Distributed Storage Codes through Hadamard Designs

Dimitris S. Papailiopoulos and Alexandros G. Dimakis
Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089
Email: {papailio, dimakis}@usc.edu

Abstract—In distributed storage systems that employ erasure coding, the issue of minimizing the total repair bandwidth required to exactly regenerate a storage node after a failure arises. This repair bandwidth depends on the structure of the storage code and the repair strategies used to restore the lost data. Minimizing it requires that undesired data during a repair align in the smallest possible spaces, using the concept of interference alignment (IA). Here, a points-on-a-lattice representation of the symbol extension IA of Cadambe et al. provides cues to perfect IA instances which we combine with fundamental properties of Hadamard matrices to construct a new storage code with favorable repair properties. Specifically, we build an explicit \((k+2, k)\) storage code over \(\mathbb{GF}(3)\), whose single systematic node failures can be repaired with bandwidth that matches exactly the theoretical minimum. Moreover, the repair of single parity node failures generates at most the same repair bandwidth as any systematic node failure. Our code can tolerate any single node failure and any pair of failures that involves at most one systematic failure.

I. INTRODUCTION

The demand for large scale data storage has increased significantly in recent years with applications demanding seamless storage, access, and security for massive amounts of data. When the deployed nodes of a storage network are individually unreliable, as is the case in modern data centers, or peer-to-peer networks, redundancy through erasure coding can be introduced to offer reliability against node failures. However, increased reliability does not come for free: the encoded representation needs to be maintained posterior to node erasures. To maintain the same redundancy when a storage node leaves the system, a new node has to join the array, access some existing nodes, and regenerate the contents of the departed node. This problem is known as the Code Repair Problem \([3], [1]\).

The interest in the code repair problem, and specifically in designing repair optimal \((n, k)\) erasure codes, stems from the fact that there exists a fundamental minimum repair bandwidth needed to regenerate a lost node that is substantially less than the size of the encoded data object. MDS erasure storage codes have generated particular interest since they offer maximum reliability for a given storage capacity; such an example is the EvenOdd construction \([2]\). However, most practical solutions for storage use existing off-the-shelf erasure codes that are repair inefficient: a single node repair generates network traffic equal to the size of the entire stored information.

Designing repair optimal MDS codes, i.e., ones achieving the minimum repair bandwidth bound that was derived in \([3]\), seems to be challenging especially for high rates \(\frac{k}{n} \geq \frac{1}{2}\). Recent works by Cadambe et al. \([1]\) and Suh et al. \([12]\) used the symbol extension IA technique of Cadambe et al. \([4]\) to establish the existence, for all \(n, k\), of asymptotically optimal MDS storage codes, that come arbitrarily close to the theoretic minimum repair bandwidth. However, these asymptotic schemes are impractical due to the arbitrarily large file size and field size that they require. Explicit and practical designs for optimal MDS storage codes are constructed roughly for rates \(\frac{k}{n} \leq \frac{1}{3}\) \([11], [13]\), and most of them are based upon the concept of interference alignment. Interestingly, as of now no explicit MDS storage code constructions exist with optimal repair properties for the high data rate regime\([1]\).

Our Contribution: In this work we introduce a new high-rate, explicit, \((k+2, k)\) storage code over \(\mathbb{GF}(3)\). Our storage code exploits fundamental properties of Hadamard designs and perfect IA instances pronounced by the use of a lattice representation for the symbol extension IA of Cadambe et al. \([4]\). This representation gives hints for coding structures that allow exact instead of asymptotic alignment. Our code exploits these structures and achieves perfect IA without requiring the file size or field size to scale to infinity. Any single systematic node failure can be repaired with bandwidth matching the theoretic minimum and any single parity node failure generates (at most) the same repair bandwidth as any systematic node repair. Our code has two parities but cannot tolerate any two failures: the form presented here can tolerate any single failure and any pair of failures that involves at most one

1During the submission of this manuscript, two independent works appeared that constructed MDS codes of arbitrary rate that can optimally repair their systematic nodes, see \([14], [15]\).
systematic data

\begin{tabular}{|c|c|}
\hline
systematic node & systematic data \\
\hline
1 & f_1 \\
\vdots & \vdots \\
k & f_k \\
\hline
parity node & parity data \\
\hline
\end{tabular}

Fig. 1. A \((k+2, k)\) Coded Storage Array.

systematic node failure. Here, in contrast to MDS codes, slightly more than \(k\), that is, \(k(1 + \frac{1}{2k})\), encoded pieces are required to reconstruct the file object.

II. DISTRIBUTED STORAGE CODES WITH 2 PARITY NODES

In this section, we consider the code repair problem for storage codes with 2 parity nodes. Let a file of size \(M = kN\) denoted by the vector \(f \in \mathbb{F}^{kN}\) be partitioned in \(k\) parts \(f = [f_1^T \ldots f_k^T]^T\), each of size \(N\). We wish to store this file with rate \(\frac{N}{k}\) across \(k\) systematic and 2 parity storage units each having storage capacity \(\frac{M}{k} = N\).

To achieve this level of redundancy, the file is encoded using a \((k+2, k)\) distributed storage code. The structure of the storage array is given in Fig. 1, where \(A_i\) and \(B_i\) are \(N \times N\) matrices of coding coefficients used by the parity nodes \(a\) and \(b\), respectively, to “mix” the contents of the \(i\)th file piece \(f_i\). We observe that the code is in systematic form: \(k\) nodes store the \(k\) parts of the file and each of the 2 parity nodes stores a linear combination of the \(k\) file pieces.

To maintain the same level of redundancy when a node fails or leaves the system, the code repair process has to take place to exactly restore the lost data in a newcomer storage component. Let for example a systematic node \(i \in \{1, \ldots, k\}\) fail. Then, a newcomer joins the storage network, connects to the remaining \(k+1\) nodes, and has to download sufficient data to reconstruct \(f_i\). Observe that the missing piece \(f_i\) exists as a term of a linear combination only at each parity node, as seen in Fig. 1. To regenerate it, the newcomer has to download from the parity nodes at least the size of what was lost, i.e., \(N\) linearly independent data elements. The downloaded contents from the parity nodes can be represented as a stack of \(N\) equations

\[\begin{bmatrix} p_i^{(a)} \\ p_i^{(b)} \end{bmatrix} \triangleq \begin{bmatrix} (A_i f_i + d_i)^T \\ (B_i f_i + s_i)^T \end{bmatrix} + \sum_{j=1, j \neq i}^{k} f_j \]

where \(p_i^{(a)} \in \mathbb{F}^N\) are the equations downloaded from parity nodes \(a\) and \(b\) respectively. Here, \(V^{(a)}_i, V^{(b)}_i \in \mathbb{F}^{N \times N}\) denote the repair matrices used to mix the parity contents. Retrieving \(f_i\) from (I) is equivalent to solving an underdetermined set of \(N\) equations in the \(kN\) unknowns of \(f\), with respect to only the \(N\) desired unknowns of \(f\). However, this is not possible due to the additive interference components that corrupt the desired information in the received equations. These terms are generated by the undesired unknowns \(f_j, j \neq i\), as noted in (I). Additional data need to be downloaded from the systematic nodes, which will “replicate” the interference terms and will be subtracted from the downloaded equations. To erase a single interference term, a download of a basis of equations that generates the corresponding interference term, say \[\begin{bmatrix} (A_i V^{(a)}_i)^T \\ (B_i V^{(b)}_i)^T \end{bmatrix} f_j\], suffices. Eventually, when all undesired terms are subtracted, a full rank system of \(N\) equations in \(N\) unknowns \[\begin{bmatrix} (A_i V^{(a)}_i)^T \\ (B_i V^{(b)}_i)^T \end{bmatrix} f_i\] has to be formed. Thus, it can be proven that the repair bandwidth to exactly regenerate systematic node \(i\) is given by

\[\gamma_i = N + \sum_{j=1, j \neq i}^{k} \text{rank} \left[ \begin{bmatrix} A_i V^{(a)}_i \\ B_i V^{(b)}_i \end{bmatrix} \right] \]

where the sum rank term is the aggregate of interference dimensions. Interference alignment plays a key role since the lower the interference dimensions are, the less repair data need to be downloaded. We would like to note that the theoretical minimum repair bandwidth of any node for optimal \((k+2, k)\) MDS codes is exactly \((k+1)\frac{N}{k}\), i.e. half of the remaining contents; this corresponds to each interference spaces having rank \(\frac{N}{k}\). This is also true for the systematic parts of non-MDS codes, as long as they have the same problem parameters that were discussed in the beginning of this section, and all the coding matrices have full rank \(N\). An abstract example of a code repair instance for a \((4, 2)\) storage code is given in Fig. 2, where interference terms are marked in red.

To minimize the repair bandwidth \(\gamma_i\), we need to carefully design both the storage code and the repair matrices.

2Our latest work expands Hadamard designs to construct 2-parity MDS codes that can optimally repair any systematic or parity node failure [16].

3\(\mathbb{F}\) denotes the finite field over which all operations are performed.
In the following, we provide a 2-parity code that achieves optimal systematic and near optimal parity repair.

### III. A New Storage Code

We introduce a \((k + 2, k)\) storage code over \(\mathbb{GF}(3)\), for file sizes \(M = k2^k\), with coding matrices

\[
A_i = I_N, \quad B_i = X_i, \quad \text{for } i \in \{1, \ldots, k\},
\]

where \(N = 2^k\), \(X_i = I_{2^k-1} \otimes \text{blkdiag} \left( I_{\frac{N}{2^k}}, -I_{\frac{N}{2^k}} \right) \), and \(i \in \{1, \ldots, k\} \). In Fig. 3, we give the coding matrices of the \((5, 3)\) version of the code.

**Theorem 1:** The code in (2) has optimally repairable systematic nodes and its parity nodes can be repaired by generating as much repair bandwidth as a systematic repair does. It can tolerate any single node failure, and any pair of failures that contains at most one systematic failure. Moreover, to reconstruct the file at most \(k + \frac{1}{2}\) coded blocks are required.

In the following, we present the tools that we use in our derivations. Then, in Sections V and VI we prove Theorem 1.

### IV. Dots-on-a-Lattice and Hadamard Designs

Optimality during a systematic repair, requires interference spaces collapsing down to the minimum of \(\frac{N}{\Delta}\), out of the total \(N\), dimensions. At the same time, useful data equations have to span \(N\) dimensions. For the constructions presented here, we consider that the same repair matrix is used by both parities, i.e., \(V_i^{(1)} = V_i^{(2)} = V_i\). Hence, for the repair of systematic node \(i \in \{1, \ldots, k\}\) we optimally require

\[
\text{rank}(V_i X_j V_i^\top) = \frac{N}{2},
\]

for all \(j \in \{1, \ldots, k\} \setminus i\), and at the same time

\[
\text{rank}(V_i X_j V_i^\top) = N.
\]

The key ingredient of our approach that eventually provides the above is Hadamard matrices.

To motivate our construction, we start by briefly discussing the repair properties of the asymptotic coding schemes of [11], [12]. Consider a 2-parity MDS storage code that requires file sizes \(M = k2^k\), i.e., \(N = 2^k\). Its \(N \times N\) diagonal coding matrices \(\{X_s\}_{s=1}^k\) have i.i.d. elements drawn uniformly at random from some arbitrarily large finite field \(\mathbb{F}\). During the repair of a systematic node \(i \in \{1, \ldots, k\}\), the repair matrix \(V_i\) that is used by both parity nodes to mix their contents, has as columns the \(\frac{N}{\Delta} = 2^k\) elements of the set

\[
\mathcal{V}_i = \left\{ \prod_{s=1, s \neq i}^k X_s x_s^\top w : x_s \in \{0, \ldots, \Delta - 1\} \right\}.
\]

Then, we define a map \(\mathcal{L}\) from vectors in the set \(\prod_{s=1}^k X_s x_s^\top w : x_s \in \mathbb{Z}\) to points on the integer lattice

\[
\mathcal{L}(X_i V_i) = \{x_s e_i : x_s \in \{0, \ldots, \Delta - 1\}\},
\]

and \(\mathcal{L}(X_i V_i) = \{x_s e_i : x_s \in \{0, \ldots, \Delta - 1\}\}\), respectively. In Fig. 2, we give an illustrative example for \(k = 3\), and \(\Delta = 2\).

**Remark 1:** Observe how matrix multiplication of \(X_i\) and elements of \(V_i\) manifests itself through the dots-on-a-lattice representation: the product of \(X_i\) with the elements of \(V_i\) shifts the corresponding arrangement of dots along the \(x_i\)-axis, i.e., the \(x_i\)-coordinate of the initial points gets increased by one.

Asymptotically optimal repair of node \(i\) is possible due to the fact that interference spaces asymptotically align

\[
\text{rank}(V_i X_j V_i) = \left\lfloor \frac{N}{\Delta} \right\rfloor = \left\lfloor \frac{2^k}{\Delta} \right\rfloor = \left\lfloor \frac{2^k}{2^k-1} \right\rfloor \Delta = o(\Delta^k),
\]

and useful spaces span \(N\) dimensions, that is,\n
\[
\text{rank}(V_i X_j V_i) = |\mathcal{L}(V_i) \cup \mathcal{L}(X_i V_i)| = 2^k - 1,
\]

with arbitrarily high probability for sufficiently large field sizes.

The question that we answer here is the following: How can we design the coding and the repair matrices such that \(i)\) exact interference alignment is possible and \(ii)\) the full
X_1 = \text{diag} \left( \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix} \right), \quad X_2 = \text{diag} \left( \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix} \right), \quad X_3 = \text{diag} \left( \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix} \right)
The above holds due to each element of $\mathcal{H}_N$ being associated with a unique power tuple. Then, the columns of $[V_i, X, V_j]$ are exactly the elements of $\mathcal{H}_N$, since

$$\mathcal{L}(V_i) \cup \mathcal{L}(X, V_j) = \left\{ \sum_{s=1}^{k} x_{i,s} e_i; \; x_i \in \{0,1\} \right\} \cup \left\{ \sum_{s=1}^{k} x_{i,s} e_i; \; x_i \in \{0,1\} \right\} = \mathcal{L}(H_N).$$

Moreover, the set of columns in $V_i$ are identical to the set of columns of $X_j V_j$, i.e., $\mathcal{L}(V_i) = \mathcal{L}(X_j V_j)$, for $j \neq i$, due to Lemmata 1 and 2. Therefore, the interference spaces span $\frac{N}{T}$ dimensions, which is the theoretical minimum, and the desired data space during any systematic node repair is full-rank, since it has as columns all columns of $H_N$.

Hence, we conclude that a single systematic node of the code can be repaired with bandwidth $(k + 1) \frac{N}{T} = \frac{k+1}{M}.$ In Fig. 4, we depict a $(6, 4)$ code of our construction, along with the illustration of the repair spaces.

B. Parity repairs

Here, we prove that a single parity node repair generates at most the repair bandwidth of a single systematic repair. Let parity node $a$ fail. Then, observe that if the newcomer uses the $N \times N$ repair matrix $V^{(b)}_a = X_1$ to multiply the contents of parity node $b$, then it downloads $X_1 \left( \sum_{f=1}^{k} X_1 f_j \right) = f_1 + \sum_{i=2}^{k} X_1 X_i f_i.$ Observe, that the component corresponding to systematic part $f_i$ appears the same in the linear combination stored at the lost parity. By Lemma 2, each of the remaining blocks, $X_1 X_i f_i$, share exactly $\frac{N}{T}$ indices with equal elements to the same $\frac{N}{T}$ indices of $X_i f_i$, which was lost, for any $i \in \{2, \ldots, k\}$. This is due to the fact that the diagonal elements of matrices $X_1 X_i$ and $X_i$ are the elements of some two columns of $H_N$. Therefore, the newcomer has to download from systematic node $j \in \{2, \ldots, k\}$, the $\frac{N}{T}$ entries that parity $a$’s component $X_j f_j$ differs from the term $X_1 X_i f_i$ of the downloaded linear combination. Hence, the first parity can be repaired with bandwidth at most $N + (k - 1) \frac{N}{T} = (k + 1) \frac{N}{T}$.

The repair of parity node $b$ can be performed in the same manner.

VI. ERASURE RESILIENCE

Our code can tolerate any single node failure and any two failures with at most one of them being a systematic one. A double systematic and parity node failure can be treated by first reconstructing the lost systematic node from the remaining parity, and then reconstructing the lost parity from all the systematic nodes. However, two simultaneous systematic node failures cannot be tolerated. Consider for example the corresponding matrix when we connect to nodes $\{1, \ldots, k - 2\}$ and both parities:

$$\begin{bmatrix}
I_N & \ldots & 0_{N \times N} & 0_{N \times N} & 0_{N \times N} \\
\vdots & \ddots & \vdots & \vdots & \vdots \\
0_{N \times N} & \ldots & I_N & 0_{N \times N} & 0_{N \times N} \\
X_1 & \ldots & X_k & X_{k-2} & X_{k-1} \\
X_{k+1} & \ldots & X_{2k-1} & X_{2k-2} & X_{2k-1} \\
\end{bmatrix}$$

The rank of this $kN \times kN$ matrix is $(k - 1)N + \frac{N}{T}$ due to the submatrix $X_i X_j$ having rank $\frac{N}{T}$. For these cases, an extra download of $\frac{N}{T}$ equations is required to decode the file, i.e., an aggregate download of $kN + \frac{N}{T}$ equations, or $k + \frac{1}{2}$ encoded pieces.

REFERENCES

[1] The Coding for Distributed Storage wiki http://tinyurl.com/storagecoding

[2] M. Blaum, J. Brady, J. Bruck, and J. Menon, “EVENODD: An efficient scheme for tolerating double node failures in raid architectures,” in IEEE Trans. on Computers, 1995.

[3] A. G. Dimakis, P. G. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, “Network coding for distributed storage systems,” in IEEE Trans. on Information Theory, vol. 56, pp. 4539 – 4551, Sep. 2010.

[4] V. R. Cadambe and S. A. Jafar, “Interference alignment and the degrees of freedom for the K user interference channel,” IEEE Trans. on Information Theory, vol. 54, pp. 3425–3441, Aug. 2008.

[5] Y. Wu and A. G. Dimakis, “Reducing repair traffic for erasure coding-based storage via interference alignment,” in Proc. IEEE Int. Symp. on Information Theory (ISIT), Seoul, Korea, Jul. 2009.

[6] D. Cullina, A. G. Dimakis, and T. Ho, “Searching for minimum storage regenerating codes,” In Allerton Conf. on Control, Comp., and Comm., Urbana-Champaign, IL, September 2009.

[7] K.V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran “Exact regenerating codes for distributed storage,” In Allerton Conf. on Control, Comp., and Comm., Urbana-Champaign, IL, September 2009.

[8] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, “Explicit codes minimizing repair bandwidth for distributed storage,” in Proc. IEEE ITW, Jan. 2010.

[9] C. Suh and K. Ramchandran, “Exact regeneration codes for distributed storage repair using interference alignment,” in Proc. 2010 IEEE Int. Symp. on Information Theory (ISIT), Seoul, Korea, Jun. 2010.

[10] Y. Wu, “A construction of systematic MDS codes with minimum repair bandwidth,” Submitted to IEEE Transactions on Information Theory, Aug. 2009. Preprint available at http://arxiv.org/abs/0910.2486.

[11] V. Cadambe, S. Jafar, and H. Maleki, “Distributed data storage with minimum storage regenerating codes - exact and functional repair are asymptotically equally efficient,” in 2010 IEEE Intern. Workshop on Wireless Network Coding (WINC), Apr. 2010.

[12] C. Suh and K. Ramchandran, “On the existence of optimal exact-repair MDS codes for distributed storage,” Apr. 2010. Preprint available online at http://arxiv.org/abs/1004.4663.

[13] K. Rashmi, N. B. Shah, and P. V. Kumar, “Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction,” submitted to IEEE Transactions on Information Theory, Preprint available online at http://arxiv.org/pdf/1005.4178.

[14] I. Tamo, Z. Wang, and J. Bruck “MDS Array Codes with Optimal Rebuilding,” to appear at ISIT 2011, preprint available at http://arxiv.org/abs/1103.3377.

[15] V. R. Cadambe, C. Huang, and J. Li, “Permutation codes: optimal exact-repair of a single failed node in MDS code based distributed storage systems,” to appear at ISIT 2011, preprint available at http://newport.eecs.uci.edu/~vcadambe/permutations.pdf.

[16] D. S. Papailiopoulos, A. G. Dimakis, and V. R. Cadambe, “Repair optimal erasure codes through hadamard designs,” preprint available at http://www.scf.usc.edu/~papailio/