Minimizing Repair Bandwidth in Distributed Storage Systems

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

by

Nihar B. Shah

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 all the server crashes and hard disk failures that inspired this area of research
Acknowledgments

I would like to thank my advisor Prof. P. Vijay Kumar for his guidance and support throughout this work. His viewpoints always made things look much simpler. The informal environment and the spacious lab are extremely conducive to research. I thank Prof. Kannan Ramchandran for introducing this problem to us and also for his timely guidance and feedback. Also my project partner Rashmi K.V. with whom I had zillions of discussions and arguments (which typically concluded with several kung-fu kicks and punches raining upon me), which finally led to the results. I thank my project reviewers Prof. Utpal Mukherji and Prof. Rajesh Sundaresan for reviewing this work. Many thanks to my labmates Poornima, Vinodh, Lalitha, Prakash, Gagan, Avik, Abu, Ramanathan, Raviteja, Sharath; and of course, all my friends and family!
Abstract

Regenerating codes are a class of distributed storage codes that optimally trade the bandwidth needed for repair of a failed node with the amount of data stored per node of the network. An \([n, k, d]\) regenerating code permits the data to be recovered by connecting to any \(k\) of the \(n\) nodes in the network, while requiring that repair of a failed node be made possible by connecting (using links of lesser capacity) to any \(d\) nodes. For any set of parameters, there exists a fundamental cut-set lower bound on the repair bandwidth. The only explicit codes available in the literature prior to this work were for cases when the number of nodes is not more than four.

The primary focus of this report is on construction of explicit and optimal families of regenerating codes, for various parameter sets. Many of these codes can also be employed for efficient data dissemination across the storage network. A second set of results answers an open question on the achievability of the cut-set bound. Furthermore, a more flexible framework for regenerating codes is developed, that can potentially reduce the downloading times by a considerable amount.
Contents

Acknowledgments i

Abstract ii

1 Introduction 1
  1.1 Setting .................................................. 1
    1.1.1 Regenerating Codes .................................. 1
    1.1.2 Cut-Set Bound and the Storage-Repair-Bandwidth Tradeoff ... 3
    1.1.3 Striping of Data ...................................... 5
    1.1.4 Additional Terminology ................................. 6
  1.2 Literature Survey ........................................ 7
  1.3 Organization of the Report ............................... 9

2 Explicit Codes using a Product-Matrix Framework 12
  2.1 The Common Product-Matrix Framework ....................... 12
    2.1.1 Product-Matrix Description ........................... 13
    2.1.2 Dispersion and Block-Symmetry ........................ 13
    2.1.3 Data Encoding for the Storage Nodes ................... 14
    2.1.4 Reconstruction of Data by the Data Collector .......... 14
    2.1.5 Regeneration of a Failed Node ........................ 15
  2.2 The Product-Matrix MBR Code Construction .................. 16
    2.2.1 An Example for the Product-Matrix MBR Code .......... 18
    2.2.2 Systematic Version of the Code ....................... 19
    2.2.3 Optimal Security from Eavesdroppers ................... 20
  2.3 The Product-Matrix MSR Code Construction ................... 22
    2.3.1 An Example for the Product-Matrix MSR Code .......... 26
    2.3.2 Systematic Version of the Code ....................... 27
    2.3.3 Explicit MSR Product-Matrix Codes for $d \geq 2k - 2$ .... 28
    2.3.4 Optimal Security from Eavesdroppers ................... 30
  2.4 The Ideal Regenerating Code: Simultaneous Minimization of Storage and Bandwidth ....................... 31
2.4.1 System Setting ............................................. 31
2.4.2 The Code Construction .................................. 32
2.4.3 An Example ................................................ 35
2.4.4 Optimal Security from Eavesdroppers ................. 36
2.4.5 Flexible Reconstruction and Regeneration ............. 39
2.5 Analysis and Advantages of the Codes ..................... 41
  2.5.1 Reduced Overhead ...................................... 41
  2.5.2 Applicability to Arbitrary $n$ .......................... 42
  2.5.3 Data Dissemination .................................... 42
  2.5.4 Complexity ............................................. 44
2.6 Description of a Previously Constructed MSR Code in the Product-Matrix Framework ........................................ 44

3 Interference Alignment based Codes for the MSR Point 48
  3.1 Global Kernels, Nodal Subspaces and Equivalent Codes . . 50
  3.2 Additional Notation and a Bound on the Repair Bandwidth .... 51
  3.3 Need for Interference Alignment in Regenerating Codes ........ 54
  3.4 Construction of the MISER Code .......................... 55
    3.4.1 An Example ........................................... 56
    3.4.2 The MISER Code for $n = 2k$, $d = n - 1$ ............. 61
    3.4.3 The MISER Code for $n \geq 2k$, $d = n - 1$ ............ 64
    3.4.4 Extension to the case $2k - 1 \leq d \leq n - 1$ .......... 64
    3.4.5 Analysis of the MISER Code ........................... 65
  3.5 Non-Existence for $d < 2k - 3$ .......................... 66
    3.5.1 Necessary Properties ................................. 66
    3.5.2 Structure of the Code ................................ 68
    3.5.3 The Non-Existence Proof ............................ 71
  3.6 Existence and Construction for $d \geq 2k - 3$ ............. 73
    3.6.1 Approach ............................................. 73
    3.6.2 Existence and Construction ........................... 74
  3.7 A Coding Scheme for any $(n, k, d)$: Exact Regeneration of Systematic Nodes 76
    3.7.1 Scheme Description .................................. 76
    3.7.2 Analysis ............................................. 78
  3.8 Towards Uniqueness of Code Construction .................. 79

4 Non-achievability of Interior Points on the Storage-Repair Bandwidth Tradeoff .................................................. 82
  4.1 A Representation for the Points on the Tradeoff .......... 83
  4.2 Subspace Properties of Linear Exact Regenerating Codes .... 84
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3</td>
<td>Non-achievability of the Interior Points On the Tradeoff for Exact Regeneration</td>
<td>86</td>
</tr>
<tr>
<td>4.3.1</td>
<td>An Achievable Curve via Storage Space Sharing</td>
<td>90</td>
</tr>
<tr>
<td>5</td>
<td>An Explicit MBR Code for $d = n - 1$, and its Graphical Description</td>
<td>92</td>
</tr>
<tr>
<td>5.1</td>
<td>An Example Code</td>
<td>93</td>
</tr>
<tr>
<td>5.2</td>
<td>Code Construction for the General Set of Parameters with $d = n - 1$</td>
<td>94</td>
</tr>
<tr>
<td>5.3</td>
<td>Size of the Finite Field and XOR based constructions</td>
<td>96</td>
</tr>
<tr>
<td>6</td>
<td>Approximately Exact Regenerating Codes</td>
<td>97</td>
</tr>
<tr>
<td>6.1</td>
<td>Construction for the Case $\alpha = 1$</td>
<td>97</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Reconstruction</td>
<td>98</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Regeneration</td>
<td>98</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Repair Bandwidth</td>
<td>98</td>
</tr>
<tr>
<td>6.2</td>
<td>Construction for the Case $\alpha = 2$</td>
<td>99</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Reconstruction</td>
<td>99</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Regeneration</td>
<td>100</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Optimum Partition Size and Repair Bandwidth</td>
<td>101</td>
</tr>
<tr>
<td>6.3</td>
<td>Construction for an Arbitrary $\alpha$</td>
<td>102</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Reconstruction</td>
<td>103</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Regeneration</td>
<td>104</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Optimum Partition Size and Repair Bandwidth</td>
<td>104</td>
</tr>
<tr>
<td>6.4</td>
<td>Analysis of the Code</td>
<td>106</td>
</tr>
<tr>
<td>7</td>
<td>A Flexible Class of Regenerating Codes</td>
<td>108</td>
</tr>
<tr>
<td>7.1</td>
<td>Motivation</td>
<td>108</td>
</tr>
<tr>
<td>7.2</td>
<td>Description of the Flexible Setting</td>
<td>109</td>
</tr>
<tr>
<td>7.3</td>
<td>Lower bound on the Repair Bandwidth</td>
<td>110</td>
</tr>
<tr>
<td>7.3.1</td>
<td>Information Flow Graph</td>
<td>111</td>
</tr>
<tr>
<td>7.3.2</td>
<td>Cut-set Lower Bound</td>
<td>111</td>
</tr>
<tr>
<td>7.3.3</td>
<td>Complete Flexibility ($\beta_{\text{max}} = \alpha$)</td>
<td>113</td>
</tr>
<tr>
<td>7.4</td>
<td>Achievability</td>
<td>113</td>
</tr>
<tr>
<td>7.5</td>
<td>An Explicit Code</td>
<td>118</td>
</tr>
<tr>
<td>7.5.1</td>
<td>An Example</td>
<td>118</td>
</tr>
<tr>
<td>7.5.2</td>
<td>General Form of the Code</td>
<td>119</td>
</tr>
<tr>
<td>7.6</td>
<td>Symbol-wise MDS Codes and Error-prone Links</td>
<td>120</td>
</tr>
<tr>
<td>8</td>
<td>Summary of the Work</td>
<td>122</td>
</tr>
</tbody>
</table>

Bibliography 123
Chapter 1

Introduction

In a distributed storage system, information pertaining to a data file is dispersed across nodes in a network in such a manner that an end-user can retrieve the data stored by tapping into a subset of the nodes. A popular option that reduces network congestion and that leads to increased resiliency in the face of node failures, is to employ erasure coding, for example by calling upon maximum-distance-separable (MDS) codes such as Reed-Solomon (RS) codes. Let $B$ be the total file size measured in terms of symbols over a finite field $\mathbb{F}_q$ of size $q$. With RS codes, data is stored across $n$ nodes in the network in such a way that the entire data can be recovered by a data collector by connecting to any $k$ nodes, a process of data recovery that we will refer to as reconstruction. Several distributed storage systems such as RAID-6 [1], OceanStore [2] and Total Recall [3] employ such an erasure-coding option.

1.1 Setting

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. 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, and 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. [4].

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.\footnote{This argument does not hold for linearity over a subfield.} 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 [4], 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 \textit{regeneration} and the total amount $d\beta$ of data downloaded for repair purposes as the \textit{repair bandwidth}. Typically, with a regenerating code, the average repair bandwidth is small compared to the size of the file $B$. The processes of data reconstruction and failed node regeneration are illustrated in Figure 1.1.

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

\begin{equation}
    k \leq d \leq n - 1, \quad (1.1)
\end{equation}

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
Figure 1.2: The distributed storage system represented as an infinite chain of failures and regenerations of nodes. Every storage node is represented by two vertices in the graph: an *in* vertex and an *out* vertex, with an \( \alpha \) capacity link between them. A cut is also depicted, that leads to the cut-set bound on repair bandwidth.

contradicting the minimality of \( k \). 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 \((\alpha, \beta, B)\). Throughout we will assume each of these parameters to be positive integers, as mandated by linearity of the codes over \( \mathbb{F}_q \).

### 1.1.2 Cut-Set Bound and the Storage-Repair-Bandwidth Trade-off

A major result in the field of regenerating codes is the proof in [5] that uses the cut-set bound of network coding, on the graph constructed in Figure 1.2, to establish that the parameters of a regenerating code must necessarily satisfy:

\[
B \leq \sum_{i=0}^{k-1} \min\{\alpha, (d - i)\beta\}. \tag{1.2}
\]

The existence of codes achieving the cut-set bound has been shown in [5, 8].
It is desirable to minimize both $\alpha$ as well as $\beta$ since, minimizing $\alpha$ results in a minimum storage solution, while minimizing $\beta$ (for fixed $d$) results in a storage solution that minimizes repair bandwidth. As can be deduced from (1.2), it is not possible to minimize both $\alpha$ and $\beta$ simultaneously and thus there is a tradeoff between choices of the parameters $\alpha$ and $\beta$. This tradeoff is plotted for the parameters $B = 27000$, $k = 10$, $d = 18$, and some $n > 18$, in Figure 1.3.

When $\alpha = 1$, the lower bound evaluates to $B = k$, for which any $[n, k]$-MDS code will trivially achieve the lower bound\(^2\). Hence, we will consider $\alpha > 1$ throughout.

For the reconstruction property to be satisfied, the value of $\alpha$ must be greater than $\frac{B}{k}$. When $\alpha$ is set equal to this minimum possible value, equation 1.2 gives

$$\alpha \geq (d - k + 1)\beta. \quad (1.3)$$

Moreover, since a node is forced to download at least as much as it stores, we get

$$\alpha \leq d\beta. \quad (1.4)$$

\(^2\)It will achieve the lower bound even for the more practical scenario of exact regeneration, described in Subsection 1.1.4.
Thus, combining equations 1.3 and 1.4, the tradeoff is defined for the range

\[(d - k + 1)\beta \leq \alpha \leq d\beta.\] (1.5)

The two extreme points in this tradeoff are termed the minimum storage regeneration (MSR) and minimum bandwidth regeneration (MBR) points, and correspond to the absolute minimum storage space and absolute minimum repair bandwidth respectively. 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

\[
\alpha = \frac{B}{k}, \\
\beta = \frac{B}{k(d - k + 1)}. \tag{1.6}
\]

Reversing the order, leads to the MBR point which thus corresponds to

\[
\beta = \frac{2B}{k(2d - k + 1)}, \\
\alpha = \frac{2dB}{k(2d - k + 1)}. \tag{1.7}
\]

Note that regenerating codes with \((\alpha = \alpha_{\text{MSR}})\) and \((\beta = \beta_{\text{MSR}})\) are necessarily MDS codes over the vector alphabet \(\mathbb{F}_q^\alpha\). This follows since the ability to reconstruct the data from any \(k\) nodes necessarily implies a minimum distance \(d_{\text{min}} = n - k + 1\). Since the code size equals \((q^\alpha)^k\), this meets the Singleton bound causing the code to be an MDS code.

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 \(\alpha\) or \(\beta\) 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 (1.6) and similarly, an MBR code as one with parameters \((\alpha, \beta, B)\) satisfying (1.7). Clearly, both MSR and MBR codes are optimal regenerating codes.

### 1.1.3 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.
1.1 Setting

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, a code with smaller \(\beta\) will involve manipulating a smaller number of message symbols and hence will in general, be of lesser complexity. For these reasons, we design codes for the atomic case \(\beta = 1\). Thus, we will assume \(\beta = 1\) throughout. We document below the values of \(\alpha\) and \(B\) when \(\beta = 1\) of MSR and MBR codes respectively:

\[
\alpha = d - k + 1, \quad \quad \quad (1.8) \\
B = k(d - k + 1). \quad \quad \quad (1.9)
\]

for MSR codes and

\[
\alpha = d, \quad \quad \quad (1.10) \\
B = kd - \binom{k}{2} \quad \quad \quad (1.11)
\]

in the case of MBR codes.

1.1.4 Additional Terminology

**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.
1.2 Literature Survey

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. Clearly, in the case of systematic regenerating codes, exact regeneration of (the systematic portion of the data stored in) the systematic nodes is mandated.

Linear regenerating codes

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

(i) the code symbols stored in each node are linear combinations over $\mathbb{F}_q$ of the $B$ message symbols $\{u_i\}$,

(ii) the symbols passed by a node $\nu$ to aid in the regeneration of a failed node $\nu_f$ are linear over $\mathbb{F}_q$ in the $\alpha$ symbols stored in node $\nu$.

It follows as an easy consequence, that linear operations suffice for a data collector to recover the data from the $k\alpha$ code symbols stored in the $k$ nodes that it has connected to. Similarly, the replacement for a failed node $\nu_f$, performs linear operations on the $d$ symbols passed on to it by the $d$ helper nodes $\{\nu_i\}_{j=1}^d$ aiding in the regeneration.

Miscellaneous Notation

The following notation will be used throughout this writeup. Superscripts will be used to refer to the node numbers, and subscripts to index the elements of any matrix. A vector, by default, will mean a row vector. Define the vector $e_i$ as an $\alpha$-length unit vector with 1 in $i^{th}$ position and 0 elsewhere. Two vectors are aligned if they are linearly dependent. We will denote an $m \times m$ zero matrix as $0_m$, and an $m \times m$ identity matrix as $I_m$.

1.2 Literature Survey

The concept of regenerating codes was introduced in [4], 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 [4,5]. Here, the evolution of the storage network through a sequence of failures and regenerations is represented as a network, with all possible data collectors
1.2 Literature Survey

represented as sinks. 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 (1.2). It can be seen that there is a tradeoff between the choice of the parameters $\alpha$ and $\beta$ for a fixed $B$ and this is termed as the storage-repair bandwidth tradeoff. It has been shown ([5], [8]) that this tradeoff is achievable. However, the coding schemes suggested are not explicit and require large field size. Also contained in the journal version of [6], is a handcrafted functional regenerating code for the MSR point with $(n = 4, k = 2, d = 3)$.

For construction of codes, authors propose to use the algorithm by Jaggi et al. [27]; the drawbacks of such an approach include high complexity of code construction as well as the requirement of a large field size.

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 [7] in the context of random linear regenerating codes that achieve functional repair.

The problem of exact regeneration was first considered independently in [9] and [10]. In [9], 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 [11], 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 [10] 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 $n^2$. It was subsequently shown by Pawar et al. [16] that this code achieves the secrecy-capacity for distributed storage in a bandwidth limited regime.

Also contained in [10] (see also [17]), is the construction of an explicit MSR code for $d = k + 1$, that performs approximately-exact regeneration of all failed nodes. Note that the only explicit codes to previously have been constructed for the MSR point are for small values of parameters, $[n = 4, k = 2, d = 3]$ and $[n = 5, k = 3, d = 4]$.

MSR codes performing a hybrid of exact and functional regeneration are provided in [21], for the parameters $d = k + 1$ and $n > 2k$. 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 [12], 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 journal-submission version [17] of [12]. Subsequently, it was shown in [15] that for this set of parameters, the code introduced in [12] 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 [15]. Also contained in [15], 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 [17].

A flexible setup for regenerating codes is described in [20], 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 [18], the authors establish that with at most one exception, the interior points on the tradeoff (i.e., other than MSR and the MBR points) are not achievable under exact regeneration.

The MSR point was shown to be achievable in the limiting case of \(B\) approaching infinity (i.e., \(\beta\) approaching infinity) in [22, 23].

Explicit code constructions for the MSR point for the parameter set \(d \geq 2k-2\), with arbitrary \(n\), and the MBR point with any arbitrary parameters was provided in [24], based on a Product-Matrix framework conceived in the same paper. Also contained in this paper is a simpler description of the MISER code in the Product-Matrix framework.

### 1.3 Organization of the Report

The next chapter, Chapter 2, introduces a Product-Matrix Framework for code construction, which is used to construct explicit codes for the MBR point for all values of \([n, k, d]\), the MSR point for all parameters \([n, k, d \geq 2k-2]\), and a hybrid code simultaneously minimizing storage and bandwidth for all values of the parameters. Each of these codes is capable of performing optimal exact regeneration of all nodes in the system. These codes are also highly useful for data dissemination across the network, and for security against eavesdroppers.

The focus of Chapter 3 is on code constructions for the MSR point, for which the concept of Interference Alignment is employed. It is shown that interference alignment is necessary for construction of such codes, using which a systematic MSR code is provided for \(d \leq 2k-1\). Also provided is a proof of non-achievability of the cut-set bound for exact regeneration is established for the MSR point for \(d < 2k-3\) for the atomic case of \(\beta = 1\). Furthermore, existence of codes achieving optimal exact regeneration of the systematic nodes for \(d = (n-1) \geq 2k-3\) is proved. A coding scheme for any \((k, \alpha)\) parameter set is provided, that is optimal for \(k \leq \alpha\).
Chapter 4 answers the question of achievability of the interior points on the storage-repair bandwidth tradeoff under exact regeneration, in the negative (with the possible exception of the region of width at-most $\beta$ in the immediate vicinity of the MSR point). This result is based on a set of properties, necessary for any exact-regenerating code, that are established along the way. Achievable values of repair bandwidths for the interior points are obtained via storage space sharing between the MSR and MBR codes presented in Chapter 2.

Chapter 5 describes the construction of an explicit exact-regenerating code for the MBR point for the parameters $(n, k, d = n - 1)$, that can perform exact regeneration of all nodes. This code has a particularly simple graphical description which makes it simple to implement. Most interestingly, when specialized to the case $[n, k = n - 2, d = n - 1]$, simple XOR operations at the nodes suffice to implement reconstruction and regeneration.

Chapter 6 provides an explicit code which reduces the repair bandwidth for all the nodes to approximately $B/2$. The codes provided are approximately exact, i.e., only half of the symbols stored in the new node are identical to those in the failed node. An important special case of this code is for the MSR point for the parameters $d = k + 1$.

In Chapter 7, a flexible framework is introduced, where the entire source data can be recovered by connecting to any number of nodes as long as the total amount of data downloaded is at least $B$, and regeneration of a failed node is possible if the new node connects to the network using links whose sum capacity equals or exceeds a predetermined parameter $\gamma$. A cut-set lower bound on the repair bandwidth is obtained, and is shown to be achievable for all values of the parameters. An explicit code construction is provided, optimal in certain parameter regimes. Another construction, though not explicit, of symbol-wise MDS codes meeting this cut-set bound on repair bandwidth is provided; such codes are highly useful when links are prone to errors.

Finally, the work done in this report is recapitulated in Chapter 8.

In Appendix A, certain insights into code construction for general non-multicast networks are provided, which were gained in course of this work. Appendices B, C and D contain complete proofs of certain theorems stated in the report.
Codes constructed in this report are tabulated as follows.

<table>
<thead>
<tr>
<th>Point on the curve</th>
<th>System parameters</th>
<th>Type of regeneration</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSR</td>
<td>( d \geq 2k - 2 )</td>
<td>Exact regeneration of all nodes</td>
<td>Optimal explicit code</td>
</tr>
<tr>
<td>MBR</td>
<td>All</td>
<td>Exact regeneration of all nodes</td>
<td>Optimal explicit code</td>
</tr>
<tr>
<td>hybrid</td>
<td>All</td>
<td>Exact regeneration of all nodes</td>
<td>Optimal explicit code</td>
</tr>
<tr>
<td>MBR</td>
<td>( d = n - 1 )</td>
<td>Exact regeneration of all nodes</td>
<td>Optimal explicit code</td>
</tr>
<tr>
<td>MSR</td>
<td>( d \geq 2k - 1 )</td>
<td>Exact regeneration of systematic nodes</td>
<td>Optimal explicit code</td>
</tr>
<tr>
<td>MSR</td>
<td>( 2k - 2 \geq d \geq 2k - 3 )</td>
<td>Exact regeneration of systematic nodes</td>
<td>Optimal scheme</td>
</tr>
<tr>
<td>MSR</td>
<td>( d \leq 2k - 4, \beta = 1 )</td>
<td>Exact regeneration of systematic nodes</td>
<td>Cut-set bound strictly loose</td>
</tr>
<tr>
<td>MSR</td>
<td>All</td>
<td>Exact regeneration of systematic nodes</td>
<td>Scheme optimal for a subset of parameters</td>
</tr>
<tr>
<td>MSR</td>
<td>( d = k + 1 )</td>
<td>Hybrid regeneration of all nodes</td>
<td>Optimal explicit code</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Hybrid regeneration of all nodes</td>
<td>Explicit code optimal for a subset of parameters</td>
</tr>
<tr>
<td>Flexible</td>
<td>All</td>
<td>Functional regeneration of all nodes</td>
<td>Optimal Scheme</td>
</tr>
</tbody>
</table>
Chapter 2

Explicit Codes using a Product-Matrix Framework

This chapter introduces a Product-Matrix Framework for code construction (Section 2.1), which is subsequently used to construct explicit codes for:

- the MBR point for all values of $[n, k, d]$, performing optimal exact regeneration of all nodes (Section 2.2),
- the MSR point for all parameters $[n, k, d \geq 2k - 2]$, performing optimal exact regeneration of all nodes (Section 2.3), and
- simultaneously minimizing storage and bandwidth for all values of the parameters, however, with a small modification in the regenerating codes setup (Section 2.4).

Each of these codes are constructed for the atomic case of $\beta = 1$, and can be concatenated to obtain optimal codes for any higher value of $\beta$. An analysis of these codes is performed in Section 2.5, and its other applications are described – in particular their applications to data dissemination across a network, and for security against eavesdroppers.

Furthermore, in Section 2.6, a description in the product-matrix framework of a previously constructed code is provided, that is termed as the MISER code and described subsequently in detail in Chapter 3 using the interference alignment concept. This code is designed for the MSR point with parameters $[n = 2k, k, d = 2k - 1]$.

2.1 The Common Product-Matrix Framework

In all of the constructions described 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 below. It also endows the codes with implementation advantages that are discussed in Section 2.5.

2.1.1 Product-Matrix Description

Let \( u = [u_1, u_2, \ldots, u_B], u_i \in \mathbb{F}_q \), denote the vector of \( B \) message symbols. Each codeword in the distributed code can be represented by an \((n \times \alpha)\) matrix \( C(u) \) whose \( i \)th row \( c_i^t \) 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

\[
C(u) = \Psi M(u)
\]

of a \((n \times m)\) encoding matrix \( \Psi \) and a \((m \times \alpha)\) message matrix \( M(u) \). The entries of the matrix \( \Psi \) are fixed apriori and independent of the message symbols. We will refer to the \( i \)th row \( \psi_i^t \) 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 \( M(u) \) are a redundant collection of linear combinations of the message symbols \( \{u_i\} \). The collection of code matrices gives rise to the regenerating code

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

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 The Common Product-Matrix Framework

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 $\psi_i^t M$ (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 $\psi_j^t M$.
2.1 The Common Product-Matrix Framework

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.

2.1.5 Regeneration of a Failed Node

![Figure 2.4: Replacement node \( f \) connects to any \( d \) of the remaining nodes for regeneration.](image)

![Figure 2.5: Flow diagram illustrating regeneration process in the product-matrix framework.](image)

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
2.2 The Product-Matrix MBR Code Construction

an arbitrary subset \( \{ h_1, \ldots, h_d \} \) of \( d \) storage nodes which we will refer to as the \( d \) helper nodes. The \( j \)th helper node \( h_j \) in this set, passes on the message vector \( \psi^t_{h_j} M_{\mu_f} \) 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 the form in which it is to be stored in the replacement node (Fig. 2.5). Here again, the precise procedure is dependent on the particular construction.

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 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 \( \mathcal{C} \) 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 \( \mathcal{C} \) 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 (1.10) and (1.11). Equation (1.11) can be rewritten in the form:

\[
B = \left( \binom{k+1}{2} \right) + k(d-k).
\]

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

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

\[
M = \begin{bmatrix}
S & T \\
T^t & 0
\end{bmatrix}.
\]
Thus in the terminology of the product matrix framework, we have set the design parameter \( m = d \) in the present construction. Next, define the information-dispersion 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 2.2.1 (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

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

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

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

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 2.2.2 (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.

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

\[
\psi^f M.
\]

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
2.2 The Product-Matrix MBR Code Construction

computes the inner product $\langle \psi^t \ M, \psi_f \rangle$ and passes on the pair $\left( \psi_j^t, \langle \psi^t \ M, \psi_f \rangle \right)$ to the replacement node. Thus, in the present construction, the digest $\mu_f$ equals $\psi_f$ itself.

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

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

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

$$(M \psi_f)^t = \psi_f^t M$$  

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

2.2.1 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:

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

so that the message matrix $M$ is given by

$$M = \begin{bmatrix} u_1 & u_2 & u_3 & u_7 \\ u_2 & u_4 & u_5 & u_8 \\ u_3 & u_5 & u_6 & u_9 \\ u_7 & u_8 & u_9 & 0 \end{bmatrix}.$$  

We choose $\Psi$ to be the $(6 \times 4)$ Vandermonde matrix over $\mathbb{F}_7$ given by

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

(2.10)
2.2 The Product-Matrix MBR Code Construction

Figure 2.6: An example for the MBR 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.

Fig. 2.6 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, and 6 in the example), pass on their respective inner products \(\langle \Psi^\ell, 1 \rangle\) 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 \(\Psi^{-1}_{\text{rep}}\) where

\[
\Psi_{\text{rep}} = \begin{bmatrix}
1 & 2 & 4 & 1 \\
1 & 4 & 2 & 1 \\
1 & 5 & 4 & 6 \\
1 & 6 & 1 & 6
\end{bmatrix}
\] (2.11)

as explained in the proof of Theorem 5.2.2 above.

2.2.2 Systematic Version of the Code

Any exact-regenerating code can be made systematic through a non-singular transformation of the message symbols (this property of exact-regenerating codes will be discussed in greater detail in Section 3.1). In the present case, the matrix \(\Psi\) can be chosen in such a way that the code is automatically systematic. We simply make the choice:

\[
\Psi = \begin{bmatrix}
I_k & 0 \\
\Phi & \Delta
\end{bmatrix},
\] (2.12)
where $I_k$ is the $k \times k$ identity matrix, $\tilde{\Phi}$ and $\tilde{\Delta}$ are matrices of sizes $(k \times n - k)$ and $(d - k \times n - k)$ respectively, such that $\begin{bmatrix} \tilde{\Phi} & \tilde{\Delta} \end{bmatrix}$ 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 2.2.1.

### 2.2.3 Optimal Security from Eavesdroppers

In a distributed storage setup, an administrator may not have complete control over all storage nodes, for example, if they are spread out geographically. In such cases, it is imperative to secure the data from an eavesdropper who may gain access to one or more of these storage nodes.

More formally, we consider the possibility of an eavesdropper obtaining complete access to at most $l$ storage nodes ($1 \leq l \leq k$), in the sense that the eavesdropper can read all data stored in these $l$ nodes, and also read any data coming into or emanating from these nodes. It is desired that even under this situation, the eavesdropper should get no more than zero bits of information about the source data. Note that for the MBR point, since the data obtained during regeneration is the same as what is stored, it suffices to consider eavesdroppers that have read access to the data stored in the nodes.

Modifying the cut-set bound arguments in Section 1.1.2 (Figure 1.2), and noting that $l$ nodes are not permitted not contribute any useful information, the total amount of source data that can be dispersed in the network is upper bounded by

$$B \leq \sum_{i=1}^{k-1} \min\{\alpha, (d - i)\beta\}.$$  \hspace{1cm} (2.13)

In particular, for the MBR point, achieving this upper bound amounts to

$$B = \left(k\alpha - \frac{k(k - 1)}{2}\beta\right) - \left(l\alpha - \frac{l(l - 1)}{2}\beta\right).$$  \hspace{1cm} (2.14)

We will now infuse randomness in the MBR code $C$ presented above, in order to achieve the desired security. The new code so obtained is denoted $C_s$.

Choose the encoding matrix $\Psi$ in $C_s$ as a Cauchy matrix, i.e., one that has every sub-matrix of full rank. In general, we will require $\Psi$ to satisfy a condition, in addition to those required for $C$, that when restricted to the first $l$ columns, any $l$ rows of $\Psi$ should be linearly independent.

Next, modify the matrix $M$ as follows. Replace the $(ld - \frac{l(l-1)}{2})$ source symbols in the first $l$ rows and first $l$ columns of $M$ in code $C$ by a symbol chosen at random, independently with a uniform distribution across the elements of $\mathbb{F}_q$.

---

$^1$In general, any matrix, all of whose submatrices are of full rank will suffice.
For instance, in the example provided in Subsection 2.2.1, if \( l = 1 \), the matrix \( M \) is replaced by a second matrix

\[
M = \begin{bmatrix}
  r_1 & r_2 & r_3 & r_7 \\
  r_2 & u_4 & u_5 & u_8 \\
  r_3 & u_5 & u_6 & u_9 \\
  r_7 & u_8 & u_9 & 0
\end{bmatrix},
\]

where \( r_1, r_2, r_3 \) and \( r_7 \) are i.i.d. random variables drawn uniformly from the set of all elements in \( \mathbb{F}_q \). Thus, the first \( l \) rows and columns of \( M \) in \( C_s \) contain zero information pertaining to the \( (k\alpha - \frac{k(k-1)}{2} \beta) - (l\alpha - \frac{l(l-1)}{2} \beta) \) source symbols.

If the random symbols also are treated as source symbols, then the code \( C_s \) becomes identical to \( C \); thus the processes of reconstruction and regeneration can be performed as in code \( C \). The following theorem proves the security aspect of this code.

**Theorem 2.2.3** The code \( C_s \), designed for a given value of \( l \), gives out no more than zero information to an eavesdropper having read access to at most \( l \) nodes.

**Proof** Let \( \Psi_e \) be the \( l \times \alpha \) submatrix of \( \Psi \), corresponding to the \( l \) rows of \( \Psi \) to which the eavesdropper has access. Thus the eavesdropper has access to the \( ld \) symbols given by the \( l \times d \) matrix \( E \):

\[
E = \Psi_e M. \tag{2.16}
\]

Since the source symbols are linearly encoded, in order to retrieve any useful information, the eavesdropper needs to obtain a linear combination of the \( ld \) symbols in matrix \( E \) such that the random variables are eliminated.

Define \( M_e \) as a \( d \times d \) matrix, having its first \( l \) rows and first \( l \) columns identical to that of \( M \), and zeros elsewhere; thus, \( M_e \) consists of all the random symbols (and none of the source symbols) in \( M \). Define a second matrix

\[
E_e = \Psi_e M_e \tag{2.17}
\]

which is the set of symbols available to the eavesdropper, restricted to the random symbols alone. Now, this is identical to the case of a normal MBR code (i.e., without the security aspect) \( C \) designed for \( k = l \); hence the eavesdropper can decode all the random variables in terms of the source symbols. Thus, there are no more than \( \frac{l(l-1)}{2} \) linear combinations of the symbols in \( E \) that eliminate all random variables.

Now, since the matrix \( M_e \) is symmetric, so is the \( l \times l \) matrix given by \( \Psi_l M_e \Psi_l^t \) (\( = E_e \Psi_l^t \)). Making use of this observation, it is easy to see that the \( \frac{l(l-1)}{2} \) linear combinations of the elements of \( E_e \) that eliminate all random variables are:

\[
\xi_i^t E_e \Psi_l^t \xi_j - \xi_j^t E_e \Psi_l^t \xi_i, \quad \forall \ 1 \leq i < j \leq l, \tag{2.18}
\]
2.3 The Product-Matrix MSR Code Construction

where the unit vectors are of length \( l \). This translates to the linear combinations

\[
e^t E \Psi^t_l e_j - e^t_j E \Psi^t_l e_i, \quad \forall \, 1 \leq i < j \leq l
\]

on the set of symbols obtained by the eavesdropper. It may also be noted that these \( \binom{l(l-1)}{2} \) linear combinations are independent.

On the other hand, it turns out that each of these linear combinations results in all source symbols also getting eliminated:

\[
e^t_i E \Psi^t_l e_j - e^t_j E \Psi^t_l e_i = e^t_i \Psi^t_l M \Psi^t_l e_j - e^t_j \Psi^t_l M \Psi^t_l e_i = 0,
\]

since the matrix \( M \) is symmetric by construction.

Thus the eavesdropper can obtain no more than zero information pertaining to the source symbols. ■

2.3 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 (1.8) and (1.9). We begin by constructing an MSR code in the product-matrix format for \( d = 2k - 2 \) and will show in Section 2.3.3 how this can be very naturally extended to yield codes with \( d > 2k - 2 \).

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

\[
\alpha = d - k + 1 = k - 1
\]

and hence

\[
d = 2\alpha.
\]

Also,

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

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

$$M = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}. \quad (2.25)$$

Thus as was the case with the MBR construction of Section 2.2, we have set design parameter $m = d$.

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

$$\Psi = \begin{bmatrix} \Phi \\ \Lambda \Phi \end{bmatrix}, \quad (2.26)$$

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 2.3.1 (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 $[\phi_f^t \quad \lambda_f \phi_f^t]$ be the row of $\Psi$ corresponding to the failed node. Thus the $\alpha$ symbols stored in the failed node were

$$\begin{bmatrix} \phi_f^t & \lambda_f \phi_f^t \end{bmatrix} M = \phi_f^t S_1 + \lambda_f \phi_f^t S_2. \quad (2.27)$$

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.
2.3 The Product-Matrix MSR Code Construction

The replacement node computes \( \Psi_{rep}M_{\hat{\phi}_f} \) by aggregating the inputs from the \( d \) helper nodes where

\[
\Psi_{rep} = \begin{bmatrix}
\psi_{h1}^t \\
\psi_{h2}^t \\
\vdots \\
\psi_{hd}^t
\end{bmatrix}.
\]

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

\[
M_{\hat{\phi}_f} = \begin{bmatrix} S_1\hat{\phi}_f \\ S_2\hat{\phi}_f \end{bmatrix}.
\]

As \( S_1 \) and \( S_2 \) are symmetric matrices, the replacement node thus has acquired through transposition, both \( \hat{\phi}_f^tS_1 \) and \( \hat{\phi}_f^tS_2 \) using which it can obtain,

\[
\hat{\phi}_f^tS_1 + \lambda_f\hat{\phi}_f^tS_2 \tag{2.28}
\]

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

**Theorem 2.3.2 (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

\[
\Psi_{dc} = \begin{bmatrix} \Phi_{dc} & \Lambda_{dc}\Phi_{dc} \end{bmatrix} \tag{2.29}
\]

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

\[
\Psi_{dc}M = \begin{bmatrix} \Phi_{dc} & \Lambda_{dc}\Phi_{dc} \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} = \begin{bmatrix} \Phi_{dc}S_1 + \Lambda_{dc}\Phi_{dc}S_2 \end{bmatrix} \tag{2.30}
\]

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

\[
\begin{bmatrix} \Phi_{dc}S_1 + \Lambda_{dc}\Phi_{dc}S_2 \end{bmatrix} \Phi_{dc}^T = \Phi_{dc}S_1\Phi_{dc}^T + \Lambda_{dc}\Phi_{dc}S_2\Phi_{dc}^T \tag{2.31}
\]
Next, let the matrices $P$ and $Q$ be defined by

\begin{align*}
P &= \Phi_{DC} S_1 \Phi_{DC}^T \\
Q &= \Phi_{DC} S_2 \Phi_{DC}^T
\end{align*}

(2.32) \hspace{1cm} (2.33)

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

\[ P + \Lambda_{DC} Q. \]

(2.34)

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

\[ P_{ij} + \lambda_i Q_{ij}, \]

(2.35)

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

\[ P_{ji} + \lambda_j Q_{ji} = P_{ij} + \lambda_j Q_{ij} \]

(2.36)

where equation (2.36) follows from the symmetry of $P$ and $Q$. By construction, all the $\lambda_i$ are distinct and hence using equations (2.35) and (2.36), the DC can solve for the values of $P_{ij}$, $Q_{ij}$ for all $i \neq j$.

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

\[ \Phi_{DC} = \begin{bmatrix} \varphi_{1}^t \\
\vdots \\
\varphi_{\alpha+1}^t \end{bmatrix} \]

(2.37)

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

\[ \varphi_{i}^t S_1 \left[ \varphi_1 \cdots \varphi_{i-1} \varphi_{i+1} \cdots \varphi_{\alpha+1} \right]. \]

(2.38)

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

\[ \varphi_{i}^t S_1 \quad 1 \leq i \leq \alpha + 1. \]

(2.39)

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

\[ \begin{bmatrix} \varphi_{1}^t \\
\vdots \\
\varphi_{\alpha}^t \end{bmatrix} S_1 \]

(2.40)
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 $S_2$.

### 2.3.1 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:

\[
S_1 = \begin{bmatrix} u_1 & u_2 \\ u_2 & u_3 \end{bmatrix}, S_2 = \begin{bmatrix} u_4 & u_5 \\ u_5 & u_6 \end{bmatrix}
\]

so that the message matrix $M$ is given by

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

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

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

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

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

Fig. 2.7 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 $\langle \psi_\ell, \mathbf{1} \rangle$ where $\mathbf{1} = [1 \ 1]^t$. 

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

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

(2.45)

and decodes $S_1\psi_1$ and $S_2\psi_1$

$$
S_1\psi_1 = \begin{bmatrix} u_1 + u_2 \\ u_2 + u_3 \end{bmatrix}, S_2\psi_1 = \begin{bmatrix} u_4 + u_5 \\ u_5 + u_6 \end{bmatrix}
$$

(2.46)

Then the re-encoder in the replacement node processes $S_1\psi_1$ and $S_2\psi_1$ to obtain the data stored in the failed node as explained in the proof of Theorem 2.3.1 above.

### 2.3.2 Systematic Version of the Code

Every regenerating code has a systematic version and further, it can be made systematic through a process of message-symbol remapping (this property of exact-regenerating codes will be discussed in greater detail in Section 3.1). In the following, we perform this task explicitly for the code described above.

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 $B$ be a $k \times \alpha$ matrix containing the $B = k\alpha$ source symbols. We use

$$
\Psi_k M = B
$$

(2.47)
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 2.3.2. Next use this $M$ to obtain the code $\mathcal{C} = \Psi M$. Clearly, in this representation, the $k$ chosen nodes store the source symbols $B$ in uncoded form.

### 2.3.3 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 lemma.

**Theorem 2.3.3** An explicit $[n', n + 1, k' = k + 1, d' = d + 1]$ regenerating code $\mathcal{C}'$ that achieves the cut-set bound at the MSR point can be used to construct an explicit $[n, k, d]$ regenerating code $\mathcal{C}$ that also achieves the cut-set bound at the MSR point. Furthermore if $d' = ak' + b$ in code $\mathcal{C}'$, $d = ak + b + (a - 1)$ in code $\mathcal{C}$. If $\mathcal{C}'$ is linear, so is $\mathcal{C}$.

**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 $\mathcal{C}'$ in systematic form with the first $k'$ rows containing the $B'$ message symbols. Let $\mathcal{C}''$ be the subcode of $\mathcal{C}'$ consisting of all code matrices in $\mathcal{C}'$ whose top row is the all-zero row. Clearly, the subcode $\mathcal{C}''$ is of size $q^{B'-\alpha} = q^B$. Note that $\mathcal{C}''$ also possesses the same regeneration and reconstruction properties as does the parent code $\mathcal{C}'$.

Let the code $\mathcal{C}$ now be formed from subcode $\mathcal{C}''$ by puncturing (i.e, deleting) the first row in each code matrix of $\mathcal{C}''$. Clearly, code $\mathcal{C}$ is also of size $q^B$. We claim that $\mathcal{C}$ 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 $\mathcal{C}$ in $\mathcal{C}$. But this follows since, by augmenting the matrices of code $\mathcal{C}$ by placing at the top an additional all-zero row, we obtain a code matrix in $\mathcal{C}''$ and code $\mathcal{C}''$ has the property that the data can be recovered from any $(k + 1)$ rows of each code matrix in $\mathcal{C}''$. A similar argument shows that code $\mathcal{C}$ also possesses the regeneration property. This is illustrated in Figure 2.8.
Figure 2.8: An illustration of going from the case $[n, k, d]$ to $[n' = n + 1, k' = k + 1, d' = d + 1]$. Node $k + 1$ stores all zeros, and any data collector or a new node is always assumed to connect to this node.

Clearly if $C'$ is linear, so is code $C$. Finally, we have

$$d' = ak' + b$$
$$d + 1 = a(k + 1) + b$$
$$d = ak + b + (a - 1).$$

By iterating the procedure in the proof of Theorem 2.3.3 above $i$ times we obtain

**Corollary 2.3.4** 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 2.3.4 above.

**Corollary 2.3.5** 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 MSR Code Construction

2.3.4 Optimal Security from Eavesdroppers

**Eavesdropper with Access to Stored and Incoming Data**  An eavesdropper may be able to read all data stored in at most \( l \) storage nodes (\( 1 \leq l \leq k \)). It is desired that the eavesdropper should get no more than zero bits of information about the source data.

From equation 2.13, one can see that the eavesdropper will gain access to \( l\alpha \) symbols, and the maximum number of source symbols is \((k - l)\alpha\).

We will first design codes for the case of \( d = 2k - 2 \). These codes can be extended to the case \( d > 2k - 2 \) as in Section 2.3.3, from the parameter set \([n, k, d, l]\) to \([n - 1, k - 1, d - 1, l - 1]\).

\( \Phi \) is assumed to have an additional property that any \( l \) rows, when restricted to the first \( l \) columns, forms an \( l \times l \) invertible matrix.

A set of \( l\alpha \) symbols in \( M \) in the original code are replaced by random symbols as follows. Each of the \( l(k - l) + \frac{l(l+1)}{2} \) symbols in the first \( l \) rows and first \( l \) columns of \( S_1 \) is replaced by a random symbol. Also, each of the \( \frac{l(l-1)}{2} \) symbols in the intersection of the first \( l - 1 \) rows and first \( l - 1 \) columns in \( S_2 \) are also replaced by random symbols.

Clearly, the processes of reconstruction and regeneration can be carried out as in the original code. By a proof analogous to Theorem 2.2.3, it can be shown that this code also achieves maximum security against eavesdroppers who may gain access to at most \( l \) nodes.

**Eavesdropper with Access to Only the Stored Data**  For this case, it is easy to see that an upper bound on the total data is

\[
B \leq (k - l)\alpha \quad (2.48)
\]

since the eavesdropper will gain access to \( l\alpha \) distinct symbols. We will achieve this upper bound.

This code is also obtained by infusing randomness into the MSR code presented here, designed first for the case \( d = 2k - 2 \), with the extension to the case \( d > 2k - 2 \) performed via pruning, and the security property carrying over to this case as well.

\( \Phi \) is assumed to have an additional property that any \( l \) rows, when restricted to the first \( l \) columns, forms an \( l \times l \) invertible matrix. For example, choosing \( \Psi \) as a Vandermonde matrix satisfies this condition.

A set of \( l\alpha \) symbols in \( M \) in the original code are replaced by random symbols as follows. Each of the \( l(k - l) + \frac{l(l+1)}{2} \) symbols in the first \( l \) rows and first \( l \) columns of \( S_1 \) is replaced by a random symbol. Also, each of the \( \frac{l(l-1)}{2} \) symbols in the intersection of the first \( l - 1 \) rows and first \( l - 1 \) columns in \( S_2 \) are also replaced by random symbols.

Clearly, the processes of reconstruction and regeneration can be carried out as in the original code. By a proof analogous to Theorem 2.4.4, it can be shown that this code
also achieves maximum security against eavesdroppers who may gain access to at most \( l \) nodes.

### 2.4 The Ideal Regenerating Code: Simultaneous Minimization of Storage and Bandwidth

As discussed before, the two most important points on the storage-repair bandwidth tradeoff curve are the Minimum Storage Regeneration (MSR) point, which absolutely minimizes amount of storage at the nodes, and the Minimum Bandwidth Regenerating (MBR) point which absolutely minimizes the amount of download required to repair a failed node. In the traditional regenerating codes setup, these two points cannot be achieved simultaneously, and one has to compromise either the storage space or the repair bandwidth. However, in this section we construct codes having advantages of both these points – minimum storage as well as minimum repair bandwidth – by relaxing a certain condition in the regenerating codes setup.

#### 2.4.1 System Setting

A data file of size \( B \) symbols, over a finite field \( \mathbb{F}_q \) of size \( q \), is distributed 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 2.9. 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 to any \( k \) nodes of the *same* type to recover the data \(^2\). This is depicted in Fig 2.9a. As in the original setup, the reconstruction property mandates

\[
\alpha \geq \frac{B}{k}.
\]

(2.49)

The code constructed here achieves the minimum amount of storage at each node, and thus we have

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

(2.50)

A failed node is replaced by a node which is its exact replica, i.e., the regeneration is exact. A replacement node connects to \( k \) nodes of the other type (i.e, the type complementary to its own) \(^3\), downloading \( \beta = \frac{q}{k} \) symbols from each. This is depicted in

\(^2\) 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. The present code chooses \( d = k \) to minimize connectivity requirements.
2.4 The Ideal Regenerating Code: Simultaneous Minimization of Storage and Bandwidth

Figure 2.9: A hybrid code simultaneously minimizing storage and repair bandwidth. (a) A data collector connecting to \( k = 4 \) storage nodes of type 0, and (b) A new/replacement node, assigned type 1, connecting to 4 storage nodes of type 0.

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, this code achieves minimum repair bandwidth as well.

Thus, the value of the parameters \( \alpha \) and \( B \) for this code are

\[
\alpha = k\beta, \quad \text{(2.51)}
\]

\[
B = k^2\beta. \quad \text{(2.52)}
\]

2.4.2 The Code Construction

As seen in previous constructions presented in this report, the relation between the parameters as in equation (2.51) and (2.52) allows a divide and conquer approach in code design. Both \( \alpha \) and \( B \) are multiples of \( \beta \) and hence we can effectively employ the concept of striping.

We construct codes for \( \beta = 1 \) which we regard as ‘the atomic case’ 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 \( \beta = 1 \),

\[
\alpha = k, \quad \text{(2.53)}
\]

\[
B = k^2. \quad \text{(2.54)}
\]

From a practical standpoint, this divide-and-conquer approach will reduce the number of message symbols to be handled at a time, and hence will in general, be of lesser
complexity. For these reasons, in this section, we design codes for the atomic case.

The code is illustrated by an example for the case of $k = 3$ in the next subsection.

The encoding and message matrices for are defined as follows.

**Message Matrix**: Define a matrix $M_0$ as a $k \times k$ matrix consisting of the $k^2$ source symbols; and a second $k \times k$ matrix $M_1 = M_0^t$. The message matrix $M$ is defined as the $2k \times k$ matrix given by

$$M = \begin{bmatrix} M_0 \\ M_1 \end{bmatrix}. \quad (2.55)$$

**Encoding Matrix**: Define $\Psi_0$ as an $n_0 \times k$ matrix such that any $k$ are rows are linearly independent; also define $\Psi_1$ is an $n_0 \times k$ matrix also having the property that any $k$ of its rows are linearly independent. For example, one can choose either of the matrices to be a Vandermonde matrix or a Cauchy matrix. The encoding matrix $\Psi$ is the $(n_0 + n_1) \times 2k$ matrix

$$\Psi = \begin{bmatrix} \Psi_0 & 0 \\ 0 & \Psi_1 \end{bmatrix}. \quad (2.56)$$

**The Code Matrix**: Under the Product-Matrix framework, the $(n_0 + n_1) \times k$ code matrix $C$ is given by

$$C = \Psi M = \begin{bmatrix} \Psi_0 & 0 \\ 0 & \Psi_1 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \end{bmatrix} = \begin{bmatrix} \Psi_0 M_0 \\ \Psi_1 M_1 \end{bmatrix}. \quad (2.58)$$

Every storage node stores the $k$ symbols in one row of the code matrix; with the type 0 nodes storing the first $n_0$ rows and the type 1 nodes storing the remaining $n_0$ rows. The row of $\Psi$ corresponding to a node is termed as the digest of that node.

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 $\Psi_0 M_0$:

$$\psi_{(0,i)}^t M_0, \quad (2.59)$$

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

$$\psi_{(1,i)}^t M_1. \quad (2.60)$$

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

**Theorem 2.4.1 (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.
2.4 The Ideal Regenerating Code: Simultaneous Minimization of Storage and Bandwidth

**Proof** Let \( p \in \{0, 1\} \) be the type chosen of the failed node, and let this node correspond to the row \( \psi_{(p,f)}^t \) 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 symbols with the digest of the replacement node, i.e.,

\[
\left\langle \psi_{(\overline{p},h_i)}^t \ M_\overline{p} , \ \psi_{(p,f)}^t \right\rangle.
\]

(2.61)

It then passes on the pair: the computed data symbol along with its own digest,

\[
\left( \psi_{(\overline{p},h_i)}^t \ M_\overline{p} \ \psi_{(p,f)}^t , \ \psi_{(\overline{p},h_i)}^t \right)
\]

(2.62)

to the replacement node.

The replacement node aggregates the inputs from the \( k \) helper nodes as:

\[
\Psi_{\text{rep}} \ M_\overline{p} \ \psi_{(p,f)}^t , \ \text{where} \ \Psi_{\text{rep}} = \begin{bmatrix}
\psi_{(\overline{p},h_1)}^t \\
\psi_{(\overline{p},h_2)}^t \\
\vdots \\
\psi_{(\overline{p},h_k)}^t
\end{bmatrix}.
\]

(2.63)

By construction, the \( k \times k \) matrix \( \Psi_{\text{rep}} \) is invertible. Thus the replacement node now has access to \( M_\overline{p} \ \psi_{(p,f)}^t \). Furthermore, by construction, the \( M_p = M_{\overline{p}}^t \); a simple transpose operation by the re-encoder at the replacement node suffices to recover the desired symbols:

\[
\left( M_\overline{p} \ \psi_{(p,f)}^t \right)^t = \psi_{(p,f)}^t \ M_p.
\]

(2.64)

**Remark 2.4.2** 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 2.4.3 (Reconstruction)** In the code \( C \) presented, a data collector connecting to any \( k \) nodes of the same type can recover all the \( B \) message symbols.

**Proof** Let \( p \in \{0, 1\} \) be the type of the \( k \) nodes to which the data collector connects; denote these nodes as \( i_1, \ldots, i_k \). The data collector downloads the \( k^2 \) symbols

\[
\left\{ \psi_{(p,i_1)}^t \ M_p , \ \psi_{(p,i_2)}^t \ M_p , \ldots , \ \psi_{(p,i_k)}^t \ M_p \right\}.
\]

(2.65)
2.4 The Ideal Regenerating Code: Simultaneous Minimization of Storage and Bandwidth

Define a $k \times k$ matrix $\Psi_{DC}$ as

$$
\Psi_{DC} = \begin{bmatrix}
\psi_{t}^{(p,i_1)} \\
\psi_{t}^{(p,i_2)} \\
\vdots \\
\psi_{t}^{(p,i_k)}
\end{bmatrix}
$$

(2.66)

Thus, the symbols that the data collector obtains are the elements of the $k \times k$ matrix $\Psi_{DC}M_p$. The $k \times k$ matrix $\Psi_{DC}$ is a sub-matrix of $\Psi_p$, containing a set of $k$ rows, and hence by construction, is invertible. Thus the data collector can decode the entire set of $k^2$ source symbols in $M_p$. ■

2.4.3 An Example

Let the number of nodes of type 0 and type 1 be $n_0 = 4$ and $n_1 = 5$ respectively. Let $k = 3$, which gives $\alpha = 3$ and $B = 9$. Let us choose $q = 7$ so we are operating over $\mathbb{F}_7$.

Fill the matrix $M_0$ (and hence $M_1$) with the 9 message symbols $\{u_i\}_{i=1}^9$ as shown
2.4 The Ideal Regenerating Code: Simultaneous Minimization of Storage and Bandwidth

below:

\[ M_0 = M_1^t = \begin{bmatrix} u_1 & u_2 & u_3 \\ u_4 & u_5 & u_6 \\ u_7 & u_8 & u_9 \end{bmatrix}. \] (2.67)

As described above, the message matrix \( M \) is given by

\[ M = \begin{bmatrix} M_0 \\ M_1 \end{bmatrix}. \] (2.68)

Further, choose the encoding matrices as:

\[ \Psi_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \Psi_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 2 \end{bmatrix}. \] (2.69)

Figure 2.10 shows at the top, the \((4 \times 3)\) code matrix \( \Psi_0 M_0 \) for the type 0 nodes, and at the bottom, the \((5 \times 3)\) code matrix \( \Psi_1 M_1 \) for the type 1 nodes, with entries expressed as functions of the message symbols \( \{u_i\} \).

The rest of the figure explains how regeneration of a failed, node 1 of type 0, takes place. The replacement node chooses to retain its type and passes its digest \([1 0 0]\) to nodes 2, 3 and 4 of type 1. These helper nodes pass back their respective inner products \( < \psi_{(1,i)} M_i, \leq_1 > \ (i = 2, 3, 4) \), along with their own digests. The aggregator in the replacement node obtains the three symbols

\[ \begin{bmatrix} u_3 \\ u_1 + u_2 + u_3 \\ u_1 + 2u_2 + 3u_3 \end{bmatrix}, \] (2.70)

from which the partial decoder recovers the data stored in the failed node by multiplying by \( \Psi_{\text{rep}}^{-1} \), where

\[ \Psi_{\text{rep}} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix}. \] (2.71)

as explained in the proof of Theorem 2.4.1 above.

2.4.4 Optimal Security from Eavesdroppers

We consider again the possibility of an eavesdropper obtaining complete access to at most \( l \) storage nodes \((1 \leq l \leq k)\), in the sense that the eavesdropper can read all data stored
in these \( l \) nodes, and also read any data coming into or emanating from these nodes. It is desired that the eavesdropper should get no more than zero bits of information about the source data.

An eavesdropper connecting to \( l \) nodes of the same type acquires access to \( l\alpha \) distinct symbols. To satisfy the reconstruction property for a data collector, it is required that these symbols carry precisely \( l\alpha \) units of information. However, since an eavesdropper should not be able to obtain any information regarding the source symbols, an easy upper bound on the total file size that can be stored across the network is given by:

\[
B \leq (k - l)\alpha. \tag{2.72}
\]

In this section, we show that the ideal regenerating code can be used to achieve this upper bound. Clearly, it suffices to prove the same for a single stripe of the code.

Define an \( l \times k \) matrix \( R \) having its \( lk \) elements as \( lk \) distinct random symbols that are i.i.d. and drawn uniformly from the elements of \( \mathbb{F}_q \). Partition this matrix into two sub-matrices

\[
R = [R_a \quad R_b], \tag{2.73}
\]

where \( R_a \) and \( R_b \) have dimensions \( l \times l \) and \( l \times (k - l) \) respectively. Further, let \( \theta_0 \) and \( \theta_1 \) be two distinct elements drawn arbitrarily from \( \mathbb{F}_q \).

Define two matrices \( U_a \) and \( U_b \) with dimensions \((k-l) \times (k-l)\) and \( l \times (k-l) \) respectively, and set the \((k-l)k\) elements of these matrices as the \((k-l)k\) source symbols.

Finally, define the message matrices as

\[
M_0 = \begin{bmatrix} R_a & R_b + \theta_1 U_b \\ R_b^t + \theta_0 U_b^t & U_a \end{bmatrix}, \quad \text{and} \quad M_1 = M_0^t. \tag{2.74}
\]

Choose the encoding matrices \( \Psi_0 \) and \( \Psi_1 \) as Cauchy matrices, or matrices that have every sub-matrix of full rank.

For instance, if \( k = 4 \) and \( l = 2 \), the message matrices can be given by

\[
M_0 = M_1^t = \begin{bmatrix} r_1 & r_2 & r_5 + u_5 & r_6 + u_6 \\ r_3 & r_4 & r_7 + u_7 & r_8 + u_8 \\ r_5 & r_7 & u_1 & u_2 \\ r_6 & r_8 & u_3 & u_4 \end{bmatrix}. \tag{2.76}
\]

where the values of \( \theta_0 \) and \( \theta_1 \) have been chosen as 0 and 1 respectively.

Clearly, the processes of reconstruction and regeneration can be carried out as in the original code \( C \). Further, the following theorem proves the security aspect of this code.

**Theorem 2.4.4** The code presented here, designed for a given value of \( l \), gives out no
more than zero information to an eavesdropper having read access to at most $l$ nodes.

**Proof** We will first consider the special case when the eavesdropper gains control of $l$ nodes of the same type, say type $p$. Let $\Psi_e$ be the $l \times k$ submatrix of $\Psi_p$, corresponding to the $l$ rows of $\Psi_p$ to which the eavesdropper has access. Thus the eavesdropper has access to the $lk$ symbols given by the $l \times k$ matrix $E$:

$$E = \Psi_e M_p.$$  \hfill (2.77)

The eavesdropper, in order to retrieve any useful information, needs to obtain a linear combination of the $ld$ symbols in matrix $E$ such that the random variables are eliminated.

Define $M_e$ as a $k \times k$ matrix, obtained from the matrix $M_p$ considering only the random variables, i.e.,

$$M_e = \begin{bmatrix} R_a & R_b \\ R_b^t & 0 \end{bmatrix}. \hfill (2.78)$$

Thus, the matrix

$$E_e = \Psi_e M_e = \Psi_e \begin{bmatrix} R_a & R_b \\ R_b^t & 0 \end{bmatrix} \hfill (2.79)$$

$$= \Psi_e \begin{bmatrix} R_a & R_b \\ R_b^t & 0 \end{bmatrix} \hfill (2.80)$$

is the set of symbols available to the eavesdropper, restricted to the random symbols alone.

Thus, the eavesdropper can recover all $lk$ random variables in terms of the source symbols; however, there are only $lk$ symbols available to the eavesdropper. Hence, the eavesdropper cannot eliminate any random variable, and is thus prevented from obtaining any information pertaining to the source symbols.

We now proceed to the general case when the eavesdropper gains control of $m$ ($\leq l$) nodes of type 0 and $l - m$ nodes of type 1. Let $\Psi_{e0}$ and $\Psi_{e1}$ be the $m \times k$ and $(l - m) \times k$ submatrices of $\Psi_0$ and $\Psi_1$ respectively, corresponding to the rows to which the eavesdropper has access. Thus the eavesdropper has access to the $lk$ symbols given by the two matrices: an $m \times k$ matrix $E_0$ and an $(l - m) \times k$ matrix $E_1$ as

$$E_0 = \Psi_{e0} M_0 \hfill (2.81)$$

$$E_1 = \Psi_{e1} M_1. \hfill (2.82)$$

First, we show that this set of $lk$ symbols contains at most $m(l - m)$ dependent combinations of the random symbols. As before, let $M_e$ be a $k \times k$ matrix, obtained from
the matrix $M_0$ considering only the random variables, i.e.,

$$M_e = \begin{bmatrix} R_a & R_b \\ R_b^t & 0 \end{bmatrix}. \tag{2.83}$$

Define three sub-matrices of $M_e$: (i) $m \times l$ matrix $U_a$ as the intersection of first $m$ rows and first $l$ columns of $M_e$, (ii) $(k-m) \times (l-m)$ matrix $U_b$ as the intersection of rows $m+1$ through $k$ and first $l-m$ columns of $M_e$, and (iii) $m \times (k-l)$ matrix $U_c$ as the intersection of rows $l-m+1$ through $l$ and columns $l+1$ through $k$ of $M_e$. Note that all symbols in these three matrices are distinct. In the following, 'other symbols' will mean the random symbols that are not a part of any of these three matrices. Using symbols from nodes of type 1, all symbols in $U_b$ can be derived in terms of the other symbols, followed by all symbols in matrices $U_c$ and $U_a$. Since there are $lk - lm + m^2$ symbols in these three matrices, there are at least these many independent combinations of the random symbols.

Next, we explicitly derive the $m(l-m)$ dependent combinations, and show that each of these combinations leads to the source symbols also being eliminated. Since $M_1 = M_0'$, the linear combinations given by

$$E_0 \Psi_1^t - \Psi_0 E_1^t \tag{2.84}$$

eliminate all random symbols. These linear combinations are clearly independent in terms of the symbols in $E_0$ and $E_1$. However, clearly, each of these linear combinations leads to the source symbols also getting cancelled. Hence, the eavesdropper will not be able to obtain any information pertaining to the source symbols.

2.4.5 Flexible Reconstruction and Regeneration

We introduce a flexible class of regenerating codes, which codes enable a data-collector (or replacement node) to perform reconstruction (or regeneration) by connecting to an arbitrary number of storage nodes, downloading arbitrary amounts of data from each, provided that the total amount of data downloaded is at least $B$ (or a pre-defined repair-bandwidth $\gamma$). This has several advantages, since nodes can exploit higher connectivity, or mitigate the effect of links having unequal bandwidths. This framework is the focus of Chapter 7, however, the definition given above will suffice to understand the rest of this subsection.

The ideal regenerating code can perform optimal flexible reconstruction and regeneration, and the two theorems below outline the explicit method to perform this task.

**Theorem 2.4.5** In the code $C$ presented here, a data collector downloading $\mu_1, \ldots, \mu_n$ symbols from nodes $1, \ldots, n$ of the same type can recover the entire source data provided
the set \( \{\mu_i\}^n_{i=1} \) satisfy the following conditions:

\[
\mu_i \in \{0, \ldots, \alpha\} \quad \forall i, \quad \text{and} \quad \sum_{i=1}^{n} \mu_i \geq B
\]

Proof Clearly, it suffices to prove for the case

\[
\sum_{i=1}^{n} \mu_i = B,
\]

and hence we will assume this case.

The parameters of the system are \((k, B)\). Let \( \beta = B/k^2 \), which gives \( \alpha = k\beta \). The code is divided into \( \beta \) stripes. Let the message matrices corresponding to each of these stripes be denoted by \( M^{(1)}, M^{(2)}, \ldots, M^{(\beta)} \), each matrix \( M^{(i)} \) having a dimension of \( k \times k \). The code for stripe \( i \) \((i = 1, \ldots, \beta)\), is given by \( \Psi M^{(i)} \). Defining a \( k \times \alpha \) matrix

\[
\mathbf{M} = \begin{bmatrix} M^{(1)} & M^{(2)} & \cdots & M^{(\beta)} \end{bmatrix},
\]

the overall code can be written as the \( n \times \alpha \) matrix:

\[
\Psi \mathbf{M},
\]

where the \( \alpha \) symbols stored in node \( l \) are the symbols in the \( l \)th row of the code matrix. Let \( \mathbf{m}_j \) be the \( j \)th column of \( \mathbf{M} \), and \( \psi_l \) as the encoding vector of node \( l \) which is also the \( l \)th row of \( \Psi \). Then, the \( j \)th symbol stored in node \( l \) is:

\[
\psi_l^t \mathbf{m}_j.
\]

At the DC, a simple algorithm needs to be run to determine which symbols it wants to ask from the it connects to nodes. For \( l = 1, \ldots, n \), the DC should ask node \( l \) to pass the \( \mu_l \) symbols:

\[
\left\{ \psi_l^t \mathbf{m}_{j_1}, \ldots, \psi_l^t \mathbf{m}_{j_{\mu_l}} \right\}
\]

where

\[
j_i = \left( \sum_{\nu=1}^{l-1} \mu_{\nu} + i \right) \mod \alpha, \quad i = 1, \ldots, \mu_l.
\]

Clearly, the data collector obtains exactly \( k \) encodings of every column of the message.
matrix $M$; denote the $k$ encodings of the $j$th column as

$$\Psi_j m_j,$$  \hspace{1cm} (2.93)

where $\Psi_j$ is a $k \times k$ sub-matrix of $\Psi$ with its rows as the encoding vectors of the $k$ nodes that contributed their $j$th symbols. By construction, the matrix $\Psi_j$ is invertible, thus enabling the data collector to decode the entire column $m_j$ of symbols, for all $j \in \{1, \ldots, \alpha\}$. $\blacksquare$

**Theorem 2.4.6** In the code $C$ presented here, a replacement node downloading $\beta_1, \ldots, \beta_n$ symbols from nodes $1, \ldots, n$ of the complementary type can perform exact regeneration provided the set $\{\beta_i\}^n_{i=1}$ satisfy the following conditions:

$$\beta_i \in \left\{0, \ldots, \frac{B}{k^2}\right\} \quad \forall i, \quad \text{and}$$

$$\sum_{i=1}^{n} \beta_i \geq \alpha$$  \hspace{1cm} (2.94)

**Proof** The first condition says that no assisting node will be forced to pass more than one symbol stored in it from the same stripe, and the second condition simply mandates the replacement node to download at least as much as it will store. The proof is identical to that for flexible reconstruction in Theorem 2.4.5. $\blacksquare$

2.5 Analysis and Advantages of the Codes

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

2.5.1 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 symbol 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 all the 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.

If one chooses to use a Reed-Solomon code as the \( \psi \) vectors, then each node can be described using only an index, which further reduces the overheads.

### 2.5.2 Applicability to Arbitrary \( n \)

All other existing, explicit constructions of exact-regenerating codes \([9–12, 15]\) restrict the value of \( n \) to be \( d + 1 \). In contrast, the codes presented here 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 the system size is very natural. For example, in peer-to-peer systems, individual nodes are free to come and go at will.

Thus, in addition to the typical distributed storage applications, these codes are highly applicable to systems with a variable number of nodes, such as peer-to-peer storage systems where nodes can come and go at their own will, or systems where new nodes may be added over time. The system works optimally 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 signatures to then new nodes. Thus, these codes can also efficiently perform dissemination of data, from the source to the storage nodes, as described below.

### 2.5.3 Data Dissemination

All three codes presented here can be effectively used to disseminate data, assumed to be generated at a point, across a network; it is also mandated that the regeneration and reconstruction properties continue to hold across the network. This is achieved by treating a storage node that has not yet received data as a replacement for a failed node. The process of distribution of data is described pictorially in Figure 2.11.

In addition to minimizing the total amount of data transfer, the distributed nature of the data dissemination has several other advantages:

(i) Load balancing is achieved, and the traffic is uniform across the network.

(ii) The source node can become unavailable after the initial transmission of coded data to the storage nodes in its neighbourhood.
Figure 2.11: Data dissemination via the product-matrix codes. (a) The point source transfers coded data to its neighbouring nodes, (b) which in turn help the nodes in their neighbourhood, that have not yet received data, to regenerate. This stage onwards, the source node need not be extant. (c) Once data has been disseminated in the manner described, a data collector can connect to any \( k \) nodes to recover the entire data, and (d) a replacement node or a new node added to the system can be regenerated by connecting to any \( d \) nodes.
(iii) The number of hops across the network that any data packet travels will be considerably lesser, as compared to the case of direct transmission from the source node.

2.5.4 Complexity

Linearity and field size

The codes are linear over a chosen finite field $\mathbb{F}_q$, i.e., the source symbols are from this finite field, and any stored symbol is a linear combination of these symbols over $\mathbb{F}_q$. 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 divide the entire data into stripes of sizes corresponding to $\beta = 1$. 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 just a scalar. Moreover with this choice, the encoding, reconstruction, and regeneration operations are, for the most part, identical to encoding or decoding of conventional Reed-Solomon codes.

2.6 Description of a Previously Constructed MSR Code in the Product-Matrix Framework

A code structure that guarantees exact repair of just the systematic nodes is provided in Chapter 3, for the MSR point with parameters $d = (n - 1) \geq 2k - 1$. Following its original presentation [12], it was shown in [15] that for this set of parameters, this code 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 also designed and presented in [15]. In this section, we provide a simple description of this code in the product-matrix framework. Chapter 3 will subsequently present the code in a subspace viewpoint, which will highlight the key role that the concept of interference alignment plays in this construction.
We begin with the case $d = 2k - 1$, since the code as well as both reconstruction and regeneration algorithms can be extended to larger values of $d$ by making use of Corollary 2.3.5.

At the MSR point, with $d = n - 1 = 2k - 1$, we have from equations (1.8) and (1.9) that

\[
\begin{align*}
\alpha &= d - k + 1 = k, \\
B &= k\alpha = k^2.
\end{align*}
\]

Let $S(u)$ be a $(k \times k)$ matrix whose entries are precisely the $B$ message symbols $\{u_i\}_{i=1}^B$ and $M(u)$ be the $(2k \times k)$ message matrix given by:

\[M(u) = \begin{bmatrix} S(u) \\ [S(u)]^t \end{bmatrix}.\]  

Next, let $\Phi$ be a $(k \times k)$ Cauchy matrix over $\mathbb{F}_q$ and $\epsilon$ a scalar chosen such that

\[\epsilon \neq 0, \quad \epsilon^2 \neq 1.\]

Let $\Psi$ be the $(n \times 2k)$ encoding matrix given by

\[\Psi = \begin{bmatrix} I & 0 \\ \Phi & \epsilon\Phi \end{bmatrix}.\]

The code constructed in [12,15] can be verified to have an alternate description as the collection of code matrices of the form

\[C(u) = \Psi M(u) = \begin{bmatrix} S(u) \\ \Phi(S(u) + \epsilon[S(u)]^t) \end{bmatrix}.\]

As before, in the sequel, we will for simplicity write $S$ in place of $S(u)$, and similarly with $M$ and $C$. Note that the first $k$ nodes store the message symbols in uncoded form and hence correspond to the systematic nodes. A simple description of the reconstruction and regeneration properties of the code is presented below.

If $A$ is an $(m_1 \times m_2)$ matrix and $P, Q$ arbitrary subsets of $\{1, \ldots, m_1\}$ and $\{1, \ldots, m_2\}$ respectively, we will use $A_{(P,Q)}$ to denote the submatrix of $A$ containing only the rows and columns respectively specified by the indices in $P$ and $Q$. For the cases when either $P = \{1,2,\ldots,m_1\}$ or $Q = \{1,2,\ldots,m_2\}$, we will simply write ‘all’ in place of either $\{1,2,\ldots,m_1\}$ or $\{1,2,\ldots,m_1\}$.

**Theorem 2.6.1 (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 \( P = \{n_1, \ldots, n_i\} \) and \( Q = \{m_1, \ldots, m_{(k-i)}\} \) be the systematic and non-systematic nodes respectively to which the data collector connects. Let \( T = \{1, \ldots, k\} \setminus \{n_1, \ldots, n_i\} \) i.e., the systematic nodes to which the DC does not connect. Then the DC is able to access the \( k \alpha \) symbols

\[
\begin{bmatrix}
S_{(P,\text{all})} \\
\Phi_{(Q,\text{all})}(S + \epsilon S^t)
\end{bmatrix}
\]  

(2.102)

Thus the DC has access to the \( i \) rows of \( S \) indexed by the entries of \( P \) and consequently, has access to the corresponding columns of \( S^t \) as well.

Consider the \( i \) columns of \( \Phi_{(Q,\text{all})}(S + \epsilon S^t) \) indexed by \( P \). Since the entries of these columns in \( S^t \) are known, the DC has access to \( \Phi_{(Q,\text{all})}S_{(\text{all},P)} \). Now since the \( i \) rows of \( S \) indexed through \( P \) are also known, the DC has thus access to the product

\[
\Phi_{(Q,T)}S_{(T,P)}.
\]

(2.103)

Now as \( \Phi_{(Q,T)} \) is non-singular, being a \((k - |P|, k - |P|)\) sub-matrix of a Cauchy matrix, the DC can recover \( S_{(T,P)} \). In this way, the DC has recovered all the entries in the rows of \( S \) indexed by \( P \) as well as all the entries in the columns of \( S \) indexed by \( P \). Clearly, the same statement holds when \( S \) is replaced by \( S^t \). Thus the DC has access to the product:

\[
\Phi_{(T,T)}(S + \epsilon S^t)_{(T,T)}.
\]

(2.104)

Again \( \Phi_{(T,T)} \) being a sub-matrix of a Cauchy matrix is of full rank and enables the DC to recover \((S + \epsilon S^t)_{(T,T)}\). It is easy to see that from the diagonal elements of this matrix, all the diagonal elements of \( S_{(T,T)} \) can be obtained. The non-diagonal elements are however of the form \( S_{ij} + \epsilon S_{ji} \) and \( S_{ji} + \epsilon S_{ij} \) for \( l \in T, j \in T, l \neq j \). As \( \epsilon^2 \neq 1 \), all the non-diagonal elements of \( S_{(T,T)} \) can also be decoded. In this way, the DC has recovered all the \( B \) entries of \( S \).

Theorem 2.6.2 (Exact Regeneration) In the code \( C \) presented, exact regeneration of any failed node can be achieved by connecting to the remaining \( n - 1 \) nodes.

Proof Systematic node regeneration: Suppose the \( i \)th systematic node has failed. The \( k \) symbols stored in it are \( e_i^tS \). The new node replacing node \( i \) obtains the following \( n - 1 \) symbols from the remaining nodes

\[
\begin{bmatrix}
\tilde{I} & 0 \\
\Phi & \epsilon \Phi
\end{bmatrix}
\begin{bmatrix}
S \\
S^t
\end{bmatrix}
\begin{bmatrix}
e_i \\
\end{bmatrix} = 
\begin{bmatrix}
\tilde{I}S e_i \\
\Phi(S + \epsilon S^t)e_i
\end{bmatrix}.
\]

(2.105)

where \( \tilde{I} \) is a \((k - 1 \times k)\) matrix which is the identity matrix with \( i \)th row removed.

Since \( \Phi \) is full rank by construction, the DC has access to

\[
[(S + \epsilon S^t)e_i],
\]

(2.106)
and equivalently,

$$[\epsilon \epsilon_S] + [S']_i. \quad (2.107)$$

Hence the new node has access to

$$\begin{bmatrix}
\hat{I} & 0 \\
\Phi & \epsilon \Phi \\
\epsilon S' & \epsilon S'
\end{bmatrix}
\begin{bmatrix}
S' \\
\xi_i
\end{bmatrix}. \quad (2.108)$$

Since $\epsilon \neq 1$, the $2k \times 2k$ matrix on the left is non-singular and the new node can recover $S'\xi_i$, which is the same as the symbols $\epsilon_S$.

**Non-systematic node regeneration:** Let $\phi'_f$ be the row of $\Phi$ corresponding to the failed node. Then the $k$ symbols stored in the failed node are $\phi'_f(S + \epsilon S')$. The replacement node replacing the failed node requests and obtains the following $n - 1$ symbols from the remaining nodes,

$$\begin{bmatrix}
I & 0 \\
\Phi_{k-1} & \epsilon \Phi_{k-1}
\end{bmatrix}
\begin{bmatrix}
S' \\
\phi_f
\end{bmatrix} = \phi'_f S' \phi_f. \quad (2.109)$$

where $\Phi_{k-1}$ corresponds to the submatrix of $\Phi$ corresponding to the remaining non-systematic nodes. This gives the replacement node access to $S\phi_f$ and therefore to

$$(S\phi_f)^t \phi_f = \phi'_f S' \phi_f. \quad (2.110)$$

Hence the replacement node has access to

$$\begin{bmatrix}
I & 0 \\
\Phi_{k-1} & \epsilon \Phi_{k-1}
\end{bmatrix}
\begin{bmatrix}
S' \\
\phi_f
\end{bmatrix} = \phi'_f S' \phi_f \quad (2.111)$$

The matrix on the left is easily verified to be non-singular and thus the replacement node acquires $S\phi_f$ and $S'\phi_f$ individually from which it can derive the desired vector $(\phi_f S + \epsilon \phi_f S')$. 

■
Chapter 3

Interference Alignment based Codes for the MSR Point

The focus of this chapter is on code constructions for the MSR point, for which the concept of Interference Alignment is employed. In fact, it is shown in Section 3.3 that interference alignment is necessary for construction of such codes.

In this chapter, we restrict our attention to systematic MDS regenerating codes where \( k \) out of the \( n \) nodes store data in uncoded form. Systematic codes possess the advantage that a data collector connecting to the \( k \) systematic nodes can reconstruct the data without need for any further processing of the data. For the data storage network to sustain itself in systematic form, the exact regeneration of a failed systematic node is called for, i.e., the new node replacing a failed systematic node should store data identical to what was previously stored in the failed node. The preferential status of a systematic node makes its fast regeneration a priority and this motivates the interest in minimizing the repair bandwidth for the exact regeneration of systematic nodes. The non-systematic nodes are not the focus, and are assumed to be regenerated by downloading higher amounts of data, possibly the complete file size \( B \).

The first major result in this chapter is the construction of a family of MDS codes for \( d = n - 1 \geq 2k - 1 \) that enable the exact regeneration of systematic nodes while achieving the cut set bound on repair bandwidth (Section 3.4). This code that makes use of the concept of interference alignment, and we have termed this code the MISER (for MDS, Interference-aligning, Systematic, Exact-Regenerating) code\(^1\). This code is constructed for the atomic case of \( \beta = 1 \), and can be concatenated to obtain optimal codes for any higher value of \( \beta \).

We also obtain insights into the structure of an MDS code meeting the lower bound on repair bandwidth; under the assumption that the global kernels passed by a node \( i \) for

---

\(^1\)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.
repair of the $k$ systematic nodes are linearly independent, it turns out that the structure of the MISER code is essentially mandated.

A second major result is the proof establishing that it is impossible to construct a code for the MSR point with $d < 2k - 3$ for the atomic case of $\beta = 1$, provided in Section 3.5. This non-existence result is clearly of interest in the light of on-going efforts to construct codes with $\beta = 1$ meeting the cut-set bound [9–11,21] on exact repair.

Furthermore, existence of codes achieving optimal exact regeneration of the systematic nodes for $d = (n - 1) \geq 2k - 3$ is proved in Section 3.6. A coding scheme for any $(k, \alpha)$ parameter set is provided in Section 3.7, that is optimal for $k \leq \alpha$. Section 3.8 shows that a large part of the design choices made for the explicit construction in Section 3.4 is in fact forced, and this leads towards the uniqueness of our code construction.

A summary of the results contained in this chapter, with respect to the parameters $d$ and $k$, is provided in Figure 3.1.

![Figure 3.1](image)

Figure 3.1: Summary of the results contained in the chapter, with respect to the existence of MDS regenerating codes enabling exact repair of systematic nodes.

The distributed storage problem for exact regeneration of systematic nodes at the MSR point is cast as a traditional network coding problem in Appendix A. Turns out that this is a non-multicast problem with a large number of sources and sinks, about which very less is known in the literature. The insights obtained en route to constructing distributed storage codes using interference alignment are generalized to obtain a set of highly intuitive conditions for code design in a general network of non-multicast type.

Recall from equations 1.6, the cut-set bound for the MSR point as:

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

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

The immediate question that this raises is 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 turns out that the same bound on the parameters \((\alpha, \beta)\) appearing in (1.6) still applies and this is established in Section 3.2.

We will first introduce the concepts of global kernels and nodal subspaces, and define equivalence between exact regenerating codes.

### 3.1 Global Kernels, Nodal Subspaces and Equivalent Codes

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 the case of linear regenerating codes, we have the relation:

\[
[u_1 \ u_2 \ \cdots \ u_B] [G_1 \ G_2 \ \cdots \ G_n] = [c_{11} \cdots c_{1\alpha} | c_{21} \cdots c_{2\alpha} | \cdots | c_{n1} \cdots c_{n\alpha}]
\]  

where the generator matrix \(G = [G_1 \ G_2 \ \cdots \ G_n]\) is composed of the \(n\) component generator sub-matrices

\[
G_i = \begin{bmatrix} g_{i1} & g_{i2} & \cdots & g_{i\alpha} \end{bmatrix},
\]

each associated to a distinct node. Adopting the terminology of network coding, the column vector \(g_{ij}\) will be termed the \(j\)th global kernel associated to the \(i\)th node. 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

\[
W_{id+1} \subseteq <h_{i_1}, h_{i_2}, \cdots, h_{id}> .
\]

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

\[
G, \quad \text{and} \quad AG
\]

are equivalent.
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.

**Systematic Version of Exact-Regenerating Codes**  It also follows that any exact-regenerating code is equivalent to a systematic, exact-regenerating code. To see this, let us introduce the following alternate notation for the columns of the generator matrix

\[
G = [\gamma_1, \gamma_2, \ldots, \gamma_{n\alpha}]
\]

and let

\[
\{\gamma_{a_1}, \gamma_{a_2}, \ldots, \gamma_{a_B}\} \subseteq \{g_{ij} | 1 \leq i \leq n, 1 \leq j \leq \alpha\}
\]

denote a set of \(B\) linearly independent column vectors drawn from the generator matrix \(G\). That such a subset is guaranteed to exist follows from the reconstruction property of a regenerating code. Let \(\Gamma\) be the \((B \times B)\) matrix

\[
\Gamma = \begin{bmatrix}
\gamma_{a_1} & \gamma_{a_2} & \cdots & \gamma_{a_B}
\end{bmatrix}.
\]

Then we have the relation:

\[
[u_1, u_2, \ldots, u_B] \Gamma = [c'_1, c'_2, \cdots c'_{a_B}],
\]

where \(\{c'_1, c'_2, \cdots, c'_{a_B}\}\) is a linear ordering of the code symbols. It follows that if we wish to encode in such a way that the code is systematic with respect to code symbols \(\{c'_1, c'_2, \ldots, c'_{a_B}\}\), the input to be “fed” to the generator matrix \(G\) is

\[
u^T \Gamma^{-1}
\]

where \(u\) is the message vector.

### 3.2 Additional Notation and a Bound on the Repair Bandwidth

As described before, let \(G^{(m)}\) denote the generator matrix of node \(m\) with dimensions \(B \times \alpha\). Denote the \(\alpha\) symbols stored in node \(m\) by the vector \(z^{(m)}\). We have

\[
z^{(m)} = u^T G^{(m)}
\]

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

\[
[ G^{(1)} \quad G^{(2)} \quad \cdots \quad G^{(n)} ]
\]
Now, let us partition the $B$ source symbols into $k$ vectors, $u_i$ for $i = 1, \ldots, k$, consisting of $\alpha$ distinct source symbols each. Then the source vector $u$ can be written as

$$u = \begin{bmatrix} u_1 \\ \vdots \\ u_k \end{bmatrix}. \quad (3.5)$$

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

$$G^{(m)} = \begin{bmatrix} G_1^{(m)} \\ \vdots \\ G_k^{(m)} \end{bmatrix} \quad (3.6)$$

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

$$z^{(m)} = u^t G^{(m)} = \sum_{l=1}^{k} u_l^t G_l^{(m)}. \quad (3.7)$$

These sub-matrices are further partitioned as

$$G_l^{(m)} = \begin{bmatrix} g_{l,1}^{(m)} & \cdots & g_{l,\alpha}^{(m)} \end{bmatrix} \quad (3.8)$$

where each $g_{l,i}^{(m)}$ 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 $u_m$. Hence,

$$z^{(m)} = u_m \quad \text{for} \quad m = 1, \ldots, k \quad (3.9)$$

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

$$G_l^{(m)} = \begin{cases} I_\alpha & \text{if } l = m \\ 0_\alpha & \text{if } l \neq m \end{cases} \quad (3.10)$$

Now, making use of the partitions in the generator matrices, for any node $m$, we refer to $G_l^{(m)}$ as the \textit{component} along $u_l$, 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.

Since $\beta = 1$, a replacement node will download one symbol – and equivalently one
vector – from \(d\) existing nodes. We will denote the vector passed by node \(m\) for the regeneration of node \(l\) as

\[
\mathbf{v}^{(m,l)} = \begin{bmatrix}
\mathbf{v}^{(m,l)}_1 \\
\vdots \\
\mathbf{v}^{(m,l)}_k
\end{bmatrix}
\]  

(3.11)

where \(\mathbf{v}^{(m,l)}_i\) \((i = 1, \ldots, k)\) are \(\alpha\)-length column vectors reflecting the partitioning of the node generator matrices. As mentioned previously in Section 3.1, although the vector passed by a helper node will depend on the identities of the other helper nodes, for brevity we will not indicate the same in the notation.

We will further denote \(\mathbf{x}^{(m,l)}\) as the linear combination vector used by node \(m\) to generate \(\mathbf{v}^{(m,l)}\), i.e.,

\[
\mathbf{v}^{(m,l)} = \mathbf{G}^{(m)} \mathbf{x}^{(m,l)}
\]  

(3.12)

All notation is depicted in Figure 3.2.

---

**Figure 3.2:** Exact regeneration of systematic node \(l\): illustrating the notation used in Chapter 3.

---

**Bound on repair bandwidth for the MSR point**  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 (1.6) still applies. The proof given below is for the case when \(\beta = 1\); proof for higher values of beta is a simple extension.

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 \( \{\mathbf{y}^{(m_1, l)}, \ldots, \mathbf{y}^{(m_d, l)}\} \), \( G^{(l)} \) needs to be recovered. This implies that the dimension of the nullspace of the matrix

\[
\begin{bmatrix}
G^{(l)} & \mathbf{y}^{(m_1, l)} & \cdots & \mathbf{y}^{(m_d, l)}
\end{bmatrix}
\tag{3.13}
\]

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

\[
\begin{bmatrix}
G^{(l)} & \mathbf{y}^{(m_1, l)} & \cdots & \mathbf{y}^{(m_k, l)}
\end{bmatrix}
\tag{3.14}
\]

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 (3.13) to be of \( \alpha \) dimensions. Thus we need

\[
d - k + 1 \geq \alpha \tag{3.15}
\]

We say two vectors are aligned if they are linearly dependent. We briefly describe the concept of interference alignment in the next section, and discuss its application in the context of distributed storage.

### 3.3 Need for Interference Alignment in Regenerating Codes

Recently, the idea of ‘Interference Alignment’ in the context of wireless communication was proposed in [28], [29]. 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. Though the principles are the same, there are significant differences in the alignment techniques in wireless communication and that in distributed storage.

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 \( \{ u_1, u_2, u_3, u_4 \} \) be the source symbols. Node 1 stores \( \{ u_1, u_2 \} \) and node 2 stores \( \{ u_3, u_4 \} \). Nodes 3 and 4 are non-systematic nodes. Every symbol stored is associated to a global kernel vector having its four components along \( u_1, \ldots, u_4 \) 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 \( u_i \).

Consider failure of node 1, with nodes 2, 3 and 4 passing vectors \( v^{(2,1)} \), \( v^{(3,1)} \) and \( v^{(4,1)} \) for its regeneration. Since node 2 can only pass some function of \( u_3 \) and \( u_4 \), \( v^{(3,1)} \) and \( v^{(4,1)} \) alone need to provide all information about \( \{ u_1, u_2 \} \). Since two units of information about \( \{ u_1, u_2 \} \) are required, the components of \( v^{(3,1)} \) and \( v^{(4,1)} \) along \( \{ u_3, u_4 \} \), i.e., \( e_2^{(3,1)} \) and \( e_2^{(4,1)} \), constitute interference and need to be eliminated. The components along \( \{ u_1, u_2 \} \), i.e., \( e_1^{(3,1)} \) and \( e_1^{(4,1)} \) constitute the desired component.

The only way to eliminate the interference is using \( v^{(2,1)} \). Since node 2 can pass only one vector, to be able to cancel the interference from \( \{ u_3, u_4 \} \), both \( e_2^{(3,1)} \) and \( e_2^{(4,1)} \) 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.

An explicit code over \( \mathbb{F}_5 \) for the parameters chosen in the example is shown in Figure 3.3. 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 \( 2u_1 + 2u_2 + u_3 \) and \( 2u_1 + 4u_2 + 2u_3 \) respectively. Observe that the interfering component in both these symbols are aligned along \( u_3 \). Hence node 2 can pass \( u_3 \) and cancel the interference.

In the context of regenerating codes, interference alignment was first used by Wu and Dimakis in [9] 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 a part of the interference could be aligned. Due to this, the scheme achieves the tradeoff curve only for the case \( k = 2 \).

In the next section, we describe the construction of the MISER code which aligns interference and achieves the lower bound on the repair bandwidth as provided by the cut-set bound.

### 3.4 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.

3.4.1 An Example

Let the parameters for the example code be, \( B = 9, \ n = 6, \ k = 3, \ d = 5, \ \beta = 1 \). This gives \( \alpha = B/k = 3 \). All symbols to 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

\[
G^{(1)} = \begin{bmatrix} I_3 \\ 0_3 \\ 0_3 \end{bmatrix}, \quad G^{(2)} = \begin{bmatrix} 0_3 \\ I_3 \\ 0_3 \end{bmatrix}, \quad G^{(3)} = \begin{bmatrix} 0_3 \\ 0_3 \\ I_3 \end{bmatrix}.
\] (3.16)

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

\[
\Psi_3 = \begin{bmatrix} \psi_1^{(4)} & \psi_1^{(5)} & \psi_1^{(6)} \\ \psi_2^{(4)} & \psi_2^{(5)} & \psi_2^{(6)} \\ \psi_3^{(4)} & \psi_3^{(5)} & \psi_3^{(6)} \end{bmatrix}
\] (3.17)

be a \( 3 \times 3 \) such that any of its sub-matrix is full rank. A Cauchy matrix [33] is one such matrix and we choose this matrix for our construction.

**Definition 3.4.1 (Cauchy matrix)** A Cauchy matrix is a matrix with its \((i, j)^{th}\) 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

$$G^{(m)} = \begin{bmatrix}
2\psi_1^{(m)} & 0 & 0 \\
2\psi_2^{(m)} & \psi_1^{(m)} & 0 \\
2\psi_3^{(m)} & 0 & \psi_1^{(m)} \\
\psi_2^{(m)} & 2\psi_1^{(m)} & 0 \\
0 & 2\psi_2^{(m)} & 0 \\
0 & 2\psi_3^{(m)} & \psi_2^{(m)} \\
\psi_3^{(m)} & 0 & 2\psi_1^{(m)} \\
0 & \psi_3^{(m)} & 2\psi_2^{(m)} \\
0 & 0 & 2\psi_3^{(m)}
\end{bmatrix} \tag{3.18}$$

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 their components from the symbols in node 4. It is left to decode the source symbols $z_1$ which are encoded using the following matrix

$$G_1^{(4)} = \begin{bmatrix}
2\psi_1^{(4)} & 0 & 0 \\
2\psi_2^{(4)} & \psi_2^{(4)} & 0 \\
2\psi_3^{(4)} & 0 & \psi_3^{(4)}
\end{bmatrix}. \tag{3.19}$$

This diagonal matrix is full rank since the elements of a Cauchy matrix are non-zero, and hence the symbols $z_1^t$ can be easily obtained.

(c) *All three non-systematic nodes:* Now lets 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

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

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 \( \Psi_3^{-1} \) to get a matrix \( C_3 \) given by

\[
C_3 = C_2 \begin{bmatrix}
\Psi_3^{-1} & 0_3 & 0_3 \\
0_3 & \Psi_3^{-1} & 0_3 \\
0_3 & 0_3 & \Psi_3^{-1}
\end{bmatrix}.
\]

\[
C_3 = \begin{bmatrix}
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2
\end{bmatrix}.
\]

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

\[
C_4 = \begin{bmatrix}
2 & 0 & 1 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 1 & 0 \\
1 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 \\
\end{bmatrix}.
\] (3.23)

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. \(\tilde{z}_1\), are available to the data collector and their components can be subtracted from all the coded symbols. The data-collector is left to decode the symbols \(\tilde{z}_2\) and \(\tilde{z}_3\) which are encoded using the matrix,

\[
B_1 = \begin{bmatrix}G_2^{(4)} & G_2^{(5)} \\G_3^{(4)} & G_3^{(5)} \end{bmatrix}.
\] (3.24)

**Claim 2:** The data collector can recover the source symbols \(\tilde{z}_2\) and \(\tilde{z}_3\) encoded by the matrix \(B_1\).

**Proof** We first interchange certain columns which will make the job simpler. For \(i = 2, 3, 1\) (in this order), group the \(i^{th}\) columns of the two non-systematic nodes together to give matrix

\[
B_2 = \begin{bmatrix}
2\psi_1^{(4)} & 2\psi_1^{(5)} & 0 & 0 & \psi_2^{(4)} & \psi_2^{(5)} \\
2\psi_2^{(4)} & 2\psi_2^{(5)} & 0 & 0 & 0 & 0 \\
2\psi_3^{(4)} & 2\psi_3^{(5)} & \psi_2^{(4)} & \psi_2^{(5)} & 0 & 0 \\
0 & 0 & 2\psi_1^{(4)} & 2\psi_1^{(5)} & \psi_3^{(4)} & \psi_3^{(5)} \\
\psi_3^{(4)} & \psi_3^{(5)} & 2\psi_2^{(4)} & 2\psi_2^{(5)} & 0 & 0 \\
0 & 0 & \psi_3^{(4)} & \psi_3^{(5)} & 2\psi_3^{(4)} & 2\psi_3^{(5)} & 0 & 0 \\
\end{bmatrix}.
\] (3.25)

Let

\[
\Psi_2 = \begin{bmatrix} \psi_2^{(4)} & \psi_2^{(5)} \\ \psi_3^{(4)} & \psi_3^{(5)} \end{bmatrix};
\] (3.26)

\(\Psi_2\) 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

$$B_3 = \begin{bmatrix}
2\psi_1^{(4)} & 2\psi_1^{(5)} & 0 & 0 & 1 & 0 \\
2\psi_2^{(4)} & 2\psi_2^{(5)} & 0 & 0 & 0 & 0 \\
2\psi_3^{(4)} & 2\psi_3^{(5)} & \psi_2^{(4)} & \psi_2^{(5)} & 0 & 0 \\
0 & 0 & 2\psi_1^{(4)} & 2\psi_1^{(5)} & 0 & 1 \\
\psi_3^{(4)} & \psi_3^{(5)} & 2\psi_2^{(4)} & 2\psi_2^{(5)} & 0 & 0 \\
0 & 0 & 2\psi_3^{(4)} & 2\psi_3^{(5)} & 0 & 0 \\
\end{bmatrix}. \quad (3.27)$$

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

$$B_4 = \begin{bmatrix}
2\psi_2^{(4)} & 2\psi_2^{(5)} & 0 & 0 \\
2\psi_3^{(4)} & 2\psi_3^{(5)} & \psi_2^{(4)} & \psi_2^{(5)} \\
\psi_3^{(4)} & \psi_3^{(5)} & 2\psi_2^{(4)} & 2\psi_2^{(5)} \\
0 & 0 & 2\psi_3^{(4)} & 2\psi_3^{(5)} \\
\end{bmatrix}. \quad (3.28)$$

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

$$\mathbf{v}^{(4,1)} = \begin{bmatrix}
2\psi_1^{(4)} \\
2\psi_2^{(4)} \\
2\psi_3^{(4)} \\
\psi_2^{(4)} \\
0 \\
0 \\
\psi_3^{(4)} \\
0 \\
0 \\
\end{bmatrix}, \quad \mathbf{v}^{(5,1)} = \begin{bmatrix}
2\psi_1^{(5)} \\
2\psi_2^{(5)} \\
2\psi_3^{(5)} \\
\psi_2^{(5)} \\
0 \\
0 \\
\psi_3^{(5)} \\
0 \\
0 \\
\end{bmatrix}, \quad \mathbf{v}^{(6,1)} = \begin{bmatrix}
2\psi_1^{(6)} \\
2\psi_2^{(6)} \\
2\psi_3^{(6)} \\
\psi_2^{(6)} \\
0 \\
0 \\
\psi_3^{(6)} \\
0 \\
0 \\
\end{bmatrix}. \quad (3.29)$$

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
\[ v^{(2,1)} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad v^{(3,1)} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{3.30} \]

As a result of the interference alignment, \( v^{(2,1)} \) and \( v^{(3,1)} \) can subtract out all the interfering components from \( v^{(4,1)} \), \( v^{(5,1)} \) and \( v^{(6,1)} \). 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.

### 3.4.2 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
\[ k = \alpha. \tag{3.31} \]

This relation plays a key role in the code construction since each non-systematic node can 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 3.4.1, \( d = 2k - 1 = 5 \) 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 sources symbols in uncoded form. Hence, for systematic node $m$, and $l = 1, \ldots, k$,

$$G^{(m)}_l = \begin{cases} I_\alpha & \text{if } l = m \\ 0_\alpha & \text{if } l \neq m \end{cases} \quad (3.32)$$

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

$$\Psi = \begin{bmatrix} \psi^{(k+1)} & \psi^{(k+2)} & \cdots & \psi^{(n)} \end{bmatrix} \quad (3.33)$$

where

$$\psi^{(i)} = \begin{bmatrix} \psi_1^{(i)} \\ \vdots \\ \psi_\alpha^{(i)} \end{bmatrix} \quad i = k + 1, \ldots, n \quad (3.34)$$

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

$$q \geq \alpha + n - k. \quad (3.35)$$

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

$$g^{(m)}_{ij} = \begin{cases} \epsilon \psi^{(m)}_{ij} & \text{if } i = j \\ \psi^{(m)}_i \epsilon_j & \text{if } i \neq j \end{cases} \quad (3.36)$$

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 $q \geq 4$.

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 components from the coded symbols. The data collector now has to recover the symbols $z_{\Omega_1}, \ldots, z_{\Omega_p}$ which are encoded using the following $p\alpha \times p\alpha$ matrix

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

(3.37)

**Theorem 3.4.2** The data collector can decode the symbols $z_{\Omega_1}, \ldots, z_{\Omega_p}$ encoded using the matrix $R$.

**Proof** See Appendix B.1. The steps followed in the proof are similar to the ones used in the example. ■

**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.,

$$v^{(m,\hat{l})} = \begin{bmatrix} g_{1,\hat{l}}^{(m)} \\ \vdots \\ g_{n,\hat{l}}^{(m)} \end{bmatrix}.$$  

(3.38)

Observe that the choice of $g_{l,\hat{l}}^{(m)}$ as in equation (3.36) makes all the interfering components in $\{v^{(k+1,\hat{l})}, \ldots, v^{(n,\hat{l})}\}$ align along $e_{\hat{l}}$. Each of the remaining systematic nodes $l$ ($l = 1, \ldots, k$, $l \neq \hat{l}$) pass

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

(3.39)

with $e_l$ as the $l^{th}$ component.
Due to the interference alignment, $\mathbf{Y}_{l,\hat{l}}^{(i,j)}$ can subtract out all the interference along $\hat{z}_l$. Thus, all the interference is cancelled out.

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

$$\begin{bmatrix} g_{l,\hat{l}}^{(k+1)} & \cdots & g_{l,\hat{l}}^{(n)} \end{bmatrix}$$

By the choice of $g_{l,\hat{l}}^{(m)}$ as given in equation (3.36), following are the vectors along the desired component

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

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.

**Remark 3.4.3 (Exact regeneration of non-systematic nodes)** The authors in [15] show that our code can also perform exact regeneration of non-systematic nodes optimally.

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

The MISER code can be easily extended to work for the parameter regime $n \geq 2k, \ d = n - 1$ by applying Corollary 2.3.4 to the $n = 2k$ code constructed above.

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

$$\begin{bmatrix} G^{(1)} & G^{(2)} & \cdots & G^{(n)} \end{bmatrix}$$

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

$$\begin{bmatrix} G_1^{(1)} & \cdots & G_1^{(k)} & G_1^{(k+1)} & \cdots & G_1^{(n)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ G_k^{(1)} & \cdots & G_k^{(k)} & G_k^{(k+1)} & \cdots & G_k^{(n)} \end{bmatrix}$$

### 3.4.4 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 3.4.2 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 Corollary 2.3.4.
3.4 Construction of the MISER Code

**Theorem 3.4.4** For $d = 2k - 1$ and $d \leq n - 1$, the code defined by the node generator matrices in equations (3.32) and (3.36), 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 3.4.2. Hence exact regeneration of systematic nodes meeting the cut-set bound is guaranteed.

3.4.5 Analysis of the MISER 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 MISER 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. This is explained in more detail in Section

*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 $\Psi$ chosen to be a Cauchy matrix, any field of size $n + d - 2k + 1$ ($\approx 2n$) 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 perform inversion of at most $k$ Cauchy matrices, each of size at most $k \times k$.

*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 perform inversion of an $\alpha \times \alpha$ Cauchy matrix along with subtraction operations for interference cancellation.
3.5 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 $\beta = 1^2$. 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

$$\alpha < k - 2.$$  

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.

3.5.1 Necessary Properties

**Theorem 3.5.1** 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 $m \in \{k+1, \ldots, n\}$, $G_l^{(m)}$ 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 effect of these symbols from the remaining $\alpha$ symbols it has obtained. Now, the data collector is left with the symbols $z_l G_l^{(m)}$, 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 $G_l^{(m)}$ 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 3.5.2** 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.,

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

must be of full rank.

---

The remaining set of parameters, namely $2k - 3 \leq d < 2k - 1$, are considered in the Section 3.6.
3.5 Non-Existence for \( d < 2k - 3 \)

\textbf{Proof} Consider exact regeneration of systematic node \( l \). Let

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

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

(3.45)

where for \( i = 1, \ldots, k \),

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

is an \( \alpha \times d \) matrix representing the component of \( V \) along \( z_i \).

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

\[
VY = \mathbf{G}^{(l)}.
\]

(3.46)

Since \( \mathbf{G}^{(l)}_i = \mathbf{I}_\alpha \), we need

\[
\text{rank}(V_iY) = \alpha.
\]

(3.47)

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

\[
\mathbf{y}^{(n_i,i)} = 0 \quad i = 1, \ldots, k - 1
\]

(3.48)

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

\[
\begin{bmatrix}
  \mathbf{y}^{(m_1,i)} & \mathbf{y}^{(m_2,i)} & \cdots & \mathbf{y}^{(m_\alpha,i)}
\end{bmatrix}
\]

must be linearly independent to satisfy the rank condition in equation (3.47).

\textbf{Corollary 3.5.3} The last \( \alpha \) rows of \( Y \) should also be linearly independent.

\textbf{Proof} From equation (3.48), the first \( k - 1 \) columns of \( V_i \) are zeros. Hence to make \( V_iY \) full rank, we need the last \( d - (k - 1) = \alpha \) rows of \( Y \) need to be linearly independent.

\textbf{Theorem 3.5.4} (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

\[
\{\mathbf{y}^{(m_1,l)}_i, \ldots, \mathbf{y}^{(m_\alpha,l)}_i\}
\]

must be aligned.

\textbf{Proof} Consider the exact regeneration of systematic node \( l \). Using the same notation as in Theorem 3.5.2, since

\[
\mathbf{G}^{(l)}_i = 0_\alpha
\]

(3.49)
we need
\[ V_i Y = 0_\alpha. \] (3.50)

Also, since
\[ v_i^{(n_i)} = 0 \quad \forall n_i \neq \hat{l}, \] (3.51)
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 \( \tilde{V}_i \),
\[ \tilde{V}_i = \begin{bmatrix} v_i^{(\hat{l},\hat{l})} & v_i^{(m_1,\hat{l})} & \cdots & v_i^{(m_\alpha,\hat{l})} \end{bmatrix} \] (3.52)

Remove the corresponding \( k - 2 \) rows in \( Y \) and denote the resultant \( \alpha + 1 \times \alpha \) submatrix by \( \tilde{Y} \). Thus, we need
\[ \tilde{V}_i \tilde{Y} = 0_\alpha \] (3.53)

From Corollary 3.5.3
\[ \text{rank}(\tilde{Y}) = \alpha \] (3.54)

which forces
\[ \text{rank}(\tilde{V}_i) \leq 1 \] (3.55)

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

**Remark 3.5.5** 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.

### 3.5.2 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.

Recall the definition of equivalence of codes, as given in Section 3.1: codes \( \mathcal{C} \) and \( \mathcal{C}' \) are equivalent iff the code \( \mathcal{C}' \) can be represented in terms of \( \mathcal{C} \) 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 nodes. We will not distinguish between two equivalent codes, and consider them as the same code.

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.

Since the matrices \( G_i^{(m)} \) are non-singular (Theorem 3.5.1), they can be represented as
\[ G_i^{(m)} = H_i^{(m)} A_i^{(m)}, \quad m = k + 1, \ldots, n, \ i = 1, \ldots, k \] (3.56)
where $\Lambda_i^{(m)} = diag\{\lambda_{i,1}^{(m)}, \ldots, \lambda_{i,\alpha}^{(m)}\}$ is a non-singular $\alpha \times \alpha$ diagonal matrix,

$$H_i^{(m)} = \begin{bmatrix} h_{i,1}^{(m)} & h_{i,2}^{(m)} & \cdots & h_{i,\alpha}^{(m)} \end{bmatrix}$$ (3.57)

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

**Lemma 3.5.6** If there exists an exact regenerating code for $d < 2k - 1$, then there exists an equivalent code with the property

$$h_{i,j}^{(m)} = h_{i,j}, \text{ for } i = 1, \ldots, \alpha, \ j = 1, \ldots, k, \ j \neq i$$ (3.58)

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 necessary for regeneration of systematic nodes, we obtain the vectors that the non-systematic nodes pass for regeneration. Since these vectors lie in the column space of the respective non-systematic node’s generator matrices, these also give 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:

$$G^{(m)} = \begin{bmatrix} \lambda_{1,1}^{(m)} & \lambda_{1,2}^{(m)} & \cdots & \lambda_{1,p}^{(m)} & \lambda_{1,1}^{(m)} & \lambda_{1,2}^{(m)} & \cdots & \lambda_{1,\alpha}^{(m)} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \lambda_{i,1}^{(m)} & \lambda_{i,2}^{(m)} & \cdots & \lambda_{i,\alpha}^{(m)} & \lambda_{i,1}^{(m)} & \lambda_{i,2}^{(m)} & \cdots & \lambda_{i,\alpha}^{(m)} \\ \ldots & \lambda_{\alpha,1}^{(m)} & \lambda_{\alpha,2}^{(m)} & \cdots & \lambda_{\alpha,\alpha}^{(m)} & \lambda_{\alpha,1}^{(m)} & \lambda_{\alpha,2}^{(m)} & \cdots & \lambda_{\alpha,\alpha}^{(m)} \\ \lambda_{\alpha+1,1}^{(m)} & \lambda_{\alpha+1,2}^{(m)} & \cdots & \lambda_{\alpha+1,p}^{(m)} & \lambda_{\alpha+1,1}^{(m)} & \lambda_{\alpha+1,2}^{(m)} & \cdots & \lambda_{\alpha+1,\alpha}^{(m)} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \lambda_{k,1}^{(m)} & \lambda_{k,2}^{(m)} & \cdots & \lambda_{k,\alpha}^{(m)} & \lambda_{k,1}^{(m)} & \lambda_{k,2}^{(m)} & \cdots & \lambda_{k,\alpha}^{(m)} \end{bmatrix}$$ (3.59)

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 (Theorem 3.5.4),

$$h_{j,1}^{(m)} = h_{j,1}, \ j = 2, \ldots, k$$ (3.60)

Hence, the first column has to be of the form given by (3.59).

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

$p - 1$ 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 (Theorem 3.5.4), the first $p - 1$ columns of $G^{(m)}$ have the form as given above in equation (3.59).

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: $M_A = \text{Set of all non-systematic nodes which pass a vector linearly dependent on the first } p - 1 \text{ columns of their generator matrices.}$

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

Now consider, $v^{(m,p)}_{\alpha+1}$ for $m = k + 1, \ldots, n$, i.e., the component along $z_{\alpha+1}$. For all $m \in M_A$, $v^{(m,p)}_{\alpha+1}$ is a linear combination of $\{h_{\alpha+1,1}, \ldots, h_{\alpha+1,p-1}\}$. Whereas for all $m \in M_B$, $v^{(m,p)}_{\alpha+1}$ is linearly independent of this set of vectors. This is because, for $m \in M_B$, $v^{(m,p)}_{\alpha+1}$, along with $v^{(m,1)}_{\alpha+1}, \ldots, v^{(m,p-1)}_{\alpha+1}$ are columns of $H^{(m)}_{\alpha+1}$, and $H^{(m)}_{\alpha+1}$ has to be non-singular (Theorem 3.5.1).

Thus, $v^{(m_A,p)}_{\alpha+1}$ and $v^{(m_B,p)}_{\alpha+1}$, 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 3.5.4. 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 $m = k + 1, \ldots, n$, $v^{(m,p)}_{\alpha+1}$ is a linear combination of $\{h_{p,1}, \ldots, h_{p,p-1}\}$. Hence out of $\{v^{(k+1,p)}_{\alpha+1}, \ldots, v^{(n,p)}_{\alpha+1}\}$, at most $p - 1$ can be linearly independent. Since $p \leq \alpha$, this violates the necessary condition given by Theorem 3.5.2.

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 (3.59). 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 (3.59).

Henceforth in this section, we consider all non-systematic node generator matrices to be of the form by equation (3.59). Also note that when the generator matrices are of this form, for $l = 1, \ldots, \alpha$,

$$x^{(m,l)} = \varepsilon_l$$

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

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

**Corollary 3.5.7** For $l = \alpha + 1, \ldots, k$, and any non-systematic nodes $m$ and $m'$,

$$H_l^{(m)} = H_l^{(m')}$$

(3.62)
3.5 Non-Existence for \( d < 2k - 3 \)

**Proof** Directly follows from the structure of the generator matrices of non-systematic nodes in (3.59).

**Corollary 3.5.8** For any non-systematic node \( m \), and \( d < 2k - 1 \), any \( \alpha \) vectors out of \( \mathbf{v}^{(m,1)}, \ldots, \mathbf{v}^{(m,k)} \) are linearly independent.

**Proof** In Lemma 3.5.6, 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 3.5.6, the vectors passed by any non-systematic node for the regeneration of these \( \alpha \) systematic nodes will be the \( \alpha \) columns of its matrix, and hence will be independent (by Theorem 3.5.1).

**Remark 3.5.9** On similar lines it can be shown that for the case \( n \geq 2k \) and \( d = n - 1 \), the vectors passed for the regeneration of any \( k - 1 \) systematic nodes by a non-systematic node should be linearly independent. Moreover, at least one non-systematic node should pass linearly independent vectors for the regeneration of each 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 3.5.10** For \( d < 2k - 1 \), for any non-systematic node \( m \), and any systematic node \( l \in \{ \alpha + 1, \ldots, k \} \),

\[
x_i^{(m,l)} \neq 0 \quad i = 1, \ldots, \alpha
\]

**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,

\[
\mathbf{v}^{(m,l)} = \sum_{j=1, j \neq i}^{\alpha} x_j^{(m,l)} \mathbf{v}^{(m,j)}
\]

Hence \( \{ \mathbf{v}^{(m,j)} \}_{j=1, j \neq i}^{\alpha} \) and \( \mathbf{v}^{(m,l)} \) are linearly dependent which contradicts Corollary 3.5.8.

**3.5.3 The Non-Existence Proof**

**Theorem 3.5.11** Linear exact regenerating codes achieving the cut-set bound on repair bandwidth tradeoff curve do not exist for \( d < 2k - 3 \) with \( \beta = 1 \).
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 given in Theorem 3.5.4, components along symbols stored in systematic nodes $(\alpha + 1)$ and $(\alpha + 3)$ are to be aligned, i.e.,

$$
G_{\alpha+1}^{(k+1)}(k+1,\alpha+1) = \kappa_1 G_{\alpha+1}^{(k+2)} (k+2,\alpha+2) \\
G_{\alpha+3}^{(k+1)}(k+1,\alpha+2) = \kappa_2 G_{\alpha+3}^{(k+2)} (k+2,\alpha+2)
$$

(3.65)

(3.66)

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 3.5.1. Thus

$$
\kappa_1 \neq 0, \quad \kappa_2 \neq 0
$$

(3.67)

From Corollary 3.5.7,

$$
H_{\alpha+1}^{(k+1)}(k+1,\alpha+1) = \kappa_1 H_{\alpha+1}^{(k+2)} (k+2,\alpha+2) \\
H_{\alpha+3}^{(k+1)}(k+1,\alpha+2) = \kappa_2 H_{\alpha+3}^{(k+2)} (k+2,\alpha+2)
$$

(3.68)

(3.69)

Since $H_{\alpha+1}$, $H_{\alpha+3}$, $\Lambda_{\alpha+1}^{(k+1)}$ and $\Lambda_{\alpha+3}^{(k+1)}$ are non-singular (Theorem 3.5.1),

$$
\Lambda_{\alpha+1}^{(k+1)}(k+1,\alpha+1) = \kappa_1 \Lambda_{\alpha+1}^{(k+2)} (k+2,\alpha+2) \\
\Lambda_{\alpha+3}^{(k+1)}(k+1,\alpha+2) = \kappa_2 \Lambda_{\alpha+3}^{(k+2)} (k+2,\alpha+2)
$$

(3.70)

(3.71)

$$
\implies \kappa_1 (\Lambda_{\alpha+1}^{(k+1)})^{-1} \Lambda_{\alpha+1}^{(k+2)} (k+2,\alpha+2) = \kappa_2 (\Lambda_{\alpha+3}^{(k+1)})^{-1} \Lambda_{\alpha+3}^{(k+2)} (k+2,\alpha+2).
$$

(3.72)

From Lemma 3.5.10,

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

(3.73)

which gives

$$
\kappa_1 (\Lambda_{\alpha+1}^{(k+1)})^{-1} \Lambda_{\alpha+1}^{(k+2)} = \kappa_2 (\Lambda_{\alpha+3}^{(k+1)})^{-1} \Lambda_{\alpha+3}^{(k+2)}.
$$

(3.74)

Now, consider the exact regeneration of systematic node $\alpha + 3$. Analogous to equations (3.70) and (3.71), we get

$$
\Lambda_{\alpha+1}^{(k+1)} x_i^{(k+1,\alpha+3)} = \tilde{\kappa}_1 \Lambda_{\alpha+1}^{(k+2)} x_i^{(k+2,\alpha+3)}
$$

(3.75)

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

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

\[
\begin{align*}
\mathbf{v}_\alpha^{(k+1,\alpha+3)} &= G_{\alpha+3}^{(k+1)} \mathbf{v}_\alpha^{(k+1,\alpha+3)} \\
&= H_{\alpha+3} \Lambda_{\alpha+3}^{(k+1)} \mathbf{v}_\alpha^{(k+1,\alpha+3)} \\
&= \kappa_1 H_{\alpha+3} \Lambda_{\alpha+3}^{(k+1)} (\Lambda_{\alpha+1}^{(k+1)} - 1) \Lambda_{\alpha+1}^{(k+2)} \mathbf{v}_\alpha^{(k+2,\alpha+3)}
\end{align*}
\]

(3.76)

where equation (3.78) is obtained by substituting for $\mathbf{v}_\alpha^{(k+1,\alpha+3)}$ from (3.75). Similarly,

\[
\begin{align*}
\mathbf{v}_\alpha^{(k+2,\alpha+3)} &= H_{\alpha+3} \Lambda_{\alpha+3}^{(k+2)} \mathbf{v}_\alpha^{(k+2,\alpha+3)} \\
&= \kappa_1 \kappa_2^{-1} H_{\alpha+3} \Lambda_{\alpha+3}^{(k+1)} (\Lambda_{\alpha+1}^{(k+1)} - 1) \Lambda_{\alpha+1}^{(k+2)} \mathbf{v}_\alpha^{(k+2,\alpha+3)}
\end{align*}
\]

(3.79)

where equation (3.79) is obtained by substituting for $\Lambda_{\alpha+3}^{(k+2)}$ from (3.74). From equations (3.78) and (3.79),

\[
\mathbf{v}_\alpha^{(k+2,\alpha+3)} = \kappa_1 \kappa_2^{-1} \mathbf{v}_\alpha^{(k+1,\alpha+3)}
\]

(3.80)

i.e., $\mathbf{v}_\alpha^{(k+1,\alpha+3)}$ and $\mathbf{v}_\alpha^{(k+2,\alpha+3)}$ are linearly dependent. This violates Theorem 3.5.2, and hence exact regeneration of node $\alpha + 3$ is not possible. Thus, in the process of adhering to the interference alignment requirements for exact regeneration of nodes $\alpha + 2$ and $\alpha + 3$, the linear independence property essential for the exact regeneration of node $\alpha + 3$ is lost. This gives the contradiction. 

### 3.6 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 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 (1.8), 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.

#### 3.6.1 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, why comply with this structure. The remaining entries in the node
generator matrices and the linear combination vectors will be kept as variables, and the
reconstruction and regeneration conditions will be cast as product of rational polynomials
in these variables. We show that there exists an assignment to the variables such that
these polynomials are all well defined and non-zero. In [26], 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

\[
\begin{align*}
    f_{p+1}(x) &= \gcd(g_1(x), \ldots, g_p(x)) .
\end{align*}
\]

(3.81)

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 [26] can be used to find the values of the variables, provided the field size
is large enough.

### 3.6.2 Existence and Construction

**Lemma 3.6.1** A necessary and sufficient condition for exact regeneration of system-
atic node \( l \) by connecting to the existing \( k-1 \) systematic nodes \( n_1, \ldots, n_{k-1} \) and \( \alpha \)
non-systematic nodes \( m_1, m_2, \ldots, m_\alpha \), is that the set of vectors passed by these non-
systematic nodes,

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

(3.82)
satisfy Theorems 3.5.2 and 3.5.4.

**Proof** **Necessity:** Shown in Theorems 3.5.2 and 3.5.4. **Sufficiency:** Suppose Theorem
3.5.4 is satisfied. Then, in the vectors passed by the \( \alpha \) non-systematic nodes, the compo-
nents along other systematic nodes are all aligned along one vector, i.e., for any existing
systematic node \( \hat{l} \)

\[
\begin{align*}
    v^{(m_i,l)}_{\hat{l}} &= \kappa^{(m_i,l)}_{\hat{l}} w^{(l)}_{\hat{l}} \
    \hat{l} &= 1, \ldots, \alpha
\end{align*}
\]

(3.83)

where \( w^{(l)}_{\hat{l}} \) is a vector independent of \( m_i \), and \( \kappa^{'}s \) are some constants in \( \mathbb{F}_q \). Let the vector
passed by the systematic node \( \hat{l} \) be such that

\[
\begin{align*}
    v^{(l,l)}_{\hat{l}} &= w^{(l)}_{\hat{l}} .
\end{align*}
\]

(3.84)

Thus, \( v^{(l,l)} \) 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,
\[ \tilde{v}^{(m,l)} = v^{(m,l)} - \sum_{i=1, i \neq l}^{k} K_i^{(m,l)} v^{(l,l)}. \] (3.85)

The set of vectors \(\{\tilde{v}^{(m_1,l)}, \ldots, \tilde{v}^{(m_\alpha,l)}\}\) have no interfering components. Since the vectors \(v^{(i,l)}, i = 1, \ldots, k, i \neq l\) have zero components along \(z_l\), we have
\[ \tilde{v}_i^{(m,l)} = v_i^{(m,l)}. \] (3.86)

Now, since Theorem 3.5.2 is satisfied, the vectors
\[ \{\tilde{v}_1^{(m_1,l)}, \ldots, \tilde{v}_{\alpha}^{(m_\alpha,l)}\} \]
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 (3.59), i.e.,
\[ G_i^{(m)} = H_i^{(m)} \Lambda_i^{(m)} \] (3.87)

for \(m = k + 1, \ldots, n\), \(i = 1, \ldots, k\), where \(\Lambda_i^{(m)} = \text{diag}\{\lambda_{i,1}^{(m)}, \ldots, \lambda_{i,\alpha}^{(m)}\}\) is a non-singular \(\alpha \times \alpha\) diagonal matrix,
\[ H_i^{(m)} = \begin{bmatrix} h_{i,1}^{(m)} & h_{i,2}^{(m)} & \cdots & h_{i,\alpha}^{(m)} \end{bmatrix} \] (3.88)

is a non-singular \(\alpha \times \alpha\) matrix, and \(h_{i,j}^{(m)}\) 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
\[ \tilde{z}^{(m,l)} = \xi_l, \quad l = 1, \ldots, \alpha. \] (3.89)

Now, to satisfy Theorem 3.5.4,
\[ h_{i,j}^{(m)} = h_{i,j}, \quad m = k + 1, \ldots, n \]
\[ i = 1, \ldots, \alpha, \]
\[ j = 1, \ldots, k, \quad j \neq i \] (3.90)

For \(m = k + 1, \ldots, n\) we set
\[ \lambda_{i,i}^{(m)} = 1, \quad i = 1, \ldots, \alpha. \] (3.91)
Next we provide a constructive proof for the existence of exact regenerating codes performing optimal exact regeneration of the systematic nodes for the parameter regime \( d = 2k - 3 \), and then invoke Corollary 2.3.4 to extend it to the case \( d \geq 2k - 3 \).

**Theorem 3.6.2** 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 B.2. \( \Box \)

Thus, as long as \( d \geq 2k - 3 \), there is no penalty for introducing exact regeneration of systematic nodes, over the scenario of functional regeneration.

### 3.7 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.

#### 3.7.1 Scheme Description

Divide the \( k \) systematic nodes into \( \alpha \) groups. Analogous to the scheme given by Wu et al. [9], 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

\[
\hat{b}_{i}^{(m)} = [b_{i,1}^{(m)} \cdots b_{i,\mu(i)-1}^{(m)} 0 b_{i,\mu(i)+1}^{(m)} \cdots b_{i,\alpha}^{(m)}]^{t} \quad (3.92)
\]

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

\[
\tilde{b}_{i}^{(m)} = [b_{i,1}^{(m)} \cdots b_{i,\mu(i)-1}^{(m)} a_{i}^{(m)} b_{i,\mu(i)+1}^{(m)} \cdots b_{i,\alpha}^{(m)}]^{t} \quad (3.93)
\]
Let the node matrix of non-systematic node \( m \in \{ k+1, \ldots, n \} \) be

\[
G_i^{(m)} = a_i^{(m)} I_\alpha + B_i^{(m)}
\]

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

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

\[
(3.95)
\]

**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 \( \begin{bmatrix} 0 \cdots 0 & \xi_{\mu(l')} & 0 \cdots 0 \end{bmatrix}^T \) 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 \( a_i^{(m)} \xi_{\mu(l)} \), 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 cancelled 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.,

\[
det \left( \tilde{b}_1^{(m_1)} \ldots \tilde{b}_\alpha^{(m_\alpha)} \right)
\]

\[
(3.96)
\]
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 3.4.2, reconstruction condition leads to the following condition of a polynomial being non-zero

$$
\det \begin{pmatrix} 
G_{t_1}^{(m_1)} & \cdots & G_{t_1}^{(m_p)} \\
\vdots & \ddots & \vdots \\
G_{t_p}^{(m_1)} & \cdots & G_{t_p}^{(m_p)}
\end{pmatrix}. 
$$

(3.97)

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

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

(3.98)

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

(3.99)

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 [26].

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.

### 3.7.2 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:

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

(3.100)
3.8 Towards Uniqueness of Code Construction

In Section 3.4, 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 3.5.1). 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 for true for the explicit constructions provided in Section 3.4.

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

\[
D_{i}^{(m)} = \begin{bmatrix}
    d_{i,1}^{(m)} & 0 & \cdots & 0 \\
    0 & d_{i,2}^{(m)} & \cdots & 0 \\
    \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & \cdots & d_{i,k}^{(m)} \\
\end{bmatrix}
\]

(3.102)

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

\[
F_{i} = \begin{bmatrix}
    f_{i}^{(k+1)} & f_{i}^{(k+2)} & \cdots & f_{i}^{(2k)} \\
\end{bmatrix}
\]

(3.103)

where \( f_{i}^{(m)}, \ m = k + 1, \ldots, 2k, \) are \( k \)-length column vectors.

Let \( \Delta_{j} \) be a \( k-1 \times k \) matrix with entries

\[
\begin{bmatrix}
    d_{1,j}^{(k+1)} & d_{1,j}^{(k+2)} & \cdots & d_{1,j}^{(m)} \\
    \vdots & \vdots & \ddots & \vdots \\
    d_{j-1,j}^{(k+1)} & d_{j-1,j}^{(k+2)} & \cdots & d_{j-1,j}^{(m)} \\
    d_{j+1,j}^{(k+1)} & d_{j+1,j}^{(k+2)} & \cdots & d_{j+1,j}^{(m)} \\
    \vdots & \vdots & \ddots & \vdots \\
    d_{k,j}^{(k+1)} & d_{k,j}^{(k+2)} & \cdots & d_{k,j}^{(m)} \\
\end{bmatrix}
\]

(3.104)

**Theorem 3.8.1** 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 \( k \) systematic nodes, there exists an equivalent code with the non-systematic node generator matrices as

\[
G_{i}^{(m)} = F_{i}^{(m)} e_{i} + D_{i}^{(m)}
\]

(3.105)

and the following conditions hold:

1) \( F_{i} \) is invertible,
2) Any sub-matrix of $\Delta_j$ is full rank.

**Proof** See Appendix B.3. ■

Note that the structure presented in Theorem 3.8.1 automatically applies to codes for $n < 2k, d = n - 1$, since Corollary 3.5.8 mandates the vectors passed by a non-systematic node for regeneration of any $\alpha$ nodes to be linearly independent.

**Assigning Explicit Values to Construct Codes**

One means of constructing codes for this parameter set is as follows. For every given value of $i$, 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 $\epsilon_i \Psi(i)$ for a $k \times k$ invertible diagonal matrix $\epsilon_i$. This choice of $F_i$ involves $\epsilon_i$ to make reconstruction feasible, and $\Psi(i)$ for ease of decoding. With this choice, the reconstruction and regeneration properties are satisfied if $\epsilon_{i_1,j_1} \neq \epsilon_{i_2,j_2}^{-1}$ for $i_1, j_1, i_2, j_2 = 1, \ldots, k$, $i_1 \neq i_2$, $j_1 \neq j_2$, $i_1 \neq i_2$, where $\epsilon_{i,j}$ is the $(j,j)$th element of $\epsilon_i$.

The explicit code presented in Section 3.4 is of this form, with each $\Psi(i)$ as $\Psi$, a Cauchy matrix, and each $\epsilon_i$ as $\epsilon I$, where $\epsilon$ is a non-zero element of the field such that $\epsilon^2 \neq 1$.

**Code Puncturing**

If each $F_i$ also has the property of having any sub-matrix full rank, then given a code for the parameter set $[n = 2k, d = 2k - 1, k]$, this code can be punctured to obtain codes for the parameter set $[n = 2k', d = 2k' - 1, k]$ for any $k' < k$. The code for $k' = k - 1$ can be obtained by removing any one non-systematic node, removing the $i^{th}$ systematic node (for any $i \in \{1, \ldots, k\}$) and deleting the $i^{th}$ columns of the generator matrices of each of the remaining nodes. The new code so obtained is optimal for $k' = k - 1$ and also has the property that its $F_i$ matrices have any sub-matrix full rank.
Chapter 4

Non-achievability of Interior Points on the Storage-Repair Bandwidth Tradeoff

The tradeoff between storage and repair bandwidth described in Section 1.1.2 was originally derived for the setting where a replacement node, after regeneration, is only constrained to perform reconstruction of data and regeneration of other failed nodes. It was also shown to be achievable for this setting.

Subsequently a more practical setting of exact regeneration was introduced, where a replacement node should obtain and store exactly the same data that was stored in the failed node. The tradeoff derived for functional regeneration is trivially a lower bound for exact regeneration as well. However, apart from a few parameter sets at the extreme points, it is not known whether the tradeoff is achievable for exact regeneration.

In this chapter, we answer this question, as to whether the interior points on the storage-repair bandwidth tradeoff are achievable under exact regeneration, in the negative (with the possible exception of the region of width at-most $\beta$ in the immediate vicinity of the MSR point). Along the way we also establish properties that any exact-regenerating code must possess.

The organization of this chapter is as follows. We first provide a representation for any point on the tradeoff in Section 4.1, which will be used throughout this chapter. Using the subspace viewpoint described earlier, we obtain a set of properties that any linear exact-regenerating code must satisfy, in Section 4.2, which are then exploited in Section 4.3 to establish the non-achievability of the storage-repair bandwidth tradeoff for exact regeneration with linear codes for nearly all interior points. Also in this section, repair bandwidths achievable for all values of system parameters are obtained via storage space sharing between the MSR and MBR codes presented in Chapter 2.
4.1 A Representation for the Points on the Tradeoff

For the reasons analyzed in the derivation of equation 1.5, the range of $\alpha$ for the tradeoff is

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

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

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

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

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

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: $\{1 \leq p \leq k - 2, \text{ any } \theta\} \cup \{p = 0, \theta > 0\}$.

**Additional notation:** As in Section 3.1, the subspace stored in node $i$ is denoted by $W_i$, and can have a dimension at-most $\alpha$. Further, 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 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 $l$.

The reconstruction and regeneration requirements can be re-stated in this subspace terminology as below. For reconstruction, for every subset of $k$ storage nodes: $\{i_j | 1 \leq j \leq k\}$, we need

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

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

$$W_{l} \subseteq \sum_{j \in D} S_{D}^{(j,l)}.$$  

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 $j$,

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

4.2 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 (1.2). 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 C.

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

**Lemma 4.2.1** For any subspaces \( V_1, V_2 \) and \( V_3 \) satisfying \( V_1 \subseteq V_2 + V_3 \),

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

(4.6)

**Proof**

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

(4.7)

\[
= |V_3| + |V_1 + V_2| - |V_3 + V_2|
\]  

(4.8)

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

(4.9)

**Lemma 4.2.2** For any subspaces \( U_1, U_2 \) and \( U_3 \),

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

(4.10)

**Proof**

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

(4.11)

\[
\geq |U_1 + U_3| - |U_3|
\]  

(4.12)

\[
= |U_1| - |U_1 \cap U_3|.
\]  

(4.13)

Next, we establish properties relating to the nodal subspaces in \( C \). Recall that \( \alpha = (d-p)\beta - \theta \), where \( \theta = (-\alpha) \mod \beta \), and the interior points correspond to the parameters \( \{1 \leq p \leq k-2, \text{ any } \theta\} \cup \{p = 0, \theta > 0\} \).

**Property 4.2.3 (Dimension of Nodal Subspaces)** For any storage node \( i \),

\[
|W_i| = \alpha, \quad \forall i \in \{1, \ldots, n\}.
\]
This condition thus mandates each of the nodes to utilize their entire storage space.

**Property 4.2.4 (Intersection Property of Nodal Subspaces)**
For any set of nodes \( A \),

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

(4.14)

where node \( l \notin A \) and \( a \) is the size of set \( A \).

We now move on to the properties governing the subspaces passed in the process of regeneration.

**Property 4.2.5 (Dimension of Passed Subspaces)**
For exact regeneration of any node \( l \), and any assisting node \( m \),

\[
| S^{(m,l)} | = \beta.
\]  

(4.15)

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

**Property 4.2.6 (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 \( m \notin A \),

\[
\left| \sum_{l \in A} S^{(m,l)} \right| \leq 2\beta - \theta,
\]  

(4.16)

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 4.2.7 (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 \( m \notin A \),

\[
\left| \bigcap_{l \in A} S^{(m,l)} \right| \geq \beta - \theta
\]  

(4.17)

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

$$\left| \sum_{l \in A} S^{(m,l)} \right| \leq \beta + (a - 1)\theta \quad (4.18)$$

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 $\{m\} \cup A\backslash\{l\}$.

4.3 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. This case automatically includes the atomic, and arguably the most important case of $\beta = 1$, whose significance is underscored by the fact that a majority of the codes and schemes available in the literature [9–12, 15, 17, 21, 24] have been constructed for $\beta = 1$. The case when $\alpha$ is not a multiple of $\beta$ is considered subsequently in Theorem 4.3.2.

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 4.3.1 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 WOLOG a system comprising of only these storage nodes with the system parameter $n$ equal to $d + 1$. 
4.3 Non-achievability of the Interior Points On the Tradeoff for Exact Regeneration

Property 4.2.7, along with \( \theta = 0 \), mandates that for any set \( A \) comprising of \( p + 1 (\geq 2) \) nodes, and node \( m \notin A \),

\[
\left| \bigcap_{l \in A} S^{(m,l)} \right| \geq \beta . \tag{4.19}
\]

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, \( \sum_{\text{all } j} S^{(j)} \) spans the nodal subspaces of all the nodes in the system, and hence for reconstruction to hold, we need

\[
(d + 1)\beta \geq \left| \sum_{\text{all } j} S^{(j)} \right| \geq B. \tag{4.20}
\]

On the other hand, from the cut-set bound in (1.2), we have the relation

\[
B = \sum_{i=0}^{k-1} \min ((d - p)\beta, (d - i)\beta) \tag{4.21}
\]

\[
\geq 2(d - p)\beta + \sum_{i=2}^{k-1} \min ((d - p)\beta, (d - i)\beta) \tag{4.22}
\]

\[
\geq 2(d - p)\beta + (k - 2)\beta \tag{4.23}
\]

\[
> (d + 1)\beta \tag{4.24}
\]

where equation (4.21) uses the fact that \( \alpha = (d - p)\beta \), equation (4.22) holds since \( p \geq 1 \), and equations (4.23) and (4.24) are derived using \( d \geq k \geq p + 2 \). This is in contradiction to equation (4.20).

**Theorem 4.3.2** 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 \( p = k - 2 \) 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 say \( x \), \( y \), 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
4.3 Non-achievability of the Interior Points On the Tradeoff for Exact Regeneration

\{y\} \cup T_1 \cup T_2 and regeneration of node \(y\) by connecting to \(\{x\} \cup T_1 \cup T_2\). We need

\[
W_x \subseteq \sum_{i \in T_1} S^{(i,x)} + \sum_{i \in T_2} S^{(i,x)} + S^{(y,x)}, \quad (4.25)
\]

\[
W_y \subseteq \sum_{i \in T_1} S^{(i,y)} + \sum_{i \in T_2} S^{(i,y)} + S^{(x,y)}. \quad (4.26)
\]

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

\[
W_y \subseteq \sum_{i \in T_1} W_i + \sum_{i \in T_2} S^{(i,y)} + W_x. \quad (4.27)
\]

Substituting \(W_x\) from equation (4.25) in equation (4.27), we get

\[
W_x + W_y \subseteq \sum_{i \in T_1} W_i + \sum_{i \in T_2} \left( S^{(i,x)} + S^{(i,y)} \right) + S^{(y,x)}. \quad (4.28)
\]

We further define three vector spaces: \(V_1 = W_x + W_y\), \(V_2 = \sum_{i \in T_1} W_i\) and \(V_3 = \sum_{i \in T_2} \left( S^{(i,x)} + S^{(i,y)} \right) + S^{(y,x)}\). Noting that \(V_1 \subseteq V_2 + V_3\), we apply Lemma 4.2.1 to get

\[
|V_3| \geq |V_1| - |V_1 \cap V_2|. \quad (4.29)
\]

Now,

\[
|V_3| = \left| \sum_{i \in T_2} \left( S^{(i,x)} + S^{(i,y)} \right) + S^{(y,x)} \right| \quad (4.30)
\]

\[
\leq \sum_{i \in T_2} \left( |S^{(i,x)} + S^{(i,y)}| + |S^{(y,x)}| \right) \quad (4.31)
\]

\[
\leq \sum_{i \in T_2} (2\beta - \theta) + \beta \quad (4.32)
\]

\[
= (2d - 2p - 1)\beta - (d - p - 1)\theta. \quad (4.33)
\]

where equation (4.32) follows from Property 4.2.6.
4.3 Non-achievability of the Interior Points On the Tradeoff for Exact Regeneration

On the other hand,

\[ |V_1| - |V_1 \cap V_2| = |W_x + W_y| - (W_x + W_y) \cap \left( \sum_{i \in T_1} W_i \right) \]  
(4.34)

\[ = \left| \sum_{i \in T_1 \cup \{x, y\}} W_i \right| - \left| \sum_{i \in T_1} W_i \right| \]  
(4.35)

\[ = ((p + 2)\alpha - (\beta - \theta)) - (p\alpha) \]  
(4.36)

\[ = 2\alpha - (\beta - \theta) \]  
(4.37)

\[ = (2d - 2p - 1)\beta - \theta, \]  
(4.38)

where equation (4.36) follows from Properties 4.2.3 and 4.2.4.

Now, since \( \theta \neq 0 \) and \( d \geq k > p + 2 \), equations (4.33) and (4.38) imply

\[ |V_3| < |V_1| - |V_1 \cap V_2|, \]  
(4.39)

which is in contradiction to equation (4.29).

We now proceed to the case of \( k = p + 2, \ p \neq 0 \). The proof for this case follows equations (4.25) through (4.31) from the previous case. However, since Property 4.2.6 need not hold for this case, the proof will diverge from here on.

Apply Corollary 4.2.8 (with \( a = 2 \)) to equation (4.31), to get

\[ |S^{(i,x)} + S^{(i,y)}| \leq \beta + \theta, \]  
(4.40)

and hence,

\[ |V_3| \leq (d - p)\beta + (d - p - 1)\theta. \]  
(4.41)

Now, equations (4.34) through (4.38) are also valid for this case. Comparing the value of \( |V_1| - |V_1 \cap V_2| \) obtained in (4.38) with that of \( V_3 \) above, we can infer that equation (4.29) is satisfied only if \( \theta \geq \frac{d - p - 1}{d - p} \beta \). Hence the cutset bound is not achievable for the parameter \( k = p + 2, \ p \neq 0 \) when

\[ \theta < \frac{d - p - 1}{d - p} \beta. \]  
(4.42)

Remark 4.3.3 The properties derived in Section 4.2, 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.
4.3 Non-achievability of the Interior Points On the Tradeoff for Exact Regeneration

4.3.1 An Achievable Curve via Storage Space Sharing

In Section 2, we constructed explicit codes for the MSR and MBR points that achieve the cut-set bound via the Product-Matrix framework. The MBR code works for all parameters \([n, k, d]\), whereas the MSR code works for the parameter set \([n, k, d \geq 2k-2]\). Furthermore, both these codes are striped to the atomic case of \(\beta = 1\). In this section, codes operating at all values of parameters \((\alpha, \beta)\) are constructed via storage-space-sharing between the two codes product-matrix codes.

The MBR and MSR points are characterized by the following relations between the parameters:

\[
B_{\text{MSR}} = k\alpha_{\text{MSR}}, \quad \alpha_{\text{MSR}} = (d - k + 1)\beta_{\text{MSR}}, \quad (4.43)
\]

\[
B_{\text{MBR}} = \left(d - \frac{(k - 1)}{2}\right) \frac{k\alpha_{\text{MBR}}}{d}, \quad \alpha_{\text{MBR}} = d\beta_{\text{MBR}}. \quad (4.44)
\]

The following analysis assumes \(\alpha\) and \(\beta\) to be system parameters. Furthermore, as required by the PM-MSR code, we will design codes for the case \(d \geq 2k - 2\). All data parameters – the source file, the storage spaces in each of the nodes, and the repair bandwidths available – are divided into two segments in the ratio \(f : 1-f\). The PM-MSR code operates on the first segment, and the PM-MBR code operates on the second.

In particular, the storage space in any node is partitioned into two: (i) a fraction \(f\alpha\) that is used to store symbols encoded by the PM-MSR code, and (ii) the remaining fraction \((1-f)\alpha\) occupied by another set of symbols encoded by the PM-MSR code.

The PM-MSR and the PM-MBR codes meet the cut-set bound, hence the repair bandwidths are given by

\[
\beta_{\text{MSR}} = \frac{f\alpha}{d-k+1}, \quad \beta_{\text{MBR}} = \frac{(1-f)\alpha}{d}, \quad (4.45)
\]

and the net \(\beta\) is the sum

\[
\beta = \beta_{\text{MSR}} + \beta_{\text{MBR}}. \quad (4.46)
\]

Solving for \(f\) in terms of the system parameters \(\alpha\) and \(\beta\), we get

\[
f = \frac{(d - k + 1)(d\beta - \alpha)}{(k - 1)\alpha}. \quad (4.47)
\]
4.3 Non-achievability of the Interior Points On the Tradeoff for Exact Regeneration

Figure 4.1: An achievable value of repair bandwidth \( d\beta \) for exact regeneration of all nodes plotted alongside the storage repair-bandwidth tradeoff curve, which is a lower bound on the repair bandwidth. A major portion of the tradeoff has been shown to be non-achievable; however, there is an uncertain region, as indicated in the plot.

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

\[
B = B_{\text{MSR}} + B_{\text{MBR}}
\]

\[
= fk\alpha + (1 - f) \left( d - \frac{(k - 1)}{2} \right) \frac{k\alpha}{d}
\]

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

Note that the product-matrix constructions require the values of \( \alpha, \beta \) and \( B \) to be integers. Hence, the value of \( f \) may be slightly higher than that computed in equation (4.47), to ensure 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. However, in practical systems, the data parameters \( B \) and \( \alpha \) will be high and this fringe effect will become negligible.

This value of achievable \( B \) 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 4.1. Also plotted is the uncertain region with respect to achievability of the interior points on the tradeoff curve.
Chapter 5

An Explicit MBR Code for $d = n - 1$, and its Graphical Description

This chapter describes the construction of an explicit exact-regenerating code for the Minimum Bandwidth Regenerating (MBR) point on the tradeoff, corresponding to the minimum possible repair bandwidth, for the parameters $(n, k, d = n - 1)$. This code has a particularly simple graphical description which makes it simple to implement. Most interestingly, when specialized to the case $[n, k = n - 2, d = n - 1]$, simple XOR operations at the nodes suffice to implement reconstruction and regeneration.

Again, we will construct codes for the atomic case $\beta = 1$, and concatenate this code to obtain optimal codes for any higher value of $\beta$. The equations governing the parameters for the MBR point, i.e., equations 1.10 and 1.11 are reproduced below:

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

We will first furnish an example that will illustrate all the key ideas of the code construction. The general code construction will be provided in the Section 5.2. An analysis of the code is provided in Section 5.3, where we also describe the property of this code to be implemented solely via XOR operations for a subset of the parameters.
5.1 An Example Code

For the example, 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

$$\begin{bmatrix} v_{1,2} & v_{1,3} & v_{1,4} & v_{1,5} \\ v_{2,3} & v_{2,4} & v_{2,5} \\ v_{3,4} & v_{3,5} \\ v_{4,5} \end{bmatrix},$$

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

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

- **Node 1:** $\{u^t v_{1,2}, u^t v_{1,3}, u^t v_{1,4}, u^t v_{1,5}\}$
- **Node 2:** $\{u^t v_{2,1}, u^t v_{2,3}, u^t v_{2,4}, u^t v_{2,5}\}$
- **Node 3:** $\{u^t v_{3,1}, u^t v_{3,2}, u^t v_{3,4}, u^t v_{3,5}\}$
- **Node 4:** $\{u^t v_{4,1}, u^t v_{4,2}, u^t v_{4,3}, u^t v_{4,5}\}$
- **Node 5:** $\{u^t v_{5,1}, u^t v_{5,2}, u^t v_{5,3}, u^t v_{5,4}\}$

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; $u$ is a 9-length vector with its elements as the 9 source symbols. A node stores the 4 symbols associated to the 4 edges incident on it.
5.2 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 \( \{v_{i,j}\} \) \( (1 \leq i < j \leq n) \), each of length \( B \), such that any \( B \) of them are linearly independent; and a second set of \( \binom{n}{2} \) vectors \( \{v_{j,i}\} \) \( (1 \leq i < j \leq n) \), such that \( v_{j,i} = v_{i,j} \). Next, represent the \( n \) storage nodes as the vertices of an undirected

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 to the 9 symbols

\[
\{ u^t v_{3,1}, u^t v_{3,2}, u^t v_{3,4}, u^t v_{3,5} \}
\]

\[
\{ u^t v_{4,1}, u^t v_{4,2}, u^t v_{4,3}, u^t v_{4,5} \}
\]

\[
\{ u^t v_{5,1}, u^t v_{5,2}, u^t v_{5,3}, u^t v_{5,4} \}
\]

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 \( u^t v_{1,3}, u^t v_{2,3}, u^t v_{4,3} \) and \( u^t v_{5,3} \) 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

\[
\binom{n}{2} = B + 1.
\]

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 \( \mathbb{F}_2 \).
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 \( u^t v_{i,j} \) (\( = u^t v_{j,i} \)).

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 5.2.1 (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:

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

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 5.2.2 (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.
5.3 Size of the Finite Field and XOR based constructions

The sole constraint on the field size required for the code construction is that it should enable construction of a \([\binom{n}{2}, B] \) 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 \([n, k = n - 2, d = n - 1]\), where \(\binom{n}{2} = B + 1\). For this setting, the \([\binom{n}{2}, B] \) MDS code can be chosen to be a single parity check code. It follows that a finite field of any arbitrary size can be employed for this set of parameters; choosing the binary field facilitates every operation to be carried out via XORs, making the code easy to implement.
Chapter 6

Approximately Exact Regenerating Codes

We present an explicit code which reduces the repair bandwidth for all the nodes to approximately $B/2$. To the best of our knowledge, this is the first explicit code construction in the literature which reduces the repair bandwidth of all the nodes for any value of the system parameters $B$, $\alpha$, $n$, and $k$.\footnote{We assume the trivial condition of $\alpha \geq \frac{B}{k}$ to be satisfied by these parameters.} The codes provided are approximately exact, i.e., only half of the symbols stored in the new node are identical to those in the failed node. Also note that unlike the codes previously presented, this code does not assume $\beta = 1$, since the chosen $B$ and $\alpha$ need not scale linearly with $\beta$.

An important special case of this code is for the MSR point for the parameters $d = k + 1$, with an arbitrary $n$ and $\beta = 1$, provided in Section 6.2. In light of the non-existence result described in Section 3.5 for $d < 2k - 3$, for this value of $d$, exact regeneration is not possible in general. Thus, the construction does the next best thing namely, it carries out regeneration that is approximately exact.

The chapter is organized as follows. Section 6.1 provides a code construction for the trivial case of $\alpha = 1$. Construction for $\alpha = 2$ is provided in Section 6.2, that includes the case of MSR $d = k + 1$. Using the constructions for $\alpha = 1$ and $\alpha = 2$ as building blocks, a construction for any value of $\alpha$ is given in Section 6.3. Finally, in Section 6.4, an analysis of the code is performed.

6.1 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.
6.1 Construction for the Case $\alpha = 1$

Let $f$ be a vector with $B$ source symbols as its elements. Let $l$ denote the length of this vector. Hence $l = B$, and $B \leq k\alpha$, we have

$$l \leq k$$

(6.1)

Let $p^{(i)} (i = 1, \ldots, n)$ be vectors of length $l$ forming a $l$-dimensional MDS code over $\mathbb{F}_q$. In the sequel, we will use superscripts to denote the node number corresponding to any symbol.

**Code:** Node $i$ stores $f^t p^{(i)}$, $i = 1, \ldots, n$.

### 6.1.1 Reconstruction

**Lemma 6.1.1** 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.

### 6.1.2 Regeneration

For regeneration of a failed node, we need $d \geq l$.

**Lemma 6.1.2** 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.

### 6.1.3 Repair Bandwidth

The repair bandwidth for any node is

$$\gamma_1(l) = l = B$$

(6.2)

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 Construction for the Case $\alpha = 2$

Partition the source symbols into two sets $S$ and $S'$ having sizes $l$ and $l'$ respectively such that
\[
0 \leq l, l' \leq k \quad (6.3)
\]
and
\[
(l + l') = B \quad (6.4)
\]

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 $(f^tp^{(i)}, g^tr^{(i)} + f^tu^{(i)})$ 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 $p^{(i)} = (1 \theta_i \theta_i^2)^t$ and $r^{(i)} = (1 \theta_i)$. $\theta_i$ ($i = 1, \ldots, n$) take distinct values from $\mathbb{F}_q(q \geq n)$. Elements of $u^{(i)}$ can be initialized to arbitrary values from $\mathbb{F}_q$.

6.2.1 Reconstruction:

**Lemma 6.2.1** 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 $f^tp^{(i)}$ 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, $f^tu^{(i)}$ can be subtracted out from the second symbols of some $l'$ out of the $k$ nodes. This gives $g^tr^{(i)}$ 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 $u^{(i)}$. 
6.2 Construction for the Case $\alpha = 2$

6.2.2 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 $(f^T p(j), g^T r(j) + f^T \tilde{u}(j))$ 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

$$d \geq \max(l, l' + 1) \quad (6.5)$$

**Lemma 6.2.2** 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 $\pi_i = a_i (f^T p(\Lambda_i)) + b_i (g^T r(\Lambda_i) + f^T \tilde{u}(\Lambda_i))$ 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

$$\left(\sum_{i=1}^{d} \delta_i \pi_i, \sum_{i=1}^{d} \rho_i \pi_i\right) \quad (6.6)$$

Choose

$$b_i = \begin{cases} 1 & \text{for } i = 1, \ldots, l' + 1 \\ 0 & \text{for } i = l' + 2, \ldots, d \end{cases} \quad (6.7)$$

Now choose $\rho_i$ ($i = 1, \ldots, l' + 1$) such that

$$\sum_{i=1}^{l' + 1} \rho_i \underline{r}(\Lambda_i) = \underline{r}(\Lambda_{d+1}) \quad (6.8)$$

and $\rho_i = 0$ for $i = l' + 2, \ldots, d$.

Equation (6.8) is a set of $l'$ non-homogeneous linear equations in $l' + 1$ unknowns. Since $\underline{r}(\Lambda_i)$’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

\[
\sum_{i=1}^{d} \delta_i p^{(\Lambda_i)} = 0 \tag{6.9}
\]

and \( \delta_i = 1 \) for \( i = l'+2, \ldots, d \).

Equation (6.9) 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 \( \mathbb{F}_q \).

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

\[
\sum_{i=1}^{d} \delta_i (a_ip^{(\Lambda_i)} + b_i u^{(\Lambda_i)}) = p^{(\Lambda_{d+1})} \tag{6.10}
\]

i.e.

\[
\sum_{i=1}^{d} \delta_i a_ip^{(\Lambda_i)} = p^{(\Lambda_{d+1})} - \sum_{i=1}^{d} \delta_i b_i u^{(\Lambda_i)} \tag{6.11}
\]

Equation (6.11) is a set of \( l \) linear equations in \( d \) unknowns. Since \( d \geq l \), none of the \( \delta_i \) \((i = 1, \ldots, d)\) are zero, and \( p^{(\Lambda_i)} \)'s form a \( l \)-dimensional MDS code, it can be solved to obtain values for \( a_i \) \((i = 1, \ldots, d)\).

This process of exact regeneration of a failed node is illustrated in Figure 6.1 when replacement node \( n \) downloads one symbol each from \( d = 4 \) nodes: nodes 1, 2, 3 and 4, with \( l \leq 4, \ l' \leq 3 \).

### 6.2.3 Optimum Partition Size and Repair Bandwidth

The repair bandwidth for any node is

\[
\gamma_2(l, l') = \max(l, l' + 1) \tag{6.12}
\]

Thus, to minimize the repair bandwidth the partition sizes should be

\[
l = \left\lceil \frac{B}{2} \right\rceil \text{ and } l' = \left\lfloor \frac{B}{2} \right\rfloor \tag{6.13}
\]

**Remark 6.2.3** The case of \( l = l' = k \) corresponds to the MSR point with \( d = k+1 \) \((i.e., \ \alpha = 2, \ \beta = 1, \text{ and } B = 2k)\); this can be extended, via concatenation, to obtain codes for this point for higher values of \( \beta \).
6.3 Construction for an Arbitrary $\alpha$

This section gives a code construction for the general case using constructions for $\alpha = 1$ and $\alpha = 2$ as building blocks.

Let

$$\tau = \left\lceil \frac{\alpha}{2} \right\rceil$$

Partition the $B$ source symbols into $2\tau + 1$ sets $S_1, S'_1, \ldots, S_\tau, S'_\tau, S_{\tau+1}$ having sizes $l_1, l'_1, \ldots, l_\tau, l'_\tau, l_{\tau+1}$ respectively satisfying the following conditions

$$0 \leq l_j, l'_j \leq k \quad \forall j \in \{1, \ldots, \tau\}$$

$$l_{\tau+1} = 0 \text{ for } \alpha \text{ even},$$

$$0 \leq l_{\tau+1} \leq k \text{ for } \alpha \text{ odd},$$

and

$$\sum_{j=1}^{\tau} (l_j + l'_j) + l_{\tau+1} = B.$$  

Let $f_1, f'_1, \ldots, f_\tau, f'_\tau, f_{\tau+1}$ be $2\tau + 1$ vectors with their elements as the constituents of the sets $S_1, S'_1, \ldots, S_\tau, S'_\tau, S_{\tau+1}$ respectively. Hence the lengths of these vectors are $l_1, l'_1, \ldots, l_\tau, l'_\tau, l_{\tau+1}$ respectively.

For $j = 1, \ldots, \tau+1$ let $\mathbf{p}_j^{(i)}$ $(i = 1, \ldots, n)$ be vectors of length $l_j$ forming a $l_j$-dimensional
MDS code over \( \mathbb{F}_q \). For \( j = 1, \ldots, \tau \) let \( r_j^{(i)} \) \( (i = 1, \ldots, n) \) be vectors of length \( l_j' \) forming a \( l_j' \)-dimensional MDS code over \( \mathbb{F}_q \).

**Code:** For every pair of vectors \( (f_j, g_j) \), \( j = 1, \ldots, \tau \) apply the construction given for \( \alpha = 2 \) in Section 6.2 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 6.1 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

\[
\left\{ \left[ \begin{array}{c}
\frac{f_j^{(i)}}{g_j^{(i)}},
g_j^{(i)} + \frac{f_j^{(i)}}{u_j^{(i)}}
\end{array} \right] \right\}_{j=1}^{\tau}, \quad \frac{f_{\tau+1}^{(i)}}{f_{\tau+1}^{(i)}}
\]

as its \( \alpha \) symbols (as shown in Figure 6.2), where the symbol corresponding to \( f_{\tau+1} \) is present only when \( \alpha \) is odd.

### 6.3.1 Reconstruction

**Theorem 6.3.1** 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. Apply Lemma 6.2.1 separately on each pair of symbols stored in the \( k \) nodes to reconstruct \( \{f_j, g_j\}_{j=1}^{\tau} \).

When \( \alpha \) is odd, the last symbol stored is constructed using the code for \( \alpha = 1 \) given in Section 6.1. Apply Lemma 6.1.1 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.
6.3 Construction for an Arbitrary $\alpha$

6.3.2 Regeneration

**Theorem 6.3.2** 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 6.2.2. 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 6.1.2 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.

6.3.3 Optimum Partition Size and Repair Bandwidth

The repair bandwidth is dependent on the partition sizes. By the method of regeneration described in Theorem 6.3.2, 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} (6.20)

$$= \sum_{j=1}^{\tau} \max(l_j, l'_j + 1) + l_{\tau+1}$$  \hspace{1cm} (6.21)

where equation (6.21) follows from equations (6.2) and (6.12).

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} (6.22)

subject to the conditions (6.15), (6.16), (6.17), (6.18) 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 6.3.3** The bandwidth required to repair a failed node is upper bounded by

$$\gamma \leq B + \frac{\alpha}{2} + k - 1$$  \hspace{1cm} (6.23)
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 (6.22).

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 repair bandwidth. The number of buckets used will be $2 \min(\lfloor \frac{B}{2k-1} \rfloor, \lfloor \frac{\alpha}{2} \rfloor)$ 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. $\lfloor \frac{B}{2k-1} \rfloor < \lfloor \frac{\alpha}{2} \rfloor$.

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 $\lfloor \frac{B \mod (2k-1)}{2} \rfloor + 1$ to the repair bandwidth.

Case 2: The number of empty buckets remaining is at most 1 i.e. $\lfloor \frac{B}{2k-1} \rfloor \geq \lfloor \frac{\alpha}{2} \rfloor$.

The number of balls left will be $B - (2k-1) \lfloor \frac{\alpha}{2} \rfloor$. 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 repair bandwidth.

The repair bandwidth for any node is given by

$$\gamma = \begin{cases} 
\lfloor k \frac{B}{2k-1} \rfloor + \lfloor \frac{B \mod (2k-1)}{2} \rfloor + 1 & \text{for } \lfloor \frac{B}{2k-1} \rfloor < \lfloor \frac{\alpha}{2} \rfloor \\
B - (k-1) \lfloor \frac{\alpha}{2} \rfloor & \text{for } \lfloor \frac{B}{2k-1} \rfloor \geq \lfloor \frac{\alpha}{2} \rfloor
\end{cases}$$ \hspace{1cm} (6.24)

This expression can be simplified to obtain an upper bound on the repair bandwidth as given by (6.23).

Thus the repair 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 6.3. 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 (6.24) 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

$$\alpha_{\text{max}} = 2 \left\lceil \frac{B}{2k-1} \right\rceil.$$ \hspace{1cm} (6.25)
6.4 Analysis of the Code

**Repair bandwidth:** The construction provided here 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 *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.

**Field size required:** If we use a Reed-Solomon code as the MDS code in the construction, the minimum field size required is just

$$q \geq n.$$  \hfill (6.26)
Chapter 7

A Flexible Class of Regenerating Codes

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 $\gamma$. We motivate this framework in Section 7.1, and describe the setting in greater detail in Section 7.2.

In this flexible setting, we obtain the cut-set lower bound on the repair bandwidth in Section 7.3, followed by a constructive proof for the existence of codes meeting this bound for all values of the parameters in Section 7.4. An explicit code construction is provided in Section 7.5, that performs flexible reconstruction and regeneration, optimal in certain parameter regimes. Finally, Section 7.6 contains a construction, though not explicit, of symbol-wise MDS codes meeting the cut-set bound on repair bandwidth, that is highly useful when links are prone to errors.

Note that this chapter will deal with functional regeneration, and not its exact variant.

7.1 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 [34,35]. 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 7.1. Hence, freedom to download different amounts of data from the different nodes helps in reducing the net download time and traffic congestion. Such a system will also be highly conducive for load-balancing across the nodes in the network.

### 7.2 Description of the Flexible Setting

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 $\mu_i$, \quad $i = 1, \ldots, n$ satisfying

$$\sum_{i=1}^{n} \mu_i \geq B, \quad 0 \leq \mu_i \leq \alpha. \quad (7.1)$$

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 $\beta_i$ symbols from node $i$, $\forall i = 1, \ldots, n, \ i \neq \ell$ as long as

$$\sum_{i=1 (i \neq \ell)}^{n} \beta_i \geq \gamma, \quad 0 \leq \beta_i \leq \beta_{\text{max}}$$

(7.2)

for some value $\gamma$, the repair bandwidth. Here $\beta_{\text{max}}$ is a constant satisfying

$$0 \leq \beta_{\text{max}} \leq \alpha.$$  

(7.3)

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 7.3.3, 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

$$\alpha \geq \frac{B}{n}.$$  

(7.4)

We assume throughout that this condition is satisfied. We also assume that all system parameters are non-negative integers.

### 7.3 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 [4], 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 7.2 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
7.3 Lower bound on the Repair Bandwidth

A lower bound on the repair bandwidth is obtained by bounding the maximum flow in this network.

### 7.3.1 Information Flow Graph

On failure of a storage node, say node \( \ell \), it is replaced by a new node by connecting nodes \( \text{out}(j), j \in \{1, \ldots, n\}, j \neq \ell \), to \( \text{in}(\ell) \), with links of capacities \( \beta_j \), satisfying equation (7.2). 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 \( \text{out} \) nodes \( j (= 1, \ldots, n) \) to the sink, satisfying equation (7.1). 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.

### 7.3.2 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 \( \text{out} \) 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 7.2.

Consider the first storage node in \( V_C^c \) in the topological ordering. Call it node \( \Lambda_1 \). Since \( \text{out}(\Lambda_1) \in V_C^c \), the cut crosses either the \( \alpha \) capacity link between \( \text{in}(\Lambda_1) \) and \( \text{out}(\Lambda_1) \) or the set of links with total capacity \( \gamma \) entering \( \text{in}(\Lambda_1) \). We take the cut across the

![Figure 7.2: An information flow network. The cut separates the source from a particular DC. The edges marked in red are the edges crossing the cut.](image)

having capacities of \( \alpha \) symbols each.
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 out(\(\Lambda_1\)) to in(\(\Lambda_2\)) which will not be a part of the cut. Again, the cut will cross either the \(\alpha\) capacity link between in(\(\Lambda_2\)) and out(\(\Lambda_2\)) or the set of links with total capacity \((\gamma - \beta_{\text{max}})^+\) entering 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 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

\[
\sum_{j=0}^{r-1} \min (\alpha, (\gamma - j\beta_{\text{max}})^+) + (B - r\alpha)^+ \tag{7.5}
\]

The file size \(B\) has to be smaller than any cut and hence

\[
B \leq \min_{0 \leq r \leq n} \left\{ \sum_{j=0}^{r-1} \min (\alpha, (\gamma - j\beta_{\text{max}})^+) + (B - r\alpha)^+ \right\} \tag{7.6}
\]

**Lemma 7.3.1** A cut-set lower bound on the repair bandwidth \(\gamma\) is given by

\[
\gamma \geq \max (\alpha - \beta_{\text{max}}, B \mod \alpha) + s\beta_{\text{max}} \tag{7.7}
\]

**Proof**

Define \(s = \lfloor B/\alpha \rfloor\). \(\tag{7.8}\)

For \(r = s\), the equation (7.6) reduces to

\[
B \leq \sum_{j=0}^{s-1} \min (\alpha, (\gamma - j\beta_{\text{max}})^+) + (B - s\alpha) \tag{7.9}
\]

\[
s\alpha \leq \sum_{j=0}^{s-1} \min (\alpha, (\gamma - j\beta_{\text{max}})^+) \tag{7.10}
\]

\[
\leq s\alpha \tag{7.11}
\]

Thus, \((\gamma - j\beta_{\text{max}})^+ \geq \alpha\) \(\forall j \in \{0, \ldots, s - 1\}\) \(\tag{7.12}\)

which gives a lower bound on the repair bandwidth as

\[
\gamma \geq \alpha + (s - 1)\beta_{\text{max}} \tag{7.13}
\]
7.4 Achievability

For \( r = s + 1 \), equation (7.6) gives

\[
B \leq \sum_{j=0}^{s} \min(\alpha, (\gamma - j\beta_{\text{max}})^+) \tag{7.14}
\]

\[
= s\alpha + \min(\alpha, (\gamma - s\beta_{\text{max}})^+) \tag{7.15}
\]

where (7.15) is due to (7.13). This evaluates to

\[
\gamma \geq B - s\alpha + s\beta_{\text{max}} \tag{7.16}
\]

Combining (7.13) and (7.16) we get

\[
\gamma \geq \max(\alpha - \beta_{\text{max}}, B \mod \alpha) + s\beta_{\text{max}}. \tag{7.17}
\]

7.3.3 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 (7.17). This gives \( \gamma \geq B \), i.e., the repair bandwidth is equal to the size of the entire file.

7.4 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 7.3.1. 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

\[
\sum_{i=1}^{n} \mu_i = B, \quad 0 \leq \mu_i \leq \alpha \tag{7.18}
\]

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

\[
\sum_{i=1}^{n} \beta_i = \gamma, \quad 0 \leq \beta_i \leq \beta_{\text{max}} \tag{7.19}
\]

where \( \gamma \) meets the lower bound on the repair bandwidth given by Lemma 7.3.1. Then, it is clear that any DC with \( \sum_{i=1}^{n} \mu_i > B \) can recover the data, and any failed node with \( \sum_{i=1}^{n} (i \neq \ell) \beta_i > \gamma \) 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 7.4.1** For any set of $\mu_i$, $i = 1, \ldots, n$ and $\beta_i$, $i = 2, \ldots, n$,

$$\sum_{i=2}^{n} \min(\alpha, \mu_i + \beta_i) \geq B$$

(7.20)

subject to the conditions

$$\sum_{i=2}^{n} \mu_i \geq B - \mu_1, \quad 0 \leq \mu_j \leq \alpha \quad \forall j = 1, \ldots, n$$

$$\sum_{i=2}^{n} \beta_i \geq \gamma, \quad 0 \leq \beta_j \leq \beta_{\max} \quad \forall j = 2, \ldots, n$$

where $\gamma$ meets the cut-set bound given in equation (7.17) with equality.

**Proof** Please refer Appendix D ■

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 7.4.2** 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 (7.18), the new node can download $\beta_i$ symbols from node $i$ ($i = 1, \ldots, n$, $i \neq \ell$) satisfying the constraints given in (7.19) 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 (7.18). 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

$$\mu_1 > 0$$

(7.21)

Since $\mu_i$ and $\beta_i$ satisfy the conditions of Lemma 7.4.1,

$$\sum_{i=2}^{n} \min(\alpha, \mu_i + \beta_i) \geq B$$

(7.22)

Now, reduce the values of $\beta_i$ to $\beta'_i$, $\forall i = 2, \ldots, n$ such that equality is attained above, i.e.

$$\sum_{i=2}^{n} \min(\alpha, \mu_i + \beta'_i) = B$$

(7.23)

Thus we have $\beta'_i$ point-wise lesser than $\beta_i$

$$0 \leq \beta'_i \leq \beta_i \quad \forall i = 2, \ldots, n$$

(7.24)

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

$$\tilde{\mu}_i = \begin{cases} 
0 & i = 1 \\
\min(\alpha, \mu_i + \beta'_i) & i = 2, \ldots, n
\end{cases}$$

(7.25)

This is a valid DC since $0 \leq \tilde{\mu}_i \leq \alpha \quad \forall i$ and

$$\sum_{i=1}^{n} \tilde{\mu}_i = \sum_{i=2}^{n} \min(\alpha, \mu_i + \beta'_i) = B$$

(7.26)

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 $i = 2, \ldots, n$,

$$(\tilde{\mu}_i - \mu_i) = \min(\alpha, \mu_i + \beta'_i) - \mu_i$$

(7.27)

$$\leq \beta'_i$$

(7.28)

$$\leq \beta_i$$

(7.29)

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

$$\sum_{i=2}^{n} (\mu_i - \bar{\mu}_i) = \sum_{i=2}^{n} \bar{\mu}_i - \sum_{i=2}^{n} \mu_i$$  \hspace{1cm} (7.30)

$$= B - (B - \mu_1)$$  \hspace{1cm} (7.31)

$$= \mu_1$$  \hspace{1cm} (7.32)

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 7.4.3** 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 (7.19) and store $\alpha$ symbols such that all DCs satisfying (7.18) are simultaneously satisfied, provided the field size is large enough.

**Proof** Without loss of generality assume $\ell = 1$. Let $G^{(1)}, \ldots, G^{(n)}$ be the node matrices at the present stage where flexible reconstruction is satisfied for all DCs. Let $\tilde{G}^{(1)}$ 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,

$$\tilde{G}^{(1)} = Z \begin{bmatrix} \gamma_{V^{(2)}} G^{(2)} \\ \gamma_{V^{(3)}} G^{(3)} \\ \vdots \\ \gamma_{V^{(n)}} G^{(n)} \end{bmatrix}$$  \hspace{1cm} (7.33)

where $V^{(i)}$ 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. $Z$ 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 (7.18). 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

$$\mathcal{P}_{\Delta} = \det \begin{bmatrix} U^{(1)}_{\Delta} \tilde{G}^{(1)} \\ U^{(2)}_{\Delta} G^{(2)} \\ \vdots \\ U^{(n)}_{\Delta} G^{(n)} \end{bmatrix} \neq 0$$  \hspace{1cm} (7.34)
The above determinant can also be viewed as a polynomial \( P_\Delta \) in \( \mathbb{F}_q \) with entries of the matrices \( U_\Delta^{(i)} \) (\( i = 1, \ldots, n \)), \( V^{(i)}(i = 2, \ldots, n) \) and \( Z \) as variables. By Lemma 7.4.2 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 (7.34) is a non-zero polynomial.

For all DCs to be satisfied simultaneously, we need

\[
\prod_{\Delta \text{over all DCs}} P_\Delta \neq 0 \quad (7.35)
\]

This product is itself a polynomial with variables being entries of the matrices \( U_\Delta^{(i)} (i = 1, \ldots, n, \text{ all DCs } \Delta) \), \( V^{(i)} \) (\( i = 2, \ldots, n \)) and \( Z \). Since each polynomial in this product is non-zero, the product polynomial is also non-zero. Hence, the Schwartz-Zippel Lemma implies that there is an assignment to variables such that equation (7.35) is satisfied, provided the field size is large enough.

**Theorem 7.4.4 (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 7.4.3 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 (1.2) evaluates to

\[
d\beta \geq \alpha + (k - 1)\beta \quad . \quad (7.36)
\]

For the same parameters as that of the MSR point with \( \beta_{\text{max}} = \beta \), in the flexible regenerating codes setting, we get

\[
s = k \quad (7.37)
\]

and thus

\[
\gamma \geq \alpha + (k - 1)\beta \quad (7.38)
\]

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.

### 7.5 An Explicit Code

In this section, we show that the code constructed in Chapter 6 is, in fact, flexible in nature. We restrict our attention to the minimum storage case when \( B = s\alpha \), for some positive integer \( s \).

#### 7.5.1 An Example

Consider the parameters \( n = 6, \alpha = 4, \beta_{\text{max}} = 2, B = 12 \). 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 \( z^{(i)}_1 \) and \( z^{(i)}_2 \) 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>( f_1 v^{(i)} )</td>
</tr>
<tr>
<td>( g_1 )</td>
<td>( g_1 v^{(i)} + f_1 z^{(i)}_1 )</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>( f_2 v^{(i)} )</td>
</tr>
<tr>
<td>( g_2 )</td>
<td>( g_2 v^{(i)} + f_2 z^{(i)}_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 \( \mu_i \) symbols corresponding to the sets

\[
\left( \sum_{j=1}^{i-1} \mu_j + 1 \right) \text{ to } \left( \sum_{j=1}^{i-1} \mu_j + \mu_i \right) \equiv \mu_i \mod B .
\]

In the example, the symbols passed by the nodes to the DC are...
Since $\mathbf{z}^{(i)}$’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 $f_t^{(i)}\mathbf{z}_j^{(i)}$ from the other symbols, which leaves it with 3 symbols of the form $g_t^{(i)}\mathbf{v}^{(i)}$ and 3 symbols of the form $g_t^{(i)}\mathbf{v}^{(i)}$. Again the MDS property of $\mathbf{z}^{(i)}$’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 $f_t^{(i)}\mathbf{v}^{(i)} - f_t^{(i)}\mathbf{z}_1^{(i)}$, and the remaining 4 symbols can be combined to obtain $f_t^{(i)}\mathbf{v}^{(i)} + f_t^{(i)}\mathbf{z}_2^{(i)}$. Here $\mathbf{z}_1^{(i)}$ and $\mathbf{z}_2^{(i)}$ are not constrained to be equal to $\mathbf{z}_1^{(i)}$ and $\mathbf{z}_2^{(i)}$ since flexible reconstruction and regeneration operations are carried out irrespective of the values of these vectors.

### 7.5.2 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

$$\beta_{\text{max}} = \left\lceil \frac{\alpha}{2} \right\rceil.$$

Define integer $\theta$ as

$$\theta = \left\lfloor \frac{\alpha}{2} \right\rfloor.$$

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 $\mathbf{v}^{(i)} (i = 1, \ldots, n)$ 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 $\mathbf{z}_j^{(i)}$ 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.
7.6 Symbol-wise MDS Codes and Error-prone Links

Consider the case where links from the storage nodes to a data collector may be prone to errors. In such a case, the data collector may request for additional symbols from the storage nodes in order to perform error detection or correction. Further consider that the quality of the links varies from data collector to data collector, and also possibly over time for a single data collector. In such a case, the data collector may want to request for a fewer number of redundant symbols when the link is good, and a greater number when the link is more noisy.

This task can be easily accomplished 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 symbolwise MDS also satisfies the flexibility conditions outlined in this chapter, and its repair bandwidth is necessarily lower bounded by equation 7.7.

In this section, we prove the non-trivial property of flexible 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.

<table>
<thead>
<tr>
<th>Vector (set)</th>
<th>Symbol stored</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_1 )</td>
<td>( f_1^{t(i)} )</td>
</tr>
<tr>
<td>( g_1 )</td>
<td>( g_1^{t(i)} + f_1^{t(i)} )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>( f_\theta )</td>
<td>( f_\theta^{t(i)} )</td>
</tr>
<tr>
<td>( g_\theta )</td>
<td>( g_\theta^{t(i)} + f_\theta^{t(i)} )</td>
</tr>
<tr>
<td>( f_{\theta+1} )</td>
<td>( f_{\theta+1}^{t(i)} )</td>
</tr>
</tbody>
</table>

where the symbol corresponding to \( f_{\theta+1} \) is present only when \( \alpha \) is odd.

The processes of flexible reconstruction and regeneration in the example can be extended, with a little thought, for the general code.

**Repair Bandwidth:** In a manner similar to Chapter 6, the repair bandwidth required for this code can be obtained as

\[
\gamma = (s + 1) \left\lceil \frac{\alpha}{2} \right\rceil + s(\alpha \mod 2). \tag{7.41}
\]

Note that the code meets the cut-set bound when the parameter values are such that \( B \) is a multiple of \( \alpha \) and \( \beta_{\text{max}} = \left\lceil \frac{\alpha}{2} \right\rceil \).
Theorem 7.6.1 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 \( B \). From the \( n\alpha \) symbols described above, choose some \( \mu_1, \ldots, \mu_n \) symbols from nodes 1, \ldots, \( n \) respectively, with \( \sum_{i=1}^{n} \mu_i = B \), given by:

\[
G_1A_1E_1, \ldots, G_nA_nE_n
\]

where \( E_i \) is a selection matrix of size \( \alpha \times \mu_i \) whose each column is a unit vector, and each row has at most one 1.

Thus, the condition of these \( B \) symbols sufficing recovery of entire source data amounts to the following \( B \times B \) matrix being non-singular:

\[
\begin{bmatrix}
G_1A_1E_1 & \cdots & G_nA_nE_n
\end{bmatrix}
\]

(7.43)

Now, since the code can perform flexible reconstruction, the nodes can provide \( \mu_1, \ldots, \mu_n \) symbols respectively such that reconstruction is satisfied. Thus, there exist values of matrices \( A_1, \ldots, A_n \) which makes the matrix in equation 7.43 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 guarantees values 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 becomes symbol-wise MDS again via change of basis.\(^1\)

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 \( \mathbb{F}_2 \).

\(^1\)The symbols in the existing nodes may not remain symbol-wise MDS with the symbols in the new node. Hence even the existing symbols need to undergo a transform.
Chapter 8

Summary of the Work

This work focuses on construction of explicit codes for distributed storage that enable a data collector to reconstruct the entire source data, and a failed node to regenerate utilizing minimum possible repair bandwidth; and on the achievability of the storage-repair bandwidth tradeoff curve for the exact regeneration scenario.

The following is a summary of the results obtained in this work.

- Introduced a desirable property of *Exact* Regeneration, which considerably reduces overheads, and makes the system highly practical.

- For the minimum bandwidth regeneration (MBR) point:
  - Obtained explicit codes for all feasible values of the system parameters $[n, k, d]$ via a Product-Matrix framework.
  - Thus, proved that the tradeoff is indeed achievable at the MBR point for all parameters.
  - Obtained an explicit code for the case when $d = n - 1$ that has a simple graphical representation, and further, can be implemented using only XOR operation for a subset of parameters.

- For the minimum storage regeneration (MSR) point:
  - Obtained explicit codes for parameters $[n, k, d \geq 2k - 2]$ via a Product-Matrix framework.
  - Obtained optimal explicit codes for exact regeneration of systematic nodes for the parameter set $d \geq 2k - 1$ using the concept of interference alignment.
  - Obtained optimal schemes for the case $d \geq 2k - 3$, again, for exact regeneration of systematic nodes.
Established necessity of interference alignment and showed that the storage-repair bandwidth tradeoff curve is not achievable for the case $d < 2k - 3$ for the atomic case of $\beta = 1$.

Obtained schemes that can lower the repair bandwidth considerably for any parameter set $(k, d)$ which achieves the storage-repair bandwidth tradeoff lower bound for $d \geq 2k - 1$ for exact regeneration of systematic nodes.

- Obtained a new hybrid class of codes that simultaneously achieve minimum possible storage (i.e., $\alpha = \frac{d}{k}$), and minimum possible repair bandwidth (i.e., $d\beta = \alpha$), for all values of the parameters.

- Unlike all other work in the literature that restrict $n$ to be $d + 1$, obtained (as mentioned above) explicit codes that work for arbitrary values of $n$, which are highly useful in dynamic systems, and can be effectively employed for data dissemination across a network.

- Also demonstrated how these codes achieve maximum security, for the case when an eavesdropper has read access to some $l$ storage nodes.

- Established a set of properties, necessary for any exact-regenerating code. Also proved that codes performing exact regeneration at the interior points on the storage-repair bandwidth tradeoff do not exist (with the possible exception of the region of width at-most $\beta$ in the immediate vicinity of the MSR point).

- Also obtained achievable values of repair bandwidths for the interior points storage space sharing between the MSR and MBR codes.

- Obtained explicit codes which perform regeneration of any failed node using a repair bandwidth of approximately half the file size, for all values of the system parameters, a special case being the MSR $d = k + 1$ code.

- Introduced a flexible framework for regenerating codes. Derived a cut-set lower bound on the repair bandwidth for this setting, and proved it to be achievable for all values of the parameters.

- An optimal construction, though not explicit, of symbol-wise MDS codes meeting the cut-set bound on repair bandwidth is provided. These codes fit into the flexible framework, and are highly useful when links are prone to errors.

- Presented a few insights into coding for general non-multicast networks, about which very little is known in the literature.
Bibliography


Appendix A

Insights into General Network Coding Problems

We first present the problem of constructing codes for the MSR point, as an instance of a network coding problem, in Section A.1. We consider optimal exact regeneration of only the systematic nodes. Turns out that this is a non-multicast problem, about which very less is known in the literature [30–32]. The insights obtained en route to constructing distributed storage codes using interference alignment are then generalized to obtain a set of highly intuitive conditions, necessary and sufficient for code design in a general network of non-multicast type, in Section A.2. To illustrate the application of these conditions, a tighter bound on networks with crosslinks is derived in Section A.3.

A.1 As a Network Coding Problem

The network is viewed as having $k$ source nodes, each corresponding to a systematic node and generating $\alpha$ symbols each per unit time. The graph of the network is directed, delay free and acyclic. 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 [5], by (i) splitting each non-systematic node $m$ into two nodes: $m_{\text{in}}$ and $m_{\text{out}}$ with an edge of capacity $\alpha$ linking the two with (ii) all incoming edges arriving into $m_{\text{in}}$ and all outgoing edges emanating from $m_{\text{out}}$. 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 A.1.
A.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: As mentioned earlier we consider delay free, acyclic, directed graphs. We also assume the networks to be error free. We consider scalar linear network
coding solution for these networks. If $\mathbf{R}$ is the sum-rate of all the sources, then we define $\mathbf{u}$ to be an $\mathbf{R}$-length vector with its elements as the $\mathbf{R}$ symbols in the network at a time. We assume that each of these symbols belong to some finite field $\mathbb{F}_q$ of size $q$. Note that the notation used in this chapter will differ slightly from that introduced in Section 3.1.

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: $C_e$. Further, every edge $e$ is associated with a matrix $M_e$ of dimension $C_e \times \mathbf{R}$ 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 are $M_e \mathbf{u}$.

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 $T_i$ if it demands source $S_i$. The source and sink nodes do not have any incoming and outgoing edges respectively. Let $R_i$ be the rate of the source $S_i$. An edge from vertex $u$ to $v$ is represented as $u \rightarrow v$.

Define tail($e$) and head($e$) to be the tail and head vertices of edge $e$ respectively. If the tail($e$) of an edge $e$ is a source node, then the columns of $M_e$, corresponding to any other systematic node are mandated to be zero. For any other edge $e$, $M_e$ is a linear
A cut $\Omega$ (as illustrated in Figure A.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. The net rate $R = \sum_{i=1}^k R_i$.

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 $M_e$ $\forall e \in \Omega$ restricted to the columns corresponding to the sources in $S$. Let $E(\Omega)$ for denote the set of all rows of $M_e$ $\forall e \in \Omega$.

**Theorem A.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 \(\Omega\),

\[
\dim(S_D(\Omega), \Omega) \geq R_D(\Omega)
\]  

(A.1)

**Theorem A.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 \(\Omega\),

\[
\dim(S_I(\Omega), \Omega) \leq C(\Omega) - R_D(\Omega)
\]  

(A.2)

**Theorem A.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 apply these properties, to demonstrate how the presence of crosslinks in a network can be exploited to tighten upper bounds on achievable rates.

**A.3 Bound for Networks with Crosslinks**

**Definition A.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 A.4. Here, \(S_2 \rightarrow T_1\) in Figure A.4a, and \(S_2 \rightarrow T_1\) and \(S_1 \rightarrow T_2\) in Figure A.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 A.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

\[
R_D(\Omega) \leq \tilde{C}(\Omega) + C_{CL}(S_{D1}(\Omega), T_{D2}(\Omega))
\]  

(A.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 A.2.2,

\[
\dim(S_{D1}(\Omega), \Omega) \leq \tilde{C}(\Omega) - R_{D2}(\Omega) + C_{CL}(S_{D1}(\Omega), T_{D2}(\Omega)) \tag{A.4}
\]

Since \( S_{D1}(\Omega) \subseteq S_D(\Omega) \), by Theorem A.2.1 we get

\[
R_{D1}(\Omega) \leq \dim(S_{D1}(\Omega), \Omega) \tag{A.5}
\]

Thus from equations (A.4) and (A.5),

\[
R_{D1}(\Omega) \leq \tilde{C}(\Omega) - R_{D2}(\Omega) + C_{CL}(S_{D1}(\Omega), T_{D2}(\Omega)) \tag{A.6}
\]

This leads to equation (A.3). \( \blacksquare \)

**Example 1** Consider the butterfly network in Figure A.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, \( S_{D2}(\Omega) = \{S_2\} \).

\[
C_{CL}(S_1(\Omega), T_2(\Omega)) = 0 \tag{A.7}
\]

Removing the crosslink \( S_2 \rightarrow T_1 \), we get

\[
\tilde{C}(\Omega) = 1 \tag{A.8}
\]

Thus from equation (A.3), we get

\[
R_1 + R_2 \leq 1 \tag{A.9}
\]

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 A.4b. Here, the cut-set bound and the bound given by Theorem A.3.2 coincide to give \( R_1 \leq 1, R_2 \leq 1 \) and \( R_1 + R_2 \leq 2 \), which is achievable.
Figure A.4: Two modifications of the butterfly network. Each edge has unit capacity.
Appendix B

Miscellaneous Proofs from Chapter 3

B.1 Proof of Theorem 3.4.2: 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

$$R = \begin{bmatrix}
G'(\delta_1) & G'(\delta_2) & \cdots & G'(\delta_p) \\
G_{\Omega_1}' & G_{\Omega_2}' & \cdots & G_{\Omega_1}' \\
\vdots & \vdots & \ddots & \vdots \\
G_{\Omega_p}' & G_{\Omega_2}' & \cdots & G_{\Omega_p}'
\end{bmatrix} \quad (B.1)$$

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 $S_{i,j} = \psi_i^{(\delta_j)}$, $i = 1, \ldots, \alpha$, $j = 1, \ldots, p$. Let $T_{a,b}$ be an $\alpha \times p$ matrix with its $a$th row as $[\psi_{b}^{(\delta_1)}, \ldots, \psi_{b}^{(\delta_p)}]$, 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

$$[R']_{(i,j)} = \begin{cases}
\epsilon S & \text{if } i = j \\
T_{\Omega_i,\Omega_i} & \text{if } i \neq j
\end{cases} \quad (B.2)$$
For $i = 1, \ldots, p$, $j = p+1, \ldots, \alpha$,

$$[R']_{(i,j)} = T_{\omega_{j-p}, \Omega_i}$$  \hfill (B.3)

Thus,

$$R' = \begin{bmatrix}
\epsilon S & T_{\Omega_2, \Omega_1} & \cdots & T_{\Omega_p, \Omega_1} & T_{\omega_1, \Omega_1} & \cdots & T_{\omega_{\alpha-p}, \Omega_1} \\
T_{\Omega_1, \Omega_2} & \epsilon S & \cdots & T_{\Omega_p, \Omega_2} & T_{\omega_1, \Omega_2} & \cdots & T_{\omega_{\alpha-p}, \Omega_2} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
T_{\Omega_1, \Omega_p} & T_{\Omega_2, \Omega_p} & \cdots & \epsilon S & T_{\omega_1, \Omega_p} & \cdots & T_{\omega_{\alpha-p}, \Omega_p}
\end{bmatrix}$$  \hfill (B.4)

Let $\tilde{S}$ be the $p \times p$ matrix formed by the columns $\Omega_1, \ldots, \Omega_p$ 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

$$\begin{bmatrix}
\epsilon S & T_{\Omega_2, \Omega_1} & \cdots & T_{\Omega_p, \Omega_1} & E_{\omega_{1, \Omega_1}} & \cdots & E_{\omega_{\alpha-p, \Omega_1}} \\
T_{\Omega_1, \Omega_2} & \epsilon S & \cdots & T_{\Omega_p, \Omega_2} & E_{\omega_{1, \Omega_2}} & \cdots & E_{\omega_{\alpha-p, \Omega_2}} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
T_{\Omega_1, \Omega_p} & T_{\Omega_2, \Omega_p} & \cdots & \epsilon S & E_{\omega_{1, \Omega_p}} & \cdots & E_{\omega_{\alpha-p, \Omega_p}}
\end{bmatrix}.$$  \hfill (B.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 $\alpha \times p$ 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

$$\begin{bmatrix}
\epsilon S & T_{2, \Omega_1} & \cdots & T_{p, \Omega_1} \\
T_{1, \Omega_2} & \epsilon S & \cdots & T_{p, \Omega_2} \\
\vdots & \vdots & \ddots & \vdots \\
T_{1, \Omega_p} & T_{2, \Omega_p} & \cdots & \epsilon S
\end{bmatrix}.$$  \hfill (B.6)

This is equivalent decoding 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

$$
\begin{bmatrix}
\epsilon I_p & E_{2,1} & E_{3,1} & \cdots & E_{p,1} \\
E_{1,2} & \epsilon I_p & E_{3,2} & \cdots & E_{p,2} \\
: & : & : & \ddots & : \\
E_{1,p} & E_{2,p} & E_{3,p} & \cdots & \epsilon I_p
\end{bmatrix}
$$

(B.7)

where $I_p$ is a $p \times p$ identity matrix and $E_{a,b}$ 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 components of these symbols along the other symbols is subtracted. 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:

$$
\begin{bmatrix}
\epsilon & 1 & 0 & 0 & \cdots & 0 & 0 \\
1 & \epsilon & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & \epsilon & 1 & \cdots & 0 & 0 \\
0 & 0 & 1 & \epsilon & \cdots & 0 & 0 \\
: & : & : & : & \ddots & : & : \\
0 & 0 & 0 & 0 & \cdots & \epsilon & 1 \\
0 & 0 & 0 & 0 & \cdots & 1 & \epsilon
\end{bmatrix}
$$

(B.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 3.4.1 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,
B.2 Proof of Theorem 3.6.2: Existence and Construction for \( d \geq 2k - 3 \)

**Proof** Let \( d = 2k - 3 \). Consider the exact regeneration of the systematic node \( \alpha + 1 \). By Theorem 3.5.4, in the vectors passed by the \( \alpha \) non-systematic nodes i.e.,

\[
\{\mathbf{v}^{(m_1,\alpha+1)}, \ldots, \mathbf{v}^{(m_\alpha,\alpha+1)}\}
\]

(B.9)

the component along the systematic nodes \( l, \forall l \in \{1, \ldots, \alpha\} \), need to be aligned in one direction. This leads to the following set of \( n - k - 1 \) equations: for \( m = k + 2, \ldots, n \),

\[
G_i^{(m)}(m,\alpha+1) x_l^{(m,\alpha+1)} = \kappa_i^{(m,\alpha+1)} G_i^{(k+1)}(k+1,\alpha+1) x_l^{(k+1,\alpha+1)}
\]

(B.10)

Similarly, alignment for the exact regeneration of the systematic node \( \alpha + 2 \) leads to another set of \( n - k - 1 \) equations: \( m = k + 2, \ldots, n \),

\[
G_i^{(m)}(m,\alpha+2) x_l^{(m,\alpha+2)} = \kappa_i^{(m,\alpha+2)} G_i^{(k+1)}(k+1,\alpha+2) x_l^{(k+1,\alpha+2)}
\]

(B.11)

for some constants \( \kappa \)'s \( \in \mathbb{F}_q \).

For all \( m \in \{k + 2, \ldots, n\} \), multiply equation (B.10) by \( (x_i^{(m,\alpha+1)})^{-1} \) and (B.11) by \( (x_i^{(m,\alpha+2)})^{-1} \) and subtract the two. \( h_{l,l}^{(m)} \) gets eliminated and a homogeneous equation in terms of \( h_{l,1}, \ldots, h_{l,t-1}, h_{l,t}^{(k+1)}, h_{l,t+1}, \ldots, h_{l,\alpha} \) remains. One way to satisfy this equation is to equate all the scalar coefficients to zero.

Making the coefficients of \( h_{l,1}, \ldots, h_{l,t-1}, h_{l,t+1}, \ldots, h_{l,\alpha} \) zero gives, for \( l = 1, \ldots, \alpha, m = \)
k + 2, . . . , n and i = 1, . . . , α, i ≠ l,

\[ \lambda^{(m)}_{l,i} = \lambda^{(k+1)}_{l,i} \cdot \left[ \kappa^{(m,\alpha+1)}_{l}(x^{(m,\alpha+1)}_{l})^{-1}x^{(k+1,\alpha+1)}_{i} - \kappa^{(m,\alpha+2)}_{l}(x^{(m,\alpha+2)}_{l})^{-1}x^{(k+1,\alpha+2)}_{i} \right]. \]

\[ \left[ (x^{(m,\alpha+1)}_{l})^{-1}x^{(m,\alpha+1)}_{i} - (x^{(m,\alpha+2)}_{l})^{-1}x^{(m,\alpha+2)}_{i} \right]^{-1} \]

(B.12)

Making the coefficient of \( h^{(k+1)}_{l,l} \) zero gives, for \( m = k + 2, . . . , n \) and \( l = 1, . . . , \alpha \)

\[ \kappa^{(m,\alpha+2)}_{l} = \kappa^{(m,\alpha+1)}_{l}x^{(k+1,\alpha+1)}_{l}(x^{(m,\alpha+1)}_{l})^{-1}(x^{(k+1,\alpha+2)}_{l})^{-1}x^{(m,\alpha+2)}_{l} \]

(B.13)

Equations (B.12) and (B.13) ensure that second set of equations (i.e. B.11) are satisfied whenever the first set (i.e. B.10) is satisfied. Note that any polynomial containing a either \( \lambda^{(m)}_{l,i} \) (i ≠ 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 (B.10) have to be satisfied, for which, we make the following assignments, for \( m = k + 2, . . . , n \) and \( l = 1, . . . , \alpha \)

\[ H^{(m)}_{l,l} = H^{(k+1)}_{l,l} \left( (x^{(m,\alpha+1)}_{l})^{-1}k^{(m,\alpha+1)}_{l}x^{(k+1,\alpha+1)}_{i} \right) + \sum_{i=1, i\neq l}^{\alpha} H^{(m)}_{l,i} \left[ (x^{(m,\alpha+1)}_{l})^{-1}(k^{(m,\alpha+1)}_{l}k^{(k+1,\alpha+1)}_{l}x^{(m,\alpha+1)}_{i} - \lambda^{(m)}_{l,i}x^{(m,\alpha+1)}_{i}) \right] \]

(B.14)

Alignment of components along systematic nodes \( \{1, . . . , \alpha\} \) 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 3.5.4, the component along the systematic node \( \alpha + 1 \) in the vector passed by non-systematic nodes need to be aligned in one direction. This leads to the following set of \( n - k - 1 \) equations: For \( m = k + 2, . . . , n \)

\[ H^{(m)}_{\alpha+1}\Lambda^{(m)}_{\alpha+1} = \kappa^{(m,\alpha+2)}_{\alpha+1}H^{(k+1)}_{\alpha+1}\Lambda^{(k+1)}_{\alpha+1} \]

(B.15)
From equation (3.90) we have
\[ H_{\alpha+1}^{(m)} = H_{\alpha+1}^{(k+1)} = H_{\alpha+1} \text{ (say)} \] (B.16)

Thus, equating the scalar coefficients to zero in equation (B.15), we get for \( i = 1, \ldots, \alpha \),
\[ \lambda_{\alpha+1,i}^{(m)} = \lambda_{\alpha+1,i}^{(k+1)} (x_i^{(m,\alpha+2)})^{-1} \kappa_{\alpha+1}^{(m,\alpha+2)} x_i^{(k+1,\alpha+2)} \] (B.17)

Similarly, for the exact regeneration of node \( \alpha + 1 \), we need to align components along node \( \alpha + 2 \) which leads to
\[ \lambda_{\alpha+2,i}^{(m)} = \lambda_{\alpha+2,i}^{(k+1)} (x_i^{(m,\alpha+1)})^{-1} \kappa_{\alpha+2}^{(m,\alpha+1)} x_i^{(k+1,\alpha+1)} \] (B.18)

**Regeneration**

Exact regeneration of each one of the systematic nodes \( l \in \{1, \ldots, \alpha \} \) results in a condition
\[ \det \left[ \begin{array}{ccc} h_{l,l}^{(m_1)} & \cdots & h_{l,l}^{(m_\alpha)} \end{array} \right] \neq 0 \] (B.19)

where \( m_1, \ldots, m_\alpha \) are the \( \alpha \) non-systematic nodes participating in the regeneration. After substituting for \( h_{l,l}^{(m_i)} \), \( i = 1, \ldots, \alpha \) from equation (B.14), this condition evaluates to a rational polynomial, which can be shown to be not identically equal to zero by the following assignments: For \( i = 1, \ldots, \alpha \), \( i \neq l \), \( m \in \{m_1, \ldots, m_\alpha\} \), \( m \neq k + 1 \)

\[
\begin{align*}
\kappa_{l}^{(m,\alpha+1)} &= 1, & \lambda_{l,i}^{(k+1)} &= -1, & h_{l,l}^{(k+1)} &= \epsilon_l, & h_{l,i}^{(k+1)} &= \epsilon_i \\
x_{l}^{(m,\alpha+2)} &= 1, & x_{l}^{(k+1,\alpha+1)} &= x_{l}^{(k+1,\alpha+1)} = 1 \\
x_{i}^{(k+1,\alpha+1)} &= 0, & x_{i}^{(k+1,\alpha+2)} &= 1, & x_{i}^{(m,\alpha+1)} &= 1, \\
x_{i}^{(m,\alpha+2)} &= (m - k)^{-j} + 1
\end{align*}
\] (B.20)

where \( j = i \) if \( i < l \), \( j = i - 1 \) if \( i > l \). These assignments make the matrix under consideration in equation (B.19) a Vandermonde matrix which is full rank, and ensures that equations (B.12), (B.17) and (B.18) 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 3.5.2 implies that the coefficients resulting from the linear combinations need to be linearly independent, i.e \( \Lambda_{\alpha+1}^{(m)} \xi^{(m,\alpha+1)} \) 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 \( i = 1, \ldots, \alpha, \ m = k + 2, \ldots, n \)

\[
\Lambda_{\alpha+1}^{(k+1)} = I, \quad \kappa_{\alpha+1}^{(m,\alpha+1)} = 1, \quad \kappa_{\alpha+1}^{(m,\alpha+2)} = 1, \\
x_i^{(m,\alpha+2)} = 1, \quad x_i^{(m,\alpha+1)} = (m-k)^i, \\
x_i^{(k+1,\alpha+2)} = 1, \quad x_i^{(k+1,\alpha+1)} = 1 .
\] (B.21)

Note that these assignments also ensure that equations (B.12), (B.17) and (B.18) remain valid. A similar argument can be used to obtain a condition for regeneration of node \( \alpha + 2 \).

Since the hypothesis of Lemma 3.6.1 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

\[
det \begin{pmatrix}
G_{l_1}^{(m_1)} & \cdots & G_{l_1}^{(m_p)} \\
\vdots & \ddots & \vdots \\
G_{l_p}^{(m_1)} & \cdots & G_{l_p}^{(m_p)}
\end{pmatrix}
\] (B.22)

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

\[
\kappa_{l_i}^{(m_i,\alpha+1)} = \begin{cases} 
1 & \text{if } i = j \\
0 & \text{if } i \neq j 
\end{cases}
\] (B.23)

For \( i = 1, \ldots, \alpha, \ m = k + 2, \ldots, n, \ l = 1, \ldots, k \), set

\[
H_l^{(k+1)} = I, \quad \Lambda_l^{(k+1)} = I, \\
x_i^{(m,\alpha+1)} = x_i^{(k+1,\alpha+1)} = \eta_i, \\
x_i^{(m,\alpha+2)} = x_i^{(k+1,\alpha+2)} = 1 .
\] (B.24)

where \( \eta_i \neq \eta_j \) for \( i \neq j \). With these values, from (B.12), (B.14), (B.17) and (B.18), we get that the matrix in equation (B.22) is full rank for a large enough field size, and also
the equations (B.12), (B.17) and (B.18) 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 $d = 2k - 3$.

B.3 Proof of Theorem 3.8.1: Uniqueness of the Explicit Construction

**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

$$\begin{bmatrix}
    v^{(m,1)} & v^{(m,2)} & \cdots & v^{(m,k)}
\end{bmatrix}
$$

(B.25)

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, $\{ v^{(m,1)}, v^{(m,2)}, \ldots, v^{(m,k)} \}$ 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 proved in Theorem 3.5.4, forces the generator matrices to have the form

$$G_i^{(m)} = \left[ h_{i,1} \cdots h_{i,i-1} h_{i,i} h_{i,i+1} \cdots h_{i,k} \right] \Lambda_i^{(m)} = \tilde{h}_{i,i} e_i^t + \tilde{H}_i \Lambda_i^{(m)}$$

(B.26)

where $\Lambda_i^{(m)}$ is a $k \times k$ invertible diagonal matrix given by

$$\Lambda_i^{(m)} = \text{diag}\{\lambda_i^{(m)}, \lambda_i^{(m)}, \ldots, \lambda_i^{(m)}\},$$

(B.27)

$\tilde{H}_i$ is a $k \times k$ non-singular matrix and

$$\tilde{h}_{i,i}^{(m)} = \lambda_i^{(m)} h_{i,i} - \tilde{H}_i \Lambda_i^{(m)} e_i.$$
Now, pre-multiply the $B \times n\alpha$ generator matrix by the non-singular matrix
\[
\begin{bmatrix}
\tilde{H}_1^{-1} & 0 & \ldots & 0 \\
0 & \tilde{H}_2^{-1} & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & \tilde{H}_k^{-1}
\end{bmatrix}
\begin{bmatrix}
G^{(1)} & \ldots & G^{(n)}
\end{bmatrix}
\]
(B.29)

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
\[
\tilde{H}_i^{-1}G_i^{(m)} = \tilde{H}_i^{-1}\underline{l}_{i,i}^{(m)}\xi_i + \Lambda_i^{(m)}
\]
(B.30)

where the RHS follows from equation (B.26). Now, choosing
\[
D_i^{(m)} = \Lambda_i^{(m)} - \lambda_i^{(m)}\xi_i\xi_i^t
\]
(B.31)

\[
f_i^{(m)} = \tilde{H}_i^{-1}\underline{l}_{i,i}^{(m)} + \lambda_i^{(m)}\xi_i
\]
(B.32)
gives the generator matrices of the non-systematic nodes of an equivalent code in the form given in equation (3.105).

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 3.5.2, the components of these vectors along $\tilde{z}_i$ should be linearly independent. These components are nothing but $f_i^{(m)}$, $m = k + 1, \ldots, 2k$ and this shows that the matrix $F_i$ is non-singular.

Now, left to show is that any square submatrix of $\Delta_j$, $\leq j \leq k$ is non-singular. Consider the following submatrix
\[
\begin{bmatrix}
d_{11,j}^{(m_1)} & \ldots & d_{11,j}^{(m_p)} \\
\vdots & \ddots & \vdots \\
d_{p1,j}^{(m_1)} & \ldots & d_{p1,j}^{(m_p)}
\end{bmatrix}
\]
(B.33)

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

\[
\begin{bmatrix}
  d_{i1,j}^{(m_1)} e_j & \cdots & d_{i1,j}^{(m_p)} e_j \\
  \vdots & \ddots & \vdots \\
  d_{ip,j}^{(m_1)} e_j & \cdots & d_{ip,j}^{(m_p)} e_j
\end{bmatrix}
\]

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 (B.33) to be non-singular. □
Appendix C

Proofs of Subspace Properties of Linear Exact Regenerating Codes

Proof of Property 4.2.3 Consider a data collector connecting to any \( k \) nodes, say \( \Lambda_1, \ldots, \Lambda_k \), and let
\[
|W_{\Lambda_i}| = \Omega_i, \quad \forall i \in \{1, \ldots, n\}. \tag{C.1}
\]
Since a node can store a subspace of dimension no more than \( \alpha \),
\[
\Omega_i \leq \alpha, \quad \forall i \in \{1, \ldots, k\}. \tag{C.2}
\]

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 \( l = 2, \ldots, k \)
\[
|W_{\Lambda_l} \cap \{W_{\Lambda_{l-1}} + \cdots + W_{\Lambda_1}\}| \geq (\Omega_l - (d-(l-1))\beta)^+ \tag{C.3}
\]
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.,
\[
|W_{\Lambda_1} + W_{\Lambda_2} + \cdots + W_{\Lambda_k}| = B. \tag{C.4}
\]
Using the expression for the dimension of sum of two subspaces recursively, we get

\[
B = |W_{\Lambda_1} + \cdots + W_{\Lambda_k}|
\]

\[
= |W_{\Lambda_1}| + |W_{\Lambda_2}| - |W_{\Lambda_1} \cap W_{\Lambda_2}|
\]

\[
+ |W_{\Lambda_3}| - |W_{\Lambda_3} \cap \{W_{\Lambda_1} + W_{\Lambda_2}\}|
\]

\[\vdots\]

\[
+ |W_{\Lambda_k}| - |W_{\Lambda_k} \cap \{W_{\Lambda_1} + \cdots + W_{\Lambda_{k-1}}\}|.
\] (C.5)

Applying inequality (C.3) to equation (C.5), we get

\[
B \leq \Omega_1 + \sum_{i=2}^{k} \left[ \Omega_i - (\Omega_i - (d - (i - 1))\beta)^+ \right]
\] (C.6)

\[
= \Omega_1 + \sum_{i=2}^{k} \min \left[ \Omega_i, (d - (i - 1))\beta \right]
\] (C.7)

\[
\leq \alpha + \sum_{i=2}^{k} \min \left[ \alpha, (d - (i - 1))\beta \right]
\] (C.8)

\[
= \alpha + \sum_{i=1}^{k-1} \min \left[ \alpha, (d - i)\beta \right]
\] (C.9)

\[
= \sum_{i=0}^{k-1} \min \left[ \alpha, (d - i)\beta \right]
\] (C.10)

\[
= B.
\] (C.11)

Here, equation (C.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 (C.8) follows from (C.2), and (C.10) follows from the range of \(\alpha\) given in (1.5).

Thus, equation (C.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 4.2.4** For \(a \geq k\) the result is trivially true since (i) the nodal subspaces of any \(k\) nodes span the entire space, and (ii) \(|W_l| = \alpha\).

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.,

$$|W_{\Lambda_1} + W_{\Lambda_2} + \cdots + W_{\Lambda_k}| = B. \quad \text{(C.12)}$$

Turning our attention again to equation (C.3), we get

$$\left| W_{\Lambda_{a+1}} \cap \sum_{i=1}^{a} W_{\Lambda_i} \right| \geq \left| W_{\Lambda_{a+1}} \right| - (d-a)\beta + \theta. \quad \text{(C.13)}$$

Substituting the value of $|W_{\Lambda_{a+1}}| = \alpha = (d-p)\beta - \theta$ (from Property 4.2.3) we get, for $1 \leq a < k$,

$$\left| W_{\Lambda_{a+1}} \cap \sum_{i=1}^{a} W_{\Lambda_i} \right| \geq \begin{cases} 0 & a \leq p \\ (a-p)\beta - \theta & p < a < k. \end{cases} \quad \text{(C.14)}$$

Now, following the same steps as in Property 4.2.3 (i.e., substituting equation (C.14) in (C.5)), one can see that the inequality (C.14) needs to be satisfied with equality. □

**Proof of Property 4.2.5** 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 $m \in B_2$. Define three subspaces:

$$V_1 = W_l, \quad V_2 = \sum_{j \in B_1} S^{(j,l)}, \quad \text{and} \quad V_3 = \sum_{j \in B_2} S^{(j,l)}.$$  

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 4.2.4 we get

$$|V_1 \cap V_2| \leq \left| W_l \cap \sum_{j \in B_1} W_j \right| \quad \text{(C.15)}$$

$$= \beta - \theta. \quad \text{(C.16)}$$

Now,

$$(d-p-1)\beta \geq |V_3| \quad \text{(C.17)}$$

$$\geq |V_3 \cap (V_1 + V_2)| \quad \text{(C.18)}$$

$$\geq |V_1| - |V_1 \cap V_2| \quad \text{(C.19)}$$

$$\geq (d-p-1)\beta, \quad \text{(C.20)}$$
where equation (C.19) follows from Lemma 4.2.1 and equation (C.20) from equation (C.16). This implies that the inequalities in (C.17) to (C.20) are satisfied with equality, and thus we have

\[ V_3 \subseteq V_1 + V_2 \]  

(C.21)

and

\[ |V_3| = (d - p - 1)\beta. \]  

(C.22)

From equation (C.22) we get

\[ |S^{(j,l)}| = \beta, \quad \forall \ j \in B_2, \]  

(C.23)

and in particular, \( |S^{(m,l)}| = \beta. \)

The case of \( p = k - 1 \) is proved in a manner similar to the preceding case. Recall that by definition, the parameter \( \theta \) is zero in this case. Choose the sets \( B_1 \) and \( B_2 \) to have cardinalities \( p \) and \( d - p \) respectively. Now, Property 4.2.4 asserts that \( |V_1 \cap V_2| = 0 \), and proceeding in a manner identical to steps (C.17) through (C.20) (using the fact that \( \theta = 0 \)), one can deduce equations (C.21) and (C.23).

**Proof of Property 4.2.6** The proof described here assumes the set \( A \) to have exactly \( p + 2 \) nodes; the scenario of \( A \) containing fewer nodes is a trivial extension.

Consider regeneration of node \( l \in A \) connecting to a set of \( d \) nodes that includes nodes \( \{m\} \cup A \setminus \{l\} \). Define a set of \( p + 1 \) nodes \( B_1 = A \setminus \{l\} \), and let \( B_2 \) be a second set comprising of the remaining \( (d - p - 1) \) nodes participating in this process of regeneration. Note that \( m \in B_2 \). Further, define three vector spaces

\[ V_1 = W_l, \quad V_2 = \sum_{j \in B_1} S^{(j,l)}, \quad \text{and} \quad V_3 = \sum_{j \in B_2} S^{(j,l)}. \]

Now, traversing the steps outlined in the proof of Property 4.2.5, we get \( V_3 \subseteq V_1 + V_2 \), and hence

\[ S^{(m,l)} \subseteq V_1 + V_2 \]  

(C.24)

\[ \subseteq \sum_{j \in A} W_j. \]  

(C.25)

Since the choice of node \( l \) from set \( A \) was arbitrary, the result in equation (C.25) holds for every node in \( A \). Moreover, the vector space \( S^{(m,l)} \) is contained in \( W_m \), which leads to

\[ \sum_{l \in A} S^{(m,l)} \subseteq \left( W_m \bigcap \sum_{j \in A} W_j \right). \]  

(C.26)
Finally, since the cardinality of set \( A \) is \((p + 2) < k\), Property 4.2.4 gives

\[
\left| W_m \cap \sum_{j \in A} W_j \right| = 2\beta - \theta, \quad \text{(C.27)}
\]

and the result immediately follows. ■

**Proof of Property 4.2.7** 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 a set of \( d \) nodes that includes nodes \( \{m\} \cup A \setminus \{l\} \). Define a set of \( p \) 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 4.2.4, and noting that the cardinality of set \( B_1 \) is \( p \), we get

\[
|V_1 \cap V_2| = 0. \quad \text{(C.28)}
\]

Furthermore, since \( V_1 \subseteq V_2 + V_3 \) for regeneration of node \( l \), Lemma 4.2.1 gives

\[
|V_3 \cap (V_1 + V_2)| = |V_1| - |V_1 \cap V_2| + |V_3 \cap V_2| \geq |V_1| \quad \text{(C.29)}
\]

\[
\geq (d - p)\beta - \theta \quad \text{(C.30)}
\]

\[
\geq |V_3| - \theta. \quad \text{(C.31)}
\]

Next, define three more subspaces

\[
U_1 = S^{(m,l)}, \quad U_2 = \sum_{j \in B_2 \setminus \{m\}} S(j,l), \quad \text{and} \quad U_3 = W_l + \sum_{j \in B_1} S(j,l). \quad \text{(C.33)}
\]

Rewriting equation (C.32) in terms of these subspaces,

\[
|(U_1 + U_2) \cap U_3| \geq |U_1 + U_2| - \theta. \quad \text{(C.34)}
\]

This, in conjunction with Lemma 4.2.2, implies

\[
|U_1 \cap U_3| \geq |U_1| - \theta. \quad \text{(C.35)}
\]
Property 4.2.5 mandates the dimension of $|U_1| = |S^{(m,l)}|$ to be $\beta$, and hence the previous equation evaluates to

$$|S^{(m,l)} \cap \left( \sum_{j \in A} W_j \right) | \geq \beta - \theta.$$  \hfill (C.36)

Since the cardinality of set $A$ is $(p + 1) < k$, Property 4.2.4 gives

$$|W_m \cap \left( \sum_{j \in A} W_j \right) | = \beta - \theta.$$  \hfill (C.37)

Now, as $S^{(m,l)} \subseteq W_m$, it follows from the two preceding equations:

$$W_m \cap \left( \sum_{j \in A} W_j \right) \subseteq S^{(m,l)}.$$  \hfill (C.38)

Since the choice of node $l$ from set $A$ was arbitrary, the result in equation (C.38) holds for every node in $A$, which leads to

$$W_m \cap \left( \sum_{j \in A} W_j \right) \subseteq \left( \bigcap_{l \in A} S^{(m,l)} \right).$$  \hfill (C.39)

The result follows immediately from equations (C.37) and (C.39). \hfill $\blacksquare$

**Proof of Corollary 4.2.8** The proof is inductive. The result is trivially true 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 4.2.7

$$\beta - \theta \leq \left| S^{(m,j)} \cap S^{(m,i)} \right|$$  \hfill (C.40)

$$\leq \left| S^{(m,j)} \cap \sum_{l \in A'} S^{(m,l)} \right|$$  \hfill (C.41)

$$= \left| S^{(m,j)} \right| + \left| \sum_{l \in A'} S^{(m,l)} \right| - \left| \sum_{l \in A} S^{(m,l)} \right|$$  \hfill (C.42)

$$\leq \beta + \beta + (a - 2)\theta - \left| \sum_{l \in A} S^{(m,l)} \right|$$  \hfill (C.43)

where we use the induction hypothesis in (C.43). \hfill $\blacksquare$
Appendix D

Proof of Lemma 7.4.1

Proof To simplify the notation, let

\[ T = \sum_{i=2}^{n} \min(\alpha, \mu_i + \beta_i) \quad (D.1) \]

We will perform a series of transformations on \( \mu_i, \ i = 1, \ldots, n \) and \( \beta_i, \ i = 2, \ldots, n \) such that the value of \( T \) evaluated after step \( i \) is such that

\[ T^{(i)} \leq T^{(i-1)} \quad (D.2) \]

Let

\[ \mu_i^{(0)} = \mu_i, \quad \beta_i^{(0)} = \beta_i \quad (D.3) \]

yielding

\[ T^{(0)} = \sum_{i=2}^{n} \min(\alpha, \mu_i + \beta_i) \quad (D.4) \]

**Step1:** Choose \( \mu_1^{(1)} = \alpha \) with \( \mu_i \) and \( \beta_i, \quad (i = 2, \ldots, n) \) unaltered. Then we have for \( i = 2, \ldots, n \)

\[ \mu_i^{(1)} = \mu_i^{(0)}, \quad \beta_i^{(1)} = \beta_i^{(0)} \quad (D.5) \]

and thus

\[ T^{(1)} = T^{(0)} \quad (D.6) \]

**Step2:** Decrease the values of \( \mu_i^{(1)} \) and \( \beta_i^{(1)} \) \( (i = 2, \ldots, n) \) such that

\[ \sum_{i=2}^{n} \mu_i^{(2)} = B - \alpha \quad (D.7) \]

\[ \sum_{i=2}^{n} \beta_i^{(2)} = \gamma \quad (D.8) \]
Decreasing the values of $\mu_i$ and $\beta_i$ cannot increase $T$. Hence

$$T^{(2)} \leq T^{(1)} \quad \text{(D.9)}$$

**Step3:** Arrange $\mu_i^{(2)} + \beta_i^{(2)}$ ($i = 2, \ldots, n$) in decreasing order of their values. If $\mu_i^{(2)} + \beta_i^{(2)} \geq \mu_j^{(2)} + \beta_j^{(2)}$,

$$\begin{align*}
\mu_i^{(3)} &= \mu_i^{(2)} + \min(\alpha - \mu_i^{(2)}, \mu_j^{(2)}) \quad \text{(D.10)} \\
\mu_j^{(3)} &= \mu_j^{(2)} - \min(\alpha - \mu_i^{(2)}, \mu_j^{(2)}) \quad \text{(D.11)} \\
\beta_i^{(3)} &= \beta_i^{(2)} + \min(\beta_{\max} - \beta_i^{(2)}, \beta_j^{(2)}) \quad \text{(D.12)} \\
\beta_j^{(3)} &= \beta_j^{(2)} - \min(\beta_{\max} - \beta_i^{(2)}, \beta_j^{(2)}) \quad \text{(D.13)}
\end{align*}$$

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 $m$

\[
\mu_i^{(m)} = \begin{cases} 
\alpha & \text{if } i = 2, \ldots, s \\
B \mod \alpha & \text{if } i = s + 1 \\
0 & \text{otherwise}
\end{cases} \quad \text{(D.14)}
\]

\[
\beta_i^{(m)} = \begin{cases} 
\beta_{\max} & \text{if } i = 2, \ldots, \lfloor \gamma/\beta_{\max} \rfloor \\
\gamma \mod \beta_{\max} & \text{if } i = \lfloor \gamma/\beta_{\max} \rfloor + 1 \\
0 & \text{otherwise}
\end{cases} \quad \text{(D.15)}
\]

Note that this process will converge in a finite number of steps and also that

$$\lfloor \gamma/\beta_{\max} \rfloor \geq s \quad \text{(D.16)}$$
The value of $T$ after these transformations,

$$T^{(m)} = \sum_{i=2}^{n} \min(\alpha, \mu_i^{(m)} + \beta_i^{(m)})$$  \hspace{1cm} (D.17)

**Case 1:** $\alpha - \beta_{\text{max}} > B \mod \alpha$

As $\gamma$ meets the cut-set bound given in equation (7.17) with equality we have

$$\gamma = \alpha + (s - 1)\beta_{\text{max}}.$$  \hspace{1cm} (D.18)

Then we have

$$T^{(m)} = (s - 1)\alpha + B \mod \alpha + \beta_{\text{max}}$$  
$$+ \gamma - s\beta_{\text{max}}$$  
$$= (s - 1)\alpha + B \mod \alpha + \alpha$$  
$$= s\alpha + (B \mod \alpha)$$  
$$= B.$$  \hspace{1cm} (D.19)

**Case 2:** $\alpha - \beta_{\text{max}} \leq B \mod \alpha$

As $\gamma$ meets the cut-set bound given in equation (7.17) with equality we have

$$\gamma = \alpha + B \mod \alpha + s\beta_{\text{max}}.$$  \hspace{1cm} (D.20)

$$T^{(m)} = (s - 1)\alpha + \alpha + \gamma - s\beta_{\text{max}}$$  
$$= s\alpha + \alpha + B \mod \alpha$$  
$$\geq B.$$  \hspace{1cm} (D.21)

Thus,

$$T = T^{(0)} \geq T^{(m)} \geq B.$$