Universitat Autònoma de Barcelona

Departament d'Enginyeria de la Informació i de les Comunicacions

Coding Techniques for Distributed Storage

Submitted to Universitat Autònoma de Barcelona in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

by Bernat Gastón Brasó

Cerdanyola del Vallès, September 2013

Advisors: Dr. Jaume Pujol and Dr. Mercè Villanueva

Professors at Universitat Autònoma de Barcelona

I certify that I have read this thesis entitled "Coding Techniques for Distributed Storage" and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Cerdanyola del Vallès, September 2013

Dr. Jaume Pujol
(Advisor)

Dr. Mercè Villanueva
(Advisor)

A la Mariona.

Abstract

Online data storage is often regarded as a growing business, yet many unresolved issues linger in this specific field and prevent researchers from driving it to full capacity. Data replication (most commonly known as backup) is simply not efficient when improving persistence and accessibility of such data. Error correcting codes are known for their efficiency when adding redundancy to avoid lose of information. Unfortunately, the use of error correcting codes entail additional problems such as the repair problem: how do we replace a storage node downloading as less data as possible from other nodes.

In this dissertation, we deepen on state-of-the-art of codes applied to distributed storage systems. Additionally, a family of regenerative codes which we call quasi-cyclic flexible regenerating codes is provided. Quasi-cyclic flexible minimum storage regenerating (QCFMSR) codes are constructed and their existence is well-proven. Quasi-cyclic flexible regenerating codes with minimum bandwidth constructed from a base QCFMSR code are also provided.

Quasi-cyclic flexible regenerating codes are very interesting because of their simplicity and low complexity. They allow exact repair-by-transfer in the minimum bandwidth case and an exact pseudo repair-by-transfer in the MSR case, where operations are needed only when a new node enters into the system replacing a lost one.

Finally, we propose a new model whereby storage nodes are placed in two racks. This unprecedented two-rack model is generalized to any number of racks. In this specific set-up, storage nodes have different repair costs depending on the rack where they are placed. A threshold function, which minimizes the amount of stored data per node and bandwidth needed to regenerate a failed node, is also shown. This latter threshold function generalizes those given by previous distributed storage models. Tradeoff curves obtained from this threshold function are compared with those obtained from previous models, and it is shown that this new model outperforms previous ones in terms of repair cost.

Resum

Encara que l'emmagatzematge online d'informació és un negoci creixent, no està exempt de problemàtiques, una d'elles és la persistència i accessibilitat de les dades. Cal replicar les dades de manera que si es perd una còpia no es perdi la informació de forma definitiva. Malauradament, la replicació de dades (coneguda com a "backup") no és una solució eficient, ja que introdueix molta redundància que provoca sobre costos. Els codis correctors d'errors són coneguts per augmentar la persistència i l'accessibilitat de les dades minimitzant la redundància necessària. Però el seu us introdueix altres problemes com l'anomenat "repair problem": com substituir un node d'emmagatzematge descarregant el mínim de dades dels altres nodes.

En aquesta dissertació, estudiem l'estat de l'art pel que fa als codis aplicats a sistemes d'emmagatzematge distribuïts, com per exemple el "cloud storage". També ens introduïm al "repair problem" des de la vessant més aplicada, usant topologies de sistemes reals com els "data centers".

Concretament, aportem una família de codis regeneratius que anomenem quasi-cyclic flexible regenerating codes i que es caracteritza per minimitzar l'ús de recursos computacionals en el procés de regeneració d'un node. Alhora, aquesta solució minimitza les dades emmagatzemades i l'ample de banda necessari per regenerar un node que falla.

També estudiem el cas en que els costos de descàrrega de les dades no són homogenis. En concret, ens centrem en el cas dels racks, on els nodes d'emmagatzematge estan distribuïts en racks, i el cost de descàrrega de dades dels nodes en el mateix rack és molt menor que el cost de descàrrega de dades dels nodes en un altre rack. Aquest nou model generalitza els models teòrics anteriors i ens permet comprovar que els costos poden disminuir si adaptem el model teòric a la topologia concreta del sistema d'emmagatzematge distribuït.

Acknowledgements

A la meva família, que em va animar a involucrar-me en aquest projecte, que sempre ha confiat en mi i que ha fet l'impossible per a que pogués arribar allà a on volia.

Als meus directors, la Mercè Villanueva i el Jaume Pujol. Per les nostres interminables reunions discutint sobre grafs, fluxos i optimitzacions. Per ajudar-me amb les parts més matemàtiques i per ensenyar-me que la comunicació és tant o més important que la recerca. També per recolzar-me en els pitjors moments, quan els resultats no arriben, tot està emboirat i trobar el camí de sortida sembla una utopia.

A tot el departament d'enginyeria de la informació i les comunicacions de la Universitat Autònoma de Barcelona. Als que hi són i als que ja han marxat, amb qui hem compartit reunions, cafès, congressos i fins hi tot equip de futbol.

I would like to express my hearthfelt gratitude to Prof. Dimakis and staff at the University of Southern California who aided me to gain a greater insight into this field. Also, special thanks to Marco Levorato, who generously supported me during my stay at the mentioned research center.

I sobretot al meu amor, qui coneix millor que ningú el que hem passat en aquest temps, els bons i els no tant bons moments. Per animar-me i per confiar en mi, i sobretot per estimar-me tant.

Contents

Abstract vii

Resum ix

Acknowledgements xi

Chapter 1 Introduction 1

Chapter 2 Background 5
2.1 Coding theory 5
2.1.1 Galois fields 6
2.1.2 Coding preliminaries 7
2.2 Network distributed storage systems 13
2.3 Codes and NDSS 16
2.3.1 The repair problem 16
2.3.2 Important properties inherited from codes 18
2.4 Coding techniques for NDSS 19
2.4.1 Locally reconstructible codes 19
2.4.2 Locally repairable codes 21
2.5 Network coding techniques for NDSS 22
2.5.1 Graph theory 23
2.5.2 Network coding 26
2.5.3 Homogeneous model 27
2.5.4 Non-homogeneous models 31
2.5.5 Regenerating codes 34
2.5.6 Flexible regenerating codes 37

Chapter 3 Quasi-cyclic Flexible Regenerating Codes 43
3.1 Quasi-cyclic Flexible MSR codes 43
3.1.1 Code Construction 44

3.1.2 Node Regeneration 46
3.1.3 Data Reconstruction 46
3.1.4 Example 49
3.2 Quasi-cyclic flexible regenerating codes with minimum bandwidth 50

Chapter 4 Two-rack model **55**
4.1 The model 56
4.1.1 MSR and MBR points 63
4.1.2 Non-feasible situation 64
4.1.3 Case $r_e^1\beta_e \geq r_c^2\tau\beta_e$ 65
4.2 General rack model 69
4.2.1 When $r = n-1$ 69
4.2.2 When $r \leq n-1$ 70
4.3 Analysis 73

Chapter 5 Conclusions **75**
5.1 Main Results 76
5.2 Further Research 78

List of Figures

2.1 Left: a $[2,1,2]$ repetition code applied to two coordinates. Right: a $[3,2,2]$ parity check code applied to two coordinates. 17
2.2 Bipartite graph associated with a square matrix. 24
2.3 Butterfly network using network coding. 26
2.4 Information flow graph of a $[4,2,3]$ NDSS with $r=3$. 29
2.5 Optimal tradeoff curve between α and γ for a $[10,5,6]$ NDSS with $r=9$. 30
2.6 General information flow graphs corresponding to the case $k \leq r_c+1$. 33
2.7 General information flow graphs corresponding to the case $k > r_c+1$. 33
2.8 Fragmentation, construction and regeneration of a 2-fault tolerance $[4,2,3]$ regenerating code. 36
2.9 A $[10,9,2]$ MDS code used to create a $[5,3,4]$ flexible regenerating code with minimum bandwidth. 39
3.1 Construction process for a $[n,k,r]$ quasi-cyclic flexible MSR code. 45
3.2 A $[6,3,4]$ QCFMSR code with coordinates over $\mathbb{F}_q$ and array coordinates over $\mathbb{F}_q^2$. 48
3.3 Construction of a $[6,2,2]$ quasi-cyclic flexible regenerating code with minimum bandwidth from a $[6,3,4]$ QCFMSR code. . . 51
3.4 Construction of a $[10,4,2]$ quasi-cyclic flexible regenerating code with minimum bandwidth from a $[10,5,6]$ QCFMSR code. 52
3.5 Parameters $[\bar{n},\bar{k},\bar{r}]$ for some quasi-cyclic flexible regenerating codes with minimum bandwidth constructed from $[n,k,r]$ QCFMSR codes, and comparison between the $\alpha=\gamma$ values given in [DGWR10] and the ones achieved with the proposed construction. 53
4.1 Rear of a real rack used in a data center. 55

4.2 Scheme of a two rack model. 57
4.3 Information flow graph corresponding to the rack model when $k > r_c^1 + 1$, with $k = 4$, $r_c^1 = 1$, $r_c^2 = 2$, $r = 4$ and $n_1 = n_2 = 3$. 58
4.4 Information flow graph corresponding to the rack model when $k > r_c^1 + 1$, with $k = 4$, $r_c^1 = 1$, $r_c^2 = 2$, $r = 4$ and $n_1 = n_2 = 3$. 61
4.5 Information flow graph corresponding to the rack model when $k > r_c^1 + 1$, with $k = 3$, $r_c^1 = 1$, $r_c^2 = 2$, $r = 4$ and $n_1 = n_2 = 3$. 62
4.6 Tradeoff curve for the rack model with $M = 1$, $k = 10$, $r_c^1 = 5$, $r_c^2 = 6$, $r = 11$, $n_1 = n_2 = 6$ and $\tau = 2$. 63
4.7 Information flow graph with $k = 3$, $n_1 = 2$, $n_2 = 5$, $r_c^1 = 1$, $r_c^2 = 4$ and $r = 6$. 65
4.8 Information flow graph corresponding to the rack model with $k = 7$, $r_c^1 = 1$, $r_c^2 = 2$, $r_c^3 = 3$ and $r = 8$. 72
4.9 Chart comparing the rack model with the static cost model for $M = 1$, $k = 10$, $r_c^1 = 5$, $r_c^2 = 6$, $r = 11$, $n_1 = n_2 = 6$ and $\tau = 2$. 73
4.10 Chart showing the tradeoff curves between α and β_e for $M = 1$, $k = 10$, $r_c^1 = 5$, $r_c^2 = 6$, $r = 11$ and $n_1 = n_2 = 6$, so with $k > r_c^1 + 1$. 73
4.11 Chart showing the repair cost in the rack model for $M = 1$, $k = 5$, $r_c^1 = 5$, $r_c^2 = 6$, $d = 11$, $n_1 = n_2 = 6$, $C_c = 1$ and $C_e = 10$. The points correspond to the $k = 5$ values given by $f(i)$, $i = 0, \ldots, 4$. 74

Chapter 1

Introduction

Since the beginning of time, humans have had a keen desire to explain activities, thoughts, discoveries, customs or any other information that we think would be relevant now or in the future. This desire has led us to find ways of representing such information.

The process of representing information requires three elements: the information, the technique used to represent it and the object used to store it. Prehistoric paintings are information about activities done by prehistoric humans, this information is represented by paintings and it is stored in stone. Nowadays digital cameras capture images of a specific instant of time, represent them using bytes, and store them in a digital storage device, for example a solid state card.

The second law of thermodynamics says that entropy always increases, which means that all the elements used for storage wear out as time passes, and the information stored in them will irretrievably be lost. From prehistoric paintings drawn on stone to digital newspaper, one of the main goals is to keep the information available over the maximum possible amount of time. If the information is lost, there is no way of recovering it, so we have to prevent this situation. For example, old prehistoric paintings are usually restored by experts, and digital data is replicated to tolerate storage device failures.

In this research, we focus on digital information. We assume that it is relevant enough to keep it available over time, and that it is stored using a digital storage device.

To approximate the reader to the big picture of digital data storage, it is interesting to know that in 2002, humanity started to store more information on digital than on analog storage devices. In 2007, Intl. Data Corp (IDC) estimated that the amount of information created, captured, or replicated

exceeded available storage for the first time. In the same year, the world's stored information was 295 exabytes, which corresponds approximately to two stacks of CDs stretching from the earth to the moon. Moreover, the amount of stored information is doubling roughly every 3 years.

The increasing use of the Internet, the appearance of lots of devices that use it (such as tablets or mobile phones), and globalization has changed the computer science paradigm in many senses. From the storage point of view, users want their information to be available from anywhere easily and instantly. For example, a user edits his data using a personal computer and wants this data to be available on his mobile phone immediately, so he can send it to his friends. Or a user wants to share a document with his work-group, so they can edit it without the mess that different versions, emails and crossed editions cause.

Network Distributed Storage Systems (NDSS) has been proposed as the main tool to store and manage information. NDSS is based on storing the data in devices which are connected through a network. The data stored in an NDSS can be accessible from anywhere with an Internet connection, edition is instantaneously applied and persistence is assured. Examples of NDSS are everywhere, email web based applications like gmail, shared document applications like Google Drive [Inc12], network storage applications like Dropbox [Inc07] or BitTorrent file sharing application [Coh09].

In general, one can divide NDSS into two big families: peer-to-peer (P2P) applications and data centers. P2P applications are based on sharing files between users. Firstly, a user has a file and shares it using a P2P application. Then, other users can download the file, store it, and share it with others. The more users have replicas of the file, the more the file is available, because if one user goes offline, the file is still accessible via the other replicas. Moreover, each file is split into pieces, so it is possible to parallelize the download process, decreasing the time needed to download the file and increasing the availability of the file.

Data centers are probably the most usual NDSS nowadays. Data centers are physical buildings keeping lots of storage devices usually organized in racks, metallic supports designed to hold electronic equipment. Each data center is typically keeping thousands of racks, each one keeping dozens of storage devices. Most of the biggest information technology companies like Google, Yahoo or Microsoft have their own data centers.

Coding theory has been proposed as the latest evolution to keep all the stored information persistent over time while adding as little redundancy as

possible in an NDSS. Nowadays, it is already assumed that replication (backups) is not a realistic option for NDSS because it is not scalable. However, coding theory does not address some of the problems that NDSS has to face, simply because coding theory was designed with a different purpose.

Many current data centers still use replication because of the drawbacks that classical coding theory introduce, especially the overhead in the bandwidth used to repair one device. However, because of the recent improvements in this field, companies are slowly introducing coding theory in their NDSS. The implementation of an upper layer to the Hadoop Distributed File System [Apa12] by Facebook, and the latest Google File System are examples of applications for NDSS where coding theory is used.

Storage devices have an approximate failure probability per year between 2% and 4%. In a data center, a file is distributed in between 3 and 14 storage devices approximately. Each data center stores hundreds of thousands of storage devices. This means that the failure of a storage device is a common occurrence, but simultaneously losing another storage device containing the same files is very improbable. In this dissertation, we address the problem of replacing a single storage device in NDSS using coding theory.

In the following two paragraphs, the goals of this dissertation are explained. The first goal of the dissertation is to study the application of regenerating codes in real environments and construct, if possible, a family of realistic regenerating codes. Nowadays, codes are increasingly used in real applications. However, regenerating codes are still in the theoretical environment. In this dissertation, we have studied the regenerating codes from a practical point of view and we have proposed a new family of regenerating codes specially suitable for real environments. The results of this study have been published in [GPV11b], [GPV11a] and [GPV13a].

The second goal is to study how the specific topology of a distributed storage system can be used to improve the performance of the regenerating codes. In this direction, we have studied the data centers and their rack based topology and we have developed a model to represent these data centers. Using this model, we have realized that the use of the specific topology allows us to decrease the theoretical limit of regenerating codes. The results of this study have been published in [GPV13b] and [GPV13c].

This memory is organized as follows. In chapter 2, we review the basic concepts of coding theory and graph theory that have been used to develop the contributions exposed in the following chapters. In chapter 3, we expose our first contribution, the creation of a family of regenerating codes specially suitable for being applied in a practical environment. We call this

family quasi-cyclic flexible regenerating codes because they flexibilize some restrictions of regenerating codes which make them more suitable for realistic environments. In chapter 4, we expose our second contribution, the rack model. This model uses the rack based topology of data centers to provide a better performance on the regenerating codes in these kind of architectures. Finally, in chapter 5, we expose our conclusions. We also discuss the contributions of this dissertation and we provide some further lines of research.

Chapter 2

Background

In this chapter, we introduce the basics of coding theory, network distributed storage systems and coding theory applied to network distributed storage systems (NDSS). Firstly, in Section 2.1, we explain some of the most important concepts of coding theory. Then, in Section 2.2, we introduce the Network Distributed Storage Systems. In Section 2.3, we explain the most important concepts that appear when coding theory techniques are applied to NDSS. Finally, we show the two most known approaches of using coding theory in an NDSS: techniques based on designing codes for NDSS are seen in Section 2.4 and network coding techniques combined with codes are seen in Section 2.5.

2.1 Coding theory

On the one hand, from the information theory point of view, the information is defined as the set of symbols which compose the message. On the other hand, the data is composed of the information plus the redundancy. The redundancy is the set of symbols which do not add information to the message.

Coding theory [MS77] is a well known mathematical theory introduced by Shannon [Sha49]. The main goal of coding theory is to produce redundancy for a given information by using codes, a mapping from a set of information symbols to a set of information and redundancy symbols. This redundancy added is used to correct errors (symbols that have changed their value) or erasures (symbols that have been erased). Moreover, such redundancy should be produced with three main goals: reduce the amount of redundancy needed,

increase the capacity to error/erasure correction and reduce the computational complexity of the algorithms used.

Codes transform a set of information symbols into codewords composed of the information symbols and the redundancy symbols, and this process is called encoding. Each one of the symbols of a codeword is called coordinate. Then, the codewords are sent through a noisy channel which may produces errors and/or erasures changing and/or erasing some coordinates. Finally, the received symbols are decoded at the output of the channel to obtain the original set of information symbols.

There are a lot of applications using coding theory techniques for transmission like ADSL+ [GDJ05], satellite communications [Eva08] or TCP/IP protocol [For81] among others. There are also coding theory techniques applied to storage applications like CD/DVD [Imm94] or RAID systems [PGK88].

2.1.1 Galois fields

Finite fields [LN96], also called Galois fields in honor of the mathematician Evariste Galois, are algebraic structures which contain a finite number of elements (symbols) and in which the operations of addition, subtraction, multiplication and division (except by zero) between any two elements of the field, result in another element of the field.

Let p be a prime number and let $q = p^m$, $m \geq 1$. We denote by $\mathbb{F}_q$, the finite field of q elements. Moreover, $\mathbb{F}_q$ contains the subfield $\mathbb{F}_p$, and it is a vector space over $\mathbb{F}_p$ of dimension m, unique up to isomorphism.

Let $f(x)$ be an irreducible polynomial of degree m in $\mathbb{F}_p[x]$. Then, $\mathbb{F}_p[x]/(f(x))$ is a finite field with p^m elements. Therefore, the elements of any finite field $\mathbb{F}_q$ can be seen as polynomials over $\mathbb{F}_p$ of degree less than m.

Example 1 (The binary field)**.** *The binary field $\mathbb{F}_2$ can be constructed from $\mathbb{Z}_2[x]/(x)$ and contains two elements, that is, $\mathbb{F}_2 = \{0, 1\}$.*

Example 2 (The field $\mathbb{F}_{2^2}$)**.** *The polynomial $f(x) = x^2 + x + 1$ is irreducible in $\mathbb{F}_2[x]$. The finite field $\mathbb{F}_4$ can be constructed from $\mathbb{F}_2[x]/(f(x))$ and their elements can be seen as polynomials of degree less than 2. This means that $\mathbb{F}_4 = \{0, 1, x, x + 1\}$.*

As we have said, an element in $\mathbb{F}_{p^m}$ can be seen as a polynomial of degree less than m with coefficients over $\mathbb{F}_p$. Any polynomial $a_0 + a_1 x + \cdots + a_{m-1}x^{m-1}$ of degree less than m can be represented by an ordered array of

m coefficients as $(a_{m-1}, \ldots, a_1, a_0)$. If $m = 2$, for example, the polynomial $x + 2$ can be represented by the array $(1, 2)$, or the polynomial 1 can be represented by $(0, 1)$. Finally, one can see the field $\mathbb{F}_{p^m}$ as an extension of the subfield $\mathbb{F}_p$, where each element of $\mathbb{F}_{p^m}$ is an array of m coefficients over $\mathbb{F}_p$. The same can be done for any subfield of a field, for example, since $\mathbb{F}_{2^2}$ is a subfield of $\mathbb{F}_{2^4}$, we can see the field $\mathbb{F}_{2^4}$ as an extension of $\mathbb{F}_{2^2}$, where each element of $\mathbb{F}_{2^4}$ is an array of two coefficients over $\mathbb{F}_{2^2}$.

2.1.2 Coding preliminaries

In this subsection, we summarize the basic concepts of coding theory [MS77]. A code over $\mathbb{F}_q$ is a map from $\mathbb{F}_q^k$ to $\mathbb{F}_q^n$ which converts vectors of k coordinates over $\mathbb{F}_q$ to vectors of n coordinates over the same field, called codewords. In this dissertation, we are only interested in linear codes, which means that the map is linear and that the set of codewords in $\mathbb{F}_q^n$ forms a subspace of $\mathbb{F}_q^n$.

Linear codes, generator and parity check matrices

Let $\mathbb{F}_q^n$ denote the vector space of all n-tuples over the finite field $\mathbb{F}_q$. An (n, M) code C over $\mathbb{F}_q$ is a subset of $\mathbb{F}_q^n$ of size M. A vector $v \in C$ is called codeword and the set of codewords is called the codebook of C.

If C is a k-dimensional subspace of $\mathbb{F}_q^n$, then C is called $[n, k]$ linear code over $\mathbb{F}_q$. Any $[n, k]$ linear code C over $\mathbb{F}_q$ has $M = q^k$ codewords, length n and dimension k. A generator matrix for an $[n, k]$ linear code C over $\mathbb{F}_q$ is any $k \times n$ matrix G, whose rows from a basis of C. Given a linear code C with generator matrix G, the encoding function from $\mathbb{F}_q^k$ to $\mathbb{F}_q^n$ can be defined by $c = vG$, where $v \in \mathbb{F}_q^k$ and $c \in C$.

For any set of k linear independent columns of a generator matrix G, the corresponding set of coordinates forms an information set for the corresponding $[n, k]$ linear code C. The remaining $n - k$ coordinates are the redundancy set of C. If the first k coordinates form an information set, the linear code has a unique generator matrix of the form $[I_k|A]$, where I_k is the $k \times k$ identity matrix. In this case, the information set is placed in the first k coordinates and the linear code C is called systematic.

Definition 1 (Transmission rate)**.** *The transmission rate, or rate of a code of length n and dimension k, is defined as $R = k/n$.*

Example 3 (The binary repetition code). *An information binary symbol $v \in \mathbb{F}_2$ can be encoded by repeating it n times. For example, if $n = 3$, the information symbol 0 is encoded by 000 and the information symbol 1 by 111. In general, a binary repetition code is an $[n, 1]$ binary linear code C with generator matrix $G = (1\ 1\ \cdots\ 1)$ and $R = \frac{1}{n}$.*

Example 4 (The binary parity check code). *An information vector $(v_1, v_2, \ldots, v_k) \in \mathbb{F}_2^k$ is encoded by adding a single parity check symbol $v_{k+1} = \sum_{i=1}^{k} v_i \in \mathbb{F}_2$. For instance, if $k = 2$ then $n = 3$, and $(v_1, v_2) \in \mathbb{F}_2^2$ is encoded as $(v_1, v_2, v_1 + v_2) \in \mathbb{F}_2^3$. In general, a single parity check code is a $[k + 1, k]$ binary linear code C with generator matrix*

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

and $R = \frac{k}{k+1}$.

If the generator matrix has the form $G = [I_k|A]$, like in the above examples, where I_k is the $k \times k$ identity matrix and A is a $k \times (n - k)$ matrix, the information vector v is placed at the beginning of the codeword $c = vG$, and the code is said to be systematic. This means that in every codeword of length n, the information is in the first k coordinates and the redundancy is in the last $n - k$ coordinates.

As a $[n, k]$ linear code C is a subspace of a vector space, it is the kernel of some linear transformation. In particular, there is an $(n - k) \times n$ matrix H, called parity check matrix for the $[n, k]$ linear code C, defined as

$$C = \{x \in \mathbb{F}_q^n \mid Hx^T = \mathbf{0}^T\}.$$

If $G = [I_k|A]$ is a generator matrix for an $[n, k]$ linear code C, then $H = [-A^T|I_{n-k}]$ is a parity check matrix for C, where A^T means the transposition of A. Let $C^\perp$ be the dual code of C, that is, $C^\perp$ is the linear code generated by the parity check matrix H. Notice that $C^\perp$ is an $[n, n - k]$ linear code. Moreover, $C^\perp$ can also be defined as $C^\perp = \{w \in \mathbb{F}_q^n \mid w \cdot c = 0,\ \forall c \in C\}$, where $w \cdot c = \sum_{j=1}^{n} w_j c_j$ denotes the ordinary inner product of vectors in $\mathbb{F}_q^n$.

Example 5 (The binary repetition code). *The parity check matrix of the binary repetition code of length 3 and dimension 1 with generator matrix*

$$G = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix},$$

is

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

Example 6 (The binary parity check code)**.** *The parity check matrix of the binary parity check code of length* 3 *and dimension* 2 *with generator matrix*

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

is

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

Note that the parity check matrix of a binary repetition code is the generator matrix of a binary parity check code and vice-versa.

Example 7 (Hamming code)**.** *The parity check matrix of the Hamming code of length* 7 *and dimension* 4 *with generator matrix*

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

is

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

We have seen that the encoding in a linear code can be done using a product matrix multiplication, involving the information vector and the generator matrix to produce a codeword. Moreover, given an $[n, k]$ linear code C and a vector v in $\mathbb{F}_q^n$, it is possible to determine whether v is a codeword of the code C by using a parity check matrix H of C. Specifically, $v \in C$ if and only if $Hv^T = \mathbf{0}^T$.

The parity check matrix can be used to detect errors in a codeword. The procedure of obtaining the information vector from the received vector is called decoding. The decoding procedure is much more complex than the encoding. In this dissertation, we are not interested in decoding algorithms, which usually depends on the specific code used [MS77]. For us, it is enough to know that the decoding is possible and that its complexity depends on the code used.

Measure concepts

Redundancy is added to correct errors and erasures introduced by the channel. In this dissertation, we only consider erasures because, as we will see in Section 2.2, NDSS is affected only by erasures.

Assume that a source sends one symbol over a noisy channel. If this symbol is erased, the receiver can not know the symbol that was sent. However, if the source uses a binary repetition code of length 3, like the one shown in Example 3, the receiver can decode the received vector even if it is affected by two erasures.

The next question to arise is, given a linear code of length n and dimension k, which is the maximum number of arbitrary coordinate erasures that the code can tolerate? In this context, tolerate means that the receiver is able to decode the received vector. In order to measure this tolerance, some concepts need to be defined.

Definition 2 (Hamming distance)**.** *The Hamming distance $d_H(u, v)$ between two vectors $u, v \in F_q^n$ is defined as the number of coordinates in which u and v differ.*

Definition 3 (Hamming weight)**.** *The Hamming weight $w_H(v)$ of a vector $v \in \mathbb{F}_q^n$ is the number of nonzero coordinates of v.*

Definition 4 (Minimum Hamming distance)**.** *The minimum Hamming distance (minimum distance) $d(C)$ of a linear code C is the minimum Hamming distance between any two different codewords $c_1, c_2 \in C$, that is,*

$$d(C) = \min_{c_1, c_2 \in C, c_1 \neq c_2} \{d(c_1, c_2)\} = \min_{c_1, c_2 \in C, c_1 \neq c_2} \{w_H(c_1 - c_2)\}.$$

Since for any $u, v \in \mathbb{F}_q^n$, $d_H(u, v) = w_H(u - v)$, if C is a linear code, the minimum distance $d(C)$ is the same as the minimum weight of the nonzero codewords of C.

A linear code C of length n, dimension k, and minimum distance $d = d(C)$ is also denoted as an $[n, k, d]$ linear code C.

Example 8. *The parity check code over $\mathbb{F}_2$ of dimension 2, so of length 3, has the following codebook $\{(0,0,0), (0,1,1), (1,0,1), (1,1,0)\}$. Therefore, its minimum distance is 2.*

A received vector may contain both *errors* and *erasures*. Let C be an $[n, k, d]$ linear code. If a codeword c is sent and the corresponding received

vector y contains t errors and e erasures, then C is capable to correct these t errors and e erasures provided that $2t + e < d$.

Since we are only interested in correcting erasures, we define the *erasure correction capability* as the maximum number of erasures that a code can correct. From now on, we will talk about erasure correction capability or correction capability indistinctly. Note that the erasure correction capability of an $[n, k, d]$ linear code C is $d - 1$.

Maximum distance separable codes

There is a relation between the redundancy of the set of codewords of a code and the minimum distance (and then the erasure correction capability) of this code. If no redundancy is added, the minimum distance is 1. If a redundancy symbol is added, the minimum distance is either 1 or 2.

Conceptually, an efficient code in terms of minimum distance (or correction capability) means a code with a fixed transmission rate and a minimum distance as higher as possible. The highest efficiency is achieved when, for each redundancy symbol added, the minimum distance is increased by one. These kind of codes are called *maximum distance separable* (MDS) codes.

Theorem 5. *[MS77] Let C be an $[n, k, d]$ linear code and let H be a parity check matrix of C. The minimum distance of C is d if and only if any set of $d - 1$ columns of H are linearly independent and some set of d columns are linearly dependent.*

Theorem 6 (Singleton bound)**.** *[MS77] Let C be an $[n, k, d]$ linear code. Then $d \leq n - k + 1$.*

Definition 7 (Maximum distance separable code)**.** *[MS77] A code is called maximum distance separable (MDS) if it achieves the Singleton bound and can tolerate (correct) up to $n - k$ erasures.*

There exist well known MDS codes, which are used in a lot of applications. An MDS code is specially suitable when the redundancy added must be minimized. Reed-Solomon codes are probably the most used MDS codes and they are based on an algebraic construction using polynomials. The main drawback of Reed-Solomon codes for data transmission is that they need a fixed set of k coordinates before encoding or decoding, and that the complexity of their decoding algorithm (in the general case) is $O(n^2)$. However, their efficiency in terms of transmission rate and their existence for a lot of parameters n and k, make them a good choice for most coding theory based applications.

Array codes

As it is explained in Subsection 2.1.1, it is possible to see a field $\mathbb{F}_{q^t}$ as an extension of the subfield $\mathbb{F}_q$, also called base field. This means that any element of $\mathbb{F}_{q^t}$ can be seen as an array of t elements over $\mathbb{F}_q$.

Recall that using an $[n, k, d]$ code over $\mathbb{F}_{q^t}$, we can encode an information vector $v \in \mathbb{F}_{q^t}^k$ and generate a codeword $c \in \mathbb{F}_{q^t}^n$. Since the symbols of this codeword can always be represented as an array of symbols over $\mathbb{F}_q$, this code is also referred to as an array code. The symbols over $\mathbb{F}_{q^t}$ of an array code are called array coordinates, while the symbols over $\mathbb{F}_q$ are called coordinates. The importance of array codes lies in that once the information vector is encoded into a codeword, the coordinates can be treated over $\mathbb{F}_q$ or over $\mathbb{F}_{q^t}$ indistinctly, depending on the properties needed at each specific moment. A well known example of an array code is the EVENODD code [TS02].

Example 9 (The EVENODD code)**.** *Assume that we want to encode a vector $v \in \mathbb{F}_{2^4}^5$ given by 5 array coordinates over $\mathbb{F}_{2^4}$ using a systematic EVENODD code of length 7 and dimension 5. Each array coordinate can be seen as an array of 4 coordinates over the base field $\mathbb{F}_2$. Therefore, we can represent the vector v as a 4×5 matrix where each column represent one array coordinate composed by the coordinates of the base field. For example, let $v = (v_1, v_2, v_3, v_4, v_5) \in \mathbb{F}_{2^4}^5$ be the vector represented by the following matrix:*

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

where the first column is v_1, the second v_2, the third v_3, the fourth v_4 and the fifth v_5.

Now, to obtain a codeword $c \in \mathbb{F}_{2^4}^7$, we need to create two array coordinates r_1 and r_2 that contain the redundancy of v. If we assume that the matrix entries are $a_{i,j}$, then the encoding defined by the EVENODD code is

$$\begin{aligned} a_{l,6} &= a_{l,1} \bigoplus a_{l,2} \bigoplus a_{l,3} \bigoplus a_{l,4} \bigoplus a_{l,5} \bigoplus, \ 1 \leq l \leq 4, \\ a_{1,7} &= S \bigoplus a_{1,1} \bigoplus a_{4,3} \bigoplus a_{3,4} \bigoplus a_{2,4} \bigoplus, \\ a_{2,7} &= S \bigoplus a_{2,1} \bigoplus a_{1,2} \bigoplus a_{4,4} \bigoplus a_{3,5} \bigoplus, \\ a_{3,7} &= S \bigoplus a_{3,1} \bigoplus a_{2,2} \bigoplus a_{1,3} \bigoplus a_{4,5} \bigoplus, \\ a_{4,7} &= S \bigoplus a_{4,1} \bigoplus a_{3,2} \bigoplus a_{2,3} \bigoplus a_{1,4} \bigoplus, \end{aligned}$$

where $S = a_{4,2} \bigoplus a_{3,3} \bigoplus a_{2,4} \bigoplus a_{1,5} \bigoplus$. Note that the encoding is given over the base field. For example, for the above information vector $v =$

$(v_1, v_2, v_3, v_4, v_5)$, the corresponding systematic codeword $(v_1, v_2, v_3, v_4, r_1, r_2)$ can be represented by the 4×7 matrix

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

where each column is an array coordinate stored in a different storage node. The first five columns are the information array coordinates and the last two columns are the redundancy array coordinates. The redundancy columns are constructed with the following encoding:

Note that the redundancy is produced by single coordinates over $\mathbb{F}_2$, but the MDS property is given by the code seen over $\mathbb{F}_{2^4}$.

2.2 Network distributed storage systems

Definition 8 (Storage node)**.** *A storage node is a network element that unites one or more physical storage devices to provide a simple block storage service. Such term can include different elements such as desktops and laptop computers, network attached storage (NAS) devices, set-top boxes (STB) or storage components from data centers [PJ11].*

Definition 9 (Network distributed storage system (NDSS))**.** *A network distributed storage system is a distributed computer system composed of multiple autonomous storage nodes that communicate through a computer network. The aim of a network distributed storage system is to integrate all theses storage nodes into a single and uniform data storage service that applications and users can access through a communication network [PJ11].*

From a formal point of view, an $[n, k, d]$ NDSS is a $(d-1)$-fault tolerance system composed of n storage nodes and where a subset of k storage nodes contain enough information to recover the file.

Definition 10 (Network bandwidth)**.** *The network bandwidth is a measurement for communication resources expressed in data units per time unit, for example bits per second.*

There are NDSS where the network bandwidth has no importance, for example the Redundant Array of Independent Disks (RAID) [PGK88]. RAID is a storage technology that combines multiple disk drive components into

one logical unit. It can be designed for two main proposes: increase the input/output (i/o) speed by parallelizing these operations into multiple independent disks, or increase the availability of the stored data by using coding theory techniques. In RAID, the bandwidth limitations introduced by the buses composing the communication network can be considered negligible compared with the latency introduced by the i/o operations.

However, there are other NDSS where the network bandwidth is limited and its reduction is a desired goal. The communication network of this kind of NDSS is usually a Local Area Network (LAN) or a Wide Area Network (WAN). An example of an NDSS using LAN can be a data center and an example of an NDSS using WAN can be a P2P file sharing system.

In data centers, the data is placed in storage nodes which are connected through a network. These storage nodes are usually organized in a rack, a metallic support designed to accommodate electronic equipment. The communication (bandwidth) cost between nodes which are in the same rack is much lower than between nodes which are in different racks. In fact, in [AGSS11], it is said that reading from a local disk is nearly as efficient as reading from the disk of another node in the same rack.

There are many drawbacks in the use of NDSS which are being studied nowadays. The problem of the data insertion is an interesting research topic: how to efficiently change the already stored data and propagate the changes through the NDSS [PJOD13]. In this dissertation, we focus on another problem: how to minimize the bandwidth in the NDSS using coding theory techniques.

In this dissertation, we consider files as a black box of information symbols. We have no interest in the techniques that may have been applied previously to the files like compression or encryption. This means that we consider the file as a sequence of information symbols to be stored.

Assume that a user wants to store a file and he wants this file to be available. The availability of a file is given by two conditions: the user is allowed to access the file and the file exists. The first condition is the object of study of computer security, while the second condition is related to the persistence of the data and it is an object of study of this research.

The first approach to increase the persistence of a file is the creation of replicas, which is usually known as backups. A backup is an exact replication of the file, which is stored in a different storage node. Then, if one storage node fails and the file is lost, there is another replica of the information. Note that the backup is in fact, redundancy of the stored information.

Definition 11 (i-fault tolerance system)**.** *An i-fault tolerance system, $1 \leq i \leq n$ where n is the number of storage nodes of the system, is defined as an NDSS that is able to tolerate any i number of storage node failures without irreversibly losing partially or completely the information stored in it.*

Note that the fault tolerance is defined over the storage nodes, and not over a specific stored file. This means that an i-fault tolerance system ensures that any information stored in any nodes of the system tolerates any i number of storage node failures.

It can be seen that to create an i-fault tolerance system with $i \geq 1$, it is necessary to create redundancy for the stored information. This means that an NDSS also has a transmission rate which is the rate between the stored information and this information plus the redundancy. Conceptually, if the transmission rate is decreased, it is possible to increase the fault tolerance of an NDSS.

A similar problem was already addressed by Shannon in 1948. Shannon-Hartley theorem [Sha49] shows the maximum rate at which information can be transmitted over a communication channel of a specified bandwidth in the presence of noise. Assume that a source sends a message to a receiver over a communication channel which is affected by noise. Depending on the amount of noise, which is the maximum transmission rate at which the entire message can be understood by the receiver? Noise affects information, changing symbols or erasing them, and the goal is to be able to recover the original information at the output of the channel.

As it can be seen, there is a direct relation between the communication of information over a noisy channel and the persistence of the stored data over an NDSS. The information to be sent is the file, the channel is the NDSS, the noise is the failure of storage nodes and finally, the receiver is the user who wants to recover the file by accessing to some storage nodes. This close relation between both problems explains why coding theory, the study of efficient and reliable data transmission methods, can also be used in NDSS.

Assume that a file is going to be stored using an NDSS, so it should be persistent. To achieve this persistence, it can be replicated and stored over different storage nodes in an NDSS. In this case, we are using a repetition erasure correcting code. However, there exist much better codes than the repetition one in terms of the transmission rate and erasure correcting capability [WK02], so the first idea is to use these better codes instead of the repetition one.

Some well known efficient codes are used for storage, like Hamming and

Reed Solomon codes [MS77] or EVENODD codes [TS02]. However, NDSS introduces a problem which is not addressed by classical coding theory and that achieves a high importance in order to establish the efficiency of an NDSS.

Classical coding theory is focused on recovering the whole set of information symbols which were sent, not a subset of them. In NDSS, it is usual that only one storage node fails, and usually, this storage node only contains a small subset of symbols from the same file (usually only one to increase the fault tolerance). If a new storage node replaces the lost one, the goal of this storage node, called newcomer, is to efficiently store the data that was lost, where efficiently means either reducing the amount of downloaded data (repair bandwidth) or the complexity of the operations needed.

2.3 Codes and NDSS

As we have seen in Section 2.2, classical coding theory do not address some of the problems that the application of codes to NDSS introduces. In this section, we show the key points in the design of efficient coding techniques for NDSS.

2.3.1 The repair problem

The repair problem is related to the amount of data needed to repair a single storage node failure. Assume that a file is stored using an erasure correcting code. This means that the file is split into a vector v of k coordinates, v is encoded into a codeword c of n coordinates, and each coordinate is stored in a different storage node. Repairing a single storage node is the same as repairing a single coordinate of c. The amount of data needed to repair this coordinate is called the repair bandwidth γ.

From the storage point of view, increasing the transmission rate means that less redundancy is stored, and increasing the correction capability means that more node failures are tolerated. The use of codes in NDSS can produce the same benefits than in the data transmission case, since the transmission rate and the fault tolerance are also two key parameters. Figure 2.1 shows a 1-fault tolerance $[2, 1, 2]$ repetition code and a 1-fault tolerance $[3, 2, 2]$ parity check code. These codes are applied to two information coordinates v_1 and v_2. One can assume that these coordinates represent a file to be stored. Using the parity check code, the information (v_1, v_2) encoded gives

Figure 2.1: Left: a $[2, 1, 2]$ repetition code applied to two coordinates. Right: a $[3, 2, 2]$ parity check code applied to two coordinates.

the codeword $(v_1, v_2, v_1 + v_2)$ and the transmission rate is $R = 2/3$. Using the repetition code, the information (v_1) and (v_2) encoded give (v_1, v_1) and (v_2, v_2), respectively, which represent (v_1, v_1, v_2, v_2), and the transmission rate is $R = 1/2$. The use of a more sophisticate coding technique in the parity check code has increased the transmission rate while maintaining the fault tolerance.

Assume that each coordinate of the codeword is stored in a different storage node, one of them fails and we want to replace it. On the one hand, in a repetition scheme, the storage node can be repaired by downloading one replica of the lost coordinate and copying it. This means that the downloaded and the stored data per newcomer are the same. On the other hand, the classical decoding using linear codes always needs and uses $n - d + 1$ correct coordinates no matter whether the codeword has one or more (up to $d - 1$) erasures. Then, the repair bandwidth needed per newcomer is grater than the stored data per newcomer.

Figure 2.1 illustrates the repair problem. To repair a single coordinate failure, using the parity check code it is necessary to download two symbols b, $a + b$ and compute $a = (a + b) - b$, while using the repetition code it is only necessary the copy of a. In general, repairing a single coordinate in a repetition code needs a repair bandwidth of $\gamma = \alpha$, where α is the amount of stored data in one storage node, in this case, the size of one coordinate. In the parity check code, repairing a single coordinate implies downloading any $n - d + 1 = k = 2$ coordinates, so $\gamma = 2\alpha = M$, where M is the size of the file.

The use of codes in NDSS dramatically decreases the amount of redundancy needed to achieve the same fault tolerance as using a repetition scheme [RL05]. However, the use of the parity check code needs a repair bandwidth of $\gamma = M$ to repair any subset of less than d coordinates, while the use of the repetition code needs $\gamma = \alpha$ for each lost coordinate. In general, we know that $\alpha \ll M$, so it can be seen that the repair problem is an important drawback. This drawback is the main reason why the application of coding

theory in NDSS is being postponed and its resolution is related to the next question: is it possible to repair one single erasure requiring less than $n-d+1$ coordinates?

2.3.2 Important properties inherited from codes

When codes are applied to NDSS, there are some properties and concepts from classical coding theory that can be seen from a different perspective.

MDS codes

As we have seen in Subsection 2.1.2, when a code is MDS it has the maximum minimum distance for a given redundancy. In other words, it is the best code in terms of the compromise between the transmission rate and the correction capability. It also means that the distance is $d = n - k + 1$, so it can correct $n-k$ erasures, which means that any k coordinates are enough to recover the information. When we say that a code has the minimum storage overhead, it means that the code is MDS, so its rate is $R = k/(d + k - 1)$.

Now, we are going to do an abuse of notation on some parameters of the codes. Let an $[n, k, d]$ NDSS be a $(d-1)$-fault tolerance system composed of n storage nodes and where a subset of k storage nodes contain enough information to recover the file. Assume that each storage node stores the same amount of data α then, the transmission rate of the NDSS is $R = (k\alpha)/(n\alpha) = k/n$. When a MDS code is applied to an NDSS and each coordinate of the codeword is stored in a different storage node, the MDS property means that the $[n, k, d]$ NDSS is a $(n-k)$-fault tolerance system and so any k storage nodes have enough information to recover the file. Note the difference between "a subset of k storage nodes" and "any subset of k storage nodes". Moreover, this "any k" property, which means that $d = n - k + 1$, is achieved by all MDS codes, but can be also achieved by non MDS codes.

A code applied to an $[n, k, d]$ NDSS is MDS if and only if it minimizes the storage overhead and $d = n - k + 1$. Note that each coordinate of the codeword is stored in one node if and only if the parameters of the NDSS coincide with the parameters of the linear codes. From the coding theory point of view, a MDS code means that $d = n - k + 1$. However, when array codes are used in NDSS, it is possible to design non MDS codes achieving $d = n - k + 1$, but they do not have the minimum storage overhead.

Example 10 (Non MDS code achieving $d = n - k + 1$)**.** *Assume that a file is split into a vector of* 2 *coordinates and encoded using a* $[8, 2, 5]$ *code, which*

is not MDS code since $5 \neq 8-2+1$. *This code has has an erasure correction capability of* 4 *and any* $8-4=4$ *coordinates are enough to recover the file.*

Now, the resulting codeword is stored in a $[4,2,3]$ *NDSS where each storage node stores* 2 *coordinates of the codeword. It can be seen that the NDSS is a* 2*-fault tolerance system, because any* 2 *storage nodes have enough information to recover the file. However, as said before, the code is not MDS.*

Locality

Let C be a linear code and c be a codeword of C. We say that a coordinate i of c has repair degree r_i if we can recover any symbol at coordinate i by accessing at least r_i other coordinates, and the set of these coordinates is called a repair set of i. In other words, the repair degree r_i of a coordinate i is the minimum cardinal of all the repair sets of i. The repair sets of i are also called the repair alternatives of i. The repair degree of C, r, is the maximum of r_i, $i = 1, \ldots, n$. When we say that a code has a high locality, it means that the number of repair alternatives is big and their cardinals are small. As a result r is also small. It is also worth to mention that the repair degree is a good metric for repair bandwidth [OD11b], [PLD+12].

In [GHSY12], it is shown that the minimum distance d of a code is upper bounded by $d \leq n-k-\lceil \frac{k}{r} \rceil +2$, which means that as r increases (approaching to k), d decreases. This bound is equivalent to the singleton bound when $r \geq k$. Note that in the MDS codes, $r = k$. When $r < k$, the minimum distance d decreases and the optimality is achieved when $d = n-k-\lceil \frac{k}{r} \rceil +2$.

2.4 Coding techniques for NDSS

In this section, we show some of the most known codes that have been designed specifically for NDSS. It is not the aim of this dissertation to deeply explain these constructions, which is already done in [OD12]. In these kind of codes, each coordinate is stored in one storage node of the NDSS, which means that the parameters of the codes and of the NDSS coincide. Moreover, the correction capability of the code also coincides with the fault tolerance of the NDSS.

2.4.1 Locally reconstructible codes

Locally reconstructible codes use a technique where codes are applied to other codes with the goal to increase their locality. They are inspired in

the product codes [Eli54], which combine two erasure codes to tolerate both random and burst erasures. The base of the product codes is a $k \times k$ matrix which contains the information symbols to be encoded. Firstly, the rows of the matrix are encoded using an $[n, k]$ linear code, resulting in a $k \times n$ matrix. Then, the columns of this matrix are encoded with another $[n, k]$ linear code, resulting in a $n \times n$ matrix. This final matrix has horizontal and vertical redundancy. Despite the minimum distance of these kind of codes is low, their locality is increased by producing two repair alternatives for each coordinate.

Hierarchical codes

Hierarchical codes [DB08] can be seen as a bottom-up approach on the application of codes on other codes. Hierarchical codes use two different types of codes. The first one takes subsets of information coordinates and encode them independently from the other subsets to create what is called local redundancy. The second one encodes the local redundancy to create what is called global redundancy.

Example 11. *Let the information vector of a file be* $v = (v_1, v_2, v_3, v_4)$. *Split* v *into two vectors* (v_1, v_2) *and* (v_3, v_4) *and encode them independently using a* $[3, 2]$ *parity check code into* $(v_1, v_2, v_1 + v_2)$ *and* $(v_3, v_4, v_3 + v_4)$. *The local redundancy coordinates can be seen as a vector* $(v_1 + v_2, v_3 + v_4)$ *and encoded using the same code into* $(v_1 + v_2, v_3 + v_4, v_1 + v_2 + v_3 + v_4)$, *where* $v_1+v_2+v_3+v_4$ *is a global redundancy coordinate. Finally, store the codeword* $(v_1, v_2, v_1+v_2, v_3, v_4, v_3+v_4, v_1+v_2+v_3+v_4)$ *in the NDSS with one coordinate in each storage node.*

Note that the information and local redundancy coordinates have repair degree 2, *however, the resulting NDSS is only a* 1*-fault tolerance system with a transmission rate of* $R = 4/7$. *This specific encoding example can be seen as a product code where the information is a* 2×2 *matrix and the vertical redundancy is computed only over the column containing the horizontal redundancy*

$$\left(\begin{array}{cc|c} v_1 & v_2 & v_1 + v_2 \\ v_3 & v_4 & v_3 + v_4 \\ - & - & v_1 + v_2 + v_3 + v_4 \end{array}\right).$$

This example can be generalized by using more sophisticated codes for the local and for the global redundancies, bigger information matrices, or the iteration of the procedure multiple times. The motivation behind hierarchical codes is the use of local redundancy to have a small repair degree for

some specific coordinates, while the global redundancy provides the correction capability.

Pyramid codes

Pyramid codes [Hua07] can be seen as a top-down approach on the application of codes on other codes. Firstly, a MDS code encodes the information vector and produces the global redundancy. Then, the information vector is split into smaller vectors which are padded with zeros and encoded again with the same code producing the local redundancy. Note that, the global redundancy coefficients are in fact linear combinations of the local redundancy coefficients. Finally, some local and global redundancy coefficients are stored.

Example 12. *An information vector* $v = (v_1, \ldots, v_8)$ *is encoded using a* $[11, 8]$ *systematic MDS code with generator matrix* G*. Then* $(v_1, \ldots, v_8)G = (v_1, \ldots, v_8, \rho_1, \rho_2, \rho_3)$*, where* ρ_1*,* ρ_2 *and* ρ_3 *are the global redundancy coefficients. Next, encode the vectors* $(v_1, \ldots, v_4, 0, 0, 0, 0)G = (v_1, \ldots, v_4, 0, 0, 0, 0, \rho_{1,1}, \rho_{2,1}, \rho_{3,1})$ *and* $(v_5, \ldots, v_6, 0, 0, 0, 0)G = (v_5, \ldots, v_6, 0, 0, 0, 0, \rho_{1,2}, \rho_{2,2}, \rho_{3,2})$*. Note that* $\rho_1 = \rho_{1,1} + \rho_{1,2}$*,* $\rho_2 = \rho_{2,1} + \rho_{2,2}$ *and* $\rho_3 = \rho_{3,1} + \rho_{3,2}$*. Finally, take some global redundancy coefficients and some local redundancy coefficients corresponding to the rejected global coefficients. For example* $\rho_{1,1}, \rho_{1,2}, \rho_2, \rho_3$*, and store* $(v_1, \ldots, v_8, \rho_{1,1}, \rho_{1,2}, \rho_2, \rho_3)$*.*

Note that the global coefficients are split into local coefficients, decreasing the minimum distance of the code but increasing its locality. In pyramid codes, larger codes are reused to build smaller codes, in contrast to hierarchical codes where smaller codes are assembled together to form a bigger code. Pyramid codes achieve similar locality compared with Hierarchical codes, but with a better fault tolerance. It is worth to mention that a variation of pyramid codes are the base codes used in Microsoft Azure [HSX+12].

2.4.2 Locally repairable codes

Locally repairable codes (LRC) are codes designed to increase the locality, while trying to keep a high fault tolerance. The difference with the codes proposed in Subsection 2.4.1, is that LRC are codes specifically designed for these two goals.

Let C be an $[n, k, d]$ linear code and let $C^\perp$ be its dual code, that is, $C^\perp = \{\omega \in \mathbb{F}_q^n \mid \omega \cdot c = 0,\ \forall c \in C\}$, where $\omega \cdot c = \sum_{j=1}^{n} \omega_j c_j$. Then, we

say that $\omega \in C^{\perp}$ is a parity check vector of C and $C^{\perp}$ is the set of parity check vectors of C. Given $c \in C$ and $\omega \in C^{\perp}$, the reparation of a single coordinate means that an specific c_i is missing and it can be recovered by solving $\sum_{j=1}^{n} \omega_j c_j = 0$, where c_i is the unknown. Note that the parity check vector ω has $w_H(\omega)$ nonzero coordinates, where $w_H(\omega)$ means the Hamming weight of ω. The reparation of a single specific coordinate c_i of a codeword c requires to have a parity check vector $\omega \in C^{\perp}$ such that $\omega_i \neq 0$. Then, retrieve $w_H(\omega) - 1$ symbols from c corresponding to the nonzero coordinates of ω except for i, and solve the above equation for c_i. Given an index $i = 1, \ldots, n$, we define $\Omega(i) = \{\omega \in C^{\perp} \mid \omega_i \neq 0\}$. This set represents all possible parity check vectors which repair the coordinate c_i, so it is the set of repair alternatives of c_i.

Definition 12 (Repair degree of an LRC)**.** *The repair degree of the ith coordinate of a LRC is defined as* $r_i = \min\{w_H(\omega) - 1 \mid \omega \in \Omega(i)\}$*, and the overall repair degree* r *is its maximum repair degree* $r = \max\{r_i\}_{i=1}^{n}$ *[PJHH13].*

Moreover, note that the MDS codes have degree $r = k$. Locally repairable codes (LRC) try to keep r at very low rates, this means that for LRC it is a goal that $r \ll k$. However, as it is explained in Subsection 2.3.2, the low repair degree decreases the upper bound on the minimum distance d of a code, which also decreases the correction capability. There are many constructions of LRC and it is a hot topic at the moment, probably, the most known ones are [OD11a], [OD11b] and [PD12].

In general, good LRC are those codes with a low repair degree and a high number of repair alternatives. To achieve a low repair degree, we need for each coordinate i, one parity check vector ω, $\omega \neq 0$ with $w_H(\omega)$ as low as possible. Moreover, to achieve a high number of repair alternatives, we need a high number of those parity check vectors.

There is a lot of research done on locally repairable codes, and an increasing number of articles being published each year. However, it is not the aim of this dissertation to deeply explain this kind of constructions. An interested reader can find more information in the cited articles and surveys of this section.

2.5 Network coding techniques for NDSS

The codes shown in the previous section decrease the repair degree, and decreasing the repair degree also means decreasing the repair bandwidth.

However, in those constructions, $d \ll n - k + 1$ which is a drawback in the case of simultaneous node fails. Regenerating codes are a family of codes designed to achieve $d = n - k + 1$, and decrease the repair bandwidth at the same time. In this section, regenerating codes are deeply explained.

In Subsection 2.5.1, some basic concepts on graph theory and information flow graphs are explained. In Subsection 2.5.2, network coding is introduced. In Subsections 2.5.3, and 2.5.4, homogeneous models and non-homogeneous modes are shown, respectively. In Subsection 2.5.5, regenerating codes are presented. Finally, in Subsection 2.5.6, we introduce a new solution which lies between regenerating codes and locally repairable codes and that we call flexible regenerating codes.

2.5.1 Graph theory

Graph theory is a well known mathematical topic which studies mathematical structures used to model pairwise relations between objects. Graph theory is an extensive topic [Ber01]. However, we focus only on those concepts which are necessary to understand regenerating codes.

Definition 13 (Graph)**.** *A graph is a collection of points and lines connecting a subset of points. The points of a graph are called vertices and the lines are called edges. A graph $G(W, E)$ is a pair of sets with $E \subseteq W \times W$. There is an edge from $w_1 \in W$ to $w_2 \in W$ if an only if $(w_1, w_2) \in E$ [Wei].*

A weighted graph is a graph where each edge has an associated weight.

A directed graph is a graph where the edges $(w_1, w_2) \in E$ have direction. This means that $(w_1, w_2) \in E$ goes from w_1 to w_2 but not from w_2 to w_1. An edge with direction is called an arc.

An acyclic graph is a directed graph where it is not possible to start at a vertex $w_1 \in W$, follow a sequence of connected vertices, and loop back to w_1.

A simple graph is an unweighted and undirected graph containing no graph loops (edges that connect a vertex to itself) or multiple edges (more than one edge that connect the same two nodes).

A bipartite graph is a graph whose vertices can be divided into two disjoint subsets W_1 and W_2, where each edge connects one vertex of W_1 with one vertex of W_2. A bipartite graph is usually denoted as $G(W_1 \cup W_2, E)$.

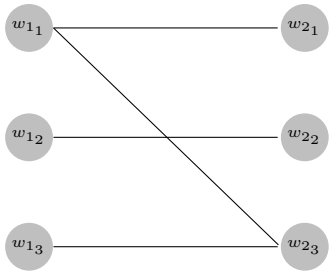

Figure 2.2: Bipartite graph associated with a square matrix.

A matching in a graph $G(W, E)$ is a set of pairwise non-adjacent edges. This means that no two edges share a common vertex. A perfect matching is a matching which matches all vertices of $G(W, E)$. In a bipartite graph $G(W_1 \cup W_2, E)$, we define a complete matching from W_1 to W_2 (resp. W_2 to W_1), if there is a matching which matches all vertices of W_1 (resp. W_2).

As we have said, a graph can be used to represent pairwise relations between objects. In [Har69], the relation between rectangular matrices and bipartite graphs is shown. Specifically, one can represent a rectangular matrix using a bipartite graph, where W_1 represent the rows and W_2 the columns of the matrix and the matrix entries are the weights of the edges.

Example 13 (Square matrices and bipartite graphs)**.** *The square matrix*

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

can be represented by the bipartite graph $G(W_1 \cup W_2, E)$ of Figure 2.2, where $w_{1_i} \in W_1$ is the i-th row and $w_{2_j} \in W_2$ is the j-th column of the matrix.

Let $p_s(\zeta_1, \zeta_2, \ldots, \zeta_m) \in \mathbb{F}_q[\zeta_1, \zeta_2, \ldots, \zeta_m]$ be the polynomial associated with the determinant of a $m' \times m'$ square matrix over $\mathbb{F}_q$ with $m \leq m'^2$. For example, the polynomial $p_s(\zeta_1, \zeta_2, \zeta_3, \zeta_4)$ associated with the 2×2 square matrix

$$\begin{pmatrix} \zeta_1 & \zeta_2 \\ \zeta_3 & \zeta_4 \end{pmatrix}$$

is $p_s(\zeta_1, \zeta_2, \zeta_3, \zeta_4) = \zeta_1\zeta_4 - \zeta_2\zeta_3$.

Let δ be the degree of $p_s(\zeta_1, \zeta_2, \ldots, \zeta_m)$. The polynomial $p_s(\zeta_1, \zeta_2, \ldots, \zeta_m)$ is defined over $\mathbb{F}_q$. If the polynomial $p_s(\zeta_1, \zeta_2, \ldots, \zeta_m)$ is not identically zero,

it means that, when it is expanded as a summation of terms, there exists at least one term with a nonzero coefficient.

Let $G(W_1 \cup W_2, E)$ be the bipartite graph associated with a square matrix. Let $E(w_{1_i})$ (resp. $E(w_{2_i})$) denote the neighbors of $w_{1_i} \in W_1$ (resp. $w_{2_i} \in W_2$) in the graph $G(W_1 \cup W_2, E)$.

Lemma 14 (Hall's theorem). *[Hal35] A bipartite graph $G(W_1 \cup W_2, E)$ contains a complete matching from W_1 to W_2 (resp. W_2 to W_1) if and only if it satisfies Hall's condition, that is, for any $T \subseteq W_1$ (resp. $T \subseteq W_2$), $|T| \leq |E(T)|$, where $T = \{t_1, \ldots, t_m\}$ and $E(T) = \bigcup_{i=1}^{m} E(t_i)$. Moreover, if $|W_1| = |W_2|$, the complete matching is achieved in both directions, so it corresponds to a perfect matching.*

Lemma 15. *[MR95] The polynomial associated with the determinant of a square matrix, $p_s(\zeta_1, \zeta_2, \ldots, \zeta_m)$, is not identically zero if and only if the bipartite graph $G(W_1 \cup W_2, E)$ associated with the square matrix has a perfect matching.*

If $p_s(\zeta_1, \zeta_2, \ldots, \zeta_m)$ is not identically zero because $G(W_1 \cup W_2, E)$ has a perfect matching, we can use the Schwartz-Zippel lemma to determine the probability that, for a random choice of the coefficients $\zeta_1, \zeta_2, \ldots, \zeta_m$ then, $p_s(\zeta_1, \zeta_2, \ldots, \zeta_m) = 0$.

Lemma 16 (Schwartz-Zippel lemma). *[Zip89] Let $p_s(\zeta_1, \ldots, \zeta_m)$ be a polynomial of degree δ over $\mathbb{F}_q$. Assume that $p_s(\zeta_1, \ldots, \zeta_m)$ is not identically zero. If $(\zeta_1, \ldots, \zeta_m)$ are chosen independently and uniformly over $\mathbb{F}_q$, then*

$$\Pr[\, p_s(\zeta_1, \ldots, \zeta_m) = 0 \,] \leq \frac{\delta}{|\mathbb{F}_q|}.$$

Information flow graphs

Definition 17 (Information flow graph). *An information flow graph is a directed acyclic and weighted graph which represents a flow of information from a set of sources to a set of sinks. The source vertices are the ones sending data and the sink vertices are the ones receiving data. Each arc is able to communicate an amount of data per time unit equivalent to its weight.*

The maximum flow of an information flow graph is given by the mincut between the sources and the sinks.

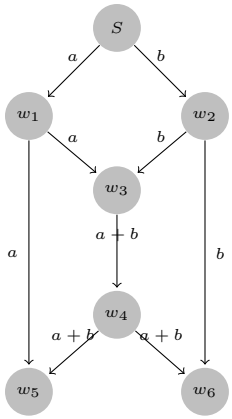

Figure 2.3: Butterfly network using network coding.

Definition 18 (Mincut of a graph)**.** *Let $G(W, E)$ be a weighted graph with nonnegative weights. A cut of G from $w_i \in W$ to $w_j \in W$ is a partition of W into two disjoint subsets where the first one contains w_i and the second one contains w_j. The weight of the cut is the sum of the weights of the edges crossing the cut. The mincut is the cut with minimum weight [Wei].*

Information flow graphs can be used to represent a NDSS. We can simulate node fails over time and apply network techniques to minimize the amount of stored data per node α and the repair bandwidth γ.

2.5.2 Network coding

Network coding is a coding technique applied to networks with the idea of improving the throughput, efficiency and scalability of the network. This technique was introduced in [ACLY00] for multicast purposes but nowadays, it has many other uses in the fields of security, compression, and coding theory among others.

The goal of network coding is to reduce the amount of data transmitted between a source and a set of sinks. To achieve this reduction, the intermediate nodes are allowed to produce and send linear combinations of the incoming symbols. The sinks receive these linear combinations, which can be treated as equations. When a sink has enough equations it can solve the system and recover the original symbols. To do that, the sinks accumulate equations in a matrix, when they have a full rank matrix, they recover the original symbols by applying the Gauss method.

If the network is modeled as an information flow graph, the vertices of

the graph are the nodes of the network and the arcs are the communication channels. Figure 2.3 shows the information flow graph known as the butterfly network. Assume that a source sends two information symbols a and b to two sinks, and that each arc has weight 1. This means that the arc $(w_1, w_2) \in E$, can communicate 1 symbol from w_1 to w_2 for each time unit. If the intermediate vertices are able to send linear combinations of the incoming symbols, and the sinks can solve systems of equations, the source can send a and b at the same time instead of sending a and then b which introduce a delay in the sinks.

The maximum number of symbols per time unit that the source can send through a network is given by the minimum mincut between the source and the sinks. In [Med03], it is shown that the use of random coefficients, over a sufficiently large field, in the linear combinations of the intermediate nodes is enough to produce full rank matrices to solve the equations in the sinks, so they can recover the original information symbols.

2.5.3 Homogeneous model

Information flow graphs can be used to simulate the life of an NDSS. The source of the graph is the file to be stored in a NDSS and a sink of the graph is the user who wants to recover the file. The intermediate nodes represent the life of the NDSS, they are the storage nodes in different time units. During the life of a NDSS some nodes fail and some others join the system in order to replace the failed ones. The nodes send information to each other in order to maintain the fault tolerance. The information flow graph is like a state machine, each step is produced by a fail and produces a new stable state. Finally, the sink wants to recover the stored information by connecting to a subset of these nodes. In this subsection, the first proposed model [DGWR10] which is based on the homogeneity of nodes and edges and is explained.

Let s_i, where $i = 1, \ldots, \infty$, be the i-th storage node. Let $G(W, E)$ be an information flow graph, with a set of vertices W and a set of arcs E. The set W contains three kinds of vertices:

- Source vertex S: it represents the file to be stored. There is only one source vertex in the graph.

- Data collector vertex DC: it is the sink vertex that represents the user who is allowed to access the data in order to reconstruct the file.

- Storage node vertices w_{in}^i and w_{out}^i: each storage node s_i, where $i = 1, \ldots, \infty$, is represented by one inner vertex w_{in}^i and one outer vertex w_{out}^i.

In general, there is an arc $(w_1, w_2) \in E$ of weight ς from vertex $w_1 \in W$ to vertex $w_2 \in W$ if w_1 can send ς data units to w_2.

At the beginning of the life of an NDSS, there is a file to be stored in n storage nodes s_i, $i = 1, \ldots, n$. This can be represented by a source vertex S with outdegree n connected to vertices w_{in}^i, $i = 1, \ldots, n$. As there is no restriction on the amount of information that the file sends to the storage nodes, the weight of the arcs is infinite. To represent that each one of the storage nodes s_i, $i = 1, \ldots, n$, stores α data units, each vertex w_{in}^i is connected to the vertex w_{out}^i with an arc of weight α.

When the first storage node fails, the first newcomer s_{n+1} connects to r, $0 < r < n$ existing storage nodes sending, each one of them, β data units. This can be represented by adding one arc from w_{out}^i, $i = 1, \ldots, n$, to w_{in}^{n+1} of weight β if s_i sends β data units to s_{n+1} in the regenerating process. The new vertex w_{in}^{n+1} is also connected to its associated vertex w_{out}^{n+1} with an arc of weight α. This process can be repeated for every failed node. Let the newcomers be denoted by s_j, where $j = n+1, \ldots, \infty$.

Finally, after some failures, a data collector wants to reconstruct the file. Therefore, a vertex DC is added to $G(W, E)$ along with one arc from vertex w_{out}^i to DC if the data collector connects to the storage node s_i. Note that if s_i has been replaced by s_j, the vertex DC can not connect to w_{out}^i, but it can connect to w_{out}^j. The vertex DC has indegree k and each arc has weight infinite, because the user is able to get as many information as he wants from each one of the storage nodes.

If the mincut from vertex S to DC, denoted by $\text{mincut}(S, DC)$, achieves that $\text{mincut}(S, DC) \geq M$, the data collector can reconstruct the file from the k storage nodes given by the k edges arriving to the DC, since there is enough information flow from the source to the data collector. If we want that any subset of k storage nodes are enough to recover the file, the data collector should be able to connect to any k nodes, so $\min(\text{mincut}(S, DC)) \geq M$, which is achieved when the data collector connects to k storage nodes that have already been replaced by a newcomer [DGWR10]. Note that using the minimum of all the mincuts we are assuming that any subset of k storage nodes have enough information to recover the file.

If we want to represent a code designed for an NDSS like the ones explained in Section 2.4 using an information flow graph, each coordinate is

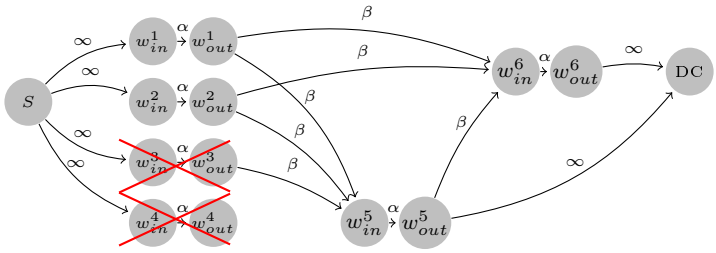

Figure 2.4: Information flow graph of a $[4, 2, 3]$ NDSS with $r = 3$.

stored in one node and $\alpha = \beta$. In this kind of codes, not any k storage nodes are enough to recover the file, so $\min(\text{mincut}(S, DC)) \geq M$ has no sense because the mincut will be different for each set of k storage nodes connected to the DC. Moreover, β does not appear in the mincut equation. Finally, one can conclude that no optimization is possible on the resulting graph using network coding techniques.

Figure 2.4 illustrates the information flow graph $G(W, E)$ associated to an $[4, 2, 3]$ NDSS with $r = 3$. Remember that an $[n, k, d]$ NDSS is a $(d-1)$-fault tolerance system composed of n storage nodes and where a subset of k storage nodes contain enough information to recover the file. In this section, we assume that $d = n-k+1$, so any subset of k storage nodes is enough to recover the file. Note that in Figure 2.4, $\text{mincut}(S, DC) = \min(3\beta, \alpha) + \min(2\beta, \alpha)$ which is the minimum mincut for this information flow graph. In general, it can be claimed [DGWR10] that

$$\text{mincut}(S, DC) \geq \sum_{i=0}^{k-1} \min((r-i)\beta, \alpha) \geq M. \tag{2.1}$$

It is possible to obtain a threshold function minimizing α and γ by using linear optimization techniques on the general equation (2.1) [DGWR10]

$$\alpha^*(r, \gamma) = \begin{cases} \frac{M}{k}, & \gamma \in [f(0), +\infty) \\ \frac{M - g(i)\gamma}{k-i}, & \gamma \in [f(i), f(i-1)) \\ & i = 1, \ldots, k-1, \end{cases} \tag{2.2}$$

where

$$f(i) = \frac{2Mr}{(2k-i-1)i + 2k(r-k+1)} \text{ and } g(i) = \frac{(2r-2k+i+1)i}{2r},$$

where $\gamma = r\beta$.

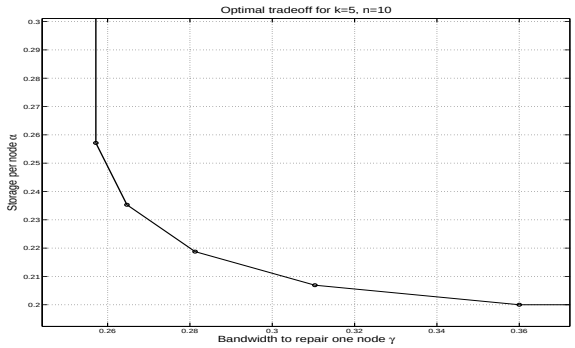

Figure 2.5: Optimal tradeoff curve between α and γ for a $[10, 5, 6]$ NDSS with $r = 9$.

To achieve this threshold function, multiple techniques have been used, like interference alignment [SRKR11], product-matrix construction [RSK11], or designs [RR10], among others. Behind these techniques, there are two main ideas: the use of array codes, which allows the NDSS to treat the data inside a node as a sequence of (small) coordinates over the base field; and the use of network coding to send linear combinations of these coordinates through the network.

This threshold function can be represented by a tradeoff curve like the one shown in Figure 2.5 for some specific parameters n, k and r. The optimal tradeoff curve represents the minimum tradeoff between α and γ. The two extremes of the curve are called the Minimum Storage Regenerating (MSR) and the Minimum Bandwidth Regenerating (MBR) points. In Figure 2.5, these points are placed approximately in $(0.36, 0.2)$ and $(0.258, 0.258)$. Between these two points, there are the called interior points.

Using the information flow graph $G(W, E)$, we can see that there are exactly k points in the tradeoff curve, or equivalently, k intervals in the threshold function $\alpha^*(r, \gamma)$, which represent k newcomers. In the mincut equation, the k terms in the summation are computed as the minimum between two parameters: the sum of the weights of the arcs that we have to cut to isolate the corresponding v_{in}^j from S, and the weight of the arc that we have to cut to isolate the corresponding v_{out}^j from S. Let the first parameter be called the *income* of the corresponding newcomer s_j. Note that the income of the newcomer s_j depends on the previous newcomers.

It can be seen that the newcomers can be ordered according to their

income from the highest to the lowest. In this model, this order is only determined by the order of replacement of the failed nodes. Moreover, the MSR point corresponds to the lowest income, which is given by the last newcomer added to the information flow graph; and the MBR point corresponds to the highest, which is given by the first newcomer. It is important to note also that, in this model, the order of replacement of the nodes does not affect to the final result, since the mincut is always the same independently of the specific set of k failed nodes.

2.5.4 Non-homogeneous models

In the previous subsections, we have introduced the concept of information flow graphs and we have associated it with an NDSS. However, we have based our analysis in the homogeneity of the parameters α and γ. This homogeneity means that every single node stores α data units and every single helper node (a node contributing to the replacement of a specific lost node) sends β data units in order to repair a failed node. A non-homogeneous model means either the amount of stored data per node or the amount of sent data per helper node depends on the specific storage node. In this work we focus on the second case, while the first one has also been studied [YSS11], [VYL12].

In [AKG10], *Akhlaghi et al.* presented another distributed storage model, where the storage nodes are partitioned into two sets W^1 and W^2. Let W^1 be the set of "cheap bandwidth" nodes, from where each data unit sent costs $\mathcal{C}_c$, and W^2 be the set of "expensive bandwidth" nodes, from where each data unit sent costs $\mathcal{C}_e$ such that $\mathcal{C}_e > \mathcal{C}_c$. This means that when a newcomer replaces a lost storage node, the cost of downloading data from a node in W^1 will be lower than the cost of downloading the same amount of data from a node in W^2.

Consider the same situation as in the model described in Subsection 2.5.3. Now, when a storage node fails, the newcomer node s_j, $j = n+1, \ldots, \infty$, connects to r_c existing storage nodes from W^1 sending each one of them β_c data units to s_j, and to r_e existing storage nodes from W^2 sending each one of them β_e data units to s_j. Let $r = r_c + r_e$ be the number of helper nodes. Assume that r, r_c, and r_e are fixed, that is, they do not depend on the newcomer s_j, $j = n+1, \ldots, \infty$. In terms of the information flow graph G, there is one arc from w_{out}^i to w_{in}^j of weight β_c or β_e, depending on whether s_i sends β_c or β_e data units, respectively, in the regenerating process. This new vertex w_{in}^j is also connected to its associated vertex w_{out}^j with an arc of weight α.

Let the repair cost be $\mathcal{C}_T = r_c\mathcal{C}_c\beta_c + r_e\mathcal{C}_e\beta_e$ and the repair bandwidth $\gamma = r_c\beta_c + r_e\beta_e$. To simplify the model, we can assume, without loss of generality, that $\beta_c = \tau\beta_e$ for some real number $\tau \geq 1$. This means that we can minimize the repair cost $\mathcal{C}_T$ by downloading more data units from the set of "cheap bandwidth" nodes W^1 than from the set of "expensive bandwidth" nodes W^2. Note that if τ is increased, the repair cost is decreased and vice-versa.

Again, it must be satisfied that $\min(\text{mincut}(S, DC)) \geq M$. Moreover, the newcomers can also be ordered according to their income from the highest to the lowest. However, in this model, the order is not only determined by the order of replacement of the failed nodes, as it happened in the model described in Subsection 2.5.3. It is important to note that, in this model, the order of replacement of the nodes affects to the final result and the mincut depends on the specific set of failed nodes.

The goal is also to find the $\min(\text{mincut}(S, DC))$, so the next problem arises: which is the set of k newcomers that minimize the mincut between S and DC? The minimum mincut is given by the set of k newcomers with the minimum sum of incomes. As it is shown in [AKG10], this set is composed of any $r_c + 1$ newcomers from W^1 plus the remaining newcomers from W^2. Moreover, the MSR point corresponds to the lowest income, which is given by the last newcomer; and the MBR point corresponds to the highest income, which is given by the first newcomer. Depending on k and r_c, it is necessary to distinguish between two cases.

Case $k \leq r_c + 1$

This case corresponds to the situation when the data collector connects to k newcomers from the set W^1. With this scenario shown in the information flow graph of Figure 2.6, the mincut analysis leads to

$$\sum_{i=0}^{k-1} \min(r_c\beta_c + r_e\beta_e - i\beta_c, \alpha) \geq M. \tag{2.3}$$

After applying $\beta_c = \tau\beta_e$ and an optimization process, the mincut equation (2.3) leads to the following threshold function:

$$\alpha^*(r_c, r_e, \beta_e) = \begin{cases} \frac{M}{k}, & \beta_e \in [f(0), +\infty) \\ \frac{2M - g(i)\beta_e}{2(k-i)}, & \beta_e \in [f(i), f(i-1)) \\ & i = 1, \ldots, k-1, \end{cases} \tag{2.4}$$

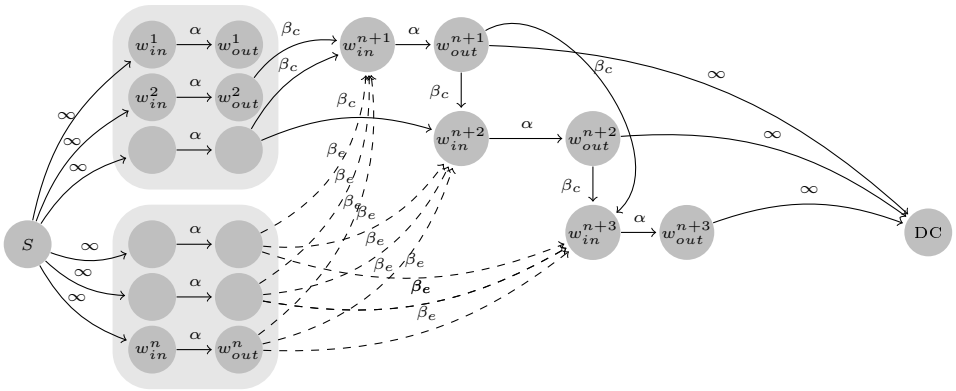

Figure 2.6: General information flow graphs corresponding to the case $k \leq r_c + 1$.

where

$$f(i) = \frac{2M}{2k(r_c\tau + r_e - \tau k) + \tau(i+1)(2k-i)} \quad \text{and}$$
$$g(i) = i(2r_c\tau + 2r_e - 2k\tau + (i+1)\tau).$$

Case $k > r_c + 1$

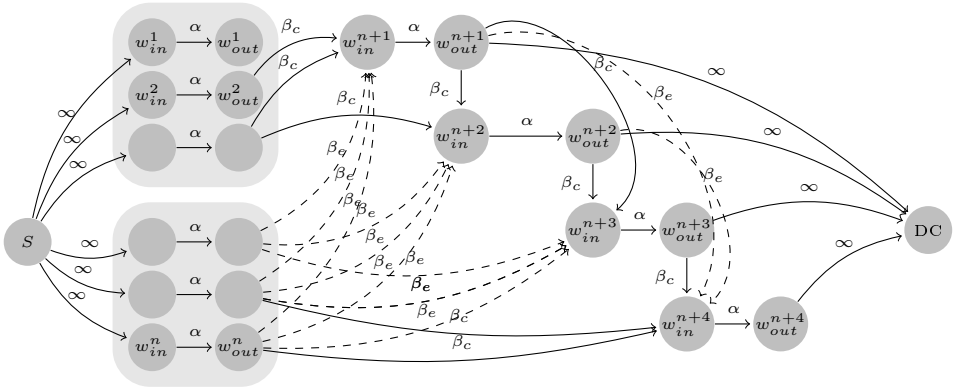

Figure 2.7: General information flow graphs corresponding to the case $k > r_c + 1$.

This case corresponds to the situation when the data collector connects to $r_c + 1$ replaced nodes from the set W^1 and to $k - r_c - 1$ replaced nodes from the set W^2. With this scenario shown in the information flow graph of Figure 2.7, the mincut analysis leads to

$$\sum_{i=0}^{r_c} \min(r_c\beta_c + r_e\beta_e - i\beta_c, \alpha) + \sum_{i=r_c+1}^{k-1} \min((r_c + r_e - i)\beta_e, \alpha) \geq M. \quad (2.5)$$

After applying $\beta_c = \tau\beta_e$ and an optimization process, the mincut equation (2.5) leads to the following threshold function:

$$\alpha^*(r_c, r_e, \beta_e) = \begin{cases} \frac{M}{k}, & \beta_e \in [f_1(0), +\infty) \\ \frac{2M - g(i)\beta_e}{2(k-i)}, & \beta_e \in [f_1(i), f_1(i-1)) \\ & i = 1, \ldots, k - r_c - 1 \\ \frac{2M - (g_1(i)(k-r_c-1)g_2(i))\beta_e}{2(r_c-i)}, & \beta_e \in [f_2(i), f_2(i-1)), \\ & i = k - r_c, \ldots, k-1, \end{cases} \quad (2.6)$$

where

$$f_1(i) = \frac{2M}{2k(r-k) + (i+1) + (2k-1)},$$

$$f_2(i) = \frac{2M}{(2kr - k^2 - r_c^2 - r_c + k + 2r_c\tau) + i\tau(2r_c - i - 1)},$$

$$g_1(i) = i(2r - 2k + i + 1), \text{ and}$$

$$g_2(i) = (i+1)(2r_e + i\tau).$$

2.5.5 Regenerating codes

Let C be a $[n, k, r]$ regenerating code, where the length n is the total number of nodes in the system; the dimension k is the value such that any k nodes contain the minimum amount of information necessary to reconstruct the file; and the cardinal of the set of helper nodes r is the number of nodes necessary to regenerate one failed node. Let the distance d of C be $d = n - k + 1$. Regenerating codes are array codes over $\mathbb{F}_{q^t}$ designed to approximate (and achieve if possible) the optimal parameters of the threshold function on the information flow graph for an $[n, k, n - k + 1]$ NDSS with repair degree r.

Let $v \in \mathbb{F}_{q^t}^k$ be the information vector representing the file to be stored in an NDSS. Let $c \in \mathbb{F}_{q^t}^n$ be the corresponding codeword after encoding v using C, which is an array code as the ones explained in Subsection 2.1.2. If each array coordinate of c is stored in a node, and the linear code C is MDS, the parameters n, k and d of the regenerating code coincide with the parameters

n, k, and d, of the linear code C. Moreover, in this case C is called a Minimum Storage Regenerating (MSR) code. When a code is MDS, it means that it minimizes the storage overhead. In other words, it is not possible to achieve a higher correction capability with equal transmission rate and it is not possible to achieve a higher transmission rate with equal correction capability. If each coordinate of a codeword is stored in one storage node, there are n storage nodes and because of the MDS property $d = n - k + 1$, k storage nodes are enough to recover the original information. We can then say that MSR codes are MDS codes, however not all MDS codes are MSR codes since not all of them are able to achieve optimal repair bandwidth γ.

Note that, it is also possible to store more data in the same storage node, for example by adding an extra set of elements over the base field $\mathbb{F}_q$, producing more redundancy which could be used to regenerate a failed node requiring less repair bandwidth γ. Using this technique, it is possible to minimize the repair problem at the cost of some extra storage overhead, but maintaining $d = n - k + 1$. When C achieves the minimum α such that $\alpha = \gamma$, C is called a Minimum Bandwidth Regenerating (MBR) code, and it can be seen that the parameters n, k and d of the regenerating code do not coincide with the ones defined for a linear code C.

Regenerating codes assume the data reconstruction condition: any k nodes must be enough to recover the file, which means that the minimum distance must be $d = n-k+1$, so it is necessary to have $\alpha k \geq M$. Moreover, if $d = n - k + 1$ and $\alpha k = M$, we have an MDS code. Another condition is that the regeneration of any node in the system must require less repair bandwidth than the total file size M, that is $\gamma < M$. If each helper node sends β data units, the repair bandwidth used is $\gamma = \beta r$. We know that any k nodes must contain enough information to recover the file so, unlike in the LRC case, the parameter r must achieve $r > k$. Note that if $r < k$ then $d < n - k + 1$ and if $r = k$, there is no possible optimization in the repair bandwidth. Moreover, as there are n storage nodes and one is the newcomer, then $k < r < n$. It is clear that as $k < r$, in order to decrease the repair bandwidth γ, each helper node must send $\beta < \alpha$ data units.

Note the difference between LRC and regenerating codes. In both cases, the repair degree is the number of helper nodes r necessary to regenerate a failed node. However, in LRC, each coordinate of a codeword c is stored in one node, and when we access to one helper node it means that we download the entire coordinate contained in it. Therefore, in order to decrease γ, we need $r \ll k$. In regenerating codes, to maintain the distance $d = n - k + 1$, we need $r \geq k$. Thus, since a codeword c can be seen as a vector of array

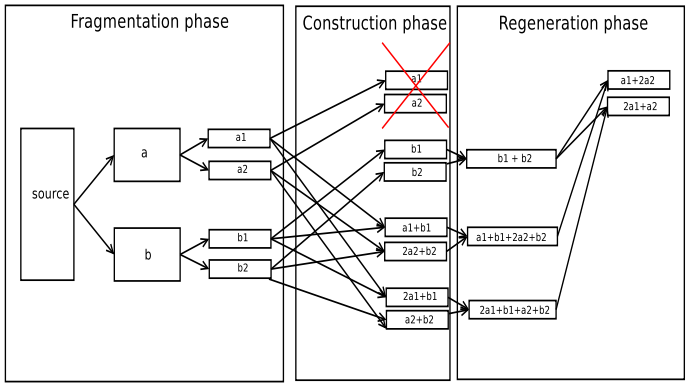

Figure 2.8: Fragmentation, construction and regeneration of a 2-fault tolerance $[4, 2, 3]$ regenerating code.

coordinates, in order to decrease γ we can increase r but downloading less than an entire array coordinate from each helper node. In other words, LRC are the result of creating new codes adapted to an NDSS, while regenerating codes are classical codes in which a network coding technique is used to reduce the repair bandwidth γ.

If any newcomer is able to exactly replicate the lost node, we say that the regenerating code has the *exact repair* property. Otherwise, if the newcomers store a linear combination that does not reduce the dimension of C and it does not coincide with the data included in the lost node, we say that the regenerating code has the *functional repair* property [DRW11]. Exact repair is much more desirable than functional repair, since despite the number of failed nodes that the NDSS has repaired over an interval of time, it is possible to use systematic encoding of the information and keep this systematic representation over the time. This means that there is always one accessible copy of the original file stored in the NDSS. It is worth to mention that in [SRKR12] it is proved that the interior points of the tradeoff curve are not achievable using exact repair.

We say that a regenerating code has the *uncoded repair* property if it is possible to replace a failed node without doing any linear operation in the newcomer neither in the helper nodes. There exist uncoded constructions for the MBR point like the ones shown in [RSK11] and [RR10]. However, for the MSR point, there only exist uncoded constructions using functional repair [HLS13].

Example 14 (Regenerating code)**.** *Assume that a file of size M is stored in a* $[4, 2, 3]$ *NDSS with $r = 3$. The information flow graph of this NDSS is shown*

in Figure 2.4 and its mincut equation is $mincut(S, DC) = \min(3\beta, \alpha) + \min(2\beta, \alpha)$. *The minimization of this equation is given by the threshold function (2.2) and provides two points in the tradeoff curve. The MSR point with* $\alpha = M/2$ *and* $\gamma = 3M/4$ *and the MBR point with* $\alpha = \gamma = 3M/5$.

Now, we design an example of a regenerating code that achieves the MSR point, a $[4, 2, 3]$ *MSR code. A code achieving these parameters is shown in Figure 2.8. Firstly, the file is divided into four coordinates over* $\mathbb{F}_3$ *and encoded into eight coordinates over the same field. Each storage node stores two of these coordinates producing an array codeword with four array coordinates over* $\mathbb{F}_{3^2}$. *It can be seen that any two storage nodes have enough information to recover the file. Finally, the functional repair of the first node is shown in the figure for* $\gamma = 3M/4$.

2.5.6 Flexible regenerating codes

In Subsection 2.5.5, regenerating codes are described. We have said that a regenerating code always achieve $d = n - k + 1$, despite if the code used is MDS or not. We have shown that this is the most important difference, from a practical point of view, between regenerating codes and LRC, because it indicates if "any" or "a" subset of k storage nodes contain enough information to recover the file.

However, we have intrinsically assumed that, in regenerating codes, the set of helper nodes consists on any r non failed nodes. If the set of helper nodes is a subset with r specific storage nodes, the resulting codes are called $[n, k, r]$ flexible regenerating codes. Note that these new codes are regenerating codes, because $r \geq k$ and any subset of k storage nodes contain enough information to recover the file. However, they share the idea of having a specific set of helper nodes given in Subsection 2.4.2 for LRC. Flexible regenerating codes can be seen as an hybrid solution between LRC and regenerating codes, where $d = n - k + 1$ but the set of helper nodes per newcomer consists of specific storage nodes.

The first flexible construction of regenerating codes was given in [SRKR12]. In that article, the authors constructed flexible regenerating codes achieving the same optimal parameters than MBR codes for $r = n-1$. Later, in [RR10], the authors used mathematical designs [MS77] to generalize the construction of these MBR codes for any r. Moreover, they discovered that by using this kind of constructions, the optimal MBR point of the tradeoff curve could be beaten. We call, to the resulting codes, flexible regenerating codes with minimum bandwidth. Note that the minimum γ is achieved when $\gamma = \alpha$.

In this section, we formalize those contributions by giving a general construction for flexible regenerating codes with minimum bandwidth along with their bounds and key parameters. In other words, we are rewriting the contributions on this topic and unifying the nomenclature, since they have been published separately. This section has three subsections which cover the three main aspects of any regenerating code, the code construction, the node regeneration and the data reconstruction.

Code construction

Let C be an $[n, k, r]$ MSR code over $\mathbb{F}_{q^t}$, where any subset of k storage nodes is enough to reconstruct the file. In this subsection, we explain how to construct a new $[\bar{n}, \bar{k}, \bar{r}]$ regenerating code $\bar{C}$ with minimum bandwidth over $\mathbb{F}_{q^{t\bar{r}}}$, from the base code C. Despite it is possible to construct $\bar{C}$ from any regenerating code, the construction makes sense if the regenerating code C is MSR (so MDS viewed as an array code), because then, $\bar{C}$ achieves the optimal parameters α and γ.

Lemma 19. *Given $n \geq 3$, there exists a simple, undirected, and $\bar{r}$-regular graph $H(W, E)$, where W is the set of vertices where $|W| = \bar{n}$ and E is the set of edges where $|E| = n$, satisfying the following conditions: $|W| = \bar{n}$, $|E| = n$, $1 < \bar{r} < \bar{n}$, and $\bar{r}\bar{n} = 2n$.*

Proof. Condition $\bar{r}\bar{n} = 2n$ is given by the Handshaking lemma for a simple, undirected, and $\bar{r}$-regular graph. By Erdos-Gallai degree sequence theorem, for $\bar{r} > 1$ and $\bar{r} < \bar{n}$, there exists at least a simple, undirected, and $\bar{r}$-regular graph such that $\bar{r}\bar{n} = 2n$. □

For example, for $\bar{r} = \bar{n} - 1$, we have the complete graph $H = K_{\bar{n}}$, and for $\bar{r} = 2$ we have the cycle graph $H = C_{\bar{n}}$. As $|E| = n$ in $H(W, E)$, it is possible to assume that each edge in E corresponds to a different coordinate c_j over $\mathbb{F}_{q^t}$ of a codeword $c = (c_1, \dots c_n) \in C$. Note that c_j could be also seen as an array coordinate composed of coordinates over the base field $\mathbb{F}_q$, but in this section, we consider the base field $\mathbb{F}_{q^t}$, so c_j is a coordinate of $\mathbb{F}_{q^t}$. Given a codeword $c \in C$, since $|W| = \bar{n}$ in $H(W, E)$, we can construct a codeword $\bar{c} = (\bar{c}_1, \dots, \bar{c}_{\bar{n}}) \in \bar{C}$, where each array coordinate $\bar{c}_i$ corresponds to a different vertex $w_i \in W$ and contains the coordinates of c given by the $\bar{r}$ edges incident to w_i. Moreover, since the graph is simple, any two vertices can not be connected by more than one edge, so each coordinate of c is contained in two array coordinates of $\bar{c}$. As C is defined over $\mathbb{F}_{q^t}$, $\bar{C}$ is defined over

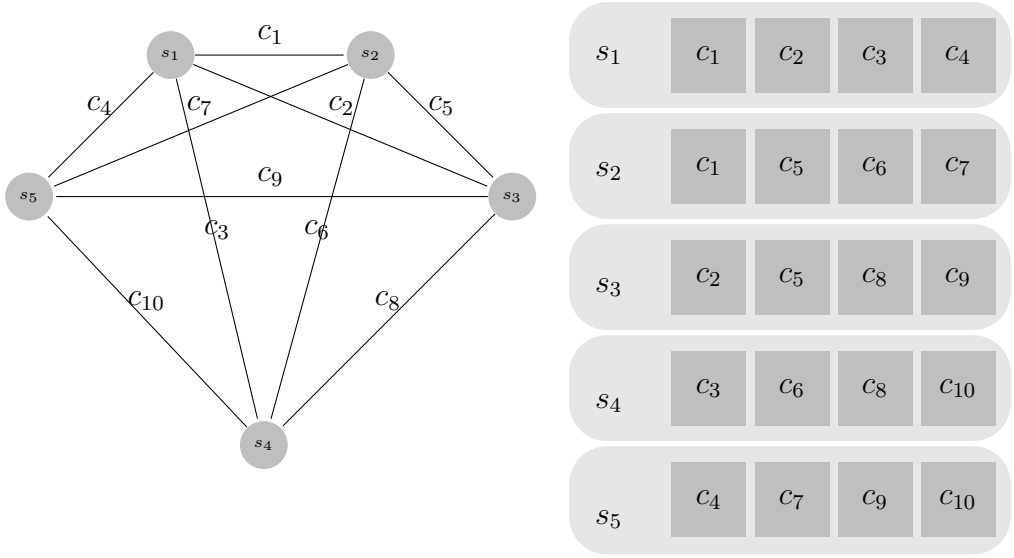

Figure 2.9: A $[10, 9, 2]$ MDS code used to create a $[5, 3, 4]$ flexible regenerating code with minimum bandwidth.

$\mathbb{F}_{q^{t\bar{r}}}$. Figure 2.9 shows an example of a $[5, 3, 4]$ flexible regenerating code with minimum bandwidth created from a $[10, 9, 2]$ MDS code, which can be used to illustrate this construction.

In the next subsections, we prove that $\bar{C}$ is a regenerating code with minimum bandwidth. Firstly, in the node regeneration subsection, we show that $\gamma = \alpha$. Then, in the data reconstruction subsection, we look for the minimum α such that any $\bar{k}$ array coordinates of $\bar{c} \in \bar{C}$ are enough to reconstruct the file. Note that $\bar{C}$ is a regenerating code, but not a code from the classical coding theory point of view, since $|\bar{C}| = |C|$ and $\bar{k}$ is not the dimension of the code but an integer such that $1 < \bar{k} < \bar{n}$.

Node regeneration

Assume that a storage node fails, which is the same as erasing one array coordinate $\bar{c}_i$, $i = 1, \ldots, \bar{n}$ of a codeword $\bar{c} = (\bar{c}_1, \ldots, \bar{c}_{\bar{n}}) \in \bar{C}$, or equivalently one vertex $w_i \in W$ of $H(W, E)$. The newcomer can replace the failed node by downloading and storing the $\bar{r}$ coordinates of c included in each one of the $\bar{r}$ neighbors of w_i, and given by the corresponding $\bar{r}$ edges incidents to w_i. According to this regeneration process, $\gamma = \alpha$.

Note that these regenerating codes with minimum bandwidth, have the exact repair and the uncoded repair properties. Also note that the node

regeneration is given by an specific subset of $\bar{r}$ helper nodes, so they are flexible regenerating codes.

Data reconstruction

Let $E(w_i)$, $i = 1, \ldots, \bar{n}$, be the set of edges incident to the vertex $w_i \in W$. Let $\bar{S}$ be the set of all subsets of $\{1, \ldots, \bar{n}\}$ of size $\bar{k} > 1$ and let $\bar{s} \in \bar{S}$. For a subset $\bar{s}$, $|\bigcup_{i \in \bar{s}} E(w_i)| = \sum_{i \in \bar{s}} |E(w_i)| - \theta_{\bar{s}}$, where $\theta_{\bar{s}}$ represents the intersection terms in the inclusion-exclusion formula. Since each edge is incident to two vertices, for $\bar{s}' \subseteq \bar{s}$ of size $|\bar{s}'| > 2$, $|\bigcap_{i \in \bar{s}'} E(w_i)| = 0$, so $\theta_{\bar{s}} = \sum_{i<j,\ i,j \in \bar{s}} |E(w_i) \cap E(w_j)|$. Let θ be the maximum of all $\theta_{\bar{s}}$, $\bar{s} \in \bar{S}$. Since $|E(w_i) \cap E(w_j)| \leq 1$ for any $i, j \in \{1, \ldots, \bar{n}\}$ and $i \neq j$, we have that $\theta \leq \binom{\bar{k}}{2}$.

Lemma 20. *Let C be an $[n, k, r]$ MSR code over $\mathbb{F}_{q^t}$ with $n \geq 3$. Choose $\bar{n}$, $\bar{k}$ and $\bar{r}$ such that $\bar{r}\bar{n} = 2n$, $1 < \bar{k} < \bar{n}$, $1 < \bar{r} < k$ and $k \leq \bar{k}\bar{r} - \theta$. Then, there is a $[\bar{n}, \bar{k}, \bar{r}]$ regenerating code $\bar{C}$ over $\mathbb{F}_{q^{t\bar{r}}}$. Moreover, the minimum α is achieved when $k = \bar{k}\bar{r} - \theta$.*

Proof. Given a file distributed using C, we know that there are n nodes and that any k of those n nodes are enough to reconstruct the file. By Lemma 19, we know that if $n \geq 3$, there exists a set of $\bar{n}$ vertices W and a set of n edges E, such that it is possible to construct $H(W, E)$ with $1 < \bar{r} < \bar{n}$ and $\bar{r}\bar{n} = 2n$. Then, from $H(W, E)$ it is possible to construct a code $\bar{C}$ as it is described in the code construction subsection.

The conditions $1 < \bar{k} < \bar{n}$ and $\bar{r} < k$ are necessary because if they are not achieved, the code $\bar{C}$ has no sense as a regenerating code. Note that a subset of cardinal $\bar{r} < k$ coordinates of c contained in $\bar{r}$ different nodes can regenerate a failed one, so $\gamma < M$. Finally, in order to reconstruct the file distributed using $\bar{C}$, any subset of $\bar{k}$ nodes must store at least k coordinates of $c \in C$, so $k \leq |\bigcup_{i \in \bar{s}} E(w_i)|$. Since $k \leq \bar{k}\bar{r} - \theta \leq \bar{k}\bar{r} - \theta_{\bar{s}} = |\bigcup_{i \in \bar{s}} E(w_i)|$ this condition is satisfied. Therefore, $\bar{C}$ is a regenerating code.

In the node regeneration subsection, it is shown that $\gamma = \alpha$. Moreover, $\bar{r}$ is the number of coordinates of c which compose an array coordinate of $\bar{c}$. Then, the minimum $\bar{r}$ will lead to the minimum α. As $\bar{r} \geq (k + \theta)/\bar{k}$, the minimum $\bar{r}$ is achieved when $k = \bar{k}\bar{r} - \theta$. □

As we are trying to minimize α, we assume the equality $k = \bar{k}\bar{r} - \theta$ given by Lemma 20, and we establish an upper bound for the parameter θ in Proposition 1.

Proposition 1. *In the graph $H(W,E)$ with $k = \bar{k}\bar{r} - \theta$, we have that*

1. $\theta \leq \binom{\bar{k}}{2}$ *if* $\bar{k} \leq \bar{r}+1$,
2. $\theta \leq \lfloor \frac{\bar{k}}{\bar{r}+1} \rfloor \binom{\bar{r}+1}{2} + \binom{\bar{k} \mod (\bar{r}+1)}{2}$ *if* $\bar{k} > \bar{r}+1$.

Proof. Each node $w_i \in W$ has $\bar{r}$ incident edges, so $\bar{k}$ nodes have $\bar{k}\bar{r}$ edges. Now, we distinguish two cases.

Case $\bar{k} \leq \bar{r}+1$: In $H(W,E)$, each vertex w_i, $i = 1, \ldots, \bar{n}$, shares one, and only one, edge with another vertex w_j. Each vertex w_i, $i \in \bar{s}$ and $|\bar{s}| = \bar{k}$, can share a maximum of one edge with each one of the other vertices $w_j, j \in \bar{s}$, $i \neq j$. Then, the maximum number of shared edges is $\binom{\bar{k}}{2}$. In other words, when $\bar{k} \leq \bar{r}+1$, it is possible to create a complete subgraph of $\bar{k}$ vertices in $H(W,E)$ with $\binom{\bar{k}}{2}$ edges.

Case $\bar{k} > \bar{r}+1$: Given $H(W,E)$ and $\bar{s}$, we are going to construct a subgraph which maximizes the number of shared edges. Each vertex w_i, $i \in \bar{s}$, can share a maximum of $\bar{r}$ edges with the remaining vertices $w_j, j \in \bar{s}$, $i \neq j$. Therefore, the maximum number of shared edges is when we consider a complete subgraph with $\bar{r}+1$ vertices and $\binom{\bar{r}+1}{2}$ edges. As $\bar{k} > \bar{r}+1$, there could be $\lfloor \frac{\bar{k}}{\bar{r}+1} \rfloor$ complete subgraphs, each one with $\binom{\bar{r}+1}{2}$ edges. The vertices out of these complete subgraphs can share a maximum of $\binom{\bar{k} \mod (\bar{r}+1)}{2}$ edges, which leads to the upper bound $\theta \leq \lfloor \frac{\bar{k}}{\bar{r}+1} \rfloor \binom{\bar{r}+1}{2} + \binom{\bar{k} \mod (\bar{r}+1)}{2}$ for $\bar{k} > \bar{r}+1$. □

Chapter 3

Quasi-cyclic Flexible Regenerating Codes

In this chapter, we present our first contribution, a new family of regenerating codes based on quasi-cyclic codes. This contribution has been partially published in international conferences [GPV11a], [GPV11b] and submitted as a journal paper in [GPV13a]. Quasi-cyclic flexible minimum storage regenerating (QCFMSR) codes are constructed and their existence is proved. Quasi-cyclic flexible regenerating codes with minimum bandwidth constructed from a base QCFMSR code are also provided. These codes not only achieve optimal MBR parameters in terms of stored data and repair bandwidth, but also for an specific choice of the parameters involved, they can be decreased under the optimal MBR point.

Quasi-cyclic flexible regenerating codes are very interesting because of their simplicity and low complexity. They allow exact repair-by-transfer in the minimum bandwidth case and an exact pseudo repair-by-transfer in the MSR case, where operations are needed only when a newcomer enters into the system.

3.1 Quasi-cyclic Flexible MSR codes

In this section, we describe the quasi-cyclic flexible minimum storage regenerating (QCFMSR) codes in detail. We show how to construct them and some of their properties; we see how to regenerate a failed node; we prove their existence by showing that the data reconstruction condition is achieved; and finally, we describe an example of a $[6, 3, 4]$ QCFMSR code.

3.1.1 Code Construction

Let C be an array code, like the ones explained in Subsection 2.1.2, of length $n = 2k$ and dimension k over $\mathbb{F}_{q^2}$ constructed from the nonzero coefficients $\zeta_1, \ldots, \zeta_k$ over $\mathbb{F}_q$, and for which the encoding is done over the base field $\mathbb{F}_q$ in the following way. An information vector $v \in \mathbb{F}_{q^2}^k$ is seen as a vector $v = (v_1, \ldots, v_n)$ over $\mathbb{F}_q$, and is encoded into $c \in \mathbb{F}_{q^2}^n$ seen as a codeword $c = (c_1, \ldots, c_{2n}) = (v_1, \ldots, v_n, \rho_1, \ldots, \rho_n)$ over $\mathbb{F}_q$, where the redundancy coordinates $\rho_1, \ldots, \rho_n$ are given by the following equation:

$$\rho_i = \sum_{j=i+1}^{k+i} \zeta_{j-i} v_j \quad i = 1, \ldots, n, \tag{3.1}$$

where $\zeta_l \in \mathbb{F}_q \setminus \{0\}$ for $l = 1, \ldots, k$ and $j = i+1, \ldots, k+i \mod n$. The rate of the code is $R = 1/2$ and the encoding over $\mathbb{F}_q$ is done by using a quasi-cyclic code [MS77] as we will see later. Quasi-cyclic codes are known by their simplicity for encoding-decoding operations.

A $[2k, k, r]$ QCFMSR code over $\mathbb{F}_{q^2}$ is a regenerating code constructed from the array code C. Take a file of size M and split it into k pieces over $\mathbb{F}_{q^2}$, or equivalently, into $n = 2k$ pieces over $\mathbb{F}_q$ organized as a vector $v = (v_1, \ldots, v_n)$ over $\mathbb{F}_q$. The $[2k, k, r]$ QCFMSR code over $\mathbb{F}_{q^2}$ is composed of a set of $n = 2k$ storage nodes, denoted by $\{s_1, s_2, \ldots, s_n\}$, where each storage node s_i, $i = 1, \ldots, n$, stores two coordinates over $\mathbb{F}_q$, (v_i, ρ_i) which can be seen as one array coordinate over $\mathbb{F}_{q^2}$. The size of each coordinate over $\mathbb{F}_q$ is $M/2k$ and the size of each array coordinate stored in s_i is $\alpha = M/k$.

Let S be the set of all subsets of $\{1, \ldots, n\}$ of size k. Let D be an $n \times n$ matrix over $\mathbb{F}_q$ and let $s = \{i_1, \ldots, i_k\} \in S$. Let $D^{\{i\}}$ denote the ith column vector of D and D^s denote the $n \times k$ submatrix of D given by the k columns determined by the set s.

Let $F = (I|Z)$ be a $n \times 2n$ matrix, where I is the $n \times n$ identity matrix, and Z is a $n \times n$ circulant matrix defined from the nonzero coefficients $\zeta_1, \ldots, \zeta_k$ as follows:

$$Z = \begin{pmatrix} 0 & 0 & \cdots & 0 & \zeta_k & \zeta_{k-1} & \cdots & \zeta_1 \\ \zeta_1 & 0 & \cdots & 0 & 0 & \zeta_k & \cdots & \zeta_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \zeta_k & \zeta_{k-1} & \cdots & \zeta_1 & 0 & 0 & \cdots & 0 \\ 0 & \zeta_k & \cdots & \zeta_2 & \zeta_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \zeta_k & \zeta_{k-1} & \zeta_{k-2} & \cdots & 0 \end{pmatrix}. \tag{3.2}$$

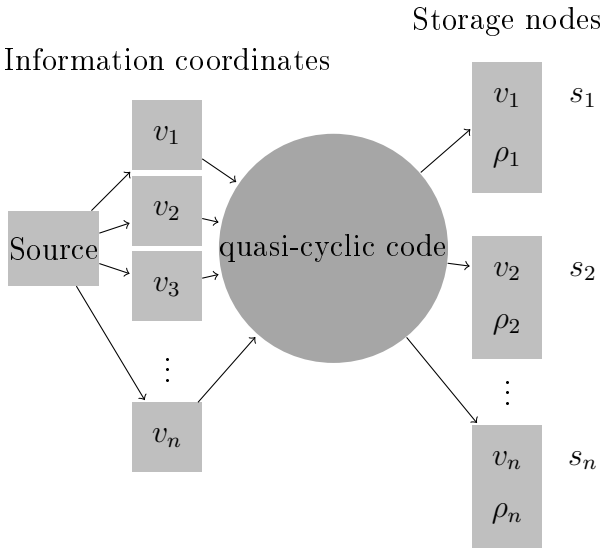

Figure 3.1: Construction process for a $[n, k, r]$ quasi-cyclic flexible MSR code.

The matrix F represents the array code C, so also the QCFMSR code constructed from C. Each row is the encoding of one coordinate over the base field $\mathbb{F}_q$, and each node is represented by two columns, one from I and another one from Z. Actually, the node s_i which stores (v_i, ρ_i), is also given by

$$(v_i, \rho_i) = (vI^{\{i\}}, vZ^{\{i\}}).$$

Note that the information coordinates are represented by the identity matrix I, while the redundancy coordinates are represented by the circulant matrix Z.

Circulant matrices have been deeply studied because of their symmetric properties [Dav79]. Moreover, F can be seen as a generator matrix of a double circulant code over $\mathbb{F}_q$ [MS77]. Double circulant codes are a special case of quasi-cyclic codes which are a family of quadratic residue codes. Quasi-cyclic codes have already been used for distributed storage [BBBM11] which points out the significance of these codes for NDSS.

Figure 3.1, shows the construction of a QCFMSR code. First, the file is split into n symbols over $\mathbb{F}_q$. Then, these symbols are encoded using F and producing $2n$ symbols over $\mathbb{F}_q$. Finally, each two symbols are stored together in one node, this creates the array code with coordinates over $\mathbb{F}_q$ that can be seen as array coordinates over $\mathbb{F}_{q^2}$.

3.1.2 Node Regeneration

In this subsection, we show how to regenerate a failed node s_i which stores (v_i, ρ_i), minimizing the required repair bandwidth. Actually, we can just follow the next algorithm:

1. Download the information coordinates v_j, $j = i+1, \ldots, i+k \mod n$, from the next k nodes. Note that due to the circulant scheme, the next node of s_n is s_1. From these information coordinates, compute the redundancy coordinate ρ_i of the newcomer.

2. Download the redundancy coordinate ρ_{i-1} from the previous node, following the same circulant scheme. Solving a simple equation, obtain the information coordinate v_i of the newcomer.

It can be seen that $r = r_i = k+1$ for any s_i, $i = 1, \ldots, n$, and when the repair problem is faced, it is clear that QCFMSR codes are optimal in terms of the tradeoff curve given by the threshold function (2.2) for $r = k+1$. Note that QCFMSR codes are in fact a family of regenerating codes because $r > k$. However, unlike regenerating codes, for these flexible regenerating codes the set of r helping nodes is not any but an specific set of remaining nodes with cardinality r. In other words, the set of nodes which is going to send data to an specific newcomer is fixed.

Note that QCFMSR codes have also the exact repair property which means that once encoded, the information and the redundancy can be represented for the whole life of the NDSS by $c = (v_1, \ldots, v_n, \rho_1, \ldots, \rho_n)$, where v_i and ρ_i are the information and redundancy coordinates, respectively. It is shown in [SRKR10] and [SRKR11] that when $r < 2k-3$, exact MSR codes do not exist. However, QCFMSR codes exist for $r = k+1$ which satisfies $r < 2k-3$ for $k > 4$. These facts illustrate the importance of the flexibility over the set of helper nodes in this construction. Moreover, despite QCFMSR codes do not achieve uncoded repair, they are very efficient regenerating one node, because they need only two simple operations on the newcomer and no operation on the helper nodes.

3.1.3 Data Reconstruction

In Subsection 3.1.1, we have seen that $M = \alpha k$. In this subsection, we prove that the array code over $\mathbb{F}_{q^2}$, used to construct a QCFMSR code, satisfies that $d = n-k+1$ for some $\zeta_1, \ldots, \zeta_k$ and, as a consequence, QCFMSR codes are MDS codes over $\mathbb{F}_{q^2}$ applied to NDSS, so they are MSR codes. In

[GPV11b], we performed a computational search to claim the existence of QCFMSR codes. In this subsection, we prove their existence theoretically.

Let $F^s = (I^s|Z^s)$ denote the $n \times n$ submatrix of F determined by $s = \{i_1, \ldots, i_k\} \in S$. Let $p_s(\zeta_1, \zeta_2, \ldots, \zeta_k) \in \mathbb{F}_q[\zeta_1, \ldots, \zeta_k]$ be the multivariate polynomial associated with the determinant of $F^s = (I^s|Z^s)$.

Assume that a DC wants to obtain the file. Then, it connects to any k nodes $\{s_{i_1}, \ldots, s_{i_k}\}$ and downloads $(v_{i_1}, \rho_{i_1}), \ldots, (v_{i_k}, \rho_{i_k})$, so the DC is downloading the encoding given by F^s. In order to obtain the file given by $v = (v_1, v_2, \ldots, v_n)$, we need F^s to be full rank. Moreover, in order to satisfy the data reconstruction condition, we need F^s to be full rank for all $s \in S$. Therefore, we want to use the Lemma 16 to prove that for a random choice of the nonzero coefficients $\zeta_1, \ldots, \zeta_k$, the polynomial $p_s(\zeta_1, \zeta_2, \ldots, \zeta_k)$ associated with the determinant of F^s is nonzero with high probability.

It has been explained in Section 2.5.1 that there exists a relation between determinants of matrices and bipartite graphs and that to use the Lemma 16, we need that $p_s(\zeta_1, \zeta_2, \ldots, \zeta_k)$ is not identically zero. Let $G(W_r \cup W_c, E)$ be the bipartite graph associated with a matrix F^s, where each row of the matrix is represented by a vertex w_{r_i} in W_r, and each column of F^s is represented by a vertex w_{c_i} in W_c, where $i = 1, \ldots, n$. Two vertices $w_{r_i} \in W_r$, $w_{c_i} \in W_c$ are adjacent if the entry in the row i and column j of F^s is nonzero. Moreover, the weight of this edge is the nonzero value of this i, jth entry. Let $E(w_{c_i})$ (resp. $E(w_{r_i})$) denote the neighbors of w_{c_i} (resp. w_{r_i}) in the graph G. Let $T = \{t_1, \ldots, t_m\} \subseteq W_c$ be a subset of vertices of W_c or $T \subseteq W_r$ be a subset of vertices of W_r indistinctly. Let $E(T)$ denote the set $\bigcup_{i=1}^{m} E(t_i)$.

By Lemma 15, we know that the polynomial associated with the determinant of F^s, $p_s(\zeta_1, \zeta_2, \ldots, \zeta_k)$, is not identically zero if and only if the bipartite graph $G(W_r \cup W_c, E)$ associated with F^s has a perfect matching. By Lemma 14, a bipartite graph $G(W_r \cup W_c, E)$ contains a complete matching from W_r to W_c (resp. W_c to W_r) if and only if it satisfies Hall's condition, that is, for any $T \subseteq W_c$ (resp. $T \subseteq W_r$), $|T| \leq |E(T)|$. Moreover, if $|W_r| = |W_c|$, the complete matching is achieved in both directions, so it corresponds to a perfect matching.

Lemma 21. *Let $T \subseteq W_c$ such that $T \neq \emptyset$, then $|E(T)| \geq |T|$.*

Proof. Note that W_c has k vertices of degree 1 and k vertices of degree k. We can decompose $T = T_1 \cup T_2$, where T_1 contains the vertices of degree 1 and T_2 contains the vertices of degree k. It is clear that $|E(T_1)| = |T_1|$ by construction, and it is easy to see that $|E(T_2)| \geq k + |T_2| - 1 \geq |T_2|$ by

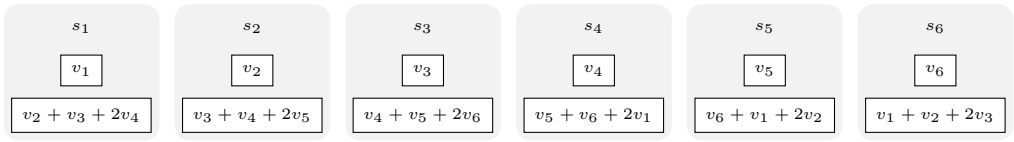

Figure 3.2: A $[6, 3, 4]$ QCFMSR code with coordinates over $\mathbb{F}_q$ and array coordinates over $\mathbb{F}_q^2$.

the circular construction of matrix Z and because $k > 1$. Therefore, we can assume that $T_1 \neq \emptyset$ and $T_2 \neq \emptyset$.

If $|T_1| \leq k-1$, since $|E(T_1) \cap E(T_2)| \leq |E(T_1)| = |T_1|$, we have that $|E(T)| = |E(T_1)| + |E(T_2)| - |E(T_1) \cap E(T_2)| \geq |T_1| + k + |T_2| - 1 - |T_1| \geq |T_1| + |T_2| = |T|$. On the other hand, if $|T_1| = k$, then $|E(T_1) \cap E(T_2)| \leq |T_1| - 1$ since for each different vertex $t_i \in T_2$, there exists a different vertex $t_j \in T_1$ such that $E(t_i) \cap E(t_j) = \emptyset$. Thus, we also have that $|E(T)| = |E(T_1)| + |E(T_2)| - |E(T_1) \cap E(T_2)| \geq |T_1| + k + |T_2| - 1 - |T_1| + 1 \geq |T_1| + |T_2| = |T|$. □

Proposition 2. *The polynomial associated with the determinant of F^s, $p_s(\zeta_1, \zeta_2, \ldots, \zeta_k)$, is not identically zero.*

Proof. Since $|W_r| = |W_c|$, by using Lemma 14 and Lemma 21, we have that the bipartite graph $G(W_r \cup W_c, E)$ associated with F^s has a perfect matching. Finally, by Lemma 15, we know that $p_s(\zeta_1, \zeta_2, \ldots, \zeta_k)$ is not identically zero. □

For the second statement, we have to prove that for a random choice of the nonzero coefficients $\zeta_1, \ldots, \zeta_n$, the multiplication of all the multivariate polynomials associated with the determinant of all matrices $F^s, s \in S$, is nonzero with high probability.

Let $p(\zeta_1, \ldots, \zeta_k) \in \mathbb{F}_q[\zeta_1, \ldots, \zeta_k]$ be the multivariate polynomial $p(\zeta_1, \ldots, \zeta_k) = \prod_{s \in S} p_s(\zeta_1, \ldots, \zeta_k)$. Note that if $p(\zeta_1, \ldots, \zeta_k) \neq 0$, then $p_s(\zeta_1, \ldots, \zeta_k) \neq 0$ for all $s \in S$.

Lemma 22. *The degree of $p(\zeta_1, \ldots, \zeta_k)$ is less than or equal to $k\binom{n}{k}$. Formally,* $\deg(p(\zeta_1, \ldots, \zeta_k)) \leq k\binom{n}{k}$.

Proof. Each ζ_i, $i = 1, \ldots, k$, can appear a maximum of k times in F^s. By Lagrange minor's theorem, $p_s(\zeta_1, \ldots, \zeta_k)$ has a maximum degree of k. By the definition of $p(\zeta_1, \ldots, \zeta_k)$, $\deg(p(\zeta_1, \ldots, \zeta_k)) \leq k\binom{n}{k}$. □

Theorem 23. *The $\binom{n}{k}$ submatrices F^s, $s \in S$, are full rank with high probability for a sufficiently large finite field $\mathbb{F}_q$.*

Proof. By Proposition 2 and using Lemma 16, we know that

$$\Pr(p(\zeta_1,\ldots,\zeta_k)=0) \leq \frac{\deg(p(\zeta_1,\ldots,\zeta_k))}{q}.$$

Therefore, $\Pr(p(\zeta_1,\ldots,\zeta_k)\neq 0) \geq 1-\frac{\deg(p(\zeta_1,\ldots,\zeta_k))}{q}$. And by Lemma 22, we know that $\deg(p(\zeta_1,\ldots,\zeta_k)) \leq k\binom{n}{k}$, so for a sufficiently large field size q, submatrices $F^s,\ s\in S$, are full rank with high probability. □

Summarizing, there is a set of full rank matrices F^s, $s\in S$, for a random choice of the nonzero coefficients $\zeta_1,\ldots,\zeta_n$ and a sufficiently large finite field. This means that there exists such F that represents a QCFMSR code with the property that any k storage nodes have enough information to reconstruct the file. In other words, a random choice of the coefficients over a sufficiently large finite field gives the encoding for a quasi-cyclic MDS array code of length n over $\mathbb{F}_{q^2}$ where each array coordinate of a codeword is (v_i,ρ_i). As this code is implemented in a NDSS following the construction given in Subsection 3.1.1, it gives a QCFMSR code.

It is worth to mention that using QCFMSR codes, an uncoded piece of the file is always kept in the system. Moreover, if more than one storage node fails, up to $n-k$, the decoding for the quasi-cyclic codes has linear complexity in contrast with the one for Reed-Solomon codes which has quadratic complexity [MS77].

3.1.4 Example

In this subsection, we describe the construction of a $[6,3,4]$ QCFMSR code over $\mathbb{F}_{5^2}$.

First, the file is fragmented into 6 information coordinates $v=(v_1,\ldots,v_6)$. Then, each v_i for $i=1,\ldots,6$ is stored in a node $s_i=(v_i,\rho_i)$, along with its corresponding redundancy symbol ρ_i which is computed using a quasi-cyclic matrix F of the following form:

$$F=\left(\begin{array}{cccccc|cccccc} 1&0&0&0&0&0&0&0&0&\zeta_3&\zeta_2&\zeta_1\\ 0&1&0&0&0&0&\zeta_1&0&0&0&\zeta_3&\zeta_2\\ 0&0&1&0&0&0&\zeta_2&\zeta_1&0&0&0&\zeta_3\\ 0&0&0&1&0&0&\zeta_3&\zeta_2&\zeta_1&0&0&0\\ 0&0&0&0&1&0&0&\zeta_3&\zeta_2&\zeta_1&0&0\\ 0&0&0&0&0&1&0&0&\zeta_3&\zeta_2&\zeta_1&0 \end{array}\right). \tag{3.3}$$

By construction, the node regenerating condition is always achieved. In order to satisfy the data reconstruction condition, we need to find nonzero coefficients $\zeta_1, \zeta_2, \zeta_3$ such that $p(\zeta_1, \zeta_2, \zeta_3) \neq 0$ over $\mathbb{F}_5$. Since $p(\zeta_1, \zeta_2, \zeta_3) = \zeta_1^{24}\zeta_2^{12}\zeta_3^5(-\zeta_2^2+\zeta_1\zeta_3)^5(\zeta_3^3+\zeta_1^3)(-\zeta_1\zeta_3^2+\zeta_2^2\zeta_3)(-\zeta_3^3-\zeta_1^3)$, a possible solution over $\mathbb{F}_5$ is $(\zeta_1, \zeta_2, \zeta_3) = (1, 1, 2)$. Figure 3.2 shows the distribution of information and redundancy coordinates in the nodes. Each array coordinate over $\mathbb{F}_{5^2}$ is represented by one storage node. It can be seen that $d = 4 - 2 + 1$ and that $\alpha k = M$, so the quasi-cyclic flexible code is a MSR code.

Using the same argument, it is also possible to construct a $[6, 3, 4]$ QCFMSR code over $\mathbb{F}_{8^2}$ with nonzero coefficients $(\zeta_1, \zeta_2, \zeta_3) = (1, 1, z)$ over $\mathbb{F}_8$, where z is a primitive element of this field. Note that there is not any $[6, 3, 4]$ QCFMSR code over $\mathbb{F}_{2^2}$, $\mathbb{F}_{3^2}$, $\mathbb{F}_{4^2}$ and $\mathbb{F}_{7^2}$.

3.2 Quasi-cyclic flexible regenerating codes with minimum bandwidth

It is possible to use QCFMSR codes as base regenerating codes to create regenerating codes with minimum bandwidth using the technique described in Subsection 2.5.6. In this subsection, we analyze the resulting parameters of these called quasi-cyclic flexible regenerating codes with minimum bandwidth.

Corollary 24. *For $\bar{k} \leq \bar{r} + 1$, there exists a $[\bar{n}, \bar{k}, \bar{r}]$ quasi-cyclic flexible regenerating code with minimum bandwidth constructed from a $[2k, k, k+1]$ QCFMSR code if the set of parameters $\{k, \bar{n}, \bar{k}, \bar{r}\}$ achieve:*

$$k = \frac{\bar{k}(2\bar{r} - \bar{k} + 1)}{2},$$

$$\bar{n} = \frac{2\bar{k}(2\bar{r} - \bar{k} + 1)}{\bar{r}},$$

$$1 < \bar{r} < k.$$

Proof. Straightforward from Lemmas 19, 20 and Proposition 1. □

Figure 3.3 shows an example of a quasi-cyclic flexible regenerating code with minimum bandwidth for $\bar{k} \leq \bar{r} + 1$ created from a $[6, 3, 4]$ QCFMSR code. Each node $\omega_i \in W$ can be repaired downloading half node ω_{i-1} and half node ω_{i+1}. Moreover, any $\bar{k} = 2$ nodes in $\bar{C}$ contain at least $k = 3$ different coordinates of $c \in C$ which allow us to reconstruct the file. Note

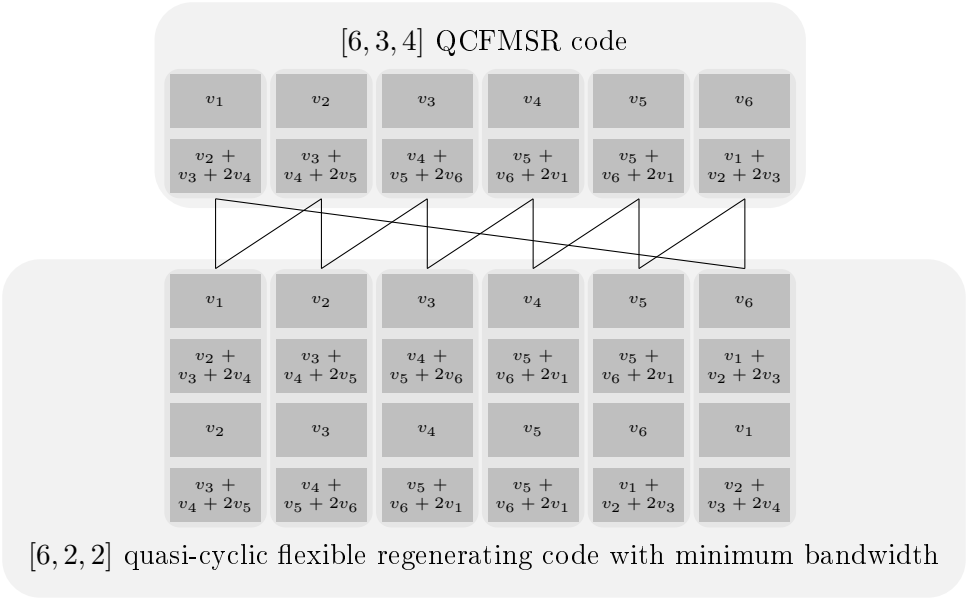

Figure 3.3: Construction of a $[6, 2, 2]$ quasi-cyclic flexible regenerating code with minimum bandwidth from a $[6, 3, 4]$ QCFMSR code.

that $\alpha = \frac{2M}{3}$ is equal to the value given in [DGWR10] for a $[6, 3, 4]$ MBR code.

Corollary 25. *For $\bar{k} > \bar{r} + 1$, there exists a $[\bar{n}, \bar{k}, \bar{r}]$ quasi-cyclic flexible regenerating code with minimum bandwidth constructed from a $[2k, k, k + 1]$ QCFMSR code if the set of parameters $\{k, \bar{n}, \bar{k}, \bar{r}\}$ achieves:*

$$k = \bar{k}\bar{r} - \left\lfloor \frac{\bar{k}}{\bar{r}+1} \right\rfloor \binom{\bar{r}+1}{2} + \binom{\bar{k} \mod (\bar{r}+1)}{2},$$

$$\bar{n} = \frac{2n}{\bar{r}},$$

$$1 < \bar{r} < k.$$

Proof. Straightforward from Lemmas 19, 20 and Proposition 1. □

Figure 3.4 shows an example of a quasi-cyclic flexible regenerating code with minimum bandwidth for $\bar{k} > \bar{r} + 1$ created from a $[10, 5, 6]$ QCFMSR code. Each node $\omega_i \in W$ can be repaired downloading half node ω_{i-1} and half node ω_{i+1}. Moreover, any $\bar{k} = 4$ nodes in $\bar{C}$ contain at least $k = 5$ different coordinates of $c \in C$ which allows us to reconstruct the file. Note that $\alpha = \frac{2}{5}M$ which is less than $\frac{4}{7}M$, the value given in [DGWR10] for a

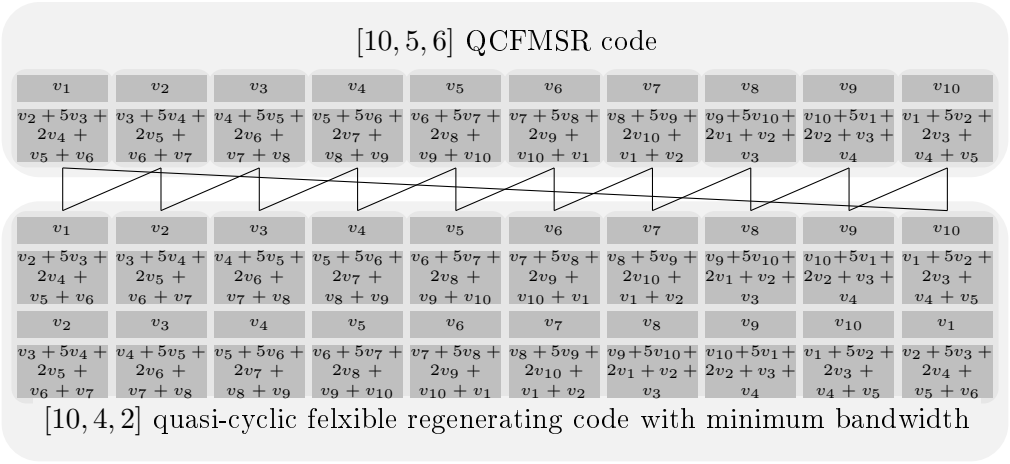

Figure 3.4: Construction of a $[10, 4, 2]$ quasi-cyclic flexible regenerating code with minimum bandwidth from a $[10, 5, 6]$ QCFMSR code.

$[10, 5, 6]$ MBR code. It is worth to mention that the reason of the decreasing on the lower bound given in [DGWR10] is the flexibility on the parameter $\bar{r}$.

Figure 3.5 shows the parameters of some quasi-cyclic flexible regenerating codes with minimum bandwidth. The first column shows the parameters $[\bar{n}, \bar{k}, \bar{r}]$ of the quasi-cyclic flexible regenerating codes with minimum bandwidth. The second column shows the parameters $[n, k, r]$ of the corresponding QCFMSR codes. The third and forth columns compare the minimum α such that $\alpha = \gamma$ for MBR codes as stated in [DGWR10] with the one achieved by the quasi-cyclic flexible regenerating codes with minimum bandwidth. First part of the table shows cases when $\bar{k} \leq \bar{r} + 1$, and the second part cases when $\bar{k} > \bar{r} + 1$

$[\bar{n}, \bar{k}, \bar{r}]$	$[n, k, r]$	$\alpha = \gamma$ [DGWR10]	$\alpha = \gamma$
$[6, 2, 2]$	$[6, 3, 4]$	$2M/3$	$2M/3$
$[8, 3, 3]$	$[12, 6, 7]$	$M/2$	$M/2$
$[7, 2, 4]$	$[14, 7, 8]$	$4M/7$	$4M/7$
$[10, 4, 4]$	$[20, 10, 11]$	$2M/5$	$2M/5$
$[10, 4, 2]$	$[10, 5, 6]$	$4M/7$	$2M/5$
$[10, 5, 2]$	$[10, 6, 7]$	$5M/9$	$M/3$
$[12, 5, 3]$	$[18, 9, 10]$	$5M/12$	$M/3$
$[16, 7, 3]$	$[24, 12, 13]$	$7M/18$	$M/4$

Figure 3.5: Parameters $[\bar{n}, \bar{k}, \bar{r}]$ for some quasi-cyclic flexible regenerating codes with minimum bandwidth constructed from $[n, k, r]$ QCFMSR codes, and comparison between the $\alpha = \gamma$ values given in [DGWR10] and the ones achieved with the proposed construction.

Chapter 4

Two-rack model

In Subsection 2.5.5, we described the model proposed in [AKG10] based on a non-homogeneous repair bandwidth which we call static cost model. The static cost model is based on different repair bandwidth costs where there is one set of "cheap" and one set of "expensive" helper nodes.

In realistic data centers, the data is placed in storage nodes which are connected through a network. These storage nodes are usually organized in a rack, a metallic support designed to accommodate electronic equipment. Figure 4.1 shows the rear of a real rack used in a data center. The communication (bandwidth) cost between nodes which are in the same rack is much lower than between nodes which are in different racks. In fact, in [AGSS11] it is said that reading from a local disk is nearly as efficient as reading from the disk of another node in the same rack.

In this chapter we present our second contribution, a model designed to represent the described situation. This contribution has been partially

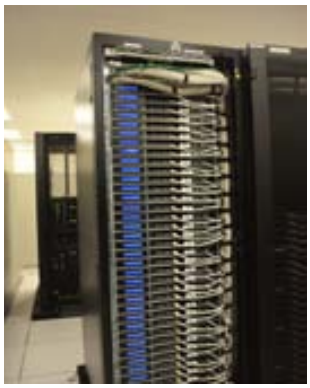

Figure 4.1: Rear of a real rack used in a data center.

published in the international conference [GPV13b] and summarized as a journal paper [GPV13c].

4.1 The model

In this model, the cost of sending data to a newcomer in a different rack is higher than the cost of sending data to a newcomer in the same rack. Note the difference of this rack model compared with the static cost model described in Subsection 2.5.5. In that model, there is a static classification of the storage nodes between the ones having "cheap bandwidth" and the ones having "expensive bandwidth". In our new model, this classification depends on each newcomer. When a storage node fails and a newcomer enters into the system, nodes from the same rack are in the "cheap bandwidth" set, while nodes in other racks are in the "expensive bandwidth" set. In this section, we analyze the case when there are only two racks. Let W_1 and W_2 be the sets of n_1 and n_2 storage nodes from the first and second rack, respectively.

Consider the same situation as in Subsection 2.5.5, but now the sets of "cheap bandwidth" and "expensive bandwidth" nodes depend on the specific replaced node. Again, we can assume, without loss of generality, that $\beta_c = \tau\beta_e$ for some real number $\tau \geq 1$. Let the newcomers be the storage nodes s_j, $j = n+1, \ldots, \infty$. Let $r = r_c^1 + r_e^1 = r_c^2 + r_e^2$ be the number of helper nodes for any newcomer, where r_c^1, r_e^1 and r_c^2, r_e^2 are the number of cheap and expensive bandwidth helper nodes of a newcomer in the first and second rack, respectively. We can always assume that $r_c^1 \leq r_c^2$, by swapping racks if it is necessary. Figure 4.2 shows a scheme of a two rack model.

In the static cost model, the repair bandwidth γ is the same for any newcomer. In the rack model, it depends on the rack where the newcomer is placed. Let $\gamma^1 = \beta_e(r_c^1\tau + r_e^1)$ be the repair bandwidth for any newcomer in the first rack with repair cost $C_T^1 = \beta_e(C_c r_c^1\tau + C_e r_e^1)$, and let $\gamma^2 = \beta_e(r_c^2\tau + r_e^2)$ be the repair bandwidth for any newcomer in the second rack with repair cost $C_T^2 = \beta_e(C_c r_c^2\tau + C_e r_e^2)$. Note that if $r_c^1 = r_c^2$ or $\tau = 1$, then $\gamma^1 = \gamma^2$, otherwise $\gamma^1 < \gamma^2$. As it is mentioned in [DGWR10], in order to represent a distributed storage system, the information flow graph is restricted to $\gamma \geq \alpha$. In the rack model, it is necessary that $\gamma^1 \geq \alpha$ which means that $\gamma^2 \geq \alpha$.

Moreover, unlike the models described in Section 2.5, where it is straightforward to establish which is the set of nodes which minimize the mincut, in the rack model, this set of nodes may change depending on the parameters k, r_c^1, r_c^2, n_1 and τ. We call to this set of newcomers, the *minimum mincut*

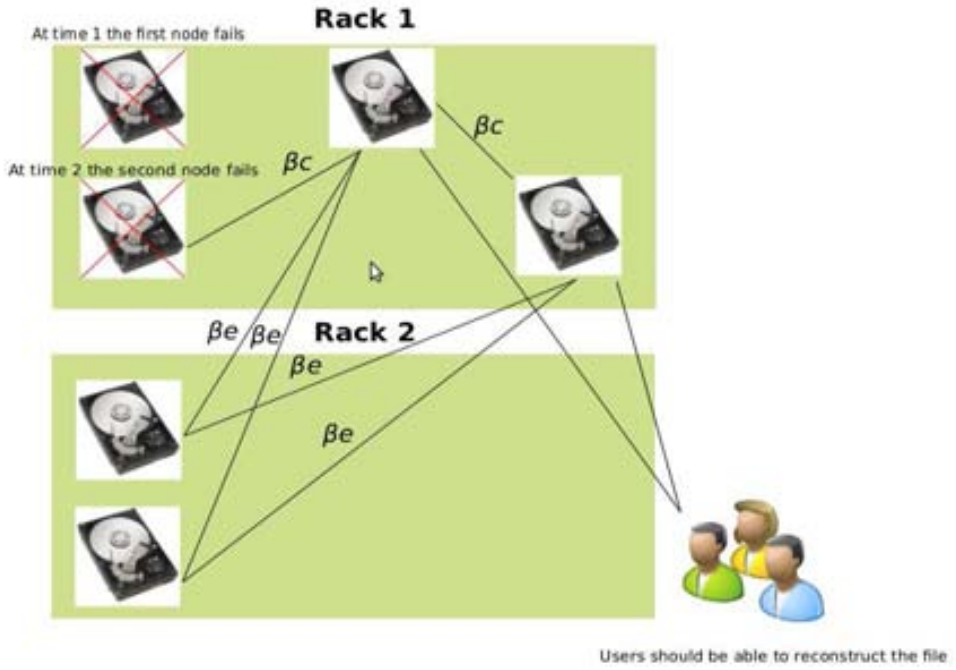

Figure 4.2: Scheme of a two rack model.

set. Recall that the income of a newcomer s_j, $j = n+1, \ldots, \infty$, is the sum of the weights of the arcs that should be cut in order to isolate w_{in}^j from S. Let I be the indexed multiset containing the incomes of k newcomers which minimize the mincut. It is easy to see that in the model described in Subsection 2.5.3, $I = \{(r-i)\beta \mid i = 0, \ldots, k-1\}$, and in the one described in Subsection 2.5.5, $I = \{((r_c - i)\tau + r_e)\beta_e \mid i = 0, \ldots, \min(r_c, k-1)\} \cup \{(r_e - i)\beta_e \mid i = 1, \ldots, k - r_c - 1\}$. Note that when $k \leq r_c + 1$, $\{(r_e - i)\beta_e \mid i = 1, \ldots, k - r_c - 1\}$ is empty. In Figure 2.4, it can be seen that $I = \{(r-i)\beta \mid i = 0, \ldots, k-1\} = \{3\beta, 2\beta\}$. In Figure 2.7, if we fix $k = 5$, $r_c = 3$ and $r_e = 2$ that $I = \{((r_c - i)\tau + r_e)\beta_e \mid i = 0, \ldots, \min(r_c, k-1)\} \cup \{(r_e - i)\beta_e \mid i = 1, \ldots, k - r_c - 1\} = \{(3\tau+2)\beta_e, (2\tau+2)\beta_e, (\tau+2)\beta_e, 2\beta_e, \beta_e\}$.

In order to establish I in the rack model, the set of k newcomers which minimize the mincut must be found. First, note that since $r_c^1 \leq r_c^2$, the income of the newcomers is minimized by replacing first $r_c^1 + 1$ nodes from the rack with less number of helper nodes, which in fact minimizes the mincut. Therefore, the indexed multiset I always contains the incomes of a set of $r_c^1 + 1$ newcomers from W_1. Define $I_1 = \{((r_c^1 - i)\tau + r_e^1)\beta_e \mid i = 0, \ldots, \min(r_c^1, k-1)\}$ as the indexed multiset where $I_1[i]$, $i = 0, \ldots, \min(r_c^1, k-1)$, are the incomes of this set of $r_c^1 + 1$ newcomers from V^1. If $k \leq r_c^1 + 1$, then $I = I_1$, otherwise

$I_1 \subset I$ and $k - r_c^1 - 1$ more newcomers which minimize the mincut must be found. Taking $\tau = 2$ in Figure 4.3, $I_1 = \{((r_c^1 - i)\tau + r_e^1)\beta_e \mid i = 0, \ldots, \min(r_c^1, k-1)\} = \{(\tau+3)\beta_e, 3\beta_e\} = \{5\beta_e, 3\beta_e\}$.

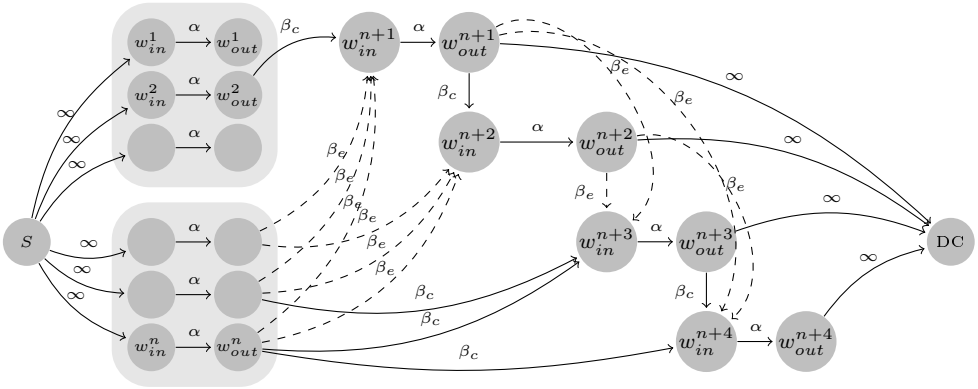

Figure 4.3: Information flow graph corresponding to the rack model when $k > r_c^1 + 1$, with $k = 4$, $r_c^1 = 1$, $r_c^2 = 2$, $r = 4$ and $n_1 = n_2 = 3$.

When $k > r_c^1 + 1$, we will see that there are two possibilities, either the remaining nodes from W_1 are in the set of newcomers which minimize the mincut or not. Define $I_2 = \{r_e^1\beta_e \mid i = 1, \ldots, \min(k - r_c^1 - 1, n_1 - r_c^1 - 1)\} \cup \{(r_c^2 - i)\tau\beta_e \mid i = 0, \ldots, \min(r_c^2, k - n_1 - 1)\}$ as the indexed multiset where $I_2[i]$, $i = 0, \ldots, k - r_c^1 - 2$, are the incomes of a set of $k - r_c^1 - 1$ newcomers, including the remaining $n_1 - r_c^1 - 1$ newcomers from W_1 and newcomers from W_2. Note that if $n_1 - r_c^1 - 1 > k - r_c^1 - 1$, it only contains newcomers from W_1. Define $I_3 = \{(r_c^2 - i)\tau\beta_e \mid i = 0, \ldots, \min(r_c^2, k - r_c^1 - 2)\}$ as the indexed multiset where $I_3[i]$, $i = 0, \ldots, k - r_c^1 - 2$, are the incomes of a set of $k - r_c^1 - 1$ newcomers from W_2. When $r_c^2 < k - r_c^1 - 1$ or $r_c^2 < k - n_1$, according to the information flow graph, the remaining incomes necessary to complete the set of $k - r_c^1 - 1$ newcomers are zero. Therefore, it can be assumed that $r_c^2 \geq k - r_c^1 - 1 \geq k - n_1$, since the mincut equation does not change when $r_c^2 < k - r_c^1 - 1$ or $r_c^2 < k - n_1$. Taking $\tau = 2$ in Figure 4.3, it can be seen that $I_2 = \{r_e^1\beta_e \mid i = 1, \ldots, \min(k - r_c^1 - 1, n_1 - r_c^1 - 1)\} \cup \{(r_c^2 - i)\tau\beta_e \mid i = 0, \ldots, \min(r_c^2, k - n_1 - 1)\} = \{3\beta_e, 2\tau\beta_e\} = \{3\beta_e, 4\beta_e\}$ and $I_3 = \{(r_c^2 - i)\tau\beta_e \mid i = 0, \ldots, \min(r_c^2, k - r_c^1 - 2)\} = \{2\tau\beta_e, \tau\beta_e\}\{4\beta_e, 2\beta_e\}$.

Proposition 3. *If $k > r_c^1 + 1$, we have that $|I_2| = |I_3| = k - r_c^1 - 1$. Moreover, if $\sum_{i=0}^{k-r_c^1-2} I_2[i] < \sum_{i=0}^{k-r_c^1-2} I_3[i]$, then $I = I_1 \cup I_2$; otherwise $I = I_1 \cup I_3$.*

Proof: We need to prove that I_2 and I_3 are the only possible sets of incomes which minimize the mincut. We will see that it is not possible

to find a set of incomes such that the sum of all its elements is less than $\min(\sum_{i=0}^{|I_2|-1} I_2[i], \sum_{i=0}^{|I_3|-1} I_3[i])$.

Let $A = I_2 - (I_2 \cap I_3) = \{a_1, a_2, \ldots, a_n \mid a_i = a_j, i < j\}$ and $B = I_3 - (I_2 \cap I_3) = \{b_1, b_2, \ldots, b_n \mid b_i > b_j, i < j\}$. Let $D = A \cup B = \{d_1, d_2, \ldots, d_{2n} \mid d_i \geq d_j, i < j\}$. Then, $\sum_{i=1}^{n} d_i \geq \sum_{i=1}^{n} b_i$ and $\sum_{i=1}^{n} d_i \geq \sum_{i=1}^{n} a_i$. Note that A, B and D are incomes of an information flow graph which means that one can not add d_2 without having added d_1 to the sum. The same happens with A or B, so the elements must be included in order from the highest to the lowest. □

If $k \leq r_c^1 + 1$, $I = I_1$ and the corresponding mincut equation is

$$\sum_{i=0}^{|I_1|-1} \min(I_1[i], \alpha) \geq M. \tag{4.1}$$

On the other hand, if $k > r_c^1 + 1$ and $I = I_1 \cup I_2$, the corresponding mincut equation is

$$\sum_{i=0}^{|I_1|-1} \min(I_1[i], \alpha) + \sum_{i=0}^{|I_2|-1} \min(I_2[i], \alpha) \geq M, \tag{4.2}$$

and if $I = I_1 \cup I_3$, the equation is

$$\sum_{i=0}^{|I_1|-1} \min(I_1[i], \alpha) + \sum_{i=0}^{|I_3|-1} \min(I_3[i], \alpha) \geq M. \tag{4.3}$$

In the previous models described in Section 2.5, the decreasing behavior of the incomes included in the mincut equation is used to find the threshold function which minimizes the parameters α and γ. In the rack model, the incomes included in the mincut equations may not have a decreasing behavior as the newcomers enter into the system, so it is necessary to find the threshold function in a different way.

Let L be the increasing ordered list of values such that for all i, $i = 0, \ldots, k-1$, $I[i]/\beta_e \in L$ and $|I| = |L|$. Note that any of the information flow graphs, which represent the rack model or any of the two models from Section 2.5, can be described in terms of I, so they can be represented by L. Therefore, once L is found, it is possible to find the parameters α and β_e (and then γ or γ^1 and γ^2) using the threshold function given in the next theorem. Note that the way to represent this threshold function for the rack model can be seen as a generalization, since it also represents the behavior of the mincut equations for the previous given models.

Theorem 26. *The threshold function $\alpha^*(\beta_e)$ (which also depends on r, r_c^1, r_c^2, k and τ) is the following:*

$$\alpha^*(\beta_e) = \begin{cases} \frac{M}{k}, & \beta_e \in [f(0), +\infty) \\ \frac{M-g(i)\beta_e}{k-i}, & \beta_e \in [f(i), f(i-1)) \\ & i = 1, \ldots, k-1, \end{cases} \tag{4.4}$$

subject to $\gamma^1 = (r_c^1\tau + r_e^1)\beta_e \geq \alpha$, where

$$f(i) = \frac{M}{L[i](k-i) + g(i)} \quad \textit{and} \quad g(i) = \sum_{j=0}^{i-1} L[j].$$

Note that $f(i)$ is a decreasing function and $g(i)$ is an increasing function.

Proof: We want to obtain the threshold function which minimizes α, that is,

$$\begin{aligned} \alpha^*(\beta_e) = \quad & \min \alpha \\ & \text{subject to: } \textstyle\sum_{i=0}^{k-1} \min(L[i]\beta_e, \alpha) \geq M. \end{aligned} \tag{4.5}$$

Therefore, we are going to show the optimization of (4.5) which leads to the threshold function (4.4).

Define M^* as

$$M^* = \sum_{i=0}^{k-1} \min(L[i]\beta_e, \alpha).$$

Note that M^* is a piecewise linear function of α. Since L is a sorted list of k values, if α is less than the lowest value $L[0]$, then $M^* = k\alpha$. As α grows, the values from L are added to the equation, so

$$M^* = \begin{cases} k\alpha, & \alpha \in [0, L[0]\beta_e] \\ (k-i)\alpha + \sum_{j=0}^{i-1} L[j]\beta_e, & \alpha \in (L[i-1]\beta_e, L[i]\beta_e] \\ & i = 1, \ldots, k-1 \\ \sum_{j=0}^{k-1} L[j]\beta_e, & \alpha \in (L[k-1]\beta_e, \infty). \end{cases} \tag{4.6}$$

Using that $M^* \geq M$, we can minimize α depending on M. Note that the term $\sum_{j=0}^{k-1} L[j]\beta_e$ of the previous equation has no significance in the minimization of α, so it can be ignored. Therefore, we obtain the function

$$\alpha^* = \begin{cases} \frac{M}{k}, & M \in [0, kL[0]\beta_e] \\ \frac{M - \sum_{j=0}^{i-1} L[j]\beta_e}{k-i}, & M \in (L[i-1]\beta_e(k-i) + \sum_{j=0}^{i-1} L[j]\beta_e, \\ & L[i]\beta_e(k-i) + \sum_{j=0}^{i-1} L[j]\beta_e] \\ & i = 1, \ldots, k-1. \end{cases} \tag{4.7}$$

Finally, define $g(i) = \sum_{j=0}^{i-1} L[j]$ and $f(i) = \frac{M}{L[i](k-i)+g(i)}$. Then, the above expression of α^* can be defined over β_e instead of over M, and the threshold function (4.4) follows. □

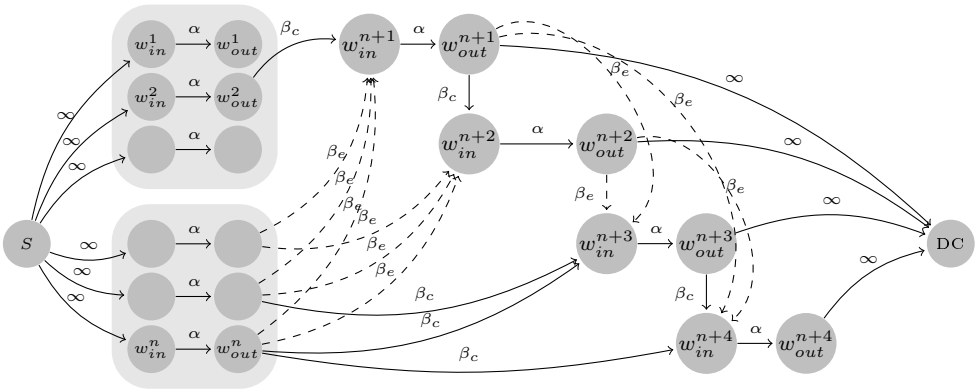

Figure 4.4: Information flow graph corresponding to the rack model when $k > r_c^1 + 1$, with $k = 4$, $r_c^1 = 1$, $r_c^2 = 2$, $r = 4$ and $n_1 = n_2 = 3$.

Example 15. *Figure 4.4 shows the example of an information flow graph corresponding to a regenerating code with $k = 4$, $r_c^1 = 1$, $r_c^2 = 2$, $r = 4$ and $n_1 = n_2 = 3$. Taking for example $\tau = 2$, we have that $I_1 = \{5\beta_e, 3\beta_e\}$, $I_2 = \{3\beta_e, 4\beta_e\}$ and $I_3 = \{4\beta_e, 2\beta_e\}$. By Proposition 3, since $\sum_{i=0}^{1} I_2[i] > \sum_{i=0}^{1} I_3[i]$, $I = I_1 \cup I_3 = \{5\beta_e, 3\beta_e, 4\beta_e, 2\beta_e\}$, and then $L = [2, 3, 4, 5]$. The corresponding mincut equation is (4.3) and applying L to the threshold function (4.4), we obtain*

$$\alpha^*(\beta_e) = \begin{cases} \frac{M}{4}, & \beta_e \in [\frac{M}{8}, +\infty) \\ \frac{M - 2\beta_e}{3}, & \beta_e \in [\frac{M}{11}, \frac{M}{8}) \\ \frac{M - 5\beta_e}{2}, & \beta_e \in [\frac{M}{13}, \frac{M}{11}) \\ M - 9\beta_e, & \beta_e \in [\frac{M}{14}, \frac{M}{13}). \end{cases} \tag{4.8}$$

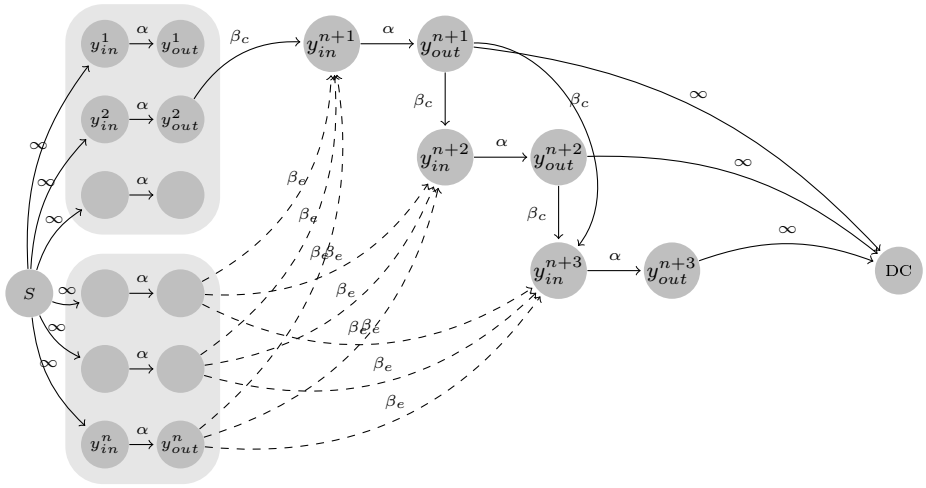

Figure 4.5: Information flow graph corresponding to the rack model when $k > r_c^1 + 1$, with $k = 3$, $r_c^1 = 1$, $r_c^2 = 2$, $r = 4$ and $n_1 = n_2 = 3$.

It can happen that two consecutive values in L are equal, that is $L[i] = L[i-1]$, so $f(i) = f(i-1)$. In this case, we consider that the interval $[f(i), f(i-1))$ is empty and it can be deleted.

Example 16. *Figure 4.5 shows the same example as Figure 4.3 with an information flow graph corresponding to a regenerating code with* $r_c^1 = 1$, $r_c^2 = 2$, $r = 4$ *and* $n_1 = n_2 = 3$, *but taking* $k = 3$ *instead of* $k = 4$. *If for example* $\tau = 2$, *we have that* $I_1 = \{5\beta_e, 3\beta_e\}$, $I_2 = \{3\beta_e\}$ *and* $I_3 = \{4\beta_e\}$. *By Proposition 3, since* $\sum_{i=0}^{0} I_2[i] < \sum_{i=0}^{0} I_3[i]$, $I = I_1 \cup I_2 = \{5\beta_e, 3\beta_e, 3\beta_e\}$, *and then* $L = [3, 3, 5]$. *The corresponding mincut equation is (4.2) and applying* L *to the threshold function (4.4), we obtain*

$$\alpha^*(\beta_e) = \begin{cases} \frac{M}{3}, & \beta_e \in [\frac{M}{9}, +\infty) \\ \frac{M-3\beta_e}{2}, & \beta_e \in [\frac{M}{9}, \frac{M}{9}) \\ M - 6\beta_e, & \beta_e \in [\frac{M}{11}, \frac{M}{9}). \end{cases} \tag{4.9}$$

Note that the second interval is empty and it can be deleted.

Finally, note that when $k \leq r_c^1 + 1$, the mincut equations and the threshold function (4.4) for the rack model are exactly the same as the ones shown in [AKG10] for the model described in Subsection 2.5.5. Actually, it can be seen that r_c^1 of the rack model is equivalent to r_c of the static cost model.

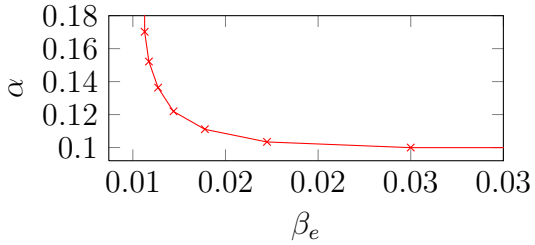

Figure 4.6: Tradeoff curve for the rack model with $M = 1$, $k = 10$, $r_c^1 = 5$, $r_c^2 = 6$, $r = 11$, $n_1 = n_2 = 6$ and $\tau = 2$.

Indeed, it can be seen that when $k \leq r_c^1 + 1$, the rack model and the static cost model have the same behavior because $I = I_1$.

4.1.1 MSR and MBR points

The threshold function (4.4) leads to a tradeoff curve between α and β_e. Note that, like in the static cost model, since there is a different repair bandwidth γ^1 and γ^2 for each rack, this curve is based on β_e instead of γ^1 and γ^2.

At the MSR point, the amount of stored data per node is $\alpha_{MSR} = M/k$. Moreover, at this point, the minimum value of β_e is $\beta_e = f(0) = \frac{M}{L[0]k}$, which leads to

$$\gamma^1_{MSR} = \frac{(r_c^1\tau + r_e^1)M}{L[0]k} \quad \text{and} \quad \gamma^2_{MSR} = \frac{(r_c^2\tau + r_e^2)M}{L[0]k}.$$

On the other hand, at the MBR point, as $f(i)$ is a decreasing function, the parameter β_e which leads to the minimum repair bandwidths is $\beta_e = f(|L| - 1) = \frac{M}{L[|L|-1](k-|L|+1)+g(|L|-1)}$. Then, the corresponding amount of stored data per node is $\alpha_{MBR} = \frac{ML[|L|-1]}{(k-|L|+1)L[|L|-1]+g(|L|-1)}$, and the repair bandwidths are

$$\gamma^1_{MBR} = \frac{(r_c^1\tau + r_e^1)M}{L[|L|-1](k-|L|+1)+g(|L|-1)} \quad \text{and}$$

$$\gamma^2_{MBR} = \frac{(r_c^2\tau + r_e^2)M}{L[|L|-1](k-|L|+1)+g(|L|-1)}.$$

Figure 4.6 shows the tradeoff curve for a rack model with $M = 1$, $k = 10$, $r_c^1 = 5$, $r_c^2 = 6$, $r = 11$, $n_1 = n_2 = 6$ and $\tau = 2$. The MSR point is the one with minimum α while the MBR point is the one with minimum β_e.

4.1.2 Non-feasible situation

As we have seen, the threshold function (4.4) is subject to $\gamma^1 = (r_c^1\tau + r_e^1)\beta_e \geq \alpha$.

Proposition 4. *If the inequality* $\gamma^1 \geq \alpha$ *is achieved, then* $\max(L) = I_1[0]/\beta_e$.

Proof: Since L is an increasing ordered list, for $i = 0, \ldots, k-1$, $\max(L) = L[k-1]$. As $I_1[0]$ is the income of the first newcomer, then $I_1[0]/\beta_e = r_c^1\tau + r_e^1 \in L$. Actually, L is constructed from all elements in I and $I_1 \subseteq I$, by Proposition 3.

If $\gamma^1 \geq \alpha$, then taking $\beta_e = f(k-1)$ in Theorem 26, we have that $\gamma^1 = (r_c^1\tau + r_e^1)\beta_e = (r_c^1\tau + r_e^1)f(k-1) \geq M - g(k-1)f(k-1)$. After some operations, we obtain that $\frac{(r_c^1\tau + r_e^1)M}{\sum_{j=0}^{k-1} L[j]} \geq \frac{L[k-1]M}{\sum_{j=0}^{k-1} L[j]}$, so $r_c^1\tau + r_e^1 \geq L[k-1]$. Since $I_1[0]/\beta_e = r_c^1\tau + r_e^1 \in L$ and $\max(L) = L[k-1]$, $r_c^1\tau + r_e^1 = L[k-1] = I_1[0]/\beta_e$. □

Since any NDSS satisfies that $\gamma^1 \geq \alpha$, we have that $\max(L) = I_1[0]/\beta_e$, by Proposition 4. In order to have this situation, we can use two different approaches. If we allow a non-homogeneous system, then, it is possible to define a different α for each rack as it is shown in [PYGP13]. However, this dissertation is based on a homogeneous behavior for α, so we need to remove from L any value $L[i]$ such that $L[i] > I_1[0]/\beta_e$, $i = 0, \ldots, k-1$. After that, we can assume that $L[|L|-1] = I_1[0]/\beta_e$. In terms of the tradeoff curve, this means that there is no point in the curve that outperforms the MBR point.

Example 17. *In order to illustrate this situation, we can consider the example of a regenerating code with* $k = 3$, $r_c^1 = 1$, $r_c^2 = 4$, $r = 6$, $n_1 = 2$ *and* $n_2 = 5$, *and the information flow graph given in Figure 4.7. Taking* $\tau = 2$, *the incomes of the newcomers* s_{n+1}, s_{n+2} *and* s_{n+3} *are* $7\beta_e$, $5\beta_e$ *and* $8\beta_e$, *respectively. Actually, we have that* $I = I_1 \cup I_2$, *where* $I_1 = \{7\beta_e, 5\beta_e\}$ *and* $I_2 = \{8\beta_e\}$. *Then,* $L = [5, 7, 8]$, *so* $\max(L) = 8 > I[0]/\beta_e = 7$. *Applying* L *to the threshold function (4.4), the resulting minimization of* α *and* β_e *is*

$$\alpha^*(\beta_e) = \begin{cases} \frac{M}{3}, & \beta_e \in [\frac{M}{15}, +\infty) \\ \frac{M-5\beta_e}{2}, & \beta_e \in [\frac{M}{19}, \frac{M}{15}) \\ M - 12\beta_e, & \beta_e \in [\frac{M}{20}, \frac{M}{19}). \end{cases}$$

Note that considering the last interval, we have that for $\beta_e = f(k-1) = \frac{M}{20}$, $\alpha_{MBR} = \frac{8M}{20}$ *and* $\gamma^1_{MBR} = (r_c^1\tau + r_e^1)f(k-1) = \frac{7M}{20}$. *Applied to the*

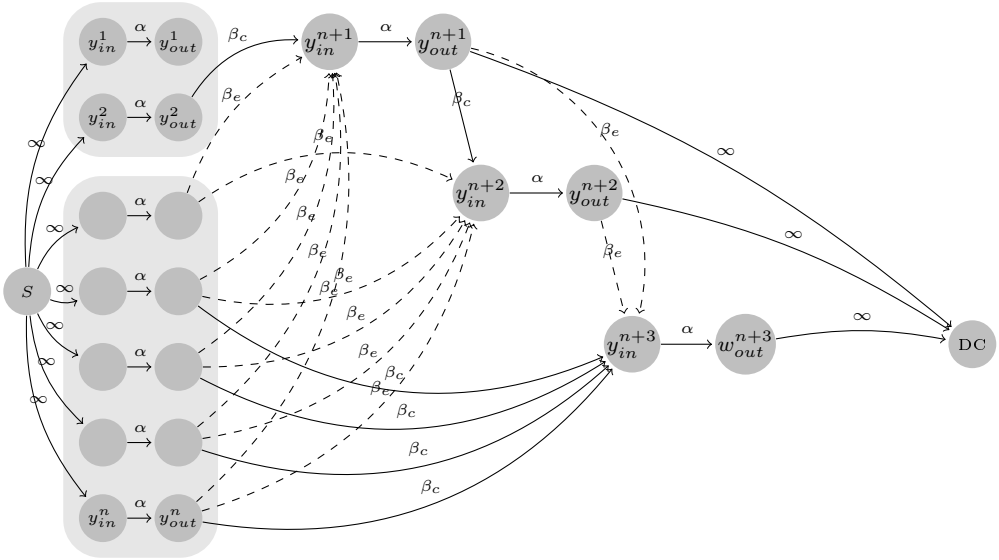

Figure 4.7: Information flow graph with $k = 3$, $n_1 = 2$, $n_2 = 5$, $r_c^1 = 1$, $r_c^2 = 4$ and $r = 6$.

information flow graph, we obtain that mincut$(S, DC) = \frac{7M}{20} + \frac{5M}{20} + \frac{8M}{20} = M$ *which is true. However, since* $\alpha_{MBR} > \gamma_{MBR}^1$*, it gives a non-feasible situation for a distributed storage scheme. Note also that if we delete this non-feasible interval, then* $\gamma_{MBR}^1 = \frac{7M}{19}$ *and* $\alpha_{MBR} = \frac{7M}{19}$ *which corresponds to the MBR point because* $\gamma_{MBR}^1 = \alpha_{MBR}$*.*

It is important to note that more than one element from I can be greater than any element from I_1, which will result in more impossible intervals. In conclusion, any value from I greater than the greatest value from I_1, must be deleted because otherwise it would lead to a non-feasible situation.

4.1.3 Case $r_e^1\beta_e \geq r_c^2\tau\beta_e$

In this case, the mincut equation has a decreasing behavior as i increases for $i = 0, \ldots, k-1$. Therefore, it is possible to define an injective function with a decreasing behavior, which will be used to determine the intervals of the threshold function. Basically, it is possible to use the same procedure shown in [AKG10] and [DGWR10] to find the threshold function. Moreover, it can be seen that the set of incomes which minimize the mincut is always the same, it does not depend on any parameter.

It is easy to see that if $r_e^1\beta_e \geq r_c^2\tau\beta_e$ and $k \leq r_c^1 + 1$, the mincut equations (and so the threshold functions) corresponding to the model explained in this

section and the model explained in Subsection 2.5.5 are exactly the same. Therefore, we will focus on the situation that $r_e^1\beta_e \geq r_c^2\tau\beta_e$ and $k > r_c^1 + 1$. Note that this is in fact a particular case of the general threshold function (4.4), where it is possible to create a decreasing function for any feasible i, and then find the threshold function giving more details.

Theorem 27. *When $r_e^1 \geq r_c^2\tau$ and $k > r_c^1 + 1$, the threshold function $\alpha^*(\beta_e)$ (which also depends on r, r_c^1, r_c^2, k and τ) is the following:*

$$\alpha^*(\beta_e) = \begin{cases} \frac{M}{k}, & \beta_e \in [f_1(0), +\infty) \\ \frac{M - g_1(i)\tau\beta_e}{k-i}, & \beta_e \in [f_1(i), f_1(i-1)) \\ & i = 1, \ldots, k - r_c^1 - 2 \\ \frac{M - g_1(k - r_c^1 - 1)\tau\beta_e}{k-i}, & \beta_e \in [f_2(k - r_c^1 - 1), \\ & f_1(k - r_c^1 - 2)) \\ \frac{M - g_1(k - r_c^1 - 1)\tau\beta_e - g_2(i - k + r_c^1 + 1)\beta_e}{k-i}, & \beta_e \in [f_2(i), f_2(i-1)) \\ & i = k - r_c^1, \ldots, k - 1, \end{cases} \tag{4.10}$$

where

$$g_1(i) = \frac{i}{2}(2r - 2k + i + 1),$$

$$g_2(i) = \frac{i}{2}(2r_e^1 + \tau i - \tau),$$

$$f_1(i) = \frac{2M}{\tau(2k(r-k) + (i+1)(2k-i))}, \text{ and}$$

$$f_2(i) = \frac{2M}{2r_e^1 + 2r_e^1 r_c^1 - \tau(i(i-2k+1) + 2(k^2 - k - kr + r_e^1 + r_e^1 r_c^1))}.$$

Note that $f_1(i)$ and $f_2(i)$, $i = 0, \ldots, k-1$, are decreasing functions, and $g_1(i)$ and $g_2(i)$, $i = 1, \ldots, k-1$, are increasing functions.

Proof: Note that $r_e^1 = r_c^2 + 1$ and $r_e^2 = r_c^1 + 1$. We consider the mincut equation (4.3) of the rack model, since if $r_e^1 \geq r_c^2\tau$, then we have that $I = I_1 \cup I_3$, by Proposition 3. In other words, the $n_1 - r_c^1 - 1$ remaining newcomers from W^1 are not in the set of newcomers which minimizes the mincut. Assume that $k \leq r = r_c^1 + r_e^1$ because if $r < k$, requiring any r storage nodes to have a flow of M will lead to the same condition as requiring any

k storage nodes to have a flow of M [DGWR10]. We want to obtain the threshold function which minimizes α, that is,

$$\begin{aligned} \alpha^*(\beta_e) = \quad & \min \alpha \\ & \text{subject to: } \textstyle\sum_{i=0}^{r_c^1} \min(r_c^1\beta_c + r_e^1\beta_e - i\beta_c, \alpha) + \\ & \textstyle\sum_{i=r_c^1+1}^{k-1} \min((r_c^1 + r_e^1 - i)\beta_c, \alpha) \geq M. \end{aligned} \tag{4.11}$$

Therefore, we are going to show the optimization of (4.11) which leads to (4.10).

Applying that $\beta_c = \tau\beta_e$, we can define the minimum M as M^*, so

$$M^* = \sum_{i=0}^{r_c^1} \min((r_c^1\tau + r_e^1 - i\tau)\beta_e, \alpha) + \sum_{i=r_c^1+1}^{k-1} \min((r_c^1 + r_e^1 - i)\tau\beta_e, \alpha).$$

In order to change the order of the above summation, we define

$$b(i_1, i_2) = r_c^1 + r_e^1 - k + 1 + i_1 + i_2\tau.$$

Note that M^* is a piecewise linear function of α. The minimum value of $\{(r_c^1\tau + r_e^1 - i\tau)\beta_e \mid i = 0, \ldots, r_c^1\} \cup \{(r_c^1 + r_e^1 - i)\tau\beta_e \mid i = r_c^1 + 1, \ldots, k-1\}$ is when $i = k-1$. Therefore, if α is less than this value, then $M^* = k\alpha$. Since $r_e^1 = r_c^2 + 1$ and $r_e^2 = r_c^1 + 1$ the lowest value of $\{(r_c^1\tau + r_e^1 - i\tau)\beta_e \mid i = 0, \ldots, r_c^1\}$ which is $r_e^1\beta_e$, is higher than or equal to the highest value of $\{(r_c^1 + r_e^1 - i)\tau\beta_e \mid i = r_c^1 + 1, \ldots, k-1\}$, which is $(r_e^1 - 1)\tau\beta_e$. This means that as α increases, the term $(r_c^1 + r_e^1 - i)\tau\beta_e$ is added more times in M^* while $i = k-1, \ldots, r_c^1$. When $i = r_c^1, \ldots, 0$, the term $(r_c^1\tau + r_e^1 - i\tau)\beta_e$ is added more times in M^*.

$$M^* = \begin{cases} k\alpha, & \alpha \in [0, b(0,0)\tau\beta_e] \\ (k-i)\alpha + \sum_{j=0}^{i-1} b(j,0)\tau\beta_e, & \alpha \in (b(i-1,0)\tau\beta_e, b(i,0)\tau\beta_e] \\ & i = 1, \ldots, k - r_c^1 - 2 \\ (r_c^1+1)\alpha + \sum_{j=0}^{k-r_c^1-2} b(j,0)\tau\beta_e, & \alpha \in (b(k-r_c^1-2,0)\tau\beta_e, \\ & b(k-r_c^1-1,0)\beta_e] \\ (k-i)\alpha + \sum_{j=0}^{k-r_c^1-2} b(j,0)\tau\beta_e + & \\ \sum_{j=0}^{i-k+r_c^1} b(k-r_c^1-1,j)\beta_e, & \alpha \in (b(k-r_c^1-1, i-k+r_c^1)\beta_e, \\ & b(k-r_c^1-1, i-k+r_c^1+1)\beta_e] \\ & i = k - r_c^1, \ldots, k-1 \\ \sum_{j=0}^{k-r_c^1-2} b(j,0)\tau\beta_e + & \\ \sum_{j=0}^{r_c^1} b(k-r_c^1-1,j)\beta_e, & \alpha \in (b(k-r_c^1-1, r_c^1)\beta_e, \infty). \end{cases} \tag{4.12}$$

Using that $M \geq M^*$, we can minimize α depending on M. Note that the last term of (4.12) does not affect in the minimization of α, so it is ignored. Therefore, we obtain the function

$$\alpha^* = \begin{cases} \frac{M}{k}, & M \in [0, kb(0,0)\tau\beta_e] \\ \frac{M - \sum_{j=0}^{i-1} b(j,0)\tau\beta_e}{k-i}, & M \in (A(i-1), A(i)] \\ & i = 1, \ldots, k - r_c^1 - 2 \\ \frac{M - \sum_{j=0}^{i-1} b(j,0)\tau\beta_e}{k-i}, & M \in (A(i-1), B(i)] \\ \frac{M - \sum_{j=0}^{k-r_c^1-2} b(j,0)\tau\beta_e - \sum_{j=0}^{i-k+r_c^1} b(k-r_c^1-1,j)\beta_e}{k-i}, & M \in (B(i-1), B(i)] \\ & i = k - r_c^1, \ldots, k-1, \end{cases} \tag{4.13}$$

where $A(i) = \tau\beta_e(b(i,0)(k-i-1) + \sum_{j=0}^{i} b(j,0))$ and $B(i) = \beta_e(b(k - r_c^1 - 1, i-k+r_c^1+1)(k-i-1) + \sum_{j=0}^{k-r_c^1-2} b(j,0)\tau + \sum_{j=0}^{i-k+r_c^1+1} b(k-r_c^1-1,j))$.

From the definition of $b(i_1, i_2)$,

$$\sum_{j=0}^{i-1} b(j,0) = \frac{i}{2}(2r - 2k + i + 1) = g_1(i),$$

$$\sum_{j=0}^{i-1} b(k - r_c^1 - 1, j) = \frac{i}{2}(2r_e^1 + \tau i - \tau) = g_2(i),$$

$$\tau((k-i-1)b(i,0) + \sum_{j=0}^{i} b(j,0)) = \frac{2M}{\tau(2k(r-k) + (i+1)(2k-i))} = \frac{M}{f_1(i)}$$

and

$$b(k-r_c^1-1, i-k+r_c^1+1)(k-i-1) + \sum_{j=0}^{k-r_c^1-2} b(j,0)\tau + \sum_{j=0}^{i-k+r_c^1+1} b(k-r_c^1-1, j) = \frac{M}{f_2(i)}.$$

The function (4.13) for α^* can be defined over β_e instead of over M, and then function (4.10) follows. □

4.2 General rack model

Let $m \geq 2$ be the number of racks of a distributed storage system. Let n_j, $j = 1, \ldots, m$, be the number of storage nodes in the j-th rack. Let r_c^j be the number of helper nodes providing cheap bandwidth and r_e^j be the number of helper nodes providing expensive bandwidth to any newcomer in the j-th rack. We assume that the total number of helper nodes r is fixed, so it is satisfied that $r = r_c^j + r_e^j$ for $j = 1, \ldots, m$. Moreover, it can be seen that $r_e^j = \sum_{z=1, z\neq j}^{m} (r_c^z + 1)$. Let the racks be increasingly ordered by number of cheap bandwidth nodes, so $i \leq j$ if and only if $r_c^i \leq r_c^j$. First, we consider the case when $r = n - 1$, and then the general case, that is, when $r \leq n - 1$

4.2.1 When $r = n - 1$

In this case, we impose that any available node in the system is a helper node, that is, $r = n - 1$. If one node fails in the j-th rack, $r_c^j = n_j - 1$ nodes from the same rack and $r_e^j = n - n_j$ nodes from other racks help in the regeneration process.

The indexed multiset I containing the incomes of the k newcomers which minimize the mincut is

$$I = \bigcup_{j=1}^{m} \{((r_c^j - i)\tau + r_e^j - \sum_{z=1}^{j-1}(r_c^z - j + 1)\beta_e \mid i = 0, \ldots, \min(r_c^j, k - \sum_{z=1}^{j-1} r_c^z - j)\}, \tag{4.14}$$

where $\sum_{z=1}^{0} x = 0$ for any value x. Therefore, the resulting mincut equation is $\sum_{i=0}^{k-1} \min(I[i], \alpha) \geq M$.

Finally, the threshold function (4.4) can be applied, so α and β_e can be minimized. Note that the set of k newcomers which minimize the mincut is fixed independently of τ, so there is only one candidate set to be the minimum mincut set.

4.2.2 When $r \leq n - 1$

In this case, there may exist nodes in the system that, after a node failure, do not help in the regeneration process. These kind of systems introduce the difficulty of finding the minimum mincut set in the information flow graph. Note that in the two-rack model, after including the first r_c^1+1 nodes from the first rack, we need to known whether the remaining $n_1 - r_c^1 - 1$ are included in the minimum mincut set or not. In order to solve this point, we create two candidate sets to be the minimum mincut set, one with these nodes and another one without them.

Define the indexed multiset $I' = \bigcup_{j=1}^{m} \{((r_c^j - i)\tau + r_e^j - \sum_{z=1}^{j-1} r_c^z - j + 1)\beta_e \mid i = 0, \ldots, r_c^j\} \cup I^j$, where $I^j = \{(r_e^j - \sum_{z=1}^{j-1} r_c^z - j + 1)\beta_e \mid i = 1, \ldots, n_j - r_c^j - 1\}$ contains the incomes of the remaining $n_j - r_c^j - 1$ newcomers once the first $r_c^j + 1$ storage nodes have already been replaced. Note that I' represents the incomes of all the n newcomers. Also note that in the m-th rack, $(r_e^m - \sum_{z=1}^{m-1} r_c^z - m + 1)\beta_e = 0$, and that Subsection 4.2.1 describes the particular case when $n_j - r_c^j - 1 = 0$ for all $j = 1, \ldots, m$.

We say that a rack is involved in the minimum mincut if at least one of its nodes is in a candidate set to be the minimum mincut set. The involved racks are always the first m' racks, where m' is the minimum number such that $\sum_{j=1}^{m'}(r_c^j + 1) \geq k$. Since the newcomers corresponding to the incomes from $I^{m'}$ are never included in the minimum mincut set, the number of candidate sets to be the minimum mincut set is $2^{m'-1}$. However, as the goal is to find the set having the minimum sum of its corresponding incomes, it is possible to design a linear algorithm with complexity $O(m' - 1)$ to solve this problem. This algorithm is described in the next paragraph.

For all $j = 1, \ldots, m' - 1$, if $\sum_{i=0}^{k-1} I'[i] > \sum_{i=0}^{k-1}(I' - I^j)[i]$, where $I' - I^j$ means removing the elements of I^j inside I', the new I' becomes $I' - I^j$. This process is repeated for every j. Finally, after $m' - 1$ comparisons, we obtain that $I = I'$. Then, we can assure that I contains the incomes of the minimum mincut set of newcomers. Once I is found, we can define L as in the two-rack model and apply the threshold function (4.4) in order to minimize α and β_e.

Example 18. *Let the number of racks be $m = 3$ with $n_1 = 3$, $n_2 = 4$, $n_3 = 4$ and $k = 7$. Let the number of helper nodes for any newcomer be $r = 8$ with $r_c^1 = 1$, $r_c^2 = 2$ and $r_c^3 = 3$, so with $r_e^1 = 7$, $r_e^2 = 6$ and $r_e^3 = 5$. Note that $r_c^1 \leq r_c^2 \leq r_c^3$. The information flow graph corresponding to these parameters is shown in Figure 4.8.*

Since $m' = 3$, the three racks are involved in the minimum mincut and the incomes in I depend on whether the sets I^1 and I^2 are included or not:

- *Including I^1 and I^2: $I'_{\{1,2\}} = \{(\tau + 7)\beta_e, 7\beta_e, 7\beta_e, (2\tau + 4)\beta_e, (\tau + 4)\beta_e, 4\beta_e, 4\beta_e\}$.*
- *Including I^1 but not I^2: $I'_{\{1\}} = \{(\tau + 7)\beta_e, 7\beta_e, 7\beta_e, (2\tau + 4)\beta_e, (\tau + 4)\beta_e, 4\beta_e, 3\tau\beta_e\}$.*
- *Including I^2 but not I^1: $I'_{\{2\}} = \{(\tau+7)\beta_e, 7\beta_e, (2\tau+4)\beta_e, (\tau+4)\beta_e, 4\beta_e, 4\beta_e\ , 3\tau\beta_e\}$.*
- *Excluding I^1 and I^2: $I'_{\emptyset} = \{(\tau+7)\beta_e, 7\beta_e, (2\tau+4)\beta_e, (\tau+4)\beta_e, 4\beta_e, 3\tau\beta_e, 2\tau\beta_e\}$.*

Then, if for example $\tau = 2.2$, the sum of the elements of the above multisets are $45.8\beta_e$, $48.4\beta_e$, $45.4\beta_e$ and $45.8\beta_e$, respectively. So $I = I'_{\{2\}}$ contains the incomes corresponding to the minimum mincut set.

We can obtain the same result by using the algorithm proposed in this section, that is, following these steps:

1. *Create $I' = \{(\tau+7)\beta_e, 7\beta_e, 7\beta_e, (2\tau+4)\beta_e, (\tau+4)\beta_e, 4\beta_e, 4\beta_e\, 3\tau\beta_e, 2\tau\beta_e, \tau\beta_e, 0\}$.*
2. *Create $I^1 = \{7\beta_e\}$. Since $\sum_{i=0}^{6} I'[i] = 45.8\beta_e > \sum_{i=0}^{6}(I' - I^1)[i] = 45.4\beta_e$, the new I' becomes $I' = I' - I^1 = I_{\{2\}}$.*
3. *Create $I^2 = \{4\beta_e\}$. Since $\sum_{i=0}^{6} I'[i] = 45.4\beta_e \leq \sum_{i=0}^{6}(I' - I^2)[i] = 45.8\beta_e$, $I = I' = I'_{\{2\}}$ and $\sum_{i=0}^{6} I[i] = 45.4\beta_e$.*

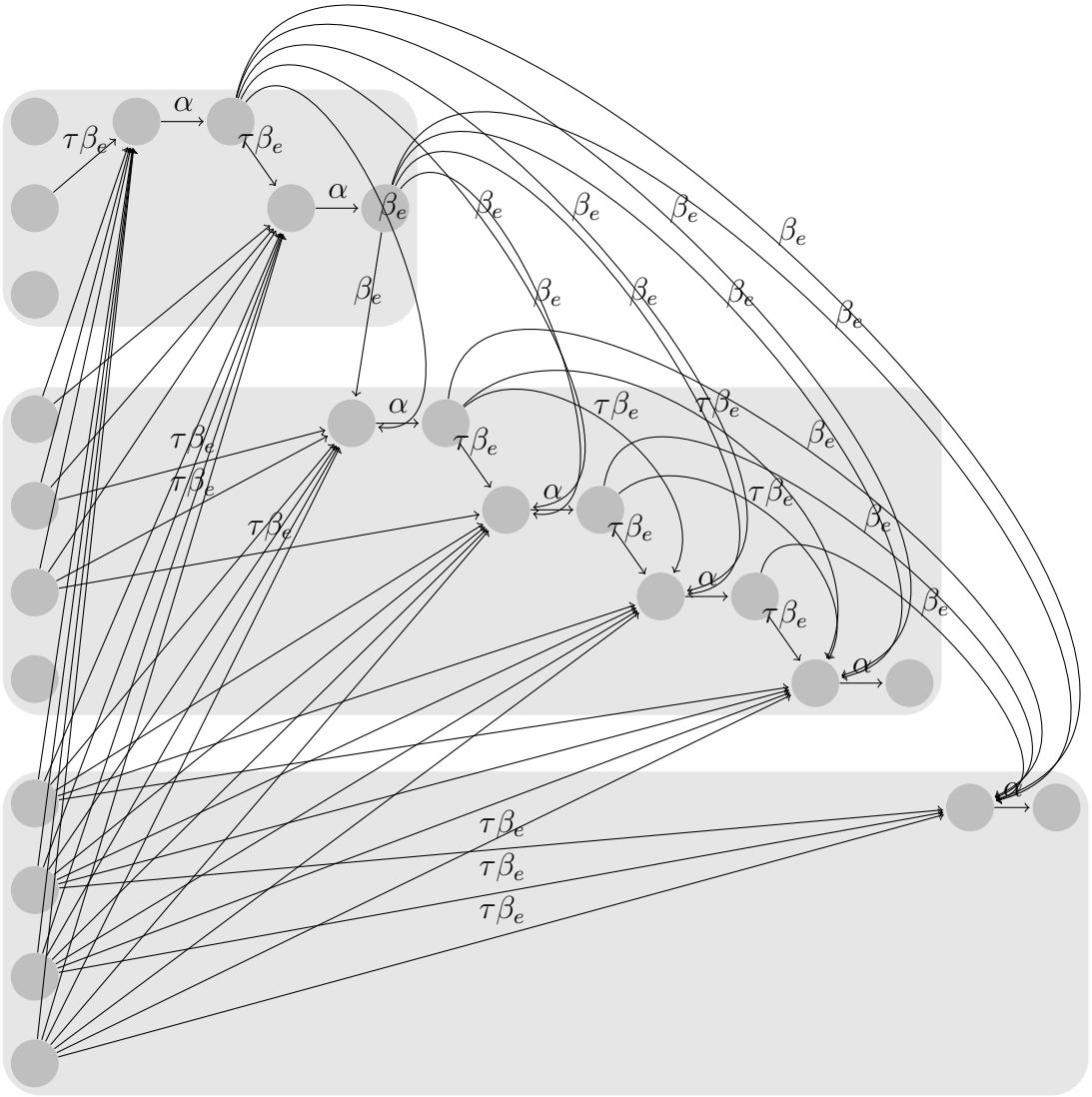

Figure 4.8: Information flow graph corresponding to the rack model with $k = 7$, $r_c^1 = 1$, $r_c^2 = 2$, $r_c^3 = 3$ and $r = 8$.

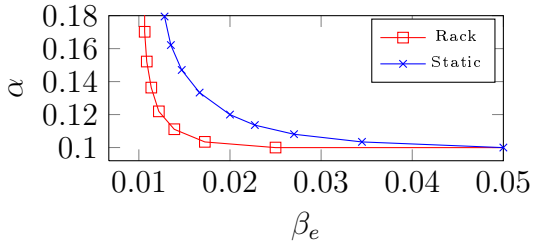

Figure 4.9: Chart comparing the rack model with the static cost model for $M = 1$, $k = 10$, $r_c^1 = 5$, $r_c^2 = 6$, $r = 11$, $n_1 = n_2 = 6$ and $\tau = 2$.

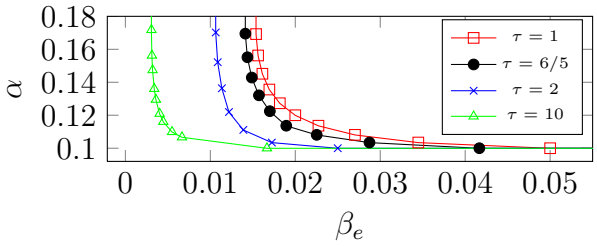

Figure 4.10: Chart showing the tradeoff curves between α and β_e for $M = 1$, $k = 10$, $r_c^1 = 5$, $r_c^2 = 6$, $r = 11$ and $n_1 = n_2 = 6$, so with $k > r_c^1 + 1$.

4.3 Analysis

When $\tau = 1$, we have that $\beta_e = \beta_c$, so $\gamma^j = \gamma = r\beta_e$ for any j. This situation corresponds to the case when the three models shown in Subsections 2.5.3, 2.5.5 and Section 4.2 coincide in terms of the threshold function, since we can assume that $\beta_c = \beta_e = \beta$. When $\tau > 1$ and $k \leq r_c^1 + 1$, the rack model coincides with the static cost model described in Subsection 2.5.5.

In order to compare the rack model with the static cost model when $\tau > 1$ and $k > r_c^1 + 1$, it is enough to consider the case $m = 2$. Moreover, it only makes sense to consider the equation $C_T^1 = \beta_e(C_c r_c^1 \tau + C_e r_e^1)$. Using the definitions given for the static cost model and the rack model, note that $r_c = r_c^1$ and $r_e = r_e^1$. When comparing both models using C_T^1, all the parameters are the same except for $\beta_e = f(i) = \frac{M}{L[i](k-i)+g(i)}$. Now, we are going to prove that the resulting L will always be greater in the rack model, so both β_e and C_T^1 will be less.

Assume that the incomes are in terms of I. For the static cost model, $I = \{((r_c^1 - i)\tau + r_e^1)\beta_e \mid i = 0, \ldots, r_c^1\} \cup \{(r_e^1 - i)\beta_e \mid i = 1, \ldots, k - r_c^1 - 1\}$. Note that $\{(r_e^1 - i)\beta_e \mid i = 1, \ldots, k - r_c^1 - 1\} = \{(r_c^2 - i)\beta_e \mid i = 0, \ldots, k - r_c^1 - 2\}$. In

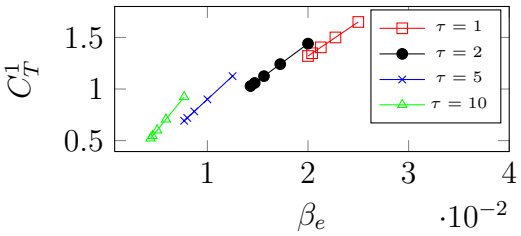

Figure 4.11: Chart showing the repair cost in the rack model for $M = 1$, $k = 5$, $r_c^1 = 5$, $r_c^2 = 6$, $d = 11$, $n_1 = n_2 = 6$, $C_c = 1$ and $C_e = 10$. The points correspond to the $k = 5$ values given by $f(i)$, $i = 0, \ldots, 4$.

this case, both models are equal for the first $r_c^1 + 1$ newcomers, and different for the remaining $k - r_c^1 - 1$ newcomers. If $I = I_1 \cup I_3$ for the rack model, the incomes of the remaining $k - r_c^1 - 1$ newcomers from the second rack are $(r_c^2 - i)\tau\beta_e$, which are greater than $(r_c^2 - i)\beta_e$ of the static cost model. If $I = I_1 \cup I_2$, it can also be seen that $r_e^1\beta_e > (r_e^1 - i)\beta_e$. Finally, we can say that the repair cost in the rack model is less than the repair cost in the static cost model.

The comparison between both, the rack model and the static model, is shown in Figure 4.9 when $M = 1$, $k = 10$, $r_c^1 = 5$, $r_c^2 = 6$, $r = 11$, $n_1 = n_2 = 6$ and $\tau = 2$. It can be seen that the curve of the rack model is below the curve of the static model, which means that the rack model requires less stored data per node α and less expensive repair bandwidth β_e than the static model. As β_e is decreased in the rack model, this means that the repair cost is going to be less in the rack model than in the static model.

The decreasing behavior of β_e as τ increases is shown in Figure 4.10 by giving several tradeoff curves for different values of τ. As we have said, if β_e is decreased, the repair cost is also decreased. This fact is shown in Figure 4.11, where it can be seen that the repair cost decreases as τ increases. Summarizing, when τ is increased, β_e decreases which also decreases the repair cost.

Chapter 5

Conclusions

The information age is the current period in human history where there is a shift from traditional industry to an economy based on information computerization. In this new era, the amount of digital data is increasing exponentially each year and the physical place where this data is stored and treated is no longer defined by the users or authors of such data [Hi011]. The computer cloud is a perfect example of this context.

Despite the fact that stored data can be accessed by users from the Internet as if by magic, the companies or organizations in charge of it must assure its security and its persistence. The data is usually stored in a Network Distributed Storage System (NDSS), a system composed of multiple independent storage nodes. However, the cost of maintaining those NDSS not only in terms of money, but also in terms of space or ecological cost, is high and must be addressed.

In this dissertation, we assume that the amount of data stored in a NDSS is minimized by using coding theory, a well known mathematical art used for data transmission. The increasing use of coding theory techniques in the current storage systems, by some of the most influential companies like Facebook or Google, is a clear example of such advantage. However, coding theory is not focused on solving some of the problems that NDSS introduces.

Firstly, we have explained the current state of the art of codes applied to NDSS. We have seen some of the most important coding theory concepts and parameters, and we have shown the advantages and problems of using codes in a NDSS. It is clear that the use of codes in NDSS has some huge advantages, specially regarding the minimization of the redundancy needed to assure the persistence of the stored data. However, coding theory techniques also introduce some problems, like the extra amount of bandwidth needed to

regenerate a failed node (repair problem) or data insertion, among others.

In this dissertation, we have seen that there are two different approaches to address the repair problem. On the one hand, the codes designed for NDSS which address the repair problem by increasing the locality of the coordinates stored in the storage nodes. This kind of codes not only decrease the repair bandwidth, but also the repair degree which is the number of other storage nodes needed to regenerate a failed one. However, they do that at the cost of decreasing the fault tolerance of the system, so the persistence of the data is reduced.

On the other hand, the regenerating codes address the repair problem by using network coding techniques. The regenerating codes treat the codes as a black box and use network coding to decrease the repair bandwidth. In this case, the fault tolerance is maintained at the cost of some extra computational complexity in the storage nodes. This dissertation is based on regenerating codes.

In real environments like a data center or a P2P system, the computational cost introduced by the use of regenerating codes is a big issue that must be addressed. If regenerating codes are used, the cost of doing linear combinations in both the helper nodes and the newcomers is a real problem. This is because most of the storage nodes are composed of storage devices without computational resources. Moreover, it is also important from a practical point of view that the repair used is exact which complicates the problem. It has been proven in [HLS13], that it is impossible to achieve an exact repair-by-transfer (an exact repair without linear combinations) when the regenerating code has the minimum storage overhead.

5.1 Main Results

In the first contribution, we construct a family of low complexity flexible regenerating codes using quasi-cyclic codes, where a specific set of helper nodes is used to repair a storage node failure. The first construction is designed to minimize the storage per node, the resulting codes are called quasi-cyclic flexible minimum storage regenerating (QCFMSR) codes. We provide an exact repair solution for all parameters achieving $r = k + 1$ and $n = 2k$. This construction is minimum according to the MSR point in the fundamental tradeoff curve. Moreover, QCFMSR codes have a very simple regenerating algorithm that approaches to the repair-by-transfer property. In our solution, the helper nodes do not need to do any linear combination

among their symbols. The only linear combination is done in the newcomer to obtain the symbols the first time that it enters into the system. As far as we are concerned, this is the first construction achieving this repair simplicity for the MSR point. We also claim that such codes exist with high probability. Moreover, it is shown in [SRKR10] and [SRKR11] that when $r < 2k-3$, exact MSR codes do not exist. However, QCFMSR codes exist for $r = k+1$ which satisfies $r < 2k-3$ for $k > 4$. These facts illustrate the importance of the flexibility over the set of helper nodes in this construction.

From a corporate point of view, it is interesting to have codes with high rates, since these are the ones desired for actual data centers. Despite there are constructions with an equal [TWB11] or a higher [PD11] rate than QCFMSR codes, their other properties (uncoded repair at the helper nodes, low decoding and repairing complexity, good rate, low repair degree $r = k+1$ and exact repair) makes them very interesting.

The second construction is designed to minimize the repair bandwidth, the resulting codes are called quasi-cyclic flexible regenerating codes with minimum bandwidth. To construct them, we use a technique shown in [RR10] and [SRKR12] which provide minimum bandwidth codes from existing MSR codes using graphs. We analyze and prove this construction giving bounds on the parameters of these codes. This construction gives the minimum possible bandwidth $\gamma = \alpha$ achieved by an specific set of helper nodes and it has the repair-by-transfer property. Finally, we show that QCFMSR codes can be used as base codes to construct quasi-cyclic flexible regenerating codes with minimum bandwidth. We provide the conditions needed on the parameters $\{n, k, d, \bar{n}, \bar{k}, \bar{r}\}$ for both cases, when $\bar{k} \leq \bar{r}+1$ and $\bar{k} > \bar{r}+1$.

The second contribution is the design of a new mathematical model used to represent a data center, where the storage nodes are placed in racks. In this new model, the cost of downloading data units from nodes in different racks is introduced. That is, the cost of downloading data units from nodes located in the same rack is much lower than the cost of downloading data units from nodes located in a different rack. The rack model is an approach to a more realistic distributed storage environment like the ones used in companies dedicated to the task of storing information.

Firstly, the rack model is deeply analyzed in the case that there are two racks. The differences between this model and previous models are shown. Due to it is a less simplified model compared to the ones presented previously, the rack model introduces more difficulties in order to be analyzed. The main contribution in this case, is the generalization of the process to

find the threshold function of a distributed storage system. This new generalized threshold function fits in the previous models and allows to represent the information flow graphs considering different repair costs. We have also provided the tradeoff curve between the repair bandwidth and the amount of stored data per node and we have compared it with the ones found in previous models. We have analyzed the repair cost of this new model, and we can conclude that the rack model outperforms previous models in terms of repair cost.

Finally, we have also studied the general rack model where there are $m \geq 2$ racks. This generalization represents two main contributions: the making of a model of a distributed storage system using any number of racks, and the description of the algorithm to find the minimum mincut set of newcomers (which is a new problem compared to the previous models). Once the minimum mincut set is found, we can apply the generalized threshold function which shows the minimum tradeoff between the amount of stored data per node and the repair bandwidth needed to regenerate a failed node.

5.2 Further Research

From the point of view of the rack model, it is for further research the case where there are three different costs: one for nodes within the same rack, another for nodes within different racks but in the same data center, and a third one for nodes within different data centers. It would also be important to give some constructions that achieve the optimal bounds. Constructing such regenerating codes is not trivial and we don't know if it is possible. However, the bounds and optimal tradeoff given in this dissertation is an initial step to provide a comparison point for further research on this field. It is also interesting to study locally repairable codes within a rack. The rack model provides some conceptual locality given by each rack and each data center, so it can be interesting to see how LRC performs in these kind of environments. In our opinion, LRC is probably the most natural solution to the problem of applying coding theory to racks. However, our dissertation can be also interesting for LRC, because there are no bounds on the efficiency of codes in terms of repair bandwidth neither in terms of fault tolerance for LRC. This dissertation is an initial step in designing LRC for racks and comparing them with the provided optimal tradeoff.

In general, we have focused on solving the problem of coding theory in real NDSS. From a big picture point of view, a lot of work can still be

done. Computational complexity is a problem which is not usually addressed by researchers but which achieves a high importance. Moreover, we have theoretical limits to the efficiency of the repair and reconstruction in NDSS that can be broken by using new models which adopt the specific topology of the NDSS. For example, we have seen that by using the rack model or the flexible regenerating codes, the theoretical bounds given in [DGWR10] can be supersede.

Another interesting and unaddressed problem is the introduction of the file context. In this moment, the research on NDSS is based on the assumption that the file is a black box of bits. However, compression techniques that use the context of the files achieve higher compression rates than general compression techniques. Designing a NDSS using coding and compression could be an interesting abstraction to provide a different point of view. Conceptually, if the context of the file is added, we should be able to provide a better compression rate, which means that the stored data can be reduced.

There are other codes that could be used in NDSS apart from LRC or regenerating codes. Convolutional codes have some interesting locality properties, since each redundancy symbol can be composed of a small set of other symbols. This property can reduce the repair degree of the NDSS. Moreover, if some memory is used in the convolutional code, each coordinate might have multiple repair alternatives. From this perspective, other codes like LDPC may also be interesting. LDPC have the desired property of containing a lot of zeros in their parity check matrix. This means that the words of the dual have low weight and low repair degree which leads to a high locality. Moreover its decoding algorithm is simple which reduces the computation complexity needed to reconstruct a file.

Bibliography

[ACLY00] R. Ahlswede, N. Cai, S.-Y.R. Li, and R.W. Yeung. Network information flow. *IEEE Trans. Inf. Theory*, 46:1204–1216, 2000.

[AGSS11] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica. Disk-locality in datacenter computing considered irrelevant. In *USENIX HotOS*, page 12, 2011.

[AKG10] S. Akhlaghi, A. Kiani, and M.R. Ghanavati. A fundamental trade-off between the download cost and repair bandwidth in distributed storage systems. *IEEE Int. Symp. on Network Coding NetCod*, pages 1–6, 2010.

[Apa12] Apache. Hadoop distributed file system using raid coding. http://wiki.apache.org/hadoop/HDFS-RAID, 2012.

[BBBM11] M. Blaum, J. Brady, J. Bruck, and J. Menon. Quasicyclic mds codes for distributed storage with efficient exact repair. In *Proceedings of the IEEE Information Theory Workshop*, pages 45 – 49, 2011.

[Ber01] C. Berge. *The Theory of Graphs*. Dover Publications, 2001.

[Coh09] B. Cohen. The bittorrent protocol specification. http://www.bittorrent.org/beps/bep_0003.html, 2009.

[Dav79] P.J. Davis. *Circulant Matrices*. John Wiley and Sons, New York, USA, 1979.

[DB08] A. Duminuco and E. Biersack. Hierarchical codes: How to make erasure codes attractive for peer-to-peer storage systems. In *Eighth International Conference on Peer-to-Peer Computing*, pages 89 – 98, 2008.

[DGWR10] A.G. Dimakis, P.B. Godfrey, M.G. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. *IEEE Trans. Inf. Theory*, 56(9):4539–4551, 2010.

[DRW11] A.G. Dimakis, K. Ramchandran, and Y. Wu. A survey on network codes for distributed storage. *IEEE Trans. Inf. Theory*, 99:1204–1216, 2011.

[Eli54] P. Elias. Error-free coding. *IEEE Trans. Inf. Theory*, 4:29–37, 1954.

[Eva08] B.G. Evans, editor. *Satellite Communication Systems.* The Institution of Engineering and Technology, 3rd edition, 2008.

[For81] Internet Engineering Task Force. Rfc 793. http://www.ietf.org/rfc/rfc793.txt, 1981.

[GDJ05] P. Golden, H. Dedieu, and K. Jacobsen, editors. *Fundamentals of DSL Technology.* Auerbach Publications, 2005.

[GHSY12] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin. On the locality of codeword symbols. *IEEE Transactions on Information Theory*, 58:6925–6934, 2012.

[GPV11a] B. Gastón, J. Pujol, and M. Villanueva. Quasi-cyclic minimum bandwidth regenerating codes. In *Proceedings of the International Castle Meeting on Coding Theory and Applications*, 2011.

[GPV11b] B. Gastón, J. Pujol, and M. Villanueva. Quasi-cyclic minimum storage regenerating codes for distributed data compression. In *Proceedings of the Data Compression Conference*, pages 33–42, 2011.

[GPV13a] B. Gastón, J. Pujol, and M. Villanueva. Quasi-cyclic regenerating codes. *IEEE Transactions on Information Theory (under revision). arXiv:1209.3977[cs.IT]*, 2013.

[GPV13b] B. Gastón, J. Pujol, and M. Villanueva. A realistic distributed storage system that minimizes data storage and repair bandwidth. In *Data Compression Conference*, page 491, 2013.

[GPV13c] B. Gastón, J. Pujol, and M. Villanueva. A realistic distributed storage system: the rack model. *arXiv:1302.5657[cs.IT]*, 2013.

[Hal35] P. Hall. On representatives of subsets. *J. London Math. Soc.*, 10:26–30, 1935.

[Har69] Frank Harary. Determinants, permanents and bipartite graphs. *Mathematical Association of America*, (3):146–148, 1969.

[Hi011] The world's technological capacity to store, communicate, and compute information. *Science*, pages 692–693, February 2011.

[HLS13] Yuchong Hu, Patrick P. C. Lee, and Kenneth W. Shum. Analysis and construction of functional regenerating codes with uncoded repair for distributed storage systems. 1013.

[HSX+12] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin. Erasure coding in windows azure storage. In *Proc. of the USENIX technical conference*, pages 2–2, 2012.

[Hua07] C. Huang. Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems. In *IEEE International Symposium on Network Computing and Applications*, pages 79–86, 2007.

[Imm94] K. A. S. Immink. *Reed-Solomon Codes and Their Applications*, chapter Reed-Solomon Codes and the Compact Disc. IEEE Press, 1994.

[Inc07] Dropbox Inc. Dropbox. http://www.dropbox.com/, 2007.

[Inc12] Google Inc. Google drive. http://www.drive.google.com/, April 2012.

[LN96] R. Lidl and H. Niederreiter. *Finite Fields*. Cambridge University Press, 1996.

[Med03] The benefits of coding over routing in a randomized setting. In *IEEE Int. Symp. on Information Theory*, 2003.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[MS77] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland Publishing, New York, USA, 1977.

[OD11a] F. Oggier and A. Datta. Self-repairing codes for distributed storage - a projective geometric construction. In *IEEE Information Theory Workshop (ITW)*, 2011.

[OD11b] F. Oggier and A. Datta. Self-repairing homomorphic codes for distributed storage systems. In *Infocom, The 30th IEEE International Conference on Computer Communications*, 2011.

[OD12] F. Oggier and A. Datta. *Coding Techniques for Repairability in Network Distributed Storage Systems.* http://sands.sce.ntu.edu.sg/CodingForNetworkedStorage/pdf/longsurvey.pdf, 2012.

[PD11] D. Papailiopoulos and A.G. Dimakis. Distributed storage codes through hadamard designs. In *Proc. of the IEEE ISIT*, pages 1230–1234, 2011.

[PD12] D. Papailiopoulos and A.G. Dimakis. Locally repairable codes. In *Proc. of the IEEE ISIT*, 2012.

[PGK88] D.A Patterson, G. Gibson, and R.H. Katz. A case for redundant arrays of inexpensive discs (raid). In *ACM SIGMOD International Conference on Managment of Data*, pages 109–116, 1988.

[PJ11] L. Pamies-Juarez. *On the design and optimization of heterogeneous distributed storage systems.* PhD thesis, Universitat Rovira i Virgili, Tarragona, 2011.

[PJHH13] L. Pamies-Juarez and F. Oggier H. Hollmann. Locally repairable codes with multiple repair alternatives. In *ISIT*, 2013.

[PJOD13] L. Pamies-Juarez, F. E. Oggier, and A. Datta. Data insertion and archiving in erasure-coding based large-scale storage systems. In *ICDCIT*, pages 47–68, 2013.

[PLD+12] D. Papailiopoulos, J. Luo, A.G. Dimakis, C. Huang, and J. Li. Simple regenerating codes: network coding for cloud storage. In *IEEE International Conference on Computer Communications (Infocom)*, page 2801, 2012.

[PYGP13] Jaume Pernas, Chau Yuen, Bernat Gastón, and Jaume Pujol. Non-homogeneous Two-Rack model for distributed storage systems. In *2013 IEEE International Symposium on Information Theory (ISIT'2013)*, Istanbul, Turkey, jul 2013.

[RL05] Rodrigo Rodrigues and Barbara Liskov. High availability in dhts: Erasure coding vs. replication. In *IPTPS*, pages 226–239, 2005.

[RR10] S. El Rouayheb and K. Ramchandran. Fractional repetition codes for repair in distributed storage systems. In *Proceedings of the Allerton conference on Communication, Control, and Computing*, 2010.

[RSK11] 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 Trans. Inf. Theory*, 57(8):5227–5239, August, 2011.

[Sha49] C. E. Shannon. Communication in the presence of noise. In *Proceedings Institute of Radio Engineers*, volume 37, pages 10–21, 1949.

[SRKR10] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran. Explicit codes minimizing repair bandwidth for distributed storage. In *Proceedings IEEE ITW*, Cairo, 2010.

[SRKR11] 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 Trans. Inf. Theory*, 58:2134–2158, 2011.

[SRKR12] N.B. Shah, K.V. Rashmi, P.V. Kumar, and K. Ramchandran. Distributed storage codes with repair-by-transfer and non-achievability of interior points on the storage-bandwidth tradeoff. *IEEE Trans. Inf. Theory*, 58:1837–1852, 2012.

[TS02] A. Thangaraj and C. Sankar. Evenodd: an optimal scheme for tolerating double disk failures in raid architectures. In *Proceedings of the 21st Annual International Symposium on Computer Architecture*, 2002.

[TWB11] I. Tamo, Z. Wang, and J. Bruck. Mds array codes with optimal rebuilding. In *Proceedings IEEE ISIT*, 2011.

[VYL12] V.T. Van, C. Yuen, and J. Li. Non-homogeneous distributed storage systems. In *Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing*, 2012.

[Wei] E.W. Weisstein. Graph. http://www.drive.google.com/.

[WK02] H. Weatherspoon and J.D. Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In *In Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS) 2002*, pages 328–338, 2002.

[YSS11] Q. Yu, K.W. Shum, and C.W. Sung. Minimization of storage cost in distributed storage systems with repair consideration. In *Proceedings of the IEEE GLOBECOM*, pages 1–5, 2011.

[Zip89] R. Zippel. An explicit separation of relativised random polynomial time and relativised deterministic polynomial time. *Journal Information Processing Letters*, Volume 33 Issue 4, December 1989.

Bernat Gastón Brasó
Cerdanyola del Vallès, September 2013