following steps.

Step 1. We divide the k variable nodes into J sets whose sizes are as uniform as possible. In order to do this, we divide the k variable nodes into $J\lceil \frac{k}{J} \rceil - k$ and $k + J - J\lceil \frac{k}{J} \rceil$ sets with cardinality $\lfloor \frac{k}{J} \rfloor$ and $\lceil \frac{k}{J} \rceil$, respectively. Then, we form J local groups by connecting all the information nodes of each set plus its parity

node to a local check node. By doing this for all the J sets, $k+J$ variable nodes in the locality Tanner graph are linked to J local check nodes, where k and J variable nodes correspond with the information and parity blocks, respectively.

Step 2. We divide the remaining $n-(k+J) = d-2$ parity nodes into $\left\lfloor \frac{d-2}{\lceil \frac{k}{J} \rceil + 1} \right\rfloor$ sets (local groups) whose size is $\lceil \frac{k}{J} \rceil + 1$; plus one set, denoted $\mathcal{R}$, whose size is $\theta := |\mathcal{R}| = d-2 \mod \lceil \frac{k}{J} \rceil + 1$. Hence, the total number of local groups associated with $\mathcal{C}_p$, denoted m_p, is

$$m_p = J + \left\lceil \frac{d-2}{\lceil \frac{k}{J} \rceil + 1} \right\rceil. \tag{5.1}$$

Step 3. If $\theta \neq 0$, in order to satisfy the constraint on the minimum distance mentioned in Remark 4.1, we have to connect l_i links from the i-th local group, for $i \in [1, m_p-1]$ to the last local check node, where $l_i \geq 0$. In order to minimize the locality of the variable nodes of $\mathcal{R}$, which is $\theta + \sum_{i\in[1,m_p-1]} l_i$, we solve the following linear integer programming in order to obtain l_i for $i \in [1, m_p - 1]$.

$$\begin{aligned} &\underset{l_i}{\text{minimize}} \sum_{i\in[1,m_p-1]} l_i \\ &\text{subject to} \\ &\sum_{i\in[1,m_p-1]\setminus\mathcal{A}} l_i \geq J + k - \theta - (J-1)\left\lceil \frac{k}{J} \right\rceil - a_{\mathcal{A}}, \end{aligned} \tag{5.2}$$

where $\mathcal{A} \in \text{comb}([1, m_p-1], J-1)$ with size $\binom{m_p-1}{J-1}$ and $a_{\mathcal{A}}$ is the number of local groups in set $[1, m_p - 1] \setminus \mathcal{A}$ with cardinality $\lceil \frac{k}{J} \rceil + 1$. Therefore, $\sum_{i\in[1,m_p-1]} l_i$ is minimized given the $\binom{m_p-1}{J-1}$ constraints shown in (5.2). This implies that the locality of the last local group $(\theta + \sum_{i\in[1,m_p-1]} l_i)$ is minimized for the given construction.

Step 4. As the last step, connect all the $n-k-m_p$ global check nodes to all

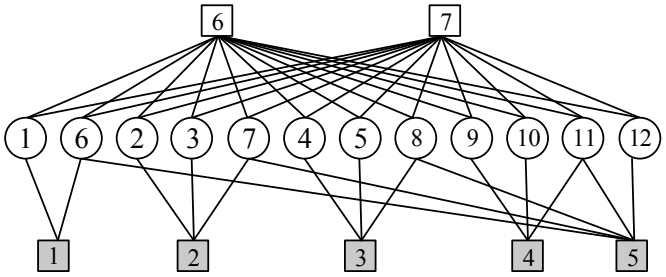


Figure 5.2: Locality Tanner graph of an r_{inf}-optimal $(n, k, d) = (12, 5, 6)$ LRC. Variable nodes indexed by 1 to 5 represent the information blocks, and the rest variable nodes represent the parity blocks.

the n variable nodes.

Remark 5.1. *We note that by doing the first step, the minimum average locality of the information blocks is obtained and the bound in Theorem 5.1 is achieved. One can connect all the remaining $n - k - J = d - 2$ parity nodes to all the variable nodes in order to satisfy the constraint on d. However, in our construction, we connect the remaining $d-2$ check nodes to variable nodes such that a small $\bar{r}$ is attained.*

Remark 5.2. *Solving (5.2) is always feasible. One straightforward solution for (5.2) is to connect the last local check node, which correspond with set $\mathcal{R}$, to all the variable nodes. By doing this, locality of the parity nodes in $\mathcal{R}$ would be k which is the maximum possible locality for an (n, k, d) code.*

Example 5.1. *Fig. 5.2 represents an $(n, k, d, J) = (12, 5, 6, 3)$ LRC constructed using the four steps described above. As the first step, we divide $k = 5$ information symbols $\{y_1, \cdots, y_5\}$ into $J\lceil \frac{k}{J} \rceil - k = 3\lceil \frac{5}{3} \rceil - 5 = 1$ set $\{y_1\}$ with cardinality $\lfloor \frac{k}{J} \rfloor = \lfloor \frac{5}{3} \rfloor = 1$; and $k + J - J\lceil \frac{k}{J} \rceil = 5 + 3 - 3\lceil \frac{5}{3} \rceil = 2$ sets $\{y_2, y_3\}$ and $\{y_4, y_5\}$ with cardinality $\lceil \frac{k}{J} \rceil = \lceil \frac{5}{3} \rceil = 2$. Then, we form linear combinations of each set. In the Tanner graph, it is equivalent to constructing the first J local groups. As the second step, we divide the rest $d - 2 = 4$ parity blocks into $\left\lfloor \frac{4}{\lceil \frac{5}{3} \rceil + 1} \right\rfloor = 1$ set with size $\lceil \frac{5}{3} \rceil + 1 = 3$; plus one group with*

size $\theta = 4 \mod 3 = 1$. *Therefore, the total number of local check nodes is* $m_p = J + \lceil \frac{d-2}{\lceil k/J \rceil + 1} \rceil = 3 + \lceil \frac{6-2}{\lceil 5/3 \rceil + 1} \rceil = 5$. *As the third step, we solve the following optimization problem in order to obtain* l_i *for* $i \in [1, 4]$

$$\begin{aligned} &\underset{l_i}{\textit{minimize}} \quad \sum_{i \in [1,4]} l_i \\ &\textit{subject to} \quad \sum_{i \in [1,4] \setminus \mathcal{A}} l_i \geq 7 - 2\left\lceil \frac{5}{3} \right\rceil - a_{\mathcal{A}} = 3 - a_{\mathcal{A}}, \end{aligned} \tag{5.3}$$

where $\mathcal{A} \in \mathsf{comb}([1, 4], 2)$ *and* $a_{\mathcal{A}}$ *is the number of local groups in set* $[1, 4] \setminus \mathcal{A}$ *with cardinality* 3. *By using PuLP, a linear programming modeler written in Python, we solve (5.3) which results in* $l_1 = l_2 = l_3 = l_4 = 1$. *This implies that each of the local groups one to four have one variable node linked to local check node five. As the last step, we connect the two global check nodes six and seven to all 12 variable nodes. Fig. 5.2 shows the Tanner graph representation of our proposed* $(n, k, d, \bar{r}_{inf}) = (12, 5, 6, 1.8)$ $\bar{r}_{inf}$*-optimal LRC.*

Proposition 5.1. *The minimum distance of the* (n, k) *LRC constructed in Section 5.2 is* d.

Proof. Please see Appendix C.2. □

5.3 Conclusion

The average locality of the information blocks ($\bar{r}_{inf}$) in the distributed storage systems (DSSs) is translated to the average data required to reconstruct an unavailable information block. The importance of the reconstruction cost as well as the frequent unavailability of information blocks in the real-world DSSs make designing codes with the minimum $\bar{r}_{inf}$ of interest. In this chapter, we obtained an achievable lower bound on $\bar{r}_{inf}$ and designed a class of locally repairable block codes which always achieves the obtained bound.

Chapter 6

Conclusion and Future Work

In this chapter, first the contributions of this dissertation is summarized. Then, new problems are described for future research directions.

6.1 Summary of the Contributions

LRCs decrease the required reconstruction bandwidth, disk I/O, and the number of active nodes that needs to be connected during the reconstruction of a missing block. The main focus of this dissertation was on LRCs which have been recently proposed and used in cloud storage systems. Motivated by the following two points about LRCs: i) binary LRCs are desirable in practice, as they eliminate the need for costly multiplication in operations such as encoding, decoding, and repair in the immense-size DSSs; and ii) the average locality of locally repairable codes is an important parameter that directly affects the amount of data required to be communicated between nodes in the case of reconstruction a failed data node, we considered the following two problems in this dissertation: i) designing binary locally repairable codes; and ii) obtaining lower bounds on the average locality of all blocks and that of the information blocks of LRCs and designing codes that achieve the obtained bounds. By using a novel approach to LRC design based on Tanner graphs, we solved these

problems and produced several design-oriented theorems for LRCs. Following is the summary of our results.

6.1.1 Binary LRCs

In Chapter 3, we proposed binary LRCs with minimum distance of four, which provide sufficient reliability for a wide range of practical code and system parameters. We verified that for practical values of code locality, our proposed binary LRCs have either identical or slightly lower code rate compared to that of non-binary LRCs. Next, using the idea behind our design, we proposed optimal LRCs over small finite fields. Comparing with the most recent proposed LRCs, we decreased the required field size from n (code length) to r (code locality). Then, by using the construction of spanning BLRC with minimum distance four as a backbone, we designed LRCs with minimum distance six. We did this by adding only one non-binary parity block to our proposed binary LRCs. Finally, we obtained a closed-form equation for MTTDL—which is used to evaluate the code reliability—in terms of parameters of DSS and erasure code. The closed-form equation of MTTDL shows how the system and code parameters affect MTTDL. Hence, by adjusting the code/DSS parameters, it is possible to simply track the value of MTTDL.

6.1.2 Average Locality of LRCs

In Chapters 4 and 5, rather than the maximum locality of the locally repairable codes considered in the literature, we focused on studying the average locality of locally repairable codes. First, in Chapter 4, we derived bounds for the average locality of all blocks in terms of codes parameters. We also designed three classes of codes that achieve the obtained bounds. Next, in Chapter 5, we focused on the average locality of information blocks which is crucial because in temporal unavailabilities, there is no need for reconstructing parity blocks but

information blocks must be reconstructed. We obtained an achievable lower bound on the average locality of information blocks. We also designed a class of LRCs that achieve the established bound. The designed LRCs in Chapters 4 and 5, which have the minimum average (information) locality, do not sacrifice two crucial parameters of the code namely the minimum distance and rate.

6.2 Future Research Directions

6.2.1 Binary LRCs with Arbitrary Minimum Distance

Although in some distributed storage systems such as HDFS, codes with a minimum distance four provide desirable reliability, in some other types such as peer-to-peer storage systems, codes with larger minimum distance are of interest. In Chapter 3, we proposed binary LRCs with minimum distance four. Then, in order to increase minimum distance of the code, we added one non-binary parity block to the existing parity blocks. One research direction can be on generalization of binary LRCs with arbitrary minimum distance.

6.2.2 Average Locality for Erasure Codes with Multiple Repair Groups

In our average locality analysis presented in Chapter 4, we focused on an important class of LRCs that can handle any *single* failures with a small number of other blocks. In some other types of LRCs, in order to improve the availability of data blocks, a missing block can be reconstructed by accessing any group out of more than one disjoint groups of active nodes. Generalizing our proposed lower bound on $\overline{r}$ and $\overline{r}_{inf}$ for LRCs with multiple groups of locality as well as constructing LRCs that achieve the obtained bounds remain open problems.

6.2.3 Explicit construction of $\overline{r}$-optimal and $\overline{r}_{inf}$-optimal LRCs

In Chapter 4, we presented the graphical construction of $\bar{r}$-optimal and $\bar{r}_{inf}$-optimal LRCs. Our graphical construction determines the zero elements of the parity-check matrix of the code. The non-zero elements can be randomly selected from a sufficiently large finite field. Another research direction can be constructing $\bar{r}$-optimal and $\bar{r}_{inf}$-optimal LRCs by determining all non-zero coefficients of the parity-check matrix fully explicitly from small finite fields.

Chapter 7

Bibliography

[1] M. Shahabinejad, M. Khabbazian, and M. Ardakani. An efficient binary locally repairable code for hadoop distributed file system. *Communications Letters, IEEE*, 18(8):1287–1290, Aug 2014.

[2] M. Shahabinejad, M. Khabbazian, and M. Ardakani. A class of binary locally repairable codes. *IEEE Transactions on Communications*, PP(99): 1–1, June 2016.

[3] M. Shahabinejad, M. Khabbazian, and M. Ardakani. On the average locality of locally repairable codes. *IEEE Transactions on Communications, Under Review*, Aug. 2016.

[4] M. Shahabinejad, M. Ardakani, and M. Khabbazian. An erasure code with reduced average locality for distributed storage systems. pages 427–431, Jan 2017.

[5] M. Shahabinejad, M. Khabbazian, and M. Ardakani. Locally repairable codes with the optimum average information locality. pages 181–185, June 2017.

[6] Maheswaran Sathiamoorthy, Megasthenis Asteris, Dimitris Papailiopoulos, Alexandros G Dimakis, Ramkumar Vadali, Scott Chen, and Dhruba

Borthakur. XORing elephants: Novel erasure codes for big data. *Proc. VLDB*, 6(5):325–336, 2013.

[7] Cheng Huang, Huseyin Simitci, Yikang Xu, Aaron Ogus, Brad Calder, Parikshit Gopalan, Jin Li, Sergey Yekhanin, et al. Erasure coding in Windows Azure storage. *Proc. USENIX Annual Technical Conference*, pages 15–26, 2012.

[8] A.G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh. A survey on network codes for distributed storage. *Proceedings of the IEEE*, 99(3):476–489, 2011.

[9] Cheng Huang, Minghua Chen, and Jin Li. Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems. *ACM Trans. Storage*, 9(1):1–28, mar 2013.

[10] Daniel Ford, Francois Labelle, Florentina Popovici, Murray Stokely, Van-Anh Truong, Luiz Barroso, Carrie Grimes, and Sean Quinlan. Availability in globally distributed storage systems. *Proc. USENIX Symposium on Operating Systems Design and Implementation*, pages 61–74, 2010.

[11] Alexandros G. Dimakis, Brighten Godfrey, Yunnan Wu, Martin J. Wainwright, and Kannan Ramchandran. Network coding for distributed storage systems. *IEEE Transactions on Information Theory*, 56:4539–4551, 2010.

[12] K.V. Rashmi, N.B. Shah, and P.V. Kumar. Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction. *IEEE Transactions on Information Theory*, 57(8): 5227–5239, 2011.

[13] H. Hou, K. W. Shum, M. Chen, and H. Li. BASIC codes: Low-complexity regenerating codes for distributed storage systems. *IEEE Transactions on Information Theory*, 62(6):3053–3069, June 2016.

[14] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran. Distributed storage codes with repair-by-transfer and nonachievability of interior points on the storage-bandwidth tradeoff. *IEEE Transactions on Information Theory*, 58(3):1837–1852, March 2012.

[15] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran. Interference alignment in regenerating codes for distributed storage: Necessity and code constructions. *IEEE Transactions on Information Theory*, 58 (4):2134–2158, April 2012.

[16] P. Gopalan, Cheng Huang, H. Simitci, and S. Yekhanin. On the locality of codeword symbols. *Information Theory, IEEE Transactions on*, 58(11): 6925–6934, Nov 2012.

[17] K. V. Rashmi, Nihar B. Shah, and Kannan Ramchandran. A piggybacking design framework for read-and download-efficient distributed storage codes. *CoRR*, abs/1302.5872, 2013.

[18] O. Khan, R. Burns, J. S. Plank, W. Pierce, and C. Huang. Rethinking erasure codes for cloud file systems: Minimizing I/O for recovery and degraded reads. In *FAST-2012: 10th Usenix Conference on File and Storage Technologies*, San Jose, February 2012.

[19] KV Rashmi, Preetum Nakkiran, Jingyan Wang, Nihar B. Shah, and Kannan Ramchandran. Having your cake and eating it too: Jointly optimal erasure codes for I/O, storage, and network-bandwidth. pages 81–94, 2015.

[20] F. Oggier and A. Datta. Self-repairing homomorphic codes for distributed storage systems. pages 1215–1223, 2011.

[21] R. Singleton. Maximum distance q -nary codes. *IEEE Transactions on Information Theory*, 10(2):116–118, Apr 1964. ISSN 0018-9448. doi: 10.1109/TIT.1964.1053661.

[22] I. S. Reed and G. Solomon. Polynomial codes over certain finite field. *Journal of The Society for Industrial and Applied Mathematics*, 8, 1960.

[23] J. S. Plank and C. Huang. Tutorial: Erasure coding for storage applications. Slides presented at FAST-2013: 11th Usenix Conference on File and Storage Technologies, February 2013.

[24] K.V. Rashmi, N.B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran. A solution to the network challenges of data recovery in erasure-coded distributed storage systems: a study on the facebook warehouse cluster. In *5th USENIX Workshop on Hot Topics in Storage and File Systems*, 2013.

[25] D.S. Papailiopoulos and A.G. Dimakis. Locally repairable codes. *Information Theory, IEEE Transactions on*, 60(10):5843–5855, Oct 2014.

[26] I. Tamo and A. Barg. A family of optimal locally recoverable codes. *Information Theory, IEEE Transactions on*, 60(8):4661–4676, Aug 2014.

[27] M. Mehrabi, M. Shahabinejad, M. Khabbazian, and M. Ardakani. Optimal locally repairable codes with improved update complexity. *IEEE Transactions on Communications, Under Review*, Nov. 2017.

[28] I. Tamo, D.S. Papailiopoulos, and A.G. Dimakis. Optimal locally repairable codes and connections to matroid theory. In *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, pages 1814–1818, July 2013.

[29] Junsheng Han and L.A. Lastras-Montao. Reliable memories with subline accesses. In *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, pages 2531–2535, June 2007.

[30] N. Silberstein, A.S. Rawat, O.O. Koyluoglu, and S. Vishwanath. Optimal locally repairable codes via rank-metric codes. pages 1819–1823, July 2013.

[31] N. Prakash, G.M. Kamath, V. Lalitha, and P.V. Kumar. Optimal linear codes with a local-error-correction property. pages 2776–2780, July 2012.

[32] Mehrtash Mehrabi and Masoud Ardakani. On minimum distance of locally repairable codes. In *IEEE 15th Canadian Workshop on Information Theory (CWIT 2017)*, pages 36–40, Quebec city, Canada, June 2017.

[33] M. Mehrabi, M. Ardakani, and M. Khabbazian. Minimizing the update complexity of facebook hdfs-raid locally repairable code. In *IEEE 86th Vehicular Technology Conference: VTC2017-Fall*, Toronto, Canada, September 2017.

[34] Chris Lomont. Introduction to intel advanced vector extensions. June 2011. URL `https://software.intel.com/en-us/articles/introduction-to-intel-advanced-vector-extensions`.

[35] S. Goparaju and R. Calderbank. Binary cyclic codes that are locally repairable. In *Information Theory (ISIT), 2014 IEEE International Symposium on*, pages 676–680, June 2014.

[36] A. Zeh and E. Yaakobi. Optimal linear and cyclic locally repairable codes over small fields. In *Information Theory Workshop (ITW), 2015 IEEE*, pages 1–5, April 2015.

[37] V. Cadambe and A. Mazumdar. An upper bound on the size of locally recoverable codes. In *Network Coding (NetCod), 2013 International Symposium on*, pages 1–5, June 2013.

[38] N. Silberstein and A. Zeh. Optimal binary locally repairable codes via anticodes. In *Information Theory (ISIT), 2015 IEEE International Symposium on*, pages 1247–1251, June 2015.

[39] S. Kadhe and A. Sprintson. Codes with unequal locality. *ArXiv e-prints*, January 2016.

[40] A. Zeh and E. Yaakobi. Bounds and constructions of codes with multiple localities. *ArXiv e-prints*, January 2016.

[41] K. Kralevska, D. Gligoroski, and H. Overby. Balanced locally repairable codes. In *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, pages 280–284, Sept 2016.

[42] A. S. Rawat, O. O. Koyluoglu, N. Silberstein, and S. Vishwanath. Optimal locally repairable and secure codes for distributed storage systems. *IEEE Transactions on Information Theory*, 60(1):212–236, Jan 2014.

[43] G. M. Kamath, N. Prakash, V. Lalitha, and P. V. Kumar. Codes with local regeneration and erasure correction. *IEEE Transactions on Information Theory*, 60(8):4637–4660, Aug 2014.

[44] N. Prakash, V. Lalitha, and P.V. Kumar. Codes with locality for two erasures. *Information Theory (ISIT), 2014 IEEE International Symposium on*, pages 1962–1966, June 2014.

[45] A.S. Rawat, D.S. Papailiopoulos, A.G. Dimakis, and S. Vishwanath. Locality and availability in distributed storage. pages 681–685, June 2014.

[46] Anyu Wang and Zhifang Zhang. Repair locality with multiple erasure tolerance. *Information Theory, IEEE Transactions on*, 60(11):6979–6987, Nov 2014.

[47] L. Pamies-Juarez, H. D. L. Hollmann, and F. Oggier. Locally repairable codes with multiple repair alternatives. In *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, pages 892–896, July 2013.

[48] W. Song, S. H. Dau, C. Yuen, and T. J. Li. Optimal locally repairable linear codes. *IEEE Journal on Selected Areas in Communications*, 32(5): 1019–1036, May 2014.

[49] Wentu Song and Chau Yuen. Locally repairable codes with functional repair and multiple erasure tolerance. *CoRR*, abs/1507.02796, 2015. URL `http://arxiv.org/abs/1507.02796`.

[50] Douglas Gantenbein. A better way to store data. URL `http://research.microsoft.com/en-us/news/features/erasurecoding-090512.aspx`.

[51] K.M. Greenan, E.L. Miller, and T.J.E. Schwarz. Optimizing galois field arithmetic for diverse processor architectures and applications. *Proc. MASCOTS*, pages 1–10, Sept 2008.

[52] Sriram Ramabhadran and Joseph Pasquale. Analysis of long-running replicated systems. *INFOCOM*, 2006.

[53] Thai Le. Optimizing storage solutions using the intel intelligent storage acceleration library. September 2014. URL `https://software.intel.com/en-us/articles/optimizing-storage-solutions-using-the-intel-intelligent-storage-acceleration-library`.

[54] Intel intelligent storage acceleration library (intel ISA-L) open source version, July 2015.

[55] M. Blaum, J.L. Hafner, and S. Hetzler. Partial-MDS codes and their application to raid type of architectures. *Information Theory, IEEE Transactions on*, 59(7):4510–4519, July 2013.

[56] P. Gopalan, Cheng Huang, B. Jenkins, and S. Yekhanin. Explicit maximally recoverable codes with locality. *Information Theory, IEEE Transactions on*, 60(9):5245–5256, Sept 2014.

[57] M. Shahabinejad, M. Khabbazian, and M. Ardakani. On the average locality of locally repairable codes. *IEEE Transactions on Communications, Accepted for Publication*, 2017.

Appendices

Appendix A

Proofs for Chapter 3

A.1 Proof of Proposition 3.1

We use the concept of epoch, described in [52], to prove Proposition 3.1. Starting in state l, let us define Q_l as the probability that the system reaches the absorbing state O_{n-d} before it reaches state n. Similarly, starting in state l, let us define T_l as the elapsed time before the system reaches either the absorbing state O_{n-d} or state n. Then, MTTDL can be obtained as [52]

$$MTTDL = \frac{t_e}{Q^*}, \tag{A.1}$$

where $t_e = \frac{1}{n\lambda} + T^*$ with $Q^* := Q_{n-1}$ and $T^* := T_{n-1}$. Hence, in order to calculate MTTDL, we first determine Q^* and T^* in terms of other parameters of the system as follows.

We have

$$Q_l = \begin{cases} 1, & l = n-d \\ p_l Q_{l-1} + q_l Q_{l+1}, & l \in [n-d+1, n-1] \\ 0, & l = n \end{cases} \tag{A.2}$$

In (A.2),

$$p_l = \frac{l\lambda}{l\lambda + \rho_l} = \frac{l}{l + \frac{\rho_l}{\lambda}} = \frac{l}{l + \gamma_l}, \text{ and } q_l = \frac{\rho_l}{\rho_l + l\lambda} = \frac{\gamma_l}{l + \gamma_l},$$

where $\gamma_l := \frac{\rho_l}{\lambda}$ and

$$\rho_l = \begin{cases} \rho, & l \in [n-d+1, n-2] \\ \rho_1, & l = n-1 \end{cases}$$

Noting that $p_l + q_l = 1$, from (A.2), we have

$$Q_{l-1} - Q_l = \frac{q_l}{p_l}(Q_l - Q_{l+1}) = \frac{\gamma_l}{l}(Q_l - Q_{l+1}), \quad l \in [n-d+1, n-1]. \tag{A.3}$$

By using the definition $Q^* := Q_{n-1}$, we have

$$\begin{aligned} Q_{n-2} - Q_{n-1} &= \tfrac{\gamma_{n-1}}{n-1}(Q_{n-1} - Q_n) = \tfrac{\gamma_{n-1}}{n-1}Q^* \\ Q_{n-3} - Q_{n-2} &= \tfrac{\gamma_{n-2}}{n-2}(Q_{n-2} - Q_{n-1}) = \prod_{j=n-2}^{n-1} \tfrac{\gamma_j}{j} Q^* \\ &\vdots \\ Q_{n-d} - Q_{n-d+1} &= \tfrac{\gamma_{n-d+1}}{n-d+1}(Q_{n-d+1} - Q_{n-d+2}) = \prod_{j=n-d+1}^{n-1} \tfrac{\gamma_j}{j} Q^* \end{aligned} \tag{A.4}$$

Equivalently,

$$Q_{l-1} - Q_l = \prod_{j=l}^{n-1} \frac{\gamma_j}{j} Q^*, l \in [n-d+1, n-1]. \tag{A.5}$$

Let us define, $\gamma_l := \gamma = \frac{\rho_1}{\lambda}$ for $l \in [n-d+1, n-2]$. Then,

$$Q_{l-1} - Q_l = \frac{\gamma_{n-1}\gamma^{n-1-l}}{\prod_{j=l}^{n-1} j} Q^*, l \in [n-d+1, n-1]. \tag{A.6}$$

Hence, we have the following chain of equalities.

$$Q_{n-d} - Q_n = 1 - 0 = \sum_{l=n-d+1}^{n} (Q_{l-1} - Q_l) =$$
$$Q_{n-1} - Q_n + \sum_{l=n-d+1}^{n-1} (Q_{l-1} - Q_l) =$$
$$\Big(1 + \sum_{l=n-d+1}^{n-1} \frac{\gamma_{n-1}\gamma^{n-1-l}}{\prod_{j=l}^{n-1} j}\Big) Q^*.$$

Hence,

$$Q^* = \Big(1 + \sum_{l=n-d+1}^{n-1} \frac{\gamma_{n-1}\gamma^{n-1-l}}{\prod_{j=l}^{n-1} j}\Big)^{-1}. \tag{A.7}$$

Similarly, for T_l, we have:

$$T_l = \begin{cases} 0, & l \in \{n-d, n\} \\ p_l T_{l-1} + q_l T_{l+1} + t_l, & l \in [n-d+1, n-1] \end{cases} \tag{A.8}$$

where

$$t_l = \frac{1}{l\lambda + \rho_l}.$$

Noting that $p_l + q_l = 1$, from (A.8), we have

$$T_{l-1} - T_l = \frac{q_l}{p_l}(T_l - T_{l+1}) - \frac{t_l}{p_l} = \frac{\gamma_l}{l}(T_l - T_{l+1}) - \frac{1}{l\lambda} \tag{A.9}$$

for $l \in [n-d+1, n-1]$. By using the definition $T^* := T_{n-1}$, we have

$$\begin{gathered} T_{n-2} - T_{n-1} = \frac{\gamma_{n-1}}{n-1}(T_{n-1} - T_n) - \frac{1}{(n-1)\lambda} = \\ \frac{\gamma_{n-1}}{n-1}T^* - \frac{1}{(n-1)\lambda} \\ T_{n-3} - T_{n-2} = \frac{\gamma_{n-2}}{n-2}(T_{n-2} - T_{n-1}) - \frac{1}{(n-2)\lambda} = \\ \prod_{j=n-2}^{n-1} \frac{\gamma_j}{j} T^* - \frac{\gamma_{n-2}}{(n-1)(n-2)\lambda} - \frac{1}{(n-2)\lambda} \\ \vdots \\ T_{n-d} - T_{n-d+1} = \prod_{j=n-d+1}^{n-1} \frac{\gamma_j}{j} T^* - \frac{1}{\lambda} \sum_{m=1}^{d-1} \frac{\prod_{p\in[m+1,d-1]} \gamma_{n-p}}{\prod_{q\in[m,d-1]}(n-q)} \end{gathered} \tag{A.10}$$

Equivalently,

$$T_{l-1} - T_l = \prod_{j=l}^{n-1} \frac{\gamma_j}{j} T^* - \frac{1}{\lambda} \sum_{m=1}^{n-l} \frac{\prod_{p\in[m+1,n-l]} \gamma_{n-p}}{\prod_{q\in[m,n-l]}(n-q)}, l \in [n-d+1, n-1].$$

By using the definition $\gamma_l := \gamma = \frac{\rho_1}{\lambda}$ for $l \in [n-d+1, n-2]$, we have

$$T_{l-1} - T_l = \frac{\gamma_{n-1}\gamma^{n-1-l}}{\prod_{j=l}^{n-1} j} T^* - \frac{1}{\lambda} \sum_{m=1}^{n-l} \frac{\gamma^{n-l-m}}{\prod_{q\in[m,n-l]}(n-q)}$$

for $l \in [n-d+1, n-1]$. Thus,

$$\begin{gathered} T_{n-d} - T_n = 0 - 0 = \sum_{l=n-d+1}^{n} (T_{l-1} - T_l) = \\ T_{n-1} - T_n + \sum_{l=n-d+1}^{n-1} (T_{l-1} - T_l) = \\ T^* + \sum_{l=n-d+1}^{n-1} \left(\frac{\gamma_{n-1}\gamma^{n-1-l}}{\prod_{j=l}^{n-1} j} T^* - \frac{1}{\lambda} \sum_{m=1}^{n-l} \frac{\gamma^{n-l-m}}{\prod_{q\in[m,n-l]}(n-q)}\right). \end{gathered}$$

Hence,

$$T^*(1+\sum_{l=n-d+1}^{n-1}\frac{\gamma_{n-1}\gamma^{n-1-l}}{\prod_{j=l}^{n-1}j}) = \frac{T^*}{Q^*}$$

$$= \frac{1}{\lambda}\sum_{l=n-d+1}^{n-1}\sum_{m=1}^{n-l}\frac{\gamma^{n-l-m}}{\prod_{q\in[m,n-l]}(n-q)} \tag{A.11}$$

From (A.1),

$$MTTDL = \frac{t_e}{Q^*} = \frac{1}{n\lambda Q^*} + \frac{T^*}{Q^*}. \tag{A.12}$$

By substituting (A.7) and (A.11) in (A.12), $MTTDL$ is calculated as

$$\frac{1}{n\lambda}\Big(1+\sum_{l=n-d+1}^{n-1}\frac{\gamma_{n-1}\gamma^{n-1-l}}{\prod_{j=l}^{n-1}j}\Big) + \frac{1}{\lambda}\sum_{l=n-d+1}^{n-1}\sum_{m=1}^{n-l}\frac{\gamma^{n-l-m}}{\prod_{q\in[m,n-l]}(n-q)} =$$

$$\frac{1}{n\lambda}\Big(1+\sum_{l=n-d+1}^{n-1}\frac{\gamma_{n-1}\gamma^{n-1-l}}{\prod_{j=l}^{n-1}j} + n\sum_{l=n-d+1}^{n-1}\sum_{m=1}^{n-l}\frac{\gamma^{n-l-m}}{\prod_{q\in[m,n-l]}(n-q)}\Big) =$$

$$\frac{1}{n\lambda}\left(1+\sum_{l=n-d+1}^{n-1}\Big(\frac{\gamma_{n-1}\gamma^{n-1-l}}{\prod_{j=l}^{n-1}j} + n\sum_{m=1}^{n-l}\frac{\gamma^{n-l-m}}{\prod_{q\in[m,n-l]}(n-q)}\Big)\right) =$$

$$\frac{\gamma^{d-2}}{n\lambda}\Big(\frac{\gamma_{n-1}}{\prod_{j=n-d+1}^{n-1}j} + \frac{n}{\prod_{q\in[m,n-l]}(n-q)}\Big) + f(\gamma),$$

where $f(\gamma)$ is a polynomial of degree $\gamma^{(d-3)}$ with positive values. Typically, $\gamma >> 1$ and $\gamma_{n-1} >> 1$ [7, 6]. Therefore, MTTDL can be estimated as follows.

$$MTTDL \approx \frac{\gamma_{n-1}\gamma^{d-2}}{n\lambda\prod_{j=n-d+1}^{n-1}j} = \frac{\gamma_{n-1}\gamma^{d-2}}{n(n-1)...(n-d+1)\lambda}. \tag{A.13}$$

A.2 Proof of Proposition 3.5

Lemma A.1. *Let* $\mathbf{V}$ *be a* $(0-1)$*-matrix. Assume* $\mathbf{V}$ *is full column rank over* $\mathbb{F}_2$*. Then, for every integer* $m > 1$*,* $\mathbf{V}$ *is full column rank over* $\mathbb{F}_{2^m}$*.*

Proof. Let us represent $\mathbf{V}$ as $(\mathbf{v}_1, \cdots, \mathbf{v}_b) \in \mathbb{F}_2^{a\times b}$ with columns $\mathbf{v}_i$ for $i \in [b]$. By contradiction, assume that one column of $\mathbf{V}$, say $\mathbf{v}_1$, is a linear combination of other columns of $\mathbf{V}$, that is $\mathbf{v} = \sum_{j=2}^{b} \alpha_j \mathbf{v}_j$. Each α_j can be considered as a vector of length m over $\mathbb{F}_2$, i.e. $\alpha_j = (\alpha_j^{(m)}, \cdots, \alpha_j^{(2)}, \alpha_j^{(1)}) \in \mathbb{F}_2^{1\times m}$, with the identity element being $(0, \cdots, 0, 1)$. By this view and using $\mathbf{v} = \sum_{j=2}^{b} \alpha_j \mathbf{v}_j$, we get $\mathbf{v}_1 = \sum_{j=2}^{b} \alpha_j^{(1)} \mathbf{v}_j$, which contradicts the assumption that $\mathbf{V}$ is full column rank over $\mathbb{F}_2$. □

Lower Bound on d: In the following, we show that any sub-matrix $\mathbf{H}_L$ constructed from arbitrary five columns $\mathbf{h}_{L_1}$, $\mathbf{h}_{L_2}$, $\mathbf{h}_{L_3}$, $\mathbf{h}_{L_4}$, and $\mathbf{h}_{L_5}$ of $\mathbf{H}_{LRC}$ in (3.21) is full-rank. Consider $\mathbf{H}_L$ as follows.

$$\mathbf{H}_L = \begin{pmatrix} \mathbf{h}_{L_1} & \mathbf{h}_{L_2} & \mathbf{h}_{L_3} & \mathbf{h}_{L_4} & \mathbf{h}_{L_5} \end{pmatrix} = \begin{pmatrix} \mathbf{h}_{B_1} & \mathbf{h}_{B_2} & \mathbf{h}_{B_3} & \mathbf{h}_{B_4} & \mathbf{h}_{B_5} \\ h^3_{B_1} & h^3_{B_2} & h^3_{B_3} & h^3_{B_4} & h^3_{B_5} \end{pmatrix} \in \mathbb{F}_{2^{\Gamma_{r,n}}}^{(1+\Gamma_{r,n})\times 5}, \tag{A.14}$$

where $\mathbf{h}_{B_1}, \mathbf{h}_{B_2}, \mathbf{h}_{B_3}, \mathbf{h}_{B_4},$ and $\mathbf{h}_{B_5}$ are the corresponding binary columns from $\mathbf{H}_{BLRC}$ in (3.20). As shown in the proof of proposition 3.2, XOR of up to three columns of $\mathbf{H}_{BLRC}$ always yields a non-zero vector. Also, considering the construction of $\mathbf{H}_{BLRC}$, it is not difficult to verify that XORing any five columns of $\mathbf{H}_{BLRC}$ results in a vector with at least one non-zero element. Hence, according to Lemma A.1, linear combination of any two, three, and five columns of $\mathbf{H}_L$ is a non-zero vector over $\mathbb{F}_{2^{\Gamma_{r,n}}}$.

In the following, we show that no four columns of $\mathbf{H}_L$ are linearly dependent which concludes the proof. Without loss of generality, assume that four columns

$\mathbf{h}_{L_1}, \mathbf{h}_{L_2}, \mathbf{h}_{L_3},$ and $\mathbf{h}_{L_4}$ are linearly dependent. Hence,

$$\mathbf{h}_{L_4} = \sum_{i=1}^{3} \lambda_i \mathbf{h}_{L_i}, \text{ for some non-zero } \lambda_i \in \mathbb{F}_{2^{\Gamma_{r,n}}}. \tag{A.15}$$

Now, we show that (A.15) is true only if $\lambda_i = 1 \ \forall i$. Equation (A.15) implies that

$$\mathbf{h}_{B_4} = \sum_{i=1}^{3} \lambda_i \mathbf{h}_{B_i}. \tag{A.16}$$

Therefore, we must have

$$\mathbf{h}_{B_4} = \sum_{i=1}^{3} \mathbf{h}_{B_i} \tag{A.17}$$

as otherwise $\mathbf{h}_{B_1}$ to $\mathbf{h}_{B_4}$ will be independent over $\mathbb{F}_2$ which by Lemma A.1 implies $\mathbf{h}_{B_1}$ to $\mathbf{h}_{B_4}$ are independent over $\mathbb{F}_{2^{\Gamma_{r,n}}}$. By adding (A.16) and (A.17), we have $\mathbf{h}_{B_4} + \mathbf{h}_{B_4} = \mathbf{0} = \sum_{i=1}^{3} (\lambda_i + 1)\mathbf{h}_{B_i}$. This can only happen when $\lambda_i = 1$ for $i \in [3]$. Hence, from (A.15), we have $\mathbf{h}_{L_4} = \sum_{i=1}^{3} \mathbf{h}_{L_i}$. This results in

$$\begin{cases} h_{B_1} + h_{B_2} = h_{B_3} + h_{B_4} \\ h_{B_1}^3 + h_{B_2}^3 = h_{B_3}^3 + h_{B_4}^3 \end{cases} \tag{A.18}$$

Note that the set of equations $x_1 + x_2 = A$ and $x_1^3 + x_2^3 = B$ has a unique set of solutions over $\mathbb{F}_{2^{\Gamma_{r,n}}}$. Hence, (A.18) is true only if $h_{B_1} = h_{B_3}$ or $h_{B_1} = h_{B_4}$ which contradicts the fact that h_{B_i} are distinct for $i \in [4]$.

Upper Bound on d: From (3.21), observe that for our proposed LRC over $\mathbb{F}_{2^{\Gamma_{n,r}}}$, we have

$$n - k = \Gamma_{r,n} + 1 = \frac{n}{r+1} + \left\lceil \log_2(r+1) \right\rceil + 1. \tag{A.19}$$

Hence,

$$\left\lceil \frac{k}{r} \right\rceil = \frac{n}{r+1} - \left\lfloor \frac{1}{r} \Big(\left\lceil \log_2(r+1) \right\rceil + 1 \Big) \right\rfloor. \tag{A.20}$$

By replacing (A.19) and (A.20) in (2.2), we have

$$d \leq \left\lceil \log_2(r+1) \right\rceil + \left\lfloor \frac{1}{r} \Big(\left\lceil \log_2(r+1) \right\rceil + 1 \Big) \right\rfloor + 3.$$

Appendix B

Proofs for Chapter 4

B.1 Proof of Theorem 4.1

Lemma B.1. *Consider function* $f(z_1, \cdots, z_m, m) = \sum_{i=1}^{m} {z_i}^2$*, where* z_i*'s are positive non-zero integers for* $i \in [1, m]$*, which satisfy* $\sum_{i=1}^{m} z_i = A$*, and* A *is a constant integer. Assume that minimum of* $f(z_1, \cdots, z_m, m)$ *is attained when* $z_i = z_i^* \ \forall i$*, i.e.*

$$f_{\min}(m) := f(z_1^*, \cdots, z_m^*, m) = \min_{z_i} f(z_1, \cdots, z_m, m).$$

Then, $f_{\min}(m)$ *is a non-increasing function.*

Proof. We need to show that $f_{\min}(m) \geq f_{\min}(m+1)$. Assume that $f_{\min}(m) = f(w_1, \cdots, w_m, m)$ and $f_{\min}(m+1) = f(v_1, \cdots, v_{m+1}, m+1)$ with $\sum_{i=1}^{m} w_i = \sum_{i=1}^{m+1} v_i = A$. We have,

$$\begin{aligned} f_{\min}(m) &= \sum_{i=1}^{m} w_i^2 = \sum_{i=1}^{m-1} w_i^2 + (\frac{w_m}{2} + \frac{w_m}{2})^2 \\ &\geq \sum_{i=1}^{m-1} w_i^2 + (\frac{w_m}{2})^2 + (\frac{w_m}{2})^2 \\ &\geq \sum_{i=1}^{m+1} v_i^2 = f_{\min}(m+1), \end{aligned}$$

where the first inequality is by $(a+b)^2 \geq a^2 + b^2$. $\square$

Proof of Theorem 4.1: First, by using Algorithm 1, the set of n VNs $\{y_1, \cdots, y_n\}$ of $\mathcal{C}$ is partitioned into m local groups $\mathcal{Y}_1$ to $\mathcal{Y}_m$, where $m \in \{1, \cdots, J, J+1, \cdots, n-k\}$. Now, we consider the following two cases: (a) $m \in [1, J]$, and (b) $m \in [J, n-k]$.

Case (a) $m \in [1, J]$: In this case,

$$n\overline{r} = \sum_{i=1}^{m} |\mathcal{Y}_i| r_i \geq \sum_{i=1}^{m} |\mathcal{Y}_i|(|\mathcal{Y}_i| - 1) = \sum_{i=1}^{m} |\mathcal{Y}_i|^2 - \sum_{i=1}^{m} |\mathcal{Y}_i| \tag{B.1}$$

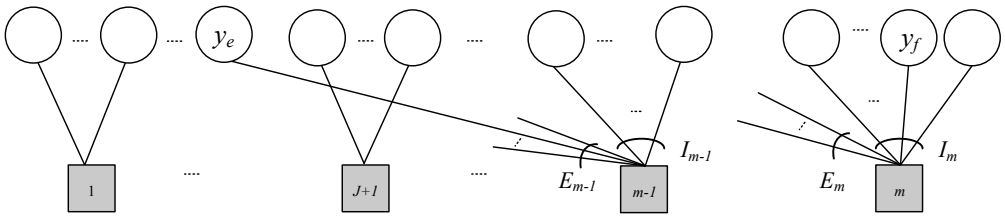
Observe that $\sum_{i=1}^{m} |\mathcal{Y}_i| = n, \ \forall m \in [1, J]$ because all the m local groups have to cover all n VNs. By Lemma B.1, $\sum_{i=1}^{m} |\mathcal{Y}_i|^2$ is a non-increasing function; hence, (B.1) takes its minimum at $m = J$.

Case (b) $m \in [J, n-k]$: In this case, for the code to have the minimum distance of d, by Lemma 4.2, the first J local groups must cover $n - \theta$ VNs for some integer $\theta \in [0, d-2]$.

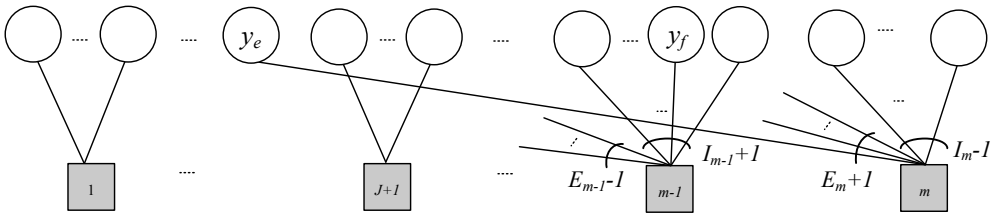
$$\sum_{i=1}^{n-\theta} \mathit{Loc}(y_i) = \sum_{j=1}^{J} |\mathcal{Y}_j| r_j \geq \sum_{j=1}^{J} |\mathcal{Y}_j|(|\mathcal{Y}_j| - 1),$$

where $\sum_{j=1}^{J} |\mathcal{Y}_j| = n - \theta$. Observe that the minimum of $\sum_{j=1}^{J} |\mathcal{Y}_j|(|\mathcal{Y}_j| - 1)$ is obtained when $\sum_{j=1}^{J} |\mathcal{Y}_j|$ is minimized, i.e., when $\sum_{j=1}^{J} |\mathcal{Y}_j| = n - (d-2) = J + k$. In this case, by Lemma 4.3, there are $(J - a_{d-2})$ local groups with cardinality $(\lfloor \frac{k}{J} \rfloor + 1)$ and locality $\lfloor \frac{k}{J} \rfloor$; and a_{d-2} local groups with cardinality $(\lceil \frac{k}{J} \rceil + 1)$ and locality $\lceil \frac{k}{J} \rceil$, where $a_{d-2} = k + J - J\lceil \frac{k}{J} \rceil$.

Note that according to Algorithm 1, all VNs not in the first J local groups have locality greater than or equal to the maximum locality of the first J local groups. In order to obtain a lower bound on $\overline{r}$, we assume that all the remaining $d-2$ VNs not in the first J local groups have locality equal to $\lceil \frac{k}{J} \rceil$, because for any θ, we have $\max_{j \in [1,J]} r_j \geq \lceil \frac{k}{J} \rceil$. Hence, the minimum of $\overline{r}$ is achieved if there are $m_1 := (J - a_{d-2})(\lfloor \frac{k}{J} \rfloor + 1)$ VNs with locality $\lfloor \frac{k}{J} \rfloor$ and $n - m_1$ VNs with



(a) Before the reformat process.



(b) After the reformat process.

Figure B.1: Locality Tanner graph of an (n, k, d) erasure code with m local CNs before and after the reformat process. E_i and I_i determine the number of external and internal edges of i-th local CN, respectively, where $i \in [m-1, m]$. Global CNs are not shown in this figure.

locality $\lceil \frac{k}{J} \rceil$. In other words,

$$n\bar{r} \geq m_1 \left\lfloor \frac{k}{J} \right\rfloor + (n - m_1) \left\lceil \frac{k}{J} \right\rceil. \tag{B.2}$$

If $J \mid k$, then by replacing $a = J$ and $m_1 = 0$ in (B.2), we have $\bar{r} \geq \frac{k}{J}$. If $J \nmid k$, then by replacing $\lfloor \frac{k}{J} \rfloor = \lceil \frac{k}{J} \rceil - 1$ in (B.2), we have $n\bar{r} \geq n\lceil \frac{k}{J} \rceil - m_1$. Also, recall that $m_1 = (J - a_{d-2})(\lfloor \frac{k}{J} \rfloor + 1) = (J\lceil \frac{k}{J} \rceil - k)\lceil \frac{k}{J} \rceil$. Hence, $\bar{r} \geq \lceil \frac{k}{J} \rceil - \frac{m_1}{n} = \lceil \frac{k}{J} \rceil (1 - \frac{\lceil \frac{k}{J} \rceil - k}{n})$.

B.2 Proof of Theorem 4.2

Definition B.1. *(External/internal edge). Considering Algorithm 1, the i-th local CN for $i \in [2, m]$, may have some edges linked to VNs of some other local groups $\mathcal{Y}_j$ for $j \in [1, i-1]$. We call the edges of a local CN linked to its own*

local group and other local groups internal and external links, respectively.

We represent the cardinality of the internal and external links of the i local CN by I_i and E_i, respectively. Note that the total number of the edges linked to a local CN determines the locality of the local groups corresponding with that CN. Also, the number of internal edges of the k-th local CN for $k \in [1, m]$ is $|\mathcal{Y}_k|$. Therefore,

$$r_k + 1 = I_k + E_k = |\mathcal{Y}_k| + E_k. \tag{B.3}$$

Definition B.2. *(Joint/disjoint VNs). We call a VN of a local group adjacent to more than one local CN a joint VN. If a VN is adjacent to only one local CN, we call it a disjoint VN.*

Lemma B.2. *For an (n, k, d) code with $R > (1 - \frac{1}{\sqrt{n}})^2$, we have $d - 3 < \left\lceil \frac{k}{n-k-d+2} \right\rceil = \left\lceil \frac{k}{J} \right\rceil$.*

Proof. We have

$$\begin{aligned}
& d - 3 < \left\lceil \frac{k}{n-k-d+2} \right\rceil \\
\Leftrightarrow\ & (n-k-d+2)(d-3) < k \\
\Leftrightarrow\ & (d-2)^2 - (n-k+1)(d-2) + n > 0 \\
\Leftrightarrow\ & (\frac{n-k+1}{2})^2 - (n-k+1)(\frac{n-k+1}{2}) + n > 0 \\
\Leftrightarrow\ & 4n - (n-k+1)^2 > 0 \\
\Leftrightarrow\ & \frac{k}{n} > (1 - \frac{1}{\sqrt{n}})^2,
\end{aligned}$$

where the forth inequality is true because the second degree equation $x^2 - (n - k + 1)x + n$ takes its minimum at $x = \frac{n-k+1}{2}$. □

Lemma B.3. *For any (n, k, d) locally repairable code $\mathcal{C}$ with $R > (1 - \frac{1}{\sqrt{n}})^2$ and locality Tanner graph $\mathcal{T}_l$, let local groups $\mathcal{Y}_1$ to $\mathcal{Y}_m$ of $\mathcal{T}_l$ be formed by Algorithm 1. Also, let the total number of local CNs be at least $J + 2$ (i.e., $m \geq J + 2$). Then, the total number of external edges of $(m-1)$-th local CN is greater than cardinality of the last local group, i.e., $E_{m-1} > |\mathcal{Y}_m|$.*

Proof. First, we show that $|\mathcal{Y}_{m-1}| + |\mathcal{Y}_m| \leq \lceil \frac{k}{J} \rceil$. Observe that the first J local CNs cover at least $J + k = n - (d-2)$ VNs by Lemma 4.2. Hence, local groups $\mathcal{Y}_{J+1}$ to $\mathcal{Y}_m$ cover at most $d-2$ VNs. According to the assumption $R > (1 - \frac{1}{\sqrt{n}})^2$. Hence, by Lemma B.2, $d - 2 \leq \lceil \frac{k}{J} \rceil$. Therefore,

$$|\mathcal{Y}_{m-1}| + |\mathcal{Y}_m| \leq \sum_{i=J+1}^{m} |\mathcal{Y}_i| \leq d - 2 \leq \left\lceil \frac{k}{J} \right\rceil \tag{B.4}$$

Next, we show that the number of external edges of the $(m-1)$-th local CN (E_{m-1}) is at least $\lceil \frac{k}{J} \rceil + 1 - |\mathcal{Y}_{m-1}|$. From (B.3),

$$r_{m-1} + 1 = E_{m-1} + |\mathcal{Y}_{m-1}| \Rightarrow E_{m-1} \geq r_{m-1} + 1 - |\mathcal{Y}_{m-1}|. \tag{B.5}$$

Also by Algorithm 1, locality of the local group $\mathcal{Y}_{m-1}$ is at least equal to the maximum locality of local groups one to J, i.e., $r_{m-1} \geq \max_{j \in [1,J]} r_j$. The maximum locality of the first J local groups is at least $\lceil \frac{J+k}{J} \rceil = \lceil \frac{k}{J} \rceil + 1$ because the first J local CNs cover at least $J + k = n - (d-2)$ VNs by Lemma 4.2, i.e., $\max_{j \in [1,J]} r_j \geq \lceil \frac{k}{J} \rceil + 1$. Therefore, considering (C.2) and (B.5),

$$\begin{aligned} E_{m-1} &\geq r_{m-1} + 1 - |\mathcal{Y}_{m-1}| \geq \max_{j \in [1,J]} r_j + 1 - |\mathcal{Y}_{m-1}| \\ &\geq \left\lceil \frac{k}{J} \right\rceil + 1 - |\mathcal{Y}_{m-1}| \geq \left\lceil \frac{k}{J} \right\rceil + 1 - (\left\lceil \frac{k}{J} \right\rceil - |\mathcal{Y}_m|) \\ &= |\mathcal{Y}_m| + 1. \end{aligned} \tag{B.6}$$

□

Lemma B.4. *The minimum average locality ($\overline{r}$) of an (n, k, d) code $\mathcal{C}$ with $R > (1 - \frac{1}{\sqrt{n}})^2$ is achieved if the number of local CNs corresponding with the locality Tanner graph of $\mathcal{C}$ is either J or $J + 1$.*

Proof. First, by using Algorithm 1, the set of n VNs $\{y_1, \cdots, y_n\}$ of $\mathcal{C}$ is partitioned into m local groups $\mathcal{Y}_1$ to $\mathcal{Y}_m$, where $m \in \{1, \cdots, J, J+1, \cdots, n-k\}$. Now, we consider the following two cases: (a) $m \in [1, J]$, and (b) $m \in$

$[J+1, n-k]$. In the following, we show that the minimum of $\overline{r}$ is achieved for $m = J$ and $m = J+1$ for the first case and the second case, respectively.

Case (a) $m \in [1, J]$: In this case,

$$n\overline{r} = \sum_{i=1}^{m} |\mathcal{Y}_i| r_i \geq \sum_{i=1}^{m} |\mathcal{Y}_i|(|\mathcal{Y}_i| - 1) = \sum_{i=1}^{m} |\mathcal{Y}_i|^2 - \sum_{i=1}^{m} |\mathcal{Y}_i| \tag{B.7}$$

Observe that $\sum_{i=1}^{m} |\mathcal{Y}_i| = n, \ \forall m \in [1, J]$ because all the m local groups have to cover all n VNs. By Lemma B.1, $\sum_{i=1}^{m} |\mathcal{Y}_i|^2$ is a non-increasing function; hence, (B.7) takes its minimum at $m = J$.

Case (b) $m \in [J+1, n-k]$: In this case, we show that combining local groups $J+1$ to m (i.e., $\mathcal{Y}_i$ for $i \in [J+1, m]$) and making them into a single local group ($\mathcal{Y}_{J+1}$) does not increase the average locality of the code. In order to show this, we first define a "reformat" process as follows.

Assume that $m \geq J+2$, $|\mathcal{Y}_m| \geq 1$, and a variable node $y_e \in \mathcal{Y}_1 \bigcup \cdots \bigcup \mathcal{Y}_{m-2}$ is linked to m-th local CN via edge l_e (Fig. B.1a)[1]. Then, considering Fig. B.1, we define the following two-phase procedure the "reformat" process.

Phase 1: remove one VN of the last local group, say y_f, and add it to $(m-1)$-th local group.

Phase 2: connect variable node y_e to $(m-1)$-th local check node and remove edge l_e.

In the following, we show that the reformat process (i) satisfies the VNs coverage presented in Lemma 4.2; (ii) does not increase the average locality; and (iii) can be done until all the VNs of the local groups $J+1$ to m are made into a whole ($\mathcal{Y}_{J+1}$).

Firstly, we show that after performing the reformat process, the VNs coverage presented in Lemma 4.2 is still satisfied. Note that the coverage of local CNs 1 and $m-2$ remains intact during the reformat process. Now, we show

[1]Note that there is always such a link because otherwise, $r_m < r_J$. To see this, note that $\sum_{i=J+1}^{m} |\mathcal{Y}_i| \leq d - 2 \leq \lceil \frac{k}{J} \rceil$. This implies that if m-th local check node is linked only to all variables of $\mathcal{Y}_{J+1}$ to $\mathcal{Y}_{m-1}$, then, r_m is at most $\lceil \frac{k}{J} \rceil - 1$, which is less than $r_J = \lceil \frac{k}{J} \rceil$.

that the coverage of local CNs $m-1$ and m does not change after the reformat process. The coverages of local CNs $m-1$ and m are $r_{m-1}+1$ and r_m+1, respectively, where r_i is the locality of the i-th local group for $i \in \{m-1, m\}$. From (B.3), $r_{m-1}+1 = E_{m-1}+|\mathcal{Y}_{m-1}|$ and $r_m+1 = E_m+|\mathcal{Y}_m|$. After performing the reformat process, $|\mathcal{Y}_{m-1}|$ and E_m increase by one; and $|\mathcal{Y}_m|$ and E_{m-1} decrease by one. Therefore, the coverages of local CNs $m-1$ and m—which are $r_{m-1}+1$ and r_m+1, respectively—are preserved. This implies that the coverage conditions presented in Lemma 4.2 are satisfied.

Secondly, we show that performing the reformat process does not increase the average locality. As mentioned, the localities of the local groups $m-1$ and m, which are r_{m-1} and r_m, respectively, do not change after the format process. By adding a VN from $\mathcal{Y}_m$ to $\mathcal{Y}_{m-1}$, we have

$$|\mathcal{Y}_{m-1}|r_{m-1} + |\mathcal{Y}_m|r_m \geq (|\mathcal{Y}_{m-1}|+1)r_{m-1} + (|\mathcal{Y}_m|-1)r_m,$$

which is true because $r_{m-1} \leq r_m$ according to Algorithm 1. Therefore, the reformat process does not increase the sum locality of the VNs in local groups $\mathcal{Y}_{m-1}$ and $\mathcal{Y}_m$. Also, note that the reformat process does not affect locality of all the local groups one to $m-2$. Hence, the average locality of the code does not increase after performing the reformat process.

Finally, we show that the reformat process can be done until all the VNs of the local groups $J+1$ to m are made into a whole ($\mathcal{Y}_{J+1}$), which concludes the proof. By Lemma (B.3), $E_{m-1} \geq |\mathcal{Y}_m|$, which means that for every VN removed from $\mathcal{Y}_m$, there exists an external edge corresponding with the $(m-1)$-th local CN to be added to the m-th local CN. □

Lemma B.5. *Assume that the total number of local CNs of the locality Tanner graph of an (n, k) LRC with the minimum distance d is strictly greater than J (i.e., $m > J$). Then, each local group $\mathcal{Y}_i$ has at least $|\mathcal{Y}_i| - (d-2)$ joint VNs if $|\mathcal{Y}_i| > d-2$ for $i \in [1, m]$.*

Proof. By contradiction, assume that the i-th local CN with $|\mathcal{Y}_i| > d-2$ has less than $|\mathcal{Y}_i| - (d-2)$ joint VNs. Then, the local group $\mathcal{Y}_i$ has at least $d-1$ disjoint VNs. Hence, by removing i-th local CN, the rest $m-1 \geq J$ local CNs cover at most $n-(d-1)$ VNs. This contradicts Lemma 4.2 which states that in the Tanner graph of a code with the minimum distance d, every J local CNs must cover at least $J+k = n-(d-2)$ VNs. □

Proof of Theorem 4.2: By using Algorithm 1, first, the set of n VNs $\{y_1, \cdots, y_n\}$ is partitioned into m local groups $\mathcal{Y}_1$ to $\mathcal{Y}_m$. By Lemma B.4, the minimum average locality ($\overline{r}$) is achieved if m is either J or $J+1$.

By Lemma 4.2, the first J local CNs must cover at least $J+k$ VNs, i.e., $\sum_{i=1}^{J} |\mathcal{Y}_i| \in [J+k, n]$. Observe that $\sum_{i=1}^{J} |\mathcal{Y}_i| = n$ is equivalent to $m = J$. For the sake of notational simplicity, let us define $\theta := |\mathcal{Y}_{J+1}| \in [0, d-2]$ which is the number of VNs covered by $(J+1)$-th local CN. We have

$$\begin{aligned} n\overline{r} &= \sum_{i=1}^{J+1} |\mathcal{Y}_i| r_i = \sum_{i=1}^{J} |\mathcal{Y}_i| r_i + \theta r_{J+1} \\ &\geq \sum_{i=1}^{J} |\mathcal{Y}_i| (|\mathcal{Y}_i| - 1) + \theta r_{J+1} \\ &\geq \sum_{i=1}^{J} |\mathcal{Y}_i|^2 - \sum_{i=1}^{J} |\mathcal{Y}_i| + \theta r_{J+1}, \end{aligned} \tag{B.8}$$

where $\sum_{i=1}^{J} |\mathcal{Y}_i| = n - \theta$ and $\theta \in [0, d-2]$.

By Lemma B.5, the i-th local group with $|\mathcal{Y}_i| > d-2$ has at least $|\mathcal{Y}_i| - (d-2)$ joint VNs. Assume that among the first J local groups, there are $a \in [1, J]$ local groups, indexed by $\mathcal{A} \subseteq [1, J]$, with $|\mathcal{Y}_i| > d-2$, and $J-a$ local groups

with $|\mathcal{Y}_i| \leq d-2$. Then, the total number of joint VNs, say Q, is at least

$$\begin{aligned} Q &= \sum_{i \in \mathcal{A}} |\mathcal{Y}_i| - a(d-2) \\ &= n - \theta - \sum_{i \in [1,J] \setminus \mathcal{A}} |\mathcal{Y}_i| - a(d-2) \\ &\geq n - \theta - (J-a)(d-2) - a(d-2) \\ &= n - \theta - J(d-2). \end{aligned} \tag{B.9}$$

In order to minimize the average locality, observe that all the joint VNs of the locality Tanner graph have to be connected to the local CN corresponding with the local group with the smallest cardinality. For now, let us assume that $(J+1)$-th local group with cardinality $\theta = |\mathcal{Y}_{J+1}|$ has the minimum cardinality among all the local groups one to $J+1$, where $\theta \in [0, \lceil \frac{k}{J} \rceil]$. By linking all the joint VNs obtained in (B.9) to local group $J+1$, we have

$$r_{J+1} \geq n - \theta - J(d-2) + \theta - 1 = n - J(d-2) - 1. \tag{B.10}$$

By replacing (B.10) in (B.8), we have:

$$n\ \overline{r} \geq \sum_{i=1}^{J} |\mathcal{Y}_i|^2 - \sum_{i=1}^{J} |\mathcal{Y}_i| + \theta(n - J(d-2) - 1),\ \theta \in [0, J] \tag{B.11}$$

where $\sum_{i=1}^{J} |\mathcal{Y}_i| = n - \theta$. Hence, for $\theta \in [0, d-2]$

$$\begin{aligned} n\overline{r} &\geq \sum_{i=1}^{J} |\mathcal{Y}_i|^2 - (n-\theta) + \theta(n - J(d-2) - 1) \\ &= \sum_{i=1}^{J} |\mathcal{Y}_i|^2 - J\theta(d-2) + n\theta - n. \end{aligned} \tag{B.12}$$

By Lemma 4.3, we have

$$\overline{r} \geq \frac{\min\limits_{\theta\in[1,d-2]}\left\{(J-a_\theta)\left\lfloor\frac{n-\theta}{J}\right\rfloor^2 + a_\theta\left\lceil\frac{n-\theta}{J}\right\rceil^2 + (n-dJ+2J)\theta\right\}}{n} - 1. \tag{B.13}$$

Now, we verify the following two points in order to conclude the proof.

Firstly, we show that by connecting all the joint VNs to $(J+1)$-th local CN, every J local CNs cover at least $k+J$ VNs. Note that the first J local CNs cover at least $J+k$ VNs. Also, each of the local CNs one to J has at least $|\mathcal{Y}_i|-(d-2)$ VNs connected to the $(J+1)$-th local CN. Therefore, every $J-1$ local CNs picked from the first J local CNs plus the last local CN cover at least $(n-\theta)-|\mathcal{Y}_i|+(|\mathcal{Y}_i|-(d-2)+\theta)=n-(d-2)=k+J$ VNs. Hence, Lemma 4.2 is satisfied.

Secondly, we show that if a local group other than $\mathcal{Y}_{J+1}$ has the minimum cardinality, then the average locality of the code does not decrease. Assume that local group $\mathcal{Y}_s$ has the minimum cardinality, where $s\in[1,J]$. Therefore, $|\mathcal{Y}_s|<\theta$. In this case, we have the following set of local groups: $\{\mathcal{Y}_i \mid i\in[1,J]\setminus\{s\}\}$, $\mathcal{Y}_s$, and $\mathcal{Y}_{J+1}$, where $|\mathcal{Y}_s|\leq\theta$ and $|\mathcal{Y}_{J+1}|=\theta$. Recall that the minimum average locality corresponding with this scenario has been already obtained because in (B.13), we have assumed that $\theta\in[0,d-2]$.

Appendix C

Proofs for Chapter 5

C.1 Proof of Theorem 5.1

Definition C.1. *(Independent check nodes). A set of check nodes are called independent if their corresponding rows in the parity check matrix are independent.*

Lemma C.1. *Let $\mathcal{Y}_e$ be a set of variable nodes. Then, check nodes in $\Phi(\mathcal{Y}_e)$ are independent.*

Proof. Suppose ξ is a check node that has the maximum degree among all the check nodes in $\Phi(\mathcal{Y}_e)$. There must be a variable node adjacent to ξ which is not adjacent to any other check nodes in $\Phi(\mathcal{Y}_e)$, as otherwise $\Phi(\mathcal{Y}_e)\backslash\{\xi\}$ will also be a locality-defining set of $\mathcal{Y}_e$ which contradicts the minimality of $\Phi(\mathcal{Y}_e)$. Consequently, the row of parity check matrix corresponding to the check node ξ cannot be a linear combination of rows corresponding to check nodes in $\Phi(\mathcal{Y}_e)\backslash\{\xi\}$. □

Let $\mathcal{C}$ be any (n, k, d) LRC code. Consider a locality Tanner graph of $\mathcal{C}$, and let $\mathcal{Y}_{inf}$ denote the set of information nodes. Let $\mathcal{Y}_i$, $i \in [1, s]$ denote the local groups corresponding to the check nodes in $\Phi(\mathcal{Y}_{inf})$, where $s = |\Phi(\mathcal{Y}_{inf})|$. Note that for every set of variable nodes $\mathcal{Y}_e$, we have $|\Phi(\mathcal{Y}_e)| \leq |\mathcal{Y}_e|$. Therefore,

$s \leq k$. Furthermore, the check nodes in $\Phi(\mathcal{Y}_{inf})$ must cover at least $s+k$ variable nodes. It is because, by Lemma C.1, the set of check nodes in $\Phi(\mathcal{Y}_{inf})$ are independent. Therefore, if the check nodes in $\Phi(\mathcal{Y}_{inf})$ cover less than $s+k$ variable nodes, then the dimension of the code $\mathcal{C}$ restricted to the coverage of $\Phi(\mathcal{Y}_{inf})$ will be strictly less than k, which is a contradiction.

We continue the proof, by considering two cases: 1) $s \leq J$, and 2) $s \geq J$.First, suppose $s \leq J$. By the above argument, we have $\sum_{i\in[1,s]} |\mathcal{Y}_i| \geq k+s$. Let us denote by x_i the number of information nodes whose locality is defined by local check node indexed by i, where $i \in [1,s]$. We have

$$\begin{aligned} k\bar{r}_{inf} &= \sum_{i\in[1,s]} x_i r_i = \sum_{i\in[1,s]} x_i(|\mathcal{Y}_i|-1) \geq \sum_{i\in[1,s]} x_i^2 \\ &\geq \min_{x_i} \sum_{i\in[1,s]} x_i^2, \text{ where } \sum_{i\in[1,s]} x_i = k, \end{aligned} \tag{C.1}$$

where the first inequality is because each local group $\mathcal{Y}_i$ has at least one parity node, i.e., $x_i \leq |\mathcal{Y}_i| - 1$. Note that $\sum_{i\in[1,s]} x_i = k$ for $s \in [1,J]$. Therefore, by Lemma B.3, the minimum of $\sum_{i\in[1,s]} x_i^2$ is obtained when $s = J$. Also, by Lemma 4.3, $\sum_{i\in[1,J]} x_i^2$ is minimized when x_i's are almost equal (i.e., they differ by at most one unit). Consequently,

$$k\bar{r}_{inf} \geq \min_{x_i} \sum_{i\in[1,J]} x_i^2 \geq (J-a)\left\lfloor \frac{k}{J} \right\rfloor^2 + a\left\lceil \frac{k}{J} \right\rceil^2, \tag{C.2}$$

where

$$a = k + J - J\left\lceil \frac{k}{J} \right\rceil. \tag{C.3}$$

Therefore,

$$\bar{r}_{inf} \geq \left\lceil \frac{k}{J} \right\rceil \left(2 - \frac{\lceil k/J \rceil - 1}{k/J}\right) - 1. \tag{C.4}$$

Now, suppose $s \geq J$. Without loss of generality, we assume that locality of local groups $\mathcal{Y}_1$ to $\mathcal{Y}_s$ are r_1 to r_s, respectively, where $r_1 \leq r_2 \leq \cdots \leq r_s$. In

what follows, we show that the minimum of $\bar{r}_{inf}$ is achieved if $s = |\Phi(\mathcal{Y}_{inf})| = J$.

To begin with, for a fixed construction of the locality Tanner graph, we define the procedure of exchanging one parity node of a local group $\mathcal{Y}_i$ with an information node of a local group $\mathcal{Y}_j$ a *parity replacement procedure*, where $i \in [1, J]$ and $j \in [J+1, s]$. Note that the parity replacement procedure does not increase the average locality of the information blocks because $r_1 \leq r_2 \leq \cdots \leq r_s$, which implies that local groups $\mathcal{Y}_{J+1}$ to $\mathcal{Y}_s$ have locality more than or equal to local groups $\mathcal{Y}_1$ to $\mathcal{Y}_J$. By Remark 4.1, in the Tanner graph of any (n, k, d) linear block code, every J check nodes must cover at least $J+k$ variable nodes. Therefore, it is possible to repeat the parity replacement procedure until all the k information nodes are moved to the first J local groups. Now, assuming that all the information nodes are placed in the first J local groups, i.e., $\sum_{i\in[1,J]} x_i = k$, we obtain the minimum average locality of the information symbols. By considering non-overlapping local groups as well as the minimum coverage of J local check nodes $(k + J)$, we have

$$\begin{aligned} k\bar{r}_{inf} &= \sum_{i\in[1,J]} x_i r_i = \sum_{i\in[1,J]} x_i(|\mathcal{Y}_i| - 1) \geq \sum_{i\in[1,J]} x_i^2 \\ &\geq \min_{x_i} \sum_{i\in[1,J]} x_i^2, \text{ where } \sum_{i\in[1,J]} x_i = k, \end{aligned} \tag{C.5}$$

which is the same as the second term obtained in $(C.2)$. Therefore, we again get (C.4). Considering (C.4) and noting that the minimum locality of a variable node is one, Theorem 5.1 is proved.

C.2 Proof of Proposition 5.1

By Remark 4.1, $\mathcal{C}_p$ has minimum distance d if every J local check nodes cover at least $J + k$ variable nodes. If $\theta = 0$, it can be easily verified by Fig. 5.1 that any J local check nodes are linked to at least $J + k$ variable nodes. Also, if $\theta \neq 0$, observe that any combination of the first $m_p - 1$ local check nodes of

size J cover at least $J+k$ variable nodes. In the following, we show that any $J-1$ local check nodes selected from the first m_p-1 plus the last local check node (indexed by m_p) also cover at least $J+k$ variable nodes. In other words, we show that the last local group ($\mathcal{Y}_{m_p}$) plus $J-1$ local groups indexed by $\mathcal{A} \in \mathit{comb}([1, m_p-1], J-1)$ are linked to at least $J+k$ variable nodes.

As shown in Fig 5.1, local check node m_p is linked to $\theta = |\mathcal{R}|$ variable nodes plus $\sum_{i\in[1,m_p-1]} l_i$, where l_i represents the total number of variable nodes from i-th local group linked to $\mathcal{Y}_{m_p}$. Let us assume that among the $J-1$ considered local check nodes indexed by $\mathcal{A}$, $J-1-a_{\mathcal{A}}$ have cardinality $\lceil \frac{k}{J} \rceil$ and the rest $a_{\mathcal{A}}$ have cardinality $\lceil \frac{k}{J} \rceil + 1$. For any $\mathcal{A}$ in set $\mathit{comb}([1, m_p-1], J-1)$, we must have

$$\begin{aligned}(J-1-a_{\mathcal{A}})\left\lceil \frac{k}{J} \right\rceil + a_{\mathcal{A}}\left(\left\lceil \frac{k}{J} \right\rceil + 1\right) \\ + \theta + \sum_{i\in[1,m_p-1]\setminus\mathcal{A}} l_i \geq J+k,\end{aligned} \tag{C.6}$$

which is equivalent to the constraints presented in (5.2). Therefore, in the Tanner graph of $\mathcal{C}_p$, every J local check nodes cover $k+J$ variable nodes and the code has minimum distance d.