Interference Alignment in Regenerating Codes for Distributed Storage: Necessity and Code Constructions

Nihar B. Shah,* K. V. Rashmi,* P. Vijay Kumar, *Fellow, IEEE*, and Kannan Ramchandran, *Fellow, IEEE*

***Abstract*—Regenerating codes are a class of recently developed codes for distributed storage that, like Reed-Solomon codes, permit data recovery from any arbitrary k of n nodes. However regenerating codes possess in addition, the ability to repair a failed node by connecting to any arbitrary d nodes and downloading an amount of data that is typically far less than the size of the data file. This amount of download is termed the repair bandwidth. Minimum storage regenerating (MSR) codes are a subclass of regenerating codes that require the least amount of network storage; every such code is a maximum distance separable (MDS) code. Further, when a replacement node stores data identical to that in the failed node, the repair is termed as *exact*.**

The four principal results of the paper are (a) the explicit construction of a class of MDS codes for $d = n - 1 \geq 2k - 1$ termed the MISER code, that achieves the cut-set bound on the repair bandwidth for the exact repair of systematic nodes, (b) proof of the necessity of interference alignment in exact-repair MSR codes, (c) a proof showing the impossibility of constructing linear, exact-repair MSR codes for $d < 2k - 3$ in the absence of symbol extension, and (d) the construction, also explicit, of high-rate MSR codes for $d = k+1$. Interference alignment (IA) is a theme that runs throughout the paper: the MISER code is built on the principles of IA and IA is also a crucial component to the nonexistence proof for $d < 2k - 3$. To the best of our knowledge, the constructions presented in this paper are the first explicit constructions of regenerating codes that achieve the cut-set bound.

***Index Terms*—Distributed storage, interference alignment, maximum-distance-separable (MDS) regenerating codes, network coding, node repair, partial data recovery.**

* Equal contribution

Manuscript received August 31, 2010; revised June 03, 2011; accepted October 25, 2011. Date of current version March 13, 2012. This work was done while N. B. Shah and K. V. Rashmi were at the Department of Electrical and Computer Engineering, Indian Institute of Science, Bangalore, India. The material in this paper was presented in part at the Information Theory and Applications Workshop, San Diego, CA, 2010, in part at the 2010 IEEE Information Theory Workshop, and in part at the Allerton Conference on Communication, Control, and Computing, Monticello, IL, Oct. 2010.

N. B. Shah, K. V. Rashmi, and K. Ramchandran are with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94710 USA (e-mail: nihar@eecs.berkeley.edu; rashmikv@eecs.berkeley.edu; kannanr@eecs.berkeley.edu).

P. V. Kumar is with the Department of Electrical and Computer Engineering, Indian Institute of Science, Bangalore, India, and also with the Electrical Engineering Systems Department, University of Southern California, Los Angeles, CA USA 90089-2565 (e-mail: vijay@ece.iisc.ernet.in).

Communicated by C. Fragouli, Associate Editor for Communication Networks.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2011.2178588

I. Introduction

In a distributed storage system, a data file (the message) is dispersed across storage nodes in a network in such a manner that an end-user (whom we term as a data collector, or a DC) can retrieve the data file by tapping into one or more storage nodes. It is also desirable that a distributed storage system be reliable in the face of node failures. The simplest means of increasing reliability of a storage system is through replication, i.e., by storing identical copies of the message in multiple storage nodes. However, for a given level of reliability, such systems are inefficient in storage space utilization as compared to other approaches. A popular option that reduces storage space utilization and leads to increased resiliency 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 number of message symbols, over a finite field $\mathbb{F}_q$ of size q. An $[n, k]$ RS code encodes a message of size $B = k$ to obtain n symbols over $\mathbb{F}_q$, and stores one distinct coded symbol in each of the n nodes in the network. Under this encoding, the entire data can be recovered by a data collector by connecting to any arbitrary k nodes, a process of data recovery that we will refer to as *reconstruction*. Several distributed storage systems such as RAID-6, OceanStore [1], Total Recall [2] and Wuala [3] employ such an erasure-coding option.

Upon failure of an individual node, a self-sustaining data storage network must necessarily possess the ability to repair the 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. For example, RS codes treat the data stored in each node as a single symbol belonging to the finite field $\mathbb{F}_q$. When this is coupled with the restriction that individual nodes perform linear operations over $\mathbb{F}_q$, it follows that the smallest unit of data that can be downloaded from a node to assist in the repair of a failed node (namely, an $\mathbb{F}_q$ symbol), equals the amount of information stored in the node itself. As a consequence of the MDS property of an RS code, when carrying out repair of a failed node, the replacement node must necessarily collect data from at least k other nodes. As a result, it follows that the total amount of data download needed to repair a failed node can be no smaller than B, the size of the entire message. But clearly, downloading entire B units of data in order to recover the data stored in a single node that stores only a fraction $(1/k)$ of the entire data is

0018-9448/$26.00 © 2011 IEEE

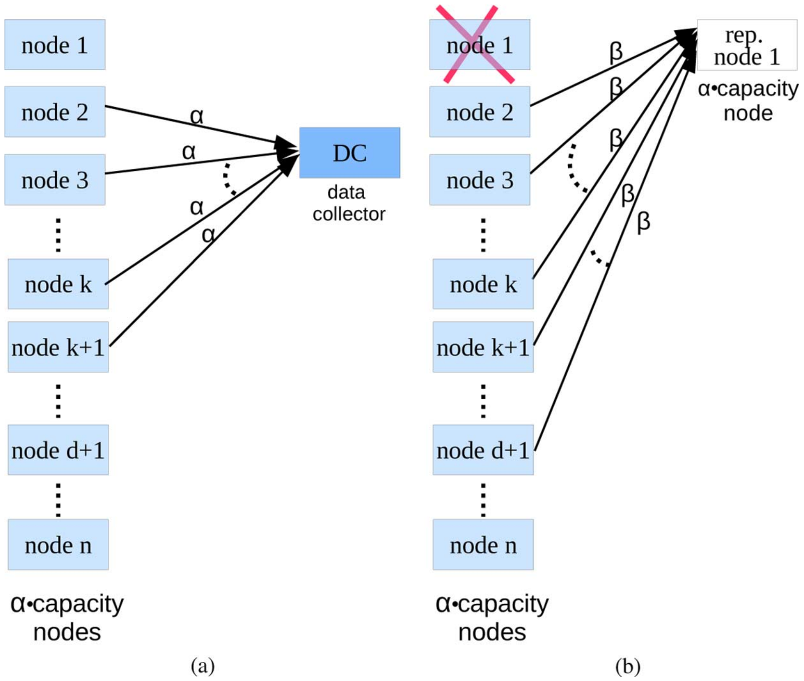


Fig. 1. Regenerating codes setup. (a) Data reconstruction and (b) repair of a failed node.

wasteful. This naturally raises the question as to whether there is a better option. Such an option is provided by the concept of a *regenerating code* introduced by Dimakis *et al.* [4], [5].

Regenerating codes overcome the difficulty encountered when working with an RS code by working with codes whose symbol alphabet is a vector over $\mathbb{F}_q$, i.e., an element of $\mathbb{F}_q^\alpha$ for some parameter $\alpha > 1$. Each node stores a vector symbol, or equivalently stores α symbols over $\mathbb{F}_q$. In this setup, it is clear that while maintaining linearity over $\mathbb{F}_q$, it is possible for an individual node to transfer a fraction of the data stored within the node.

Apart from this new parameter α, two other parameters, d and β, are associated with regenerating codes. Under the definition of regenerating codes introduced in [4], a failed node is permitted to connect to an arbitrary subset of d nodes out of the remaining $(n-1)$ nodes while downloading $\beta \leq \alpha$ symbols from each node. The total amount $d\beta$ of data downloaded for repair purposes is termed the *repair bandwidth*. Typically, with a regenerating code, the average repair bandwidth $d\beta$ is small compared to the size of the message B. Thus we have

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

as the parameter set of a regenerating code. Fig. 1(a) and (b) illustrate reconstruction and node repair respectively, also depicting the relevant parameters.

The cut-set bound of network coding can be invoked to show that the parameters of a regenerating code must necessarily satisfy [6]:

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

Since both storage and bandwidth come at a cost, it is naturally desirable to minimize both α as well as β. However, it can be deduced (see [6]) that achieving equality in (1), for fixed values of B and $[n,\ k,\ d]$, leads to a tradeoff between the storage space α and the repair bandwidth $d\beta$. The two extreme points in this tradeoff are termed the minimum storage regenerating (MSR) and minimum bandwidth regenerating (MBR) points. The parameters α and β for the MSR point on the tradeoff can be obtained by first minimizing α and then minimizing β to obtain

$$\begin{aligned} \alpha_{\text{MSR}} &= \frac{B}{k} \\ \beta_{\text{MSR}} &= \frac{B}{k(d-k+1)}. \end{aligned} \tag{2}$$

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

$$\begin{aligned} \beta_{\text{MBR}} &= \frac{2B}{k(2d-k+1)} \\ \alpha_{\text{MBR}} &= \frac{2dB}{k(2d-k+1)}. \end{aligned} \tag{3}$$

The focus of the present paper is on the MSR point. 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 arbitrary k nodes necessarily implies a minimum distance of $(n-k+1)$. Since the code size equals $(q^\alpha)^k$, this meets the Singleton bound causing the code to be an MDS code.

A. Choice of the Parameter β (Striping)

Let us next rewrite (2) in the form

$$\begin{aligned} \alpha_{\text{MSR}} &= \beta_{\text{MSR}}(d-k+1) \\ B &= \beta_{\text{MSR}}(d-k+1)(k). \end{aligned} \tag{4}$$

Thus if one is able to construct an $[n,\ k,\ d]$ MSR code with repair bandwidth achieving the cut-set bound for a given value of β, then both $\alpha_{\text{MSR}} = (d-k+1)\beta_{\text{MSR}}$ and the size $B = k\,\alpha_{\text{MSR}}$ of the message are necessarily fixed. It thus makes sense to speak of an achievable triple

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

However if a triple (β, α, B) is achievable, then so is the triple $(\ell\beta, \ell\alpha, \ell B)$ simply through a process of divide and conquer, i.e., we divide up the message into ℓ sub-messages and apply the code for (β, α, B) to each of the ℓ sub-messages. It follows that if one can construct an (optimal) $[n,\ k,\ d]$ MSR code with $\beta = 1$, then one can construct an (optimal) $[n,\ k,\ d]$ MSR code for any larger value of β via a concatenation of the $\beta = 1$ code. This motivates us to consider the case of $\beta = 1$ as the first step towards design of exact-repair codes. In addition, a code with a small value of β will involve manipulating a smaller number of message symbols, and may also lead to algorithms of lesser complexity. For these reasons, in the present paper, codes are constructed for the case $\beta = 1$. In the terminology of distributed storage, such a process is called *striping*. Setting $\beta = 1$ at the MSR point yields

$$\alpha_{\text{MSR}} = d - k + 1. \tag{5}$$

Note that when $\alpha = 1$, we have $B = k$ and meeting the cut-set bound would imply $d = k$. In this case, any $[n, k]$-MDS code will achieve the bound. Hence, we will consider $\alpha > 1$ throughout.

B. *Additional Terminology*

1) Exact Versus Functional Repair: The cut-set bound (1) is derived in [6] for the case when failed nodes undergo "functional repair." Under functional repair, a failed node f is replaced by a new node f' such that following replacement, the resulting system continues to possess the data-reconstruction and repair properties. In contrast, under exact repair, a failed node f is replaced by a new node f' which stores exactly the same data as was stored in node f prior to failure. We will use the term exact-repair MSR code to denote a regenerating code operating at the minimum storage point, that is capable of exact repair.

Exact repair is to be preferred over functional repair wherever possible, due to the following reasons. In a system where the code coefficients are globally known, under functional repair there is need for the network to inform all nodes of the new code coefficients at the replacement node. Moreover, the repair and decoding algorithms also need to be re-tuned for the new set of coefficients. These additional overheads are clearly unnecessary under exact repair. In addition, exact repair permits the code to be systematic, as described below.

2) Systematic 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 repair of the systematic nodes is mandated. A data collector connecting to the k systematic nodes obtains the B message symbols in an uncoded form, making systematic nodes a preferred choice for data recovery. This makes the fast repair of systematic nodes a priority, motivating the interest in minimizing the repair bandwidth for the exact repair of systematic nodes.

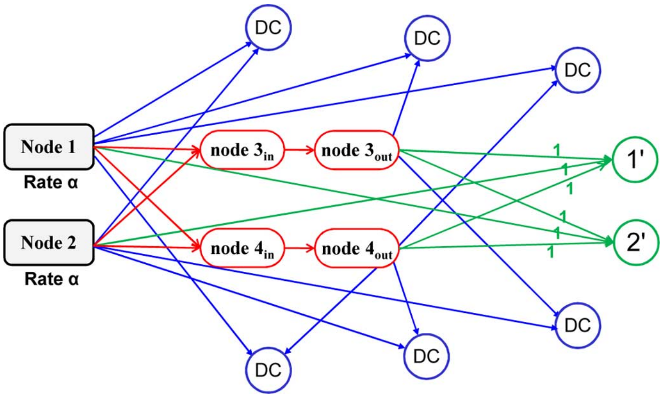


Fig. 2. MSR code design problem for the exact repair of just the systematic nodes, as a non-multicast network coding problem. Here, $[n = 4,\ k = 2,\ d = 3]$ with $\beta = 1$ giving $(\alpha = 2,\ B = 4)$. Unmarked edges have capacity α. Nodes labeled *DC* are data-collector sinks, and those labeled $1'$ and $2'$ are replacement node sinks.

As mentioned above, the cut-set bound (as derived in [6]) applies to functional repair. 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 repair of the systematic nodes leads to a bound on the parameters $(\beta,\ \alpha,\ B)$ different from the cut-set bound. It turns out that the same bound on the parameters $(\beta,\ \alpha,\ B)$ appearing in (2) still applies and this is established in Section III.

C. *Exact-Repair MSR Codes as Network Codes*

The existence of regenerating codes for the case of functional repair was proved [4], [6] after casting the reconstruction and repair problems as a multicast network coding problem, and using random network codes to achieve the cut-set bound. As shown in [7], construction of exact-repair MSR codes for the repair of systematic nodes is most naturally mapped to a non-multicast problem in network coding, for which very few results are available.

The non-multicast network for the parameter set $[n = 4,\ k = 2,\ d = 3]$ with $\beta = 1$ is shown in Fig. 2. In general, the network can be viewed as having k source nodes, corresponding to the k systematic nodes, generating α symbols each per channel use. The parity nodes correspond to downlink nodes in the graph. To capture the fact that a parity node can store only α symbols, it is split (as in [6]) into two parts connected by a link of capacity α: parity node m is split into m_{in} and m_{out} with all incoming edges arriving at m_{in} and all outgoing edges emanating from m_{out}.

The sinks in the network are of two types. The first type correspond to data collectors which connect to an arbitrary 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 represent a replacement node that is attempting to duplicate a failed systematic node, with the node replacing systematic node ℓ denoted by ℓ'. Sinks of this type connect to an arbitrary set of d out of the remaining $(n-1)$ nodes, and hence they are $k\binom{n-1}{d}$ in number. It is the presence of these sinks that gives the problem a non-multicast nature.

The MISER code construction presented in Section V of the present paper is an explicit MSR code performing exact repair of systematic nodes. Hence, this code serves as an instance where an explicit code construction achieves the cut-set bound for a non-multicast network, by exploiting the specific structure of the network.

Relation Between β and Scalar/Vector Network Coding: The choice of β as unity (as in Fig. 2) may be viewed as an instance of scalar network coding. Upon increase in the value of β, the capacity of each data pipe is increased by a factor of β, thereby transforming the problem into a vector network coding problem. Thus, $\beta = 1$ implies the absence of *symbol extension*, which may reduce the complexity of system implementation and is thus of greater practical interest.

D. Results of the Present Paper

The primary results of the present paper are:

- The construction of a family of MDS codes for $d = n-1 \geq 2k-1$ that enable exact repair of systematic nodes while achieving the cut-set bound on repair bandwidth. We have termed this code the MISER[1] code.
- Proof that interference alignment is *necessary* for every exact-repair MSR code.
- The proof of nonexistence of linear exact-repair MSR codes for $d < 2k-3$ in the absence of symbol extension (i.e., $\beta = 1$). This result is clearly of interest in the light of ongoing efforts to construct exact-repair codes with $\beta = 1$ meeting the cut-set bound [8]–[17].
- The construction, also explicit, of an MSR code for $d = k+1$. For most values of the parameters, $d = k+1$ falls under the $d < 2k-3$ regime, and in light of the nonexistence result above, exact repair is not possible. The construction does the next best thing, namely, it carries out repair that is approximately-exact.[2]
- A shortening technique for constructing high-redundancy MSR codes from low-redundancy MSR codes, i.e., constructing an $[n,\ k,\ d]$ MSR code from any $[n+i,\ k+i,\ d+i]$ MSR code.

Note that the only explicit regenerating codes of the MDS type to previously have been constructed are for certain specific values of parameters, $[n = 4,\ k = 2,\ d = 3]$ and $[n = 5,\ k = 3,\ d = 4]$. Prior work is described in greater detail in Section II.

The repair algorithm of the MISER code also minimizes the number of disk reads and computations required at the helper nodes during repair. Moreover, in a work ([14]) following the initial submission of the MISER code ([10]), Suh and Ramchandran show that MISER code can also perform exact repair of parity nodes optimally, and provide explicit mechanisms for the same.

The remainder of the paper is organized as follows. A brief overview of the prior literature in this field is given in the next section, Section II. The setting and notation are explained in Section III. The appearance of interference alignment in the context of distributed storage for construction of regenerating codes is detailed in Section IV along with an illustrative example. Section V describes the MISER code. A shortening technique for MSR codes is also provided in this section. The nonexistence of linear exact-repair MSR codes for $d < 2k-3$ in the absence of symbol extension can be found in Section VI, along with the proof establishing the necessity of interference alignment. Section VII describes the explicit construction of an MSR code for $d = k+1$. The final section, Section VIII, draws conclusions.

II. Related Work

The concept of regenerating codes, introduced in [4], [6], permit storage nodes to store more than the minimal B/k units of data in order to reduce the repair bandwidth. Several distributed systems are analysed, and estimates of the mean node availability in such systems are obtained. Using these values, the substantial performance gains offered by regenerating codes in terms of bandwidth savings are demonstrated.

The problem of minimizing repair bandwidth for the *functional* repair of nodes is considered in [4], [6] where it is formulated as a multicast network-coding problem in a network having an infinite number of nodes. A cut-set lower bound on the repair bandwidth is derived. Coding schemes achieving this bound are presented in [6], [12] which however, are nonexplicit. These schemes require large field size and the repair and reconstruction algorithms are also of high complexity.

Computational complexity is identified as a principal concern in the practical implementation of distributed storage codes in [18] and a treatment of the use of random, linear, regenerating codes for achieving functional repair can be found there.

The notion of exact repair is independently introduced in [8] and [9]. The idea of using interference alignment in the context of exact-repair codes for distributed storage appears first in [8]. Code constructions of the MDS type are provided, which meet the cut-set lower bound when $k = 2$. Even here, the constructions are not explicit, and have large complexity and field-size requirement.

The first explicit construction of regenerating codes for the MBR point appears in [9], for the case $d = n-1$. These codes carry out repair-by-transfer (uncoded exact-repair) and hence have zero repair complexity. The required field size is of the order of n^2, and in terms of minimizing bandwidth, the codes achieve the cut-set bound.

A computer search for exact-repair MSR codes for the parameter set $[n = 5,\ k = 3,\ d = 4],\ \ \beta = 1$, is carried out in [11], and for this set of parameters, codes for several values of field size are obtained.

[1] Short for an MDS, Interference-aligning, Systematic, Exact-Regenerating code, that is miserly in terms of bandwidth expended to repair a systematic node.

[2] The code consists of an exact part which is exactly repaired, along with an auxiliary part which is only *functionally* repaired. This is explained in greater detail in Section VII.

A slightly different setting, from the exact-repair situation is considered in [12], where optimal MDS codes are given for the parameters $d = k + 1$ and $n > 2k$. Again, the schemes given here are nonexplicit, and have high complexity and large field-size requirement.

The bound in (1), which is known to be tight for functional repair, also trivially turns out to be a bound for the case of exact repair. However, it is not known whether the bound is tight under this setting. After the initial submission of the present paper, some new results have shed more light on this question. In the limiting case of B (and hence α and β) approaching infinity, the MSR point is shown to be achievable under exact repair for all $[n,\ k,\ d]$ in [19] and [20]. On the other hand, it is shown in [17] that essentially all interior points on the tradeoff are not achievable under exact repair.

We next describe the setting and notation to be used in the present paper.

III. Setting and Notation

The distributed storage system considered in this paper consists of n storage nodes, each having the capacity to store α symbols. Let $\underline{\mathbf{u}}$ be the message vector of length B comprising the B message symbols. Each message symbol can independently take values from $\mathbb{F}_q$, a finite field of size q.

In this paper, we consider only linear storage codes. As in traditional coding theory, by a linear storage code, we mean that every stored symbol is a linear combination of the message symbols, and only linear operations are permitted on the stored symbols. Thus all symbols considered belong to $\mathbb{F}_q$.

For $m = 1, \ldots, n$, let the $(B \times \alpha)$ matrix $\mathbf{G}^{(m)}$ denote the generator matrix of node m. Node m stores the following α symbols

$$\underline{\mathbf{u}}^t \mathbf{G}^{(m)}. \tag{6}$$

In the terminology of network coding, each column of the nodal generator matrix $\mathbf{G}^{(m)}$ corresponds to the *global kernel* (linear combination vector) associated with a symbol stored in the node. The $(B \times n\alpha)$ generator matrix for the entire distributed-storage code, is given by

$$\mathbb{G} = [\,\mathbf{G}^{(1)} \quad \mathbf{G}^{(2)} \quad \cdots \quad \mathbf{G}^{(n)}\,]. \tag{7}$$

Note that under exact repair, the generator matrix of the code remains unchanged.

We will interchangeably speak of a node as either storing α symbols, by which we will mean the symbols $\underline{\mathbf{u}}^t \mathbf{G}^{(m)}$ or else as storing α vectors, by which we will mean the corresponding set of α global kernels that form the columns of nodal generator matrix $\mathbf{G}^{(m)}$.

We partition the $B(= k\alpha)$-length vector $\underline{\mathbf{u}}$ into k components, $\underline{u}_i$ for $i = 1, \ldots, k$, each comprising α distinct message symbols:

$$\underline{\mathbf{u}} = \begin{bmatrix} \underline{u}_1 \\ \vdots \\ \underline{u}_k \end{bmatrix}. \tag{8}$$

We also partition the nodal generator matrices analogously into k submatrices as

$$\mathbf{G}^{(m)} = \begin{bmatrix} G_1^{(m)} \\ \vdots \\ G_k^{(m)} \end{bmatrix} \tag{9}$$

where each $G_i^{(m)}$ is an $(\alpha \times \alpha)$ matrix. We will refer to $G_i^{(m)}$ as the ith component of $\mathbf{G}^{(m)}$. Thus, node m stores the α symbols

$$\underline{\mathbf{u}}^t \mathbf{G}^{(m)} = \sum_{i=1}^{k} \underline{u}_i^t G_i^{(m)}. \tag{10}$$

Out of the n nodes, the first k nodes (i.e., nodes $1, \ldots, k$) are systematic. Thus, for systematic node ℓ

$$G_i^{(\ell)} = \begin{cases} I_\alpha & \text{if } i = \ell \\ 0_\alpha & \text{if } i \neq \ell \end{cases}, \quad \forall i \in \{1, \ldots, k\} \tag{11}$$

where 0_α and I_α denote the $(\alpha \times \alpha)$ zero matrix and identity matrix respectively; systematic node ℓ $(1 \leq \ell \leq k)$ thus stores the α message symbols that $\underline{u}_\ell$ is comprised of.

Upon failure of a node, the replacement node connects to an arbitrary set of d remaining nodes, termed as *helper nodes*, downloading β symbols from each. Thus, each helper node passes a collection of β linear combinations of the symbols stored within the node. As described in Section I-A, an MSR code with $\beta = 1$ can be used to construct an MSR code for every higher integral value of β. Thus it suffices to provide constructions for $\beta = 1$ and that is what we do here. When $\beta = 1$, each helper node passes just a single symbol. Again, we will often describe the symbol passed by a helper node in terms of its associated global kernel, and hence will often speak of a helper node passing a *vector*.[3]

Throughout the paper, we use superscripts to refer to node indices, and subscripts to index the elements of a matrix. The letters m and ℓ are reserved for node indices; in particular, the letter ℓ is used to index systematic nodes. All vectors are assumed to be column vectors. The vector $\underline{e}_i$ represents the standard basis vector of length α, i.e., $\underline{e}_i$ is an α-length unit vector with 1 in the ith position and 0 s elsewhere. For a positive integer p, we denote the $(p \times p)$ zero matrix and the $(p \times p)$ identity matrix by 0_p and I_p respectively. For a τ-length vector $\underline{v}$, we use the notation $\text{diag}[\,\underline{v}^t\,]$ to denote the $(\tau \times \tau)$ diagonal matrix with elements of the vector $\underline{v}$ as the elements of its main diagonal. We say that a set of vectors is *aligned* if the vector-space spanned by them has dimension at most one.

We next turn our attention to the question as to whether or not the combination of (a) restriction to systematic-node repair and (b) requirement of exact repair of the systematic nodes leads to a bound on the parameters $(\beta,\ \alpha,\ B)$ different from the cut-set bound appearing in (1).

The theorem below shows that the bound in (1) for the MSR point continues to hold even if *functional repair* of a *single* node is required.

[3] A simple extension to the case of $\beta > 1$ lets us treat the global kernels of the β symbols passed by a helper node as a *subspace* of dimension at most β. This "subspace" viewpoint has been found useful in proving certain general results at the MBR point in [9], and for the interior points of the tradeoff in [13].

Theorem 1: Any $[n,\ k,\ d]$-MDS regenerating code (i.e., a regenerating code satisfying $B = k\alpha$) that guarantees the functional repair of even a single node, must satisfy the cut-set lower bound of (1), i.e., must satisfy

$$\beta \geq \frac{B}{k(d-k+1)}. \tag{12}$$

Proof: First, consider the case when $\beta = 1$. Let ℓ denote the node that needs to be repaired, and let $\{m_i \mid 1 \leq i \leq d\}$ denote the d helper nodes assisting in the repair of node ℓ. Further, let $\{\underline{\boldsymbol{\gamma}}^{(m_i,\ \ell)} \mid 1 \leq i \leq d\}$ denote the vectors passed by these helper nodes. At the end of the repair process, let the $(B \times \alpha)$ matrix $\mathbf{G}^{(\ell)}$ denote the generator matrix of the replacement node (since we consider only functional repair in this theorem, $\mathbf{G}^{(\ell)}$ need not be identical to the generator matrix of the failed node).

Looking back at the repair process, the replacement node obtains $\mathbf{G}^{(\ell)}$ by operating linearly on the collection of d vectors $\{\underline{\boldsymbol{\gamma}}^{(m_i,\ \ell)} \mid 1 \leq i \leq d\}$ of length B. This, in turn, implies that the dimension of the nullspace of the matrix

$$[\,\mathbf{G}^{(\ell)} \quad \underline{\boldsymbol{\gamma}}^{(m_1,\ \ell)} \quad \cdots \quad \underline{\boldsymbol{\gamma}}^{(m_d,\ \ell)}\,] \tag{13}$$

should be greater than or equal to the dimension of $\mathbf{G}^{(\ell)}$, which is α. However, the MDS property requires that at the end of the repair process, the global kernels associated with any k nodes be linearly independent, and in particular, that the matrix

$$[\,\mathbf{G}^{(\ell)} \quad \underline{\boldsymbol{\gamma}}^{(m_1,\ \ell)} \quad \cdots \quad \underline{\boldsymbol{\gamma}}^{(m_{k-1},\ \ell)}\,] \tag{14}$$

have full-rank. It follows that we must have

$$d \geq k - 1 + \alpha. \tag{15}$$

The proof for the case $\beta > 1$, when every helper node passes a set of β vectors, is a straightforward extension that leads to:

$$d\beta \geq (k-1)\beta + \alpha. \tag{16}$$

Rearranging the terms in the inequality above, and substituting $\alpha = \frac{B}{k}$ leads to the desired result. ■

Since exact repair is a special case of functional repair, the result in Theorem 1 trivially holds for exact repair as well. Thus, we recover (2), and in an optimal code with $\beta = 1$, we will continue to have

$$d = k - 1 + \alpha.$$

In this way, we have shown that even in the setting that we address here, namely that of the exact repair of the systematic nodes, leads us to the same bound on repair bandwidth as in (1).

Remark 1: The result in Theorem 1 can also be derived information-theoretically (see [17] for an information-theoretic perspective on regenerating codes), and hence holds for nonlinear codes as well. However, since the present paper deals only with linear codes, we restrict our attention to this case.

The next section explains how the concept of interference alignment arises in the context of distributed storage.

IV. Interference Alignment in Regenerating Codes

The idea of interference alignment has recently been proposed in [21] and [22] in the context of wireless communication. The idea here is to design the signals of multiple users in such a way that at every receiver, signals from all the unintended users occupy a subspace of the given space, leaving the remainder of the space free for the signal of the intended user.

In the distributed-storage context, the concept of "interference" comes into play during the exact repair of a failed node in an MSR code. We present the example of a systematic MSR code with $[n = 4,\ k = 2,\ d = 3]$ and $\beta = 1$, which gives $(\alpha = d - k + 1 = 2,\ B = k\alpha = 4)$. Let $\{u_1,\ u_2,\ u_3,\ u_4\}$ denote the four message symbols. Since $k = 2$ here, we may assume that nodes 1 and 2 are systematic and that node 1 stores $\{u_1,\ u_2\}$ and node 2 stores $\{u_3,\ u_4\}$. Nodes 3 and 4 are then the parity nodes, each storing two linear functions of the message symbols.

Consider repair of systematic node 1 wherein the $d = 3$ nodes, nodes 2, 3 and 4 serve as helper nodes. The second systematic node, node 2, can only pass a linear combination of message symbols u_3 and u_4. The two symbols passed by the parity nodes are in general, functions of all four message symbols: $(a_1u_1 + a_2u_2 + a_3u_3 + a_4u_4)$ and $(b_1u_1 + b_2u_2 + b_3u_3 + b_4u_4)$ respectively.

Using the symbols passed by the three helper nodes, the replacement of node 1 needs to be able to recover message symbols $\{u_1, u_2\}$. For obvious reasons, we will term $(a_1u_1 + a_2u_2)$ and $(b_1u_1 + b_2u_2)$ as the *desired* components of the messages passed by parity nodes 3 and 4 and the terms $(a_3u_3 + a_4u_4)$ and $(b_3u_3 + b_4u_4)$ as *interference* components.

Since node 2 cannot provide any information pertaining to the desired symbols $\{u_1,\ u_2\}$, the replacement node must be able to recover the desired symbols from the desired components $(a_1u_1 + a_2u_2)$ and $(b_1u_1 + b_2u_2)$ of the messages passed to it by the parity nodes 3 and 4. To access the desired components, the replacement node must be in a position to subtract out the interference components $(a_3u_3 + a_4u_4)$ and $(b_3u_3 + b_4u_4)$ from the received linear combinations $(a_1u_1 + a_2u_2 + a_3u_3 + a_4u_4)$ and $(b_1u_1 + b_2u_2 + b_3u_3 + b_4u_4)$; the only way to subtract out the interference component is by making use of the linear combination of $\{u_3, u_4\}$ passed by node 2. It follows that this can only happen if the interference components $(a_3u_3 + a_4u_4)$ and $(b_3u_3 + b_4u_4)$ are aligned, meaning that they are scalar multiples of each other.

An explicit code over $\mathbb{F}_5$ for the parameters chosen in the example is shown in Fig. 3. The exact repair of systematic node 1 is shown, for which the remaining nodes pass the first of the two symbols stored in them. Observe that under this code, the interference component in the two symbols passed by the parity nodes are aligned in the direction of u_3, i.e., are scalar multiples of u_3. Hence node 2 can simply pass u_3 and the replacement node can then make use of u_3 to cancel (i.e., subtract out) the interference.

In the context of regenerating codes, interference alignment was first used by Wu *et al.* [8] to provide a scheme (although, not explicit) for the exact repair at the MSR point. However, interference alignment is employed only to a limited extent as only a portion of the interference components is aligned and as a result, the scheme is optimal only for the case $k = 2$.

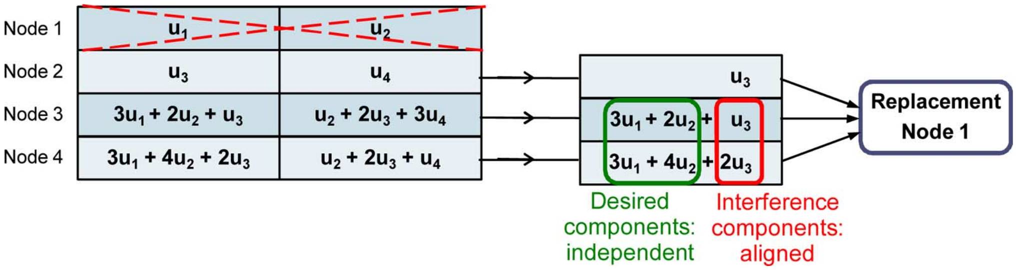


Fig. 3. Illustration of interference alignment during exact repair of systematic node 1.

In Section V, we describe the construction of the MISER code which aligns interference and achieves the cut-set bound on the repair bandwidth for repair of systematic nodes for all parameters satisfying $d = n - 1 \geq 2k - 1$. This is the *first* interference-alignment-based explicit code construction that meets the cut-set bound.

V. Construction of the MISER Code

In this section we provide an explicit construction for a systematic, MDS code that achieves the lower bound on repair bandwidth for the exact repair of systematic nodes and which we term as the MISER code. Under this code, the parameter d takes the largest permissible value of $n-1$, resulting in the lowest possible repair bandwidth for any MDS code.[4] This choice of the parameter d also allows for a greater degree of parallelization in the process of data download during repair.

We begin with an illustrative example that explains the key ideas behind the construction. The general code construction for parameter sets of the form $n = 2k$, $d = n - 1$ closely follows the construction in the example. A code-shortening technique for MSR codes is then described, and employed to extend this code construction to the more general parameter set $n \geq 2k$, $d = n - 1$.

The construction technique can also be extended to the even more general case of arbitrary n, $d \geq 2k - 1$, under the added requirement however, that the replacement node connect to all of the remaining systematic nodes.

A. An Example

The example deals with the parameter set, $[n = 6,\ k = 3,\ d = 5]$, $\beta = 1$, so that $(\alpha = d - k + 1 = 3,\ B = k\alpha = 9)$. We select $\mathbb{F}_7$ as the underlying finite field so that all message and code symbols are drawn from $\mathbb{F}_7$. Note that we have $\alpha = k = 3$ here. This is true in general: whenever $n = 2k$ and $d = n - 1$, we have $\alpha = d - k + 1 = k$ which simplifies the task of code construction.

1) Design of Nodal Generator Matrices: As $k = 3$, the first three nodes are systematic and store data in uncoded form. Hence

$$\mathbf{G}^{(1)} = \begin{bmatrix} I_3 \\ 0_3 \\ 0_3 \end{bmatrix} \quad \mathbf{G}^{(2)} = \begin{bmatrix} 0_3 \\ I_3 \\ 0_3 \end{bmatrix} \quad \mathbf{G}^{(3)} = \begin{bmatrix} 0_3 \\ 0_3 \\ I_3 \end{bmatrix}. \tag{17}$$

A key ingredient of the code construction presented here is the use of a Cauchy matrix [23]. Let

$$\Psi_3 = \begin{bmatrix} \psi_1^{(4)} & \psi_1^{(5)} & \psi_1^{(6)} \\ \psi_2^{(4)} & \psi_2^{(5)} & \psi_2^{(6)} \\ \psi_3^{(4)} & \psi_3^{(5)} & \psi_3^{(6)} \end{bmatrix} \tag{18}$$

be a (3×3) matrix such that each of its submatrices is full rank. Cauchy matrices have this property and in our construction, we will assume Ψ_3 to be a Cauchy matrix.

We choose the generator matrix of parity node m $(m = 4, 5, 6)$ to be

$$\mathbf{G}^{(m)} = \left[\begin{array}{ccc} 2\psi_1^{(m)} & 0 & 0 \\ 2\psi_2^{(m)} & \psi_1^{(m)} & 0 \\ 2\psi_3^{(m)} & 0 & \psi_1^{(m)} \\ \hline \psi_2^{(m)} & 2\psi_1^{(m)} & 0 \\ 0 & 2\psi_2^{(m)} & 0 \\ 0 & 2\psi_3^{(m)} & \psi_2^{(m)} \\ \hline \psi_3^{(m)} & 0 & 2\psi_1^{(m)} \\ 0 & \psi_3^{(m)} & 2\psi_2^{(m)} \\ 0 & 0 & 2\psi_3^{(m)} \end{array}\right] \tag{19}$$

where the location of the nonzero entries in the ith component are restricted to lie either along the diagonal or else within the ith column. The generator matrix is designed keeping in mind the need for interference alignment and this will be made clear in the discussion below concerning the exact repair of systematic nodes. The choice of scalar "2" plays an important role in the data reconstruction property; the precise role of this scalar will become clear when this property is discussed. An example of the $[6,\ 3,\ 5]$ MISER code over $\mathbb{F}_7$ is provided in Fig. 4, where the Cauchy matrix Ψ is chosen as

$$\Psi = \begin{bmatrix} 5 & 4 & 1 \\ 2 & 5 & 4 \\ 3 & 2 & 5 \end{bmatrix}. \tag{20}$$

Also depicted in the figure is the exact repair of node 1, for which each of the remaining nodes pass the first symbol that they store. It can be seen that the first symbols stored in the

[4] It can be inferred from the storage-bandwidth tradeoff (1) that for fixed values of the parameters $n,\ k,\ \alpha$ and B, the repair bandwidth $d\beta$ decreases with increase in d.

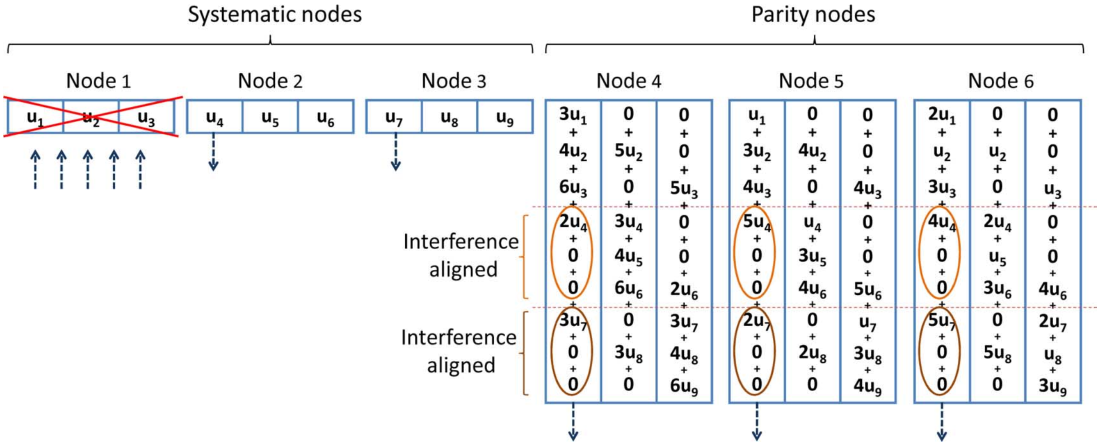


Fig. 4. Example of the $[6,\ 3,\ 5]$ MISER code over $\mathbb{F}_7$. Here, $\{u_1, \ldots, u_9\}$ denote the message symbols and the code symbols stored in each of the nodes are shown. Exact repair of node 1 is also depicted.

three parity nodes 4, 5, and 6 have their interference components (components 2 and 3) aligned and their desired components (component 1) linearly independent.

The key properties of the MISER code will now be established:

- that the code is an MDS code over alphabet $\mathbb{F}_q^\alpha$ and this property enables data reconstruction and
- that the code has the ability to carry out exact repair of the systematic nodes while achieving the cut-set bound on repair bandwidth.

We begin by establishing the exact-repair property.

2) Exact Repair of Systematic Nodes: Our algorithm for systematic node repair is simple. As noted above, each node stores $\alpha = k$ symbols. These k symbols are assumed to be ordered so that we may speak of the first symbol stored by a node, etc. To repair systematic node ℓ, $1 \leq \ell \leq k$, each of the remaining nodes passes their respective ℓth symbol.

Suppose in our example construction here, node 1 fails. Each of the parity nodes then pass on their first symbol, or equivalently, in terms of global kernels, the first column of their generator matrices for the repair of node 1. Thus, from nodes 4, 5, and 6, the replacement node obtains

$$\left[\begin{array}{c} 2\psi_1^{(4)} \\ 2\psi_2^{(4)} \\ 2\psi_3^{(4)} \\ \hline \psi_2^{(4)} \\ 0 \\ 0 \\ \hline \psi_3^{(4)} \\ 0 \\ 0 \end{array}\right], \quad \left[\begin{array}{c} 2\psi_1^{(5)} \\ 2\psi_2^{(5)} \\ 2\psi_3^{(5)} \\ \hline \psi_2^{(5)} \\ 0 \\ 0 \\ \hline \psi_3^{(5)} \\ 0 \\ 0 \end{array}\right], \quad \left[\begin{array}{c} 2\psi_1^{(6)} \\ 2\psi_2^{(6)} \\ 2\psi_3^{(6)} \\ \hline \psi_2^{(6)} \\ 0 \\ 0 \\ \hline \psi_3^{(6)} \\ 0 \\ 0 \end{array}\right]. \tag{21}$$

Note that in each of these vectors, the desired (first) components are a scaled version of the respective columns of the Cauchy matrix Ψ_3. The interference (second and third) components are aligned along the vector $[1\ \ 0\ \ 0]^t$. Thus, each interference component is aligned along a single dimension. Systematic nodes 2 and 3 then pass a single vector each that is designed to cancel out this interference. Specifically, nodes 2 and 3 respectively pass the vectors

$$\left[\begin{array}{c} 0 \\ 0 \\ 0 \\ \hline 1 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \end{array}\right], \quad \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ \hline 1 \\ 0 \\ 0 \end{array}\right]. \tag{22}$$

The net result is that after interference cancellation has taken place, replacement node 1 is left with access to the columns of the matrix

$$\left[\begin{array}{c} 2\Psi_3 \\ \hline 0_3 \\ \hline 0_3 \end{array}\right].$$

Thus the desired component is a scaled Cauchy matrix Ψ_3. By multiplying this matrix on the right by $\frac{1}{2}\Psi_3^{-1}$, one recovers

$$\left[\begin{array}{c} I_3 \\ \hline 0_3 \\ \hline 0_3 \end{array}\right]$$

as desired.

Along similar lines, when nodes 2 or 3 fail, the parity nodes pass the second or third columns of their generator matrices respectively. The design of generator matrices for the parity nodes is such that interference alignment holds during the repair of any systematic node, hence enabling the exact repair of all the systematic nodes.

3) Data Reconstruction (MDS Property): For the reconstruction property to be satisfied, a data collector downloading symbols stored in any three nodes should be able to recover all the

$$C_2 = \left[\begin{array}{ccc|ccc|ccc}
2\psi_1^{(4)} & 2\psi_1^{(5)} & 2\psi_1^{(6)} & 0 & 0 & 0 & 0 & 0 & 0 \\
2\psi_2^{(4)} & 2\psi_2^{(5)} & 2\psi_2^{(6)} & \psi_1^{(4)} & \psi_1^{(5)} & \psi_1^{(6)} & 0 & 0 & 0 \\
2\psi_3^{(4)} & 2\psi_3^{(5)} & 2\psi_3^{(6)} & 0 & 0 & 0 & \psi_1^{(4)} & \psi_1^{(5)} & \psi_1^{(6)} \\
\hline
\psi_2^{(4)} & \psi_2^{(5)} & \psi_2^{(6)} & 2\psi_1^{(4)} & 2\psi_1^{(5)} & 2\psi_1^{(6)} & 0 & 0 & 0 \\
0 & 0 & 0 & 2\psi_2^{(4)} & 2\psi_2^{(5)} & 2\psi_2^{(6)} & 0 & 0 & 0 \\
0 & 0 & 0 & 2\psi_3^{(4)} & 2\psi_3^{(5)} & 2\psi_3^{(6)} & \psi_2^{(4)} & \psi_2^{(5)} & \psi_2^{(6)} \\
\hline
\psi_3^{(4)} & \psi_3^{(5)} & \psi_3^{(6)} & 0 & 0 & 0 & 2\psi_1^{(4)} & 2\psi_1^{(5)} & 2\psi_1^{(6)} \\
0 & 0 & 0 & \psi_3^{(4)} & \psi_3^{(5)} & \psi_3^{(6)} & 2\psi_2^{(4)} & 2\psi_2^{(5)} & 2\psi_2^{(6)} \\
0 & 0 & 0 & 0 & 0 & 0 & 2\psi_3^{(4)} & 2\psi_3^{(5)} & 2\psi_3^{(6)}
\end{array}\right].$$
$$\begin{array}{ccc}\underbrace{\qquad\qquad\qquad}_{\text{group 1}} & \underbrace{\qquad\qquad\qquad}_{\text{group 2}} & \underbrace{\qquad\qquad\qquad}_{\text{group 3}}\end{array}$$

Fig. 5. Matrix C_2 in the proof of Claim 1.

nine message symbols. That is, the (9×9) matrix formed by column-wise concatenation of any three nodal generator matrices, should be nonsingular. We consider the different possible sets of three nodes that the data collector can connect to, and provide appropriate decoding algorithms to handle each case.

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

b) Two Systematic Nodes and One Parity Node: Suppose the data collector connects to systematic nodes 2 and 3, and parity node 4. It obtains all the symbols stored in nodes 2 and 3 in uncoded form and proceeds to subtract their effect from the symbols in node 4. It is thus left to decode the message symbols $\underline{u}_1$, that are encoded using matrix $G_1^{(4)}$ given by

$$G_1^{(4)} = \begin{bmatrix} 2\psi_1^{(4)} & 0 & 0 \\ 2\psi_2^{(4)} & \psi_2^{(4)} & 0 \\ 2\psi_3^{(4)} & 0 & \psi_3^{(4)} \end{bmatrix}. \tag{23}$$

This lower-triangular matrix is nonsingular since by definition, all the entries in a Cauchy matrix are nonzero. The message symbols $\underline{u}_1$ can hence be recovered by inverting $G_1^{(4)}$.

c) All Three Parity Nodes: We consider next the case when a data collector connects to all three parity nodes. Let C_1 be the (9×9) matrix formed by the column-wise concatenation of the generator matrices of these three nodes.

Claim 1: The data collector can recover all the message symbols encoded using the matrix C_1, formed by the column-wise concatenation of the generator matrices of the three parity nodes:

$$C_1 = \left[\mathbf{G}^{(4)} \quad \mathbf{G}^{(5)} \quad \mathbf{G}^{(6)}\right]. \tag{24}$$

Proof: We permute the columns of C_1 to obtain a second matrix C_2 in which the ith $(i = 1, 2, 3)$ columns of all the three nodes are adjacent to each other as shown in Fig. 5.

Note that a permutation of the columns does not alter the information available to the data collector and hence is a permissible operation. This rearrangement of coded symbols, while not essential, simplifies the proof. We then post-multiply by a block-diagonal matrix Ψ_3^{-1} to obtain the matrix C_3 given by

$$C_3 = C_2 \begin{bmatrix} \Psi_3^{-1} & 0_3 & 0_3 \\ 0_3 & \Psi_3^{-1} & 0_3 \\ 0_3 & 0_3 & \Psi_3^{-1} \end{bmatrix} \tag{25}$$

$$= \left[\begin{array}{ccc|ccc|ccc}
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 & 0 \\
\hline
0 & 1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 & 0 \\
\hline
0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2
\end{array}\right]. \tag{26}$$

To put things back in perspective, the data collector at this point, has access to the coded symbols

$$\underline{\mathbf{u}}^t C_3$$

associated with the three parity nodes. From the nature of the matrix it is evident that message symbols u_1, u_5 and u_9 are now available to the data collector, and their effect can be subtracted from the remaining symbols to obtain the matrix

$$[u_2\ u_3\ u_4\ u_6\ u_7\ u_8] \underbrace{\begin{bmatrix} 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 & 0 \\ 1 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 2 \end{bmatrix}}_{C_4}. \tag{27}$$

As $2^2 \neq 1$ in $\mathbb{F}_7$, the matrix C_4 above can be verified to be nonsingular and thus the remaining message symbols can also be recovered by inverting C_4. ■

d) One Systematic Node and Two Parity Nodes: Suppose the data collector connects to systematic node 1 and parity nodes 4 and 5. All symbols of node 1, i.e., $\underline{u}_1$ are available to the data collector. Thus, it needs to decode the message-vector components $\underline{u}_2$ and $\underline{u}_3$ which are encoded using a matrix B_1 given by

$$B_1 = \begin{bmatrix} G_2^{(4)} & G_2^{(5)} \\ G_3^{(4)} & G_3^{(5)} \end{bmatrix}. \tag{28}$$

Claim 2: The block-matrix B_1 above is nonsingular and in this way, the message-vector components $\underline{u}_2$ and $\underline{u}_3$ can be recovered.

Proof: Once again, we begin by permuting the columns of B_1. For $i = 2, 3, 1$ (in this order), we group the ith columns of the two parity nodes together to give the matrix

$$B_2 = \left[\begin{array}{cc|cc|cc} 2\psi_1^{(4)} & 2\psi_1^{(5)} & 0 & 0 & \psi_2^{(4)} & \psi_2^{(5)} \\ 2\psi_2^{(4)} & 2\psi_2^{(5)} & 0 & 0 & 0 & 0 \\ 2\psi_3^{(4)} & 2\psi_3^{(5)} & \psi_2^{(4)} & \psi_2^{(5)} & 0 & 0 \\ \hline 0 & 0 & 2\psi_1^{(4)} & 2\psi_1^{(5)} & \psi_3^{(4)} & \psi_3^{(5)} \\ \psi_3^{(4)} & \psi_3^{(5)} & 2\psi_2^{(4)} & 2\psi_2^{(5)} & 0 & 0 \\ 0 & 0 & 2\psi_3^{(4)} & 2\psi_3^{(5)} & 0 & 0 \end{array}\right]. \tag{29}$$

Let Ψ_2 be the (2×2) submatrix of the Cauchy matrix Ψ_3, given by

$$\Psi_2 = \begin{bmatrix} \psi_2^{(4)} & \psi_2^{(5)} \\ \psi_3^{(4)} & \psi_3^{(5)} \end{bmatrix}. \tag{30}$$

Since every submatrix of Ψ_3 is nonsingular, so is Ψ_2. Keeping in mind the fact that the data collector can perform any linear operation on the columns of B_2, we next multiply the last two columns of B_2 by Ψ_2^{-1} (while leaving the other four columns unchanged) to obtain the matrix

$$B_3 = \left[\begin{array}{cc|cc|cc} 2\psi_1^{(4)} & 2\psi_1^{(5)} & 0 & 0 & 1 & 0 \\ 2\psi_2^{(4)} & 2\psi_2^{(5)} & 0 & 0 & 0 & 0 \\ 2\psi_3^{(4)} & 2\psi_3^{(5)} & \psi_2^{(4)} & \psi_2^{(5)} & 0 & 0 \\ \hline 0 & 0 & 2\psi_1^{(4)} & 2\psi_1^{(5)} & 0 & 1 \\ \psi_3^{(4)} & \psi_3^{(5)} & 2\psi_2^{(4)} & 2\psi_2^{(5)} & 0 & 0 \\ 0 & 0 & 2\psi_3^{(4)} & 2\psi_3^{(5)} & 0 & 0 \end{array}\right]. \tag{31}$$

The message symbols associated with the last two columns of B_2 are now available to the data collector and their effect on the rest of the encoded symbols can be subtracted out to get

$$B_4 = \left[\begin{array}{cc|cc} 2\psi_2^{(4)} & 2\psi_2^{(5)} & 0 & 0 \\ 2\psi_3^{(4)} & 2\psi_3^{(5)} & \psi_2^{(4)} & \psi_2^{(5)} \\ \hline \psi_3^{(4)} & \psi_3^{(5)} & 2\psi_2^{(4)} & 2\psi_2^{(5)} \\ 0 & 0 & 2\psi_3^{(4)} & 2\psi_3^{(5)} \end{array}\right]. \tag{32}$$

Along the lines of the previous case, the matrix B_4 above can be shown to be nonsingular. We note that this condition is equivalent to the reconstruction in a MISER code with $k = 2$ and a data collector that attempts to recover the data by connecting to the two parity nodes. ■

B. General MISER Code for $n = 2k$, $d = n - 1$

In this section, the construction of MISER code for the general parameter set $n = 2k$, $d = n - 1$ is provided. Since the MISER code is built to satisfy the cut-set bound, we have that $d = \alpha + k - 1$ which implies that

$$k = \alpha. \tag{33}$$

This relation will play a key role in the design of generator matrices for the parity nodes as this will permit each parity node to reserve $\alpha = k$ symbols associated with linearly independent global kernels for the repair of the k systematic nodes. In the example just examined, we had $\alpha = k = 3$. The construction of the MISER code for the general parameter set $n = 2k, d = n-1$ is very much along the lines of the construction of the example code.

1) Design of Nodal Generator Matrices: The first k nodes are systematic and store the message symbols in uncoded form. Thus the component generator matrices $G_i^{(\ell)}$, $1 \le i \le k$ of the ℓth systematic node, $1 \le \ell \le k$, are given by

$$G_i^{(\ell)} = \begin{cases} I_\alpha & \text{if } i = \ell \\ 0_\alpha & \text{if } i \ne \ell. \end{cases} \tag{34}$$

Let Ψ be an $(\alpha \times (n-k))$ matrix with entries drawn from $\mathbb{F}_q$ such that every submatrix of Ψ is of full rank. Since $n - k = \alpha = k$, we have that Ψ is a square matrix.[5] Let the columns of Ψ be given by

$$\Psi = \begin{bmatrix} \underline{\psi}^{(k+1)} & \underline{\psi}^{(k+2)} & \cdots & \underline{\psi}^{(n)} \end{bmatrix} \tag{35}$$

where the mth column is given by

$$\underline{\psi}^{(m)} = \begin{bmatrix} \psi_1^{(m)} \\ \vdots \\ \psi_\alpha^{(m)} \end{bmatrix}. \tag{36}$$

A Cauchy matrix is an example of such a matrix, and in our construction, we will assume Ψ to be a Cauchy matrix.

Definition 1 (Cauchy Matrix): An $(s \times t)$ Cauchy matrix Ψ over a finite field $\mathbb{F}_q$ is a matrix whose (i, j)th element $(1 \le i \le s, 1 \le j \le t)$ equals $\frac{1}{(x_i - y_j)}$ where $\{x_i\} \cup \{y_j\}$ is an injective sequence, i.e., a sequence with no repeated elements.

Thus the minimum field size required for the construction of a $(s \times t)$ Cauchy matrix is $s + t$. Hence if we choose Ψ to be a Cauchy matrix,

$$q \ge \alpha + n - k. \tag{37}$$

Any finite field satisfying this condition will suffice for our construction. Note that since $n - k \ge \alpha \ge 2$, we have $q \ge 4$.

We introduce some additional notation at this point. Denote the jth column of the $(\alpha \times \alpha)$ matrix $G_i^{(m)}$ as $\underline{g}_{i,j}^{(m)}$, i.e.,

$$G_i^{(m)} = \begin{bmatrix} \underline{g}_{i,1}^{(m)} & \cdots & \underline{g}_{i,\alpha}^{(m)} \end{bmatrix}. \tag{38}$$

[5] In Section V-D, we extend the construction to the even more general case of arbitrary $n, d \ge 2k-1$, under the added requirement however, that the replacement node connect to all of the remaining systematic nodes. In that section, we will be dealing with a rectangular $(\alpha \times (n-k))$ matrix Ψ.

The code is designed assuming a regeneration algorithm under which each of the α parity nodes passes its ℓth column for repair of the ℓth systematic node. With this in mind, for $k+1 \le m \le n, 1 \le i, j \le \alpha$, we choose

$$\underline{g}_{i,j}^{(m)} = \begin{cases} \epsilon \underline{\psi}^{(m)} & \text{if } i = j \\ \psi_i^{(m)} \underline{e}_j & \text{if } i \neq j. \end{cases} \tag{39}$$

where ϵ is an element from $\mathbb{F}_q$ such that $\epsilon \neq 0$ and $\epsilon^2 \neq 1$ (in the example provided in the previous section, $\epsilon \in \mathbb{F}_7$ was set equal to 2). The latter condition $\epsilon^2 \neq 1$ is needed during the reconstruction process, as was seen in the example. Note that there always exists such a value ϵ as long as $q \ge 4$.

As in the example, the generator matrix is also designed keeping in mind the need for interference alignment. This property is utilized in the exact repair of systematic nodes, as described below.

2) Exact Repair of Systematic Nodes: The repair process we associate with the MISER code is simple. The repair of a failed systematic node, say node ℓ, involves each of the remaining $d = n-1$ nodes passing their ℓth symbols (or equivalently, associated global kernels) respectively. In the set of α vectors passed by the parity nodes, the ℓth (desired) component is independent, and the remaining (interference) components are aligned. The interference components are cancelled using the vectors passed by the remaining systematic nodes. Independence in the desired component then allows for recovery of the desired message symbols.

The next theorem describes the repair algorithm in greater detail.

Theorem 2: In the MISER code, a failed systematic node can be exactly repaired by downloading one symbol from each of the remaining $d = n - 1$ nodes.

Proof: Consider repair of the systematic node ℓ. Each of the remaining $(n-1)$ nodes passes its ℓth column, so that the replacement node has access to the global kernels represented by the columns shown below:

$$\left[\underbrace{\begin{array}{c|c|c|c|c|c|}
\underline{e}_\ell & \cdots & \underline{0} & \underline{0} & \cdots & \underline{0} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\underline{0} & \cdots & \underline{e}_\ell & \underline{0} & \cdots & \underline{0} \\
\underline{0} & \cdots & \underline{0} & \underline{0} & \cdots & \underline{0} \\
\underline{0} & \cdots & \underline{0} & \underline{e}_\ell & \cdots & \underline{0} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\underline{0} & \cdots & \underline{0} & \underline{0} & \cdots & \underline{e}_\ell
\end{array}}_{\text{From systematic nodes}}
\underbrace{\begin{array}{c|c|c}
\psi_1^{(k+1)} \underline{e}_\ell & \cdots & \psi_1^{(n)} \underline{e}_\ell \\
\vdots & \ddots & \vdots \\
\psi_{\ell-1}^{(k+1)} \underline{e}_\ell & \cdots & \psi_{\ell-1}^{(n)} \underline{e}_\ell \\
\epsilon \underline{\psi}^{(k+1)} & \cdots & \epsilon \underline{\psi}^{(n)} \\
\psi_{\ell+1}^{(k+1)} \underline{e}_\ell & \cdots & \psi_{\ell+1}^{(n)} \underline{e}_\ell \\
\vdots & \ddots & \vdots \\
\psi_k^{(k+1)} \underline{e}_\ell & \cdots & \psi_k^{(n)} \underline{e}_\ell
\end{array}}_{\text{From parity nodes}}\right]$$

where $\underline{e}_\ell$ denotes the ℓth unit vector of length α and $\underline{0}$ denotes a zero vector of length α.

Observe that apart from the desired ℓth component, every other component is aligned along the vector $\underline{e}_\ell$. The goal is to show that some α linear combinations of the columns above will give us a matrix whose ℓth component equals the $(\alpha \times \alpha)$ identity matrix, and has zeros everywhere else. But this is clear from the interference alignment structure just noted in conjunction with linear independence of the α vectors in the desired component:

$$\{\underline{\psi}^{(k+1)}, \ldots, \underline{\psi}^{(n)}\}. \tag{40}$$

■

Next, we discuss the data reconstruction property.

3) Data Reconstruction (MDS Property): For reconstruction to be satisfied, a data collector downloading all symbols stored in any arbitrary k nodes should be able to recover the B message symbols. For this, we need the $(B \times B)$ matrix formed by the column-wise concatenation of any arbitrary collection of k nodal generator matrices to be nonsingular. The proof of this property is along the lines of the proof in the example. For completeness, a proof is presented in the appendix.

Theorem 3: A data collector connecting to any k nodes in the MISER code can recover all the B message symbols.

Proof: Please see the Appendix. ■

Remark 2: It is easily verified that both reconstruction and repair properties continue to hold even when we choose the generator matrices of the parity nodes to be given by: for $k+1 \le m \le n, 1 \le i, j \le \alpha$

$$\underline{g}_{i,j}^{(m)} = \begin{cases} \Sigma_i \underline{\psi}^{(m)} & \text{if } i = j \\ \psi_i^{(m)} \underline{e}_j & \text{if } i \neq j \end{cases} \tag{41}$$

where $\Sigma_i = \operatorname{diag}[\epsilon_{i,1} \cdots \epsilon_{i,\alpha}]$ is an $(\alpha \times \alpha)$ diagonal matrix satisfying

1) $\epsilon_{i,j} \neq 0, \forall\, i, j$
2) $\epsilon_{i,j}\, \epsilon_{j,i} \neq 1, \forall\, i \neq j$.

The first condition suffices to ensure exact repair of systematic nodes. The two conditions together ensure that the (MDS) reconstruction property holds as well.

C. The MISER Code for $n \ge 2k$, $d = n-1$ and Shortening of MSR Codes

In this section we show how the MISER code construction for $n = 2k$, $d = n-1$ can be extended to the more general case $n \ge 2k$, $d = n-1$. From the cut-set bound (5), for this parameter regime, we get

$$k \le \alpha. \tag{42}$$

We begin by first showing how an incremental change in parameters is possible.

Theorem 4 (Shortening of MSR Codes): An $[n, k, d]$, linear, systematic, exact-repair MSR code $\mathcal{C}$ can be derived from an $[n' = n+1, k' = k+1, d' = d+1]$ linear, systematic, exact-repair MSR code $\mathcal{C}'$. Furthermore if $d' = ak' + b$ in code $\mathcal{C}'$, $d = ak + b + (a-1)$ in code $\mathcal{C}$.

Proof: We begin by noting that

$$n - k = n' - k' \tag{43}$$

$$\alpha' = \alpha = d - k + 1 \tag{44}$$

$$B' = k'(d' - k' + 1) = B + \alpha. \tag{45}$$

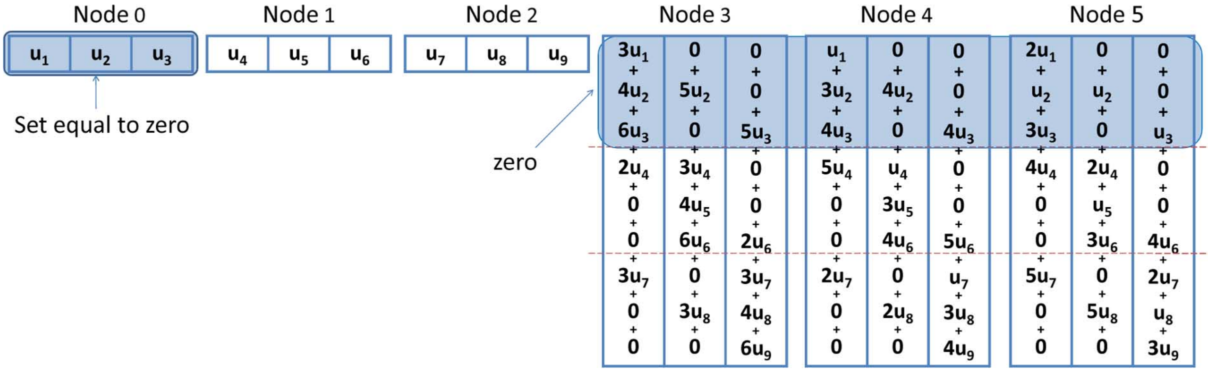

Node 0
Node 1
Node 2
Node 3
Node 4
Node 5
u_1 u_2 u_3
u_4 u_5 u_6
u_7 u_8 u_9
Set equal to zero
zero
$3u_1+4u_2+6u_3$ $0+5u_2+0$ $0+0+5u_3$
$2u_4+0+0$ $3u_4+4u_5+6u_6$ $0+0+2u_6$
$3u_7+0+0$ $0+3u_8+0$ $3u_7+4u_8+6u_9$
$u_1+3u_2+4u_3$ $0+4u_2+0$ $0+0+4u_3$
$5u_4+0+0$ $u_4+3u_5+4u_6$ $0+0+5u_6$
$2u_7+0+0$ $0+2u_8+0$ $u_7+3u_8+4u_9$
$2u_1+u_2+3u_3$ $0+u_2+0$ $0+0+u_3$
$4u_4+0+0$ $2u_4+u_5+3u_6$ $0+0+4u_6$
$5u_7+0+0$ $0+5u_8+0$ $2u_7+u_8+3u_9$

Fig. 6. Shortening technique for MSR codes: construction of a $[n = 5,\ k = 2,\ d = 4]$ MISER code from a $[n' = 6,\ k' = 3,\ d' = 5]$ MISER code. Shortening the code with respect to node zero is equivalent to removing systematic node 0 as well as the top component of every nodal generator matrix. The resulting $[n = 5,\ k = 2,\ d = 4]$ MISER code has $\{u_4, \ldots, u_9\}$ as its $B = k\alpha = 6$ message symbols.

In essence, we use code shortening [24] to derive code $\mathcal{C}$ from code $\mathcal{C}'$. Specification of code $\mathcal{C}$ requires that given a collection of $B = k\alpha$ message symbols, we identify the α code symbols stored in each of the n nodes. We assume without loss of generality, that in code $\mathcal{C}$, the nodes are numbered 1 through n, with nodes 1 through k representing the systematic nodes. We next create an additional node numbered 0.

The encoding algorithm for code $\mathcal{C}$ is based on the encoding algorithm for code $\mathcal{C}'$. Given a collection of B message symbols to be encoded by code $\mathcal{C}$, we augment this collection by an additional α message symbols all of which are set equal to zero. The first set of B message symbols will be stored in systematic nodes 1 through k and the string of α zeros will be stored in node 0. Nodes 0 through k are then regarded as constituting a set of $k'(= k+1)$ systematic nodes for code $\mathcal{C}'$. The remaining $(n-k)$ parity nodes are filled using the encoding process associated with code $\mathcal{C}'$ using the message symbols stored in the k' nodes numbered 0 through k. Note that both codes $\mathcal{C}$ and $\mathcal{C}'$ share the same number $(n-k)$ of parity nodes.

To prove the data reconstruction property of $\mathcal{C}$, it suffices to prove that all the B message symbols can be recovered by connecting to an arbitrary set of k nodes. Given a data collector connecting to a particular set of k nodes, we examine the corresponding scenario in code $\mathcal{C}'$ in which the data collector connects to node 0 in addition to these k nodes. By the assumed MDS property of code $\mathcal{C}'$, all the B message symbols along with the α message symbols stored in node 0 can be decoded using the data stored these $(k+1)$ nodes. However, since the α symbols stored in node 0 are all set equal to zero, they clearly play no part in the data-reconstruction process. It follows that the B message symbols can be recovered using the data from the k nodes (leaving aside node 0), thereby establishing that code $\mathcal{C}$ possesses the required MDS data-reconstruction property.

A similar argument can be used to establish the repair property of code $\mathcal{C}$ as well. Finally, we have

$$\begin{aligned} d' &= ak' + b \\ \Rightarrow d + 1 &= a(k+1) + b \\ \Rightarrow d &= ak + b + (a-1). \end{aligned}$$

■

Example: The code-shortening procedure presented in Theorem 4 is illustrated by an example of MISER code shown in Fig. 6. Here it is shown how a MISER code having code parameters $[n' = 6,\ k' = 3,\ d' = 5]$, $\beta' = 1$ and $(\alpha' = d' - k' + 1 = 3, B' = \alpha' k' = 9)$ yields upon shortening with respect to the message symbols in node 0, a MISER code having code parameters $[n = 5,\ k = 2,\ d = 4]$, $\beta = 1$ and $(\alpha = d - k + 1 = 3, B = \alpha k = 6)$.

By iterating the procedure in the proof of Theorem 4 above i times we obtain:

Corollary 5 (Shortening of MSR Codes): An $[n,\ k,\ d]$ linear, systematic, exact-repair MSR code $\mathcal{C}$ can be constructed by shortening a $[n' = n+i, k' = k+i, d' = d+i]$ linear, systematic, exact-repair MSR code $\mathcal{C}'$. Furthermore if $d' = ak' + b$ in code $\mathcal{C}'$, $d = ak + b + i(a-1)$ in code $\mathcal{C}$.

Remark 3: It is shown in the sequel Section VI-B that every linear, exact-repair MSR code can be made systematic. Thus, Theorem 4 and Corollary 5 apply to any linear, exact-repair MSR code (not necessarily systematic). In addition, note that the theorem and the associated corollary hold for general values of $[n,\ k,\ d]$ and are not restricted to the case of $d = n-1$. Furthermore, a little thought will show that they apply to linear codes $\mathcal{C}'$ that perform functional repair as well.

The next theorem follows from Corollary 5, and the code-shortening method employed in the Theorem 4.

Theorem 6: The MISER code for $n \geq 2k$, $d = n-1$ can be obtained by shortening the MISER code for $n' = n + (n-2k)$, $k' = k + (n-2k)$, $d' = d + (n-2k) = n' - 1$.

Proof: Follows from Theorem 4 and Corollary 5. ■

D. *Extension to $2k-1 \leq d \leq n-1$ When the Set of Helper Nodes Includes All Remaining Systematic Nodes*

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 additional constraint however, that the set of d helper nodes assisting a failed systematic node includes the remaining $(k-1)$ systematic nodes. The theorem below, shows that the code provided in Section V-B for $n = 2k, d = n-1$ supports the case $d = 2k-1$, $d \leq n-1$ as long as this additional requirement is met. From here on, extension to the general case $d \geq 2k-1$, $d \leq n-1$

is straightforward via the code-shortening result in Theorem 4. Note that unlike in the previous instance, the $(\alpha \times (n-k))$ Cauchy matrix used in the construction for $d < n-1$ is a rectangular matrix.

Theorem 7: For $d = 2k-1$, $d \leq n-1$, the code defined by the nodal generator matrices in (34) and (39), achieves reconstruction and optimal, exact repair of systematic nodes, provided the replacement node connects to all the remaining systematic nodes.

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

1) Exact Repair of Systematic Nodes: The replacement node connects to the $(k-1)$ remaining systematic nodes and an arbitrary α parity nodes (since, meeting the cut-set bound requires $d = k-1+\alpha$). Consider a distributed storage system having only these $(k-1+\alpha)$ nodes along with the failed node as its n nodes. Such a system has $d = n-1$, $d = 2k-1$ and is identical to the system described in Section V-B. Hence exact repair of systematic nodes meeting the cut-set bound is guaranteed. ■

E. Analysis of the MISER Code

1) Field Size Requirement: The only constraint on the field size comes due to the construction of the $(\alpha \times (n-k))$ matrix Ψ having all its submatrices of full rank. For our constructions, Ψ is chosen to be a Cauchy matrix. Hence from (37) and the fact that $\alpha = (d-k+1)$ for an MSR code, any field of size $(n+d-2k+1)$ or higher suffices. For specific parameters, the matrix Ψ can be handcrafted to yield smaller field sizes.

2) Complexity of Repair of Systematic Nodes: Each node participating in the exact repair of systematic node i, simply passes its ith symbol. This property not only alleviates the need for any computations to be performed at these helper nodes, but also minimises the total number of symbols that need to be read. The latter feature of our code is practically appealing since in certain applications of interest, the speed of repair may be limited by the number of disk reads required at the helper nodes.

The replacement node has to multiply the inverse of an $(\alpha \times \alpha)$ Cauchy matrix with an α-length vector and then perform $(k-1)$ subtractions for interference cancellation.

3) 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 (in the worst case) multiply the inverse of a $(k \times k)$ Cauchy matrix with k vectors, and perform other operations having a lower order of complexity.

F. Contiguous Reads for Repair

In many storage devices, data stored in contiguous locations can be read faster than that stored noncontiguously. Recall that under the MISER code (operating on one stripe of the data), the repair of the ith systematic node requires every other node to pass its ith symbol. Since repair is performed independently on each stripe, if the data is stored stripe-wise, repair operations would require noncontiguous reads at the helper nodes. In the present section, we describe a scheme for interleaving the data, that makes these reads contiguous. This scheme, while making the repair process faster, retains the optimality of the code with respect to its MDS property and the amount of data downloaded during repair.

The message is assumed to be a block of size ηB symbols (for some integer η). This block is divided into η stripes of B symbols each, and each stripe is encoded independently using the MISER code. Each node stores the encoded symbols in the following interleaved manner. The node first stores the first symbols of each stripe, followed by the second symbols of each stripe, and so on. More formally, denoting the ith symbol $(1 \leq i \leq (d-k+1))$ of stripe j $(1 \leq j \leq \eta)$ of the node as $s_i^{(j)}$, the symbols are stored in the node in the order $(s_1^{(1)}, \ldots, s_1^{(\eta)}, s_2^{(1)}, \ldots, s_2^{(\eta)}, \ldots, s_{d-k+1}^{(1)}, \ldots, s_{d-k+1}^{(\eta)})$. When the encoded data is stored in this fashion, the repair of any failed systematic node requires reading only contiguous data (within the block).

The interleaving scheme described above can also be used, in a straightforward manner, to enable contiguous reads during repair in other regenerating code constructions such as [9], [17], [25]. Moreover, this also raises the interesting question of whether one can design joint coding and data placement algorithms that can optimize both the repair bandwidth and the disk access time at the helper nodes.

G. Relation to Subsequent Work [14]

Two regenerating codes are equivalent if one code can be transformed into the other via a nonsingular symbol remapping (this definition is formalized in Section VI-B). The capabilities of equivalent codes are thus identical.

The initial presentation of the MISER code in [10] (the name "MISER" was coined only subsequently) provided the construction of the code, along with two (of three) parts of what may be termed as a complete decoding algorithm, namely: (a) reconstruction by a data collector, and (b) exact repair of failed systematic nodes. It was not known whether the third part of decoding, i.e., repair of a failed parity node could be carried out by the MISER code. Following the initial presentation of the MISER code [10], in [14] the authors establish that the MISER code can also perform exact repair of parity nodes and provide explicit mechanisms for the same.[6]

VI. Necessity of Interference Alignment and Nonexistence of Scalar, Linear, Exact-Repair MSR Codes for $d < 2k-3$

In Section V, explicit, exact-repair MSR codes are constructed for the parameter regimes $(d \geq 2k-1,\ d = n-1)$ performing reconstruction and exact repair of systematic nodes. These constructions are based on the concept of interference alignment. Furthermore, these codes have a desirable property

[6] In [14] a class of regenerating codes is presented that have the same parameters as the MISER code. This class of codes can however, be shown to be equivalent to the MISER code (and hence to each other) under the equivalence notion presented in Section VI-B.

of having the smallest possible value for the parameter β, i.e., $\beta = 1$.

As previously discussed in Section I-C, the problem of constructing exact-repair MSR codes is (in part) a non-multicast network coding problem. In particular, for the case of $\beta = 1$, it reduces to a scalar network coding problem. Upon increase in the value of β, the capacity of every data pipe is increased by a factor of β, thereby transforming it into a vector network coding problem. Thus, $\beta = 1$ corresponds to the absence of symbol extension, which in general, reduces the complexity of system implementation. Furthermore, as noted in Section I-A, an MSR code for every larger integer value of β, can be obtained by concatenating multiple copies of a $\beta = 1$ code. For this reason, the case of $\beta = 1$ is of special interest and a large section of the literature in the field of regenerating codes ([8]–[16]) is devoted to this case.

In the present section, we show that for $d < 2k-3$, there exist no linear, exact-repair MSR codes achieving the cut-set bound on the repair bandwidth in the absence of symbol extension. In fact, we show that the cut-set bound cannot be achieved even if exact repair of only the systematic nodes is desired. We first assume the existence of such a linear, exact-repair MSR code $\mathcal{C}$ satisfying:

$$(\beta = 1,\ \alpha = d - k + 1,\ B = k\alpha) \tag{46}$$

and

$$(d < 2k - 3 \Rightarrow \alpha < k - 2). \tag{47}$$

Subsequently, we derive properties that this code must necessarily satisfy. Many of these properties hold for a larger regime of parameters and are therefore of independent interest. In particular, we prove that *interference alignment*, in the form described in Section IV, is *necessary*. We will show that when $d < 2k - 3$ the system becomes over-constrained, leading to a contradiction.

Remark 4: In recent work, subsequent to the original submission of this paper, it is shown in [19], [20] that the MSR point under exact repair can be achieved asymptotically for all $[n,\ k,\ d]$ via an infinite symbol extension, i.e., in the limit as $\beta \rightarrow \infty$. This is established by presenting a scheme under which $\lim_{\beta\rightarrow\infty} \frac{\nu}{\beta} = d$, where ν denotes the repair bandwidth. Note that in the asymptotic setup, the parameters α and B also tend to infinity as they are multiples of β.

A. Additional Notation

We introduce some additional notation for the vectors passed by the helper nodes to the replacement node. For ℓ, $m \in \{1,\ldots,n\}$, a set $\mathcal{D}$ of d nodes with $m \in \mathcal{D}$ and $\ell \notin \mathcal{D}$, let $\mathcal{D}\underline{\gamma}^{(m,\ell)}$, denote the vector passed by node m for repair of node ℓ, where $\mathcal{D}$ is the set of d nodes assisting in the repair. In keeping with our component notation, we will use $\mathcal{D}\underline{\gamma}_i^{(m,\ell)}$ to denote the ith component, $1 \leq i \leq k$, of this vector.

Recall that a set of vectors are *aligned* when the vector-space spanned by them has a dimension no more than one. Given a matrix A, we denote its column-space by $\mathrm{colspace}[A]$ and its (right) null space by $\mathrm{nullspace}[A]$. Clearly, $\mathcal{D}\underline{\gamma}^{(m,\ell)} \in \mathrm{colspace}\left[\mathbf{G}^{(m)}\right]$.

B. Equivalent Codes

Two codes $\mathcal{C}$ and $\mathcal{C}'$ are equivalent if $\mathcal{C}'$ can be represented in terms of $\mathcal{C}$ by

i) a change of basis of the vector space generated by the message symbols (i.e., a remapping of the message symbols), and

ii) a change of basis of the column-spaces of the nodal generator matrices (i.e., a remapping of the symbols stored within a node).

A more rigorous definition is as follows.

Definition 2 (Equivalent Codes): Two Codes $\mathcal{C}$ and $\mathcal{C}'$ are equivalent if

$$\mathbf{G}'^{(m)} = W\ \mathbf{G}^{(m)}\ U^{(m)} \tag{48}$$

$$\mathcal{D}\underline{\gamma}'^{(m,\ell)} = W_{\mathcal{D}}\underline{\gamma}^{(m,\ell)} \tag{49}$$

$\forall\ \ell, m \in \{1,\ldots,n\}$, $\ell \neq m$, all sets $\mathcal{D}$ of d helper nodes (with $m \in \mathcal{D}$ and $\ell \notin \mathcal{D}$), for some $(B \times B)$ nonsingular matrix W, and some $(\alpha \times \alpha)$ nonsingular matrices $U^{(m)}$.

Since the only operator required to transform a code to its equivalent is a symbol remapping, two equivalent codes are identical with respect to data reconstruction and repair bandwidth. Hence, in the sequel, we will not distinguish between two equivalent codes and the notion of code equivalence will play an important role in the present section. Here, properties of a code that is equivalent to a given code are first derived and the equivalence then guarantees that these properties hold for the given code as well. The next theorem uses the notion of equivalent codes to show that every linear exact-repair MSR code can be made systematic.

Theorem 8: Every linear, exact-repair MSR code can be made systematic via a nonsingular linear transformation of the rows of the generator matrix, which simply corresponds to a re-mapping of the message symbols. Furthermore, the choice of the k nodes that are to be made systematic can be arbitrary.

Proof: Let the generator matrix of the given linear, exact-repair MSR code $\mathcal{C}$ be $\mathbb{G}$. We will derive an equivalent code $\mathcal{C}'$ that has its first k nodes in systematic form. The reconstruction (MDS property) of code $\mathcal{C}$ implies that the $(B \times B)$ submatrix of $\mathbb{G}$,

$$\left[\mathbf{G}^{(1)}\ \mathbf{G}^{(2)}\ \cdots\ \mathbf{G}^{(k)}\right]$$

is nonsingular. Define an equivalent code $\mathcal{C}'$ having its generator matrix $\mathbb{G}'$ as:

$$\mathbb{G}' = \left[\mathbf{G}^{(1)}\ \mathbf{G}^{(2)}\ \cdots\ \mathbf{G}^{(k)}\right]^{-1}\ \mathbb{G}. \tag{50}$$

Clearly, the B left-most columns of $\mathbb{G}'$ form a $B \times B$ identity matrix, thus making the equivalent code $\mathcal{C}'$ systematic. As the repair is exact, the code will retain the systematic form following any number of failures and repairs.

The transformation in (50) can involve any arbitrary set of k nodes in $\mathcal{C}$, thus proving the second part of the theorem. ■

The theorem above permits us to restrict our attention to the class of systematic codes, and assume the first k nodes (i.e.,

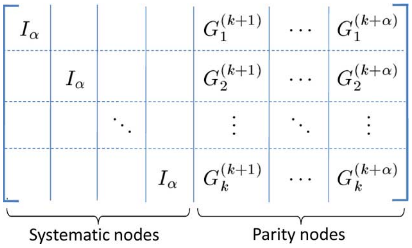


Fig. 7. Generator matrix $\mathbb{G}$ of the entire code. First k (block) columns are associated with the systematic nodes 1 to k and the next α (block) columns to the parity nodes $(k+1)$ to $(k+\alpha)$. Empty blocks denote zero matrices.

nodes $1,\ldots,k$) to be systematic. Recall that, for systematic node ℓ $(\in \{1,\ldots,k\})$,

$$G_i{}^{(\ell)} = \begin{cases} I_\alpha & \text{if } i=\ell \\ 0_\alpha & \text{if } i\neq \ell \end{cases} \quad \forall i \in \{1,\ldots,k\}. \tag{51}$$

Thus, systematic node ℓ stores the α symbols in $\underline{u}_\ell$.

C. Approach

An exact-repair MSR code should be capable of performing exact repair of any failed node by connecting to any arbitrary subset of d of the remaining $(n-1)$ nodes, while meeting the cut-set bound on repair bandwidth. This requires a number of repair scenarios to be satisfied. Our proof of nonexistence considers a less restrictive setting, in which exact repair of only the systematic nodes is to be satisfied. Further, we consider only the situation where a failed systematic node is to be repaired by downloading data from a specific set of d nodes, comprised of the $(k-1)$ remaining systematic nodes, and some collection of α parity nodes. Thus, for the remainder of this section, we will restrict our attention to a subset of the n nodes in the distributed storage network, of size $(k+\alpha)$ nodes, namely, the set of k systematic nodes and the first α parity nodes. Without loss of generality, within this subset, we will assume that nodes 1 through k are the systematic nodes and that nodes $(k+1)$ through $(k+\alpha)$ are the α parity nodes. Then with this notation, upon failure of systematic node ℓ, $1\leq \ell \leq k$, the replacement node is assumed to connect to nodes $\{1,\ldots,k+\alpha\}\backslash\{\ell\}$. It follows that in the notation $_\mathcal{D}\underline{\gamma}^{(m,\ell)}$ introduced earlier in the paper, fixing the replacement node ℓ also fixes the set of helper nodes $\mathcal{D}$. Thus, in the rest of the paper, we drop the subscript $\mathcal{D}$ from this notation.

The generator matrix $\mathbb{G}$ of the entire code can be written in a block-matrix form as shown in Fig. 7. In the figure, each (block) column represents a node and each (block) row, a component. The first k and the remaining α columns contain respectively, the generator matrices of the k systematic nodes and the α parity nodes.

We now outline the steps involved in proving the nonexistence result. Along the way, we will uncover some interesting and insightful properties possessed by linear, exact-repair MSR codes.

1) We begin by establishing that in order to satisfy the data reconstruction property, each component in the parity-node section of the generator matrix (see Fig. 7) must be nonsingular.

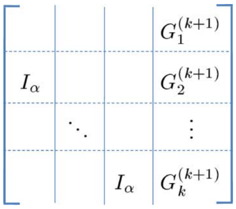


Fig. 8. Block matrix accessed by a data collector connecting to systematic nodes 2 through k and parity node $(k+1)$. Empty blocks denote zero matrices.

2) Next, we show that the vectors passed by the α parity nodes for the repair of any systematic node must necessarily satisfy two properties:
   - alignment of the interference components, and
   - linear independence of the desired component.
3) We then prove that in the collection of k vectors passed by a parity node for the respective repair of the k systematic nodes, every α-sized subset must be linearly independent. This is a key step that links the vectors stored in a node to those passed by it, and enables us to replace the α columns of the generator matrix of a parity node with the vectors it passes to aid in the repair of some subset of α systematic nodes. We will assume that these α systematic nodes are in fact, nodes 1 through α.
4) Finally, we will show that the necessity of satisfying multiple interference-alignment conditions simultaneously, turns out to be over-constraining, forcing alignment in the desired components as well. This leads to a contradiction, thereby proving the nonexistence result.

D. Deduced Properties

Property 1 (Nonsingularity of the Component Submatrices): Each of the components $\{G_i^{(m)} \mid k+1 \leq m \leq k+\alpha,\ 1\leq i \leq k\}$ is nonsingular.

Proof: Consider a data collector connecting to systematic nodes 2 to k and parity node $(k+1)$. The data collector has thus access to the block matrix shown in Fig. 8.

For the data collector to recover all the data, this block matrix must be nonsingular, forcing $G_1^{(k+1)}$ to be nonsingular. A similar argument shows that the same must hold in the case of each of the other components. ■

Corollary 9: Let $H = [H_1^t\ H_2^t \cdots H_k^t]^t$ be a $(k\alpha \times \ell)$ matrix each of whose $\ell \geq 1$ columns is a linear combination of the columns of $\mathbf{G}^{(m)}$ for some $m \in \{k+1,\ldots,k+\alpha\}$, and having k components $\{H_i\}$ of size $(\alpha \times \ell)$. Thus

$$\text{colspace}[H] \subseteq \text{colspace}[\mathbf{G}^{(m)}].$$

Then for every $i \in \{1,\ldots,k\}$, it must be that

$$\text{nullspace}[H_i] = \text{nullspace}[H]. \tag{52}$$

Proof: Clearly,

$$\text{nullspace}[H] \subseteq \text{nullspace}[H_i]. \tag{53}$$

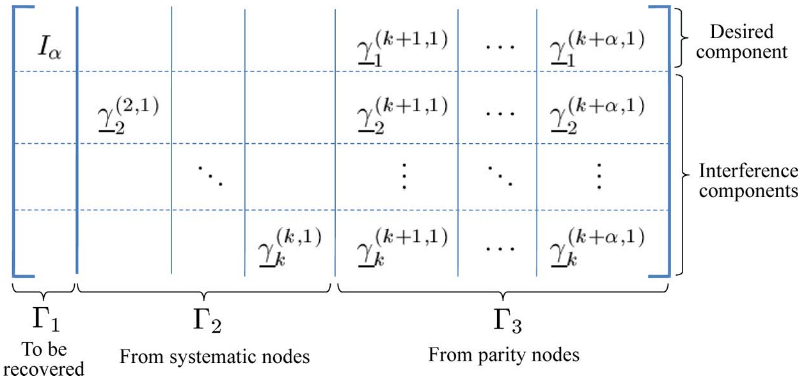

I_α
$\underline{\gamma}_1^{(k+1,1)}$ $\cdots$ $\underline{\gamma}_1^{(k+\alpha,1)}$
Desired component
$\underline{\gamma}_2^{(2,1)}$
$\underline{\gamma}_2^{(k+1,1)}$ $\cdots$ $\underline{\gamma}_2^{(k+\alpha,1)}$
Interference components
$\underline{\gamma}_k^{(k,1)}$
$\underline{\gamma}_k^{(k+1,1)}$ $\cdots$ $\underline{\gamma}_k^{(k+\alpha,1)}$
Γ_1
To be recovered
Γ_2
From systematic nodes
Γ_3
From parity nodes

Fig. 9. Matrix depicting the α (global-kernel) vectors to be recovered by replacement node 1 (represented by the matrix Γ_1), alongside the d vectors passed by the helper nodes $2,\ldots,k+\alpha$ (represented by $[\Gamma_2 \mid \Gamma_3]$). Empty blocks denote zero matrices.

Let $H = \mathbf{G}^{(m)}A$, for some $(\alpha \times \ell)$ matrix A. Then

$$H_i = G_i^{(m)}A. \tag{54}$$

For a vector $\underline{v} \in \text{nullspace}[H_i]$,

$$H_i\ \underline{v} = G_i^{(m)} A\ \underline{v} = \underline{0}. \tag{55}$$

However, since $G_i^{(m)}$ is of full rank (Property 1) it follows that

$$A\ \underline{v} = \underline{0} \tag{56}$$

$$\Rightarrow \mathbf{G}^{(m)} A\ \underline{v} = H\underline{v} = \underline{0} \tag{57}$$

$$\Rightarrow \text{nullspace}[H_i] \subseteq \text{nullspace}[H]. \tag{58}$$

■

The corollary says, in essence, that any linear dependence relation that holds amongst the columns of any of the components H_i, also extends to the columns of the entire matrix H itself.

We next establish properties that are mandated by the repair capabilities of exact regenerating codes. Consider the situation where a failed systematic node, say node ℓ, $1 \leq \ell \leq k$, is repaired using one vector (as $\beta = 1$) from each of the remaining $k-1+\alpha$ nodes.

Definition 3: When considering repair of systematic node ℓ, $1 \leq \ell \leq k$, the ℓth component of each of the α vectors passed by the α parity nodes $\{\underline{\gamma}_\ell^{(m,\ell)} \mid k+1 \leq m \leq k+\alpha\}$ will be termed as *desired components*. The remaining components $\{\underline{\gamma}_i^{(m,\ell)} \mid k+1 \leq m \leq k+\alpha\}$, for every $i \in \{1,\ldots,k\}\backslash\{\ell\}$ will be termed as *interference components*.

The next property highlights the necessity of interference alignment in any exact-repair MSR code. Clearly, the vectors passed by the remaining $(k-1)$ systematic nodes have ℓth component equal to $\underline{0}$, and thus the onus of recovering the "desired" ℓth component of replacement node ℓ falls on the α parity nodes. However, the vectors passed by the parity nodes have nonzero "interference" components that can be nulled out only by the vectors passed by the systematic nodes. This forces an alignment in these interference components, and this is shown more formally below.

Property 2 (Necessity of Interference Alignment): In the vectors $\{\underline{\gamma}^{(m,\ell)} \mid k+1 \leq m \leq k+\alpha\}$ passed by the α parity nodes for the repair of any systematic node (say, node ℓ), for every $i \in \{1,\ldots,k\}\backslash\{\ell\}$ the α interference components $\{\underline{\gamma}_i^{(m,\ell)} \mid k+1 \leq m \leq k+\alpha\}$ must necessarily be *aligned*, and the desired components $\{\underline{\gamma}_\ell^{(m,\ell)} \mid k+1 \leq m \leq k+\alpha\}$ must necessarily be linearly independent.

Proof: We assume without loss of generality that $\ell = 1$, i.e., we consider repair of systematic node 1. The matrix depicted in Fig. 9 consists of the α vectors needed to be recovered in the replacement node ℓ, alongside the d vectors passed by the d helper nodes $2,\ldots,k+\alpha$. This matrix may be decomposed into three submatrices, namely: a $(B \times \alpha)$ matrix Γ_1, comprising the α columns to be recovered at the replacement node; a $(B \times (k-1))$ matrix Γ_2, comprising the $(k-1)$ vectors passed by the remaining systematic nodes; and a $(B \times \alpha)$ matrix Γ_3, comprising the α vectors passed by the parity nodes.

The vectors $\{\underline{\gamma}_1^{(k+1,1)}, \ \ldots\ , \underline{\gamma}_1^{(k+\alpha,1)}\}$ appearing in the first row of the matrix constitute the desired component; for every $i \in \{2,\ldots,k\}$, the set of vectors $\{\underline{\gamma}_i^{(k+1,1)}, \ \ldots\ , \underline{\gamma}_i^{(k+\alpha,1)}\}$, constitute interference components. Exact repair of node 1 is equivalent to the recovery of Γ_1 from the columns of Γ_2 and Γ_3 through a linear transformation, and hence it must be that

$$\text{colspace}[\Gamma_1] \subseteq \text{colspace}\,[\Gamma_2|\Gamma_3] \tag{59}$$

where the "$|$" operator denotes concatenation. When we restrict attention to the first components of the matrices, we see that we must have

$$\text{colspace}[I_\alpha] \subseteq \text{colspace}\left[\underline{\gamma}_1^{(k+1,1)}\ \cdots\ \underline{\gamma}_1^{(k+\alpha,1)}\right] \tag{60}$$

thereby forcing the desired components $\{\underline{\gamma}_1^{(k+1,1)}, \ \ldots\ , \underline{\gamma}_1^{(k+\alpha,1)}\}$ to be linearly independent. Further, from(59) it follows that

$$\text{colspace}\,[\Gamma_1|\Gamma_2] \subseteq \text{colspace}\,[\Gamma_2|\Gamma_3]\,. \tag{61}$$

Clearly, $\text{rank}[\Gamma_1] = \alpha$, and from Fig. 9 it can be inferred that

$$\text{rank}[\Gamma_1|\Gamma_2] = \alpha + \text{rank}[\Gamma_2]. \tag{62}$$

Moreover, as the first component in Γ_3 is of rank α (from (60)),

$$\text{rank}[\Gamma_2|\Gamma_3] \leq \text{rank}[\Gamma_2] + \alpha \tag{63}$$

$$= \text{rank}[\Gamma_1|\Gamma_2]. \tag{64}$$

TO (systematic node)

FROM (parity node)	1	2	3	$\cdots$	α
$k+1$	$\underline{\gamma}^{(k+1,1)}$	$\underline{\gamma}^{(k+1,2)}$	$\underline{\gamma}^{(k+1,3)}$	$\cdots$	$\underline{\gamma}^{(k+1,\alpha)}$
$k+2$	$\underline{\gamma}^{(k+2,1)}$	$\underline{\gamma}^{(k+2,2)}$	$\underline{\gamma}^{(k+2,3)}$	$\cdots$	$\underline{\gamma}^{(k+2,\alpha)}$
$k+3$	$\underline{\gamma}^{(k+3,1)}$	$\underline{\gamma}^{(k+3,2)}$	$\underline{\gamma}^{(k+3,3)}$	$\cdots$	$\underline{\gamma}^{(k+3,\alpha)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$k+\alpha$	$\underline{\gamma}^{(k+\alpha,1)}$	$\underline{\gamma}^{(k+\alpha,2)}$	$\underline{\gamma}^{(k+\alpha,3)}$	$\cdots$	$\underline{\gamma}^{(k+\alpha,\alpha)}$

Fig. 10. Table indicating the vectors passed by the α parity nodes to repair the first α systematic nodes.

It follows from (61) and (64), that

$$\text{colspace}\,[\Gamma_1|\Gamma_2] = \text{colspace}\,[\Gamma_2|\Gamma_3] \qquad (65)$$

and this forces, for $i \in \{2,\ldots,k\}$,

$$\text{colspace}\left[\underline{\gamma}_i^{(k+1,1)} \ \cdots \ \underline{\gamma}_i^{(k+\alpha,1)}\right] \subseteq \text{colspace}\left[\underline{\gamma}_i^{(i,1)}\right]. \qquad (66)$$

Thus, the interference components in Γ_3 are forced to be aligned. ■

Remark 5: Properties 1 and 2 also hold for all $\beta \geq 1$, in which case, each of the α helper parity nodes pass a β-dimensional subspace, and each interference component needs to be confined to a β-dimensional subspace. Furthermore, the two properties also hold for all $[n,\ k,\ d]$ exact-repair MSR codes (not necessarily systematic), when $(k-1)$ of the d helper nodes along with the replacement node are viewed as systematic.

The next property links the vectors stored in a parity node to the vectors it passes to aid in the repair of any set of α systematic nodes.

Property 3: For $d < 2k-1$, the vectors passed by a parity node to repair any arbitrary set of α systematic nodes are linearly independent, i.e., for $m \in \{k+1,\ldots,k+\alpha\}$, it must be that every subset of size α drawn from the set of vectors

$$\left\{\underline{\gamma}^{(m,1)},\ldots,\underline{\gamma}^{(m,k)}\right\}$$

is linearly independent. (Thus the matrix $[\underline{\gamma}^{(m,1)} \ \cdots \ \underline{\gamma}^{(m,k)}]$ may be viewed as the generator matrix of a $[k,\alpha]$-MDS code.)

Proof: Consider Fig. 10 which depicts the vectors passed by parity nodes $\{k+1,\ldots,k+\alpha\}$ to repair systematic nodes $\{1,\ldots,\alpha\}$. From Property 2 one can infer that in column $i \in \{1,\ldots,\alpha\}$, the ith (desired) components of the α vectors are independent, and the jth (interference) components for all $j \in \{1,\ldots,k\}\backslash\{i\}$ are aligned. In particular, for all $j \in \{\alpha+1,\ldots,k\}$, the jth components of each column are aligned. Note that as $d < 2k-1$ we have $k > \alpha$, which guarantees that the set $\{\alpha+1,\ldots,k\}$ is nonempty, and hence the presence of an $(\alpha+1)$th component.

We will prove Property 3 by contradiction. Suppose, for example, we were to have

$$\underline{\gamma}^{(k+1,1)} \subseteq \text{colspace}\left[\underline{\gamma}^{(k+1,2)} \ \cdots \ \underline{\gamma}^{(k+1,\alpha)}\right] \qquad (67)$$

which is an example situation under which the α vectors passed by parity node $(k+1)$ for the respective repair of the first α systematic nodes would fail to be linearly independent. Restricting our attention to component $(\alpha+1)$, we get

$$\underline{\gamma}_{\alpha+1}^{(k+1,1)} \subseteq \text{colspace}\left[\underline{\gamma}_{\alpha+1}^{(k+1,2)} \ \cdots \ \underline{\gamma}_{\alpha+1}^{(k+1,\alpha)}\right]. \qquad (68)$$

Now, recall that the $(\alpha+1)^{\text{th}}$ components of each column are aligned. Hence from (68), in all other parity nodes, it must be that

$$\underline{\gamma}_{\alpha+1}^{(m,1)} \subseteq \text{colspace}\left[\underline{\gamma}_{\alpha+1}^{(m,2)} \ \cdots \ \underline{\gamma}_{\alpha+1}^{(m,\alpha)}\right] \quad \forall\, m \in \{k+2,\ldots,k+\alpha\}. \qquad (69)$$

Noting that a vector passed by a helper node lies in the column-space of its generator matrix, we now invoke Corollary 9:

$$\text{nullspace}\left[\underline{\gamma}_{\alpha+1}^{(m,1)} \ \cdots \ \underline{\gamma}_{\alpha+1}^{(m,\alpha)}\right] = \text{nullspace}\left[\underline{\gamma}^{(m,1)} \ \cdots \ \underline{\gamma}^{(m,\alpha)}\right] \quad \forall m \in \{k+1,\ldots,k+\alpha\}. \qquad (70)$$

This, along with (68) and (69), implies

$$\underline{\gamma}^{(m,1)} \subseteq \text{colspace}\left[\underline{\gamma}^{(m,2)} \ \cdots \ \underline{\gamma}^{(m,\alpha)}\right] \quad \forall m \in \{k+1,\ldots,k+\alpha\}. \qquad (71)$$

Thus the dependence in the vectors passed by one parity node carries over to every other parity node.

In particular, we have

$$\underline{\gamma}_1^{(m,1)} \subseteq \text{colspace}\left[\underline{\gamma}_1^{(m,2)} \ \cdots \ \underline{\gamma}_1^{(m,\alpha)}\right] \quad \forall d \in \{k+1,\ldots,k+\alpha\}. \qquad (72)$$

However, from Property 2, we know that the vectors passed to systematic nodes 2 to α have their first components aligned, i.e.,

$$\text{rank}\left[\underline{\gamma}_1^{(k+1,\ell)} \ \cdots \ \underline{\gamma}_1^{(k+\alpha,\ell)}\right] \leq 1 \ \ \forall \ell \in \{2,\ldots,\alpha\}. \qquad (73)$$

Aggregating all instantiations (w.r.t. m) of (72), the desired component is confined to:

$$\begin{aligned}
&\text{colspace}\left[\underline{\gamma}_1^{(k+1,1)} \ \cdots \ \underline{\gamma}_1^{(k+\alpha,1)}\right] \\
&\ \subseteq \text{colspace}\left[\underline{\gamma}_1^{(k+1,2)} \cdots \underline{\gamma}_1^{(k+\alpha,2)} | \cdots | \underline{\gamma}_1^{(k+1,\alpha)} \cdots \underline{\gamma}_1^{(k+\alpha,\alpha)}\right] \\
&\ \Rightarrow \text{rank}\left[\underline{\gamma}_1^{(k+1,1)} \ \cdots \ \underline{\gamma}_1^{(k+\alpha,1)}\right] \\
&\ \leq \text{rank}\left[\underline{\gamma}_1^{(k+1,2)} \cdots \underline{\gamma}_1^{(k+\alpha,2)} | \cdots | \underline{\gamma}_1^{(k+1,\alpha)} \cdots \underline{\gamma}_1^{(k+\alpha,\alpha)}\right] \\
&\ \leq \sum_{\ell=2}^{\alpha} \text{rank}\left[\underline{\gamma}_1^{(k+1,\ell)} \ \cdots \ \underline{\gamma}_1^{(k+\alpha,\ell)}\right] \\
&\ \leq \alpha - 1, \qquad (74)
\end{aligned}$$

where the last inequality follows from (73). This contradicts the assertion of Property 2 with respect to the desired component:

$$\text{rank}\left[\underline{\gamma}_1^{(k+1,1)} \ \cdots \ \underline{\gamma}_1^{(k+\alpha,1)}\right] = \alpha. \qquad (75)$$

■

Remark 6: It turns out that an attempted proof of the analogue of this theorem for the case $\beta > 1$, fails to hold.

The connection between the vectors passed by a parity node and those stored by it, resulting out of Property 3, is presented in the following corollary.

Corollary 10: If there exists a linear, exact-repair MSR code for $d < 2k - 1$, then there exists an equivalent linear, exact-repair MSR code, where, for each parity node, the α columns of the generator matrix are respectively the vectors passed for the repair of the first α systematic nodes.

Proof: Since a node can pass only a function of what it stores, the vectors passed by a parity node $m \in \{k+1, \ldots, k+\alpha\}$, for repair of the systematic nodes must belong to the column-space of its generator matrix, i.e.,

$$\text{colspace}\left[\underline{\gamma}^{(m,1)} \cdots \underline{\gamma}^{(m,\alpha)}\right] \subseteq \text{colspace}\left[\mathbf{G}^{(m)}\right]. \tag{76}$$

Further, Property 3 asserts that the vectors it passes for repair of the first α systematic nodes are linearly independent, i.e.,

$$\text{rank}\left[\underline{\gamma}^{(m,1)} \cdots \underline{\gamma}^{(m,\alpha)}\right] = \alpha = \text{rank}\left[\mathbf{G}^{(m)}\right]. \tag{77}$$

It follows from (76) and (77) that the generator matrix $\mathbf{G}^{(m)}$ is a nonsingular transformation of the matrix $\left[\underline{\gamma}^{(m,1)} \cdots \underline{\gamma}^{(m,\alpha)}\right]$. The two codes with generator matrices given by the two representations are hence equivalent. ■

In the equivalent code, each row of Fig. 10 corresponds to the generator matrix $\mathbf{G}^{(m)}$ of the associated parity node, i.e.,

$$\mathbf{G}^{(m)} = \left[\underline{\gamma}^{(m,1)} \cdots \underline{\gamma}^{(m,\alpha)}\right] \qquad \forall\, m \in \{k+1, \ldots, k+\alpha\}. \tag{78}$$

Since the capabilities of a code are identical to an equivalent code, we will restrict our attention to this generator matrix for the remainder of this section.

It follows from the discussion above that in the equivalent code, for $1 \le i \le \alpha$, each parity node passes its ith symbol for repair of the ith systematic node. Coupled with the necessity of interference alignment (Property 2), this implies that for any $j \in \{\alpha+1, \ldots, k\}$ and $i \in \{1, \ldots, \alpha\}$, the jth component of the ith column of the respective generator matrices of any two parity nodes are scalar multiples of each other. This is formalized in the following property.

Property 4 (Code Structure—What Is Stored): For $d < 2k - 1$, any component ranging from $(\alpha+1)$ to k across the generator matrices of the parity nodes differ only by the presence of a multiplicative diagonal matrix on the right, i.e.,

$$\begin{array}{ccc} G_{\alpha+1}^{(k+1)} = H_{\alpha+1}\,\Lambda_{\alpha+1}^{(k+1)}, & \ldots, & G_{\alpha+1}^{(k+\alpha)} = H_{\alpha+1}\,\Lambda_{\alpha+1}^{(k+\alpha)} \\ \vdots & \ddots & \vdots \\ G_{k}^{(k+1)} = H_{k}\,\Lambda_{k}^{(k+1)}, & \ldots, & G_{k}^{(k+\alpha)} = H_{k}\,\Lambda_{k}^{(k+\alpha)} \end{array} \tag{79}$$

where the matrices of the form $\Lambda_*^{(*)}$ are $(\alpha \times \alpha)$ diagonal matrices (and where, for instance, we can choose $H_{\alpha+1} = G_{\alpha+1}^{(k+1)}$, in which case $\Lambda_{\alpha+1}^{(k+1)} = I_\alpha$).

Proof: Consider the first column in Fig. 10, comprising the vectors passed by the α parity nodes to repair node 1. Property 2 tells us that in these α vectors, the components ranging from $(\alpha+1)$ to k constitute interference, and are hence aligned. Clearly, the same statement holds for every column in Fig. 10. Since the generator matrices of the parity nodes are as in (78), the result follows. ■

For the repair of a systematic node, a parity node passes a vector from the column-space of its generator matrix, i.e., the vector $\underline{\gamma}^{(m,\ell)}$ passed by parity node m for repair of failed systematic node ℓ can be written in the form:

$$\underline{\gamma}^{(m,\ell)} = \mathbf{G}^{(m)}\,\underline{\theta}^{(m,\ell)} \tag{80}$$

for some α-length vector $\underline{\theta}^{(m,\ell)}$.

In the equivalent code obtained in (78), a parity node simply stores the α vectors it passes to repair the first α systematic nodes. On the other hand, the vector passed to systematic node $\ell, \alpha+1 \le \ell \le k$, is a linear combination of these α vectors. The next property employs Property 3 to show that every coefficient in this linear combination is nonzero.

Property 5 (Code Structure—What is Passed): For $d < 2k - 1$, and a helper parity node m assisting a failed systematic node ℓ

(a) For $\ell \in \{1, \ldots, \alpha\}$, $\underline{\theta}^{(m,\ell)} = \underline{e}_\ell$, and
(b) For $\ell \in \{\alpha+1, \ldots, k\}$, every element of $\underline{\theta}^{(m,\ell)}$ is nonzero.

Proof: Part (a) is a simple consequence of the structure of the generator matrix of the code. We will prove part (b) by contradiction. Suppose $\theta_\alpha^{(m,\ell)} = 0$, for some $\ell \in \{\alpha+1, \ldots, k\}$. Then $\underline{\gamma}^{(m,\ell)}$ is a linear combination of only the first $(\alpha - 1)$ columns of $\mathbf{G}^{(m)}$. This implies,

$$\underline{\gamma}^{(m,\ell)} \subseteq \text{colspace}\left[\underline{\gamma}^{(m,1)} \cdots \underline{\gamma}^{(m,\alpha-1)}\right]. \tag{81}$$

This clearly violates Property 3, thus leading to a contradiction. ■

E. Proof of Nonexistence

We now present the main theorem of this section, namely, the nonachievability proof. This proof is based on the necessity of alignment of the interference components, and independence of the desired components in the vectors passed for repair of a failed node. The proof, in essence, shows that the conditions of interference alignment necessary for exact repair of systematic nodes, coupled with the MDS property of the code, over-constrain the system, and lead to alignment in the desired components as well.

We begin with a toy example that will serve to illustrate the proof technique. Consider the case when $[n = 7,\ k = 5,\ d = 6]$, which falls under the parameter range of interest as $d < 2k - 3$. It follows from (5) that $(\alpha = d - k + 1 = 2,\ B = k\alpha = 10)$. Consider the vectors passed by the parity nodes 6 and 7 for the repair of the systematic nodes 3 and 4. From Property 2, the vectors passed by nodes 6 and 7 for the repair of node 3 must be aligned along components 4 and 5 [Fig. 11(a)]. Also from Property 2, the vectors passed by nodes 6 and 7 for the repair

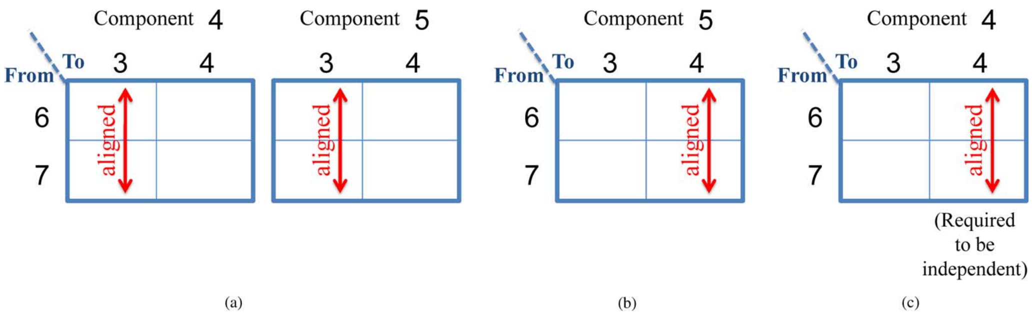


Fig. 11. Toy-example, with parameters $[n = 7,\ k = 5,\ d = 6]$, to illustrate the proof of nonexistence.

of node 4 must be aligned along component 5 [Fig. 11(b)]. It is shown that the structure of the code established in Properties 4 and 5 along with the above alignment conditions forces alignment in component 4 (desired component) during repair of node 4 [Fig. 11(c)]. This is in contradiction to the assertion of Property 2 with respect to the requirement of linear independence of the desired components.

Theorem 11: Linear, exact-repair MSR codes achieving the cut-set bound on the repair bandwidth do not exist for $d < 2k-3$ in the absence of symbol extension (i.e., when $\beta = 1$).

Proof: Recall that achieving the cut-set bound on the repair bandwidth in the absence of symbol extension gives $d = k - 1 + \alpha$. For the parameter regime $d < 2k-3$ under consideration, we get $k \geq \alpha+3$. Furthermore, since $\alpha > 1$, we have $n \geq k+2$ (as $n \geq d+1 = k+\alpha$).[7] Hence the system contains at least $(\alpha+3)$ systematic nodes and at least two parity nodes.

We use Property 4 to express the generator matrix of any parity node, say node m, in the form:

$$\mathbf{G}^{(m)} = \begin{bmatrix} G_1^{(m)} \\ \vdots \\ G_\alpha^{(m)} \\ H_{\alpha+1}\Lambda_{\alpha+1}^{(m)} \\ \vdots \\ H_k\ \Lambda_k^{(m)} \end{bmatrix}.$$

In this proof, we will use the notation $A \prec B$ to indicate that the matrices A and B are scalar multiples of each other, i.e., $A = \kappa B$ for some nonzero scalar κ and write $A \not\prec B$ to indicate that matrices A and B are not scalar multiples of each other.

We will restrict our attention to components $(\alpha+2)$ and $(\alpha+3)$. First, consider repair of systematic node $(\alpha+1)$. By the interference alignment property, Property 2

$$\underline{\gamma}_{\alpha+2}^{(k+1,\alpha+1)} \prec \underline{\gamma}_{\alpha+2}^{(k+2,\alpha+1)} \tag{82}$$

$$\text{i.e.,} \quad G_{\alpha+2}^{(k+1)}\ \underline{\theta}^{(k+1,\alpha+1)} \prec G_{\alpha+2}^{(k+2)}\ \underline{\theta}^{(k+2,\alpha+1)} \tag{83}$$

$$\Rightarrow H_{\alpha+2}\ \Lambda_{\alpha+2}^{(k+1)}\ \underline{\theta}^{(k+1,\alpha+1)} \prec H_{\alpha+2}\ \Lambda_{\alpha+2}^{(k+2)}\ \underline{\theta}^{(k+2,\alpha+1)} \tag{84}$$

$$\Rightarrow \quad \Lambda_{\alpha+2}^{(k+1)}\ \underline{\theta}^{(k+1,\alpha+1)} \prec \Lambda_{\alpha+2}^{(k+2)}\ \underline{\theta}^{(k+2,\alpha+1)}, \tag{85}$$

where, (85) uses the nonsingularity of $H_{\alpha+2}$ (which is a consequence of Property 1).

We will use the notation $\Theta^{(*,*)}$ to denote an $(\alpha \times \alpha)$ diagonal matrix, with the elements on its diagonal as the respective elements in $\underline{\theta}^{(*,*)}$. Observing that the matrices $\Lambda_*^{(*)}$ are diagonal matrices, we rewrite (85) as

$$\Lambda_{\alpha+2}^{(k+1)}\Theta^{(k+1,\alpha+1)} \prec \Lambda_{\alpha+2}^{(k+2)}\Theta^{(k+2,\alpha+1)}. \tag{86}$$

Similarly, alignment conditions on the $(\alpha+3)$th component in the vectors passed for repair of systematic node $(\alpha+1)$ give

$$\Lambda_{\alpha+3}^{(k+2)}\Theta^{(k+2,\alpha+1)} \prec \Lambda_{\alpha+3}^{(k+1)}\Theta^{(k+1,\alpha+1)} \tag{87}$$

and those on the $(\alpha+3)$th component in the vectors passed for repair of systematic node $(\alpha+2)$ give

$$\Lambda_{\alpha+3}^{(k+1)}\Theta^{(k+1,\alpha+2)} \prec \Lambda_{\alpha+3}^{(k+2)}\Theta^{(k+2,\alpha+2)}. \tag{88}$$

Taking a product of the respective terms on the left and right sides of (86), (87), and (88), we get

$$\begin{aligned}&\Lambda_{\alpha+2}^{(k+1)}\Theta^{(k+1,\alpha+1)}\Lambda_{\alpha+3}^{(k+2)}\Theta^{(k+2,\alpha+1)}\Lambda_{\alpha+3}^{(k+1)}\Theta^{(k+1,\alpha+2)} \prec \\ &\Lambda_{\alpha+2}^{(k+2)}\Theta^{(k+2,\alpha+1)}\Lambda_{\alpha+3}^{(k+1)}\Theta^{(k+1,\alpha+1)}\Lambda_{\alpha+3}^{(k+2)}\Theta^{(k+2,\alpha+2)}.\end{aligned} \tag{89}$$

Observe that in (89), the matrices $\Lambda_*^{(*)}$ and $\Theta^{(*,*)}$ are nonsingular, diagonal matrices. As a consequence, the matrices in (89) can be rearranged to yield:

$$\begin{aligned}&\Theta^{(k+1,\alpha+1)}\Lambda_{\alpha+3}^{(k+2)}\Theta^{(k+2,\alpha+1)}\Lambda_{\alpha+3}^{(k+1)}\Lambda_{\alpha+2}^{(k+1)}\Theta^{(k+1,\alpha+2)} \\ &\prec \Theta^{(k+1,\alpha+1)}\Lambda_{\alpha+3}^{(k+2)}\Theta^{(k+2,\alpha+1)}\Lambda_{\alpha+3}^{(k+1)}\Lambda_{\alpha+2}^{(k+2)}\Theta^{(k+2,\alpha+2)}.\end{aligned} \tag{90}$$

By the nonsingularity of the matrices involved, this leads to:

$$\Lambda_{\alpha+2}^{(k+1)}\Theta^{(k+1,\alpha+2)} \prec \Lambda_{\alpha+2}^{(k+2)}\Theta^{(k+2,\alpha+2)}. \tag{91}$$

This is clearly in contradiction to Property 2, which mandates linear independence of the desired components in vectors passed for repair of systematic node $(\alpha+2)$:

$$H_{\alpha+2}\Lambda_{\alpha+2}^{(k+1)}\underline{\theta}^{(k+1,\alpha+2)} \not\prec H_{\alpha+2}\Lambda_{\alpha+2}^{(k+2)}\underline{\theta}^{(k+2,\alpha+2)} \tag{92}$$

$$\text{i.e.,} \quad \Lambda_{\alpha+2}^{(k+1)}\Theta^{(k+1,\alpha+2)} \not\prec \Lambda_{\alpha+2}^{(k+2)}\Theta^{(k+2,\alpha+2)}. \tag{93}$$

■

[7] As discussed previously in Section I, $\alpha = 1$ corresponds to a trivial scalar MDS code; hence, we omit this case from consideration.

VII. Explicit MSR Codes for $d = k + 1$

In this section, we give an explicit construction of a high-rate MSR code for the parameter set $[n,\ k,\ d = k + 1]$, capable of repairing any failed node with a repair bandwidth equal to that given by the cut-set bound. This parameter set is relevant since

(a) the total number of nodes n in the system can be arbitrary (and is not constrained to be equal to $d + 1$), making the code pertinent for real-world distributed storage systems where it is natural that the number of nodes may go up or down: in due course of time, new nodes may be added to the system, or multiple nodes may fail or exit the system. For example, in peer-to-peer systems, individual nodes are free to enter and leave at will.,

(b) $k + 1$ is the smallest value of the parameter d that offers a reduction in repair bandwidth, making the code suitable for networks with low connectivity.

The code is constructed for $\beta = 1$, i.e., the code does not employ any symbol extension. All subsequent discussion in this section will implicitly assume $\beta = 1$.

For most values of the parameters $[n,\ k,\ d]$, $d = k + 1$ falls under $d < 2k - 3$ regime, where we have shown (Section VI) that exact repair is not possible. This mandates the repair to be functional in nature. Recall (from Section I-B-I) that under functional repair, a failed node is replaced by a new node such that following replacement, the resulting system continues to possess the data-reconstruction and optimal repair properties. Thus, following replacement, only the nodal generator matrix of the failed node undergoes a change without any loss of data or any compromise in the optimality of the system. This means that, the end-users can continue to connect to any k nodes to recover the entire data, and failed nodes can be repaired by connecting to *any* d nodes, and downloading the minimum amount of data [meeting the bound in (15)]. The code presented here performs a special type of functional repair, which we term as "approximately-exact" repair, where a part of the data stored in the nodes is exactly repaired and the other part is functionally repaired.

When repair is not exact, a nodal generator matrix is liable to change after a repair process. Thus, for the code construction presented in this section, we drop the global kernel viewpoint and refer directly to the symbols stored or passed. As a build up to the code construction, we first inspect the trivial case of $d = k$. In this case, the cut-set lower bound on repair bandwidth is given by

$$d \geq k = B. \tag{94}$$

Thus the parameter regime $d = k$ mandates the repair bandwidth to be no less than the message size B, and has the remaining parameters satisfying

$$(\alpha = 1,\ B = k)\,. \tag{95}$$

An MSR code for these parameters is necessarily an $[n,\ k]$ scalar MDS code. Thus, in this code, node i stores the symbol

$$\left(\underline{p}_i^t\,\underline{u}\right) \tag{96}$$

where $\underline{u}$ is a k-length vector containing all the message symbols, and $\{\underline{p}_i\}_{i=1}^n$ is a set of k-length vectors such that any arbitrary k of the n vectors are linearly independent. Upon failure of a node, the replacement node can connect to any arbitrary $d = k$ nodes and download one symbol each, thereby recovering the entire message from which the desired symbol can be extracted.

When $d = k + 1$, the cut-set bound (5) gives

$$(\alpha = d - k + 1 = 2,\ B = \alpha k = 2k)\,. \tag{97}$$

Let the $2k$ message symbols be the elements of the $2k$-dimensional column vector

$$\begin{bmatrix} \underline{u}_1 \\ \underline{u}_2 \end{bmatrix}$$

where $\underline{u}_1$ and $\underline{u}_2$ are k-length column vectors. In the case of $d = k + 1$, a code analogous to the $d = k$ code would have node i storing the two symbols:

$$\left(\underline{p}_i^t\,\underline{u}_1,\ \ \underline{p}_i^t\,\underline{u}_2\right). \tag{98}$$

Maintaining the code as in (98), after one or more node repairs, necessitates *exact* repair of any failed node. Since in this regime, exact repair is not possible for most values of the parameters, we allow an auxiliary component in our code, as described below.

Under our construction, the symbols stored in the nodes are initialized as in (98). On repair of a failed node, the code allows for an auxiliary component in the second symbol. Thus, under this code, the two symbols stored in node i, $1 \leq i \leq n$, are

$$(\underbrace{\underline{p}_i^t\,\underline{u}_1,\qquad \underline{p}_i^t\,\underline{u}_2}_{\text{Exact component}}\ +\ \underbrace{\underline{r}_i^t\,\underline{u}_1}_{\text{Auxiliary component}}) \tag{99}$$

where $\underline{r}_i$ is a k-length vector corresponding to the auxiliary component. Further, the value of $\underline{r}_i$ is allowed to alter every time node i undergoes repair. Hence we term this repair process as *approximately-exact repair*. For a better understanding, the system can be viewed as analogous to a Z-channel; this is depicted in Fig. 12, where the evolution of a node through successive repair operations is shown. In the latter half of this section, we shall see that the reconstruction and repair properties of the code hold irrespective of the values of the set of vectors $\{\underline{r}_i\}_{i=1}^n$. It is for this reason that in spite of the repair not being exact, the code always retains the reconstruction and repair properties.

We now proceed to a formal description of the code construction.

A. Code Construction

Let $\{\underline{p}_i\}_{i=1}^n$ be a set of k-length vectors such that any arbitrary k of the n vectors are linearly independent. Further, let $\{\underline{r}_i\}_{i=1}^n$ be a set of k-length vectors initialized to arbitrary values. Unlike $\{\underline{p}_i\}$, the vectors $\{\underline{r}_i\}$ do not play a role either in reconstruction or in repair. In our code, node i $(1 \leq i \leq n)$ stores the two symbols:

$$\left(\underline{p}_i^t\,\underline{u}_1,\ \ \underline{p}_i^t\,\underline{u}_2 + \underline{r}_i^t\,\underline{u}_1\right). \tag{100}$$

Upon failure of a node, the exact component, as the name suggests, is exactly repaired. However, the auxiliary component may undergo a change. The net effect is what we term as *approximately-exact repair*.

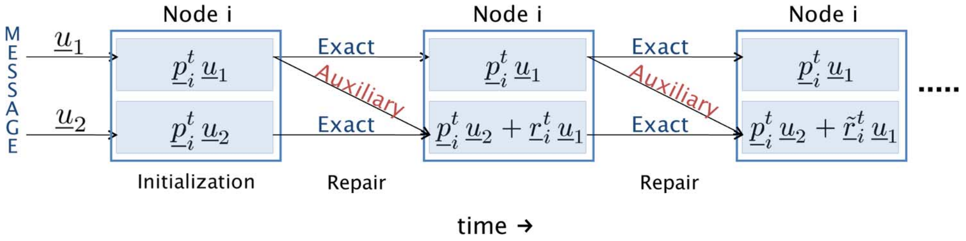
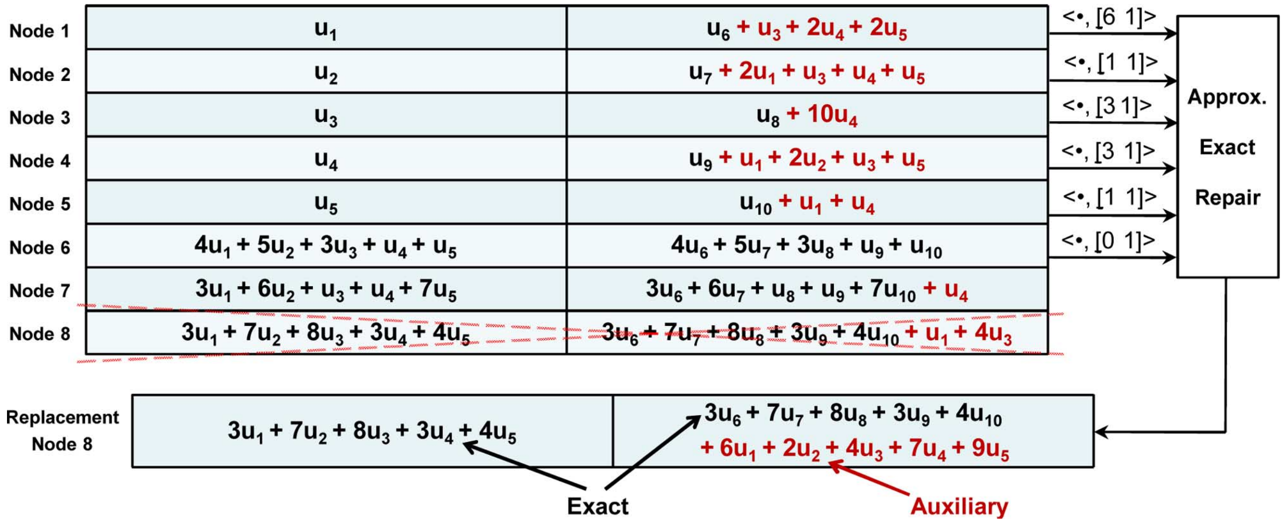

MESSAGE
$\underline{u}_1$
$\underline{u}_2$
Node i
$\underline{p}_i^t\,\underline{u}_1$
$\underline{p}_i^t\,\underline{u}_2$
Initialization
Exact
Auxiliary
Exact
Repair
Node i
$\underline{p}_i^t\,\underline{u}_1$
$\underline{p}_i^t\,\underline{u}_2 + \underline{r}_i^t\,\underline{u}_1$
Exact
Auxiliary
Exact
Repair
Node i
$\underline{p}_i^t\,\underline{u}_1$
$\underline{p}_i^t\,\underline{u}_2 + \tilde{\underline{r}}_i^t\,\underline{u}_1$
.....
time →

Fig. 12. Evolution of a node through multiple repairs in the MSR $d = k + 1$ code.



Node 1	u_1	$u_6 + u_3 + 2u_4 + 2u_5$
Node 2	u_2	$u_7 + 2u_1 + u_3 + u_4 + u_5$
Node 3	u_3	$u_8 + 10u_4$
Node 4	u_4	$u_9 + u_1 + 2u_2 + u_3 + u_5$
Node 5	u_5	$u_{10} + u_1 + u_4$
Node 6	$4u_1 + 5u_2 + 3u_3 + u_4 + u_5$	$4u_6 + 5u_7 + 3u_8 + u_9 + u_{10}$
Node 7	$3u_1 + 6u_2 + u_3 + u_4 + 7u_5$	$3u_6 + 6u_7 + u_8 + u_9 + 7u_{10} + u_4$
Node 8	$3u_1 + 7u_2 + 8u_3 + 3u_4 + 4u_5$	$3u_6 + 7u_7 + 8u_8 + 3u_9 + 4u_{10} + u_1 + 4u_3$
Replacement Node 8	$3u_1 + 7u_2 + 8u_3 + 3u_4 + 4u_5$	$3u_6 + 7u_7 + 8u_8 + 3u_9 + 4u_{10}$ $+ 6u_1 + 2u_2 + 4u_3 + 7u_4 + 9u_5$

Fig. 13. Sample MSR $d = k + 1$ code for the parameters $[n = 8,\ k = 5,\ d = 6]$, $(\beta = 1,\ \alpha = 2,\ B = 10)$, over $\mathbb{F}_{11}$. Also depicted is the repair of node 8, assisted by helper nodes 1 to 6.

The code is defined over the finite field $\mathbb{F}_q$ of size q. As discussed subsequently in Section VII-B2, any field finite field of size $q \geq n$ suffices for this construction.

Example: Fig. 13 depicts a sample code construction over $\mathbb{F}_{11}$ for the parameters $[n = 8,\ k = 5,\ d = 6]$ with $\beta = 1$ giving $(\alpha = 2,\ B = 10)$. Here,

$$\begin{bmatrix} \underline{p}_1^t \\ \vdots \\ \underline{p}_8^t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 4 & 5 & 3 & 1 & 1 \\ 3 & 6 & 1 & 1 & 7 \\ 3 & 7 & 8 & 3 & 4 \end{bmatrix}$$

$$\begin{bmatrix} \underline{r}_1^t \\ \vdots \\ \underline{r}_8^t \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 2 & 2 \\ 2 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 10 & 0 \\ 1 & 2 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 4 & 0 & 0 \end{bmatrix}.$$

The two theorems below show that the code described above is an $[n,\ k,\ d = k+1]$ MSR code by establishing respectively, the reconstruction and the repair properties of the code.

Theorem 12 (Reconstruction, i.e., MDS Property): In the code presented, all the B message symbols can be recovered by a data collector connecting to any arbitrary k nodes.

Proof: Due to symmetry we assume (without loss of generality) that the data collector connects to the first k nodes. Then the data collector obtains access to the $2k$ symbols stored in the first k nodes:

$$\left\{ \underline{p}_i^t\,\underline{u}_1, \quad \underline{p}_i^t\,\underline{u}_2 + \underline{r}_i^t\,\underline{u}_1 \right\}_{i=1}^{k}. \tag{101}$$

By construction, the vectors $\{\underline{p}_i\}_{i=1}^{k}$ are linearly independent, allowing the data collector to recover the first message vector $\underline{u}_1$. Next, the data collector subtracts the effect of $\underline{u}_1$ from the second term. Finally, in a manner analogous to the decoding of $\underline{u}_1$, the data collector recovers the second message vector $\underline{u}_2$. ■

Theorem 13 (Node Repair): In the code presented, *approximately* exact repair of any failed node can be achieved by connecting to an arbitrary subset of $d\ (= k+1)$ of the remaining $(n-1)$ nodes. Furthermore, the resulting code retains the data-reconstruction and the node-repair properties.

Proof: Due to symmetry, it suffices to consider the case where helper nodes $\{1, \ldots, k+1\}$ assist in the repair of a failed node f, $f \notin \{1, \ldots, k+1\}$. The two symbols stored in node f prior to failure are

$$\left(\underline{p}_f^t\,\underline{u}_1, \quad \underline{p}_f^t\,\underline{u}_2 + \underline{r}_f^t\,\underline{u}_1 \right).$$

However, since repair is guaranteed to be only approximately-exact, it suffices for the replacement node to obtain

$$\left(\underline{p}_f^t\, \underline{u}_1, \quad \underline{p}_f^t\, \underline{u}_2 + \tilde{\underline{r}}_f^t\, \underline{u}_1\right)$$

where $\tilde{\underline{r}}_f$ is an arbitrary vector that need not be identical to $\underline{r}_f$.

The helper nodes $\{1, \ldots, k+1\}$ pass one symbol each, formed by a linear combination of the symbols stored in them. More specifically, helper node i, $1 \leq i \leq k+1$, under our repair algorithm, passes the symbol

$$\lambda_i \left(\underline{p}_i^t\, \underline{u}_1\right) + \left(\underline{p}_i^t\, \underline{u}_2 + \underline{r}_i^t\, \underline{u}_1\right). \tag{102}$$

We introduce some notation at this point. For $\ell \in \{k, k+1\}$, let P_ℓ be a $(\ell \times k)$ matrix comprising the vectors $\{\underline{p}_i^t \mid 1 \leq i \leq \ell\}$ as its ℓ rows respectively. Let R_ℓ be a second $(\ell \times k)$ matrix comprising the vectors $\{\underline{r}_i^t \mid 1 \leq i \leq \ell\}$ as its ℓ rows respectively. Further, let $\Lambda_\ell = \operatorname{diag}[\lambda_1 \cdots \lambda_\ell]$. In terms of these matrices, the $k+1$ symbols obtained by the replacement node can be written as the $(k+1)$-length vector

$$(\Lambda_{k+1} P_{k+1} + R_{k+1})\, \underline{u}_1 + (P_{k+1})\, \underline{u}_2\,. \tag{103}$$

The precise values of the scalars $\{\lambda_i\}_{i=1}^{k+1}$ are derived below.

Recovery of the First Symbol: Let $\underline{\xi}$ be the linear combination of the received symbols that the replacement node takes to recover the first symbol that was stored in the failed node, i.e., we need

$$\underline{\xi}^t\left((\Lambda_{k+1} P_{k+1} + R_{k+1})\, \underline{u}_1 + (P_{k+1})\, \underline{u}_2\right) = \underline{p}_f^t\, \underline{u}_1. \tag{104}$$

This requires elimination of $\underline{u}_2$, i.e., we need

$$\underline{\xi}^t P_{k+1} = \underline{0}^t. \tag{105}$$

To accomplish this, we first choose

$$\underline{\xi} = \begin{bmatrix} \underline{\xi}_1 \\ -1 \end{bmatrix} \tag{106}$$

and in order to satisfy (105), we set

$$\underline{\xi}_1^t = \underline{p}_{k+1}^t P_k^{-1}. \tag{107}$$

Note that the $(k \times k)$ matrix P_k is nonsingular by construction.

Now as $\underline{u}_2$ is eliminated, to obtain $\underline{p}_f^t\, \underline{u}_1$, we need

$$\underline{\xi}^t\left(\Lambda_{k+1} P_{k+1} + R_{k+1}\right) = \underline{p}_f^t \tag{108}$$

$$\Rightarrow\ \underline{\xi}_1^t\left(\Lambda_k P_k + R_k\right) = \underline{p}_f^t + \left(\lambda_{k+1}\, \underline{p}_{k+1}^t + \underline{r}_{k+1}^t\right). \tag{109}$$

Choosing $\lambda_{k+1} = 0$ and substituting the value of $\underline{\xi}_1^t$ from (107), a few straightforward manipulations yield

$$\underline{p}_{k+1}^t\, P_k^{-1}\, \Lambda_k = \left(\underline{p}_f^t - \underline{p}_{k+1}^t\, P_k^{-1}\, R_k + \underline{r}_{k+1}^t\right) P_k^{-1} \tag{110}$$

Now, choosing

$$\Lambda_k = \left(\operatorname{diag}\left[\underline{p}_{k+1}^t\ \ P_k^{-1}\right]\right)^{-1} \operatorname{diag}\left[\left(\underline{p}_f^t - \underline{p}_{k+1}^t P_k^{-1} R_k + \underline{r}_{k+1}^t\right) P_k^{-1}\right]$$

satisfies (110), thereby enabling the replacement node to exactly recover the first symbol. The nonsingularity of the matrix $\operatorname{diag}\left[\underline{p}_{k+1}^t\ P_k^{-1}\right]$ used here is justified as follows. Consider

$$\left(\underline{p}_{k+1}^t\ P_k^{-1}\right) P_k = \underline{p}_{k+1}^t. \tag{111}$$

Now, if any element of the vector $\left(\underline{p}_{k+1}^t\ P_k^{-1}\right)$ is zero, it would imply that a linear combination of $(k-1)$ rows of P_k can yield $\underline{p}_{k+1}^t$. However, this contradicts the linear independence of every subset of k vectors in $\{\underline{p}_i\}_{i=1}^n$.

Recovery of the Second Symbol: Since the scalars $\{\lambda_i\}_{i=1}^{k+1}$ have already been utilized in the exact recovery of the first symbol, we are left with fewer degrees of freedom. This, in turn, gives rise to the presence of an auxiliary term in the second symbol.

Let $\underline{\delta}$ be the linear combination of the received symbols that the replacement node takes to obtain its second symbol $(\underline{p}_f^t\, \underline{u}_2 + \tilde{\underline{r}}_f^t\, \underline{u}_1)$, i.e., we need

$$\underline{\delta}^t\left((\Lambda_{k+1} P_{k+1} + R_{k+1})\, \underline{u}_1 + (P_{k+1})\, \underline{u}_2\right) = \underline{p}_f^t\, \underline{u}_2 + \tilde{\underline{r}}_f^t\, \underline{u}_1. \tag{112}$$

Since the vector $\tilde{\underline{r}}_f$ is allowed to take any arbitrary value, the condition in (112) is reduced to the requirement

$$\underline{\delta}^t P_{k+1} = \underline{p}_f^t. \tag{113}$$

To accomplish this, we first choose

$$\underline{\delta} = \begin{bmatrix} \underline{\delta}_1 \\ 0 \end{bmatrix} \tag{114}$$

where, in order to satisfy (113), we choose

$$\underline{\delta}_1^t = \underline{p}_f^t P_k^{-1}. \tag{115}$$

This completes the description of approximately-exact repair of the failed node f.

Finally, since both data-reconstruction and node-repair algorithms work irrespective of the values of the auxiliary components, following the repair process the code retains the data-reconstruction and node-repair properties. ■

In the example provided in Fig. 13, node 8 is repaired by downloading one symbol each from nodes 1 to 6. The linear combination coefficients used by the helper nodes are:

$$[\lambda_1 \cdots \lambda_6] = [6\ 1\ 3\ 3\ 1\ 0]\,.$$

The replacement node retains the exact part, and obtains a different auxiliary part, with $\tilde{\underline{r}}_8 = [6\ 2\ 4\ 7\ 9]\,.$

B. Analysis of the Code

1) Approximately-Exact Nature of Repair: The MSR code presented in this section consists of an exact part which is exactly repaired, along with an auxiliary part which is only *functionally* repaired. Thus this code does not enjoy the complete convenience of exact-repair (see Section I-B-I), however, it continues to possess important features that are associated with exact-repair, as described below.

The nonexact nature of repair does not lead to any loss of data or any compromise in the optimality of the system in terms of the download required for reconstruction and node repair. Moreover, the encoding vectors $\{\underline{p}_i\}_{i=1}^n$ associated with the nodes remain unchanged across any number of repairs. This specific structure of the code that is maintained throughout[8] permits the design of efficient decoding algorithms exploiting this structure. For instance, one can choose the set of vectors $\{\underline{p}_i\}_{i=1}^n$ from a Vandermonde or a Cauchy matrix, and employ the special properties of these matrices, such as fast matrix multiplication, for efficient data reconstruction and node repair.

2) Field Size Requirement: The sole restriction on the size q of the finite field $\mathbb{F}_q$ of the code arises from the construction of the set of vectors $\{\underline{p}_i\}_{i=1}^n$ such that every subset of k vectors are linearly independent. For instance, these vectors can be chosen from the rows of an $(n \times k)$ Vandermonde matrix or an $(n \times k)$ Cauchy matrix, in which case any finite field of size $q \geq n$ or $q \geq n + k$ respectively will suffice.

VIII. Conclusions

This paper considers the problem of constructing MDS regenerating codes achieving the cut-set bound on repair bandwidth, and presents four major results. First, the construction of an explicit code, termed the MISER code, that is capable of performing data reconstruction as well as optimal exact repair of the systematic nodes, is presented. The construction is based on the concept of interference alignment. The repair algorithm of the MISER code also minimizes the number of disk reads and computations required at the helper nodes during repair of systematic nodes. Moreover, repair can be performed by reading contiguous memory locations at the helper nodes by using a simple interleaving scheme. A shortening technique for MSR codes is also provided. Second, we show that interference alignment is, in fact, necessary to enable exact repair in an MSR code. Thirdly, using the necessity of interference alignment as a stepping stone, several properties that every exact-repair MSR code must possess, are derived. It is then shown that these properties over-constrain the system in the absence of symbol extension for $d < 2k - 3$, leading to the nonexistence of any linear, exact-repair MSR code in this regime. Finally, an explicit MSR code for $d = k+1$, suited for networks with low connectivity, is presented. This is the first explicit high-rate MDS regenerating code in the literature. Furthermore, this code does not impose any restriction on the total number of nodes n in the system.

Appendix
Proof of Theorem 3: Reconstruction in the MISER Code

Proof: The reconstruction property is equivalent to showing that the $(B \times B)$ matrix, obtained by columnwise concatenation of the generator matrices of the k nodes to which the data collector connects, is nonsingular. We denote this $(B \times B)$ matrix by D_1. The proof proceeds via a series of linear, elementary row and column transformations of D_1, obtaining new $(B \times B)$ matrices at each intermediate step, and the nonsingularity of the matrix obtained at the end of this process will establish the nonsingularity of D_1.

Since we need to employ a substantial amount of notation here, we will make the connection between any notation that we introduce here with the notation employed in the example presented in Section V-A. This example provided the MISER code construction for the case $k = \alpha = 3$, with the scalar selection $\epsilon = 2$; we will track the case of reconstruction (Section V-A-III, case (d)) when the data collector connects to the first systematic node (node 1), and the first two parity nodes (nodes 4 and 5).

Let $\delta_1, \ldots, \delta_p$ be the p parity nodes to which the data collector connects. Let $\omega_1, \ldots, \omega_{k-p}$ $(\omega_1 < \cdots < \omega_{k-p})$ be the $k - p$ systematic nodes to which the data collector connects, and $\Omega_1, \ldots, \Omega_p$ $(\Omega_1 < \cdots < \Omega_p)$ be the p systematic nodes to which it does *not* connect. In terms of this notation, the matrix D_1 is

$$D_1 = \left[\mathbf{G}^{(\omega_1)} \cdots \mathbf{G}^{(\omega_{k-p})}\, \mathbf{G}^{(\delta_1)} \cdots \mathbf{G}^{(\delta_p)}\right]. \tag{116}$$

Clearly, the sets $\{\omega_1, \ldots, \omega_{k-p}\}$ and $\{\Omega_1, \ldots, \Omega_p\}$ are disjoint. In the example, the notation corresponds to $p = 2$, $\delta_1 = 4$, $\delta_2 = 5$, $\omega_1 = 1$, $\Omega_1 = 2$ and $\Omega_2 = 3$.

Since the data collector can directly obtain the $(k - p)\alpha$ symbols stored in the $(k - p)$ systematic nodes it connects to, the corresponding components, i.e., components $\omega_1, \ldots, \omega_{k-p}$, are eliminated from D_1. Now, reconstruction is possible if the $(p\alpha \times p\alpha)$ matrix D_2 is nonsingular, where D_2 is given by

$$\begin{aligned} D_2 &= \left[\mathbf{G'}^{(\delta_1)} \quad \mathbf{G'}^{(\delta_2)} \cdots \quad \mathbf{G'}^{(\delta_p)}\right] \\ &= \begin{bmatrix} G_{\Omega_1}^{(\delta_1)} & G_{\Omega_1}^{(\delta_2)} & \cdots & G_{\Omega_1}^{(\delta_p)} \\ \vdots & \vdots & \ddots & \vdots \\ G_{\Omega_p}^{(\delta_1)} & G_{\Omega_p}^{(\delta_2)} & \cdots & G_{\Omega_p}^{(\delta_p)} \end{bmatrix}. \end{aligned} \tag{117}$$

The (6×6) matrix B_1 in the example corresponds to the matrix D_2 here.

The remaining proof uses certain matrices having specific structure. These matrices are defined in Table I, along with their values in the case of the example.

Note first that $\tilde{S}$, being a submatrix of the Cauchy matrix Ψ, is nonsingular. Further, note the following relations between the matrices:

$$T_{a,b}\, \tilde{S}^{-1} = E_{a,b} \tag{118}$$

and

$$\tilde{T}_{a,b}\, \tilde{S}^{-1} = \tilde{E}_{a,b}. \tag{119}$$

We begin by permuting the columns of D_2. Group the Ω_1th columns of $\{\mathbf{G'}^{(\delta_m)} \mid 1 \leq m \leq p\}$ as the first p columns of D_3, followed by Ω_2th columns of $\{\mathbf{G'}^{(\delta_m)} \mid 1 \leq m \leq p\}$ as the next p columns, and so on. Thus, column number Ω_i of $\mathbf{G'}^{(\delta_m)}$ moves to the position $p \times (i - 1) + m$. Next, group the ω_1th columns of $\{\mathbf{G'}^{(\delta_m)} \mid 1 \leq m \leq p\}$ and append this group to the already permuted columns, followed by the ω_2th columns, and so on. Thus, column number ω_i of $\mathbf{G'}^{(\delta_m)}$ moves to the position $p^2 + p \times (i-1) + m$. Let D_3 be the $(p\alpha \times p\alpha)$ matrix obtained

[8] Unlike a general functional repair code, where the coefficients of the replacement node may be arbitrary.

TABLE I
NOTATION: MATRICES USED IN THE PROOF OF THEOREM 3

Matrix	Dimension	Value	In the Example
S	$\alpha \times p$	$[S]_{i,j} = \psi_i^{(\delta_j)} \qquad \forall i, j$	$S = \begin{bmatrix} \psi_1^{(4)} & \psi_1^{(5)} \\ \psi_2^{(4)} & \psi_2^{(5)} \\ \psi_3^{(4)} & \psi_3^{(5)} \end{bmatrix}$
$\tilde{S}$	$p \times p$	$[\tilde{S}]_{i,j} = \psi_{\Omega_i}^{(\delta_j)} \qquad \forall i, j$	$\tilde{S} = \begin{bmatrix} \psi_2^{(4)} & \psi_2^{(5)} \\ \psi_3^{(4)} & \psi_3^{(5)} \end{bmatrix} = \Psi_2$
$T_{a,b}$	$\alpha \times p$	a^{th} row as $[\psi_{\Omega_b}^{(\delta_1)} \ \dots \ \psi_{\Omega_b}^{(\delta_p)}]$, all other elements 0	$T_{1,2} = \begin{bmatrix} \psi_3^{(4)} & \psi_3^{(5)} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$
$\tilde{T}_{a,b}$	$p \times p$	a^{th} row as $[\psi_{\Omega_b}^{(\delta_1)} \ \dots \ \psi_{\Omega_b}^{(\delta_p)}]$, all other elements 0	$\tilde{T}_{1,2} = \begin{bmatrix} \psi_3^{(4)} & \psi_3^{(5)} \\ 0 & 0 \end{bmatrix}$
$E_{a,b}$	$\alpha \times p$	Element at position (a, b) as 1, all other elements 0	$E_{1,2} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$
$\tilde{E}_{a,b}$	$p \times p$	Element at position (a, b) as 1, all other elements 0	$\tilde{E}_{1,2} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$

after these permutations. The (6×6) matrix B_2 in the example, corresponds to the matrix D_3 here.

Next, we note that there are α groups with p columns each in D_3. The component-wise grouping of the rows in the parent matrix D_2 induces a natural grouping in D_3, with its rows grouped into p groups of α rows each. Thus D_3 can be viewed as a block matrix, with each block of size $(\alpha \times p)$, and the dimension of D_3 being $(p \times \alpha)$ blocks. Now, in terms of the matrices defined in Table I, the matrix D_3 can be written as

$$D_3 = \left[\begin{array}{cccc|ccc} \epsilon S & T_{\Omega_2,1} & \cdots & T_{\Omega_p,1} & T_{\omega_1,1} & \cdots & T_{\omega_{k-p},1} \\ T_{\Omega_1,2} & \epsilon S & \cdots & T_{\Omega_p,2} & T_{\omega_1,2} & \cdots & T_{\omega_{k-p},2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ T_{\Omega_1,p} & T_{\Omega_2,p} & \cdots & \epsilon S & T_{\omega_1,p} & \cdots & T_{\omega_{k-p},p} \end{array}\right]. \tag{120}$$

Next, as the data collector can perform any linear operation on the columns of D_3, we multiply the last $(k - p)$ block-columns (i.e., blocks of p columns each) in D_3 by $\tilde{S}^{-1}$ (while leaving the other block-columns unchanged). Using (118), the resulting $(p\alpha \times p\alpha)$ matrix is

$$D_4 = \left[\begin{array}{cccc|ccc} \epsilon S & T_{\Omega_2,1} & \cdots & T_{\Omega_p,1} & E_{\omega_1,1} & \cdots & E_{\omega_{k-p},1} \\ T_{\Omega_1,2} & \epsilon S & \cdots & T_{\Omega_p,2} & E_{\omega_1,2} & \cdots & E_{\omega_{k-p},2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ T_{\Omega_1,p} & T_{\Omega_2,p} & \cdots & \epsilon S & E_{\omega_1,p} & \cdots & E_{\omega_{k-p},p} \end{array}\right]. \tag{121}$$

The (6×6) matrix B_3 in the example, corresponds to the matrix D_4 here.

Observe that in the block-columns ranging from $p + 1$ to α of the matrix D_4, every individual column has exactly one nonzero element. The message symbols associated with these columns of D_4 are now available to the data collector and their effect on the rest of the encoded symbols can be subtracted out to get the following $(p^2 \times p^2)$ matrix

$$D_5 = \begin{bmatrix} \epsilon\tilde{S} & \tilde{T}_{2,1} & \cdots & \tilde{T}_{p,1} \\ \tilde{T}_{1,2} & \epsilon\tilde{S} & \cdots & \tilde{T}_{p,2} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{T}_{1,p} & \tilde{T}_{2,p} & \cdots & \epsilon\tilde{S} \end{bmatrix}. \tag{122}$$

The matrix D_5 here, is the (4×4) matrix B_4 in the example. This is equivalent to reconstruction in the MISER code with the parameter k equal to p when a data collector is attempting data recovery from the p parity nodes. Hence, general decoding algorithms for data collection from the parity nodes alone can also be applied, as in the present case, where data collection is done partially from systematic nodes and partially from parity nodes. The decoding procedure for this case is provided below.

In the example detailed in case (c) of Section V-A-III, where the data collector connects to all three parity nodes, is related to this general case with $p = 3$, $\tilde{S} = \Psi_3$ and $D_5 = C_2$. We will track this case in the sequel.

The data collector multiplies each of the p block-columns in D_5 by $\tilde{S}^{-1}$. From (119), the resultant $(p^2 \times p^2)$ matrix is

$$D_6 = \begin{bmatrix} \epsilon I_p & \tilde{E}_{2,1} & \tilde{E}_{3,1} & \cdots & \tilde{E}_{p,1} \\ \tilde{E}_{1,2} & \epsilon I_p & \tilde{E}_{3,2} & \cdots & \tilde{E}_{p,2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{E}_{1,p} & \tilde{E}_{2,p} & \tilde{E}_{3,p} & \cdots & \epsilon I_p \end{bmatrix}. \tag{123}$$

The (9×9) matrix C_3 in the example, corresponds to the matrix D_6 here.

For $i = 1, \dots, p$, the ith column in the ith block-column contains exactly one nonzero element (which is in the ith row of the ith block-row). It is evident that message symbols corresponding to these columns are now available to the data collector, and their effect can be subtracted from the remaining symbols. This intermediate matrix corresponds to the (6×6) matrix C_4 in the example. Next we rearrange the resulting matrix by first placing the ith column of the jth block-column adjacent to the jth column of the ith block-column and repeating the same procedure for rows to get a $\big((p^2 - p) \times (p^2 - p)\big)$ matrix D_7 as

$$D_7 = \begin{bmatrix} \epsilon & 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & \epsilon & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \epsilon & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \epsilon & \cdots & 0 & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \epsilon & 1 \\ 0 & 0 & 0 & 0 & \cdots & 1 & \epsilon \end{bmatrix}. \tag{124}$$

This is a block diagonal matrix which is nonsingular since $\epsilon^2 \neq 1$. Thus the remaining message symbols can be recovered by decoding them in pairs. ■

References

[1] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz, "Pond: The Oceanstore prototype," in *Proc. 2nd USENIX Conf. File and Storage Technologies (FAST)*, 2003, pp. 1–14.

[2] R. Bhagwan, K. Tati, Y. C. Cheng, S. Savage, and G. M. Voelker, "Total recall: System support for automated availability management," in *Proc. 1st Conf. Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

[3] Wuala—Secure Online Storage [Online]. Available: http://www.wuala.com

[4] A. G. Dimakis, P. B. Godfrey, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," in *Proc. 26th IEEE International Conf. Computer Communications (INFOCOM)*, Anchorage, May 2007, pp. 2000–2008.

[5] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, 2010.

[6] Y. Wu, A. G. Dimakis, and K. Ramchandran, "Deterministic regenerating codes for distributed storage," in *Proc. 45th Ann. Allerton Conf. Control, Computing, and Communication*, Urbana-Champaign, Sep. 2007.

[7] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Interference alignment as a tool in network coding as applied to distributed storage," in *National Conf. Communications (NCC)*, Chennai, Jan. 2010.

[8] Y. Wu and A. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Seoul, Jul. 2009, pp. 2276–2280.

[9] K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *Proc. 47th Ann. Allerton Conf. Communication, Control, and Computing*, Urbana-Champaign, Sep. 2009, pp. 1243–1249.

[10] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Explicit codes minimizing repair bandwidth for distributed storage," in *Proc. IEEE Information Theory Workshop (ITW)*, Cairo, Jan. 2010.

[11] D. Cullina, A. G. Dimakis, and T. Ho, "Searching for minimum storage regenerating codes," in *Proc. 47th Annu. Allerton Conf. Communication, Control, and Computing*, Urbana-Champaign, Sep. 2009.

[12] Y. Wu, "Existence and construction of capacity-achieving network codes for distributed storage," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 277–288, 2010.

[13] K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Explicit and optimal exact-regenerating codes for the minimum-bandwidth point in distributed storage," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Austin, Jun. 2010, pp. 1938–1942.

[14] C. Suh and K. Ramchandran, "Exact-repair MDS code construction using interference alignment," *IEEE Trans. Inf. Theory*, vol. 57, no. 3, pp. 1425–1442, Mar. 2011.

[15] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.

[16] B. Gastón and J. Pujol, Double Circulant Minimum Storage Regenerating Codes 2010, arXiv:1007.2401 [cs.IT].

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

[18] A. Duminuco and E. Biersack, "A practical study of regenerating codes for peer-to-peer backup systems," in *Proc. 29th IEEE Int. Conf. Distributed Computing Systems (ICDCS)*, Jun. 2009, pp. 376–384.

[19] V. R. Cadambe, S. A. Jafar, and H. Maleki, Distributed Data Storage With Minimum Storage Regenerating Codes—Exact and Functional Repair are Asymptotically Equally Efficient arXiv:1004.4299 [cs.IT].

[20] C. Suh and K. Ramchandran, On the Existence of Optimal Exact-Repair MDS Codes for Distributed Storage arXiv:1004.4663 [cs.IT].

[21] V. Cadambe and S. Jafar, "Interference alignment and spatial degrees of freedom for the k user interference channel," *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3425–3441, Aug. 2008.

[22] M. Maddah-Ali, A. Motahari, and A. Khandani, "Communication over MIMO X channels: Interference alignment, decomposition, and performance analysis," *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3457–3470, Aug. 2008.

[23] D. S. Bernstein*, Matrix Mathematics: Theory, Facts, and Formulas With Application to Linear Systems Theory*. : Princeton University Press, 2005.

[24] F. MacWilliams and N. Sloane*, The Theory of Error-Correcting Codes, Part I*. : North-Holland, 1977.

[25] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Enabling node repair in any erasure code for distributed storage," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Jul. 2011, arXiv:1008.0064 [cs.IT].

Nihar B. Shah is currently a Ph.D. student at the Department of Electrical Engineering and Computer Science at the University of California at Berkeley. He received his M.E. degree from the Indian Institute of Science (IISc), Bangalore in 2010. He is a recipient of the Prof. S.V.C. Aiya medal for the best master-of-engineering student in the ECE Department at IISc. His research interests include coding and information theory, algorithms, and statistical inference.

K. V. Rashmi is currently pursuing her Ph.D. degree in the Department of Electrical Engineering and Computer Science at the University of California at Berkeley. She received her M.E. degree from the Indian Institute of Science (IISc), Bangalore in 2010. Her research interests include coding theory, information theory, networks, communications and signal processing, with a current focus on coding for distributed data storage and network coding.

P. Vijay Kumar (S'80–M'82–SM'01–F'02) received the B.Tech. and M.Tech. degrees from the Indian Institutes of Technology (Kharagpur and Kanpur), and the Ph.D. degree from the University of Southern California (USC) in 1983, all in electrical engineering. From 1983–2003 he was on the faculty of the EE-Systems Department at USC. Since 2003 he has been on the faculty of the Indian Institute of Science, Bangalore and also holds the position of adjunct research professor at USC. His current research interests include codes for distributed storage, distributed function computation, sensor networks and space-time codes for MIMO and cooperative communication networks. He is a Fellow of the IEEE and an ISI highly-cited author. He is co-recipient of the 1995 IEEE Information Theory Society prize paper award as well as of a best paper award at the DCOSS 2008 Conference on Sensor Networks.

Kannan Ramchandran (F'05) received the Ph.D. degree in 1993 from Columbia University, New York. He is currently a Professor of Electrical Engineering and Computer Science at the University of California at Berkeley, where he has been since 1999. Prior to that, he was with the University of Illinois at Urbana-Champaign from 1993 to 1999, and was at AT&T Bell Laboratories from 1984 to 1990. He is a Fellow of the IEEE and has won numerous awards including the Eli Jury thesis award at Columbia, a couple of Best Paper awards from the IEEE Signal Processing Society, a Hank Magnusky Scholar award at Illinois, an Okawa Foundation Research Prize at Berkeley, an Outstanding Teaching Award from the EECS Department at UC Berkeley, and has coauthored several best student paper awards at conferences and workshops. His current research interests include distributed signal processing and coding for wireless systems, coding for distributed storage, peer-to-peer networking and video content delivery, security, and multiuser information and communication theory.