Explicit Code Constructions for Distributed Storage
Minimizing Repair Bandwidth

A Project Report
Submitted in partial fulfilment of the
requirements for the Degree of
Master of Engineering
in
Telecommunication

by

Rashmi K. V.

under the guidance of
Prof. P. Vijay Kumar

Department of Electrical Communication Engineering
Indian Institute of Science, Bangalore
Bangalore – 560 012 (INDIA)

June 2010
To my parents
Acknowledgments

With deep sense of gratitude, I thank my advisor Prof. P. Vijay Kumar for his guidance and support throughout this work. He has been very encouraging and motivating. The discussions and meetings with him always provided great insights and made the problem at hand look much simpler. The informal environment of the lab has been very conducive to and supportive of research work.

I would like to thank Prof. Kannan Ramchandran, for introducing the problem to us and also for his timely guidance and feedback.

This work is done in collaboration with Nihar B. Shah. I extend my sincere gratitude to him, for the myriads of discussions that we had, which has finally led to these results.

I thank my project examiners Prof. Anurag Kumar and Prof. Chockalingam for reviewing this work.

Many thanks to my lab mates Poornima, Gagan, Vinodh, Lalitha, Prakash, Abu, Ramanathan, Avik, Ravi Teja and Sharath for all the help.

Finally, I would like to extend my greatest appreciation to my family and friends for their constant support.
Abstract

Regenerating codes are a class of recently developed codes for distributed storage, that permit data recovery from any $k$ of $n$ nodes, and also have the capability of repairing a failed node by connecting to any $d$ nodes and downloading an amount of data, termed the repair bandwidth, that is on average, significantly less than the size of the data file. These codes optimally trade the storage space with the repair bandwidth, and the resulting tradeoff is termed ‘storage-repair bandwidth tradeoff’. While the existence of regenerating codes has been proven, no general explicit constructions were available prior to this work.

The main focus of our work is on obtaining explicit constructions for regenerating codes. The second part of our work answers an open problem in the area of regenerating codes related to the achievability of the tradeoff under the additional constraint of replacing a failed node by its exact replica. A new framework for regenerating codes is introduced which provides a great deal of flexibility compared to the original setup, and can potentially decrease the data recovery time by a considerable amount.
# Contents

Acknowledgments i

Abstract ii

1 Introduction 1

1.1 Background .................................................. 2
  1.1.1 Regenerating Codes .................................. 2
  1.1.2 The Storage-Repair Bandwidth Tradeoff ............... 4
  1.1.3 MSR and MBR points on the tradeoff .................. 6

1.2 Additional Terminology ................................... 7
  1.2.1 Striping of Data ........................................ 7
  1.2.2 Exact versus functional regeneration ................. 8
  1.2.3 Systematic regenerating codes ......................... 8
  1.2.4 Linear regenerating codes ............................ 9

1.3 Overview of the Literature ................................ 9

1.4 Summary of the Results ................................... 10

1.5 Organization of the Report ................................ 12

2 Explicit Codes via a Product-Matrix Framework 14

2.1 Framework Description .................................... 15
  2.1.1 Product-Matrix ......................................... 15
  2.1.2 Dispersion and Block-Symmetry ....................... 15
  2.1.3 Data Encoding for the Storage Nodes ................. 16
  2.1.4 Reconstruction of Data by the Data Collector ......... 16
  2.1.5 Regeneration of a Failed Node ....................... 17

2.2 The Product-Matrix MSR Code Construction ............... 18
  2.2.1 An Example for the Product-Matrix MSR code .......... 22
  2.2.2 Systematic Version of the Code ....................... 23
  2.2.3 Explicit MSR Product-Matrix Codes for \(d \geq 2k - 2\) .... 24

2.3 The Product-Matrix MBR Code Construction ............... 25
  2.3.1 An Example for the Product-Matrix MBR Code .......... 27
### CONTENTS

2.3.2 Systematic Version of the Code .............................................. 28
2.4 Analysis and Advantages of the Codes ...................................... 29
  2.4.1 Reduced Overhead ............................................................ 29
  2.4.2 Applicability to Arbitrary $n$ ............................................. 29
  2.4.3 Complexity ................................................................. 30
2.5 An Ideal Regenerating Code: Simultaneously Minimizing Storage and Bandwidth ......................................................... 30
  2.5.1 Description of the Setting ................................................. 31
  2.5.2 Explicit Ideal Regenerating Code ....................................... 32

3 Subspace Framework: Non-achievability of the Interior Points of the Storage-Repair Bandwidth Tradeoff ................................. 35
  3.1 Subspace Framework Description ............................................ 35
  3.2 A Representation for the Points on the Tradeoff ......................... 36
  3.3 Subspace Properties of Linear Exact Regenerating Codes .............. 37
  3.4 Non-achievability of the Interior Points On the Tradeoff for Exact Regeneration ......................................................... 40

4 Interference Alignment for Minimum Storage Regenerating Codes 46
  4.1 Notation .................................................................................. 47
  4.2 Interference Alignment in Regenerating Codes ............................ 50
  4.3 Construction of the MISER Code ............................................. 51
    4.3.1 An Example .................................................................... 51
    4.3.2 The MISER Code for $n = 2k$, $d = n - 1$ ........................ 57
    4.3.3 The MISER Code for $n \geq 2k$, $d = n - 1$ ....................... 60
    4.3.4 Extension to the case $2k - 1 \leq d \leq n - 1$ .................... 61
    4.3.5 Analysis of the MISER Code .......................................... 62
  4.4 Non-Existence for $d < 2k - 3$ ................................................. 62
    4.4.1 Necessary Properties ...................................................... 63
    4.4.2 Structure of the Code ..................................................... 65
    4.4.3 The Non-Existence Proof .............................................. 69
  4.5 Towards Uniqueness of Code Construction .................................. 71
  4.6 Existence and Construction for $d \geq 2k - 3$ ............................. 73
    4.6.1 Approach ................................................................. 73
    4.6.2 Existence and Construction ............................................ 74
  4.7 A Coding Scheme for any $(n, k, d)$: Exact Regeneration of Systematic Nodes ......................................................... 76
    4.7.1 Scheme Description ...................................................... 76
    4.7.2 Analysis ................................................................. 78
5 An MBR Code for $d = n - 1$ with a Simple Graphical Description 80
  5.1 Explicit Exact-Regenerating MBR Code for $d = n - 1$ ................. 80
    5.1.1 An Example Code ........................................... 81
    5.1.2 Code Construction for the General Set of Parameters with $d = n - 1$ 82
    5.1.3 Size of the Finite Field ..................................... 83
  6 An Approximately-Exact Regenerating MSR Code for $d = k + 1$ 85
    6.1 Explicit Codes for $d = k + 1$ .................................. 85
      6.1.1 Code construction: ......................................... 86
      6.1.2 Reconstruction: ............................................ 86
      6.1.3 Regeneration: ............................................... 87
    6.2 A General Explicit Code for All Parameters ........................ 88
      6.2.1 Construction for the Case $\alpha = 1$ ....................... 88
      6.2.2 Construction for the Case $\alpha = 2$ ....................... 89
      6.2.3 Construction for Any $\alpha$ ............................... 92
      6.2.4 Analysis of the code ...................................... 95
  7 Insights into General Network Coding Problems 97
    7.1 As a Network Coding Problem ................................... 97
    7.2 Necessary and sufficient conditions for a general network .......... 99
    7.3 Bound for Networks with crosslinks ............................ 101
  8 A Flexible Class of Regenerating Codes 104
    8.1 Motivation ..................................................... 104
    8.2 Flexible Setting Description ................................... 105
    8.3 Lower bound on the Repair Bandwidth .......................... 106
      8.3.1 Information Flow Graph .................................... 107
      8.3.2 Cut-set Lower Bound ....................................... 107
      8.3.3 Complete Flexibility ($\beta_{\text{max}} = \alpha$)? .......... 109
    8.4 Achievability .................................................. 109
    8.5 An Explicit Code ............................................... 114
      8.5.1 An Example .................................................. 114
      8.5.2 General Form of the Code .................................. 115
    8.6 Symbol-wise MDS Codes and Error-prone Links ...................... 116
  9 Conclusion .......................................................... 119
A Subspace Properties of Linear Exact Regenerating Codes: Proofs 121
B Proof of Lemma 8.4.1 ............................................... 127
C Proof of Theorem 4.3.2: MISER Reconstruction .......................... 130
CONTENTS

D Proof of Theorem 4.5.1: Uniqueness of the MISER code 134

E Proof of Theorem 4.6.3: Existence and Construction of MSR code for 
   $d = 2k - 3$ 137

Bibliography 141
Chapter 1

Introduction

We all have witnessed the phenomenal increase in the *volume* and more importantly the *value* of the digital data that is stored. No single storage medium is completely reliable, and hence there is a need for a reliable storage system. This, coupled with the tremendous growth in connectivity, makes the distributed architecture a natural choice for storing data reliably across multiple unreliable storage nodes.

![Image](Figure 1.1: Real life distributed storage systems.)

There are many real world examples running on distributed storage. *Internet Archive* ([www.archive.org](http://www.archive.org)) which archives the history of the internet, presently stores over 3 million gigabytes of data on its several servers. The *Large Hadron Collider* is expected to generate around 15 peta bytes of data per year, which will be accessed by researchers all over the world. For this purpose, the CERN has designed a *Grid* which implements cloud computing and cloud storage. *Google* stores huge amount of data for Google maps, Google analytics etc. Peer to peer storage systems like *wuala.com* offer volunteer based collaborative storage systems on the internet.

It is well known that adding redundancy increases the reliability of the system but at the cost of increased storage. A popular option that reduces the storage overhead is to employ erasure coding, for example by calling upon maximum-distance-separable...
1.1 Background

1.1.1 Regenerating Codes

Upon failure of an individual node, a self-sustaining data-storage network must necessarily possess the ability to regenerate (i.e., repair) a failed node by downloading data from the remaining nodes. This amount of data that needs to be downloaded to regenerate a failed node is one of the significant parameters of a distributed storage system. The scenario is depicted in Figure 1.2.

Figure 1.2: Depiction of regeneration of a failed node - unavailability and repair bandwidth.

An obvious means to accomplish this is to permit the replacement node to connect to any $k$ nodes, download the entire data, and extract the data that was stored in the failed node. But downloading the entire $B$ units of data in order to recover the data
stored in a single node that stores only a fraction of the entire data file is wasteful. As the regeneration process consumes the precious network bandwidth and also as it is important to bring back the failed node as soon as possible, this raises the question as to whether there is a better option. Such an option is indeed available and provided by the concept of a regenerating code introduced in the pioneering paper by Dimakis et al. [12].

Conventional RS codes treat each fragment stored in a node as a single symbol belonging to the finite field $\mathbb{F}_q$. It can be shown that when individual nodes are restricted to perform only linear operations over $\mathbb{F}_q$, the total amount of data download needed to repair a failed node, can be no smaller than $B$, the size of the entire file. In contrast, regenerating codes are codes over a vector alphabet and hence treat each fragment as being comprised of $\alpha$ symbols over the finite field $\mathbb{F}_q$. Linear operations over $\mathbb{F}_q$ in this case, permit the transfer of a fraction of the data stored at a particular node. Apart from this new parameter $\alpha$, two other parameters $(d, \beta)$ are associated with regenerating codes. Under the definition of regenerating codes introduced in [12], a failed node is permitted to connect to a fixed number $d$ of the remaining nodes while downloading $\beta \leq \alpha$ symbols from each node. This process is termed as regeneration and the total amount of data downloaded for repair purposes as the repair bandwidth. Typically, with a regenerating code, the average repair bandwidth is small compared to the size of the file $B$. Both reconstruction and regeneration processes are depicted in Fig 1.3.
It will be assumed throughout, that whenever mention is made of an \([n, k, d]\) regenerating code, the code is such that \(k\) and \(d\) are the minimum values under which reconstruction and regeneration can always be guaranteed. This restricts the range of \(d\) to

\[
k \leq d \leq n - 1,
\]

for, if the regeneration parameter \(d\) were less than the reconstruction parameter \(k\), this would imply that one could in fact, reconstruct data by connecting to \(d\) nodes thereby contradicting the minimality of \(k\). Also note that for reconstruction to be satisfied, we need

\[
\alpha \geq B/k .
\]

By feasible values for the set of parameters, we will mean, any set of positive integer values of the parameters, satisfying equations (1.1) and (1.2).

Finally, while a regenerating code over \(\mathbb{F}_q\) is associated with the collection of parameters

\[
\{n, k, d, \alpha, \beta, B\},
\]

it will be found more convenient to regard parameters \(\{n, k, d\}\) as primary and \(\{\alpha, \beta, B\}\) as secondary and thus we will make frequent references in the sequel, to a code with these six parameters as an \((n, k, d)\) regenerating code having parameter set \((\alpha, \beta, B)\).

1.1.2 The Storage-Repair Bandwidth Tradeoff

In [12], authors represent a distributed storage system as a graph termed as information flow graph. In such a graph, each storage node is modeled using two nodes - an in node and an out node and a link of capacity \(\alpha\) connecting the two. This captures the constraint that each node can store only \(\alpha\) symbols. On failure of a storage node, say node \(\ell\), it is replaced by a new node by connecting nodes \(\text{out}(j), j \in \{i_1, \ldots, i_d\}, j \neq \ell\),
to in(ℓ), with links of capacities β. Thus the network evolves through an infinite chain of failures and regenerations. A depiction of information flow graph from [13] is shown in Fig 1.4.

A major result in the field of regenerating codes is the proof in [13] that uses the cut-set bound of network coding to establish that the parameters of a regenerating code must necessarily satisfy:

$$B \leq \sum_{i=0}^{k-1} \min\{\alpha, (d-i)\beta\}.$$  \tag{1.3}

The reconstruction requirement is formulated as a multicast network coding problem, with the network having an infinite number of nodes. The cut-set analysis of this network leads to the relation between the parameters of a regenerating code as given in equation (1.3). The kind of cuts in the information flow graph used in [13] to establish the cut-set bound is shown in Fig 1.5.

It is desirable to minimize both α as well as β since, minimizing α results in a minimum storage solution, while minimizing β (for fixed d) results in a storage solution that minimizes repair bandwidth. As can be deduced from (1.3), it is not possible to minimize both α and β simultaneously and thus there is a tradeoff between choices of the parameters α and β. The tradeoff for the parameters $B = 27000$, $k = 10$, $d = 18$ is given in Fig 1.6.

We define an optimal $[n, k, d]$ regenerating code as a code with parameters $(\alpha, \beta, B)$ satisfying the twin requirements that

(i) the parameter set $(\alpha, \beta, B)$ achieve the cut-set bound with equality and

(ii) decreasing either α or β or increasing B will result in a new parameter set that violates the cut set bound.
An MSR code is then defined as an \([n, k, d]\) regenerating code whose parameters \((\alpha, \beta, B)\) satisfy (1.4) and similarly, an MBR code as one with parameters \((\alpha, \beta, B)\) satisfying (1.5). Clearly, both MSR and MBR codes are optimal regenerating codes.

For \(\alpha = 1\), we get \(B = k\). In this case, any \([n, k]\)-MDS code will trivially achieve the cut-set bound. Hence, we will consider \(\alpha > 1\) throughout.

### 1.1.3 MSR and MBR points on the tradeoff

The two extreme points on the storage-repair bandwidth tradeoff are termed the minimum storage regeneration (MSR) and minimum bandwidth regeneration (MBR) points respectively.

**The MSR Point** The parameters \(\alpha\) and \(\beta\) for the MSR point on the tradeoff can be obtained by first minimizing \(\alpha\) and then minimizing \(\beta\) to obtain

\[
\alpha = \frac{B}{k},
\]

\[
\beta = \frac{B}{k(d - k + 1)}.
\]  

\[1.4\]
Note that at this point, each node stores the minimum amount that is necessary to satisfy the reconstruction property.

**The MBR Point**  First minimizing $\beta$ and then $\alpha$, leads to the MBR point which thus corresponds to

$$\beta = \frac{2B}{k(2d - k + 1)},$$

$$\alpha = \frac{2dB}{k(2d - k + 1)}. \quad (1.5)$$

Note that the MBR point corresponds to the minimum possible repair bandwidth as a replacement node downloads not more than $\alpha$ symbols which is the storage capacity of each node.

## 1.2 Additional Terminology

### 1.2.1 Striping of Data

The nature of the cut-set bound permits a divide-and-conquer approach to be used in the application of optimal regenerating codes to large file sizes, thereby simplifying system implementation. This is explained below.

Given an optimal $[n, k, d]$ regenerating code with parameter set $(\alpha, \beta, B)$, a second regenerating code with parameter set $(\alpha' = \delta\alpha, \beta' = \delta\beta, B' = \delta B)$ for any positive integer $\delta$ can be constructed, by dividing the $\delta B$ message symbols into $\delta$ groups of $B$ symbols each, and applying the $(\alpha, \beta, B)$ code to each group independently. Secondly, a common feature of both MSR and MBR regenerating codes is that in either case, their parameter set $(\alpha, \beta, B)$ is such that both $\alpha$ and $B$ are multiples of $\beta$ and further that $\frac{\alpha}{\beta}, \frac{B}{\beta}$ are functions only of $n, k, d$. It follows that if one can construct an $[n, k, d]$ MSR or MBR code with $\beta = 1$, then one can construct an $[n, k, d]$ MSR or MBR code for any larger value of $\beta$.

Also, from a practical standpoint, this divide-and-conquer approach can be employed to achieve a host of desirable properties, which are briefly discussed here.

**Low Complexity:** A code with smaller $\beta$ will involve manipulating a smaller number of message symbols and hence will in general, be of lesser complexity. Reconstruction and regeneration are performed separately on the smaller chunks thereby greatly reducing the additional processing and storage requirement.

**Variable $\alpha$:** When storage capacities of the nodes are different, each stripe can be encoded separately with a different $n$ (and possibly a different $k$ and $d$ as well). A stripe will see only those storage nodes in which the data pertaining to that stripe is stored, thus seamlessly handling the case of variable $\alpha$. 
1.2 Additional Terminology

**Load Balancing:** Striping of data can be employed such that an end-user can connect to different of \( k \) nodes for each stripes there by not burdening a particular set of nodes. Similarly a replacement node for a failed node can connect to different set of \( d \) nodes to balance the load.

**Partial Recovery of data:** Since each stripe can be decoded independently, when only a small chunk of intermediate data is required, only the stripes containing the desired data can be decoded, instead of decoding the entire data.

For these reasons, we design codes for the atomic case \( \beta = 1 \) and thus, we will assume that \( \beta = 1 \) throughout unless otherwise mentioned. We document below the values of \( \alpha \) and \( B \) when \( \beta = 1 \) of MSR and MBR codes respectively:

\[
\alpha = d - k + 1 ,
\]
\[
B = k(d - k + 1) ,
\]

for MSR codes and

\[
\alpha = d ,
\]
\[
B = kd - \binom{k}{2} ,
\]

in the case of MBR codes.

### 1.2.2 Exact versus functional regeneration

In the context of a regenerating code, by functional regeneration of a failed node \( \nu \), we will mean, replacement of the failed node by a new node \( \nu' \) in such a way that following replacement, the resulting network of \( n \) nodes continues to possess the reconstruction and regeneration properties. In contrast, by exact regeneration, we mean replacement of a failed node \( \nu \) by a replacement node \( \nu' \) that stores exactly the same data as was stored in node \( \nu \). We will use the term *exact-regenerating code* to denote a regenerating code that has the capability of exactly regenerating each instance of a failed node. Clearly where it is possible, an exact-regeneration code is to be preferred over a functional-regeneration code since, under functional regeneration, there is need for the network to inform all nodes in the network of the replacement, whereas this is clearly unnecessary under exact regeneration. This makes the storage system practical and easy to maintain.

### 1.2.3 Systematic regenerating codes

A systematic regenerating code can be defined as a regenerating code designed in such a way that the \( B \) message symbols are explicitly present amongst the \( k\alpha \) code symbols stored in a select set of \( k \) nodes, termed as the systematic nodes. If the DC preferably
connects to this set of $k$ nodes, no additional processing is necessary for decoding. Clearly, in the case of systematic regenerating codes, exact regeneration of (the systematic portion of the data stored in) the systematic nodes is mandated.

### 1.2.4 Linear regenerating codes

A linear regenerating code is defined as a regenerating code in which

1. the code symbols stored in each node are linear combinations over $\mathbb{F}_q$ of the $B$ message symbols $\{u_i\}$,
2. only linear operations (over $\mathbb{F}_q$) are permitted.

### 1.3 Overview of the Literature

The concept of regenerating codes was introduced in [12], where it was shown that permitting the storage nodes to store more than $B/k$ units of data helps in reducing the repair bandwidth. Several distributed systems were analyzed, and estimates of the mean node availability in such systems obtained. Using these values, it was shown through simulation, that regenerating codes can reduce repair bandwidth compared to other designs, while simplifying system architecture.

The problem of minimizing repair bandwidth for functional repair of a failed storage node is considered in [12, 13]. In [13], [16], it has been shown that the storage-repair bandwidth tradeoff is achievable. However, the coding schemes suggested are not explicit and require large field size. Also contained in the journal version of [14], is a handcrafted functional regenerating code for the MSR point with $(n = 4, k = 2, d = 3)$.

A principal concern in the practical implementation of distributed storage codes is computational complexity and a practical study of this aspect is carried out in [15] in the context of random linear regenerating codes that achieve functional repair.

The problem of exact regeneration was first considered independently in [17] and [6]. In [17], it is shown that the MSR point is achievable under exact regeneration when $(k = 2, d = n - 1)$. The coding scheme proposed is based on the concept of interference alignment developed in the context of wireless communication. However, the construction is not explicit and has a large field size requirement. In [18], the authors carry out a computer search to find exact regenerating codes at the MSR point, resulting in identification of codes with parameters $(n = 5, k = 3, d = 4)$.

The first, explicit construction of regenerating codes for a general set of parameters was provided for the MBR point in [6] with $d = n - 1$ and arbitrary $k$. These codes have low regeneration complexity as no computation is involved during the regeneration of a failed node. The field size required is on the order of $n^2$. Also contained in [6]
1.4 Summary of the Results

(see also [2]), is the construction of an explicit MSR code for \( d = k + 1 \), that performs approximately-exact regeneration of all failed nodes.

MSR codes performing a hybrid of exact and functional regeneration are provided in [22], for the parameters \( d = k + 1 \) and \( n \geq 2k \). The codes given even here are non-explicit, and have high complexity and large field-size requirement.

A code structure that guarantees exact repair of just the systematic nodes is provided in [5], for the MSR point with parameters \( d = (n - 1) \geq 2k - 1 \). This code makes use of interference alignment, and is termed as the ‘MISER’ code in [2]. Subsequently, it was shown in [19] that for this set of parameters, the code introduced in [5] for exact repair of only the systematic nodes can also be used to repair the non-systematic (parity) node failures exactly provided repair construction schemes are appropriately designed. Such an explicit repair scheme is indeed designed and presented in [19]. Also contained in [19], is an exact-regenerating MSR code for parameter set \((n = 5, k = 3, d = 4)\).

A proof of non-achievability of the cut-set bound on exact regeneration at the MSR point, for the parameters \( d < 2k - 3 \) when \( \beta = 1 \), is provided in [2].

A flexible setup for regenerating codes is described in [3], where a data-collector (or a replacement node) can perform reconstruction (or regeneration) irrespective of the number of nodes to which it connects, provided the total data downloaded exceeds a certain threshold.

In [4], the authors establish that the interior points on the tradeoff (i.e., other than MSR and the MBR points) are not achievable under exact regeneration, except for a region of length at most \( \beta \) at the immediate vicinity of the MSR point. The MSR point was shown to be achievable in the limiting case of \( B \) approaching infinity (i.e., \( \beta \) approaching infinity) in [23, 24].

1.4 Summary of the Results

The work mainly deals with explicit construction of regenerating codes for distributed storage and the achievability or otherwise of the storage-repair bandwidth tradeoff under the practically relevant exact regeneration scenario. Following is a summary of the results presented in this report:

(i) MBR point

- Explicit exact-regenerating MBR codes for all feasible values of the parameters \((n, k, d)\) via a Product-Matrix construction
- Achievability of the MBR point under exact regeneration for all parameters
- Explicit exact-regenerating MBR codes for all parameters \((n, k, d = n - 1)\), which has a simple graphical description, and when specialized to the parameter
1.4 Summary of the Results

set \([n, k = n - 2, d = n - 1]\), can operate over binary field using solely XOR operations

(ii) MSR point

- Explicit exact-regenerating MSR codes for \((n, k, d \geq 2k - 2)\) via a Product-Matrix construction
- MISER code: An explicit code structure that guarantees exact repair of the systematic nodes for parameters \(d = (n - 1) \geq 2k - 1\) using the concept of interference alignment
- Proof that interference alignment is necessary for exact regeneration at the MSR point
- Proof of non-achievability of the MSR point under exact regeneration for the parameters satisfying \(d < 2k - 3\) for the atomic case of \(\beta = 1\)
- Explicit approximately exact-regenerating MSR codes for \((n, k, d = k + 1)\); this parameter set falls in the non-achievable region under exact regeneration
- An achievable scheme minimizing storage space, though not explicit, for exact regeneration of systematic nodes, and which is optimal for \(d \geq 2k - 1\)

(iii) Interior points

- Subspace properties to be satisfied by any exact regenerating code
- Proof of non-achievability of the interior points of the tradeoff under exact regeneration except for a region of length at most \(\beta\) at the immediate vicinity of the MSR point
- An achievable curve for the interior points using storage space sharing

(iv) An explicit code for all values of the parameters uniformly reducing repair bandwidth to approximately half the file size

(v) An explicit ideal regenerating code which achieves minimum storage and minimum repair bandwidth simultaneously via a Product-Matrix construction

(vi) A Flexible class of regenerating codes

- A new class of regenerating codes which removes the restrictions in the original setting and allows a DC or a replacement node to freely connect to any number of nodes and download any amount of data from each of the nodes as long as a set of feasibility conditions are met
- An explicit construction of flexible regenerating codes for a certain parameter regime
1.5 Organization of the Report

- Proof that any code satisfying flexible reconstruction is symbol wise MDS, provided field size is large enough; a useful result for error prone networks

(vii) Insights into coding for general non-multicast networks

- Useful heuristics for code design
- Tightens cut-set bound for certain class of networks

1.5 Organization of the Report

The rest of the report is organised as follows:

In Chapter 2, a product-matrix framework is introduced using which explicit exact-regenerating MBR codes for all feasible values of the parameters \((n, k, d)\) and MSR codes for \((n, k, d \geq 2k - 2)\) are constructed. An ideal regenerating code is introduced which achieves both minimum storage and minimum repair bandwidth simultaneously, by relaxing certain conditions in the original setup. An explicit construction is also provided via product-matrix framework.

Chapter 3 deals with the achievability of the storage-repair bandwidth tradeoff under exact regeneration. Using a subspace based approach, necessary properties to be satisfied by any regenerating is established. Then it is shown that the interior points of the tradeoff are not achievable under exact regeneration, except for a region of width at most \(\beta\) at the immediate vicinity of the MSR point.

In Chapter 4, the necessity of interference alignment for any exact regenerating MSR code is established. An explicit code structure termed MISER code, that guarantees exact repair of the systematic nodes at the MSR point for parameters \(d = (n - 1) \geq 2k - 1\), based on the concept of interference alignment, is presented. And subsequently, non-achievability of the MSR point under exact regeneration for the parameters satisfying \(d < 2k - 3\) for the atomic case of \(\beta = 1\) is shown. Also provided is a coding scheme minimizing the storage space, though not explicit, for exact regeneration of systematic nodes. This scheme is optimal for \(d \geq 2k - 1\).

Chapter 5 presents an MBR code for the parameters \((n, k, d = n - 1)\) which has a simple graphical description, and when specialized to the parameter set \((n, k = n - 2, d = n - 1)\), can operate over binary field using solely XOR operations.

Chapter 6 presents an explicit approximately exact-regenerating MSR code for the parameter set \((n, k, d = k + 1)\). This code is used as a basis for an explicit code for all values of the parameters, which uniformly reduces the repair bandwidth to approximately half the file size.

In Chapter 7, the problem of obtaining exact-regenerating MSR codes is casted as a non-multicast network coding problem. The insights obtained in constructing codes for
distributed storage is translated to provide a few insights into coding for general non-multicast networks. This provides useful heuristics for code design as well as helps in tightening cut-set bound for certain class of networks.

Chapter 8 introduces a new class of regenerating codes termed as *flexible regenerating codes*. This class of regenerating codes allow a DC or a replacement node to freely connect to arbitrary number of nodes, downloading arbitrary amounts of data from each, as long as a set of feasibility conditions are met. An explicit code construction is also provided which is optimal in certain parameter regimes. It is established that any code satisfying flexible reconstruction is a symbol wise MDS code, provided the field size is large enough. This result is useful in the case of error prone links.

Finally, a conclusion is provided in Chapter 9.
Chapter 2

Explicit Codes via a Product-Matrix Framework

In this chapter, we present explicit-exact-regenerating MBR and MSR codes via a product-matrix construction. All the previously known explicit and general constructions for exact regenerating codes at the MSR point have been found only for the case \( d = n - 1 \geq 2k - 1 \). Similarly at the MBR point, the only explicit code to previously have been constructed is for the case \( d = n - 1 \). Thus, all previously known code constructions limit the total number of nodes in the system to \( d + 1 \). This is restrictive since in this case, the system can handle only a single node failure at a time, and also the system does not permit additional storage nodes to be brought into the system.

The MSR code for all values of the parameters \([n, k, 2k - 2 \leq d \leq n - 1]\) is presented in Section 2.2 and the MBR code for all feasible values of \([n, k, d]\) in Section 2.3. The explicit constructions presented in this chapter prove that the MBR point on the storage-repair bandwidth tradeoff can be achieved under exact regeneration for all feasible values of the parameters and the MSR point for all the parameters satisfying \((2k - 2 \leq d \leq n - 1)\).

In the traditional regenerating codes, either the storage space or the repair bandwidth is compromised and both cannot be minimized simultaneously. Using the product-matrix construction, we present an ideal regenerating code in Section 2.5 which achieves both minimum storage and minimum repair bandwidth simultaneously.

Let \( C \) be the code matrix associated with a regenerating code. Thus, \( \{c_{ij}\} \), denotes the \( j \)th symbol, \( 1 \leq j \leq \alpha \), stored in the \( i \)th node \( 1 \leq i \leq n \). In all the constructions provided in this chapter, the code matrix is the product of an encoder matrix and a message matrix. The encoder matrix serves to disperse the information contained in the message matrix in such a way so as to enable data reconstruction and failed-node regeneration, from the information stored in a small subset of the nodes. The message matrix contains the message symbols in redundant form, invoking a form of block-symmetry that also aids in the reconstruction and regeneration process. This common structure of the code matrices leads to common architectures for both reconstruction and regeneration, as explained in
greater detail in Section 2.1. It also endows the codes with implementation advantages that are discussed in Section 2.4.

2.1 Framework Description

2.1.1 Product-Matrix

Let \( \mathbf{u} = [u_1, u_2, \ldots, u_B], \) \( u_i \in \mathbb{F}_q, \) denote the vector of \( B \) message symbols. Each codeword in the distributed code can be represented by an \( (n \times \alpha) \) matrix \( C(\mathbf{u}) \) whose \( i \)th row \( c_i \) contains the \( \alpha \) symbols stored by the \( i \)th node. Under the product-matrix framework of the constructions presented here, each code matrix is the product

\[
C(\mathbf{u}) = \Psi M(\mathbf{u})
\]

of a \( (n \times m) \) encoding matrix \( \Psi \) and a \( (m \times \alpha) \) message matrix \( M(\mathbf{u}) \). The entries of the matrix \( \Psi \) are fixed apriori and independent of the message symbols. We will refer to the \( i \)th row \( \psi_i \) of \( \Psi \) as the encoding vector of node \( i \) as it is this vector that is used to encode the data into the form in which it is stored within the \( i \)th node. The entries of the message matrix \( M(\mathbf{u}) \) are a redundant collection of linear combinations of the message symbols \( \{u_i\} \). The collection of code matrices gives rise to the regenerating code

\[
\mathcal{C} = \{C(\mathbf{u}) \mid \mathbf{u} \in \mathbb{F}_q^B\}
\]

of size \( q^B \).

2.1.2 Dispersion and Block-Symmetry

The encoding matrix serves to disperse information contained in the message matrix across the \( n \) nodes in such a way as to facilitate reconstruction as well as regeneration. This is similar to the manner in which the generator matrix of an MDS code disperses information across code symbols. In the constructions presented here, the message matrix \( M \) possesses the following symmetry property: it can be expressed in block-matrix form in such a way that if \( A \) is one of the blocks, so is \( A^t \), i.e., the collection of block matrices is closed under transposition. From this it is clear that redundancy is built into the message matrix. We will refer to this property of the message matrix as the block-symmetry property. The dispersion properties of the encoding matrix and block-symmetry of the message matrix are key to establishing the reconstruction and regeneration properties of the codes constructed here.
2.1 Framework Description

2.1.3 Data Encoding for the Storage Nodes

To encode data prior to storage in the $n$ nodes, the $B$ message symbols are first reassembled into the form of the $(m \times \alpha)$ message matrix $M$. The $\alpha$ symbols stored in node $i$ are then given by $\psi^t_i M$ (Fig. 2.1).

2.1.4 Reconstruction of Data by the Data Collector

To reconstruct the data, the data collector connects to an arbitrary subset \{\(i_1, \ldots, i_k\)\} of $k$ storage nodes (Fig. 2.2). The $j$th node in this set passes on the message vector $\psi^t_{i_j} M$ to the data collector (Fig. 2.3). The data collector aggregates the collection of $k$ message vectors to obtain the product matrix

$$\Psi_{DC} M$$

where $\Psi_{DC}$ is the submatrix of $\Psi$ consisting of the $k$ rows $\{i_1, \ldots, i_k\}$. It then uses the dispersion properties of the matrix $\Psi$ as well as the block-symmetry within message matrix $M$ to recover the data. The precise procedure for recovering $M$ is a function of the particular construction.
2.1 Framework Description

2.1.5 Regeneration of a Failed Node

As noted above, each node in the network is associated to a distinct \((1 \times m)\) encoding vector \(\psi_i\). In the constructions, we will at times, need to call upon a related vector \(\mu_i\) of smaller size, that contains a subset of the components of \(\psi_i\). We will refer to \(\mu_i\) as the *digest* of \(\psi_i\). To regenerate a failed node \(f\), the node replacing the failed node connects to an arbitrary subset \(\{h_1, \ldots, h_d\}\) of \(d\) storage nodes which we will refer to as the *helper nodes*. The \(j\)th helper node \(h_j\) in this set, passes on the message vector \(\psi_{h_j}^t M\mu_f\) to the replacement node (Fig. 2.4). The replacement node aggregates the message vectors to obtain the product matrix

\[
\Psi_{\text{rep}} M\mu_f
\]

where \(\Psi_{\text{rep}}\) (for repair) is the submatrix of \(\Psi\) consisting of the \(d\) rows \(\{h_1, \ldots, h_d\}\). It then uses the dispersion properties of the matrix \(\Psi\) to recover the vector \(M\mu_f\) a process that we will refer to as partial-decoding. The output of the partial-decoder is fed to a re-encoder that uses the symmetry within message matrix \(M\) to transform the data into
Remark 2.1.1. An important feature of the product-matrix construction presented here, is that each of the nodes \( i_j \) participating in the regeneration of node \( f \), need only have knowledge of the encoding vector of the failed node \( f \) and not the identity of the other nodes participating in the regeneration. This significantly simplifies system operation.

2.2 The Product-Matrix MSR Code Construction

In this section, we identify the specific make-up of encoding matrix \( \Psi \) and message matrix \( M \) that results in an \([n,k,d]\) MSR code \( C \) with \( \beta = 1 \). The construction applies to all \([n,k,d]\) with \( d \geq 2k-2 \). Since \( C \) is required to be an MSR code with \( \beta = 1 \), it must possess the reconstruction and regeneration properties required of a regenerating code and in addition, have parameters \( \{\alpha,B\} \) that satisfy equations (1.6) and (1.7). We begin by constructing an MSR code in the product-matrix format for \( d = 2k-2 \) and will show in Section 2.2.3 how this can be very naturally extended to yield codes with \( d > 2k-2 \).

At the MSR point with \( d = 2k-2 \) we have

\[ \alpha = d - k + 1 = k - 1 \] (2.3)

and hence

\[ d = 2\alpha. \] (2.4)

Also,

\[ B = k\alpha = \alpha(\alpha + 1). \] (2.5)

Let the collection \( \{u_i\}_{i=1}^{B} \) of message symbols be partitioned into two subsets \( \mathcal{M}_1, \mathcal{M}_2 \), each of size \( \binom{\alpha+1}{2} \). Let \( S_1, S_2 \) be a pair of \( \alpha \times \alpha \) symmetric matrices constructed so that
the \( \binom{\alpha+1}{2} \) entries in the upper-triangular half of each of the two matrices are filled up by the \( \binom{\alpha+1}{2} \) distinct message symbols belonging to sets \( \mathcal{M}_1, \mathcal{M}_2 \) respectively. The \( \binom{\alpha}{2} \) entries in the strictly lower-triangular portion of the two matrices \( S_1, S_2 \) be chosen so as to make the matrices \( S_1, S_2 \) symmetric.

The message matrix \( M \) is then defined as the \( d \times \alpha \) matrix given by

\[
M = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}.
\] (2.6)

Thus as was the case with the MBR construction of Section 2.3, we have set design parameter \( m = d \).

Next, define the encoding matrix \( \Psi \) to be the \( n \times d \) matrix given by

\[
\Psi = \begin{bmatrix} \Phi & \Lambda \Phi \end{bmatrix},
\] (2.7)

where \( \Phi \) is an \( n \times \alpha \) matrix and \( \Lambda \) is an \( n \times n \) diagonal matrix. Let the elements of \( \Psi \) be chosen such that the following conditions are satisfied:

(i) any \( d \) rows of \( \Psi \) are linearly independent,
(ii) any \( \alpha \) rows of \( \Phi \) are linearly independent and
(iii) the \( n \) diagonal elements of \( \Lambda \) are distinct.

The above requirements can be met, for example, by choosing \( \Psi \) to be a Vandermonde matrix with elements chosen in particular to satisfy condition 3. Then under our code-construction framework, the \( i \)th row of the \( (n \times \alpha) \) product matrix \( C = \Psi M \), contains the \( \alpha \) code symbols stored by the \( i \)th node.

The two theorems below establish that the code \( C \) is an \([n,k,d]\) MSR code by establishing respectively, the reconstruction and regeneration properties of the code. We begin in the case of the present construction, with the simpler proof of the regeneration property.

**Theorem 2.2.1 (MSR Exact Regeneration)** *In the code \( C \) presented, exact regeneration of any failed node can be achieved by connecting to any \( d = 2k - 2 \) of the remaining \((n - 1)\) nodes.*

**Proof** Let \( \begin{bmatrix} \phi_f^t & \lambda_f \phi_f^t \end{bmatrix} \) be the row of \( \Psi \) corresponding to the failed node. Thus the \( \alpha \) symbols stored in the failed node were

\[
\begin{bmatrix} \phi_f^t & \lambda_f \phi_f^t \end{bmatrix} M = \phi_f^t S_1 + \lambda_f \phi_f^t S_2.
\] (2.8)

The replacement for the failed node \( f \) connects to an arbitrary set \( \{h_i \mid i = 1, 2, \ldots, d\} \) of \( d \) helper nodes (Fig. 2.4). Upon being contacted by the replacement node, the helper
node $h_j$ first computes the digest $\mu_f = \phi_f$ and then computes the inner product $< \psi_{h_j}^t, M, \phi_f >$. It then passes on the pair $\left( \psi_{h_j}, < \psi_{h_j}^t, M, \phi_f > \right)$ to the replacement node. The replacement node computes $\Psi_{\text{rep}}M\phi_f$ by aggregating the inputs from the $d$ helper nodes where

$$
\Psi_{\text{rep}} = \begin{bmatrix}
\psi_{h_1}^t \\
\psi_{h_2}^t \\
\vdots \\
\psi_{h_d}^t
\end{bmatrix}.
$$

By construction, the $d \times d$ matrix $\Psi_{\text{rep}}$ is invertible. Thus the replacement node now has access to

$$
M\phi_f = \begin{bmatrix}
S_1 \phi_f \\
S_2 \phi_f
\end{bmatrix}.
$$

As $S_1$ and $S_2$ are symmetric matrices, the replacement node thus has acquired through transposition, both $\phi_f^t S_1$ and $\phi_f^t S_2$ using which it can obtain,

$$
\phi_f^t S_1 + \lambda_f \phi_f^t S_2
$$

which is precisely the data previously stored in the failed node. \hfill \blacksquare

**Theorem 2.2.2 (MSR Reconstruction)** *In the code $C$ presented, all the $B$ message symbols can be recovered by connecting to any $k$ nodes i.e, the message symbols can be recovered through linear operations on the entries of any $k$ rows of the code matrix $C$.*

**Proof** Let

$$
\Psi_{\text{dc}} = \begin{bmatrix}
\Phi_{\text{dc}} & \Lambda_{\text{dc}} \Phi_{\text{dc}}
\end{bmatrix}
$$

be the $k \times d$ submatrix of $\Psi$, containing the $k$ rows of $\Psi$ which correspond to the $k$ data recovery nodes to which the DC connects. Hence the DC obtains the symbols

$$
\Psi_{\text{dc}} M = \begin{bmatrix}
\Phi_{\text{dc}} & \Lambda_{\text{dc}} \Phi_{\text{dc}}
\end{bmatrix} \begin{bmatrix}
S_1 \\
S_2
\end{bmatrix} = \begin{bmatrix}
\Phi_{\text{dc}} S_1 + \Lambda_{\text{dc}} \Phi_{\text{dc}} S_2
\end{bmatrix}.
$$

The DC can post-multiply this with $\Phi_{\text{dc}}^T$ to obtain

$$
[\Phi_{\text{dc}} S_1 + \Lambda_{\text{dc}} \Phi_{\text{dc}} S_2] \Phi_{\text{dc}}^T = \Phi_{\text{dc}} S_1 \Phi_{\text{dc}}^T + \Lambda_{\text{dc}} \Phi_{\text{dc}} S_2 \Phi_{\text{dc}}^T.
$$
Next, let the matrices $P$ and $Q$ be defined by

\[ P = \Phi_{\text{DC}} S_1 \Phi_{\text{DC}}^T, \quad \text{(2.13)} \]
\[ Q = \Phi_{\text{DC}} S_2 \Phi_{\text{DC}}^T. \quad \text{(2.14)} \]

As $S_1$ and $S_2$ are symmetric, the same is true of the matrices $P$ and $Q$. In terms of $P$ and $Q$, the DC has access to the symbols of the matrix

\[ P + \Lambda_{\text{DC}} Q. \quad \text{(2.15)} \]

The $(i, j)^{\text{th}}$, $1 \leq i, j \leq k$, element of this matrix is

\[ P_{ij} + \lambda_i Q_{ij}, \quad \text{(2.16)} \]

while the $(j, i)^{\text{th}}$ element is given by

\[ P_{ji} + \lambda_j Q_{ji} = P_{ij} + \lambda_j Q_{ij} \quad \text{(2.17)} \]

where equation (2.17) follows from the symmetry of $P$ and $Q$. By construction, all the $\lambda_i$ are distinct and hence using equations (2.16) and (2.17), the DC can solve for the values of $P_{ij}$, $Q_{ij}$ for all $i \neq j$.

Consider first the matrix $P$. Let $\Phi_{\text{DC}}$ be given by,

\[ \Phi_{\text{DC}} = \begin{bmatrix} \phi^t_1 & \cdots & \phi^t_\alpha + 1 \end{bmatrix}. \quad \text{(2.18)} \]

All the non-diagonal elements of $P$ are known. The elements in the $i$th row (excluding the diagonal element) are given by

\[ \phi^t_i S_1 \begin{bmatrix} \phi_1 & \cdots & \phi_{i-1} & \phi_{i+1} & \cdots & \phi_{\alpha + 1} \end{bmatrix}. \quad \text{(2.19)} \]

However, the matrix to the right is non-singular by construction and hence the DC can obtain

\[ \phi^t_i S_1 \quad 1 \leq i \leq \alpha + 1. \quad \text{(2.20)} \]

Thus selecting the first $\alpha$ of these, the DC has access to

\[ \begin{bmatrix} \phi^t_1 \\ \vdots \\ \phi^t_\alpha \end{bmatrix} S_1. \quad \text{(2.21)} \]
2.2 The Product-Matrix MSR Code Construction

The matrix on the left is also non-singular by construction and hence the DC can recover \( S_1 \). Similarly, using the values of the non-diagonal elements of \( Q \), the DC can recover \( S_2 \).

### 2.2.1 An Example for the Product-Matrix MSR code

Let \( n = 6 \), \( k = 3 \), \( d = 4 \). Then \( \alpha = d - k + 1 = 2 \) and \( B = k\alpha = 6 \). Let us choose \( q = 13 \), so we are operating over \( \mathbb{F}_{13} \). The matrices \( S_1, S_2 \) are filled up by the 6 message symbols \( \{u_i\}_{i=1}^6 \) as shown below:

\[
S_1 = \begin{bmatrix} u_1 & u_2 \\ u_2 & u_3 \end{bmatrix}, \quad S_2 = \begin{bmatrix} u_4 & u_5 \\ u_5 & u_6 \end{bmatrix}.
\] (2.22)

Then the message matrix \( M \) is given by

\[
M = \begin{bmatrix} u_1 & u_2 \\ u_2 & u_3 \\ u_4 & u_5 \\ u_5 & u_6 \end{bmatrix}.
\] (2.23)

We choose \( \Psi \) to be the \((6 \times 4)\) Vandermonde matrix over \( \mathbb{F}_{13} \) given by

\[
\Psi = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 1 \\ 1 & 3 & 9 & 1 \\ 1 & 4 & 3 & 12 \\ 1 & 5 & 12 & 8 \\ 1 & 6 & 10 & 8 \end{bmatrix}.
\] (2.24)

Hence the \((6 \times 2)\) matrix \( \Phi \) and the \((6 \times 6)\) diagonal matrix \( \Lambda \) are

\[
\Phi = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \\ 1 & 6 \end{bmatrix}, \quad \Lambda = \begin{bmatrix} 1 & 4 \\ & 9 \\ & & 3 \\ & & & 12 \\ & & & & 10 \end{bmatrix}.
\] (2.25)

Fig. 2.6 shows at the top, the \((6 \times 2)\) code matrix \( C = \Psi M \) with entries expressed as functions of the message symbols \( \{u_i\} \). The rest of the figure explain how regeneration of failed node 1 takes place. To regenerate node node 1, the helper nodes (nodes 2, 4, 5, 6 in the example), pass on their respective inner products \( \langle \psi_\ell, 1 \rangle \) where \( 1 = [1 \ 1]^t \).
2.2 The Product-Matrix MSR Code Construction

Figure 2.6: An example for the MSR code construction: On failure of node 1, the replacement node downloads one symbol each from nodes 2, 4, 5, and 6, using which node 1 is exactly regenerated.

for \( \ell = 2, 4, 5, 6 \). The partial decoder in the replacement node multiplies the symbols it receives with \( \Psi_{\text{rep}}^{-1} \) where

\[
\Psi_{\text{rep}} = \begin{bmatrix}
1 & 2 & 4 & 8 \\
1 & 4 & 3 & 12 \\
1 & 5 & 12 & 8 \\
1 & 6 & 10 & 8
\end{bmatrix}
\]

and decodes \( S_{1,1} \psi_{1-1} \) and \( S_{2,1} \psi_{1-1} \)

\[
S_{1,1} \psi_{1-1} = \begin{bmatrix} u_1 + u_2 \\ u_2 + u_3 \end{bmatrix}, \quad S_{2,1} \psi_{1-1} = \begin{bmatrix} u_4 + u_5 \\ u_5 + u_6 \end{bmatrix}.
\]

Then the re-encoder in the replacement node processes \( S_{1,1} \psi_{1-1} \) and \( S_{2,1} \psi_{1-1} \) to obtain the data stored in the failed node as explained in the proof of Theorem 2.2.1 above.

2.2.2 Systematic Version of the Code

Every exact-regenerating code has a systematic version and further, that the code could be made systematic through a process of message-symbol remapping. In the following, we make this more explicit.

Let \( \Psi_k \) be a \( k \times d \) submatrix of \( \Psi \) containing the \( k \) rows of \( \Psi \) corresponding to the \( k \) nodes which are chosen to be made systematic. The set of \( B \) symbols stored in these \( k \) nodes are given by the elements of the \( k \times \alpha \) matrix \( \Psi_k M \). Let \( \mathcal{B} \) be a \( k \times \alpha \) matrix
containing the $B = k\alpha$ source symbols. We use
\[
\Psi_k M = B
\] (2.28)
to solve for the entries of $M$ in terms of the source symbols in $B$. This is precisely the reconstruction process that takes place when a data collector connects to the $k$ chosen nodes. Thus, the value of the entries in $M$ can be obtained by following the procedure outlined in Theorem 2.2.2. Next use this $M$ to obtain the code $C = \Psi M$. Clearly, in this representation, the $k$ chosen nodes store the source symbols $B$ in uncoded form.

### 2.2.3 Explicit MSR Product-Matrix Codes for $d \geq 2k - 2$

In this section, we show how an MSR code for $d = 2k - 2$ can be used to obtain MSR codes for all $d \geq 2k - 2$. Our starting point is the following theorem.

**Theorem 2.2.3** An explicit $[n', k', d'] = [n + 1, k + 1, d + 1]$ regenerating code $C'$ that achieves the cut-set bound at the MSR point can be used to construct an explicit $[n, k, d]$ regenerating code $C$ that also achieves the cut-set bound at the MSR point. Furthermore if $d' = ak' + b$ in code $C'$, $d = ak + b + (a - 1)$ in code $C$. If $C'$ is linear, so is $C$.

**Proof** If both codes operate at the MSR point, then the number of message symbols $B', B$ in the two cases must satisfy
\[
B' = k'(d' - k' + 1) \quad \text{and} \quad B = k(d - k + 1)
\]
respectively so that
\[
B' - B = d - k + 1 = \alpha.
\]

We begin by constructing an MSR-point-optimal $[n', k', d']$ regenerating code $C'$ in systematic form with the first $k'$ rows containing the $B'$ message symbols. Let $C''$ be the subcode of $C'$ consisting of all code matrices in $C'$ whose top row is the all-zero row. Clearly, the subcode $C''$ is of size $q^{B' - \alpha} = q^B$. Note that $C''$ also possesses the same regeneration and reconstruction properties as does the parent code $C'$.

Let the code $C$ now be formed from subcode $C''$ by puncturing (i.e., deleting) the first row in each code matrix of $C''$. Clearly, code $C$ is also of size $q^B$. We claim that $C$ is an $[n, k, d]$ regenerating code. The regeneration requirement requires that the $B$ underlying message symbols be recoverable from the contents of any $k$ rows of a code matrix $C$ in $C$. But this follows since, by augmenting the matrices of code $C$ by placing at the top an additional all-zero row, we obtain a code matrix in $C''$ and code $C''$ has the property that the data can be recovered from any $(k + 1)$ rows of each code matrix in $C''$. A similar argument shows that code $C$ also possesses the regeneration property. Clearly if $C'$ is
linear, so is code $C$. Finally, we have

$$d' = ak' + b$$

$$\Rightarrow d + 1 = a(k + 1) + b$$

$$\Rightarrow d = ak + b + (a - 1).$$

By iterating the procedure in the proof of Theorem 2.2.3 above $i$ times we obtain

**Corollary 2.2.4** An explicit $[n' = n + i, k' = k + i, d' = d + i]$ regenerating code $C'$ that achieves the cut-set bound at the MSR point can be used to construct an explicit $[n, k, d]$ regenerating code $C$ that also achieves the cut-set bound at the MSR point. Furthermore if $d' = ak' + b$ in code $C'$, $d = ak + b + i(a - 1)$ in code $C$. If $C'$ is linear, so is $C$.

The corollary below follows from Corollary 2.2.4 above.

**Corollary 2.2.5** An MSR-point optimal regenerating code $C$ with parameters $[n, k, d]$ for any $2k - 2 \leq d \leq n - 1$ can be constructed from an MSR-point optimal regenerating $[n' = n + i, k' = k + i, d' = d + i]$ code $C'$ with $d' = 2k' - 2$ and $i = d - 2k + 2$. If $C'$ is linear, so is $C$.

### 2.3 The Product-Matrix MBR Code Construction

In this section, we identify the specific make-up of encoding matrix $\Psi$ and message matrix $M$ that results in an $[n, k, d]$ MBR code $C$ with $\beta = 1$. A notable feature of the construction is that it is applicable to all feasible values of $[n, k, d]$, i.e., all $\{n, k, d\}$ satisfying $k \leq d \leq n - 1$. Since $C$ is required to be an MBR code with $\beta = 1$, it must possess the reconstruction and regeneration properties required of a regenerating code and in addition, have parameters $\{\alpha, B\}$ that satisfy equations (5.1) and (5.2). Equation (5.2) can be rewritten in the form:

$$B = \binom{k + 1}{2} + k(d - k).$$

Thus the parameter set of the desired $[n, k, d]$ MBR code $C$ is $(\alpha = d, \beta = 1, B = \binom{k+1}{2} + k(d-k))$.

Let $S$ be a $k \times k$ matrix constructed so that the $\binom{k+1}{2}$ entries in the upper-triangular half of the matrix are filled up by $\binom{k+1}{2}$ distinct message symbols drawn from the set $\{u_i\}_{i=1}^B$. The $\binom{k}{2}$ entries in the strictly lower-triangular portion of the matrix are then chosen so as to make the matrix $S$ a symmetric matrix. The remaining $k(d-k)$ message
symbols are used to fill up a second \((k \times (d - k))\) matrix \(T\). The message matrix \(M\) is then defined as the \(d \times d\) symmetric matrix given by

\[
M = \begin{bmatrix}
S & T \\
T^t & 0
\end{bmatrix}.
\] (2.29)

Thus in the terminology of the product matrix framework, we have set the design parameter \(m = d\) in the present construction. Next, define the encoding matrix \(\Psi\) to be any \((n \times d)\) matrix of the form

\[
\Psi = \begin{bmatrix}
\Phi & \Delta
\end{bmatrix},
\]

where \(\Phi\) and \(\Delta\) are \((n \times k)\) and \((n \times d - k)\) matrices respectively chosen in such a way that

(i) any \(d\) rows of \(\Psi\) are linearly independent,

(ii) any \(k\) rows of \(\Phi\) are linearly independent.

The above requirements can be met, for example, by choosing \(\Psi\) to be either a Cauchy [33] or else a Vandermonde matrix. As per the product-construction framework, the code matrix is then given by \(C = \Psi M\).

The two theorems below establish that the code \(C\) is an \([n, k, d]\) MBR code by establishing respectively, the reconstruction and regeneration properties of the code.

**Theorem 2.3.1 (MBR Reconstruction)** In the code \(C\) presented, all the \(B\) message symbols can be recovered by connecting to any \(k\) nodes, i.e., the message symbols can be recovered through linear operations on the entries of any \(k\) rows of the matrix \(C\).

**Proof** Let

\[
\Psi_{DC} = \begin{bmatrix}
\Phi_{DC} & \Delta_{DC}
\end{bmatrix}
\] (2.30)

be the \(k \times \alpha\) submatrix of \(\Psi\), corresponding to the \(k\) rows of \(\Psi\) to which the data collector connects. Thus the DC has access to the symbols

\[
\Psi_{DC} M = \begin{bmatrix}
\Phi_{DC} S + \Delta_{DC} T^t & \Phi_{DC} T
\end{bmatrix}.
\] (2.31)

By construction, \(\Phi_{DC}\) is a non-singular matrix. Hence, by multiplying the matrix \(\Psi_{DC} M\) on the left by \(\Phi_{DC}^{-1}\), one can recover first \(T\) and subsequently, \(S\).

**Theorem 2.3.2 (MBR Exact Regeneration)** In the code \(C\) presented, exact regeneration of any failed node can be achieved, by connecting to any \(d\) of the \((n - 1)\) nodes remaining.
2.3 The Product-Matrix MBR Code Construction

Proof: Let $\psi^t_f$ be the row of $\Psi$ corresponding to the failed node $f$. Thus the $d$ symbols stored in the failed node correspond to the vector $\psi^t_f M$. (2.32)

The replacement for the failed node $f$ connects to an arbitrary set $\{h_i | i = 1, 2, \ldots, d\}$ of $d$ helper nodes (Fig. 2.4). Upon being contacted by the replacement node, the helper node computes the inner product $<\psi^t_{h_j}, \psi^t_f M>$ and passes on the pair $\left(\psi^t_{h_j}, <\psi^t_{h_j}, \psi^t_f M>\right)$ to the replacement node. Thus, in the present construction, the digest $\mu_f$ equals $\psi^t_f$ itself.

The replacement node computes $\Psi_{\text{rep}} M \psi^t_f$ by aggregating the inputs from the $d$ helper nodes where

$$\Psi_{\text{rep}} = \begin{bmatrix} \psi^t_{h_1} \\ \psi^t_{h_2} \\ \vdots \\ \psi^t_{h_d} \end{bmatrix}.$$  

By construction, the $d \times d$ matrix $\Psi_{\text{rep}}$ is invertible. The partial decoder (Fig. 2.5) in the replacement node “decodes” $M \psi^t_f$ through multiplication on the left by $\Psi_{\text{rep}}^{-1}$. As $M$ is symmetric,

$$\left(M \psi^t_f\right)^t = \psi^t_f M \quad (2.33)$$

which is precisely the data previously stored in the failed node. ■

2.3.1 An Example for the Product-Matrix MBR Code

Let $n = 6$, $k = 3$, $d = 4$. Then $\alpha = d = 4$ and $B = 9$. Let us choose $q = 7$ so we are operating over $\mathbb{F}_7$. The matrices $S, T$ are filled up by the 9 message symbols $\{u_i\}_{i=1}^9$ as shown below:

$$S = \begin{bmatrix} u_1 & u_2 & u_3 \\ u_2 & u_4 & u_5 \\ u_3 & u_5 & u_6 \end{bmatrix}, \quad T = \begin{bmatrix} u_7 \\ u_8 \\ u_9 \end{bmatrix}, \quad (2.34)$$

so that the message matrix $M$ is given by

$$M = \begin{bmatrix} u_1 & u_2 & u_3 & u_7 \\ u_2 & u_4 & u_5 & u_8 \\ u_3 & u_5 & u_6 & u_9 \\ u_7 & u_8 & u_9 & 0 \end{bmatrix}. \quad (2.35)$$
We choose $\Psi$ to be the $(6 \times 4)$ Vandermonde matrix over $\mathbb{F}_7$ given by

$$
\Psi = \begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 2 & 4 & 1 \\
1 & 3 & 2 & 6 \\
1 & 4 & 2 & 1 \\
1 & 5 & 4 & 6 \\
1 & 6 & 1 & 6
\end{bmatrix}.
$$  \hfill (2.36)

Fig. 2.7 shows at the top, the $(6 \times 4)$ code matrix $C = \Psi M$ with entries expressed as functions of the message symbols $\{u_i\}$. The rest of the figure explains how regeneration of failed node 1 takes place. To regenerate node node 1, the helper nodes (nodes 2, 4, 5, 6 in the example), pass on their respective inner products $<\Psi_\ell, 1>$ where $1 = [1 1 1 1]^t$ for $\ell = 2, 4, 5, 6$. The partial decoder in the replacement node then recovers the data stored in the failed node by multiplying by $\Psi_{\text{rep}}^{-1}$ where

$$
\Psi_{\text{rep}} = \begin{bmatrix}
1 & 2 & 4 & 1 \\
1 & 4 & 2 & 1 \\
1 & 5 & 4 & 6 \\
1 & 6 & 1 & 6
\end{bmatrix},
$$  \hfill (2.37)

as explained in the proof of Theorem 5.1.2 above.

### 2.3.2 Systematic Version of the Code

Any exact-regenerating code can be made systematic through a non-singular transformation of the message symbols. In the present case, there is a simpler approach, the matrix
2.4 Analysis and Advantages of the Codes

Ψ can be chosen in such a way that the code is automatically systematic. We simply make the choice:

\[
\Psi = \begin{bmatrix}
I_k & 0 \\
\Phi & \Delta
\end{bmatrix},
\]

(2.38)

where \(I_k\) is the \(k \times k\) identity matrix, \(\Phi\) and \(\Delta\) are matrices of sizes \((k \times n - k)\) and \((d - k \times n - k)\) respectively, such that \(\begin{bmatrix} \Phi & \Delta \end{bmatrix}\) is a Cauchy matrix \(^1\). Clearly the code is systematic. It can be verified that the matrix \(\Psi\) has the properties listed just above Theorem 2.3.1.

2.4 Analysis and Advantages of the Codes

In this section, we detail the system-implementation advantages of the MSR and MBR code constructions presented in this chapter.

2.4.1 Reduced Overhead

In the product-matrix based constructions provided, the data stored in the \(i\)th storage node in the system is completely determined by the single encoding vector \(\psi_i\) of length \(d\) as opposed to a \((B \times \alpha)\) generator matrix in general that is comprised of the \(\alpha\) global kernels each of length \(B\), each associated to a different symbols stored in the node. The encoding vector suffices for encoding, reconstruction, and regeneration purposes. The short length of the encoding vector reduces the overhead associated with the need for nodes to communicate their encoding vectors to the DC during reconstruction, and to the replacement node during regeneration of a failed node.

Also, in both MBR and MSR code constructions, during regeneration of a failed node, the information passed on to the replacement node by a helper node is only a function of the index of the failed node. Thus, it is independent of the identity of the \(d-1\) other nodes that are participating in the regeneration. Once again, this reduces the communication overhead by requiring less information to be disseminated.

2.4.2 Applicability to Arbitrary \(n\)

The existing, explicit constructions of exact-regenerating codes \([5, 6, 17–19]\) restrict the value of \(n\) to be \(d+1\). In contrast, the codes presented in this chapter are applicable for all values of \(n\), and independent of the values of the parameters \(k\) and \(d\). This makes the code constructions presented here practically appealing, as in any real-world distributed storage application such as a peer-to-peer storage, cloud storage, etc, expansion or shrinking of

\(^1\)In general, any matrix, all of whose submatrices are of full rank will suffice.
the system size is very natural. For example, in peer-to-peer systems, individual nodes are free to come and go at their own will.

The system can perform optimal regeneration as long as at least $d$ nodes are active. Addition of new nodes to the system can be accomplished by treating it as regeneration of failed nodes by assigning suitable encoding vectors to the new nodes.

2.4.3 Complexity

**Field size:** The size of the finite field required in the constructions is of the order of $n$, and no further restrictions are imposed on the field size.

**Striping:** The codes presented here are for the atomic case of $\beta = 1$ (See Section 1.2.1). Since each stripe is of minimal size, the complexity of encoding, reconstruction and regeneration operations, are considerably lowered, and so are the buffer sizes required at data collectors and replacement nodes. Furthermore, the operations that need to be performed on each stripe are identical and independent, and hence can be performed in parallel efficiently by a GPU/FPGA/multi-core processor.

**Choice of the encoding matrices:** The encoding matrices for all the codes described, can be chosen as Vandermonde matrices, in which case, each encoding vector can be described by a scalar. Moreover, the encoding, reconstruction, and regeneration operations will be, for the most part, identical to encoding or decoding of conventional Reed-Solomon codes.

2.5 An Ideal Regenerating Code: Simultaneously Minimizing Storage and Bandwidth

As seen in Section 1.1.2, in the traditional regenerating codes setup, there exists a tradeoff between the amount of storage and the repair bandwidth and both cannot be minimized simultaneously. One of the extreme points of the tradeoff, Minimum Storage Regeneration (MSR) point corresponds to the minimum amount of storage at the nodes, and the other extreme point, Minimum Bandwidth Regenerating (MBR) point corresponds to the minimum repair bandwidth.

In this section, we present an explicit construction of a regenerating code termed *ideal regenerating code*, which minimizes both the amount of storage and the repair bandwidth simultaneously, by relaxing certain conditions in the original regenerating codes setup. i.e., we can simultaneously have

$$\alpha = B/k,$$  \hspace{1cm} (2.39)
and
\[ d\beta = \alpha. \] (2.40)

Thus the code has advantages of both the MSR and the MBR points and the construction is via the product-matrix framework.

### 2.5.1 Description of the Setting

Compared to the original setup, there are only a few deviations in the case of ideal regenerating code. Nevertheless, the complete system is described here for clarity.

The ideal regenerating code distributes a file of size \( B \) symbols, over a finite field \( \mathbb{F}_q \) of size \( q \), across a network of storage nodes having capacity to store \( \alpha \) symbols. Each storage node in the network is designated to be of one of two types – type 0 or type 1, as depicted in Figure 2.8. The type of each node can be arbitrary, and can be fixed at run-time. Let \( n_0 \) and \( n_1 \) be the number of nodes of type 0 and type 1 respectively.

A data collector connects any \( k \) nodes of the same type to recover the data. This is depicted in Fig 2.8a. As in the original setup, the reconstruction property mandates
\[ \alpha \geq \frac{B}{k}. \] (2.41)

The ideal regenerating code achieves the minimum amount of storage at each node, and thus we have
\[ \alpha = \frac{B}{k}. \] (2.42)

We assume that the value of \( B \) is a multiple of \((k^2)\). This assumption is valid since the source file can always be padded with zeros in order to satisfy this condition, and the loss in efficiency due to padding will be negligible since the file size \( B \) is typically large (of the order of Gigabytes to Petabytes) as compared to the value of \( k \) (of the order of tens to hundreds).

A failed node is replaced by a node which is its exact replica, i.e., regeneration is exact. A replacement node connects to \( k \) nodes of the other type (the complementary type of the failed node) \(^3\), downloading \( \beta = \frac{\alpha}{k} \) symbols from each. This is depicted in Fig 2.8b. Note that the parameter \( d \) in the original setup is equal to \( k \) in this case.

Thus the repair bandwidth \( d\beta \) which is equal to \( \alpha \). Since the replacement node desires \( \alpha \) symbols, the minimum repair bandwidth possible is \( \alpha \). Thus, ideal regenerating code achieves minimum repair bandwidth as well.

\(^2\)This is a relaxation from the original regenerating codes, where a data collector can connect to any \( k \) nodes.

\(^3\)This is a relaxation from the original setup, where a replacement node can connect to any \( d \) nodes, \( d \geq k \) being a design parameter of the code.
2.5 An Ideal Regenerating Code: Simultaneously Minimizing Storage and Bandwidth

Thus, the value of the parameters $\alpha$ and $B$ for ideal regenerating code are

$$\alpha = k\beta,$$  
(2.43)

$$B = k^2\beta.$$  
(2.44)

There is indeed a slight loss in the symmetry of the system as the storage nodes are divided into two types and are no longer alike. However the advantage of simultaneously minimizing both the amount of storage and the repair bandwidth, outweighs the disadvantage of losing the symmetric nature.

2.5.2 Explicit Ideal Regenerating Code

Observing equations (2.43) and (2.44), we see that both $\alpha$ and $B$ are multiples of $\beta$ and hence we can use the concept of striping discussed in Section 1.2.1. As before, we construct codes for the atomic case of $\beta = 1$ and codes for higher values of $\beta$ can be easily obtained by concatenation. We document below the values of the parameters $\alpha$ and $B$ for $\beta = 1$,

$$\alpha = k,$$  
(2.45)

$$B = k^2.$$  
(2.46)
2.5 An Ideal Regenerating Code: Simultaneously Minimizing Storage and Bandwidth

**Code Construction:** Let $M_0$ be a $(k \times k)$ matrix consisting of the $k^2$ source symbols. Let $M_1$ be the transpose of $M_0$. Hence

$$M_1 = M_0^t. \quad (2.47)$$

The message matrix $M$ is defined as the $(2k \times k)$ matrix given by

$$M = \begin{bmatrix} M_0 \\ M_1 \end{bmatrix}. \quad (2.48)$$

Next we define the encoding matrix $\Psi$ to be a $((n_0 + n_1) \times 2k)$ matrix

$$\Psi = \begin{bmatrix} \Psi_0 & 0 \\ 0 & \Psi_1 \end{bmatrix}, \quad (2.49)$$

where $\Psi_1$ is an $(n_0 \times k)$ matrix with any $k$ rows are linearly independent, and $\Psi_2$ is an $(n_1 \times k)$ matrix also with any $k$ rows linearly independent. For instance, one can choose either of the matrices to be a Vandermonde matrix or a Cauchy matrix.

Thus, under the Product-Matrix framework, the $k$ symbols contained in a node are given by the $k$ symbols in the corresponding row of the code matrix:

$$C = \Psi M. \quad (2.50)$$

Denote the $i^{th}$ rows of $\Psi_0$ and $\Psi_1$ by $\psi_{(0,i)}^t$ and $\psi_{(1,i)}^t$ respectively. Thus, node $i$ of type 0 stores the $i^{th}$ row of $\Psi_0 M_0$:

$$\psi_{(0,i)}^t M_0, \quad (2.51)$$

and node $i$ of type 1 stores the $i^{th}$ row of $\Psi_1 M_1$:

$$\psi_{(1,i)}^t M_1. \quad (2.52)$$

The two theorems below establish the regeneration and the reconstruction properties of the code.

**Theorem 2.5.1 (Exact Regeneration)** In the code $C$ presented, exact regeneration of any failed node can be achieved by connecting to any $k$ nodes of the type different from that of the failed node.

**Proof** Let $p \ (\in \{0, 1\})$ be the type chosen of the failed node, and let this node correspond to the row $\psi_{(0,p)}^t$ of $\Psi_p$.

Now, the replacement node $f$ connects to an arbitrary $\{h_i \mid i = 1, 2, \ldots, k\}$ of $k$ helper nodes, each of type $\overline{p} = 1 - p$. Helper node $h_i$ computes the inner product of its stored
2.5 An Ideal Regenerating Code: Simultaneously Minimizing Storage and Bandwidth

symbols with the digest of the replacement node, i.e.,

\[ \left< \psi^t_{\tau(h_i)} M_{\tau} , \psi_{\tau(p,f)} \right> \]  \hspace{1cm} (2.53)

It then passes on the pair \( \left< \psi^t_{\tau(h_i)} , \psi_{\tau(p,f)} \right> \).

The replacement node aggregates the inputs from the \( k \) helper nodes as \( \Psi_{\text{rep}}M_{\tau}\psi_{\tau(p,f)} \) where

\[ \Psi_{\text{rep}} = \begin{bmatrix} \psi^t_{\tau(h_1)} \\ \psi^t_{\tau(h_2)} \\ \vdots \\ \psi^t_{\tau(h_k)} \end{bmatrix} \]  \hspace{1cm} (2.54)

By construction, the \( k \times k \) matrix \( \Psi_{\text{rep}} \) is invertible. Thus the replacement node now has access to \( M_{\tau}\psi_{\tau(p,f)} \). Furthermore, by construction, the \( M_{\tau} = M_{\tau}^t \); hence by taking a transpose the replacement node obtains its desired symbols:

\[ \left( M_{\tau}\psi_{\tau(p,f)} \right)^t = \psi^t_{\tau(p,f)}M_{\tau}. \]  \hspace{1cm} (2.55)

\[ \square \]

**Remark 2.5.2** On failure of a node, the replacement node can alternatively be chosen of a type different from that of the failed node. The choice of the type to be assigned to the replacement can be chosen arbitrarily at the time of regeneration.

**Theorem 2.5.3 (Reconstruction)** In the code \( C \) presented, all the \( B \) message symbols can be recovered by connecting to any \( k \) nodes of same type.

**Proof** A data collector connects to \( k \) nodes of a particular type, say nodes \( i_1, \ldots, i_k \), each of type \( p \); it downloads the \( k^2 \) symbols \( \left\{ \psi^t_{\tau(p,i_1)}M_{\tau}, \psi^t_{\tau(p,i_2)}M_{\tau}, \ldots, \psi^t_{\tau(p,i_k)}M_{\tau} \right\} \). Define a matrix \( \Psi_{\text{DC}} \) as

\[ \Psi_{\text{DC}} = \begin{bmatrix} \psi^t_{\tau(p,i_1)} \\ \psi^t_{\tau(p,i_2)} \\ \vdots \\ \psi^t_{\tau(p,i_k)} \end{bmatrix} \]  \hspace{1cm} (2.56)

Thus, the symbols that the data collector obtains are precisely the elements of the matrix \( \Psi_{\text{DC}}M_{\tau} \). Since by construction, the \( k \) rows of \( \Psi_{\tau} \) are linearly independent, the matrix \( \Psi_{\text{DC}} \) is invertible, and the data collector can recover all the source symbols in \( M_{\tau} \). \[ \square \]
An open problem in the area of regenerating codes that has recently drawn attention is as to whether or not the storage-repair bandwidth tradeoff is achievable under the additional requirement of exact regeneration. It has previously been shown that the MSR point is not achievable when \( d \leq 2k - 3 \) for the atomic case of \( \beta = 1 \) (Section 4.4, see also [5], [2]), but is achievable for all parameters \([n, k, d]\) when \( B \) (and hence \( \beta \) as well) is allowed to approach infinity [23], [24]. The present chapter answers this question in negative for the interior points.

In this chapter, we switch to a subspace based viewpoint, where the nodes are considered to store and pass subspaces instead of symbols. This perspective of looking at regenerating codes can be employed to describe any linear storage code. In Section 3.3, certain necessary properties that any exact regenerating code must satisfy are derived. These properties will subsequently be used in Section 3.4 to establish non-achievability of the interior points of the storage-repair bandwidth tradeoff under exact regeneration, except for a region of length at most \( \beta \) at the immediate vicinity of the MSR point.

### 3.1 Subspace Framework Description

Define a column vector \( \mathbf{u} \) of length \( B \) consisting of the source symbols, with each source symbol independently taking values from finite field \( \mathbb{F}_q \) of size \( q \). In a linear code, any stored symbol can be written as \( \mathbf{u}' \mathbf{g} \) a unique \( B \)-length vector \( \mathbf{g} \) having elements from \( \mathbb{F}_q \); and adopting the terminology of network coding, the column vector \( \mathbf{g} \) will be termed the
global kernel associated to that stored symbol. Furthermore, since a storage node stores \( \alpha \) symbols, it is associated with \( \alpha \) global kernels. We say that the \( i \)th node \textit{stores} the vectors \( g_1^{(i)}, \ldots, g_\alpha^{(i)} \). Linear operations performed on the stored symbols can be visualized by the same operations performed on the global kernels. As a node is free to perform any linear operation on the symbols stored in it, we say that each node \textit{stores a subspace} of dimension at most \( \alpha \), and term it as its \textit{nodal subspace}. For \( i = 1, \ldots, n \), denote the \( i \)th nodal space by \( W_i \):

\[
W_i = \langle g_1^{(i)}, \ldots, g_\alpha^{(i)} \rangle
\]

where \( \langle \cdot \rangle \) indicates the span of vectors.

For regeneration of a failed node, \( d \) existing nodes provide \( \beta \) symbols each to the replacement node. As the replacement node is free to perform linear operations on the symbols downloaded, we say that each node \textit{passes a subspace} of dimension at most \( \beta \) to the replacement node. Consider exact regeneration of node \( l \) using a set \( D \) of \( d \) arbitrary nodes, and let \( j \in D \). Further, let \( S_D^{(j,l)} \) denote the subspace passed by node \( j \) for the regeneration of node \( l \).

With a little thought, the reconstruction and regeneration requirements can be stated in the subspace terminology as below. For reconstruction, for every subset of \( k \) storage nodes: \( \{i_j \mid 1 \leq j \leq k\} \), we need

\[
|W_{i_1} + W_{i_2} + \cdots + W_{i_k}| = B. 
\]  

(3.1)

where \(|.\)| indicates the dimension of the vector space. Furthermore, exact regeneration of node \( l \), when the replacement node connects to the \( d \) nodes in set \( D \) is possible iff

\[
W_l \subseteq \sum_{j \in D} S_D^{(j,l)}. 
\]  

(3.2)

In the sequel, we will drop the subscript \( D \), and the set of \( d \) nodes participating in the regeneration will be clear from the context. Note that for any node \( j \),

\[
S^{(j,l)} \subseteq W_j. 
\]  

(3.3)

\section*{3.2 A Representation for the Points on the Tradeoff}

Clearly, to enable data recovery from any \( k \) nodes, we need \( \alpha \geq \frac{B}{k} \). From Section 1.1.3, we know that the case of \( \alpha = \frac{B}{k} \) corresponds the MSR point and the value of \( \alpha \) is given by

\[
\alpha = (d - k + 1)\beta. 
\]  

(3.4)
On the other hand, noting that a replacement node need to necessarily download at least as many symbols as were stored in the failed node, we get

$$\alpha \leq d\beta.$$  \hfill (3.5)

Again, from Section 1.1.3, we know that the case of $\alpha = d\beta$ corresponds the MBR point. Thus, we get the range of $\alpha$ as

$$(d - (k - 1))\beta \leq \alpha \leq d\beta.$$  \hfill (3.6)

Taking a cue from this, we characterize any point on the tradeoff in terms of $\alpha$ as

$$\alpha = (d - p)\beta - \theta,$$  \hfill (3.7)

where $p$ is a positive integer taking value from the set $\{0, 1, \ldots, k - 1\}$, and $\theta$ is a positive integer such that

$$\theta = \begin{cases} (-\alpha) \mod \beta & \text{if } p < k - 1 \\ 0 & \text{if } p = k - 1. \end{cases}$$  \hfill (3.8)

The storage-repair bandwidth tradeoff can thus be classified into three sections:

(i) The MSR point: $p = k - 1$ (which implies $\theta = 0$),

(ii) The MBR point: $p = 0$, $\theta = 0$, and

(iii) The Interior points: $\{1 \leq p \leq k - 2, \text{ any } \theta\} \cup \{p = 0, \theta > 0\}$.

### 3.3 Subspace Properties of Linear Exact Regenerating Codes

In this section, we consider a hypothetical linear exact regenerating code $C$, whose parameters satisfy the network coding bound given by equation (1.3). A set of properties that the nodal subspaces and the subspaces passed for regeneration must necessarily satisfy are derived; the proofs are relegated to Appendix A.

First, we present two lemmas establishing relations between the dimension of subspaces of a vector space.

**Lemma 3.3.1** For any subspaces $V_1$, $V_2$ and $V_3$ satisfying $V_1 \subseteq V_2 + V_3$,

$$|V_1| - |V_1 \cap V_2| = |V_3 \cap (V_1 + V_2)| - |V_3 \cap V_2|$$  \hfill (3.9)
3.3 Subspace Properties of Linear Exact Regenerating Codes

Proof

\[ |V_3 \cap (V_1 + V_2)| = |V_3| + |V_1 + V_2| - |V_1 + V_2 + V_3| \]  
\[ = |V_3| + |V_1 + V_2| - |V_3 + V_2| \]  
\[ = |V_1| - |V_1 \cap V_2| + |V_3 \cap V_2|. \]  
(3.10)

\[ = |V_3| + |V_1 + V_2| - |V_3 + V_2| \]  
\[ = |V_1| - |V_1 \cap V_2| + |V_3 \cap V_2|. \]  
(3.11)

\[ = |V_1| - |V_1 \cap V_2| + |V_3 \cap V_2|. \]  
(3.12)

Lemma 3.3.2 For any subspaces \( U_1, U_2 \) and \( U_3 \),

\[ |U_1 + U_2| - |(U_1 + U_2) \cap U_3| \geq |U_1| - |U_1 \cap U_3|. \]  
(3.13)

Proof

\[ |U_1 + U_2| - |(U_1 + U_2) \cap U_3| = |U_1 + U_2 + U_3| - |U_3| \]  
\[ \geq |U_1 + U_3| - |U_3| \]  
\[ = |U_1| - |U_1 \cap U_3|. \]  
(3.14)

\[ \geq |U_1| - |U_1 \cap U_3|. \]  
(3.15)

\[ = |U_1| - |U_1 \cap U_3|. \]  
(3.16)

Next, we establish properties relating to the nodal subspaces in \( C \). The proofs are relegated to Appendix A.

Property 1 (Dimension of Nodal Subspaces) For any storage node \( i \),

\[ |W_i| = \alpha, \quad \forall i \in \{1, \ldots, n\}. \]

Thus, this condition mandates each of the nodes to store \( \alpha \) linearly independent symbols.

Property 2 (Intersection Property of Nodal Subspaces) For any set of nodes \( A \) of cardinality \( a \),

\[ |W_l \cap \sum_{i \in A} W_i| = \begin{cases} 0 & a \leq p \\ (a - p)\beta - \theta & p < a < k \\ \alpha & a \geq k \end{cases} \]  
(3.17)

where node \( l \notin A \).

We now move on to the properties governing the subspaces passed in the process of regeneration.
3.3 Subspace Properties of Linear Exact Regenerating Codes

Property 3 (Dimension of Passed Subspaces) For exact regeneration of any node $l$, and any assisting node $m$,

$$|S^{(m,l)}| = \beta.$$  \hspace{1cm} (3.18)

Thus, the number of symbols passed by an assisting node to a replacement node can be no less than $\beta$.

Property 4 (Union Property of Passed Subspaces) For $p < k - 2$, a set $A$ comprising of an arbitrary selection of at most $p + 2$ nodes, and an arbitrary node $m \notin A$,

$$\left| \sum_{l \in A} S^{(m,l)} \right| \leq 2\beta - \theta,$$  \hspace{1cm} (3.19)

where $S^{(m,l)}$ is the subspace that node $m$ passes for the regeneration of node $l$, when the set of nodes to which node $l$ connects includes $\{m\} \cup A \setminus \{l\}$.

Property 5 (Intersection Property of Passed Subspaces) For $p < k - 1$, a set $A$ comprising of an arbitrary selection of at most $p + 1$ nodes, and an arbitrary node $m \notin A$,

$$\left| \bigcap_{l \in A} S^{(m,l)} \right| \geq \beta - \theta$$  \hspace{1cm} (3.20)

where $S^{(m,l)}$ is the subspace that node $m$ passes for the regeneration of node $l$, when the set of nodes to which node $l$ connects includes $\{m\} \cup A \setminus \{l\}$.

Corollary 3.3.3 For $0 < p < k - 1$, a set $A$ comprising of an arbitrary selection of a ($\leq d$) nodes, and an arbitrary node $m \notin A$,

$$\left| \sum_{l \in A} S^{(m,l)} \right| \leq \beta + (a - 1)\theta,$$  \hspace{1cm} (3.21)

where $S^{(m,l)}$ is the subspace that node $m$ passes for the regeneration of node $l$, when the set of nodes to which node $l$ connects includes $\{m\} \cup A \setminus \{l\}$. 
3.4 Non-achievability of the Interior Points On the Tradeoff for Exact Regeneration

In the preceding section, we derived a set of properties that the subspaces stored and passed by the nodes must satisfy. It turns out, as we will see in this section, that these properties over-constrain the system, thereby rendering most of the points on the storage-repair bandwidth tradeoff as non-achievable for exact regeneration (with respect to linear codes).

We first consider the case when $\alpha$ is a multiple of $\beta$, and present a proof establishing the non-achievability of any interior point on the tradeoff for exact regeneration using linear codes. Note that this covers the important and atomic case of $\beta = 1$. The case when $\alpha$ is not a multiple of $\beta$ is considered subsequently in Theorem 3.4.2.

Recall that $\alpha = (d - p)\beta - \theta$, with $\theta = (-\alpha) \mod \beta$, and the interior points correspond to the parameters \{1 \leq p \leq k - 2, \text{ any } \theta\} \cup \{p = 0, \theta > 0\}.

**Theorem 3.4.1** When $\alpha$ is a multiple of $\beta$, linear exact-regenerating codes for any interior point of the storage-repair bandwidth tradeoff meeting the cut-set bound do not exist.

**Proof** Since $\alpha$ is a multiple of $\beta$, we have $\theta = 0$, thereby restricting the value of $p$ for the interior points to $1 \leq p \leq k - 2$.

The proof is by contradiction: we assume there exists a linear exact-regenerating code meeting the cut-set bound for interior points of the storage-repair bandwidth tradeoff. For the rest of the proof, we will restrict our attention to an arbitrary set of $d + 1$ storage nodes, and equivalently considering, without loss of generality, a system comprising of only these storage nodes with the system parameter $n$ equal to $d + 1$.

Property 5, along with $\theta = 0$, mandates that for any set $A$ comprising of $p + 1(\geq 2)$ nodes, and node $m \notin A$,

$$\left| \bigcap_{l \in A} S^{(m,l)} \right| \geq \beta .$$ (3.22)

Since $|S^{(m,l)}| = \beta$, the subspaces $S^{(m,l)}$ are equal $\forall l \in A$. Furthermore, since the choice of set $A$ arbitrary, it follows that the subspaces $S^{(m,l)}$ are equal $\forall l \neq m$. Thus, any node $m$ passes the same subspace for regeneration of all the other $d$ nodes in the system; and we denote this subspace by $S^{(m)}$. Thus, $\sum_{\forall j} S^{(j)}$ spans the nodal subspaces of all the nodes in the system, and hence for reconstruction to hold, we need

$$(d + 1)\beta \geq \left| \sum_{\forall j} S^{(j)} \right| \geq B.$$ (3.23)
3.4 Non-achievability of the Interior Points On the Tradeoff for Exact Regeneration

On the other hand, from the cut-set bound in (1.3), we have the relation

\[ B = \sum_{i=0}^{k-1} \min \left( (d - p)\beta, (d - i)\beta \right) \]  

\[ \geq 2(d - p)\beta + \sum_{i=2}^{k-1} \min \left( (d - p)\beta, (d - i)\beta \right) \]  

\[ \geq 2(d - p)\beta + (k - 2)\beta \]  

\[ > (d + 1)\beta \]  

where equation (3.24) uses the fact that \( \alpha = (d - p)\beta \), equation (3.25) holds since \( p \geq 1 \), and equations (3.26) and (3.27) are derived using \( d \geq k \geq p + 2 \). This is in contradiction to equation (3.23).

**Theorem 3.4.2** When \( \alpha \) is not a multiple of \( \beta \), for any interior point of the storage-repair bandwidth tradeoff, linear exact-regenerating codes meeting the cut-set bound do not exist, except possibly for the case

\[ p = k - 2 \text{ with } \left( \theta \geq \frac{d - p - 1}{d - p} \beta \text{ or } k = 2, \theta > 0 \right) . \]

**Proof** The proof is again via contradiction: suppose that there exists such a code. We first consider the case of \( k > p + 2 \). Consider any two nodes \( x, y \), and \( (d - 1) \) other nodes. Partition the \( (d - 1) \) other nodes considered into two sets, \( T_1 \) of cardinality \( p \) and \( T_2 \) of cardinality \( (d - p - 1) \). Consider regeneration of node \( x \) by connecting to \( \{y\} \cup T_1 \cup T_2 \) and regeneration of node \( y \) by connecting to \( \{x\} \cup T_1 \cup T_2 \). We need

\[ W_x \subseteq \sum_{i \in T_1} S^{(i,x)} + \sum_{i \in T_2} S^{(i,x)} + S^{(y,x)} , \]  

\[ W_y \subseteq \sum_{i \in T_1} S^{(i,y)} + \sum_{i \in T_2} S^{(i,y)} + S^{(x,y)} . \]

However, since \( S^{(i,y)} \subseteq W_i \) we have

\[ W_y \subseteq \sum_{i \in T_1} W_i + \sum_{i \in T_2} S^{(i,y)} + S^{(y,x)} . \]  

Substituting \( W_x \) from equation (3.28) in equation (3.30), we get

\[ W_x + W_y \subseteq \sum_{i \in T_1} W_i + \sum_{i \in T_2} (S^{(i,x)} + S^{(i,y)}) + S^{(y,x)} . \]

We further define three vector spaces: \( V_1 = W_x + W_y, V_2 = \sum_{i \in T_1} W_i \) and \( V_3 = \sum_{i \in T_2} (S^{(i,x)} + S^{(i,y)}) + S^{(y,x)} \). Noting that \( V_1 \subseteq V_2 + V_3 \), we apply Lemma 3.3.1 to
get

$$|V_3| \geq |V_1| - |V_1 \cap V_2|.$$  

(3.32)

Now,

$$|V_3| = \left| \sum_{i \in T_2} \left( S^{(i,x)} + S^{(i,y)} \right) + S^{(y,x)} \right|$$  

(3.33)

$$\leq \sum_{i \in T_2} \left( |S^{(i,x)} + S^{(i,y)}| + |S^{(y,x)}| \right)$$  

(3.34)

$$\leq \sum_{i \in T_2} (2\beta - \theta) + \beta$$  

(3.35)

$$= (2d - 2p - 1)\beta - (d - p - 1)\theta.$$  

(3.36)

where equation (3.35) follows from Property 4.

On the other hand,

$$|V_1| - |V_1 \cap V_2| = |W_x + W_y| - \left| (W_x + W_y) \cap \left( \sum_{i \in T_1} W_i \right) \right|$$  

(3.37)

$$= \left| \sum_{i \in T_1 \cup \{x,y\}} W_i \right| - \left| \sum_{i \in T_1} W_i \right|$$  

(3.38)

$$= ((p + 2)\alpha - (\beta - \theta)) - (p\alpha)$$  

(3.39)

$$= 2\alpha - (\beta - \theta)$$  

(3.40)

$$= (2d - 2p - 1)\beta - \theta;$$  

(3.41)

where equation (3.39) follows from Properties 1 and 2.

Now, since $\theta \neq 0$ and $d \geq k > p + 2$, equations (3.36) and (3.41) imply

$$|V_3| < |V_1| - |V_1 \cap V_2|,$$  

(3.42)

which is in contradiction to equation (3.32).

We now proceed to the case of $k = p + 2, \ p \neq 0$. The proof for this case follows equations (3.28) through (3.34) from the previous case. However, since Property 4 need not hold for this case, the proof will diverge from here on.

Apply Corollary 3.3.3 (with $a = 2$) to equation (3.34), to get

$$\left| S^{(i,x)} + S^{(i,y)} \right| \leq \beta + \theta;$$  

(3.43)

and hence,

$$|V_3| \leq (d - p)\beta + (d - p - 1)\theta.$$  

(3.44)
Now, equations (3.37) through (3.41) are also valid for this case. Comparing the value of $|V_1| - |V_1 \cap V_2|$ obtained in (3.41) with that of $V_3$ above, we can infer that equation (3.32) is satisfied only if $\theta \geq \frac{d-p-1}{d-p} \beta$. Hence the cut-set bound is not achievable for the parameter $k = p + 2, \ p \neq 0$ when

$$\theta < \frac{d-p-1}{d-p} \beta.$$  \hfill (3.45)

Thus the interior points of the tradeoff are not achievable under exact regeneration except possibly for an uncertain region in the vicinity of the MSR point. The uncertain region for the parameters $B = 27000, \ k = 10, \ d = 18, \ n > 18$ is depicted in Figure 3.1.

**Remark 3.4.3** The properties derived in Section 3.3, and the non-achievability results in the present section continue to hold even if optimal exact regeneration of only $k$ of the nodes is desired, and the remaining $n - k$ nodes are permitted to be regenerated by downloading the full file.

**An Achievable Curve via Storage Space Sharing**

Recall that for a fixed $B$, at the MSR point

$$\alpha = (d-k+1) \beta,$$

and at the MBR point

$$\alpha = d \beta.$$

Due to the striped nature of our MSR and MBR codes, the system can be operated at any intermediate point (i.e., $(d-k+1) \beta \leq \alpha \leq d \beta$) between the MSR and MBR points via storage-space sharing. Here, the source file of size $B$ is split into two parts: $fB$ and $(1-f)B$ for some fraction $f$, and the two fractions are encoded using codes for the MSR and MBR points respectively, that were introduced in Sections 2.2 and 2.3. Note that we will assume $d \geq 2k - 2$, as desired by the MSR code.

The following analysis assumes $\alpha$ and $\beta$ as the system parameters and aims to find the maximum data file size $B$ that can be stored. Let $f$ be the fraction of the storage space used for MSR codes, and $(1-f)$ for MBR codes. Then

$$\alpha_{\text{MSR}} = f \alpha, \quad \alpha_{\text{MBR}} = (1-f) \alpha,$$  \hfill (3.46)
and the net $\beta$ as

$$\beta = \beta_{MSR} + \beta_{MBR}$$

$$= \frac{f\alpha}{d-k+1} + \frac{(1-f)\alpha}{d}.$$  \hfill \text{(3.47)}

Thus we get

$$f = \frac{(d-k+1)(d\beta - \alpha)}{(k-1)\alpha}. \quad \text{ (3.49)}$$

Thus the amount of data that can be stored is given by

$$B = B_{MSR} + B_{MBR}$$

$$= \frac{k(\alpha + (d-k+1)\beta)}{2}. \quad \text{ (3.51)}$$
Note that we require the values of $\alpha$, $\beta$ to be integers for both the MSR and the MBR codes involved and hence $f$ needs to have a value such that $\frac{f^\alpha}{d-k+1}$ and $\frac{(1-f)^\alpha}{d}$ are integers. This fringe effect might cause the value of $B$ to be slightly less than that derived above. But in practical systems, $B$ will be high and this fringe effect will become negligible.

For the parameters $B = 27000$, $k = 10$, $d = 18$, $n > 18$, the value of $B$ achievable via storage space sharing is translated to achievable values of repair bandwidth $d\beta$, and plotted alongside the storage-repair bandwidth tradeoff curve in Figure 3.1.
Chapter 4

Interference Alignment for Minimum Storage Regenerating Codes

Recently, the idea of ‘Interference Alignment’ was proposed in [27, 28], in the context of wireless communication. Here, signals of multiple users are designed in such a way that at every receiver, signals from all the unintended users occupy the same half of the signal space, leaving the other half free for the intended user. In this chapter, we use the interference alignment concept in constructing codes for the MSR point. A construction of a family of MDS codes for \( d = n - 1 \geq 2k - 1 \) using interference alignment is presented in Section 4.3, that enable the exact regeneration of systematic nodes while achieving the cut set bound on repair bandwidth. We have termed this code the MISER (for MDS, Interference-aligning, Systematic, Exact-Regenerating) code\(^1\). In Section 4.4.1 we prove that interference alignment is necessary for any code meeting the lower bound on the repair bandwidth. It turns out that, under the assumption that the global kernels passed by a node \( i \) for repair of the \( k \) systematic nodes are linearly independent, the structure of the MISER code is essentially mandated and this is shown in Section 4.5. In Section 4.4.3, we provide a proof that for \( d < 2k - 3 \), it is not possible to construct an MSR code for the atomic case of \( \beta = 1 \) under exact regeneration. This non-existence result is clearly of interest in the light of on-going efforts to construct exact-regenerating MSR codes for the atomic case, meeting the cut-set bound [6, 17, 18, 22]. Also provided in Section 4.6 is a proof of the existence of a code such as the MISER code for the remaining cases \( 2k - 2 \leq d = n - 1 \leq 2k - 3 \).

\(^1\)The name of the code is also a reflection on the miserly nature of the code in terms of bandwidth expended to repair a systematic node.
4.1 Notation

Let $u^{(m)}$ denote the $\alpha$ code symbols stored in node $m$. For $m = 1, \ldots, n$, let $G^{(m)}$ denote the generator matrix of node $m$ with dimensions $B \times \alpha$. We have

$$u^{(m)t} = z^t G^{(m)}. \quad (4.1)$$

Each column of the node generator matrix corresponds to the linear combination vector of a particular symbol stored in the node. We call this vector as *global kernel* of the stored symbol, analogous to its usage in network coding literature.

Bringing all the nodes together, the $B \times n\alpha$ generator matrix for the entire storage code is given by

$$[G^{(1)} \quad G^{(2)} \quad \ldots \quad G^{(n)}]. \quad (4.2)$$

Now, let us partition the $B$ source symbols into $k$ vectors, $z_i$ for $i = 1, \ldots, k$, consisting of $\alpha$ distinct source symbols each. Then the source vector $z$ can be written as

$$z = \begin{bmatrix} z_1 \\ \vdots \\ z_k \end{bmatrix}. \quad (4.3)$$

As a result of this partition of the source vector, it is natural to partition the generator matrix of each node as well into $k$ sub-matrices as

$$G^{(m)} = \begin{bmatrix} G^{(m)}_1 \\ \vdots \\ G^{(m)}_k \end{bmatrix} \quad (4.4)$$

where $G^{(m)}_l$, for $l = 1, \ldots, k$, is an $\alpha \times \alpha$ matrix. Thus,

$$u^{(m)t} = z^t G^{(m)} = \sum_{l=1}^{k} z_l^t G^{(m)}_l. \quad (4.5)$$

These sub-matrices are further partitioned as

$$G^{(m)}_l = \begin{bmatrix} g^{(m)}_{l,1} & \ldots & g^{(m)}_{l,\alpha} \end{bmatrix} \quad (4.6)$$

where each $g^{(m)}_{l,i}$ is an $\alpha$-length column vector.

Out of the $n$ storage nodes, $k$ are systematic and store source symbols in uncoded form. Without loss of generality, let the first $k$ nodes be systematic, i.e., let systematic
node $m$ (for $m = 1, \ldots, k$) store the $\alpha$ source symbols in $z_m$. Hence,

$$u^{(m)} = z_m \quad \text{for} \quad m = 1, \ldots, k. \quad (4.7)$$

Thus, for systematic node $m$, and $l = 1, \ldots, k$,

$$G_l^{(m)} = \begin{cases} I_\alpha & \text{if } l = m \\ 0_\alpha & \text{if } l \neq m \end{cases}. \quad (4.8)$$

We will denote an $\alpha \times \alpha$ zero matrix as $0_\alpha$, and an $\alpha \times \alpha$ identity matrix as $I_\alpha$.

Now, making use of the partitions in the generator matrices, for any node $m$, we refer to $G_l^{(m)}$ as the component along $z_l$, i.e. the component along symbols stored in systematic node $l$. As we will see later, this viewpoint leads to a very elegant framework which facilitates the usage of the interference alignment concept.

When a node fails, it is replaced by a new node which connects to some $d$ of the existing nodes, and downloads one symbol from each node. A node participating in the regeneration of a failed node, passes a linear combination of the symbols stored in it. Equivalently, it passes a linear combination of the columns of its generator matrix. Thus, we can view each node passing a \textit{vector} for regeneration of a failed node and this vector represents the global kernel of the symbol passed. We will use the symbol and the vector viewpoints interchangeably.

Let $D$ denote the set of $d$ existing nodes participating in the regeneration of node $l$. Let

$$\mathbf{v}^{(m,l)}_D = \begin{bmatrix} \mathbf{v}^{(m,l)}_{D,1} \\ \vdots \\ \mathbf{v}^{(m,l)}_{D,k} \end{bmatrix},$$

denote the vector passed by node $m \in D$ for the regeneration of node $l \notin D$ where $\mathbf{v}^{(m,l)}_{D,i}$ ($i = 1, \ldots, k$) are $\alpha$-length column vectors reflecting the partitioning of the node generator matrices. Let $\mathbf{x}^{(m,l)}_D$ be the linear combination vector used by node $m$ to generate $\mathbf{v}^{(m,l)}_D$, i.e.

$$\mathbf{v}^{(m,l)}_D = \mathbf{G}^{(m)} \mathbf{x}^{(m,l)}_D. \quad (4.10)$$

For brevity, we will discard the subscript $D$ from the notation and the set of $d$ nodes being used for regeneration will be clear from the context. Thus we have,

$$\mathbf{v}^{(m,l)} = \mathbf{G}^{(m)} \mathbf{z}^{(m,l)}. \quad (4.11)$$

All notation is depicted in Figure 4.1.

Throughout the chapter, $e_i$ represents an $\alpha$-length unit vector with 1 in $i^{th}$ position and 0 elsewhere. We say two vectors are \textit{aligned} if they are linearly dependent. We briefly describe the concept of interference alignment in the next section, and discuss its
We next turn our attention to the questions as to whether or not the combination of (a) restriction to repair of systematic nodes and (b) requirement for exact regeneration of the systematic nodes leads to a bound on the parameters \((\alpha, \beta)\) different from the cut-set bound. It is shown below that the same bound on the parameters \((\alpha, \beta)\) appearing in (1.4) still applies. Here we consider \(\beta = 1\). However, the same argument holds for higher values of \(\beta\).

Consider regeneration of the systematic node \(l\), \(1 \leq l \leq k\), by connecting to the \(d\) nodes \(\{m_1, \ldots, m_d\}\). That is, using \(\{v^{(m_1,l)}, \ldots, v^{(m_d,l)}\}\), \(G^{(l)}\) needs to be recovered. This implies that the dimension of the nullspace of the matrix

\[
\begin{bmatrix}
G^{(l)} & v^{(m_1,l)} & \cdots & v^{(m_d,l)}
\end{bmatrix}
\]

should be at least equal to the dimension of \(G^{(l)}\) which is \(\alpha\). But, the MDS property requires that any \(k\) nodes are linearly independent. Hence, the dimension of the nullspace of the matrix

\[
\begin{bmatrix}
G^{(l)} & v^{(m_1,l)} & \cdots & v^{(m_{k-1},l)}
\end{bmatrix}
\]

is zero. This implies that at least \(\alpha\) more columns need to be added to this matrix to make the nullspace of the matrix in equation (4.12) to be of \(\alpha\)-dimensional. Thus we need

\[
d - (k - 1) \geq \alpha .
\]

And to meet the bound, we need \(\alpha = d - k + 1\).
4.2 Interference Alignment in Regenerating Codes

In the distributed storage context, the concept of ‘interference’ comes into picture during exact regeneration of a failed node. Let us consider an example with $B = 4$, $n = 4$, $k = 2$, $d = 3$ and $\beta = 1$, giving $\alpha = B/k = 2$. Let $\{z_1, z_2, z_3, z_4\}$ be the source symbols. Node 1 stores $\{z_1, z_2\}$ and node 2 stores $\{z_3, z_4\}$. Nodes 3 and 4 are non-systematic nodes. Every symbol stored is associated to a global kernel vector having its four components along $z_1, \ldots, z_4$ respectively. By a slight abuse of notation, we will call the $i$th component of a global kernel vector as the component along source symbol $z_i$.

Consider failure of node 1, with nodes 2, 3 and 4 passing vectors $\mathbf{v}^{(2,1)}$, $\mathbf{v}^{(3,1)}$ and $\mathbf{v}^{(4,1)}$ for its regeneration. Since node 2 can only pass some function of $z_3$ and $z_4$, $\mathbf{v}^{(3,1)}$ and $\mathbf{v}^{(4,1)}$ need to provide all the information about $\{z_3, z_4\}$. Since two units of information about $\{z_1, z_2\}$ are required, the components of $\mathbf{v}^{(3,1)}$ and $\mathbf{v}^{(4,1)}$ along $\{z_3, z_4\}$, i.e., $v_3^{(3,1)}$ and $v_4^{(4,1)}$, constitute interference and need to be eliminated. The components along $\{z_1, z_2\}$, i.e., $v_1^{(3,1)}$ and $v_1^{(4,1)}$ constitute the desired component.

The only way to eliminate the interference is using $\mathbf{v}^{(2,1)}$. Since node 2 can pass only one vector, to be able to cancel the interference from $\{z_3, z_4\}$, both $v_3^{(3,1)}$ and $v_4^{(4,1)}$ need to be scalar multiples of each other. In other words, the interference along the node 2 needs to be aligned! This is the interference alignment in the context of regenerating codes. The necessity of interference alignment for any code meeting the cut-set bound on repair bandwidth at the MSR point will be formally proved in Section 4.4.

An explicit code over $\mathbb{F}_5$ for the parameters chosen in the example is shown in Figure 4.2. The figure depicts the actual symbols that are stored by the 4 nodes, as opposed to depicting their global kernels (the two viewpoints are equivalent and we will often switch between them). The exact regeneration of systematic node 1 is also shown, for which nodes 3 and 4 pass the first symbols stored in them, i.e. pass $2z_1 + 2z_2 + z_3$ and $2z_1 + 4z_2 + 2z_3$ respectively. Observe that the interfering component in both these symbols are aligned along $z_3$. Hence node 2 can pass $z_3$ and cancel the interference.

In the context of regenerating codes, interference alignment was first used by Wu and Dimakis in [17] to provide a scheme for exact regeneration at the MSR point. However the authors employed interference alignment to a limited extent, as in their scheme, only
4.3 Construction of the MISER Code

In this section we provide the explicit construction for the MISER code, a systematic, MDS code which achieves the lower bound on the repair bandwidth for exact regeneration of systematic nodes.

First, an illustrative example is provided which explains the key ideas behind the code construction. The general code construction for the parameter set \( n = 2k, \ d = n - 1 \) follows the example. Then, a simple puncturing method is provided to extend this code construction to obtain a family of codes for the parameter set \( n \geq 2k, \ d = n - 1 \). Further, this code is extended to the more general case of any \( n, \ d \geq 2k - 1 \), which however requires the node replacing a failed systematic node to connect to the remaining systematic nodes.

4.3.1 An Example

Let the parameters for the example code be, \( B = 9, \ n = 6, \ k = 3, \ d = 5, \ \beta = 1 \). This gives \( \alpha = B/k = 3 \). Let all symbols belong to the finite field \( \mathbb{F}_7 \).

Design of Node Generator Matrices

As \( k = 3 \), the first three nodes are systematic and store data in uncoded form. Hence

\[
G^{(1)} = \begin{bmatrix} I_3 \\ 0_3 \\ 0_3 \end{bmatrix}, \ G^{(2)} = \begin{bmatrix} 0_3 \\ I_3 \\ 0_3 \end{bmatrix}, \ G^{(3)} = \begin{bmatrix} 0_3 \\ 0_3 \\ I_3 \end{bmatrix}.
\]

(4.15)

The crux of the code construction is the design of the node generator matrices of the non-systematic nodes. Let

\[
\Psi_3 = \begin{bmatrix} \psi_1^{(4)} & \psi_1^{(5)} & \psi_1^{(6)} \\ \psi_2^{(4)} & \psi_2^{(5)} & \psi_2^{(6)} \\ \psi_3^{(4)} & \psi_3^{(5)} & \psi_3^{(6)} \end{bmatrix}.
\]

(4.16)

be a \( 3 \times 3 \) matrix such that any of its sub-matrix is full rank. A Cauchy matrix [33] is one such matrix and we choose \( \Psi_3 \) to be a Cauchy matrix for our construction.
4.3 Construction of the MISER Code

Definition 4.3.1 (Cauchy matrix) A Cauchy matrix is a matrix with its $\frac{1}{x_i - y_j}$ element as $\frac{1}{x_i - y_j}$ where $\{x_i\} \cup \{y_j\}$ is an injective sequence, i.e., a sequence with no repeated elements.

Let the generator matrix of non-systematic node $m$ ($m = 4, 5, 6$) be

$$
G^{(m)} = \begin{bmatrix}
2\psi_1^{(m)} & 0 & 0 \\
2\psi_2^{(m)} & \psi_1^{(m)} & 0 \\
2\psi_3^{(m)} & 0 & \psi_1^{(m)} \\
\psi_2^{(m)} & 2\psi_1^{(m)} & 0 \\
0 & 2\psi_2^{(m)} & 0 \\
0 & 2\psi_3^{(m)} & \psi_2^{(m)} \\
\psi_3^{(m)} & 0 & 2\psi_1^{(m)} \\
0 & \psi_3^{(m)} & 2\psi_2^{(m)} \\
0 & 0 & 2\psi_3^{(m)}
\end{bmatrix}
$$

(4.17)

where the non-zero entries of the $i^{th}$ submatrix are restricted to the diagonal and the $i^{th}$ column, $1 \leq i \leq 3$. We now show that this choice of node generator matrices makes the code MDS and also minimizes the repair bandwidth for the exact regeneration of systematic nodes.

Reconstruction (MDS property)

For the reconstruction property to be satisfied, a data collector downloading all symbols stored in any three nodes should be able to recover all the nine source symbols. That is, the $9 \times 9$ matrix formed by columnwise juxtaposing any three node generator matrices, need to be non-singular. We consider the different set nodes that the data collector can connect to, and give decoding procedures for each.

(a) Three systematic nodes: When data collector connects to all three systematic nodes, it obtains all the source symbols in uncoded form and hence reconstruction is trivially satisfied.

(b) Two systematic and one non-systematic nodes: Suppose the data collector connects to systematic nodes 2 and 3, and non-systematic node 4. It obtains all the symbols stored in nodes 2 and 3 in uncoded form, and subtracts out their contribution from the coded symbols in node 4. It is left to decode the source symbols $z_i$ which are encoded using the following matrix

$$
G_1^{(4)} = \begin{bmatrix}
2\psi_1^{(4)} & 0 & 0 \\
2\psi_2^{(4)} & \psi_2^{(4)} & 0 \\
2\psi_3^{(4)} & 0 & \psi_3^{(4)}
\end{bmatrix}
$$

(4.18)
This diagonal matrix is full rank since the elements of a Cauchy matrix are non-zero, and hence the symbols $z^t_1$ can be easily obtained.

(c) All three non-systematic nodes: Now let's consider the case of a data collector connecting to all three non-systematic nodes. Let $C_1$ be the matrix formed by columnwise juxtaposing the generator matrices of these three nodes.

Claim 1 The data collector can recover all the source symbols encoded using the matrix $C_1$.

Proof In $C_1$, group the $i^{th}$ ($i = 1, 2, 3$) columns of all the three nodes together to obtain the matrix $C_2$ as

$$C_2 = \begin{bmatrix}
2\psi_1^{(4)} & 2\psi_1^{(5)} & 2\psi_1^{(6)} & 0 & 0 & 0 & 0 & 0 & 0 \\
2\psi_2^{(4)} & 2\psi_2^{(5)} & 2\psi_2^{(6)} & \psi_1^{(4)} & \psi_1^{(5)} & \psi_1^{(6)} & 0 & 0 & 0 \\
2\psi_3^{(4)} & 2\psi_3^{(5)} & 2\psi_3^{(6)} & 0 & 0 & 0 & \psi_1^{(4)} & \psi_1^{(5)} & \psi_1^{(6)} \\
\psi_2^{(4)} & \psi_2^{(5)} & \psi_2^{(6)} & 2\psi_1^{(4)} & 2\psi_1^{(5)} & 2\psi_1^{(6)} & 0 & 0 & 0 \\
0 & 0 & 0 & 2\psi_2^{(4)} & 2\psi_2^{(5)} & 2\psi_2^{(6)} & 0 & 0 & 0 \\
0 & 0 & 0 & 2\psi_3^{(4)} & 2\psi_3^{(5)} & 2\psi_3^{(6)} & \psi_2^{(4)} & \psi_2^{(5)} & \psi_2^{(6)} \\
\psi_3^{(4)} & \psi_3^{(5)} & \psi_3^{(6)} & 0 & 0 & 0 & 2\psi_1^{(4)} & 2\psi_1^{(5)} & 2\psi_1^{(6)} \\
0 & 0 & 0 & \psi_3^{(4)} & \psi_3^{(5)} & \psi_3^{(6)} & 2\psi_2^{(4)} & 2\psi_2^{(5)} & 2\psi_2^{(6)} \\
0 & 0 & 0 & 0 & \psi_3^{(4)} & \psi_3^{(5)} & \psi_3^{(6)} & 2\psi_3^{(4)} & 2\psi_3^{(5)} & 2\psi_3^{(6)}
\end{bmatrix}$$

Note that, interchanging columns is equivalent to interchanging corresponding coded symbols. Though this step is not necessary, it helps to explain the decoding procedure better.
Multiply the 3 groups of 3 symbols(columns) each by $\Psi^{-1}_3$ to get a matrix $C_3$ given by

$$
C_3 = C_2 \begin{bmatrix}
\Psi^{-1}_3 & 0_3 & 0_3 \\
0_3 & \Psi^{-1}_3 & 0_3 \\
0_3 & 0_3 & \Psi^{-1}_3
\end{bmatrix}
$$

(4.20)

\[
\begin{bmatrix}
2 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 0 & 1 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
1 & 0 & 0
\end{bmatrix}
\]

(4.21)

Symbols 1, 5 and 9 are now available to the data collector, and their contribution can be subtracted from the remaining symbols to obtain

$$
C_4 = \begin{bmatrix}
2 & 0 & 1 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 1 & 0 \\
1 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 2
\end{bmatrix}
$$

(4.22)

Simple arithmetic operations can now be performed on this set of symbols to recover all the remaining source symbols, as $C_4$ is non-singular.

(d) **One systematic and two non-systematic nodes:** Suppose the data collector connects to systematic node 1 and non-systematic nodes 4 and 5. All symbols of node 1, i.e. $z_1$, are available to the data collector and their contribution can be subtracted from all the coded symbols. The data-collector is left to decode the symbols $z_2$ and $z_3$ which are encoded using the matrix,

$$
B_1 = \begin{bmatrix}
G_2^{(4)} & G_2^{(5)} \\
G_3^{(4)} & G_3^{(5)}
\end{bmatrix}
$$

(4.23)

**Claim 2** The data collector can recover the source symbols $z_2$ and $z_3$ encoded by the matrix $B_1$.

**Proof** We first interchange certain columns which will make the task simpler. For $i = 2, 3, 1$ (in this order), group the $i^{th}$ columns of the two non-systematic nodes together to
4.3 Construction of the MISER Code

give matrix

\[
B_2 = \begin{bmatrix}
2\psi_1^{(4)} & 2\psi_1^{(5)} & 0 & 0 & \psi_2^{(4)} & \psi_2^{(5)} \\
2\psi_2^{(4)} & 2\psi_2^{(5)} & 0 & 0 & 0 & 0 \\
2\psi_3^{(4)} & 2\psi_3^{(5)} & \psi_2^{(4)} & \psi_2^{(5)} & 0 & 0 \\
0 & 0 & 2\psi_1^{(4)} & 2\psi_1^{(5)} & \psi_3^{(4)} & \psi_3^{(5)} \\
\psi_3^{(4)} & \psi_3^{(5)} & 2\psi_2^{(4)} & 2\psi_2^{(5)} & 0 & 0 \\
0 & 0 & 2\psi_3^{(4)} & 2\psi_3^{(5)} & 0 & 0 \\
\end{bmatrix}
\]

Let

\[
\Psi_2 = \begin{bmatrix}
\psi_2^{(4)} \\
\psi_2^{(5)} \\
\end{bmatrix}
\]

(4.25)

\(\Psi_2\) is a submatrix of the Cauchy matrix \(\Psi_3\) and hence is invertible. Multiply the last two symbols (columns) by \(\Psi_2^{-1}\) to obtain

\[
B_3 = \begin{bmatrix}
2\psi_1^{(4)} & 2\psi_1^{(5)} & 0 & 0 & 1 & 0 \\
2\psi_2^{(4)} & 2\psi_2^{(5)} & 0 & 0 & 0 & 0 \\
2\psi_3^{(4)} & 2\psi_3^{(5)} & \psi_2^{(4)} & \psi_2^{(5)} & 0 & 0 \\
0 & 0 & 2\psi_1^{(4)} & 2\psi_1^{(5)} & 0 & 1 \\
\psi_3^{(4)} & \psi_3^{(5)} & 2\psi_2^{(4)} & 2\psi_2^{(5)} & 0 & 0 \\
0 & 0 & 2\psi_3^{(4)} & 2\psi_3^{(5)} & 0 & 0 \\
\end{bmatrix}
\]

(4.26)

The last two symbols are now available to the data collector and their contribution can be subtracted out from the rest of the symbols to get,

\[
B_4 = \begin{bmatrix}
2\psi_2^{(4)} & 2\psi_2^{(5)} & 0 & 0 \\
2\psi_3^{(4)} & 2\psi_3^{(5)} & \psi_2^{(4)} & \psi_2^{(5)} \\
\psi_3^{(4)} & \psi_3^{(5)} & 2\psi_2^{(4)} & 2\psi_2^{(5)} \\
0 & 0 & 2\psi_3^{(4)} & 2\psi_3^{(5)} \\
\end{bmatrix}
\]

(4.27)

This matrix is equivalent to the reconstruction matrix of a system with \(k = 2\) with a data collector connecting to two non-systematic nodes, and on the lines similar to the previous case, can be shown to be invertible.

---

**Exact Regeneration of Systematic Nodes**

Suppose node 1 fails. Let each non-systematic node pass its first symbol i.e. the first column of their generator matrices for the regeneration of node 1. Thus, from nodes 4, 5,
and 6, the new node gets

\[
\mathbf{v}^{(4,1)} = \begin{bmatrix}
2\psi_1^{(4)} \\
2\psi_2^{(4)} \\
2\psi_3^{(4)} \\
\psi_2^{(4)} \\
0 \\
0 \\
\psi_3^{(4)} \\
0 \\
0 \\
0
\end{bmatrix}, \quad \mathbf{v}^{(5,1)} = \begin{bmatrix}
2\psi_1^{(5)} \\
2\psi_2^{(5)} \\
2\psi_3^{(5)} \\
\psi_2^{(5)} \\
0 \\
0 \\
\psi_3^{(5)} \\
0 \\
0 \\
0
\end{bmatrix}, \quad \mathbf{v}^{(6,1)} = \begin{bmatrix}
2\psi_1^{(6)} \\
2\psi_2^{(6)} \\
2\psi_3^{(6)} \\
\psi_2^{(6)} \\
0 \\
0 \\
\psi_3^{(6)} \\
0 \\
0 \\
0
\end{bmatrix}.
\] (4.28)

In these vectors, observe that the component along node 1 are scaled columns of the Cauchy matrix \(\Psi_3\). The components along other existing systematic nodes are all aligned along the vector \([1 \, 0 \, 0]\). Hence all the interfering components are aligned in a single dimension.

Now, nodes 2 and 3 pass the following vectors

\[
\mathbf{v}^{(2,1)} = \begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
1 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}, \quad \mathbf{v}^{(3,1)} = \begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
1 \\
0 \\
0
\end{bmatrix}.
\] (4.29)

As a result of the interference alignment, \(\mathbf{v}^{(2,1)}\) and \(\mathbf{v}^{(3,1)}\) can subtract out all the interfering components from \(\mathbf{v}^{(4,1)}\), \(\mathbf{v}^{(5,1)}\) and \(\mathbf{v}^{(6,1)}\). The new node 1 is left with three symbols encoded using the matrix

\[
\begin{bmatrix}
2\Psi_3 \\
0_3 \\
0_3
\end{bmatrix}.
\]

The desired component, i.e., the component along \(z_1\) is a scaled Cauchy matrix. The new node 1 can operate on the three received symbols with the matrix \(\frac{1}{2}\Psi_3^{-1}\) and recover all the three source symbols that node 1 stored prior to its failure. Thus node 1 is exactly regenerated.

On similar lines, when nodes 2 or 3 fail, non-systematic nodes pass the 2\textsuperscript{nd} or 3\textsuperscript{rd} columns of their generator matrices respectively. The design of generator matrices for non-systematic nodes is such that interference alignment holds during regeneration of every
systematic node, and hence exact regeneration of all systematic nodes can be achieved.

4.3.2 The MISER Code for \( n = 2k, \ d = n - 1 \)

The parameter range under consideration, along with \( d = \alpha + k - 1 \) which needs to be satisfied to meet the lower bound on the repair bandwidth gives

\[
k = \alpha . \tag{4.30}
\]

This relation plays a key role in the code construction as this allows each non-systematic node to reserve \( k \) distinct symbols (i.e. symbols having linearly independent global kernels) for the regeneration of the \( k \) systematic nodes. Recall that, in the example code in Section 4.3.1, \( d = 2k - 1 = 5 \) and \( k = \alpha = 3 \). The construction of the MISER code in the parameter regime \( n = 2k, \ d = n - 1 \) follows closely on the lines of the example code.

Design of Node Generator Matrices

The first \( k \) nodes are systematic and store the source symbols in uncoded form. Hence, for systematic node \( m \), and \( l = 1, \ldots, k \),

\[
G^{(m)}_l = \begin{cases} I_{\alpha} & \text{if } l = m \\ 0_{\alpha} & \text{if } l \neq m \end{cases} \tag{4.31}
\]

Let \( \Psi \) be an \( \alpha \times (n - k) \) matrix with elements drawn from \( \mathbb{F}_q \) such that any submatrix of \( \Psi \) is full rank. Let

\[
\Psi = \begin{bmatrix} \psi^{(k+1)} & \psi^{(k+2)} & \cdots & \psi^{(n)} \end{bmatrix} \tag{4.32}
\]

where

\[
\psi^{(i)} = \begin{bmatrix} \psi_1^{(i)} \\ \vdots \\ \psi_{\alpha}^{(i)} \end{bmatrix} \quad i = k + 1, \ldots, n \tag{4.33}
\]

We choose \( \Psi \) to be a Cauchy matrix, and the minimum field size required for the construction of this Cauchy matrix is

\[
q \geq \alpha + n - k. \tag{4.34}
\]

Note that since \( n - k \geq \alpha \geq 2 \), we have \( q \geq 4 \).

Now we come to the crux of the code, which is the design of the generator matrices
for the non-systematic nodes. For \( m = k + 1, \ldots, n \), \( i, j = 1, \ldots, \alpha \), choose

\[
\mathbf{g}_{ij}^{(m)} = \begin{cases} 
\epsilon \psi_{ij}^{(m)} & \text{if } i = j \\
\psi_i \epsilon_j & \text{if } i \neq j 
\end{cases}
\]  

where \( \epsilon \) is an element from \( \mathbb{F}_q \) such that \( \epsilon \neq 0 \) and \( \epsilon^2 \neq 1 \). Note that there always exists such a value provided \( q \geq 4 \), which is true in this case.

**Reconstruction**

For reconstruction to be satisfied, a data collector downloading all symbols stored in any \( k \) nodes should be able to recover the \( B \) source symbols. For this, we need the \( B \times B \) matrix formed by columnwise juxtaposing any \( k \) node generator matrices to be non-singular.

If the data collector connects to the \( k \) systematic nodes, then reconstruction is trivially satisfied. Consider the case when a data collector connects to \( p \) non-systematic nodes, and \( k - p \) systematic nodes, for some \( 1 \leq p \leq k \). Let \( \delta_1, \ldots, \delta_p \) be the \( p \) non-systematic nodes and \( \omega_1, \ldots, \omega_{k-p} (\omega_1 < \ldots < \omega_{k-p}) \) be the \( k - p \) systematic nodes to which the data collector connects, and let \( \Omega_1, \ldots, \Omega_p (\Omega_1 < \ldots < \Omega_p) \) be the \( p \) systematic nodes to which data collector does not connect.

The data collector can obtain the symbols \( z_{\omega_1}, \ldots, z_{\omega_{k-p}} \) from the systematic nodes it connects to, and subtract out their contribution from the coded symbols. The data collector now has to recover the symbols \( \bar{z}_{\Omega_1}, \ldots, \bar{z}_{\Omega_p} \) which are encoded using the following \( p\alpha \times p\alpha \) matrix

\[
R = \begin{bmatrix}
G^{(\delta_1)} & G^{(\delta_2)} & \cdots & G^{(\delta_p)} \\
G^{(\delta_1)}_{\Omega_1} & G^{(\delta_2)}_{\Omega_1} & \cdots & G^{(\delta_p)}_{\Omega_1} \\
\vdots & \vdots & \ddots & \vdots \\
G^{(\delta_1)}_{\Omega_p} & G^{(\delta_2)}_{\Omega_p} & \cdots & G^{(\delta_p)}_{\Omega_p}
\end{bmatrix}
\]

(4.36)

**Theorem 4.3.2** The data collector can decode the symbols \( \bar{z}_{\Omega_1}, \ldots, \bar{z}_{\Omega_p} \) encoded using the matrix \( R \).

**Proof** See Appendix C. The steps followed in the proof are similar to the ones used in the example. ■
4.3 Construction of the MISER Code

Exact Regeneration of Systematic Nodes

Consider regeneration of systematic node $\hat{l}$, $1 \leq \hat{l} \leq k$. Each non-systematic node passes the $\hat{l}^{th}$ column of its generator matrix, i.e.,

$$
\mathbf{v}^{(m,\hat{l})} = \begin{bmatrix}
g^{(m)}_{1,\hat{l}} \\
g^{(m)}_{2,\hat{l}} \\
\vdots \\
g^{(m)}_{\alpha,\hat{l}}
\end{bmatrix}
$$

(4.37)

Observe that the choice of $g^{(m)}_{i,\hat{l}}$ as in equation (4.35) makes all the interfering components in $\{\mathbf{v}^{(k+1,\hat{l})}, \ldots, \mathbf{v}^{(n,\hat{l})}\}$ align along $e_{\hat{l}}$. Each of the remaining systematic nodes $l$ ($l = 1, \ldots, k$, $l \neq \hat{l}$) pass

$$
\mathbf{v}^{(l,\hat{l})} = \begin{bmatrix}
0 \\
0 \\
\vdots \\
0 \\
e_{\hat{l}} \\
0 \\
\vdots \\
0
\end{bmatrix}
$$

(4.38)

with $e_{\hat{l}}$ as the $l^{th}$ component.

Due to the interference alignment, $\mathbf{v}^{(l,\hat{l})}$ can subtract out all the interference along $z_{\hat{l}}$. Thus, all the interference is canceled out.

Along the symbols stored in systematic node $\hat{l}$, i.e., in the desired component we are left with

$$
\left[ g^{(k+1)}_{\hat{l},\hat{l}} \cdots g^{(n)}_{\hat{l},\hat{l}} \right]
$$

(4.39)

By the choice of $g^{(m)}_{i,\hat{l}}$ as given in equation (4.35), following are the vectors along the desired component

$$
\epsilon \left[ \psi^{(k+1)} \cdots \psi^{(n)} \right] = \epsilon \Psi
$$

(4.40)

In other words, the data collector obtains the symbols stored in failed node $\hat{l}$ coded using the Cauchy matrix $\Psi$. These symbols can be operated upon by $\frac{1}{\epsilon} \Psi^{-1}$ to obtain all the symbols that were stored in the failed systematic node $\hat{l}$. Thus, the failed systematic node $\hat{l}$ is exactly regenerated.
4.3 Construction of the MISER Code

Figure 4.3: An illustration of going from the case $d = 2k + c$ to $d = 2k + c + 1$. Node $k + 1$ stores all zeros, and any data collector or a new node is always assumed to connect to this node.

4.3.3 The MISER Code for $n \geq 2k$, $d = n - 1$

The parameter regime $n \geq 2k$, $d = n - 1$ along with $d = \alpha + k - 1$ gives

\[ k \leq \alpha . \]  

The following theorem gives a general result, which in this context says that codes for the parameters $n \geq 2k$, $d = n - 1$ can be obtained from codes for $n = 2k$, $d = n - 1$ via simple puncturing operations.

**Theorem 4.3.3** Given a family of linear exact regenerating codes for all parameters satisfying $n = d + c_1$ and $d = 2k + c$ for some constants $c_1$ and $c$, linear exact regenerating codes can be constructed for parameters $\hat{n}$, $\hat{d}$ and $\hat{k}$ satisfying $\hat{n} = \hat{d} + c_1$ and $\hat{d} = 2\hat{k} + \hat{c}$ for any $\hat{c} \geq c$.

**Proof** From the hypothesis, construct a code $C$ for the parameters $(n = \hat{n} + \hat{c} - c, \ d = \hat{d} + \hat{c} - c, \ k = \hat{k} + \hat{c} - c)$. Consider a set of $(\hat{c} - c)$ systematic nodes, and assume that the source symbols stored in these systematic nodes are always zero. Call these nodes the ‘nullified’ systematic nodes. The remaining $\hat{n}$ nodes form the desired code $\hat{C}$, with the source symbols as those stored in the remaining $\hat{k}$ systematic nodes.

During any reconstruction (or regeneration) operation in $\hat{C}$, one can consider an equivalent operation in $C$ where the data collector (or the new node replacing the failed node) connects to the $(\hat{c} - c)$ nullified systematic nodes as well. This is illustrated in Figure 4.3 for $\hat{c} - c = 1$. 

4.3 Construction of the MISER Code

Since the data stored in the nullified systematic nodes is globally known to be zero, the linearity of the code implies that all the symbols that these nodes pass for reconstruction (or regeneration) are zero. And since code $C$ can perform reconstruction and regeneration, so can the code $\hat{C}$.

Thus, if the $k\alpha \times n\alpha$ generator matrix for code $C$ is

$$[G^{(1)} \ G^{(2)} \ \cdots \ G^{(n)}]$$

then the $\hat{k}\alpha \times \hat{n}\alpha$ generator matrix for the desired code $\hat{C}$ is

$$
\begin{bmatrix}
G^{(1)}_1 & \cdots & G^{(k)}_1 & G^{(k+1)}_1 & \cdots & G^{(n)}_1 \\
\vdots & \ddots & \vdots & \ddots & \ddots & \vdots \\
G^{(1)}_k & \cdots & G^{(k)}_k & G^{(k+1)}_k & \cdots & G^{(n)}_k
\end{bmatrix}
$$

(4.43)

4.3.4 Extension to the case $2k - 1 \leq d \leq n - 1$

In this section, we present a simple extension of the MISER code to the case when $2k - 1 \leq d \leq n - 1$, under the constraint that a new node replacing a failed systematic node includes all the remaining systematic nodes in the set of $d$ nodes that it connects to.

The following theorem shows that the code provided in Section 4.3.2 for $n = 2k$, $d = n - 1$ supports the case of $d = 2k - 1, d \leq n - 1$ as long as the constraint is met. From here, extension to the case $d \geq 2k - 1, d \leq n - 1$ is straightforward via Theorem 4.3.3.

**Theorem 4.3.4** For $d = 2k - 1 \leq n - 1$, the code defined by the node generator matrices in equations (4.31) and (4.35), achieves reconstruction and optimal exact regeneration of systematic nodes, provided the new node connects to all the remaining systematic nodes.

**Proof** *Reconstruction:* The reconstruction property follows directly from the reconstruction property of the original code.

**Exact regeneration of systematic nodes** New node replacing a failed systematic node connects to the $k - 1$ existing systematic nodes and any $\alpha$ non-systematic nodes (as meeting the cut-set bound requires $d = k - 1 + \alpha$). Consider a distributed storage system having only these $d$ nodes along with the failed node as its $n$ nodes. Such a system has $d = n - 1$ and is identical to the system described in Section 4.3.2. Hence exact regeneration of systematic nodes meeting the cut-set bound is guaranteed.
4.3.5 Analysis of the Misher Code

**Uniqueness:** Let us assume that the $k$ global kernels respectively passed by a non-systematic node for the regeneration of the $k$ systematic nodes are linearly independent. Under this assumption, it can be shown that the structure of the Misher code (namely that the $i^{th}$ component of the generator matrix of any non-systematic node is a superposition of a diagonal matrix and a matrix with only its $i^{th}$ column being non-zero) is essentially unique up to linear transformations that either leave invariant the subspace of global kernels stored within a node, or else that correspond to a re-labeling of the message symbols. For lack of space and time, we do not present the proof here.

**Field size required:** The constraint on the field size comes due to construction of the $\alpha \times (n - k)$ matrix $\Psi$ having all sub-matrices full rank. For our constructions, since $\Psi$ is chosen to be a Cauchy matrix, any field of size $n + d - 2k + 1 (< 2n)$ or higher suffices.

**Complexity of reconstruction:** The complexity analysis is provided for the case $n = 2k$, $d = n - 1$, other cases follow on the similar lines. A data collector connecting to the $k$ systematic nodes can recover all the data without any additional processing. A data collector connecting to some $k$ arbitrary nodes has to multiply the inverse of a $k \times k$ Cauchy matrix with $k$ vectors.

**Complexity of exact regeneration of systematic nodes:** Any node participating in the exact regeneration of systematic node $i$, simply passes its $i^{th}$ symbol, without any processing. The new node replacing the failed node has to multiply the inverse of an $(\alpha \times \alpha)$ Cauchy matrix with an $\alpha$ length vector and then perform $k - 1$ subtractions for interference cancellation.

4.4 Non-Existence for $d < 2k - 3$

In this section, we show that for $d < 2k - 3$, there exist no linear codes achieving the cut-set bound on the repair bandwidth with $\beta = 1$. In fact, we show that even for the case when exact regeneration of only the systematic nodes is desired, the cut-set bound cannot be achieved. This proves that, indeed there is a penalty in repair bandwidth if we insist on *exact* regeneration of the failed nodes, as opposed to *functional* regeneration, in this parameter regime. Note that since $d = \alpha + k - 1$, this parameter set corresponds to

$$\alpha < k - 2. \quad (4.44)$$

We first derive necessary properties for any linear exact regenerating code. Specifically, we prove that interference alignment is, in fact, necessary for exact regeneration of
systematic nodes. This establishes the basic structure of linear exact regenerating codes.

4.4.1 Necessary Properties

Theorem 4.4.1 In the non-systematic node generator matrices, component along the symbols stored in any systematic node must be of full rank, i.e., for any $m \in \{k+1, \ldots, n\}$, $G^{(m)}_i$ must be non-singular $\forall l \in \{1, \ldots, k\}$.

Proof Consider the case of a data collector connecting to all systematic nodes other than node $l$, and non-systematic node $m$. The data collector recovers $(k - 1)\alpha$ source symbols from the $k - 1$ systematic nodes, and can subtract the contribution of these symbols from the remaining $\alpha$ coded symbols obtained from the non-systematic nodes. Now, the data collector is left with the symbols $z_lG^{(m)}_i$, using which it needs to recover the remaining $\alpha$ source symbols $z_l$ (the other $k - 1$ systematic nodes cannot provide any information about $z_l$). This is possible only if $G^{(m)}_i$ is non-singular. ■

For the rest of the necessary properties, dealing with only a subset of the possibilities of regeneration suffices. Consider the case when the new node replacing a failed systematic node connects to the $k - 1$ existing systematic nodes $n_1, \ldots, n_{k-1}$, and $\alpha$ non-systematic nodes $m_1, \ldots, m_\alpha$.

Theorem 4.4.2 In the vectors passed by the $\alpha$ non-systematic nodes, participating in the regeneration of systematic node $l$, the components along the symbols stored in the systematic node $l$ must be linearly independent, i.e.,

\[
\begin{bmatrix}
  v^{(m_1,l)}_1 & v^{(m_1,l)}_2 & \cdots & v^{(m_\alpha,l)}_1 \\
  v^{(m_1,l)}_2 & v^{(m_1,l)}_3 & \cdots & v^{(m_\alpha,l)}_2 \\
  \vdots & \vdots & \ddots & \vdots \\
  v^{(m_1,l)}_\alpha & v^{(m_1,l)}_{\alpha+1} & \cdots & v^{(m_\alpha,l)}_\alpha \\
\end{bmatrix}
\]

must be of full rank.

Proof Consider exact regeneration of systematic node $l$. Let

\[
V = \begin{bmatrix}
  v^{(n_1,l)} & \cdots & v^{(n_{k-1},l)} & v^{(m_1,l)} & \cdots & v^{(m_\alpha,l)} \\
\end{bmatrix}

= \begin{bmatrix}
  V_1 \\
  \vdots \\
  V_k
\end{bmatrix}
\]

(4.45)

where for $i = 1, \ldots, k$,

\[
V_i = \begin{bmatrix}
  v^{(n_1,l)}_i & \cdots & v^{(n_{k-1},l)}_i & v^{(m_1,l)}_i & \cdots & v^{(m_\alpha,l)}_i
\end{bmatrix}
\]

is an $\alpha \times d$ matrix representing the component of $V$ along $z_i$. 
4.4 Non-Existence for $d < 2k - 3$

For exact regeneration of systematic node $l$, we need an $d \times \alpha$ matrix $Y$ such that

$$VY = G^{(l)}. \quad (4.46)$$

Since $G^{(l)}_i = I_\alpha$, we need

$$\text{rank}(V_i Y) = \alpha. \quad (4.47)$$

For the $k - 1$ systematic nodes $\{n_1, \ldots, n_{k-1}\}$,

$$v_i^{(n_i,l)} = 0 \quad i = 1, \ldots, k - 1 \quad (4.48)$$

Thus the first $k - 1$ columns of $V_i$ are all zeros. Hence, the remaining $\alpha$ columns of $V_i$,

$$\begin{bmatrix}
  v_i^{(m_1,l)} & v_i^{(m_2,l)} & \cdots & v_i^{(m_\alpha,l)}
\end{bmatrix}$$

must be linearly independent to satisfy the rank condition in equation (4.47).

**Corollary 4.4.3** The last $\alpha$ rows of $Y$ should be linearly independent.

**Proof** From equation (4.48), the first $k - 1$ columns of $V_i$ are zeros. Hence to make $V_i Y$ full rank, we need the last $d - (k - 1) = \alpha$ rows of $Y$ to be linearly independent.

**Theorem 4.4.4** (Necessity of Interference Alignment) For exact regeneration of a systematic node $l$, and for any $\hat{l} \in \{1, \ldots, k\}$, $\hat{l} \neq l$, the vectors

$$\{v_i^{(m_1,l)} \hat{l}, \ldots, v_i^{(m_\alpha,l)} \hat{l}\}$$

must be aligned.

**Proof** Consider the exact regeneration of systematic node $l$. Using the same notation as in Theorem 4.4.2, since

$$G^{(l)}_i = 0_\alpha \quad (4.49)$$

we need

$$V_i Y = 0_\alpha. \quad (4.50)$$

Also, since

$$v_i^{(n_i)} = 0 \quad \forall n_i \neq \hat{l}, \quad (4.51)$$

$k - 2$ columns out of the first $k - 1$ columns of $V_i$ will be zero. Hence, the number of non-zero columns in $V_i$ is at most $d - (k - 2) = \alpha + 1$.

Remove the $k - 2$ zero columns in $V_i$ and denote the resultant submatrix by $\tilde{V}_i$,

$$\tilde{V}_i = \begin{bmatrix}
  v_i^{(l,l)} & v_i^{(m_1,l)} & \cdots & v_i^{(m_\alpha,l)}
\end{bmatrix} \quad (4.52)$$
4.4 Non-Existence for $d < 2k - 3$

Remove the corresponding $k - 2$ rows in $Y$ and denote the resultant $\alpha + 1 \times \alpha$ submatrix by $\tilde{Y}$. Thus, we need

$$\tilde{V}_l\tilde{Y} = 0_\alpha \quad (4.53)$$

From Corollary 4.4.3

$$\text{rank}(\tilde{Y}) = \alpha \quad (4.54)$$

which forces

$$\text{rank}(\tilde{V}_l) \leq 1 \quad (4.55)$$

and this proves that result. \(\blacksquare\)

**Remark 4.4.5** In the case of $\beta > 1$, the interfering components in the $\beta$-dimensional subspaces passed by the $\alpha$ non-systematic nodes need to be aligned along a $\beta$-dimensional subspace.

### 4.4.2 Structure of the Code

The necessary properties derived will now be put to use in establishing the structure of any linear exact regenerating code satisfying the desired properties for the set of parameters under consideration.

**Equivalence of Codes:**

Let $W_i$ denote the $i$th nodal subspace, i.e., the column-space of $G^{(i)}$. A little thought will show that the distributed storage code $\mathcal{C}$ is a regenerating code iff

(i) for every subset $\{i_j \mid 1 \leq j \leq k\}$,

$$\dim(W_{i_1} + W_{i_2} + \cdots + W_{i_k}) = B$$

and

(ii) for every subset $\{i_j \mid 1 \leq j \leq (d + 1)\}$, the subspaces $\{W_{i_j} \}_{j=1}^d$ contain a vector $h_j$ such that

$$W_{i_{d+1}} \subseteq \langle h_{i_1}, h_{i_2}, \cdots, h_{i_d} \rangle.$$

We can thus define two regenerating codes to be equivalent if the associated subspaces $\{W_i\}_{i=1}^n$ are identical. It is also clear that two codes are equivalent if one can be obtained from the other through a non-singular transformation of the message symbols. With these
two observations, it follows that the generator matrices

\[ G, \quad \text{and} \quad AG \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix} \]

where the \((B \times B)\) matrix \(A\) and the \((\alpha \times \alpha)\) matrices \(\{B_i\}_{i=1}^n\) are all non-singular, define equivalent regenerating codes.

In other words, two codes \(C\) and \(C'\) are equivalent if the code \(C'\) can be represented in terms of \(C\) by (i) changing the basis of the vector space generated by the source symbols, and (ii) changing the basis of the column spaces of the generator matrices of the storage nodes.

Using this concept of equivalent codes, we first present two Lemmas which establish the structure of the non-systematic node generator matrices of a code (assuming that it exists) which achieves the cut-set bound in the parameter regime of interest. We consider only a subset of possible regeneration scenarios, where any failed systematic node connects to the existing systematic nodes along with the non-systematic nodes \(k + 1, \ldots, k + \alpha\). We will focus only on these \(d + 1\) nodes consisting of the \(k\) systematic nodes and the first \(\alpha\) non-systematic nodes, and hence it suffices to consider \(n = d + 1 = k + \alpha\).

Since the matrices \(G_i^{(m)}\) are non-singular (by Theorem 4.4.1), they can be represented as

\[ G_i^{(m)} = H_i^{(m)} \Lambda_i^{(m)}, \quad m = k + 1, \ldots, n, \ i = 1, \ldots, k \]  \hspace{1cm} (4.56)

where \(\Lambda_i^{(m)} = diag\{\lambda_i^{(m)}, \ldots, \lambda_i^{(m)}\}\) is a non-singular \(\alpha \times \alpha\) diagonal matrix and

\[ H_i^{(m)} = \begin{bmatrix} h_{i,1}^{(m)} \\ h_{i,2}^{(m)} \\ \vdots \\ h_{i,\alpha}^{(m)} \end{bmatrix} \]  \hspace{1cm} (4.57)

is a non-singular \(\alpha \times \alpha\) matrix, and \(h_{i,j}^{(m)}\) is an \(\alpha\)-length column vector.

**Lemma 4.4.6** If there exists an exact regenerating code for \(d < 2k - 1\), then there exists an equivalent code with the property

\[ h_{i,j}^{(m)} = h_{i,j}, \quad \text{for} \ i = 1, \ldots, k, \ j = 1, \ldots, \alpha, \ j \neq i \]  \hspace{1cm} (4.58)

for all non-systematic nodes \(m \in \{k + 1, \ldots, n\}\).

**Proof** Suppose there exists an exact regenerating code for some \(d < 2k - 1\) (which corresponds to \(\alpha < k\)). To prove the lemma, we take the following approach. Using properties which are necessary for exact regeneration of systematic nodes, we obtain the vectors that the non-systematic nodes pass for regeneration of the systematic nodes. Since these vectors lie in the column spaces of the respective non-systematic node generator
matrices, these vectors define the structure of the generator matrices of the non-systematic nodes.

By induction we prove that the generator matrix of any non-systematic node \( m \) (\( \in \{k + 1, \ldots, n\} \)) can be written as:

\[
G^{(m)} = 
\begin{bmatrix}
\lambda_{1,1}^{(m)} h_{1,1}^{(m)} & \cdots & \lambda_{1,p-1}^{(m)} h_{1,p-1}^{(m)} & \lambda_{1,p}^{(m)} h_{1,p}^{(m)} & \cdots & \lambda_{1,\alpha}^{(m)} h_{1,\alpha}^{(m)} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\lambda_{p-1,1}^{(m)} h_{p-1,1}^{(m)} & \cdots & \lambda_{p-1,p-1}^{(m)} h_{p-1,p-1}^{(m)} & \lambda_{p-1,p}^{(m)} h_{p-1,p}^{(m)} & \cdots & \lambda_{p-1,\alpha}^{(m)} h_{p-1,\alpha}^{(m)} \\
\lambda_{p,1}^{(m)} h_{p,1}^{(m)} & \cdots & \lambda_{p,p-1}^{(m)} h_{p,p-1}^{(m)} & \lambda_{p,p}^{(m)} h_{p,p}^{(m)} & \cdots & \lambda_{p,\alpha}^{(m)} h_{p,\alpha}^{(m)} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\lambda_{\alpha-1,1}^{(m)} h_{\alpha-1,1}^{(m)} & \cdots & \lambda_{\alpha-1,p-1}^{(m)} h_{\alpha-1,p-1}^{(m)} & \lambda_{\alpha-1,p}^{(m)} h_{\alpha-1,p}^{(m)} & \cdots & \lambda_{\alpha-1,\alpha}^{(m)} h_{\alpha-1,\alpha}^{(m)} \\
\lambda_{\alpha,1}^{(m)} h_{\alpha,1}^{(m)} & \cdots & \lambda_{\alpha,p-1}^{(m)} h_{\alpha,p-1}^{(m)} & \lambda_{\alpha,p}^{(m)} h_{\alpha,p}^{(m)} & \cdots & \lambda_{\alpha,\alpha}^{(m)} h_{\alpha,\alpha}^{(m)} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\lambda_{k,1}^{(m)} h_{k,1}^{(m)} & \cdots & \lambda_{k,p-1}^{(m)} h_{k,p-1}^{(m)} & \lambda_{k,p}^{(m)} h_{k,p}^{(m)} & \cdots & \lambda_{k,\alpha}^{(m)} h_{k,\alpha}^{(m)}
\end{bmatrix}
\] (4.59)

Let the first column of \( G^{(m)} \) represent the vector passed by non-systematic node \( m \) for the regeneration of the first systematic node. Since the interference along the symbols stored in remaining \( k - 1 \) systematic nodes need to be aligned (by Theorem 4.4.4),

\[
h_{j,1}^{(m)} = h_{j,1}^{(1)}, \quad j = 2, \ldots, k
\] (4.60)

Hence, the first column has to be of the form given by (4.59).

For every non-systematic node, suppose the vectors passed by it for the regeneration of the systematic nodes \( 1, \ldots, p - 1 \), where \( 1 < p \leq \alpha \), are linearly independent. These \( p - 1 \) vectors can be set as the first \( p - 1 \) columns of the generator matrix of that non-systematic node. Thus, by the interference alignment argument (in Theorem 4.4.4), the first \( p - 1 \) columns of \( G^{(m)} \) have the form as given above in equation (4.59).

Consider the regeneration of systematic node \( p \). Classify the non-systematic nodes into two types based on the vectors they pass for regeneration of node \( p \) as follows:

Type A: \( M_A = \) Set of all non-systematic nodes which pass a vector linearly dependent on the first \( p - 1 \) columns of their generator matrices.

Type B: \( M_B = \) Set of all non-systematic nodes which pass a vector linearly independent of the first \( p - 1 \) columns of their generator matrices.

Now consider, \( v_{\alpha+1}^{(m,p)} \) for \( m = k + 1, \ldots, n \), i.e., the component along \( z_{\alpha+1} \). For all \( m \in M_A \), \( v_{\alpha+1}^{(m,p)} \) is a linear combination of \( \{h_{\alpha+1,1}, \ldots, h_{\alpha+1,p-1}\} \). Whereas for all \( m \in M_B \), \( v_{\alpha+1}^{(m,p)} \) is linearly independent of this set of vectors. This is because, for \( m \in M_B \), \( v_{\alpha+1}^{(m,p)} \), along with \( v_{\alpha+1}^{(1)}, \ldots, v_{\alpha+1}^{(p-1)} \) are columns of \( H_{\alpha+1}^{(m)} \), and \( H_{\alpha+1}^{(m)} \) has to be non-singular (by Theorem 4.4.1).
Thus, $\mathbf{v}^{(m_A,p)}_{\alpha+1}$ and $\mathbf{v}^{(m_B,p)}_{\alpha+1}$, for some $m_A \in M_A$ and $m_B \in M_B$, are linearly independent, which violates the interference alignment condition given by Theorem 4.4.4. Hence, both types of non-systematic nodes cannot be present simultaneously.

Suppose all the non-systematic nodes are of type A. Then all the vectors passed by non-systematic nodes for the regeneration of node $p$ are linearly dependent on the first $p-1$ columns of their generator matrices. Hence, for all $m = k + 1, \ldots, n$, $\mathbf{v}^{(m,p)}_{p}$ is a linear combination of $\{h_{p,1}, \ldots, h_{p,p-1}\}$. Hence out of $\{v^{(k+1,p)}_{p}, \ldots, v^{(n,p)}_{p}\}$, at most $p-1$ can be linearly independent. Since $p \leq \alpha$, this violates the necessary condition given by Theorem 4.4.2.

Hence all the non-systematic nodes are of type B and the $p^{th}$ column of the generator matrices of non-systematic nodes is also as shown in equation (4.59). Inducting on the value of $p$ establishes that given an exact regenerating code for $d > 2k - 1$, the non-systematic node generator matrices can be re-written in the form given by equation (4.59).

Henceforth in this section, we consider all non-systematic node generator matrices to be of the form by equation (4.59). Also note that when the generator matrices are of this form, for $l = 1, \ldots, \alpha$,

$$\mathbf{x}^{(m,l)} = e_l$$  \hspace{1cm} (4.61)

and hence $\mathbf{y}^{(m,l)}$ is the $l^{th}$ column of the generator matrix.

The structure of the non-systematic node generator matrices establishes following two additional properties.

**Corollary 4.4.7** For $l = \alpha + 1, \ldots, k$, and any non-systematic nodes $m$ and $m'$,

$$H^{(m)}_l = H^{(m')}_l$$  \hspace{1cm} (4.62)

**Proof** Directly follows from the structure of the generator matrices of non-systematic nodes in (4.59).

**Corollary 4.4.8** For any non-systematic node $m$, and $d < 2k - 1$, any $\alpha$ vectors out of $\mathbf{y}^{(m,1)}, \ldots, \mathbf{y}^{(m,k)}$ are linearly independent.

**Proof** In Lemma 4.4.6, the choice of the first $\alpha$ systematic nodes is arbitrary. Hence, the given set of $\alpha$ systematic nodes can be considered as the first $\alpha$ nodes, and the generator matrices for non-systematic nodes can be re-written to obtain an equivalent code. By Lemma 4.4.6, the vectors passed by any non-systematic node for the regeneration of these $\alpha$ systematic nodes will be the $\alpha$ columns of its generator matrix, and hence will be independent (by Theorem 4.4.1).

**Remark 4.4.9** On similar lines it can be shown that for the case $n \geq 2k$ and $d = n - 1$, the vectors passed by a non-systematic node for the regeneration of any $k - 1$ systematic
nodes should be linearly independent. Moreover, at least one non-systematic node should pass linearly independent vectors for the regeneration of the $k$ systematic nodes.

Now, we prove a property of the linear combination vectors $\mathbf{x}^{(m,l)}$ used by the non-systematic nodes.

**Lemma 4.4.10** For $d < 2k - 1$, for any non-systematic node $m$, and any systematic node $l \in \{\alpha + 1, \ldots, k\}$,

$$x_i^{(m,l)} \neq 0 \quad i = 1, \ldots, \alpha$$  \hspace{1cm} (4.63)

**Proof** Suppose not. For some non-systematic node $m$, some $l \in \{\alpha + 1, \ldots, k\}$ and some $i \in \{1, \ldots, \alpha\}$ let

$$x_i^{(m,l)} = 0$$

Hence,

$$\mathbf{v}^{(m,l)} = \sum_{j=1, j \neq i}^{\alpha} x_j^{(m,l)} \mathbf{v}^{(m,j)}$$  \hspace{1cm} (4.64)

Hence $\{\mathbf{v}^{(m,j)}\}_{j=1, j \neq i}^{\alpha}$ and $\mathbf{v}^{(m,l)}$ are linearly dependent which contradicts Corollary 4.4.8.

**4.4.3 The Non-Existence Proof**

**Theorem 4.4.11** Linear exact regenerating codes for the MSR point with $\beta = 1$, achieving the cut-set bound on repair bandwidth do not exist for $d < 2k - 3$.

**Proof** Suppose there exists such a code. For this parameter regime, since $d = k - 1 + \alpha$, we get $k > \alpha + 2$. Also, since $\alpha > 1$, we have $n > k + 1$. Hence there are at least $\alpha + 3$ systematic nodes and at least two non-systematic nodes.

Consider regeneration of systematic node $(\alpha + 2)$. By the interference alignment property of Theorem 4.4.4, components along symbols stored in systematic nodes $(\alpha + 1)$ and $(\alpha + 3)$ are to be aligned, i.e.,

$$G^{(k+1)}_{\alpha+1} x^{(k+1,\alpha+2)} = \kappa_1 G^{(k+2)}_{\alpha+1} x^{(k+2,\alpha+2)}$$  \hspace{1cm} (4.65)

$$G^{(k+1)}_{\alpha+3} x^{(k+1,\alpha+2)} = \kappa_2 G^{(k+2)}_{\alpha+3} x^{(k+2,\alpha+2)}$$  \hspace{1cm} (4.66)

where $\kappa_1$ and $\kappa_2$ are some constants in $\mathbb{F}_q$. The vector passed by a non-systematic node cannot have a zero component along any systematic node, else it will violate Theorem 4.4.1. Thus

$$\kappa_1 \neq 0, \quad \kappa_2 \neq 0$$  \hspace{1cm} (4.67)
From Corollary 4.4.7,

\begin{align}
H_{\alpha+1}^{(k+1)} \alpha_{\alpha+1}^{(k+1,\alpha+2)} & = \kappa_1 H_{\alpha+1} \alpha_{\alpha+1}^{(k+2)} \alpha_{\alpha+1}^{(k+2,\alpha+2)} \\
H_{\alpha+3}^{(k+1)} \alpha_{\alpha+3}^{(k+1,\alpha+2)} & = \kappa_2 H_{\alpha+3} \alpha_{\alpha+3}^{(k+2)} \alpha_{\alpha+3}^{(k+2,\alpha+2)}.
\end{align}

(4.68) \hspace{2cm} (4.69)

Since \( H_{\alpha+1}, H_{\alpha+3}, \alpha_{\alpha+1}^{(k+1)} \) and \( \alpha_{\alpha+3}^{(k+1)} \) are non-singular (Theorem 4.4.1),

\begin{align}
\alpha_{\alpha+1}^{(k+1)} \alpha_{\alpha+1}^{(k+1,\alpha+2)} & = \kappa_1 \alpha_{\alpha+1}^{(k+2)} \alpha_{\alpha+1}^{(k+2,\alpha+2)} \\
\alpha_{\alpha+3}^{(k+1)} \alpha_{\alpha+3}^{(k+1,\alpha+2)} & = \kappa_2 \alpha_{\alpha+3}^{(k+2)} \alpha_{\alpha+3}^{(k+2,\alpha+2)}
\end{align}

(4.70) \hspace{2cm} (4.71)

\[ \implies \kappa_1 (\alpha_{\alpha+1}^{(k+1)})^{-1} \alpha_{\alpha+1}^{(k+2)} \alpha_{\alpha+1}^{(k+2,\alpha+2)} = \kappa_2 (\alpha_{\alpha+3}^{(k+1)})^{-1} \alpha_{\alpha+3}^{(k+2)} \alpha_{\alpha+3}^{(k+2,\alpha+2)} . \]

(4.72)

From Lemma 4.4.10,

\[ x_i^{(k+2,\alpha+2)} \neq 0 \quad \forall \ i = 1, \ldots, \alpha \]

(4.73)

which gives

\[ \kappa_1 (\alpha_{\alpha+1}^{(k+1)})^{-1} \alpha_{\alpha+1}^{(k+2)} = \kappa_2 (\alpha_{\alpha+3}^{(k+1)})^{-1} \alpha_{\alpha+3}^{(k+2)} . \]

(4.74)

Now, consider the exact regeneration of systematic node \( \alpha + 3 \). Analogous to equation (4.70), we get

\begin{align}
\alpha_{\alpha+1}^{(k+1)} \alpha_{\alpha+1}^{(k+1,\alpha+3)} & = \tilde{\kappa}_1 \alpha_{\alpha+1}^{(k+2)} \alpha_{\alpha+1}^{(k+2,\alpha+3)}
\end{align}

(4.75)

where \( \tilde{\kappa}_1 \) is some non-zero constant in \( \mathbb{F}_q \).

The component along the symbols stored in node \( \alpha + 3 \) in \( y_{(k+1,\alpha+3)} \),

\[ v_{(k+1,\alpha+3)} = G_{(k+1)} y_{(k+1,\alpha+3)} \]

(4.76)

\[ = H_{\alpha+3} \alpha_{\alpha+3}^{(k+1)} \alpha_{\alpha+3}^{(k+1,\alpha+3)} \]

(4.77)

\[ = \tilde{\kappa}_1 H_{\alpha+3} \alpha_{\alpha+3}^{(k+1)} (\alpha_{\alpha+1}^{(k+1)})^{-1} \alpha_{\alpha+1}^{(k+2)} \alpha_{(k+2,\alpha+3)} \]

(4.78)

where equation (4.78) is obtained by substituting for \( y_{(k+1,\alpha+3)} \) from (4.75). Similarly,

\[ v_{(k+2,\alpha+3)} = H_{\alpha+3} \alpha_{\alpha+3}^{(k+2)} \alpha_{(k+2,\alpha+3)} \]

(4.79)

\[ = \kappa_1 \kappa_2^{-1} H_{\alpha+3} \alpha_{\alpha+3}^{(k+1)} (\alpha_{\alpha+1}^{(k+1)})^{-1} \alpha_{\alpha+1}^{(k+2)} \alpha_{(k+2,\alpha+3)} \]

(4.79)

where equation (4.79) is obtained by substituting for \( \alpha_{\alpha+3}^{(k+2)} \) from (4.74). From equations
4.5 Towards Uniqueness of Code Construction

In Section 4.3, a family of explicit codes for the parameters \( n \geq 2k, \ d = n - 1 \) are constructed by first constructing codes for \( n = 2k, \ d = n - 1 \) and then extending this construction through puncturing. The construction of the generator matrices provided for the base case \( n = 2k, \ d = n - 1 \), possess all the necessary properties to make the code MDS and for the optimal exact regeneration of systematic nodes (provided in Section 4.4.1). In this section, we examine the extent to which these necessary properties force the structure of the code. We show that, most of the structure of the code is in fact forced, once we assume, that each non-systematic nodes passes linearly independent vectors for the regeneration of failed systematic nodes. This assumption holds true for the explicit constructions provided in Section 4.3.

Introducing some notation, for \( m = k+1, \ldots, 2k \) and \( i = 1, \ldots, k \), let \( D^{(m)}_i \) be a \( k \times k \) diagonal matrix given by

\[
D^{(m)}_i = \begin{bmatrix}
  d^{(m)}_{i,1} & 0 & \cdots & 0 \\
  0 & d^{(m)}_{i,2} & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & d^{(m)}_{i,k}
\end{bmatrix}
\] (4.81)

with \( d^{(m)}_{i,i} = 0 \). Let \( C_i \) be a \( k \times k \) matrix as in

\[
C_i = \begin{bmatrix}
  \xi^{(k+1)}_i & \xi^{(k+2)}_i & \cdots & \xi^{(2k)}_i
\end{bmatrix}
\] (4.82)

where \( \xi^{(m)}_i, \ m = k+1, \ldots, 2k, \) are \( k \)-length column vectors.
Let \( \Delta_j \) (1 \( \leq \) \( j \) \( \leq \) \( n \)) be a \((k-1 \times k)\) matrix with entries
\[
\begin{bmatrix}
d_{1,j}^{(k+1)} & d_{1,j}^{(k+2)} & \cdots & d_{1,j}^{(m)} \\
\vdots & \vdots & \ddots & \vdots \\
d_{j-1,j}^{(k+1)} & d_{j-1,j}^{(k+2)} & \cdots & d_{j-1,j}^{(m)} \\
d_{j+1,j}^{(k+1)} & d_{j+1,j}^{(k+2)} & \cdots & d_{j+1,j}^{(m)} \\
\vdots & \vdots & \ddots & \vdots \\
d_{k,j}^{(k+1)} & d_{k,j}^{(k+2)} & \cdots & d_{k,j}^{(m)}
\end{bmatrix}
\]
(4.83)

**Theorem 4.5.1** For any MDS code performing optimal exact regeneration of systematic nodes for \( n = 2k \), \( d = n - 1 \), with each non-systematic node passing linearly independent vectors for the regeneration of the \( \alpha \)(= \( k \)) systematic nodes, there exists an equivalent code with the non-systematic node generator matrices as
\[
G_i^{(m)} = \xi_i^{(m)} \xi_i^t + D_i^{(m)}
\]
(4.84)

and the following conditions hold:
1) \( C_i \) is invertible,
2) Any sub-matrix of \( \Delta_j \) is full rank.

**Proof** See Appendix D. \( \blacksquare \)

Note that the structure presented in Theorem 4.5.1 automatically applies to codes for \( n < 2k \), \( d = n - 1 \), since Corollary 4.4.8 mandates the vectors passed by a non-systematic node for regeneration of any \( \alpha \) nodes to be linearly independent.

**Constructing codes**

One means of constructing codes for the parameter set \( n = 2k \), \( d = n - 1 \) is as follows. For every \( i \) \((i \in \{1,\ldots,k\}\)) choose a \((k \times k)\) matrix \( \Psi(i) \) such that all of its sub-matrices are full rank. Set the \((k-1 \times k)\) matrix \( \Delta_i \) as rows 1, \( \ldots \), \( i - 1 \), \( i + 1 \), \( \ldots \), \( k \) of \( \Psi(i) \). Also, set \( C_i \) as \( \varepsilon_i \Psi(i) \) where \( \varepsilon_i \) is a \((k \times k)\) invertible diagonal matrix. This choice of \( C_i \) involves \( \varepsilon_i \) to make reconstruction feasible, and \( \Psi(i) \) for ease of decoding. With this choice, the reconstruction and regeneration properties are satisfied if \( \varepsilon_{i_1,j_1} \neq \varepsilon_{i_2,j_2}^{-1} \) for \( i_1, j_1, i_2, j_2 \in \{1,\ldots,k\}, j_1 \neq i_1, j_2 \neq i_2, i_1 \neq i_2 \), where \( \varepsilon_{i,j} \) is the \((j,j)\)th element of \( \varepsilon_i \).

The explicit code presented in Section 4.3 is of this form, with each \( \Psi(i) \) as \( \Psi \) - a Cauchy matrix, and each \( \varepsilon_i \) as \( \epsilon I \), where \( \epsilon \) is a non-zero element of the field such that \( \epsilon^2 \neq 1 \).
4.6 Existence and Construction for $d \geq 2k - 3$

In this section, we show that the exact regeneration of systematic nodes meeting the storage-repair bandwidth tradeoff at the MSR point, can be achieved for the parameter regime $d \geq 2k - 3$, under the assumption that on failure of a systematic node all the $k - 1$ existing systematic nodes participate in its regeneration. Note that this condition is automatically satisfied when $d = n - 1$.

Rewriting the lower bound on the repair bandwidth, i.e., equation (1.6), we get $d = \alpha + k - 1$. Hence, the new node replacing a failed systematic node, connects to $\alpha$ non-systematic nodes along with $k - 1$ existing systematic nodes.

4.6.1 Approach

We have seen that the properties mandated by reconstruction and regeneration dictate a lot of structure into the non-systematic node generator matrices. We will first choose a subset of the entries in the non-systematic node generator matrices, and certain linear combination vectors, which comply with this structure. The remaining entries in the node generator matrices and the linear combination vectors will be kept as variables. The reconstruction and regeneration conditions will be cast as condition on a product of rational polynomials in these variables to be non-zero. We show that each of these polynomials are well defined and non-zero. Then the following well-known Lemma (see, for e.g., [34]) is invoked to prove that there exist an assignment to the variables such that the product of the polynomials is also non-zero.

**Lemma 4.6.1 (Schwartz-Zippel Lemma)** Let $Q(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be a multivariate polynomial of total degree $d$ (the total degree is the maximum degree of the additive terms and the degree of a term is the sum of exponents of the variables). Let $r_1, \ldots, r_n$ be chosen independently and uniformly at random from $\mathbb{F}$. Then if $Q(x_1, \ldots, x_n)$ is not equal to a zero polynomial,

$$Pr[Q(r_1, \ldots, r_n) = 0] \leq \frac{d}{|\mathbb{F}|}$$

(4.85)
Thus if \(|F| > d\), there exists values \(\{r_1, \ldots, r_n\}\) such that \(Q(r_1, \ldots, r_n)\) is non-zero.

In [25], a similar problem arises in proving the existence of capacity achieving multicast network codes, though with respect to polynomials. But the argument can be easily extended to rational polynomials. If \(f_1(x), \ldots, f_p(x)\) are rational polynomials, define

\[
f_{p+1}(x) = \gcd(g_1(x), \ldots, g_p(x)).
\]

(4.86)

There exists a solution to \(x\) such that the product of the rational polynomials is well defined and non-zero if and only if there exists a solution to \(x\) such that the product of the polynomials \(f_1(x), \ldots, f_{p+1}(x)\) is non-zero. Hence, the algorithm given by Koetter and Medard in [25] can be used to find the values of the variables, provided the field size is large enough.

### 4.6.2 Existence and Construction

**Lemma 4.6.2** A necessary and sufficient condition for exact regeneration of systematic node \(l\) by connecting to the existing \(k - 1\) systematic nodes \(n_1, \ldots, n_{k-1}\) and \(\alpha\) non-systematic nodes \(m_1, \ldots, m_\alpha\), is that the set of vectors passed by these non-systematic nodes,

\[
\{v^{(m_1,l)}, \ldots, v^{(m_\alpha,l)}\}
\]

satisfy Theorems 4.4.2 and 4.4.4.

**Proof** Necessity: Shown in Theorems 4.4.2 and 4.4.4. Sufficiency: Suppose Theorem 4.4.4 is satisfied. Then, in the vectors passed by the \(\alpha\) non-systematic nodes, the components along other systematic nodes are all aligned along one vector, i.e., for any existing systematic node \(\hat{l}\)

\[
v_i^{(m_i,l)} = \kappa_i^{(m_i,l)} w_i^{(l)} \quad i = 1, \ldots, \alpha
\]

(4.88)

where \(w_i^{(l)}\) is a vector independent of \(m_i\), and \(\kappa_i's\) are some constants in \(F_q\). Let the vector passed by the systematic node \(\hat{l}\) be such that

\[
v_i^{(l,l)} = w_i^{(l)}.
\]

(4.89)

Thus, \(v_i^{(l,l)}\) can be used to cancel interfering components along systematic node \(\hat{l}\) in \(\{v^{(m_1,l)}, \ldots, v^{(m_\alpha,l)}\}\). Consider the vectors after removing out all the interfering components,

\[
\tilde{v}^{(m_i,l)} = v^{(m_i,l)} - \sum_{l=1, l \neq \hat{l}}^{k} \kappa_i^{(m_i,l)} v_i^{(l,l)}.
\]

(4.90)

The set of vectors \(\{\tilde{v}^{(m_1,l)}, \ldots, \tilde{v}^{(m_\alpha,l)}\}\) have no interfering components. Since the vectors
4.6 Existence and Construction for $d \geq 2k - 3$

$\chi^{(i,l)}$, $i = 1, \ldots, k$, $i \neq l$ have zero components along $z_l$, we have

$$\tilde{v}^{(m_i,l)}_l = v^{(m_i,l)}_l \quad (4.91)$$

Now, since Theorem 4.4.2 is satisfied, the vectors

$$\{\tilde{v}^{(m_1,l)}_l, \ldots, \tilde{v}^{(m_\alpha,l)}_l\}$$

are linearly independent. Hence all the symbols stored in node $l$ can be recovered. ■

To satisfy the conditions as outlined above, we choose the structure of the generator matrices of the non-systematic nodes of the form given by equation (4.59), i.e.,

$$G^{(m)}_i = H^{(m)}_i \Lambda^{(m)}_i \quad (4.92)$$

for $m = k + 1, \ldots, n$, $i = 1, \ldots, k$, where $\Lambda^{(m)}_i = \text{diag}\{\lambda^{(m)}_{i,1}, \ldots, \lambda^{(m)}_{i,\alpha}\}$ is a non-singular $\alpha \times \alpha$ diagonal matrix,

$$H^{(m)}_i = \begin{bmatrix} h^{(m)}_{i,1} & h^{(m)}_{i,2} & \cdots & h^{(m)}_{i,\alpha} \end{bmatrix} \quad (4.93)$$

is a non-singular $\alpha \times \alpha$ matrix, and $h^{(m)}_{i,j}$ is an $\alpha$-length column vector.

We design the code to be such that for regeneration of systematic node $l$ (for $1 \leq l \leq \alpha$), each non-systematic node passes the $l^{th}$ column of its generator matrix, i.e., we choose

$$\varphi^{(m,l)} = e_l \quad l = 1, \ldots, \alpha. \quad (4.94)$$

Now, to satisfy Theorem 4.4.4,

$$h^{(m)}_{i,j} = h_{i,j}, \quad m = k + 1, \ldots, n$$

$$i = 1, \ldots, k, \quad j = 1, \ldots, \alpha, \quad j \neq i \quad (4.95)$$

For $m = k + 1, \ldots, n$ we set

$$\lambda^{(m)}_{i,i} = 1, \quad i = 1, \ldots, \alpha. \quad (4.96)$$

Next we provide a constructive proof for the existence of exact regenerating codes at the MSR point, performing optimal exact regeneration of the systematic nodes for the parameter regime $d = 2k - 3$, and then invoke Theorem 4.3.3 to extend it to the case $d \geq 2k - 3$.

Theorem 4.6.3 Exact regeneration of systematic nodes meeting the cut-set lower bound
is possible with linear codes for the parameter set $d = 2k - 3$ when a failed systematic node connects to the $k - 1$ existing systematic nodes and any $\alpha$ non-systematic nodes.

**Proof** See Appendix E. ■

Now, applying Theorem 4.3.3, this result is true for all $d \geq 2k - 3$.

### 4.7 A Coding Scheme for any $(n, k, d)$: Exact Regeneration of Systematic Nodes

In the previous sections we showed that with $\beta = 1$, the cut-set bound cannot be met for the set of parameters $d < 2k - 3$. In this section, we give a coding scheme which can be used for any $(n, k, d)$ parameter set. This scheme assumes that when a systematic node fails, the existing $k - 1$ systematic nodes and any $\alpha$ non-systematic nodes participate in the regeneration. This scheme is optimal for $d \geq 2k - 1$, and achieves a repair bandwidth close to the cut-set lower bound for the remaining set of parameters.

#### 4.7.1 Scheme Description

Divide the $k$ systematic nodes into $\alpha$ groups. Analogous to the scheme given by Wu et al. [17], for regeneration of a systematic node, the existing systematic nodes in the same group as the failed node pass all their $\alpha$ symbols. The remaining systematic nodes and some $\alpha$ non-systematic nodes pass one symbol each.

The structure of the code is as follows. Let $\mu(l) \in \{1, \ldots, \alpha\}$ denote the group to which the systematic node $l$ belongs. Consider a set of variables $a_i^{(m)}$ and $b_{i,j}^{(m)}$, for $m = k + 1, \ldots, n$, $i = 1, \ldots, k$, $j = 1, \ldots, \alpha$, $j \neq \mu(i)$. Let

$$b_i^{(m)} = [b_{i,1}^{(m)} \cdots b_{i,\mu(i)-1}^{(m)} 0 b_{i,\mu(i)+1}^{(m)} \cdots b_{i,\alpha}^{(m)}]^t$$  \hspace{1cm} (4.97)

Let matrix $B_i^{(m)}$ be an $\alpha \times \alpha$ matrix such that it has $b_i^{(m)}$ as its $\mu(i)$th row, and zeros elsewhere. Also let

$$\tilde{b}_i^{(m)} = [b_{i,1}^{(m)} \cdots b_{i,\mu(i)-1}^{(m)} a_i^{(m)} b_{i,\mu(i)+1}^{(m)} \cdots b_{i,\alpha}^{(m)}]^t$$  \hspace{1cm} (4.98)

Let the node matrix of non-systematic node $m \in \{k + 1, \ldots, n\}$ be

$$G_i^{(m)} = a_i^{(m)} I_\alpha + B_i^{(m)}$$  \hspace{1cm} (4.99)

for $i = 1, \ldots, k$.

For example, suppose $k = 5$, $\alpha = 3$, and the systematic nodes are grouped as: $\{1, 2\}$, $\{3\}$, $\{4, 5\}$. Then, the node matrix stored by non-systematic node $m$, $m \in \{k + 1, \ldots, n\}$
is

\[
G^{(m)} = \begin{bmatrix}
  a_1^{(m)} & 0 & 0 \\
  b_1^{(m)} & a_1^{(m)} & 0 \\
  b_1^{(m)} & 0 & a_1^{(m)} \\
  a_2^{(m)} & 0 & 0 \\
  b_2^{(m)} & a_2^{(m)} & 0 \\
  b_2^{(m)} & 0 & a_2^{(m)} \\
  a_3^{(m)} & b_3^{(m)} & 0 \\
  0 & a_3^{(m)} & 0 \\
  0 & b_3^{(m)} & a_3^{(m)} \\
  a_4^{(m)} & 0 & b_4^{(m)} \\
  0 & a_4^{(m)} & b_4^{(m)} \\
  0 & 0 & a_4^{(m)} \\
  a_5^{(m)} & 0 & b_5^{(m)} \\
  0 & a_5^{(m)} & b_5^{(m)} \\
  0 & 0 & a_5^{(m)} 
\end{bmatrix}
\] (4.100)

**Regeneration**

Consider regeneration of systematic node \( l \ (\in \{1, \ldots, k\}) \). \( \alpha \) non-systematic nodes, say \( m_1, \ldots, m_\alpha \) pass the \( \mu(l)^{th} \) symbol, i.e., the \( \mu(l)^{th} \) column of their node matrices. The systematic nodes in other groups, say node \( l' \) in group \( \mu(l') \) (\( \mu(l') \neq \mu(l) \)), pass the vector \([0 \cdots 0 e_{\mu(l)} 0 \cdots 0]^t\) where the unit vector is in the position \( l' \). Since the component along node \( l' \) in the vector passed by any non-systematic node is \( a_{l'}^{(m)} e_{\mu(l)} \), it can be subtracted out. The existing systematic nodes in group \( \mu(l) \) pass all their symbols and hence components along these nodes can also be canceled out. Hence, for regeneration, the components given out by the non-systematic nodes along the direction of the \( l^{th} \) systematic node should be linearly independent. Thus, regeneration condition for systematic node \( l \) with this choice of \( \alpha \) non-systematic nodes reduces to a polynomial being non-zero, i.e.,

\[
det \left( \begin{bmatrix} \vec{b}_{l_1}^{(m_1)} & \ldots & \vec{b}_{l_1}^{(m_\alpha)} \end{bmatrix} \right)
\] (4.101)

Similar polynomials are obtained for all values of \( l \), and for all sets of \( \alpha \) non-systematic nodes. Clearly, none of these polynomials are identically zero.
Reconstruction

If the data collector connects to the $k$ systematic nodes, reconstruction is trivially satisfied. Consider data collector connecting to $p$ non-systematic nodes, and $k-p$ systematic nodes, $1 \leq p \leq k$. Let $m_1, \ldots, m_p$, $(m_1 < \ldots < m_p)$ be the non-systematic nodes to which it connects. Let $l_1, \ldots, l_p$, $(l_1 < \ldots < l_p)$ be the $p$ systematic nodes to which it does not connect. As in Section 4.3.2, reconstruction condition leads to the following condition of a polynomial being non-zero

$$
\det \begin{pmatrix}
G^{(m_1)}_{l_1} & \cdots & G^{(m_p)}_{l_1} \\
\vdots & \ddots & \vdots \\
G^{(m_1)}_{l_p} & \cdots & G^{(m_p)}_{l_p}
\end{pmatrix}.
$$

(4.102)

There exists an assignment of the variables such that this polynomial is non-zero,

$$
b^{(m)}_{i} = 0 \quad \forall i, m
$$

(4.103)

$$
a^{(m)}_{ij} = \begin{cases} 
1 & \text{if } i = j \\
0 & \text{if } i \neq j
\end{cases}
$$

(4.104)

By these assignments, the reconstruction matrix becomes an identity matrix, which is non-singular. Thus, the regeneration and reconstruction properties evaluate to the condition of the product of certain polynomials being non-zero. It is shown that none of these polynomials is identically zero. Assignment of values to the variables satisfying all the conditions can be obtained using the algorithm given by Koetter and Medard [25].

This scheme can be extended to regeneration using any combination of systematic and non-systematic nodes provided that the systematic nodes in the same group as the failed node participate in regeneration. The extended proof will involve a few more conditions of polynomials being non-zero.

4.7.2 Analysis

For $k \leq \alpha$, if all the $\alpha$ nodes are kept in different groups, this scheme achieves the minimum possible repair bandwidth and hence is optimal.

For $k > \alpha$, the amount of data to be downloaded for exact regeneration of a systematic node depends on the number of nodes in its group. If there are $\eta$ nodes in a group, the total number of symbols required to regenerate a node in that group, is given by:

$$
\gamma = (\eta - 1)\alpha + (d - \eta + 1)
$$

(4.105)

Lemma 4.7.1 The average repair bandwidth for exact regeneration of systematic nodes using the above described scheme is minimum when the groups are uniformly divided.
4.7 A Coding Scheme for any \((n, k, d)\): Exact Regeneration of Systematic Nodes

Figure 4.4: Average repair bandwidth(\(\gamma\)) required for exact regeneration of the systematic nodes with \(\beta = 1\) is plotted for various values of \(d\) for \(k = 9\).

**Proof** Directly follows from equation (4.105)

Let

\[
    s = \lfloor k/\alpha \rfloor \quad (4.106)
\]

Uniform division of groups would imply that out of the \(\alpha\) groups, \(k \mod \alpha\) groups contain \(s + 1\) nodes each and the rest contain \(s\) nodes each.

The average amount of download required for exact regeneration of the systematic nodes in our scheme is compared with the scheme proposed by Wu and Dimakis [17] (*group interference alignment*) in Figure 4.4. The lower bound on the repair bandwidth is also plotted along side. It can be seen that for \(d \geq 2k - 1\) (i.e. \(k \leq \alpha\)) our scheme achieves the lower bound. For smaller values of \(d\), the amount of data downloaded is higher. However, whether this achieved value of repair bandwidth is optimal or not is not known for \(d < 2k - 3\).
Chapter 5

An MBR Code for $d = n - 1$ with a Simple Graphical Description

The Minimum Bandwidth Regenerating (MBR) point is the fastest recovery point (on the storage-repair bandwidth tradeoff curve) in terms of the data to be downloaded for regeneration per unit of the source data. Also, among all the possible values of $d$, $d = n - 1$, gives the fastest recovery as all the existing nodes simultaneously help in the regeneration of the failed node. Hence the MBR point with $d = n - 1$ is highly suitable for applications such as distributed mail servers, where it is crucial to restore the system in the shortest possible time. In this chapter, an exact-regenerating MBR code for all parameters $(n, k, d = n - 1)$ is constructed, which has a simple graphical description. This code, when specialized to the parameter set $(n, k = n - 2, d = n - 1)$, can operate over binary field using solely XOR operations. The binary field operation makes the code practically relevant.

5.1 Explicit Exact-Regenerating MBR Code for $d = n - 1$

The MBR point corresponds to the least possible repair bandwidth, as discussed in Section 1.1.3. We reproduce below the values of $\alpha$ and $B$ when $\beta = 1$ for the MBR point:

$$\alpha = d , \quad (5.1)$$
$$B = kd - \binom{k}{2} . \quad (5.2)$$

We first furnish an example that will illustrate all the key ideas of the code construction. The general code construction will be provided in the succeeding subsection.
5.1 Explicit Exact-Regenerating MBR Code for $d = n - 1$

5.1.1 An Example Code

Let $n = 5$ and $k = 3$. Thus $d = n - 1 = 4$, and setting $\beta = 1$ gives $\alpha = 4$ and $B = 9$. Define the set of $\binom{n}{2} = 10$ vectors

$$\left\{ \mathbf{v}_{1,2}, \mathbf{v}_{1,3}, \mathbf{v}_{1,4}, \mathbf{v}_{1,5} \right\}, \left\{ \mathbf{v}_{2,3}, \mathbf{v}_{2,4}, \mathbf{v}_{2,5} \right\}, \left\{ \mathbf{v}_{3,4}, \mathbf{v}_{3,5} \right\}, \mathbf{v}_{4,5}$$

each of length $B$, such that any $B$ of them are linearly independent. Also, for $1 \leq j < i \leq 5$, define vector $\mathbf{v}_{i,j} = \mathbf{v}_{j,i}$.

Now, define a vector $\mathbf{u} = [u_1, \ldots, u_9]^t$ with its elements as the 9 source symbols. Then, the 5 nodes store the following symbols:

Node 1: \{\mathbf{u}^t \mathbf{v}_{1,2}, \mathbf{u}^t \mathbf{v}_{1,3}, \mathbf{u}^t \mathbf{v}_{1,4}, \mathbf{u}^t \mathbf{v}_{1,5}\}
Node 2: \{\mathbf{u}^t \mathbf{v}_{2,1}, \mathbf{u}^t \mathbf{v}_{2,3}, \mathbf{u}^t \mathbf{v}_{2,4}, \mathbf{u}^t \mathbf{v}_{2,5}\}
Node 3: \{\mathbf{u}^t \mathbf{v}_{3,1}, \mathbf{u}^t \mathbf{v}_{3,2}, \mathbf{u}^t \mathbf{v}_{3,4}, \mathbf{u}^t \mathbf{v}_{3,5}\}
Node 4: \{\mathbf{u}^t \mathbf{v}_{4,1}, \mathbf{u}^t \mathbf{v}_{4,2}, \mathbf{u}^t \mathbf{v}_{4,3}, \mathbf{u}^t \mathbf{v}_{4,5}\}
Node 5: \{\mathbf{u}^t \mathbf{v}_{5,1}, \mathbf{u}^t \mathbf{v}_{5,2}, \mathbf{u}^t \mathbf{v}_{5,3}, \mathbf{u}^t \mathbf{v}_{5,4}\}

A graphical representation of this code is provided in Figure 5.1.

**Data Reconstruction:** Suppose the data collector connects to nodes 1, 2 and 3. It gains access the 9 symbols

$$\left\{ \mathbf{u}^t \mathbf{v}_{1,2}, \mathbf{u}^t \mathbf{v}_{1,3}, \mathbf{u}^t \mathbf{v}_{1,4}, \mathbf{u}^t \mathbf{v}_{1,5} \right\}.$$

The 9 vectors corresponding to these symbols are linearly independent by construction, permitting the data collector to recover the source symbols $u_1, \ldots, u_9$. The same holds for any choice of 3 nodes.

**Exact Regeneration:** Suppose node 3 fails. Nodes 1, 2, 4 and 5 pass the symbols $\mathbf{u}^t \mathbf{v}_{1,3}, \mathbf{u}^t \mathbf{v}_{2,3}, \mathbf{u}^t \mathbf{v}_{4,3}$ and $\mathbf{u}^t \mathbf{v}_{5,3}$ respectively. These are precisely the four symbols that were stored in node 3 prior to failure, hence node 3 is exactly regenerated.

The set of parameters chosen for this example gives

$$\binom{n}{2} = B + 1.$$
Figure 5.1: A graphical representation of an exact-regenerating code for the MBR point with \([n = 5, \, d = 4, \, k = 3]\). The set of vectors \(v_{i,j}\)'s, each of length 10, form a single parity check code.

Thus the 10 vectors used for the construction can be chosen to form a single parity check code of dimension 9. Hence, the exact-regenerating code for this set of parameters can be obtained in any finite field, including \(\mathbb{F}_2\).

5.1.2 Code Construction for the General Set of Parameters with 
\(d = n − 1\)

In this section, we provide an explicit exact regenerating MBR code \(C\) for the general set of parameters \([n, \, k, \, d = n − 1]\). The source symbols are assumed to be drawn from a finite field \(\mathbb{F}_q\) of size \(q\), and the code is linear over \(\mathbb{F}_q\). As described previously, codes will be constructed for the atomic parameter \(\beta = 1\); codes for any higher value of \(\beta\) can be obtained via concatenation.

First, define a set of \(\binom{n}{2}\) vectors \(\{v_{i,j}\} \quad (1 \leq i < j \leq n)\), each of length \(B\), such that any \(B\) of them are linearly independent; and a second set of \(\binom{n}{2}\) vectors \(\{v_{j,i}\} \quad (1 \leq i < j \leq n)\), such that \(v_{j,i} = v_{i,j}\). Next, represent the \(n\) storage nodes as the vertices of an undirected complete graph with \(n\) vertices. Associate every edge in the graph with one symbol: the symbol associated to the edge between nodes \(i\) and \(j\) being \(u^t v_{i,j}\), where \(u^t v_{i,j}\) is the dot product of \(u^t\) and \(v_{i,j}\).

The set of \(n−1\) symbols stored in node \(p\) \((p = 1, \ldots, n)\) are the \(n−1\) symbols associated
to the edges incident on vertex $p$ of the graph, namely,

$$\{u^t v_{p,j}\} \quad 1 \leq j \leq n, \ j \neq p.$$ 

The following theorems describe the algorithms to perform reconstruction and regeneration.

**Theorem 5.1.1 (Data Reconstruction)** In the code $C$ presented, a data collector can recover all the $B$ message symbols by connecting to any $k$ storage nodes.

**Proof** Let $R = \{r_1, \ldots, r_k\}$ be the set of $k$ storage nodes to which the data collector connects to. The data collector downloads all the $d$ symbols stored in each of the nodes in $R$. Thus the data collector has access to these $kd$ symbols. Since every pair of nodes in the set $R$ has exactly one symbol in common, there are $\binom{k}{2}$ redundant symbols. Thus, the data collector has access to $kd - \binom{k}{2} = B$ distinct symbols, which are:

$$u^t v_{r_i,j} \quad \forall \ i \in \{1, \ldots, k\}, \ j \in \{1, \ldots, d\}\backslash\{r_1, \ldots, r_i\}.$$ 

By construction, the vectors $v_{r_i,j}$ are linearly independent thereby enabling the data collector to recover the $B$ source symbols.

Thus, the data reconstruction (decoding) procedure is identical to that of decoding an MDS code over an erasure channel.

**Theorem 5.1.2 (Exact Regeneration)** In the code $C$ presented, exact regeneration of any failed node can be achieved by connecting to remaining $(n-1)$ nodes.

**Proof** On failure of a storage node, the replacement node downloads one symbol each from the $n-1$ remaining nodes. Each of the remaining nodes pass the symbol associated with the edge it has in common with the failed node. However, these are precisely the $n-1$ symbols that were stored in the node prior to failure. Thus, the replacement node simply stores the symbols that it so obtains, completing the process of exact regeneration.

Clearly, the process of exact regeneration has a very low complexity, since the code does not entail any arithmetic operations in the process.

**5.1.3 Size of the Finite Field**

The sole constraint on the field size required for the code construction is that it should enable construction of a $\left[\binom{n}{2}, B\right]$ MDS code. For instance, if we use a Reed-Solomon code, code construction is possible using any finite field of size at least $\binom{n}{2}$.
**XOR based constructions:** A special case of the construction is for the parameter set \([n, k = n - 2, d = n - 1]\). Here \(\binom{n}{2} = B + 1\). For this setting, the \([\binom{n}{2}, B]\) MDS code can be chosen to be a single parity check code. Thus, for this set of parameters, a finite field of any arbitrary size can be employed; choosing the binary field facilitates every operation to be carried out via XORs, making the code easy to implement.
Chapter 6

An Approximately-Exact Regenerating MSR Code for $d = k + 1$

At the MSR point, the parameter set $(n, k, d = k + 1)$, for the atomic case of $\beta = 1$, falls in the non-achievable region (See Section 4.4). In this chapter, an explicit approximately-exact regenerating MSR code is presented. In these codes we have $d = k + 1$, which is the minimum value of $d$ for which a reduction in repair bandwidth can be achieved using regenerating codes. This code is relevant in highly volatile systems such as peer-to-peer storage systems.

This code is further used as a basis for an explicit code construction, for all values of the parameters which uniformly reduces the repair bandwidth to approximately half the file size, provided in Section 6.2.

6.1 Explicit Codes for $d = k + 1$

In Section 4.4, it was established that exact regenerating MSR codes with $\beta = 1$, do not exist when the parameters are such that $d < 2k - 3$. In this section, we give an explicit code construction for a approximately exact regenerating MSR code for the parameter set $d = k + 1$. These codes are easier described in terms of the message symbols, and hence we adopt the message symbol viewpoint here.

As $d$ is independent of $n$, this code enjoys all the advantages of being applicable to arbitrary $n$ provided in Section 2.4.2. The code can handle any number of node failures optimally as long as at least $k + 1$ nodes remain functional. Note that, by definition, if less than $k$ nodes are functional then a part of the data will be permanently lost. If exactly $k$ nodes are functional, then these nodes will have to pass all the information stored in them for regeneration, hence no reduction of the repair bandwidth is possible.
With \( d = k + 1 \), the cut-set bound \( d = \alpha + k - 1 \), we have

\[
B = 2k, \quad (6.1)
\]
\[
\alpha = 2. \quad (6.2)
\]

Let the \( 2k \) source symbols be the elements of the \( 2k \)-dimensional column vector \( \underline{u} \). Let

\[
\underline{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad (6.3)
\]

where both \( u_1 \) and \( u_2 \) are \( k \) length column vectors.

### 6.1.1 Code construction:

Node \( i \) (\( i = 1, \ldots, n \)) stores \( \langle u_1^i p_i, u_2^i p_i + u_1^i q_i \rangle \) as its two symbols. We shall refer to the vectors \( p_i \) and \( q_i \) as the main vector and the auxiliary vector of a node respectively. The elements of the auxiliary vectors are known but can take any arbitrary values from \( F_q \). The main vectors are the ones which are actually used for reconstruction and regeneration.

Let the set of main vectors \( p_i (i = 1, \ldots, n) \) form a \( k \)-dimensional MDS code over \( F_q \). The required field size is the minimum field size required to construct an \( [n, k] \) MDS code. If we use a Reed-Solomon code, the minimum field size required turns out to be just \( n \).

For example, consider \( n = 5 \), \( k = 3 \) and \( d = 4 \). We have \( B = 6 \) and the elements of three dimensional vectors \( u_1 \) and \( u_2 \) as the source symbols. Let the main vectors \( p_i (i = 1, \ldots, 5) \) form a Reed-Solomon code, with \( p_i = (1 \theta_i \theta_i^2)^t \). \( \theta_i \) (\( i = 1, \ldots, 5 \)) take distinct values from \( F_q (q \geq 5) \). We can initialize elements of \( q_i (i = 1, \ldots, 5) \) to any arbitrary values from \( F_q \).

### 6.1.2 Reconstruction:

A data collector will connect to any \( k \) nodes and download both the symbols stored in each of these nodes. The first symbols of the \( k \) nodes provide \( u_1^i p_i \) at \( k \) different values of \( i \). To solve for \( u_1 \), we have \( k \) linear equations in \( k \) unknowns. Since \( p_i \)'s form a \( k \)-dimensional MDS code, these equations are linearly independent, and can be solved easily to decode the source symbols \( u_1 \).

Now, as \( u_1 \) and \( q_i \) are known, \( u_1^t q_i \) can be subtracted out from the second symbols of each of the \( k \) nodes. This leaves us with the values of \( u_2^t p_i \) at \( k \) different values of \( i \). Using these, the source symbols \( u_2 \) can be decoded.

Thus all \( B \) data units can be recovered by a data collector which connects to any \( k \) nodes. We also see that reconstruction is possible irrespective of the values of the auxiliary vectors \( q_i \). The decoding procedure requires independent decoding of a set of two \( [n, k] \) MDS codes (such as RS codes) in the presence of erasures, along with the subtraction.
6.1 Explicit Codes for $d = k + 1$

operations required to remove the auxiliary component.

6.1.3 Regeneration:

In our construction, when a node fails, the main vector of the regenerated node has the same value as that of the failed node, although the auxiliary vector is allowed to be different. This gives the code its *approximately exact* regenerating property. Suppose node $j$ fails. The node replacing it would contain $(u_1^tp_j, u_2^tp_j + u_1^t \tilde{q}_j)$ where elements of $\tilde{q}_j$ can take arbitrary values from $\mathbb{F}_q$ and are not constrained to be equal to those of $q_j$. As the reconstruction property holds irrespective of the values of $q_j$, the regenerated node along with the existing nodes has all the desired properties.

For regeneration of a failed node, say node $\Lambda_{d+1}$, $d$ existing nodes, say $\Lambda_1, \ldots, \Lambda_d$, give one symbol each formed by a linear combination of the symbols stored in them.

Let $a_i$ and $b_i$ $(i = 1, \ldots, d)$ be the coefficients of the linear combination for the symbol given out by node $\Lambda_i$. Let $v_i = a_i(u_1^t p_{\Lambda_i}) + b_i(u_2^t p_{\Lambda_i} + u_1^t q_{\Lambda_i})$ be this symbol. Let $\delta_i$ and $\rho_i$ $(i = 1, \ldots, d)$ be the coefficients of the linear combination used to generate the two symbols of the regenerated node. Thus the regenerated node will be

$$\left( \sum_{i=1}^{d} \delta_i v_i, \sum_{i=1}^{d} \rho_i v_i \right).$$  (6.4)

Choose $b_i = 1$ $(i = 1, \ldots, d)$. Now choose $\rho_i$ $(i = 1, \ldots, d)$ such that

$$\sum_{i=1}^{d} \rho_i b_i p_{\Lambda_i} = p_{\Lambda_{d+1}}$$  (6.5)

and $\delta_i$ $(i = 1, \ldots, d)$ such that

$$\sum_{i=1}^{d} \delta_i b_i p_{\Lambda_i} = 0.$$  (6.6)

Equations (6.16) and (6.17) are sets of $k$ linear equations in $d = k + 1$ unknowns each. Since $p_{\Lambda_i}$’s form a $k$–dimensional MDS code these can be solved easily in $\mathbb{F}_q$. This also ensures that we can find a solution to equation (6.17) with all $\delta_i$’s non-zero.

Now, choose $a_i$ $(i = 1, \ldots, d)$ such that

$$\sum_{i=1}^{d} \delta_i (a_i p_{\Lambda_i} + b_i q_{\Lambda_i}) = p_{\Lambda_{d+1}},$$  (6.7)
6.2 A General Explicit Code for All Parameters

i.e.,

\[
\sum_{i=1}^{d} \delta_{i}a_{i}p_{N_{i}} = p_{N_{d+1}} - \sum_{i=1}^{d} \delta_{i}b_{i}q_{N_{i}}.
\]  

(6.8)

Equation (6.19) is a set of \( k \) linear equations in \( d = k + 1 \) unknowns which can be easily solved in \( \mathbb{F}_{q} \). Since none of the \( \delta_{i} \) \((i = 1, \ldots, d)\) are zero, the particular choice of \( p_{N_{i}} \)'s used guarantees a solution for \( a_{i} \) \((i = 1, \ldots, d)\). Hence, regeneration of any node using any \( d \) other nodes is achieved.

6.2 A General Explicit Code for All Parameters

In this section, we present an explicit code for all values of the parameters, which reduces the repair bandwidth for all the nodes to approximately \( B/2 \). This construction is based on the MSR code provided in Section 6.1. To the best of our knowledge, this is the first explicit code which reduces the repair bandwidth of all the nodes for any feasible values of the system parameters \( B, \alpha, n \) and \( k \).

First, a code construction for the trivial case of \( \alpha = 1 \) is provided following with a construction for \( \alpha = 2 \). Constructions for \( \alpha = 1 \) and \( \alpha = 2 \) are used as building blocks in the code construction for any value of \( \alpha \).

6.2.1 Construction for the Case \( \alpha = 1 \)

\( \alpha = 1 \) is a trivial case in which any \((n, k)\) MDS code minimizes the repair bandwidth. However, code construction for this case is provided here as this will be used in the construction for the general case.

Let \( f \) be a vector with \( B \) source symbols as its elements. Let \( l \) denote the length of this vector. Hence \( l = B \), and from equation (1.2) we have

\[
l \leq k
\]

(6.9)

Let \( p^{(i)} \) \((i = 1, \ldots, n)\) be vectors of length \( l \) forming a \( l \)-dimensional MDS code over \( \mathbb{F}_{q} \). In the rest of the chapter, we will use superscripts to denote the node number corresponding to any symbol.

**Code:** Node \( i \) stores \( f^{t}p^{(i)} \), \( i = 1, \ldots, n \).

**Reconstruction**

**Lemma 6.2.1** This code for \( \alpha = 1 \) can achieve successful reconstruction for a DC connecting to any \( k \) nodes.
Proof The DC will obtain $f^t p^{(i)}$ evaluated at $k$ different values. Since $p^{(i)}$’s form a $l$-dimensional MDS code and $k \geq l$, the DC can solve for the values of the $B$ source symbols.

Regeneration

For regeneration of a failed node, we need $d \geq l$.

Lemma 6.2.2 Any failed node can be regenerated by downloading one symbol each from any $d$ existing nodes.

Proof Since $l = B$, the entire file can be reconstructed, using which the symbol stored in the failed node can be regenerated.

Repair Bandwidth

The repair bandwidth for any node is

$$\gamma_1(l) = l = B$$

(6.10)

where the subscript $1$ indicates that there is only one symbol to be regenerated in the failed node. This is the minimum possible repair bandwidth for $\alpha = 1$. Note that the repair bandwidth is a function of the length $l$ of the source vector.

6.2.2 Construction for the Case $\alpha = 2$

Partition the source symbols into two sets $S$ and $S'$ having sizes $l$ and $l'$ respectively such that

$$0 \leq l, l' \leq k$$

(6.11)

and

$$(l + l') = B$$

(6.12)

Let $f$, $g$ be two vectors with their elements as the constituents of the sets $S$ and $S'$ respectively. Hence these vectors have lengths $l$ and $l'$ respectively. For $i = 1, \ldots, n$ let $p^{(i)}$ be vectors of length $l$ forming a $l$-dimensional MDS code over $\mathbb{F}_q$ and $r^{(i)}$ be vectors of length $l'$ forming a $l'$-dimensional MDS code over $\mathbb{F}_q$.

Code: Node $i$ ($i = 1, \ldots, n$) stores $(f^t p^{(i)}, g^t r^{(i)} + f^t u^{(i)})$ as its two symbols. The vectors $p^{(i)}$ and $r^{(i)}$ will be referred to as the main vectors and $u^{(i)}$ as the auxiliary vector of the node $i$. The elements of $u^{(i)}$ can be initialized to any arbitrary values from $\mathbb{F}_q$.

For example, consider $k = 3$ and $B = 5$. Let $b_0$, $b_1$, $b_2$, $b_3$, $b_4$ be the source symbols. Let $l = 3$ and $l' = 2$. Set $f = (b_0, b_1, b_2)^t$ and $g = (b_3, b_4)^t$. For $i = 1, \ldots, n$ let the main
vectors $p^{(i)}$ and $r^{(i)}$ form a Reed-Solomon code with $p^{(i)} = (1 \theta_i \theta^2_i)$ and $r^{(i)} = (1 \theta_i)$. Elements of $u^{(i)}$ can be initialized to arbitrary values from $\mathbb{F}_q$.

Reconstruction:

**Lemma 6.2.3** The code given for $\alpha = 2$ can achieve successful reconstruction for a DC connecting to any $k$ nodes.

**Proof** The first symbols of some $l$ out these $k$ nodes provide $f^t p^{(i)}$ at $l$ different values of $i$. To solve for $f$, we have $l$ linear equations in $l$ unknowns. Since $p^{(i)}$’s form a $l$–dimensional MDS code, these equations are linearly independent. As $l \leq k$, they can be solved to obtain values of the elements of $f$.

Now, as $f$ and $u^{(i)}$ are known, $f^t u^{(i)}$ can be subtracted out from the second symbols of some $l'$ out of the $k$ nodes. This gives $g^t r^{(i)}$ evaluated at $l'$ different values of $i$. As $l' \leq k$, this can be used to recover the elements of $g$.

Thus the $B$ symbols can be recovered by a DC which connects to any $k$ nodes. We also see that reconstruction can be performed irrespective of the values of the auxiliary vectors $u^{(i)}$.

Regeneration:

In this construction, when a node fails, the main vectors of the regenerated node will be identical to that of the failed node and the auxiliary vector is allowed to be different. Suppose node $j$ fails. The node replacing it would contain $(f^t p^{(j)}, g^t r^{(j)} + f^t u^{(j)})$ where elements of $\tilde{u}^j$ can take arbitrary values from $\mathbb{F}_q$ and are not constrained to be equal to those of $u^j$. As the reconstruction property holds irrespective of the values of $u^j$, the regenerated node along with the existing nodes has all the desired properties.

For regeneration of the failed node we need

$$d \geq \max(l, l' + 1) \quad (6.13)$$

**Lemma 6.2.4** A failed node can be successfully regenerated by downloading one symbol each from any $d$ existing nodes.

**Proof** The node replacing the failed node downloads one symbol each from some $d$ of the $n - 1$ existing nodes. Consider failure of node $\Lambda_{d+1}$, where nodes $\Lambda_1, \ldots, \Lambda_d$ participate in regeneration. Here the set $\{\Lambda_1, \ldots, \Lambda_{d+1}\}$ is some subset of $\{1, \ldots, n\}$.

For $i = 1, \ldots, d$, node $\Lambda_i$ passes a symbol which is a linear combination of the two symbols stored in it. Let $a_i$ and $b_i$ be the coefficients of this linear combination. Thus $\pi_i = a_i(f^t p^{(\Lambda_i)}) + b_i(g^t r^{(\Lambda_i)} + f^t u^{(\Lambda_i)})$ is the symbol passed by node $\Lambda_i$. 


The two symbols stored in the new node will be linear combinations of these downloaded symbols. Let \( \delta_i \) and \( \rho_i \) be the coefficients of these linear combinations. Thus the regenerated node will store

\[
\left( \sum_{i=1}^{d} \delta_i \pi_i, \sum_{i=1}^{d} \rho_i \pi_i \right)
\]  

(6.14)

Choose

\[
b_i = \begin{cases} 
1 & \text{for } i = 1, \ldots, l' + 1 \\
0 & \text{for } i = l' + 2, \ldots, d
\end{cases}
\]  

(6.15)

Now choose \( \rho_i \) \((i = 1, \ldots, l' + 1)\) such that

\[
\sum_{i=1}^{l'+1} \rho_i p^{(\Lambda_i)} = p^{(\Lambda_{d+1})}
\]  

(6.16)

and \(\rho_i = 0\) for \(i = l' + 2, \ldots, d\).

Equation (6.16) is a set of \(l'\) non-homogeneous linear equations in \(l' + 1\) unknowns. Since \(p^{(\Lambda_i)}\)'s form a \(l'\)-dimensional MDS code, a solution is guaranteed and can be easily obtained.

Choose \( \delta_i \) \((i = 1, \ldots, l' + 1)\) such that

\[
\sum_{i=1}^{d} \delta_i p^{(\Lambda_i)} = 0
\]  

(6.17)

and \(\delta_i = 1\) for \(i = l' + 2, \ldots, d\).

Equation (6.17) is a set of \(l'\) homogeneous linear equations in \(l' + 1\) unknowns. Since \(p^{(\Lambda_i)}\)'s form a \(l'\)-dimensional MDS code, a solution with all \(\delta_i \) \((i = 1, \ldots, l' + 1)\) non-zero can be obtained in \(F_q\).

Now, choose \(a_i \) \((i = 1, \ldots, d)\) such that

\[
\sum_{i=1}^{d} \delta_i (a_i p^{(\Lambda_i)} + b_i u^{(\Lambda_i)}) = p^{(\Lambda_{d+1})}
\]  

(6.18)

i.e

\[
\sum_{i=1}^{d} \delta_i a_i p^{(\Lambda_i)} = p^{(\Lambda_{d+1})} - \sum_{i=1}^{d} \delta_i b_i u^{(\Lambda_i)}
\]  

(6.19)

Equation (6.19) is a set of \(l\) linear equations in \(d\) unknowns. Since \(d \geq l\), none of the
δ_i (i = 1, ..., d) are zero, and \( p^{(A_i)} \)'s form a \( l \)-dimensional MDS code, it can be solved to obtain values for \( a_i \) (i = 1, ..., d).

**Optimum Partition Size and Repair Bandwidth**

The repair bandwidth for any node is

\[
\gamma_2(l, l') = \max(l, l' + 1)
\]  
(6.20)

Thus, to minimize the repair bandwidth the partition sizes should be

\[
l = \left\lceil \frac{B}{2} \right\rceil \quad \text{and} \quad l' = \left\lfloor \frac{B}{2} \right\rfloor
\]  
(6.21)

### 6.2.3 Construction for Any \( \alpha \)

This section gives a code construction for the general case using constructions for \( \alpha = 1 \) and \( \alpha = 2 \) as building blocks.

Let

\[
\tau = \left\lfloor \frac{\alpha}{2} \right\rfloor
\]  
(6.22)

Partition the \( B \) source symbols into \( 2\tau + 1 \) sets \( S_1, S'_1, ..., S_\tau, S'_\tau, S_{\tau+1} \) having sizes \( l_1, l'_1, ..., l_\tau, l'_\tau, l_{\tau+1} \) respectively satisfying the following conditions

\[
0 \leq l_j, l'_j \leq k \quad \forall j \in \{1, ..., \tau\}
\]  
(6.23)

\[
l_{\tau+1} = 0 \quad \text{for } \alpha \text{ even}
\]  
(6.24)

\[
0 \leq l_{\tau+1} \leq k \quad \text{for } \alpha \text{ odd}
\]  
(6.25)

and

\[
\sum_{j=1}^{\tau} (l_j + l'_j) + l_{\tau+1} = B
\]  
(6.26)

Let \( f_1, g_1, ..., f_{\tau}, g_{\tau}, f_{\tau+1} \) be \( 2\tau + 1 \) vectors with their elements as the constituents of the sets \( S_1, S'_1, ..., S_\tau, S'_\tau, S_{\tau+1} \) respectively. Hence the lengths of these vectors are \( l_1, l'_1, ..., l_\tau, l'_\tau, l_{\tau+1} \) respectively.

For \( j = 1, ..., \tau+1 \) let \( p^{(i)}_j \) (i = 1, ..., n) be vectors of length \( l_j \) forming a \( l_j \)-dimensional MDS code over \( \mathbb{F}_q \). For \( j = 1, ..., \tau \) let \( r^{(i)}_j \) (i = 1, ..., n) be vectors of length \( l'_j \) forming a \( l'_j \)-dimensional MDS code over \( \mathbb{F}_q \).

**Code:** For every pair of vectors \((f_j, g_j), j = 1, ..., \tau\) apply the construction given for \( \alpha = 2 \) in Section 6.2.2 by taking \( f_j \) as \( f \) and \( g_j \) as \( g \). Each pair of vectors determines two symbols to be stored in that node. When \( \alpha \) is odd, the construction given for \( \alpha = 1 \) in
Section 6.2.1 is applied on \( f_{\tau+1} \) to obtain one symbol. The symbols so obtained constitute the \( \alpha \) symbols stored at each node.

Hence node \( i (\in \{1, \ldots, n\}) \) stores

\[
\left[ \{ f_t^{i} p_{t}^{(i)}, g_t^{i} r_{t}^{(i)} + f_t^{i} u_{t}^{(i)} \}_{j=1}^{\tau}, f_t^{\tau+1} p_{\tau+1}^{(i)} \} \right]
\]

as its \( \alpha \) symbols (as shown in Figure 6.1), where the symbol corresponding to \( f_{\tau+1} \) is present only when \( \alpha \) is odd.

**Reconstruction**

**Theorem 6.2.5** *The code given can achieve successful reconstruction for a DC connecting to any \( k \) nodes.*

**Proof** Each of the first \( \tau \) pair of symbols stored in any node is separately constructed using the code given for \( \alpha = 2 \) in Section 6.2.2. Apply Lemma 6.2.3 separately on each pair of symbols stored in the \( k \) nodes to reconstruct \( \{ f_t^{i}, g_t^{i} \}_{i=1}^{\tau} \).

When \( \alpha \) is odd, the last symbol stored is constructed using the code for \( \alpha = 1 \) given in Section 6.2.1. Apply Lemma 6.2.1 on the last symbol stored in the \( k \) nodes to reconstruct \( f_{\tau+1} \). Thus all the \( B \) source symbols can be reconstructed by the DC. \( \square \)

**Regeneration**

**Theorem 6.2.6** *The code given can perform successful regeneration of any failed node.*

**Proof** The symbols to be stored in the new node replacing the failed node are regenerated in the following manner. Each of the first \( \tau \) pairs of symbols are regenerated separately
as described in Lemma 6.2.4. The amount of download required for the regeneration of the $j^{th}$ pair of symbols is $\max(l_j, l'_j + 1)$.

When $\alpha$ is odd, the last symbol to be stored in the new node is regenerated as described in Lemma 6.2.2 by downloading $l_{\tau+1}$ symbols.

Encoding, reconstruction and regeneration is performed on each pair of vectors separately, thereby immensely reducing the complexity of each of these operations.

**Optimum partition size and repair bandwidth**

The repair bandwidth is dependent on the partition sizes. By the method of regeneration described in Theorem 6.2.6, the repair bandwidth for any node is given by

$$\gamma = \sum_{j=1}^{\tau} \gamma_2(l_j, l'_j + 1) + \gamma_1(l_{\tau+1})$$  \hspace{1cm} (6.28)

$$= \sum_{j=1}^{\tau} \max(l_j, l'_j + 1) + l_{\tau+1}$$  \hspace{1cm} (6.29)

where equation (6.29) follows from equations (6.10) and (6.20).

Thus the optimum size of the partitions is the solution to the following optimization problem:

$$\min \left[ \sum_{j=1}^{\tau} \max(l_j, l'_j + 1) + l_{\tau+1} \right]$$  \hspace{1cm} (6.30)

subject to the conditions (6.23), (6.24), (6.25), (6.26) and $l_j, l'_j, l_{\tau+1} \forall j \in \{1, \ldots, \tau\}$ being integers.

The following theorem provides a method to pick the partition sizes in order to minimize the repair bandwidth.

**Theorem 6.2.7** The bandwidth required to repair a failed node is upper bounded by

$$\gamma \leq \frac{B}{2} + \frac{\alpha}{2} + k - 1$$  \hspace{1cm} (6.31)

**Proof** The following is an intuitive explanation of the strategy to optimally allocate sizes of the partitions. Consider the $B$ source symbols as balls and the $\alpha$ partitions as buckets. The capacity of each bucket is $k$. We need to distribute all the balls in the buckets in a manner which satisfies the optimization problem given in (6.30).

Choose a pair of empty buckets. Put $k$ balls in the first bucket and $k-1$ in the second. Continue picking more empty pairs of buckets and filling them in this manner, until you
cannot proceed. Each such pair of buckets consumes $2k - 1$ balls and contributes $k$ to the download bandwidth. The number of buckets used will be $2 \min\left(\left\lfloor \frac{B}{2k-1} \right\rfloor, \left\lfloor \frac{\alpha}{2} \right\rfloor\right)$ If there are any more balls left, then one of the two cases must arise:

**Case 1:** At least one pair of empty buckets is available and the number of balls remaining is less than $2k - 1$ i.e. $\left\lfloor \frac{B}{2k-1} \right\rfloor < \left\lfloor \frac{\alpha}{2} \right\rfloor$.

The number of balls left will be $B \mod (2k-1)$. If this number is even, then distribute the balls equally in the two buckets. If it is odd, then distribute the remaining balls in the two buckets such that the first bucket gets one more than the second. This step contributes $\left\lfloor \frac{B \mod (2k-1)}{2} \right\rfloor + 1$ to the download bandwidth.

**Case 2:** The number of empty buckets remaining is at most 1 i.e. $\left\lfloor \frac{B}{2k-1} \right\rfloor \geq \left\lfloor \frac{\alpha}{2} \right\rfloor$.

The number of balls left will be $B - (2k - 1)\left\lfloor \frac{\alpha}{2} \right\rfloor$. Consider the set of all buckets, and put each remaining ball in any bucket which is not yet full. Each such ball will contribute 1 to the download bandwidth.

The repair bandwidth for any node is given by

$$\gamma = \begin{cases} \left\lfloor k \frac{B}{2k-1} \right\rfloor + \left\lfloor \frac{B \mod (2k-1)}{2} \right\rfloor + 1 & \text{for } \left\lfloor \frac{B}{2k-1} \right\rfloor < \left\lfloor \frac{\alpha}{2} \right\rfloor \\ B - (k - 1)\left\lfloor \frac{\alpha}{2} \right\rfloor & \text{for } \left\lfloor \frac{B}{2k-1} \right\rfloor \geq \left\lfloor \frac{\alpha}{2} \right\rfloor \end{cases}$$

(6.32)

This expression can be simplified to obtain an upper bound on the repair bandwidth as given by (6.31).

Thus the download bandwidth for any node is approximately half the file size.

The repair bandwidths required for regeneration of a failed node for parameters $B = 10000$ and $k = 10$ are plotted for various values of $\alpha$ in Figure 6.2. The graph is a downward step since the repair bandwidth decreases when $\alpha$ increases from odd to even, but remains constant when $\alpha$ increases from even to odd.

It follows from equation (6.32) that if the storage per node is increased beyond a certain threshold, the repair bandwidth does not reduce any further. This threshold is given by

$$\alpha_{\text{max}} = 2 \left\lceil \frac{B}{2k-1} \right\rceil$$

(6.33)

### 6.2.4 Analysis of the code

**Repair bandwidth:** The construction provided in the present chapter reduces the repair bandwidth uniformly for all the nodes in the system to approximately half the file size. To the best of our knowledge, this is the first explicit code to do so for any feasible value of the system parameters.

**Complexity:** As the code is explicit, the construction is immediate provided the field size is greater than $n$. In our construction, the main vectors of the regenerated node are
identical to the main vectors of the failed node. However the auxiliary vector is permitted to be different. We term this as an \textit{approximately exact repair}. Since the matrix inversions performed for solving the linear equations during reconstruction and regeneration depend only on the main vectors, these matrix inversions need to be computed just once. Hence, system maintenance has a low complexity.

\textbf{Field size required:} If we use a Reed-Solomon code as the MDS code in the construction, the minimum field size required is just

\begin{equation}
q \geq n
\end{equation}
Chapter 7

Insights into General Network Coding Problems

In this chapter, we first present the problem of obtaining exact-regenerating MSR codes as a non-multicast network coding problem. Very few results are available for the problem of non-multicast network coding. In [29] it is shown that determining whether a linear network coding solution exists for an arbitrary network is NP-hard. The insufficiency of linear coding for the non-multicast case is shown in [30].

Using the insights obtained in the distributed storage problem, we outline an approach to the task of designing network codes in a non-multicast setting. We make the observation that in this class of networks, the notions of interference alignment and useful information flow are essential. We also show how these concepts can be applied to a larger class of networks as well as provide insight that will aid in code design. It will also be shown to help tighten existing cut-set based upper bounds.

7.1 As a Network Coding Problem

We now present the distributed storage problem as an instance of a non-multicast network coding problem in which the graph of the network is directed, delay free and acyclic. The network is viewed as having $k$ source nodes, each corresponding to a systematic node and generating $\alpha$ symbols each per unit time. The non-systematic nodes are simply viewed as downlink nodes. Since it is only possible to download $\alpha$ symbols from a downlink node, this is taken care of in the graph as in [13], by (i) splitting each non-systematic node $m$ into two nodes: $m_{in}$ and $m_{out}$ with an edge of capacity $\alpha$ linking the two with (ii) all incoming edges arriving into $m_{in}$ and all outgoing edges emanating from $m_{out}$. The sinks in the network are of two types. The first type correspond to data collectors which connect to some collection of $k$ nodes in the network for the purposes of data reconstruction. Hence there are $\binom{n}{k}$ sinks of this type. The second type of sinks represents a new node that
is attempting to duplicate a failed systematic node. Nodes of this type are assumed to connect to the remaining $k - 1$ systematic nodes and any $d - (k - 1)$ of the non-systematic nodes. Hence there are \( \binom{n-k}{d-k+1} \) sinks of this type. The non-multicast network for the parameter set $n = 4, k = 2, d = 3$ is shown in Figure 7.1.

![Figure 7.1: Complete network for the distributed storage problem for $n = 4, k = 2$ and $d = 3$. Unmarked edges have capacity $\alpha$.](image)

Figure 7.2 shows a part of the network for the general problem, and depicts one of the DCs which connects to some $k$ nodes and two possible new nodes corresponding to failure of the first systematic node connecting to two different sets of $d$ existing nodes.

We now introduce some additional notation with respect to this non-multicast network. Let $\mathbf{f}$ be a column vector of length $B$ corresponding to the $B$ symbols produced by the $k$ sources (systematic nodes). Let $\mathbf{f}' = [\mathbf{f}_1 \ldots \mathbf{f}_k]$ where $\mathbf{f}_l$ is an $\alpha$ length row vector corresponding to the $\alpha$ symbols generated by systematic node $l$.

Every edge $e$ is associated with a matrix $M_e$ of dimension $C_e \times B$ where $C_e$ is the capacity of that edge. The rows of the matrix $M_e$ are the $C_e$ global kernels associated with $C_e$ symbols flowing along the edge. The actual symbols carried by the edge is $M_e \mathbf{f}$. Columns $(l-1)\alpha + 1$ to $l\alpha$ of $M_e$ for any $l \in \{1, \ldots, k\}$ are referred to as the columns corresponding to systematic node $l$.

Let tail($e$) and head($e$) be the tail and head vertices of edge $e$ respectively. If the tail($e$) of an edge $e$ is a source node i.e. systematic node $l$ ($\in \{1, \ldots, k\}$) then in $M_e$, the columns corresponding to any other systematic node have to be zero. For any other edge $e$, $M_e$ is a linear combination of matrices $M_{e'}$ $\forall e'$ such that head($e'$) = tail($e$), where
the coefficients of linear combination themselves are matrices.

We have already seen in chapter 4 that the concept of interference alignment can be used to solve the distributed storage problem. We apply these insights to obtain a set of intuitive conditions for code design in a general network of non-multicast type.

### 7.2 Necessary and sufficient conditions for a general network

The theorems stated in this section give a set of necessary and sufficient conditions that need to be satisfied by any linear coding solution to a general network of non-multicast type.

While the theorems on one hand are intuitively obvious, they nevertheless provide a new and useful perspective to the problem of code design and this will be illustrated in the subsequent sections. In general, the viewpoint yields heuristics that aid in code construction and sometimes permit tighter upper bounds on achievable rates than the cut-based bounds.

**Setting and notation:** The subspace framework introduced in Chapter 3 will be used in deriving the conditions. As mentioned earlier we consider delay free, acyclic, directed
7.2 Necessary and sufficient conditions for a general network

graphs. We also assume the networks to be error free. We consider scalar linear network coding solution for these networks. All symbols belong to some finite field $\mathbb{F}_q$. An edge $e$ in the network can carry an integral number of symbols from $\mathbb{F}_q$, and the maximum number of such symbols it can carry at a time is called the capacity of that edge $C_e$.

There are $k$ independent sources $S_1, \ldots, S_k$. Without loss of generality we assume that each sink demands all the information from exactly one source. If a sink demands multiple sources, or a part of a source, then an equivalent network can be constructed by splitting the sink or the source respectively. Also, we assume that there is at least one sink corresponding to every source. A sink is named $T_l$ if it demands source $S_l$. The source and sink nodes do not have any incoming and outgoing edges respectively. Let $R_l$ be the rate of the source $S_l$. An edge from vertex $u$ to $v$ is represented as $u \rightarrow v$.

A cut $\Omega$ (as illustrated in Figure 7.3) is a partition of vertices into two sets, called the source side and the sink side partitions. Edges going across the cut from source side to sink side are said to belong to the cut. The capacity of a cut $\Omega$, denoted by $C(\Omega)$ is the sum of capacities of all the edges in the cut.

For any cut, the set of sources are divided into three sets: $S_D(\Omega)$ is the set of desired sources, i.e. those which are on the source side of the cut and having at least one of its corresponding sinks on the sink side; $S_I(\Omega)$ is the set of interfering sources, i.e. those which are on the source side of the cut and having none of its corresponding sinks on the sink side; and $S_N(\Omega)$ is the set of neutral sources, i.e. those which are on the sink side of the cut. $R_D(\Omega)$, $R_I(\Omega)$ and $R_N(\Omega)$ are the sum rates of the three sets of sources respectively.

Let $R = \sum_{l=1}^{k} R_l$. Each edge is associated with a matrix of dimension $C_e \times R$.

Consider any cut $\Omega$ and an arbitrary set of sources $S$. The dimension of $S$ in the cut, denoted as $\dim(S, \Omega)$ is defined as the dimension of the vector space spanned by the rows of $M_e \forall e \in \Omega$ considering only the columns corresponding to the sources in $S$. Let $E(\Omega)$ for denote the set of all rows of $M_e \forall e \in \Omega$. 

![Figure 7.3: An illustration of a cut $\Omega$ and associated source-sets](image-url)
7.3 Bound for Networks with crosslinks

**Theorem 7.2.1 (Useful Information Flow)** A necessary condition for a code to achieve the rate tuple \((R_1, \ldots, R_k)\) is that for any cut \(\Omega\),

\[
dim(S_D(\Omega), \Omega) \geq R_D(\Omega) \quad (7.1)
\]

**Theorem 7.2.2 (Interference Alignment)** A necessary condition for a code to achieve the rate tuple \((R_1, \ldots, R_k)\) is that for any cut \(\Omega\),

\[
dim(S_I(\Omega), \Omega) \leq C(\Omega) - R_D(\Omega) \quad (7.2)
\]

**Theorem 7.2.3** A necessary and sufficient condition for a code to achieve the rate tuple \((R_1, \ldots, R_k)\) is that for any cut \(\Omega\) there exists a linear transformation \(T\) on \(E(\Omega)\) and \(F \subseteq E(\Omega)\) of size \(R_D\) such that

- The dimension of the vector space spanned by the rows of \(F\) considering only the columns corresponding to the sources in \(S_D\) is \(R_D\).
- The vector space spanned by the rows of \(F\) considering only the columns corresponding to the sources in \(S_I\) is linearly dependent to that in \(E(\Omega) - F\).
- The linear transformation \(T\) on \(E(\Omega)\) results in \(F\) with columns corresponding to the sources in \(S_I\) nulled out and columns corresponding to the sources in \(S_D\) retaining rank \(R_D\).

In the next section, we show how the presence of crosslinks can be used to tighten upper bounds on achievable rates.

### 7.3 Bound for Networks with crosslinks

**Definition 7.3.1 (Crosslink)** A crosslink is an edge from source \(S_i\) to sink \(T_j\) where \(i \neq j\).

Networks with a large number of crosslinks arise in quite a few applications such as peer-to-peer file sharing where nodes simultaneously upload and download parts of files, in distributed storage etc. A well-known example is the butterfly network, whose modified versions are shown in Figure 7.4. Here, \(S_2 \rightarrow T_1\) in Figure 7.4a, and \(S_2 \rightarrow T_1\) and \(S_1 \rightarrow T_2\) in Figure 7.4b are the crosslinks. These crosslinks help to cancel out the interference at the sinks, but do not contribute directly to useful information flow.

**Theorem 7.3.2 (Upper bound for networks with crosslinks)** For any cut \(\Omega\) and any partition of \(S_D(\Omega)\) into \(S_{D1}(\Omega)\) and \(S_{D2}(\Omega)\), let \(\tilde{C}(\Omega)\) be the total capacity of the cut after removing all the crosslinks, then

\[
R_D(\Omega) \leq \tilde{C}(\Omega) + C_{CL}(S_{D1}(\Omega), T_{D2}(\Omega)) \quad (7.3)
\]
where $C_{CL}(S_{D1}(\Omega), T_{D2}(\Omega))$ is the sum capacity of all the crosslinks which originate in $S_{D1}(\Omega)$ and terminate in any sink corresponding to $S_{D2}(\Omega)$.

**Proof** The set of sources $S_{D1}(\Omega)$ act as interference for sinks $T_{D2}(\Omega)$. Hence by Theorem 7.2.2,

$$\dim(S_{D1}(\Omega), \Omega) \leq \tilde{C}(\Omega) - R_{D2}(\Omega) + C_{CL}(S_{D1}(\Omega), T_{D2}(\Omega)) \tag{7.4}$$

Since $S_{D1}(\Omega) \subseteq S_{D}(\Omega)$, by Theorem 7.2.1 we get

$$R_{D1}(\Omega) \leq \dim(S_{D1}(\Omega), \Omega) \tag{7.5}$$

Thus from equations (7.4) and (7.5),

$$R_{D1}(\Omega) \leq \tilde{C}(\Omega) - R_{D2}(\Omega) + C_{CL}(S_{D1}(\Omega), T_{D2}(\Omega)) \tag{7.6}$$

This leads to equation (7.3). ■

**Example 1** Consider the butterfly network in Figure 7.4a. The cut-set bound gives $R_1 \leq 1$, $R_2 \leq 1$ and $R_1 + R_2 \leq 2$. Choose a cut $\Omega$ as shown in the figure. We get $S_{D}(\Omega) = \{S_1, S_2\}$, $C(\Omega) = 2$. Choose $S_{D1}(\Omega) = \{S_1\}$. Hence, $S_{D2}(\Omega) = \{S_2\}$.

$$C_{CL}(S_1(\Omega), T_2(\Omega)) = 0 \tag{7.7}$$

Removing the crosslink $S_2 \to T_1$, we get

$$\tilde{C}(\Omega) = 1 \tag{7.8}$$

Thus from equation (7.3), we get

$$R_1 + R_2 \leq 1 \tag{7.9}$$

Applying this theorem to the remaining cuts and partitions gives $R_1 \leq 1$ and $R_2 \leq 1$. Thus, the cut-set bound is tightened in this case.

**Example 2** Now consider the butterfly network in Figure 7.4b. Here, the cut-set bound and the bound given by Theorem 7.3.2 coincide to give $R_1 \leq 1$, $R_2 \leq 1$ and $R_1 + R_2 \leq 2$, which is achievable.
Figure 7.4: Two modifications of the butterfly network. Each edge has unit capacity.
Chapter 8

A Flexible Class of Regenerating Codes

In the original regenerating codes setup, $B$ units of data is stored across $n$ nodes in the network in such a way that the data can be recovered by connecting to any $k$ nodes. Additionally one can repair a failed node by connecting to any $d$ nodes while downloading at most $\beta$ units of data from each node. In this chapter, we introduce a flexible framework in which the data can be recovered by connecting to any number of nodes as long as the total amount of data downloaded is at least $B$. Similarly, regeneration of a failed node is possible if the new node connects to the network using links whose individual capacity is bounded above by $\beta_{\text{max}}$ and whose sum capacity equals or exceeds a predetermined parameter $\gamma$. In this flexible setting, we obtain the cut-set lower bound on the repair bandwidth along with a constructive proof for the existence of codes meeting this bound for all values of the parameters. An explicit code construction is also provided which is optimal in certain parameter regimes.

8.1 Motivation

In a practical scenario, the storage nodes may be spread out geographically, say over the internet, and may have routes of different capacities between them. The assumption in the original setup, of the DC connecting to only $k$ nodes and a new node connecting to only $d$ nodes with downloading $\beta$ symbols from each is very restrictive in nature. It is desirable to enable the DC or the new node to make use of parallel downloads according to the availability of other storage nodes in its vicinity and this can greatly reduce the total download time [35,36]. Also, the links from the DC or the new node to other nodes may not be symmetric in general; some links may have higher capacities (or lower RTTs) than the other links at a given instant depending on the topology of the network and the prevailing traffic in different parts of the network. This is illustrated in Figure 8.1.
8.2 Flexible Setting Description

In this section, a flexible framework for distributed storage systems is introduced. Data is stored in a distributed manner across $n$ storage nodes, each having a capacity to store $\alpha$ symbols. A DC downloads $\mu_1, \ldots, \mu_n$ symbols from nodes $1, \ldots, n$ respectively. The DC should be able to recover back the entire data for any choice of $\mu_i, \ i = 1, \ldots, n$ satisfying

$$\sum_{i=1}^{n} \mu_i \geq B, \quad 0 \leq \mu_i \leq \alpha.$$  \hspace{1cm} (8.1)

We call this Flexible Reconstruction.

Compared to the original setup of regenerating codes, where the DC is restricted to connect to $k$ nodes, this framework provides a great deal of flexibility to the DC to choose the link capacities with which it wants to connect to each of the nodes. This choice could be based on the network conditions at that instant, and the DC can even possibly connect to all the $n$ nodes.

When a storage node $\ell$ fails, it is replaced by a new node which downloads $\beta_{\ell}$ symbols.
from node $i$, $\forall i = 1, \ldots, n$, $i \neq \ell$ as long as

$$\sum_{i=1 \ (i \neq \ell)}^{n} \beta_i \geq \gamma, \quad 0 \leq \beta_i \leq \beta_{\text{max}}$$

for some value $\gamma$, the repair bandwidth. Here $\beta_{\text{max}}$ is a constant satisfying

$$0 \leq \beta_{\text{max}} \leq \alpha.$$  \hspace{3cm} (8.3)

The new node along with the existing nodes should satisfy the flexible reconstruction property and should be able to participate in the regeneration of any other failed node in the future.

The parameter $\beta_{\text{max}}$ puts a cap on the maximum amount of data that the new node can download from an existing node. The most general setting would be to allow the new node to download any amount of data from the existing nodes, i.e., choosing $\beta_{\text{max}} = \alpha$. However, as it will be shown in Section 8.3.3, it turns out that this is not a wise choice and it results in new node having to download the entire file.

Again, in this flexible framework, the new node has the freedom to choose the link capacities to each node. Unlike in the original regenerating code setup, the new node is not constrained to download equal amounts of data from each node it connects to, and can download non-uniformly depending on the prevailing network conditions. We term this Flexible Regeneration.

Any code satisfying the flexible reconstruction and flexible regeneration properties is called a Flexible Regenerating Code. Note that for any storage system to be feasible, we need

$$\alpha \geq \frac{B}{n}.$$  \hspace{3cm} (8.4)

We assume throughout that this condition is satisfied. We also assume that all system parameters are non-negative integers.

### 8.3 Lower bound on the Repair Bandwidth

In this section we provide a lower bound on the repair bandwidth required to maintain a flexible distributed storage system. As in [12], we model the distributed storage system as an information flow graph. In such a graph, each storage node is modeled in the form of two nodes - an in node and an out node and a link of capacity $\alpha$ connecting the two. This captures the constraint that each node can store only $\alpha$ symbols. Figure 8.2 gives an example of such a network where $S$ is the source producing data at the rate of $B$ symbols per unit time (the data file). The source connects to the $n$ nodes with links having capacities of $\alpha$ symbols each.
8.3 Lower bound on the Repair Bandwidth

8.3.1 Information Flow Graph

On failure of a storage node, say node $\ell$, it is replaced by a new node by connecting nodes out$(j)$, $j \in \{1, \ldots, n\}$, $j \neq \ell$, to in$(\ell)$, with links of capacities $\beta_j$, satisfying equation (8.2). Thus the network evolves through an infinite chain of failures and regenerations. For every instantiation of the network, there can be a different sequence of failures and regenerations with different sets of $\{\beta_j\}$ for each regeneration, and all these instantiations have to be satisfied by a flexible regenerating code.

For reconstruction, at any stage of the network evolution, a DC (sink) can connect to the $n$ existing nodes. This is represented by links of capacities $\mu_j$ from the out nodes $j$ ($= 1, \ldots, n$) to the sink, satisfying equation (8.1). Each DC can connect to the storage nodes with a different set of $\{\mu_j\}$, and all these instantiations too have to be satisfied by a flexible regenerating code.

A lower bound on the repair bandwidth is obtained by bounding the maximum flow in this network.

8.3.2 Cut-set Lower Bound

Any cut $C$ partitions the set of nodes $V$ in the network into $V_C$ and $V_C^c$ where $S \in V_C$ and $DC \in V_C^c$. Consider the set of cuts where $V_C^c$ contains only the DC along with the out parts of some $r$ of the $n$ existing storage nodes. Since the network considered is a directed acyclic graph, nodes in the network can be topologically ordered, as illustrated in Figure 2.

Consider the first storage node in $V_C^c$ in the topological ordering. Call it node $\Lambda_1$. Since out$(\Lambda_1) \in V_C^c$, the cut crosses either the $\alpha$ capacity link between in$(\Lambda_1)$ and out$(\Lambda_1)$ or the set of links with total capacity $\gamma$ entering in$(\Lambda_1)$. We take the cut across the minimum of the two contributing $\min(\alpha, \gamma)$ to the value of the cut.
Consider the next storage node in \( V_C \) in the topological ordering and call it node \( \Lambda_2 \). To decrease the value of the cut, we assume that there is a link of value \( \beta_{\text{max}} \) from \( \text{out}(\Lambda_1) \) to \( \text{in}(\Lambda_2) \) which will not be a part of the cut. Again, the cut will cross either the \( \alpha \) capacity link between \( \text{in}(\Lambda_2) \) and \( \text{out}(\Lambda_2) \) or the set of links with total capacity \((\gamma - \beta_{\text{max}})^+\) entering \( \text{in}(\Lambda_2) \) from nodes in \( V_C \). Again, we take the cut across the minimum of the two contributing \( \min(\alpha, (\gamma - \beta_{\text{max}})^+) \) to the value of the cut.

In general, node \( \Lambda_j \), \( 1 \leq j \leq r \) in \( V_C \) has links with capacities \( \beta_{\text{max}} \) each from \( \text{out}(\Lambda_i) \), \( i = 1, \ldots, j-1 \) which will not be a part of the cut. Hence we take the cut across \( \alpha \) or \((\gamma - (j - 1)\beta_{\text{max}})^+\), whichever is less. The DC connects to these \( r \) nodes and downloads \( \alpha \) symbols each. The rest \((B - r\alpha)^+\) needs to come from nodes in \( V_C \), and hence will be a part of the cut. Thus the value of the cut is

\[
\sum_{j=0}^{r-1} \min(\alpha, (\gamma - j\beta_{\text{max}})^+) + (B - r\alpha)^+ \quad (8.5)
\]

The file size \( B \) has to be smaller than any cut and hence

\[
B \leq \min_{0 \leq r \leq n} \left\{ \sum_{j=0}^{r-1} \min(\alpha, (\gamma - j\beta_{\text{max}})^+) + (B - r\alpha)^+ \right\} \quad (8.6)
\]

**Lemma 8.3.1** A cut-set lower bound on the repair bandwidth \( \gamma \) is given by

\[
\gamma \geq \max(\alpha - \beta_{\text{max}}, B \mod \alpha) + s\beta_{\text{max}} \quad (8.7)
\]

**Proof**

Define \( s = \lfloor B/\alpha \rfloor \).

For \( r = s \), the equation (8.6) reduces to

\[
B \leq \sum_{j=0}^{s-1} \min(\alpha, (\gamma - j\beta_{\text{max}})^+) + (B - s\alpha) \quad (8.9)
\]

\[
s\alpha \leq \sum_{j=0}^{s-1} \min(\alpha, (\gamma - j\beta_{\text{max}})^+) \quad (8.10)
\]

\[
\leq s\alpha \quad (8.11)
\]

Thus, \((\gamma - j\beta_{\text{max}})^+ \geq \alpha \quad \forall \ j \in \{0, \ldots, s-1\} \quad (8.12)\]

which gives a lower bound on the repair bandwidth as

\[
\gamma \geq \alpha + (s - 1)\beta_{\text{max}} \quad (8.13)
\]
For \( r = s + 1 \), equation (8.6) gives

\[
B \leq \sum_{j=0}^{s} \min(\alpha, (\gamma - j\beta_{\text{max}})^+) \tag{8.14}
\]

\[
= s\alpha + \min(\alpha, (\gamma - s\beta_{\text{max}})^+) \tag{8.15}
\]

where (8.15) is due to (8.13). This evaluates to

\[
\gamma \geq B - s\alpha + s\beta_{\text{max}} \tag{8.16}
\]

Combining (8.13) and (8.16) we get

\[
\gamma \geq \max(\alpha - \beta_{\text{max}}, B \mod \alpha) + s\beta_{\text{max}}. \tag{8.17}
\]

\section*{8.3.3 Complete Flexibility (\( \beta_{\text{max}} = \alpha \)) ?}

An obvious question in the flexible framework is, how much can we optimize the repair bandwidth if we give complete freedom to the new node replacing a failed node, i.e. allowing \( \beta_{\text{max}} = \alpha \). The answer to this is obtained by substituting \( \beta_{\text{max}} = \alpha \) in equation (8.17). This gives \( \gamma \geq B \), i.e., the repair bandwidth is equal to the size of the entire file.

\section*{8.4 Achievability}

In this section, we prove the existence of a linear flexible regenerating code which meets the lower bound on the repair bandwidth given by Lemma 8.3.1. We prove the existence of a linear code where any DC connecting to node \( i \) with a link of capacity \( \mu_i, \forall i = 1, \ldots, n \) with

\[
\sum_{i=1}^{n} \mu_i = B, \quad 0 \leq \mu_i \leq \alpha \tag{8.18}
\]

can recover the data, and any failed node \( \ell \) can be regenerated by downloading \( \beta_i \) symbols from node \( i (i = 1, \ldots, n, i \neq \ell) \) with

\[
\sum_{i=1, i \neq \ell}^{n} \beta_i = \gamma, \quad 0 \leq \beta_i \leq \beta_{\text{max}} \tag{8.19}
\]

where \( \gamma \) meets the lower bound on the repair bandwidth given by Lemma 8.3.1. Then, it is clear that any DC with \( \sum_{i=1}^{n} \mu_i > B \) can recover the data, and any failed node with \( \sum_{i=1, i \neq \ell}^{n} \beta_i > \gamma \) can be regenerated.
Define a vector \( f \) of length \( B \), consisting of the source symbols. Each source symbol can independently take values from \( \mathbb{F}_q \), a finite field of size \( q \).

Any stored symbol is written as \( u^t f \) for some \( B \)-length vector \( u \) which corresponds to the global kernel of this stored symbol. These global kernels for the stored symbols define the code, and the actual symbols stored depend on the instantiation of \( f \). Since a node stores \( \alpha \) symbols, it can be considered as storing \( \alpha \) vectors of the code, and hence can be represented by a \( \alpha \times B \) matrix. We will say that the node stores this matrix.

**Lemma 8.4.1** For any set of \( \mu_i, i = 1, \ldots, n \) and \( \beta_i, i = 2, \ldots, n \),
\[
\sum_{i=2}^{n} \min(\alpha, \mu_i + \beta_i) \geq B
\]
subject to the conditions
\[
\sum_{i=2}^{n} \mu_i \geq B - \mu_1, \quad 0 \leq \mu_j \leq \alpha \quad \forall j = 1, \ldots, n
\]
\[
\sum_{i=2}^{n} \beta_i \geq \gamma, \quad 0 \leq \beta_j \leq \beta_{\text{max}} \quad \forall j = 2, \ldots, n
\]
where \( \gamma \) meets the cut-set bound given in equation (8.17) with equality.

**Proof (Sketch)** It can be shown that the given term attains its minimum when \( \mu_i(i = 1, \ldots, n) \) and \( \beta_i(i = 2, \ldots, n) \) have the most skewed distribution, i.e. \( \mu_i = \alpha \) for \( i = 1, \ldots, s, \mu_{s+1} = B \mod \alpha \) and \( \mu_i = 0 \) elsewhere; \( \beta_i = \beta_{\text{max}} \) for \( i = 2, \ldots, \left\lfloor \frac{\gamma}{\beta_{\text{max}}} \right\rfloor \), \( \beta_i = \gamma \mod \beta_{\text{max}} \) for \( i = \left\lfloor \frac{\gamma}{\beta_{\text{max}}} \right\rfloor + 1 \), and zeros elsewhere. For these values of \( \mu_i \) and \( \beta_i \), using the conditions given by equations (8.18) and (8.19), it can be shown that, this term is lower bounded by \( B \).

The complete proof is relegated to Appendix B

The following Lemmas show that given a system which can achieve flexible reconstruction at a particular stage, then upon failure of a node, it can be regenerated such that the system retains the flexible reconstruction property, while meeting the cut-set bound with equality.

**Lemma 8.4.2** Suppose flexible reconstruction is satisfied for all DCs in the present stage. Suppose node \( \ell \) fails and is replaced by a new node. Given a particular DC, in the next stage, i.e. after \( \ell \) is regenerated, connecting to node \( i \) with a link of capacity \( \mu_i \), \( \forall i = 1, \ldots, n \), satisfying constraints given in equation (8.18), the new node can download \( \beta_i \) symbols from node \( i \) \( (i = 1, \ldots, n, \ i \neq \ell) \) satisfying the constraints given in (8.19) and store \( \alpha \) symbols such that this DC is satisfied.
Proof The main idea is to show that the a DC in the future stage (i.e., after regeneration of node $\ell$) is equivalent to some DC connecting to the nodes in the present stage (before failure of node $\ell$), satisfying (8.18). As flexible reconstruction is satisfied for all DCs in the present stage, this would imply that the given DC will also be satisfied. Without loss of generality assume that the first node fails and is regenerated i.e., $\ell = 1$.

If $\mu_1 = 0$ then this DC is trivially satisfied as flexible reconstruction is possible for all DCs in the present stage. Hence consider

$$\mu_1 > 0$$ \hspace{1cm} (8.21)

Since $\mu_i$ and $\beta_i$ satisfy the conditions of Lemma 8.4.1,

$$\sum_{i=2}^{n} \min(\alpha, \mu_i + \beta_i) \geq B .$$ \hspace{1cm} (8.22)

Now, reduce the values of $\beta_i$ to $\beta_i'$, $\forall i = 2, \ldots, n$ such that equality is attained above, i.e.

$$\sum_{i=2}^{n} \min(\alpha, \mu_i + \beta_i') = B$$ \hspace{1cm} (8.23)

Thus we have $\beta_i'$ point-wise lesser than $\beta_i$

$$0 \leq \beta_i' \leq \beta_i \hspace{0.5cm} \forall i = 2, \ldots, n$$ \hspace{1cm} (8.24)

Consider a virtual DC in the present stage connecting to node $i$ with links of capacity $\tilde{\mu}_i$, $\forall i = 2, \ldots, n$ given by

$$\tilde{\mu}_i = \begin{cases} 0 & i = 1 \\ \min(\alpha, \mu_i + \beta_i') & i = 2, \ldots, n \end{cases}$$ \hspace{1cm} (8.25)

This is a valid DC since $0 \leq \tilde{\mu}_i \leq \alpha \hspace{0.5cm} \forall i$ and

$$\sum_{i=1}^{n} \tilde{\mu}_i = \sum_{i=2}^{n} \min(\alpha, \mu_i + \beta_i') = B$$ \hspace{1cm} (8.26)

Consider each node passing $\mu_i$ out of the $\tilde{\mu}_i$ symbols directly to the DC and the remaining $(\tilde{\mu}_i - \mu_i)$ symbols via the new node.

In the real scenario, this means that the new node downloads the $(\tilde{\mu}_i - \mu_i)$ symbols which are flowing through the new node in the virtual case, from each existing node $i$. 
This is a valid regeneration process since for all \( i = 2, \ldots, n, \)

\[
(\tilde{\mu}_i - \mu_i) = \min(\alpha, \mu_i + \beta'_i) - \mu_i \tag{8.27}
\]

\[
\leq \beta'_i \tag{8.28}
\]

\[
\leq \beta_i \tag{8.29}
\]

Hence, what the new node is downloading is point-wise lesser than \( \beta_i \). This just means that, to satisfy the given DC we utilize only a part of the link capacity \( \beta_i \) that is available from node \( i \).

Also, the number of symbols downloaded by the virtual DC through the new node is

\[
\sum_{i=2}^{n} (\tilde{\mu}_i - \mu_i) = \sum_{i=2}^{n} \tilde{\mu}_i - \sum_{i=2}^{n} \mu_i \tag{8.30}
\]

\[
= B - (B - \mu_1) \tag{8.31}
\]

\[
= \mu_1 \tag{8.32}
\]

Thus the given DC becomes equivalent to the virtual DC. Since flexible reconstruction is satisfied for any DC in the present stage, the virtual DC can recover all the data. This implies that the given DC can also recover all the data. ■

**Lemma 8.4.3** Suppose flexible reconstruction is satisfied for all DCs at the present stage. Suppose node \( \ell \) fails and is replaced by a new node. The new node can download \( \beta_i \) symbols from node \( i \) \( (i = 1, \ldots, n, \ i \neq \ell) \) satisfying the constraints given in (8.19) and store \( \alpha \) symbols such that all DCs satisfying (8.18) are simultaneously satisfied, provided the field size is large enough.

**Proof** Without loss of generality assume \( \ell = 1 \). Let \( \mathbf{G}^{(1)}, \cdots, \mathbf{G}^{(n)} \) be the node matrices at the present stage where flexible reconstruction is satisfied for all DCs. Let \( \tilde{\mathbf{G}}^{(1)} \) be the matrix stored in the new node replacing node 1.

The new node downloads \( \beta_i \) symbols from node \( i \) \( (i = 2, \ldots, n) \) and stores \( \alpha \) linear combinations of the symbols downloaded. Thus,

\[
\tilde{\mathbf{G}}^{(1)} = \mathbf{Z} \begin{bmatrix} \mathbf{V}^{(2)} \mathbf{G}^{(2)} \\ \mathbf{V}^{(3)} \mathbf{G}^{(3)} \\ \vdots \\ \mathbf{V}^{(n)} \mathbf{G}^{(n)} \end{bmatrix} \tag{8.33}
\]

where \( \mathbf{V}^{(i)} \) is \( \beta_i \times \alpha \) matrix representing the linear combinations used by node \( i \) to compute the \( \beta_i \) symbols that it passes to the new node. \( \mathbf{Z} \) is \( \alpha \times \gamma \) matrix representing the linear transformation that the new node performs on the downloaded symbols to compute the \( \alpha \) symbols that it stores.
Consider a DC \( \Delta \) connecting to the nodes (after regeneration of node 1) with link capacities satisfying equation (8.18). Every node \( i \) uses a \( \mu_i \times B \) matrix \( U^{(i)}_{\Delta} \) to compute the linear combinations to be passed to this DC. Thus, for the DC to be able to recover the data, we need

\[
P_{\Delta} = \det \begin{bmatrix} U^{(1)}_{\Delta} \tilde{G}^{(1)} \\
U^{(2)}_{\Delta} G^{(2)} \\
\vdots \\
U^{(n)}_{\Delta} G^{(n)} \end{bmatrix} \neq 0
\] (8.34)

The above determinant can also be viewed as a polynomial \( P_{\Delta} \) in \( \mathbb{F}_q \) with entries of the matrices \( U^{(i)}_{\Delta} \) (\( i = 1, \ldots, n \)), \( V^{(i)} (i = 2, \ldots, n) \) and \( Z \) as variables. By Lemma 8.4.2 we know that the DC \( \Delta \) can be satisfied, i.e. there exist values of the variables such that the above determinant is non-zero. Thus the polynomial in (8.34) is a non-zero polynomial.

For all DCs to be satisfied simultaneously, we need

\[
\prod_{\Delta \text{over all DCs}} P_{\Delta} \neq 0
\] (8.35)

This product is itself a polynomial with variables being entries of the matrices \( U^{(i)}_{\Delta} \) (\( i = 1, \ldots, n, \) all DCs \( \Delta \)), \( V^{(i)} (i = 2, \ldots, n) \) and \( Z \). Since each polynomial in this product is non-zero, the product polynomial is also non-zero. Hence, the Schwartz-Zippel Lemma (given in Section 4.6.1) implies that there is an assignment to variables such that equation (8.35) is satisfied, provided the field size is large enough. ■

**Theorem 8.4.4 (Existence of Flexible Regenerating Codes)** Given any set of system parameters \( (n, B, \alpha, \beta_{\text{max}}) \), there exists a linear flexible regenerating code satisfying the lower bound on the repair bandwidth \( \gamma \) provided that the size of the finite field is large enough.

**Proof** The proof is by induction. Initialize the \( n\alpha \) symbols in the nodes with an \([n\alpha, B]\)-MDS code. This clearly satisfies the flexible reconstruction property. Lemma 8.4.3 implies that when a node fails, it can be regenerated such that flexible reconstruction property is retained, while satisfying the cut-set bound with equality. Hence the code maintains flexible reconstruction property after any number of node regenerations if the field size is large enough. ■

**Comparison of the bound with the original regenerating codes setup** At the MSR point in the original setup, \( \alpha = (d - k + 1)\beta \) and the cut-set bound on the repair bandwidth given in equation (1.3) evaluates to

\[
d\beta \geq \alpha + (k - 1)\beta .
\] (8.36)
For the same parameters as that of the MSR point with $\beta_{max} = \beta$, in the flexible regenerating codes setting, we get

$$s = k$$

(8.37)

and thus

$$\gamma \geq \alpha + (k - 1)\beta$$

(8.38)

Thus, the repair bandwidth required in both the settings are identical. Thus there is no additional penalty in making the system flexible at the MSR point.

At other parameter values, the repair bandwidth required in the flexible case is higher. An intuitive explanation for this is that, there is very less redundancy in the system in flexible case, as any set of $\lfloor B/\alpha \rfloor$ nodes need to store ($\lfloor B/\alpha \rfloor \alpha - B$) linearly independent symbols to support flexible reconstruction.

## 8.5 An Explicit Code

In this section, we provide an explicit construction for flexible regenerating codes, which based on the codes constructed in Section 6.2. First an example is provided followed by the general construction.

### 8.5.1 An Example

Consider the parameters $n = 6, \alpha = 4, \beta_{max} = 2, B = 12$. This gives $s = 3$ and a lower bound on the repair bandwidth as $\gamma \geq 8$. Divide the $B = 12$ data symbols into $\alpha = 4$ sets, represented by the vectors $f_1, g_1, f_2$ and $g_2$, each of length 3. Let $v^{(i)} (i = 1, \ldots, 6)$ be 6 vectors of length $s = 3$, forming an 3-dimensional MDS code over $\mathbb{F}_q$. Also, for $i = 1, \ldots, 6$ let $z_1^{(i)}$ and $z_2^{(i)}$ be arbitrary vectors of length 3.

**Code:** Node $i$ ($i = 1, \ldots, 6$) stores the following 4 symbols, one symbol corresponding to each of the 4 sets:

<table>
<thead>
<tr>
<th>Vector (set)</th>
<th>Symbol stored</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>$\frac{f_1^t v^{(i)}}{\lambda_1}$</td>
</tr>
<tr>
<td>$g_1$</td>
<td>$\frac{g_1^t v^{(i)}}{\lambda_2} + \frac{f_1^t z_1^{(i)}}{\lambda_1}$</td>
</tr>
<tr>
<td>$f_2$</td>
<td>$\frac{f_2^t v^{(i)}}{\lambda_2}$</td>
</tr>
<tr>
<td>$g_2$</td>
<td>$\frac{g_2^t v^{(i)}}{\lambda_2} + \frac{f_2^t z_2^{(i)}}{\lambda_2}$</td>
</tr>
</tbody>
</table>

**Flexible Reconstruction:** Suppose a DC connects to the 6 nodes with link capacities $\mu = [3, 1, 1, 1, 2, 4]$. DC needs 3 symbols from each of the 4 sets. Consider node $i$ passing
8.5 An Explicit Code

\( \mu_i \) symbols corresponding to the sets

\[
\left( \sum_{j=1}^{i-1} \mu_j + 1 \text{ to } \sum_{j=1}^{i-1} \mu_j + \mu_i \right) \mod B .
\]

In the example, the symbols passed by the nodes to the DC are

<table>
<thead>
<tr>
<th>Node</th>
<th>Symbols passed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( f^t_1 v^{(1)} , \ g^t_1 v^{(1)} + f^t_1 z^{(1)} , \ f^t_2 v^{(1)} )</td>
</tr>
<tr>
<td>2</td>
<td>( g^t_2 v^{(2)} + f^t_3 z^{(2)} )</td>
</tr>
<tr>
<td>3</td>
<td>( f^t_1 v^{(3)} )</td>
</tr>
<tr>
<td>4</td>
<td>( g^t_1 v^{(4)} + f^t_1 z^{(4)} )</td>
</tr>
<tr>
<td>5</td>
<td>( f^t_1 v^{(5)} , \ g^t_1 v^{(5)} + f^t_2 z^{(6)} )</td>
</tr>
<tr>
<td>6</td>
<td>( f^t_1 v^{(6)} , \ g^t_1 v^{(6)} + f^t_1 z^{(6)} , \ f^t_3 v^{(6)} , \ g^t_2 v^{(6)} + f^t_2 z^{(6)} )</td>
</tr>
</tbody>
</table>

Since \( \bar{v}^{(i)} \)'s form a 3 dimensional MDS code, the DC can use the symbols downloaded to decode \( f_1 \) and \( f_2 \). Then the DC subtracts out the terms \( f^t_j \bar{z}^{(i)} \) from the other symbols, which leaves it with 3 symbols of the form \( g^t_2 v^{(i)} \) and 3 symbols of the form \( g^t_2 v^{(i)} \). Again the MDS property of \( \bar{v}^{(i)} \)'s ensures that the values of \( g_1 \) and \( g_2 \) can also be decoded.

**Flexible Regeneration:** Suppose node 1 fails. The new node replacing it can download at most \( \beta_{\text{max}} = 2 \) symbols from any existing node, while downloading \( \gamma = 8 \) symbols in total. Hence, it can obtain 4 symbols which are linear combinations of \( f_1 \) and \( g_1 \), and the remaining 4 as linear combinations of \( f_2 \) and \( g_2 \). The existing nodes can pass these linear combinations in such a way that the first 4 symbols can be combined to obtain \( f^t_1 v^{(1)} \) and \( g^t_1 v^{(1)} + f^t_1 z^{(1)} \), and the remaining 4 symbols can be combined to obtain \( f^t_2 v^{(1)} \) and \( g^t_2 v^{(1)} + f^t_2 z^{(1)} \). Here \( z_1^{(1)} \) and \( z_2^{(1)} \) are not constrained to be equal to \( z_1^{(1)} \) and \( z_2^{(1)} \) since flexible reconstruction and regeneration operations are carried out irrespective of the values of these vectors.

### 8.5.2 General Form of the Code

The following is the general explicit construction of flexible regenerating codes for the parameters for which \( B \) is a multiple of \( \alpha \) and with

\[
\beta_{\text{max}} = \left\lceil \frac{\alpha}{2} \right\rceil \quad (8.39)
\]

Define,

\[
\theta = \left\lfloor \frac{\alpha}{2} \right\rceil \quad (8.40)
\]
Partition the $B$ source symbols into $\alpha$ sets of $s$ elements each. Let these sets correspond to the elements of the $s$ length vectors $f_1, g_1, \ldots, f_\theta, g_\theta, f_{\theta+1}$, where $f_{\theta+1}$ is the zero vector if $\alpha$ is even.

Let $v_i^{(i)}$ ($i = 1, \ldots, n$) be $n$ vectors of length $s$ forming an $s$-dimensional MDS code over $\mathbb{F}_q$. For $i = 1, \ldots, n$, $j = 1, \ldots, \theta + 1$ let $z_j^{(i)}$ be arbitrary vectors of length $s$.

**The Code**  Node $i$ ($\in \{1, \ldots, n\}$) stores one symbol corresponding to each of the $\alpha$ sets as follows.

<table>
<thead>
<tr>
<th>Vector (set)</th>
<th>Symbol stored</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>$f_1^{(i)}$</td>
</tr>
<tr>
<td>$g_1$</td>
<td>$g_1^{(i)} + f_1^{(1)}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$f_\theta$</td>
<td>$f_\theta^{(i)}$</td>
</tr>
<tr>
<td>$g_\theta$</td>
<td>$g_\theta^{(i)} + f_\theta^{(1)}$</td>
</tr>
<tr>
<td>$f_{\theta+1}$</td>
<td>$f_{\theta+1}^{(i)}$</td>
</tr>
</tbody>
</table>

where the symbol corresponding to $f_{\theta+1}$ is present only when $\alpha$ is odd.

Flexible reconstruction and regeneration can be clearly achieved by following similar steps as described in the example and Section 6.2.

**Repair Bandwidth:** Following exactly the same steps used to calculate the repair bandwidth in Section 6.2.3, we can obtain the repair bandwidth of this code as

$$\gamma = (s + 1) \left\lfloor \frac{\alpha}{2} \right\rfloor + s(\alpha \mod 2)$$

(8.41)

Note that the code meets the cut-set bound when the parameter values are such that $B$ is a multiple of $\alpha$ and $\beta_{\text{max}} = \left\lceil \frac{\alpha}{2} \right\rceil$.

### 8.6 Symbol-wise MDS Codes and Error-prone Links

Up till now, all the links in the distributed storage system were assumed to be error-free. Now we consider the case where links from the nodes to a data collector are prone to errors. In such a scenario, it is desirable to have the property the data collector can request for additional symbols from the storage nodes in order to perform error detection or correction. Further, the quality of the links may vary from one data collector to another, or possibly over time for a single data collector. In such a case, it is desirable for the data collector to have the freedom to choose the number of additional symbol downloads.
These properties can be easily satisfied if the $n\alpha$ symbols stored in the storage nodes form a $[n\alpha, B]$ symbol-wise MDS code, i.e., where any $B$ out of the $n\alpha$ symbols suffice for the purpose of reconstruction.

Clearly, any regenerating code that is symbol wise MDS also satisfies the flexibility conditions outlined in this chapter, and its repair bandwidth is necessarily lower bounded by equation (8.7).

In this section, we prove a non-trivial property pertaining to flexible regenerating codes, that any code that can perform flexible reconstruction can be converted to a symbol-wise MDS code, via a change of basis, provided the field size is large enough. In proving this, we use the notation of global kernels and generator matrices introduced in Section 4.1 and also the concept of equivalent codes in Section 4.4.2.

**Theorem 8.6.1** Any code that can perform flexible reconstruction can be converted to a symbol-wise MDS code provided the field size is large enough.

**Proof** At any point in time, denote the generator matrices of the $n$ storage nodes as $G_1, \ldots, G_n$, each having dimensions of $B \times \alpha$. Apply subspace transformations to each of the nodes via $\alpha \times \alpha$ matrices $A_1, \ldots, A_n$ such that the new generator matrices of the nodes become $G_1A_1, \ldots, G_nA_n$. Now, we have to assign values to the matrices $A_1, \ldots, A_n$ such that the code becomes a symbol-wise MDS code.

The necessary and sufficient condition for a code to be symbol-wise MDS is that any $B$ symbols should suffice to recover the entire source of size $B$. In other words, the global kernels corresponding to any $B$ symbols need to be linearly independent.

From the $n\alpha$ symbols stored in the nodes, choose some $\mu_1, \ldots, \mu_n$ symbols from nodes $1, \ldots, n$ respectively, with $\sum_{i=1}^{n} \mu_i = B$. Let $E_i$, $(i = 1, \ldots, n)$ be selection matrices, of sizes $(\alpha \times \mu_i)$ respectively, where each column is an unit vector, and each row has at most one 1. Thus the $B$ global kernels obtained by the data collector are given by:

$$\{G_1A_1E_1, \ldots, G_nA_nE_n\}.$$ \hspace{1cm} (8.42)

Thus, we need the following $(B \times B)$ matrix to be non-singular:

$$[G_1A_1E_1 \cdots G_nA_nE_n].$$ \hspace{1cm} (8.43)

Now, since the code can perform flexible reconstruction, we known that the nodes can provide $\mu_1, \ldots, \mu_n$ symbols respectively such that reconstruction is satisfied. Thus, there exist values for the entries of the matrices $A_1, \ldots, A_n$ which makes the matrix in equation (8.43) non-singular.

However, we need to find values of $A_1, \ldots, A_n$ such that the $(B \times B)$ matrices for all permissible values of $E_1, \ldots, E_n$ are non-singular. Since each such individual matrix is non-singular, the Schwartz-Zippel lemma (given in Section 4.6.1) guarantees values for the entries of the matrices $A_1, \ldots, A_n$ satisfying each of these conditions simultaneously.
Upon node failure and subsequent regeneration, the process outlined above is repeated, and the code can be converted to an equivalent symbol-wise MDS via a change of basis.\footnote{The symbols in the existing nodes may not remain symbol-wise MDS with the symbols in the replacement node. Hence even the existing nodes need to undergo a change of basis.}

Note that the requirement of a minimum field size is not just a requirement of this proof, as seen in the following example. Let $n = 3$, $B = 4$, $\alpha = 2$. We construct a flexible reconstructing code over the binary field $\mathbb{F}_2$. Let the four source symbols be $u_1, \ldots, u_4$. The three node store the symbols

Node 1: $u_1, u_2$
Node 2: $u_3, u_4$
Node 3: $u_1 + u_3, u_2 + u_4$.

Clearly, this code is capable of performing flexible reconstruction. However, it cannot be a symbol-wise MDS code since no $[6, 4]$ MDS code exists over $\mathbb{F}_2$. 
Chapter 9

Conclusion

The work presented in this report mainly deals with explicit construction of regenerating codes for distributed storage, and the achievability or otherwise of the storage-repair bandwidth tradeoff under the practically relevant exact regeneration scenario.

A Product-Matrix framework is introduced using which explicit exact-regenerating MBR codes for all feasible values of the parameters \((n, k, d)\) and explicit exact-regenerating MSR codes for \((n, k, d \geq 2k - 2)\) are constructed. In comparison to all existing constructions of exact-regenerating codes in the literature which restrict the value of \(n\) to be \(d + 1\), both the MBR and the MSR codes presented here are applicable for all values of \(n\), independent of the values of the parameters \(k\) and \(d\). This makes the code constructions practically appealing. Also, the product-matrix structure of these codes leads to a host of implementation advantages.

The traditional regenerating codes setup necessitates a compromise in either the storage space or the repair bandwidth; both cannot be minimized simultaneously. However, in this report, we presented an ideal regenerating code which achieves both minimum storage and minimum repair bandwidth simultaneously, by relaxing certain conditions in the original setup. An explicit construction is also provided via the product-matrix framework.

An open problem in the area of regenerating codes is the question of whether the tradeoff is achievable under exact regeneration. Using a subspace based approach we proved that the interior points of the tradeoff are not achievable under exact regeneration, except for a region of width at most \(\beta\) in the immediate vicinity of the MSR point. On the other hand, the product-matrix based MBR and MSR codes prove that the MBR point can be achieved under exact regeneration for all feasible values of the parameters and the MSR point for all the parameters satisfying \((2k - 2 \leq d \leq n - 1)\). Achievable values for the repair bandwidth at the interior points are obtained using storage space sharing between the MBR and MSR codes.

Interference alignment is a concept recently introduced in the context of wireless communication. The necessity of interference alignment for any exact regenerating MSR code
is proved and subsequently, non-achievability of the MSR point under exact regeneration for the parameters satisfying $d < 2k - 3$ for the atomic case of $\beta = 1$ is established. An explicit code structure, termed the MISER code, guaranteeing exact repair of the systematic nodes at the MSR point for $d = (n - 1) \geq 2k - 1$ is presented; the code is again based on the concept of interference alignment. An achievable scheme minimizing the storage space is also presented, though not explicit, for exact regeneration of systematic nodes. This scheme is optimal for $d \geq 2k - 1$.

The parameter set $(n, k, d = k + 1)$ at the MSR point falls in the non-achievable region for the atomic case. For this parameter set, explicit approximately exact-regenerating MSR codes are presented. Using this code as basis, an explicit code is constructed, for all values of the parameters which uniformly reduces the repair bandwidth to approximately half the file size.

An exact-regenerating MBR code for all parameters $(n, k, d = n - 1)$ is constructed, that has a simple graphical description, and when specialized to the parameter set $(n, k = n - 2, d = n - 1)$, can operate over the binary field using XOR operations only.

Insights obtained in constructing codes for distributed storage is translated to insights into coding for general non-multicast networks.

Finally, a new class of regenerating codes termed flexible regenerating codes is introduced, which is much less restrictive as compared to the original setting: this class of codes allow a DC or a replacement node to freely connect to an arbitrary number of nodes, downloading arbitrary amounts of data from each, provided that a set of feasibility conditions are met. An explicit construction is also provided for a certain parameter regime. It is established that any code satisfying flexible reconstruction is a symbol wise MDS code, if the field size is large enough. This result is applicable to the case when the links are prone to errors.
Appendix A

Subspace Properties of Linear Exact Regenerating Codes: Proofs

Proof of Property 1 Consider a data collector connecting to any $k$ nodes, say $\Lambda_1, \ldots, \Lambda_k$, and let

$$|W_{\Lambda_i}| = \Omega_i, \quad \forall i \in \{1, \ldots, n\}. \quad (A.1)$$

Since a node can store a subspace of dimension no more than $\alpha$,

$$\Omega_i \leq \alpha, \quad \forall i \in \{1, \ldots, k\}. \quad (A.2)$$

Next, for some $l \in \{2, \ldots, k\}$, consider the scenario wherein nodes $\Lambda_1, \ldots, \Lambda_{l-1}$ and some other $(d-(l-1))$ nodes participate in the regeneration of node $\Lambda_l$. The maximum number of linearly independent vectors that the $(d-(l-1))$ nodes (other than $\Lambda_1, \ldots, \Lambda_{l-1}$) can contribute is $(d-(l-1))\beta$. If this quantity is less than $\Omega_l$, then it falls upon nodes $\Lambda_1, \ldots, \Lambda_{l-1}$ to provide the remaining dimensions to node $\Lambda_l$. Thus for $l = 2, \ldots, k$

$$|W_{\Lambda_l} \cap \{W_{\Lambda_{l-1}} + \cdots + W_{\Lambda_1}\}| \geq (\Omega_l - (d-(l-1))\beta)^+ \quad (A.3)$$

where $(x)^+$ stands for $\max(x, 0)$.

For the data collector to be able to reconstruct the data, dimension of the sum of the nodal subspaces of nodes $\Lambda_1, \ldots, \Lambda_k$ should be $B$, i.e.,

$$|W_{\Lambda_1} + W_{\Lambda_2} + \cdots + W_{\Lambda_k}| = B \quad (A.4)$$
Using the expression for the dimension of sum of two subspaces recursively we get,

\[ B = |W_{\Lambda_1} + \cdots + W_{\Lambda_k}| \]
\[ = |W_{\Lambda_1}| + |W_{\Lambda_2}| - |W_{\Lambda_1} \cap W_{\Lambda_2}| \]
\[ + |W_{\Lambda_3}| - |W_{\Lambda_3} \cap \{W_{\Lambda_1} + W_{\Lambda_2}\}| \]
\[ \vdots \]
\[ + |W_{\Lambda_k}| - |W_{\Lambda_k} \cap \{W_{\Lambda_1} + \cdots + W_{\Lambda_{k-1}}\}|. \quad (A.5) \]

Applying the inequality (A.3) in equation (A.5) gives

\[ B \leq \Omega_1 + \sum_{i=2}^{k} \left[ \Omega_i - (\Omega_i - (d - (i - 1))\beta)^+ \right] \quad (A.6) \]
\[ = \Omega_1 + \sum_{i=2}^{k} \min \left( \Omega_i, (d - (i - 1))\beta \right) \quad (A.7) \]
\[ \leq \alpha + \sum_{i=2}^{k} \min \left( \alpha, (d - (i - 1))\beta \right) \quad (A.8) \]
\[ = \alpha + \sum_{i=1}^{k-1} \min \left( \alpha, (d - i)\beta \right) \quad (A.9) \]
\[ = \sum_{i=0}^{k-1} \min \left( \alpha, (d - i)\beta \right) \quad (A.10) \]
\[ = B. \quad (A.11) \]

Here, equation (A.7) follows from the property that any two non-negative numbers \(y_1\) and \(y_2\) satisfy \((y_1 - (y_1 - y_2)^+) = \min (y_1, y_2)\). Equation (A.8) follows from (A.2), and (A.10) follows from the range of \(\alpha\) given in (3.6).

Now, for equation (A.4) to hold, (A.8) should be satisfied with equality, which forces \(\Omega_1 = \alpha\). Since the choice of node \(\Lambda_1\) was arbitrary, an identical argument can be used to prove the same for all nodes, i.e., \(\Omega_i = \alpha, \forall i \in \{1, \ldots, n\}\).

**Proof of Property 2** For \(a \geq k\) the result is trivially true since the nodal subspaces of any \(k\) nodes span the entire space and \(|W_l| = \alpha\).

Now, for the case of \(a < k\), consider a data collector connecting to \(k\) nodes \(\Lambda_1, \ldots, \Lambda_k\), of which the nodes \(\Lambda_1, \ldots, \Lambda_a\) are the \(a\) nodes in the set \(A\) and \(\Lambda_{a+1} = l\). For the data collector to be able to reconstruct all the data, the dimension of the sum of the nodal subspaces of \(\Lambda_1, \ldots, \Lambda_k\) should be \(B\), i.e.,

\[ |W_{\Lambda_1} + W_{\Lambda_2} + \cdots + W_{\Lambda_k}| = B. \quad (A.12) \]
Turning our attention again to equation (A.3), we get

\[
\left| W_{\Lambda_{a+1}} \cap \sum_{i=1}^{a} W_{\Lambda_i} \right| \geq \left[ \left| W_{\Lambda_{a+1}} \right| - (d - a)\beta \right]^+.
\]  

(A.13)

Substituting the value of \( \left| W_{\Lambda_{a+1}} \right| = \alpha = (d - p)\beta - \theta \) (from Property 1) we get, for \( 1 \leq a < k \),

\[
\left| W_{\Lambda_{a+1}} \cap \sum_{i=1}^{a} W_{\Lambda_i} \right| \geq \begin{cases} 
0 & a \leq p \\
(a - p)\beta - \theta & p < a < k
\end{cases}
\]

(A.14)

Now, following the same steps as in Property 1, it follows that for equation (A.12) to hold, the inequality (A.14) needs to be satisfied with equality.

\[\text{Proof of Property 3} \]

First we consider the case of \( p < k - 1 \), i.e., all points except the MSR point.

Consider exact regeneration of node \( l \). Partition the \( d \) nodes participating in the regeneration into two sets: \( B_1 \) and \( B_2 \) of cardinalities \( p + 1 \) and \( d - p - 1 \) respectively, with \( m \in B_2 \). Let

\[
V_1 = W_l, \ V_2 = \sum_{j \in B_1} S^{(j,l)} \text{ and } V_3 = \sum_{j \in B_2} S^{(j,l)}.
\]

Exact regeneration of node \( l \) mandates \( V_1 \subseteq V_2 + V_3 \). Also, since the cardinality of \( B_1 \) is \( p + 1 < k \), from Property 2 we get

\[
|V_1 \cap V_2| \leq \left| W_l \cap \sum_{j \in B_1} W_j \right| \leq \beta - \theta.
\]

(A.15)  

(A.16)

Now,

\[
(d - p - 1)\beta \geq |V_3| \geq |V_3 \cap (V_1 + V_2)| \geq |V_1| - |V_1 \cap V_2| \geq (d - p - 1)\beta,
\]

(A.17)  

(A.18)  

(A.19)  

(A.20)

where equation (A.19) follows from Lemma 3.3.1 and equation (A.20) from equation (A.16). This implies that the inequalities in (A.17) to (A.20) are satisfied with equality, and thus we have

\[
V_3 \subseteq V_1 + V_2
\]

(A.21)
and
\[ |V_3| = (d - p - 1)\beta. \tag{A.22} \]

From equation (A.22) we get
\[ |S^{(j,l)}| = \beta, \quad \forall j \in \mathbf{B}_2, \tag{A.23} \]
and in particular, \(|S^{(m,l)}| = \beta.

The case of \( p = k - 1 \) will be proved in a manner akin to the preceding case. Recall that by definition, the parameter \( \theta \) is zero here. Choose the sets \( \mathbf{B}_1 \) and \( \mathbf{B}_2 \) to have cardinalities \( p \) and \( d - p \) respectively. Now, Property 2 asserts that \(|V_1 \cap V_2| = 0\), and proceeding in a manner identical to steps (A.17) through (A.20) (using the fact that \( \theta = 0 \)), one can deduce equations (A.21) and (A.23).

**Proof of Property 4** The proof described here assumes the set \( \mathbf{A} \) to have exactly \( p + 2 \) nodes; the scenario of \( \mathbf{A} \) containing fewer nodes is a trivial extension.

Consider regeneration of node \( l \in \mathbf{A} \) connecting to \( d \) nodes as in the hypothesis. Define a set of nodes \( \mathbf{B}_1 = \mathbf{A} \setminus \{l\} \), and let \( \mathbf{B}_2 \) be a second set comprising of the remaining \( (d - p - 1) \) nodes participating in this process of regeneration. Note that \( m \in \mathbf{B}_2 \). Further, define three vector spaces

\[ V_1 = W_l, \quad V_2 = \sum_{j \in \mathbf{B}_1} S^{(j,l)} \quad \text{and} \quad V_3 = \sum_{j \in \mathbf{B}_2} S^{(j,l)}. \]

Now, traversing the steps outlined in the proof of Property 3, we get \( V_3 \subseteq V_1 + V_2 \), and hence

\[ S^{(m,l)} \subseteq V_1 + V_2 \subseteq \sum_{j \in \mathbf{A}} W_j. \tag{A.24} \]

Since the choice of node \( l \) from set \( \mathbf{A} \) was arbitrary, the result in equation (A.25) holds for every node in \( \mathbf{A} \). Moreover, the vector space \( S^{(m,l)} \) is contained in \( W_m \), which leads to

\[ \sum_{l \in \mathbf{A}} S^{(m,l)} \subseteq \left( W_m \cap \sum_{j \in \mathbf{A}} W_j \right). \tag{A.26} \]

Finally, since the cardinality of set \( \mathbf{A} \) is \( (p + 2) < k \), Property 2 gives

\[ \left| W_m \cap \sum_{j \in \mathbf{A}} W_j \right| = 2\beta - \theta, \tag{A.27} \]
and the result immediately follows.

\[ \blacksquare \]
**Proof of Property 5**  The proof described here assumes the set $A$ to have exactly $p+1$ nodes; the scenario of $A$ containing fewer nodes is a trivial extension.

Consider regeneration of node $l \in A$ connecting to $d$ nodes as in the hypothesis. Define a set of nodes $B_1 = A \setminus \{l\}$, and let $B_2$ be a second set comprising of the remaining $(d - p)$ nodes participating in this process of regeneration. Note that $m \in B_2$. Further, define three vector spaces

$$V_1 = W_l, \quad V_2 = \sum_{j \in B_1} S^{(j,l)} \quad \text{and} \quad V_3 = \sum_{j \in B_2} S^{(j,l)}$$

Applying Property 2, and noting that the cardinality of set $B_1$ is $p$, we get

$$|V_1 \cap V_2| = 0. \quad (A.28)$$

Furthermore,

$$|V_3 \cap (V_1 + V_2)| \geq |V_1| + |V_2 + V_3| - |V_1 \cap V_2| \geq |V_1| \geq (d - p)\beta - \theta \geq |V_3| - \theta \quad (A.30)$$

where equation (A.29) is derived from Lemma 3.3.1, and equation (A.30) is deduced from equation (A.28).

Next, define three more subspaces $U_1 = S^{(m,l)}$, $U_2 = \sum_{j \in B_2 \setminus \{m\}} S^{(j,l)}$ and $U_3 = W_l + \sum_{j \in B_1} S^{(j,l)}$. Rewriting equation (A.32) in terms of these subspaces,

$$|(U_1 + U_2) \cup U_3| \geq |(U_1 + U_2)| - \theta. \quad (A.33)$$

This, in conjunction with Lemma 3.3.2 implies

$$|U_1 \cap U_3| \geq |U_1| - \theta. \quad (A.34)$$

Property 3 mandates the dimension of $|U_1| = |S^{(m,l)}|$ to be $\beta$, and hence the previous equation evaluates to

$$\left|S^{(m,l)} \cap \left( \sum_{j \in A} W_j \right) \right| \geq \beta - \theta. \quad (A.35)$$

Since the cardinality of set $A$ is $(p + 1) < k$, Property 2 gives

$$\left| W_m \cap \left( \sum_{j \in A} W_j \right) \right| = \beta - \theta. \quad (A.36)$$
Now, as $S^{(m,l)} \subseteq W_m$, it follows from the two preceding equations:

$$
(W_m \cap \sum_{j \in A} W_j) \subseteq S^{(m,l)}.
$$

(A.37)

Since the choice of node $l$ from set $A$ was arbitrary, the result in equation (A.37) holds for every node in $A$, which leads to

$$
W_m \cap \left( \sum_{j \in A} W_j \right) \subseteq \left( \bigcap_{l \in A} S^{(m,l)} \right).
$$

(A.38)

The result follows immediately from equations (A.36) and (A.38).

Proof of Corollary 3.3.3 The proof is inductive. First, it is apparent that the conjecture holds for $a = 1$. Next, assume it holds true when the size of the set under consideration is $a - 1$. Define a set $A'$ of size $a - 1$ as $A' = A \setminus \{j\}$, for some node $j$ in $A$. Consider a node $i \in A'$. Since $p > 0$, from Property 5

$$
\beta - \theta \leq |S^{(m,j)} \cap S^{(m,i)}| \quad \text{(A.39)}
$$

$$
\leq \left| S^{(m,j)} \cap \sum_{l \in A'} S^{(m,l)} \right| \quad \text{(A.40)}
$$

$$
= \left| S^{(m,j)} \right| + \left| \sum_{l \in A'} S^{(m,l)} \right| - \left| \sum_{l \in A} S^{(m,l)} \right| \quad \text{(A.41)}
$$

$$
\leq \beta + \beta + (a - 2)\theta - \left| \sum_{l \in A} S^{(m,l)} \right| \quad \text{(A.42)}
$$

where we use the induction hypothesis in (A.42).
Appendix B

Proof of Lemma 8.4.1

Proof To simplify the notation, let

\[ T = \sum_{i=2}^{n} \min(\alpha, \mu_i + \beta_i) \]  

(B.1)

We will perform a series of transformations on \( \mu_i, \quad i = 1, \ldots, n \) and \( \beta_i, \quad i = 2, \ldots, n \) such that the value of \( T \) evaluated after step \( i \) is such that

\[ T^{(i)} \leq T^{(i-1)} \]  

(B.2)

Let

\[ \mu_i^{(0)} = \mu_i \quad \beta_i^{(0)} = \beta_i \]  

(B.3)

yielding

\[ T^{(0)} = \sum_{i=2}^{n} \min(\alpha, \mu_i + \beta_i) \]  

(B.4)

**Step1:** Choose \( \mu_1^{(1)} = \alpha \) with \( \mu_i \) and \( \beta_i, \quad (i = 2, \ldots, n) \) unaltered. Then we have for

\[ \mu_i^{(1)} = \mu_i^{(0)}, \quad \beta_i^{(1)} = \beta_i^{(0)} \]  

(B.5)

and thus

\[ T^{(1)} = T^{(0)} \]  

(B.6)

**Step2:** Decrease the values of \( \mu_i^{(1)} \) and \( \beta_i^{(1)} \) \( (i = 2, \ldots, n) \) such that

\[ \sum_{i=2}^{n} \mu_i^{(2)} = B - \alpha \]  

(B.7)

\[ \sum_{i=2}^{n} \beta_i^{(2)} = \gamma \]  

(B.8)
Decreasing the values of \( \mu_i \) and \( \beta_i \) cannot increase \( T \). Hence

\[
T^{(2)} \leq T^{(1)} \tag{B.9}
\]

**Step3:** Arrange \( \mu_i^{(2)} + \beta_i^{(2)} \) \( (i = 2, \ldots, n) \) in decreasing order of their values. If \( \mu_i^{(2)} + \beta_i^{(2)} \geq \mu_j^{(2)} + \beta_j^{(2)} \),

\[
\begin{align*}
\mu_i^{(3)} &= \mu_i^{(2)} + \min(\alpha - \mu_i^{(2)}, \mu_j^{(2)}) \tag{B.10} \\
\mu_j^{(3)} &= \mu_j^{(2)} - \min(\alpha - \mu_i^{(2)}, \mu_j^{(2)}) \tag{B.11} \\
\beta_i^{(3)} &= \beta_i^{(2)} + \min(\beta_{\text{max}} - \beta_i^{(2)}, \beta_j^{(2)}) \tag{B.12} \\
\beta_j^{(3)} &= \beta_j^{(2)} - \min(\beta_{\text{max}} - \beta_i^{(2)}, \beta_j^{(2)}) \tag{B.13}
\end{align*}
\]

These transformations do not increase the value of \( T \). Justification is as given below:

**Case I:** Suppose \( \mu_i + \beta_i \leq \alpha \). This implies \( \mu_j + \beta_j \leq \alpha \). We have

\[
\min(\alpha, \mu_i + \beta_i) + \min(\alpha, \mu_j + \beta_j) = (\mu_i + \beta_i) + (\mu_j + \beta_j)
\]

Thus increasing \( \mu_i \) and \( \beta_i \) by one and can decrease \( \mu_j \) and \( \beta_j \) by one does not alter the value of \( T \).

**Case II:** Suppose \( \mu_i + \beta_i > \alpha \). We have

\[
\min(\alpha, \mu_i + \beta_i) + \min(\alpha, \mu_j + \beta_j) = \alpha + \min(\alpha, \mu_j + \beta_j)
\]

Here, decreasing the value of \( \mu_j \) and \( \beta_j \) can only decrease the value of \( T \).

Repeatedly apply Step3 until the system converges to the following values at step \( m \)

\[
\mu_i^{(m)} = \begin{cases} 
\alpha & \text{if } i = 2, \ldots, s \\
B \mod \alpha & \text{if } i = s + 1 \\
0 & \text{otherwise}
\end{cases} \tag{B.14}
\]

\[
\beta_i^{(m)} = \begin{cases} 
\beta_{\text{max}} & \text{if } i = 2, \ldots, \lfloor \gamma/\beta_{\text{max}} \rfloor \\
\gamma \mod \beta_{\text{max}} & \text{if } i = \lfloor \gamma/\beta_{\text{max}} \rfloor + 1 \\
0 & \text{otherwise}
\end{cases} \tag{B.15}
\]

Note that this process will converge in a finite number of steps and also that

\[
\lfloor \gamma/\beta_{\text{max}} \rfloor \geq s \tag{B.16}
\]
The value of $T$ after these transformations,

$$T^{(m)} = \sum_{i=2}^{n} \min(\alpha, \mu_i^{(m)} + \beta_i^{(m)}) \quad (B.17)$$

**Case 1:** $\alpha - \beta_{\text{max}} > B \mod \alpha$

As $\gamma$ meets the cut-set bound given in equation (8.17) with equality we have

$$\gamma = \alpha + (s - 1)\beta_{\text{max}} \cdot \quad (B.18)$$

Then we have

$$T^{(m)} = (s - 1)\alpha + B \mod \alpha + \beta_{\text{max}} + \gamma - s\beta_{\text{max}}$$

$$= (s - 1)\alpha + B \mod \alpha + \alpha$$

$$= s\alpha + (B \mod \alpha)$$

$$= B \quad (B.19)$$

**Case 2:** $\alpha - \beta_{\text{max}} \leq B \mod \alpha$

As $\gamma$ meets the cut-set bound given in equation (8.17) with equality we have

$$\gamma = \alpha + B \mod \alpha + s\beta_{\text{max}} \cdot \quad (B.20)$$

$$T^{(m)} = (s - 1)\alpha + \alpha + \gamma - s\beta_{\text{max}}$$

$$= s\alpha + \alpha + B \mod \alpha$$

$$\geq B \quad (B.21)$$

Thus,

$$T = T^{(0)} \geq T^{(m)} \geq B \cdot$$
Appendix C

Proof of Theorem 4.3.2: Miser Reconstruction

Proof Let \( \omega_1, \ldots, \omega_{k-p} \) (\( \omega_1 < \ldots < \omega_{k-p} \)) be the \( k-p \) systematic nodes to which the data collector connects, and \( \Omega_1, \ldots, \Omega_p \) (\( \Omega_1 < \ldots < \Omega_p \)) be the \( p \) systematic nodes to which it does not connect. The sets \( \omega_1, \ldots, \omega_{k-p} \) and \( \Omega_1, \ldots, \Omega_p \) are disjoint. Let \( \delta_1, \ldots, \delta_p \) be the \( p \) non-systematic nodes to which the data collector connects. The matrix \( R \) is given by

\[
R = \begin{bmatrix}
G^{(\delta_1)} & G^{(\delta_2)} & \cdots & G^{(\delta_p)} \\
G^{(\delta_1)}_{\Omega_1} & G^{(\delta_2)}_{\Omega_1} & \cdots & G^{(\delta_p)}_{\Omega_1} \\
\vdots & \vdots & \ddots & \vdots \\
G^{(\delta_1)}_{\Omega_p} & G^{(\delta_2)}_{\Omega_p} & \cdots & G^{(\delta_p)}_{\Omega_p}
\end{bmatrix}
\]  (C.1)

Group the \( \Omega_1 \)th columns of \( G^{(\delta_m)} \) (\( m = 1, \ldots, p \)) as the first \( p \) columns of a new matrix \( R' \), then \( \Omega_2 \)th columns as the next \( p \) columns, and so on. Hence, column number \( \Omega_i \) of \( G^{(\delta_m)} \) becomes the column number \( p \times (i-1) + m \) in \( R' \). Next, group the \( \omega_1 \)th columns, then the \( \omega_2 \)th and so on. Column number \( \omega_i \) of \( G^{(\delta_m)} \) becomes the column number \( p^2 + p \times (i-1) + m \) in \( R' \). Hence there are \( \alpha \) groups with \( p \) columns each in \( R' \).

Let \( S \) be an \( \alpha \times p \) matrix with elements \( S_{i,j} = \psi^{(\delta_i)}_i \), \( i = 1, \ldots, \alpha, j = 1, \ldots, p \). Let \( T_{a,b} \) be an \( \alpha \times p \) matrix with its \( a \)th row as \( [\psi^{(\delta_1)}_b, \ldots, \psi^{(\delta_p)}_b] \), and rest of the elements zero. Thus, the \( a \)th row of \( T_{a,b} \) is identical to the \( b \)th row of \( S \).

The rows of \( R' \) are grouped into \( p \) groups of \( \alpha \) rows each. Thus the matrix \( R' \) can be viewed as a block matrix, with each block of size \( \alpha \times p \), and the dimension of \( R' \) being \( p \times \alpha \) blocks.

Let \([R']_{(i,j)}\) represent the \((i,j)^{th}\) block of \( R' \). For \( i = 1, \ldots, p, j = 1, \ldots, p \) we get

\[
[R']_{(i,j)} = \begin{cases} 
\epsilon S & \text{if } i = j \\
T_{\Omega_i, \Omega_i} & \text{if } i \neq j
\end{cases}
\]  (C.2)
For $i = 1, \ldots, p$, $j = p + 1, \ldots, \alpha$,

$$[R']_{(i,j)} = T_{\omega_{j-p}, \Omega_i}$$  \hspace{1cm} (C.3)

Thus,

$$R' =
\begin{bmatrix}
\epsilon S & T_{\Omega_2, \Omega_1} & \cdots & T_{\Omega_p, \Omega_1} & T_{\omega_1, \Omega_1} & \cdots & T_{\omega_{\alpha-p}, \Omega_1} \\
T_{\Omega_1, \Omega_2} & \epsilon S & \cdots & T_{\Omega_p, \Omega_2} & T_{\omega_1, \Omega_2} & \cdots & T_{\omega_{\alpha-p}, \Omega_2} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
T_{\Omega_1, \Omega_p} & T_{\Omega_2, \Omega_p} & \cdots & \epsilon S & T_{\omega_1, \Omega_p} & \cdots & T_{\omega_{\alpha-p}, \Omega_p}
\end{bmatrix}$$  \hspace{1cm} (C.4)

Let $\tilde{S}$ be the $p \times p$ matrix formed by the rows $\Omega_1, \ldots, \Omega_p$ of $S$. As $\tilde{S}$ is a submatrix of Cauchy matrix $\Psi$, it is invertible. Let $E_{a,b}$ be an $\alpha \times p$ matrix with the element at position $(a,b)$ as 1 and all other elements 0. Multiply the rightmost $\alpha - p$ groups of $p$ columns by $\tilde{S}^{-1}$. The resultant matrix is of the form

$$\begin{bmatrix}
\epsilon \tilde{S} & T_{\Omega_2, \Omega_1} & \cdots & T_{\Omega_p, \Omega_1} & E_{\omega_1, 1} & \cdots & E_{\omega_{\alpha-p}, 1} \\
T_{\Omega_1, \Omega_2} & \epsilon S & \cdots & T_{\Omega_p, \Omega_2} & E_{\omega_1, 2} & \cdots & E_{\omega_{\alpha-p}, 2} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
T_{\Omega_1, \Omega_p} & T_{\Omega_2, \Omega_p} & \cdots & \epsilon S & E_{\omega_1, p} & \cdots & E_{\omega_{\alpha-p}, p}
\end{bmatrix}$$  \hspace{1cm} (C.5)

In the column groups $p + 1, \ldots, \alpha$, every column has exactly one non-zero element. Hence the data collector obtains the corresponding source symbols, and subtracts their components from the remaining symbols.

Let $\tilde{T}_{a,b}$ be an $p \times p$ matrix with its $a^{th}$ row as $[\psi_b^{(\delta_1)}, \ldots, \psi_b^{(\delta_p)}]$, and rest of the elements zero. The data collector is left with the source symbols encoded using the following matrix

$$\begin{bmatrix}
\epsilon \tilde{S} & \tilde{T}_{2, \Omega_1} & \cdots & \tilde{T}_{p, \Omega_1} \\
\tilde{T}_{1, \Omega_2} & \epsilon \tilde{S} & \cdots & \tilde{T}_{p, \Omega_2} \\
\vdots & \vdots & \ddots & \vdots \\
\tilde{T}_{1, \Omega_p} & \tilde{T}_{2, \Omega_p} & \cdots & \epsilon \tilde{S}
\end{bmatrix}$$  \hspace{1cm} (C.6)

This is equivalent to reconstruction in a distributed storage system with $k = p$ with a data collector connecting to $p$ non-systematic nodes. Hence, general decoding algorithms for data collection from only non-systematic nodes can be applied effectively in such cases where data collection is done partially from systematic and partially from non-systematic nodes. The decoding procedure for such encoding matrices is as follows.

Now the data collector multiplies each of the remaining $p$ groups of $\alpha$ symbols by $\tilde{S}^{-1}$.
to get the following $p^2 \times p^2$ matrix

$$\begin{bmatrix}
\epsilon I_p & \tilde{E}_{2,1} & \tilde{E}_{3,1} & \cdots & \tilde{E}_{p,1} \\
\tilde{E}_{1,2} & \epsilon I_p & \tilde{E}_{3,2} & \cdots & \tilde{E}_{p,2} \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
\tilde{E}_{1,p} & \tilde{E}_{2,p} & \tilde{E}_{3,p} & \cdots & \epsilon I_p
\end{bmatrix}$$

(C.7)

where $I_p$ is a $p \times p$ identity matrix and $\tilde{E}_{a,b}$ is an $p \times p$ matrix with the element in the position $(a, b)$ as 1 and all other elements 0.

Now, the decoder only has to perform multiplications by $2 \times 2$ matrices to decode the symbols. For the sake of clarity, we first perform simple matrix manipulations. For $i = 1, \ldots, p$, the $i^{th}$ column of the $i^{th}$ column group respectively contains exactly one non-zero element (which is in the $i^{th}$ row of the $i^{th}$ row group), and hence the corresponding symbol can be recovered by the data collector. The contribution of these symbols is subtracted from the other symbols. The remaining matrix is rearranged by placing the $i^{th}$ column(row) of the $j^{th}$ group adjacent to the $j^{th}$ column(row) of the $i^{th}$ group to form:

$$\begin{bmatrix}
\epsilon & 1 & 0 & 0 & \cdots & 0 & 0 \\
1 & \epsilon & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & \epsilon & 1 & \cdots & 0 & 0 \\
0 & 0 & 1 & \epsilon & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & \cdots & \epsilon & 1 \\
0 & 0 & 0 & 0 & \cdots & 1 & \epsilon
\end{bmatrix}$$

(C.8)

This is a block diagonal matrix, and since $\epsilon^2 \neq 1$, is non-singular. The remaining source symbols can be recovered by decoding pairs of columns together. ■

In the example of $k = \alpha = 3$ considered in section 4.3.1 when the data collector connects to the first systematic node, and the first two non-systematic nodes, we have $p = 2$, $\omega_1 = 1$, $\Omega_1 = 2$, $\Omega_2 = 3$, $\delta_1 = 4$, $\delta_2 = 5$ and $\epsilon = 2$. Here,
\[ R = \begin{bmatrix} G_2^{(4)} & G_2^{(5)} \\ G_3^{(4)} & G_3^{(5)} \end{bmatrix}, \]
\[ S = \begin{bmatrix} \psi_1^{(4)} & \psi_1^{(5)} \\ \psi_2^{(4)} & \psi_2^{(5)} \\ \psi_3^{(4)} & \psi_3^{(5)} \end{bmatrix}, \]
\[ \tilde{S} = \begin{bmatrix} \psi_2^{(4)} & \psi_2^{(5)} \\ \psi_3^{(4)} & \psi_3^{(5)} \end{bmatrix}, \]
\[ T_{\Omega_1, \Omega_2} = \begin{bmatrix} 0 & 0 \\ \psi_3^{(4)} & \psi_3^{(5)} \\ 0 & 0 \end{bmatrix}, \]
\[ \tilde{T}_{\Omega_1, \Omega_2} = \begin{bmatrix} \psi_3^{(4)} & \psi_3^{(5)} \\ 0 & 0 \end{bmatrix}, \]
\[ E_{1,2} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \]
\[ \tilde{E}_{1,2} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}. \]
Appendix D

**Proof of Theorem 4.5.1: Uniqueness of the MISER code**

**Proof** Let $C$ be an MDS code performing optimal exact regeneration of the systematic nodes, with each non-systematic node passing $k$ linearly independent vectors for the regeneration of $k$ systematic nodes. Then, for $m = k+1, \ldots, 2k$, the $k \times k$ matrix

$$\begin{bmatrix} v^{(m,1)} & v^{(m,2)} & \cdots & v^{(m,k)} \end{bmatrix} \quad (D.1)$$

is non-singular.

Recall that two codes are equivalent if one code can be obtained from the other by either non-singular transformations of any of the node generator matrices or a change of basis of the entire vector space.

For non-systematic every node $m$ ($m = k+1, \ldots, 2k$), perform a non-singular transformation on the generator matrices such that, $\{v^{(m,1)}, v^{(m,2)}, \cdots, v^{(m,k)}\}$ are the $k$ columns of the transformed generator matrix. With this we move from code $C$ to an equivalent code. Let the node generator matrices of this code be denoted by $G^{(m)}$, $1 \leq m \leq 2k$.

In the transformed code, the necessity of interference alignment (Theorem 4.4.4), forces the generator matrices to have the form

$$G_i^{(m)} = \begin{bmatrix} h_{i,1} \cdots h_{i,i-1} h_{i,i}^{(m)} h_{i,i+1} \cdots h_{i,k} \end{bmatrix} \Lambda_i^{(m)}$$

$$= \tilde{h}_{i,i}^{(m)} \xi_i + \tilde{H}_i \Lambda_i^{(m)} \quad (D.2)$$

where $\Lambda_i^{(m)}$ is a $(k \times k)$ invertible diagonal matrix given by

$$\Lambda_i^{(m)} = \text{diag}\{\lambda_i^{(m)}_1, \lambda_i^{(m)}_2, \ldots, \lambda_i^{(m)}_k\}$$

$$
\text{(D.3)}
$$
$H_i$ is a $(k \times k)$ non-singular matrix and
\[
\tilde{h}_{i,i}^{(m)} = \lambda_{i,i}^{(m)} h_{i,i}^{(m)} - H_i \Lambda_i^{(m)} e_i.
\] (D.4)

Now, pre-multiply the $B \times n\alpha$ generator matrix by the non-singular matrix
\[
\begin{bmatrix}
\tilde{H}_1^{-1} & 0 & \ldots & 0 \\
0 & \tilde{H}_2^{-1} & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & \tilde{H}_k^{-1}
\end{bmatrix}
\] (D.5)

which amounts to a change of basis of the entire vector space, and thereby moving into another equivalent code. The node generator matrices of this equivalent code are given by
\[
\tilde{H}_i^{-1} G_i^{(m)} = \tilde{H}_i^{-1} h_{i,i}^{(m)} e_i^t + \Lambda_i^{(m)}
\] (D.6)

where the RHS follows from equation (D.2). Now, choosing
\[
D_i^{(m)} = \Lambda_i^{(m)} - \lambda_{i,i}^{(m)} e_i e_i^t
\] (D.7)
\[
\bar{C}_i^{(m)} = \tilde{H}_i^{-1} h_{i,i}^{(m)} + \lambda_{i,i}^{(m)} e_i
\] (D.8)

gives the generator matrices of the non-systematic nodes of this equivalent code in the form given in equation (4.84).

For regeneration of the $i$th systematic node, each non-systematic node passes the $i$th column of its generator matrix. Since the code performs optimal exact regeneration of the systematic nodes, by Theorem 4.4.2, the components of these vectors along $z_i$ should be linearly independent. These components are nothing but $\bar{C}_i^{(m)}$, $m = k + 1, \ldots, 2k$ and thus by the hypothesis, the matrix $C_i$ is non-singular.

Now, left to show is that any square submatrix of $\Delta_j$, $1 \leq j \leq k$ is non-singular. Consider the following submatrix
\[
\begin{bmatrix}
d_{i_1,j}^{(m_1)} & \ldots & d_{i_1,j}^{(m_p)} \\
\vdots & \ddots & \vdots \\
d_{i_p,j}^{(m_1)} & \ldots & d_{i_p,j}^{(m_p)}
\end{bmatrix}
\] (D.9)

for some $1 \leq p \leq k$, $\{m_1, \ldots, m_p\} \in \{k + 1, \ldots, 2k\}$, $\{i_1, \ldots, i_p\} \in \{1, \ldots, k\}\{j\}$. To show that this is non-singular, we make use of the MDS property of the code. Consider a data collector connecting to the $p$ non-systematic nodes $m_1, \ldots, m_p$ and the $k - p$ systematic nodes other than $i_1, \ldots, i_p$. Since the data collector directly obtains the source symbols stored in the $k - p$ systematic nodes it connects to, their effect can be subtracted out from the symbols stored in the $p$ non-systematic nodes $\{m_1, \ldots, m_p\}$. This leaves
behind a \( p\alpha \times p\alpha \) matrix which needs to be non-singular. From this \( p\alpha \times p\alpha \) matrix, pick the columns corresponding to the \( j^{th} \) column of the generator matrices of each of the \( p \) non-systematic nodes to form the matrix

\[
\begin{bmatrix}
\ell_{j1}^{(m_1)} & \cdots & \ell_{jp}^{(m_1)} \\
\vdots & \ddots & \vdots \\
\ell_{j1}^{(m_p)} & \cdots & \ell_{jp}^{(m_p)}
\end{bmatrix}
\]

Since all the columns of \( p\alpha \times p\alpha \) matrix are linearly independent, the columns of the above matrix are also linearly independent. For this, clearly one needs the matrix given in (D.9) to be non-singular. \[\blacksquare\]
Appendix E

Proof of Theorem 4.6.3: Existence and Construction of MSR code for 
\( d = 2k - 3 \)

Proof In the scheme provided here, for the regeneration of a systematic node, a non-
systematic node passes the same vector irrespective of the other \( \alpha - 1 \) non-systematic nodes 
which participate in the regeneration. Consider the exact regeneration of the systematic 
node \( \alpha + 1 \). By Theorem 4.4.4, in the vectors passed by the \( \alpha \) non-systematic nodes i.e.,
\[
\{v^{(m,\alpha+1)}, \ldots, v^{(m,\alpha+1)}\}
\]
the component along the systematic nodes \( l, \forall \ l \in \{1, \ldots, \alpha\} \), need to be aligned in one 
direction. To satisfy this we choose, for \( m = k + 2, \ldots, n \),
\[
G_l^{(m)} x^{(m,\alpha+1)} = \kappa_l^{(m,\alpha+1)} G_l^{(k+1)} x^{(k+1,\alpha+1)}
\]
(E.2)
Similarly, alignment for the exact regeneration of the systematic node \( \alpha + 2 \) leads to 
another set of \( n - k - 1 \) equations: \( m = k + 2, \ldots, n \),
\[
G_l^{(m)} x^{(m,\alpha+2)} = \kappa_l^{(m,\alpha+2)} G_l^{(k+1)} x^{(k+1,\alpha+2)}
\]
(E.3)
for some constants \( \kappa 's \in \mathbb{F}_q \).

For all \( m \in \{k + 2, \ldots, n\} \), multiply equation (E.2) by \( (x_l^{(m,\alpha+1)})^{-1} \) and (E.3) by 
\( (x_l^{(m,\alpha+2)})^{-1} \) and subtract the two. \( h_{l,l}^{(m)} \) gets eliminated and a homogeneous equation in 
terms of \( h_{l,1}, \ldots, h_{l,l-1}, h_{l,l+1}^{(k+1)}, h_{l,l+1}, \ldots, h_{l,\alpha} \) remains. One way to satisfy this equation
is to equate all the scalar coefficients to zero.

Making the coefficients of \( h_{l,1}, \ldots, h_{l,l-1}, h_{l,l+1}, \ldots, h_{l,\alpha} \) zero gives, for \( l = 1, \ldots, \alpha, \ m = \)
\[ \lambda^{(m)}_{l,i} = \lambda^{(k+1)}_{l,i} \cdot \left[ \kappa^{(m,\alpha+1)}_{l} (x^{(m,\alpha+1)}_{l})^{-1} x^{(k+1,\alpha+1)}_{i} - \kappa^{(m,\alpha+2)}_{l} (x^{(m,\alpha+2)}_{l})^{-1} x^{(k+1,\alpha+2)}_{i} \right]. \]

(E.4)

Making the coefficient of \( h^{(k+1)}_{l,l} \) zero gives, for \( m = k + 2, \ldots, n \) and \( l = 1, \ldots, \alpha \)

\[ \kappa^{(m,\alpha+2)}_{l} = \kappa^{(m,\alpha+1)}_{l} x^{(k+1,\alpha+1)}_{i} (x^{(m,\alpha+1)}_{l})^{-1} (x^{(k+1,\alpha+2)}_{i})^{-1} (x^{(m,\alpha+2)}_{l})^{-1} \]

(E.5)

Equations (E.4) and (E.5) ensure that second set of equations (i.e. E.3) are satisfied whenever the first set (i.e. E.2) is satisfied. Note that any polynomial containing a either \( \lambda^{(m)}_{l,i} \ (i \neq l) \) or \( \kappa^{(m,\alpha+2)}_{l} \) term will be a rational polynomial. For such polynomials, we need to obtain an assignment for variables which simultaneously ensure that none of the inverted terms are zero, and the polynomial is also not zero.

Now, only the set of equations in (E.2) have to be satisfied, for which, we make the following assignments, for \( m = k + 2, \ldots, n \) and \( l = 1, \ldots, \alpha \)

\[ H^{(m)}_{l,i} = h^{(k+1)}_{l,l} \left( (x^{(m,\alpha+1)}_{l})^{-1} \kappa^{(m,\alpha+1)}_{l} x^{(k+1,\alpha+1)}_{i} \right) + \sum_{i=1,i\neq l}^{\alpha} H_{l,i} \left[ (x^{(m,\alpha+1)}_{l})^{-1} \left( \kappa^{(m,\alpha+1)}_{l} \lambda^{(k+1)}_{l,i} x^{(k+1,\alpha+1)}_{i} - \lambda^{(m)}_{l,i} x^{(m,\alpha+1)}_{i} \right) \right] \]

(E.6)

Alignment of components along systematic nodes \( \{1, \ldots, \alpha\} \) is taken care of. In the vectors passed for the regeneration of the systematic node \( \alpha + 2 \), the component along systematic node \( \alpha + 1 \) needs to be aligned and vice versa. Hence the alignment of systematic nodes \( \alpha + 1 \) and \( \alpha + 2 \) result only in one set of \( n - k - 1 \) equations each. Consider the exact regeneration of systematic node \( \alpha + 2 \). By Theorem 4.4.4, the component along the systematic node \( \alpha + 1 \) in the vector passed by non-systematic nodes need to be aligned in one direction. To satisfy this we choose, for \( m = k + 2, \ldots, n \)

\[ H^{(m)}_{\alpha+1} \Lambda^{(m)}_{\alpha+1} = \kappa^{(m,\alpha+2)}_{\alpha+1} H^{(k+1)}_{\alpha+1} \Lambda^{(k+1)}_{\alpha+1} \]

(E.7)
From equation (4.95) we have

\[ H^{(m)}_{\alpha+1} = H^{(k+1)}_{\alpha+1} = H_{\alpha+1} \quad \text{(say)} \]  

(E.8)

Thus, equating the scalar coefficients to zero in equation (E.7), we get for \( i = 1, \ldots, \alpha, \)

\[ \lambda^{(m)}_{\alpha+1,i} = \lambda^{(k+1)}_{\alpha+1,i} (x^{(m,\alpha+2)}_i)^{-1} x^{(k+1,\alpha+2)}_i \]  

(E.9)

Similarly, for the exact regeneration of node \( \alpha + 1, \) we need to align components along node \( \alpha + 2 \) which leads to

\[ \lambda^{(m)}_{\alpha+2,i} = \lambda^{(k+1)}_{\alpha+2,i} (x^{(m,\alpha+2)}_i)^{-1} x^{(k+1,\alpha+2)}_i \]  

(E.10)

**Regeneration**

Exact regeneration of each one of the systematic nodes \( l \in \{1, \ldots, \alpha\} \) results in a condition

\[ \det \begin{bmatrix} h^{(m)}_{l,l} & \cdots & h^{(m)}_{l,m} \\ \vdots & \ddots & \vdots \\ h^{(m)}_{l,m} & \cdots & h^{(m)}_{l,l} \end{bmatrix} \neq 0 \]  

(E.11)

where \( m_1, \ldots, m_\alpha \) are the \( \alpha \) non-systematic nodes participating in the regeneration. After substituting for \( h^{(m)}_{l,i} \), \( i = 1, \ldots, \alpha \) from equation (E.6), this condition evaluates to a rational polynomial, which can be shown to be not identically equal to zero by the following assignments: For \( i = 1, \ldots, \alpha, i \neq l, m \in \{m_1, \ldots, m_\alpha\}, m \neq k + 1 \)

\[
\begin{align*}
\kappa^{(m,\alpha+1)}_l &= 1, & \lambda^{(k+1)}_{l,i} &= -1, & h^{(k+1)}_{l,i} &= \xi_i, & h^{(m)}_{l,i} &= \xi_i \\
x^{(m,\alpha+2)}_l &= x^{(m,\alpha+2)}_i = 1, & x^{(k+1,\alpha+2)}_l &= x^{(k+1,\alpha+2)}_i = 1 \\
x^{(k+1,\alpha+1)}_l &= 0, & x^{(k+1,\alpha+1)}_i &= x^{(k+1,\alpha+1)}_i = 1, & x^{(m,\alpha+1)}_l &= x^{(m,\alpha+1)}_i = 1 \\
x^{(m,\alpha+2)}_l &= (m - k)^{-j} + 1 
\end{align*}
\]  

(E.12)

where \( j = i \) if \( i < l, \) \( j = i - 1 \) if \( i > l. \) These assignments make the matrix under consideration in equation (E.11) a Vandermonde matrix which is full rank, and ensures that equations (E.4), (E.9) and (E.10) remain valid, provided the field size is large enough.

Exact regeneration of systematic nodes \( \alpha + 1 \) and \( \alpha + 2 \) also result in conditions of rational polynomials being not equal to zero. Consider exact regeneration of systematic node \( (\alpha + 1). \) We choose \( H_{\alpha+1} \) to be a full rank matrix. Now, Theorem 4.4.2 implies that the coefficients resulting from the linear combinations need to be linearly independent, i.e \( \Lambda^{(m)}_{\alpha+1} \xi^{(m,\alpha+1)} \) should be linearly independent for any \( \alpha \) out of the \( n - k \) non-systematic nodes. Express the determinant of each such matrix as a polynomial. To show that this
polynomial is not identically zero, we choose, for \( i = 1, \ldots, \alpha, \ m = k + 2, \ldots, n \)

\[
\begin{align*}
\Lambda_{\alpha+1}^{(k+1)} &= I, \quad \kappa_{\alpha+2}^{(m, \alpha+1)} = 1, \quad \kappa_{\alpha+1}^{(m, \alpha+2)} = 1, \\
x_i^{(m, \alpha+2)} &= 1, \quad x_i^{(m, \alpha+1)} = (m - k)^i, \\
x_i^{(k+1, \alpha+2)} &= 1, \quad x_i^{(k+1, \alpha+1)} = 1.
\end{align*}
\] (E.13)

Note that these assignments also ensure that equations (E.4), (E.9) and (E.10) remain valid. A similar argument can be used to obtain a condition for regeneration of node \( \alpha + 2 \).

Since the hypothesis of Lemma 4.6.2 is satisfied for all systematic nodes, the systematic nodes can be regenerated with the repair bandwidth meeting the cut-set bound.

**Reconstruction**

For reconstruction, the node matrices corresponding to any \( k \) nodes, when juxtaposed one next to the other, should form a \( B \times B \) full rank matrix. If all the \( k \) nodes are systematic, then reconstruction is trivially satisfied. Suppose \( p \) out of the \( k \) nodes are non-systematic nodes and \( k - p \) systematic, \( 1 \leq p \leq k \). Let \( m_1, \ldots, m_p, \ (m_1 < \ldots < m_p) \) be the non-systematic nodes to which it connects. Let \( l_1, \ldots, l_p, \ (l_1 < \ldots < l_p) \) be the \( p \) systematic nodes to which it does not connect. Due to the structure of node matrices of the systematic nodes, we will be left with the condition of the \( l\alpha \times l\alpha \) matrix formed by the column sets \( l_1, \ldots, l_p \) of the node matrices of the \( p \) non-systematic nodes being non-singular. Thus the polynomial corresponding to this choice of \( k \) nodes is

\[
\det \begin{pmatrix} G_{l_1}^{(m_1)} & \cdots & G_{l_1}^{(m_p)} \\ \vdots & \ddots & \vdots \\ G_{l_p}^{(m_1)} & \cdots & G_{l_p}^{(m_p)} \end{pmatrix} \] (E.14)

To show that this polynomial is not identically zero, make the following assignments to the variables: For \( i, j = 1, \ldots, p, \ m_1 \neq k + 1 \), set

\[
\kappa_{l_j}^{(m_i, \alpha+1)} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}
\] (E.15)

For \( i = 1, \ldots, \alpha, \ m = k + 2, \ldots, n, \ l = 1, \ldots, k \), set

\[
\begin{align*}
H_l^{(k+1)} &= I, \quad \Lambda_l^{(k+1)} = I, \\
x_i^{(m, \alpha+1)} &= x_i^{(k+1, \alpha+1)} = \eta_i, \\
x_i^{(m, \alpha+2)} &= x_i^{(k+1, \alpha+2)} = 1.
\end{align*}
\] (E.16)

where \( \eta_i \neq \eta_j \) for \( i \neq j \). With these values, from (E.4), (E.6), (E.9) and (E.10), we get that the matrix in equation (E.14) is full rank for a large enough field size, and also the
equations (E.4), (E.9) and (E.10) remain valid. Thus, determinant corresponding to every choice of \( k \) nodes is a non-zero polynomial. This implies that, the product of all such polynomials is also non-zero.

Thus, provided that the field size is large enough, one can find solutions for these variables such that both reconstruction and exact regeneration of systematic nodes are satisfied in the parameter regime \( d = 2k - 3 \).
Bibliography


