

Explicit and Optimal Codes for Distributed Storage

K. V. Rashmi[†], Nihar B. Shah[†], P. Vijay Kumar[†], Kannan Ramchandran[#]

[†] Dept. of ECE, Indian Institute Of Science, Bangalore. {rashmikv, nihar, vijay}@ece.iisc.ernet.in

[#] Dept. of EECS, University of California, Berkeley. kannanr@eecs.berkeley.edu

Abstract—In the distributed storage coding problem we consider, data is stored across n nodes in a network, each capable of storing α symbols. It is required that the complete data can be reconstructed by downloading data from any k nodes. There is also the key additional requirement that a failed node be regenerated by connecting to any d nodes and downloading β symbols from each of them. Our goal is to minimize the repair bandwidth $d\beta$. In this paper we provide explicit constructions for several parameter sets of interest.

I. INTRODUCTION

We consider the setting of data storage across n nodes in a distributed manner. A data collector (DC) should be able to obtain the entire data by connecting to any k out of the n nodes and downloading all the data stored in them. This is termed as *reconstruction*. The total data to be stored is of size B units (symbols).

One more key requirement of any distributed storage system is that, when a node fails, it is replaced by a new node by downloading data from the existing nodes. This process is termed as *regeneration*, and the amount of download which a new node performs is called *repair bandwidth*. Since the data downloaded for repair consumes a considerable amount of network bandwidth, it is of interest to minimize the repair bandwidth.

A naive method of satisfying the reconstruction property is to choose $B = k$ and use an $[n, k]$ Reed-Solomon type MDS code where each node stores a single symbol. We consider only linear codes, which will forbid extraction of partial data from nodes. This forces the repair bandwidth to be B itself. Hence we store multiple symbols in each node, which facilitates extraction of partial information and helps in reducing the repair bandwidth.

Each node is allowed to store α symbols. When a node fails, the new node replacing it has a capacity of α symbols, and downloads β symbols each from d existing nodes as shown in Figure 1. It is clear that the minimum α required to satisfy the reconstruction property is

$$\alpha_{\min} = B/k \quad (1)$$

Authors in [1] show the importance of repair bandwidth in distributed storage systems and the throughput improvements that can be obtained by reducing the same. In [2], authors show that repair bandwidth can be reduced by storing slightly more than the minimum required in each node. They establish a tradeoff between the amount of storage in each node α and the repair bandwidth $d\beta$, and call codes meeting the tradeoff as *Regenerating Codes*. The most important points on the tradeoff

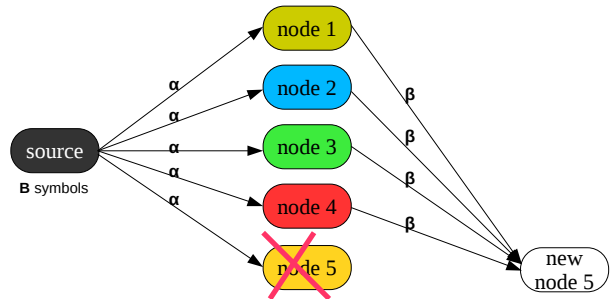


Fig. 1. An illustration of regeneration: On failure of node 5, data from nodes 1 to 4 is used to regenerate it. Here $n = 5$ and $d = 4$.

are the two extreme points - the Minimum Storage Regeneration (MSR) point and the Minimum Bandwidth Regeneration (MBR) point. They obtain the following lower bounds on the repair bandwidth, and also prove these bounds to be tight.

The MSR point has the minimum storage requirement per node, i.e

$$\alpha = \alpha_{\min} = B/k \quad (2)$$

for which the repair bandwidth is

$$\beta \geq \alpha/(d - k + 1) \quad (3)$$

Hence the advantage in repair bandwidth over the naive method is

$$\frac{d\beta}{B} = \frac{d}{k(d - k + 1)} \quad (4)$$

On the other extreme, the MBR point corresponds to the minimum repair bandwidth possible with a higher storage requirement per node. At this operating point, a new node downloads the minimum amount required, i.e

$$\alpha = d\beta \quad (5)$$

and the repair bandwidth is

$$\beta \geq \frac{B}{kd - \binom{k}{2}} \quad (6)$$

Here the advantage in repair bandwidth over the naive method turns out to be

$$\frac{d\beta}{B} = \frac{d}{kd - \binom{k}{2}} \quad (7)$$

Several additional properties like *exact regeneration* [3] and *systematic nodes* [4] have been introduced for regenerating codes in the literature. However, no explicit construction

is available for regenerating codes. Recently, authors in [6] performed a computer search and obtained a regenerating code at the MSR point for parameters $n = 5$, $k = 3$, $d = 4$. In the present paper, we summarize our explicit constructions for regenerating codes at the MSR and MBR points which achieve the lower bounds (3) and (6) respectively.

In all our constructions, we choose $\beta = 1$. At both MSR and MBR points, all data units scale linearly with β . Hence, optimum code for any β can be obtained by concatenating the optimum code for $\beta = 1$. Thus, this design choice is without loss of generality. Also, reconstruction and regeneration are performed independently on the concatenated codes thereby greatly reducing the processing and storage requirements.

The rest of the paper is organized as follows. Optimal explicit codes for the setting $d = n - 1$ for the MBR point are provided in Section II and for the MSR point in Section III. In Section IV, optimal explicit codes are provided for the MSR point for the parameter set $d = k + 1$.

II. EXPLICIT CODES FOR THE MBR POINT WITH $d = n - 1$

The MBR point has the minimum possible repair bandwidth. We choose $d = n - 1$ which corresponds to greatest reduction in repair bandwidth. And thus is suitable for applications such as distributed mail servers, where it is of importance to restore the system in the shortest possible time.

This section gives the construction of linear exact regenerating codes at the MBR point for $d = n - 1$ and any k which achieve the cutset bound (6). The code is explained in more detail along with an example in [3]. All symbols are assumed to belong to a finite field \mathbb{F}_q of size q .

The choice of $\beta = 1$ gives

$$\alpha = n - 1 \quad (8)$$

$$B = k\alpha - \binom{k}{2} \quad (9)$$

A. Code Construction

Denote the source symbols of the file by $\underline{f} = (f_0 \ f_1 \ f_2 \ \dots \ f_{B-1})^t$. Let $\theta = \frac{n(n-1)}{2}$. Let V be a $n \times \theta$ matrix with the following properties:

- 1) Each element is either 0 or 1.
- 2) Each row has exactly α 1's.
- 3) Each column has exactly two 1's.
- 4) Any two rows have exactly one intersection of 1's.

It is easy to see that V is the incidence matrix of a fully connected undirected graph with n vertices. Our construction of exact regenerating codes for the MBR point uses the above described matrix V . Consider a set of θ vectors $\{\underline{v}_1, \underline{v}_2, \dots, \underline{v}_\theta\}$ forming a $[\theta, B]$ MDS code. The vectors \underline{v}_i ($i = 1, \dots, \theta$) are of length B with the constituent elements taken from the field \mathbb{F}_q . Node j stores the symbol $\underline{v}_i^t \underline{f}$ if and only if $V(j, i) = 1$. Thus in the graph corresponding to V , vertices represent the nodes, and edges represent the vectors corresponding to the symbols stored, as illustrated in Figure 2. By property 2 of the matrix V , we get n nodes each storing $n - 1$ ($= \alpha$) symbols. Properties 3 and 4 ensure that each row intersects every other row in distinct columns. The validity of

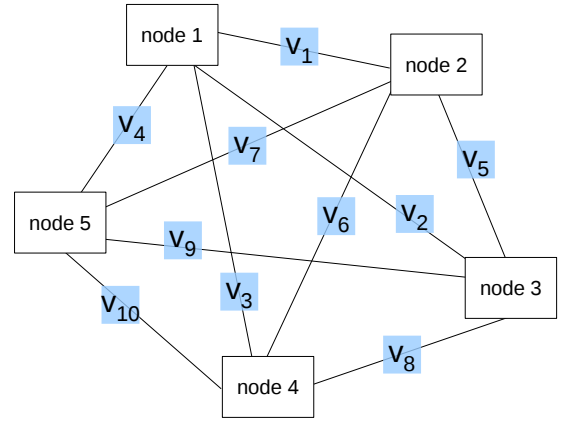


Fig. 2. An example of the MBR code with $n = 5$. The figure depicts a fully connected undirected graph with 5 vertices. Vertices represent nodes and edges represent vectors corresponding to the common symbol between two nodes.

this code as a exact regenerating code for the MBR point is as follows.

Example Let $n = 5$, $k = 3$. We get $d = n - 1 = 4$ and $\theta = 10$. Putting $\beta = 1$ gives $\alpha = 4$ and $B = 9$. The matrix V is the incidence matrix of a fully connected undirected graph with 5 vertices, as shown in Figure 2. The 5 nodes store the following symbols:

$$\begin{aligned} \text{Node 1: } & \{\underline{f}^t \underline{v}_1, \underline{f}^t \underline{v}_2, \underline{f}^t \underline{v}_3, \underline{f}^t \underline{v}_4\} \\ \text{Node 2: } & \{\underline{f}^t \underline{v}_1, \underline{f}^t \underline{v}_5, \underline{f}^t \underline{v}_6, \underline{f}^t \underline{v}_7\} \\ \text{Node 3: } & \{\underline{f}^t \underline{v}_2, \underline{f}^t \underline{v}_5, \underline{f}^t \underline{v}_8, \underline{f}^t \underline{v}_9\} \\ \text{Node 4: } & \{\underline{f}^t \underline{v}_3, \underline{f}^t \underline{v}_6, \underline{f}^t \underline{v}_8, \underline{f}^t \underline{v}_{10}\} \\ \text{Node 5: } & \{\underline{f}^t \underline{v}_4, \underline{f}^t \underline{v}_7, \underline{f}^t \underline{v}_9, \underline{f}^t \underline{v}_{10}\} \end{aligned}$$

The working of the general code is as follows.

1) *Data Reconstruction*:: The DC connects to any k out of the n storage nodes and downloads all the $k\alpha$ symbols stored. As any two rows of the matrix V intersect in only one column and any row intersects all other rows in distinct columns, out of the $k\alpha$ symbols downloaded, exactly $\binom{k}{2}$ symbols are repetitions and do not add any value. Hence the DC has $k\alpha - \binom{k}{2} = B$ distinct symbols of a B -dimensional MDS code, using which the values of the source symbols f_0, \dots, f_{B-1} can be easily obtained.

2) *Exact Regeneration*:: The matrix V provides a special structure to the code which helps in exact regeneration, where the new node replacing a failed node stores data identical to what was stored in the failed node. Properties 3 and 4 of the matrix V imply that the each of the existing $n - 1$ nodes contain one distinct symbol of the failed node. Thus exact regeneration of the failed node is possible by downloading one symbol each from the remaining $n - 1$ nodes.

B. Analysis of the Code

The minimum field size required is the minimum field size required to construct a $[\theta, B]$ MDS code. If we use a

Reed-Solomon code, the minimum field size required for our construction turns out to be $\theta (= \frac{n(n-1)}{2})$.

Given the incidence matrix \mathbf{V} of a fully connected graph with n vertices, no arithmetic operations are required for code construction and regeneration. Since regeneration is exact, one set of k nodes can be maintained as systematic, i.e. these k nodes will store the source symbols in an uncoded form. If the DC connects to this set of k nodes, no decoding is necessary.

III. EXPLICIT CODES FOR THE MSR POINT WITH $d = n - 1$

The MSR point requires the least possible storage at the nodes. We choose $d = n - 1$ since this parameter corresponds to the greatest savings in the repair bandwidth. The parameters for this point specialized to $\beta = 1$ are

$$B = k\alpha \quad (10)$$

$$\alpha = d - k + 1 \quad (11)$$

In this section an explicit linear construction is given, which achieves optimal exact regeneration of systematic nodes for $d \geq 2k - 1$. This parameter set corresponds to $k \leq \alpha$. First, we construct codes for $k = \alpha$. Codes for any $k \leq \alpha$ can be obtained by puncturing the code for $k = \alpha$. The code is explained in more detail along with the proofs and an example in [5].

A. Subspace Viewpoint for Linear Codes

Define a vector $\underline{\mathbf{f}}$ of length B consisting of the source symbols. Let

$$\underline{\mathbf{f}} = \begin{bmatrix} \underline{f}_1 \\ \vdots \\ \underline{f}_k \end{bmatrix}$$

where \underline{f}_i is a column vector of length α . Each source symbol can independently take values from \mathbb{F}_q , a finite field of size q . Hence the B source symbols can be thought of as forming a B -dimensional vector space over \mathbb{F}_q .

We construct linear codes, i.e. any stored symbol is written as $\underline{\ell}^t \underline{\mathbf{f}}$ for some column vector $\underline{\ell}$. These vectors which specify the kernels for the stored symbols define the code, and the actual symbols stored depend on the instantiation of $\underline{\mathbf{f}}$. Linear operations performed on the stored symbols are equivalent to the same operations performed on the corresponding vectors.

Since a node stores α symbols, it can be considered as storing α vectors of the code, and hence can be represented by an $\alpha \times B$ matrix. We will say that the node *stores* this matrix. Storing an $\alpha \times B$ matrix is equivalent to *storing a subspace* of dimension at most α . However, from (10) it is clear that each node must store a subspace of dimension equal to α .

For $m = 1, \dots, n$ denote the matrix stored by node m as $\mathbf{G}^{(m)} = [G_1^{(m)} \ G_2^{(m)} \ \dots \ G_k^{(m)}]$, where $G_l^{(m)}$ ($l = 1, \dots, k$) are $\alpha \times \alpha$ matrices. The α symbols actually stored by the node are $\mathbf{G}^{(m)} \underline{\mathbf{f}} = \sum_{l=1}^k G_l^{(m)} \underline{f}_l$. We will denote the j^{th} row of $G_l^{(m)}$ as $\underline{g}_{jl}^{(m)}$.

There are n storage nodes, out of which k are systematic and store α data symbols each in uncoded form. For $m =$

$1, \dots, k$, systematic node m stores the symbol set \underline{f}_m . Thus for $l = 1, \dots, k$,

$$G_l^{(m)} = \begin{cases} I_\alpha & \text{if } l = m \\ 0_\alpha & \text{if } l \neq m \end{cases} \quad (12)$$

where 0_α is $\alpha \times \alpha$ zero matrix, and I_α is $\alpha \times \alpha$ identity matrix. Hence, for any non-systematic node m , $G_l^{(m)}$ denotes the *components* along systematic node l that are stored in node m .

For regeneration of a failed systematic node, d other nodes provide one symbol each. We say that each node *passes a vector* for the regeneration of the failed node. In the vectors passed by the non-systematic nodes, the components along the existing systematic nodes constitute *interference*. We say two vectors are *aligned* if they are linearly dependent.

B. Explicit Code Construction for $k = \alpha$

Let Ψ be an $(n - k) \times \alpha$ Cauchy matrix [7] with elements drawn from \mathbb{F}_q , i.e

$$\Psi = \begin{bmatrix} \underline{\psi}^{(k+1)} \\ \vdots \\ \underline{\psi}^{(n)} \end{bmatrix} \quad (13)$$

where $\underline{\psi}^{(i)} = [\psi_1^{(i)} \ \dots \ \psi_\alpha^{(i)}]$, $i = k + 1, \dots, n$ are α -length vectors. A Cauchy matrix has the property that any submatrix is full rank.

For $m = k + 1, \dots, n$, $i, j = 1, \dots, \alpha$, set

$$\underline{g}_{ij}^{(m)} = \begin{cases} \epsilon \underline{\psi}_i^{(m)} & \text{if } i = j \\ \psi_j^{(m)} \underline{e}_i, & \text{if } i \neq j \end{cases} \quad (14)$$

where ϵ is any arbitrary value such that $\epsilon \neq 0$ and $\epsilon^2 \neq 1$. \underline{e}_i denotes an α -length unit row vector with 1 in i^{th} position and 0 elsewhere. Note that there always exists such a value for $q \geq 4$.

This choice makes the interference in the vectors passed by non-systematic nodes for the regeneration of a failed systematic node *aligned*. This enables the existing systematic nodes to cancel the interference by passing one symbol each.

Example: Take $k = \alpha = 3$, $n = 6$. This gives $d = 5$ and $B = 9$. Thus each node stores a 3×9 matrix. Let $q = 7$.

Let the first three nodes be systematic. Hence,

$$\mathbf{G}^{(1)} = [I_3 \ 0_3 \ 0_3] \quad (15)$$

$$\mathbf{G}^{(2)} = [0_3 \ I_3 \ 0_3] \quad (16)$$

$$\mathbf{G}^{(3)} = [0_3 \ 0_3 \ I_3] \quad (17)$$

Let $\Psi_3 = \begin{bmatrix} \psi_1^{(4)} & \psi_2^{(4)} & \psi_3^{(4)} \\ \psi_1^{(5)} & \psi_2^{(5)} & \psi_3^{(5)} \\ \psi_1^{(6)} & \psi_2^{(6)} & \psi_3^{(6)} \end{bmatrix}$ be a 3×3 Cauchy matrix.

The three non-systematic nodes store the matrices $\mathbf{G}^{(m)}$, $m = 4, 5, 6$, given by

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

The working of the general code is as follows.

1) *Exact Regeneration of Systematic Nodes*: Consider regeneration of systematic node \hat{l} ($\in \{1, \dots, k\}$). All non-systematic nodes participating in the regeneration pass their \hat{l}^{th} row, i.e. non-systematic node m ($\in \{k+1, \dots, n\}$) passes the vector $[g_{\hat{l},1}^{(m)} \dots g_{\hat{l},k}^{(m)}]$. Systematic node l ($l = 1, \dots, k, l \neq \hat{l}$) passes $[\underline{0} \dots \underline{0} \ e_{\hat{l}} \ \underline{0} \dots \underline{0}]$ where $e_{\hat{l}}$ is in the \hat{l}^{th} position. From (14), $g_{\hat{l},i}^{(m)}$ are all aligned along the direction of $e_{\hat{l}}$. Hence the vectors passed by other systematic nodes can be used to remove interference from the vectors passed by the non-systematic nodes participating in the regeneration.

Also from (14), $g_{\hat{l},i}^{(m)}$ are rows of the Cauchy matrix Ψ , and hence are linearly independent. Using these α linearly independent vectors, the systematic node \hat{l} can be regenerated.

2) *Exact Regeneration of Non-Systematic Nodes*: Recently, the authors in [8] have shown that our code can exactly regenerate non-systematic nodes as well, without any loss in performance.

3) *Reconstruction*: For reconstruction to be successful, the matrices stored in the k nodes to which the DC connects, when juxtaposed one below the other, should form a $B \times B$ full rank matrix. Reconstruction is trivially satisfied if the DC connects to k systematic nodes. Consider the DC connecting to p non-systematic nodes, and $k - p$ systematic nodes, $1 \leq p \leq k$. Let $\delta_1, \dots, \delta_p$ be the p non-systematic nodes to which the DC connects and let $\Omega_1, \dots, \Omega_p$ be the p systematic nodes to which the DC does *not* connect. Reconstruction is successful if and only if the $p\alpha \times p\alpha$ matrix R formed by removing the components along the systematic nodes to which DC connects in $\mathbf{G}^{(\delta_1)}, \dots, \mathbf{G}^{(\delta_p)}$ is non-singular.

$$R = \begin{bmatrix} G_{\Omega_1}^{(\delta_1)} & G_{\Omega_2}^{(\delta_1)} & \dots & G_{\Omega_p}^{(\delta_1)} \\ \vdots & \vdots & & \vdots \\ G_{\Omega_1}^{(\delta_p)} & G_{\Omega_2}^{(\delta_p)} & \dots & G_{\Omega_p}^{(\delta_p)} \end{bmatrix} \quad (18)$$

Theorem 1: R is full rank.

Proof: (Sketch) By a series of row operations and using the full rank property of the Cauchy matrix we show that R reduces to a block diagonal matrix. Each of the resulting block matrices turn out to be non-singular as $\epsilon^2 \neq 1$. ■

C. Explicit Code Construction for $k < \alpha$

For a given α , first construct the code for $k = \alpha$. The theorem given below shows the existence and construction for any $\hat{k} < \alpha$.

Theorem 2: If there exists a (k, α) linear exact regenerating code achieving the cutset bound, then there also exists a (\hat{k}, α) linear exact regenerating code for any $\hat{k} \leq k$ which achieves the cutset bound.

Proof: (Sketch) In the (k, α) code, remove the last $(k - \hat{k})\alpha$ columns from each node matrix to obtain the new node matrices of size $\alpha \times \hat{k}\alpha$. Consider only the set of first \hat{k} systematic nodes along with all the non-systematic nodes. This

forms a (\hat{k}, α) code. Assume that the data stored in the $(k - \hat{k})$ other systematic nodes is known globally. Now regeneration and reconstruction are same as in the (k, α) code. ■

D. Extension to the Case $d < n - 1$

These codes can also achieve optimal exact regeneration of systematic nodes provided that the new node replacing a failed node connects to the $k - 1$ systematic nodes along with some other $d - (k - 1)$ non-systematic nodes.

E. Analysis of the Code

The minimum field size required is the minimum field size required to construct an $(n - k) \times (d - k + 1)$ Cauchy matrix, which is $n + d - 2k + 1$.

Since the code is explicit and exact regenerating, it has low complexity.

IV. EXPLICIT CODES FOR THE MSR POINT WITH $d = k + 1$

This operating point particularly suits applications like storage in peer-to-peer systems where storage capacity available from the participating nodes is very low. In such systems, multiple node failures are quite frequent as nodes enter and exit the system at their own will. Hence the system should be capable of regenerating a failed node using only a small number of existing nodes. Also, the number of nodes in the system changes dynamically. Hence the code should work even if the number of nodes keeps varying with time.

In this section we give an explicit construction for regenerating codes at the MSR point for $d = k + 1$ and any n . This set of parameters makes the code capable of handling any number of failures provided that at least $k + 1$ nodes remain functional.

Note that by definition, if less than k nodes are functional then a part of the data will be permanently lost. If exactly k nodes are functional, then these nodes will have to pass all the information stored in them for regeneration, hence no optimization of the repair bandwidth is possible in this case.

The parameters for the MSR point specialized to $d = k + 1$ and $\beta = 1$ are

$$B = 2k \quad (19)$$

$$\alpha = 2 \quad (20)$$

A. Code construction:

Partition the source symbols into two sets: f_0, \dots, f_{k-1} , and g_0, \dots, g_{k-1} . Let $\underline{f}^t = (f_0 \ f_1 \ \dots \ f_{k-1})$, and $\underline{g}^t = (g_0 \ g_1 \ \dots \ g_{k-1})$.

Node i ($i = 1, \dots, n$) stores $(\underline{p}_i^t \underline{f}, \underline{p}_i^t \underline{g} + \underline{u}_i^t \underline{f})$ as its two symbols. We shall refer to the vectors \underline{p}_i and \underline{u}_i as the *main vector* and the *auxiliary vector* of a node respectively. The elements of the auxiliary vectors are known but can take any arbitrary values from \mathbb{F}_q . The main vectors are the ones which are actually used for reconstruction and regeneration. Let the set of main vectors \underline{p}_i ($i = 1, \dots, n$) form an $[n, k]$ MDS code over \mathbb{F}_q .

For example, consider $n = 5$, $k = 3$ and $d = 4$. We have $B = 6$ and f_0, f_1, f_2, g_0, g_1 and g_2 as the source symbols.

Let the main vectors \underline{p}_i ($i = 1, \dots, n$) form a Reed-Solomon code, with $\underline{p}_i = (1 \ \theta_i \ \theta_i^2)^t$. θ_i ($i = 1, \dots, 5$) take distinct values from \mathbb{F}_q ($q \geq 5$). We can initialize elements of \underline{u}_i ($i = 1, \dots, 5$) to any arbitrary values from \mathbb{F}_q .

B. Reconstruction:

A data collector will connect to any k nodes and download both the symbols stored in each of these nodes. The first symbols of the k nodes provide $\underline{p}_i^t \underline{f}$ at k different values of i . To solve for \underline{f} , we have k linear equations in k unknowns. Since \underline{p}_i 's form a k -dimensional MDS code, these equations are linearly independent, and can be solved easily to obtain the values of f_0, \dots, f_{k-1} .

Now, as \underline{f} and \underline{u}_i are known, $\underline{u}_i^t \underline{f}$ can be subtracted out from the second symbols of each of the k nodes. This leaves us with the values of $\underline{p}_i^t \underline{g}$ at k different values of i . Using these, values of g_0, \dots, g_{k-1} can be recovered.

Thus all B data units can be recovered by a DC which connects to any k nodes. We also see that reconstruction is possible irrespective of the values of the auxiliary vectors \underline{u}_i .

C. Regeneration:

In our construction, when a node fails, the main vector of the regenerated node will have the same value as that of the failed node, although the auxiliary vector is allowed to be different. Suppose node j fails. The node replacing it would contain $(\underline{p}_j^t \underline{f}, \underline{p}_j^t \underline{g} + \tilde{\underline{u}}_j^t \underline{f})$ where elements of $\tilde{\underline{u}}_j$ can take any arbitrary value from \mathbb{F}_q and are not constrained to be equal to those of \underline{u}_j . As the reconstruction property holds irrespective of the values of \underline{u}_j , the regenerated node along with the existing nodes has all the desired properties.

For regeneration of a failed node, some d nodes pass one symbol each, obtained by taking a linear combination of the symbols stored in them. Assume that node Λ_{d+1} fails and nodes $\Lambda_1, \dots, \Lambda_d$ are used to regenerate it, where the set $\{\Lambda_1, \dots, \Lambda_{d+1}\}$ is some subset of $\{1, \dots, n\}$, with all elements distinct.

Let a_i and b_i ($i = 1, \dots, d$) be the coefficients of the linear combination for the symbol given out by node Λ_i . Let $v_i = a_i(\underline{p}_{\Lambda_i}^t \underline{f}) + b_i(\underline{p}_{\Lambda_i}^t \underline{g} + \underline{u}_{\Lambda_i}^t \underline{f})$ be this symbol. Let δ_i and ρ_i ($i = 1, \dots, d$) be the coefficients of the linear combination used to generate the two symbols of the regenerated node. Thus the regenerated node will store

$$\left(\sum_{i=1}^d \delta_i v_i, \sum_{i=1}^d \rho_i v_i \right) \quad (21)$$

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

$$\sum_{i=1}^d \rho_i b_i \underline{p}_{\Lambda_i} = \underline{p}_{\Lambda_{d+1}} \quad (22)$$

and δ_i ($i = 1, \dots, d$) such that

$$\sum_{i=1}^d \delta_i b_i \underline{p}_{\Lambda_i} = \underline{0} \quad (23)$$

Equations (22) and (23) are sets of k linear equations in $d = k + 1$ unknowns each. Since $\underline{p}_{\Lambda_i}$'s form a k -dimensional MDS code these can be solved easily in \mathbb{F}_q . This also ensures that we can find a solution to equation (23) with all δ_i 's non-zero.

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

$$\sum_{i=1}^d \delta_i (a_i \underline{p}_{\Lambda_i} + b_i \underline{u}_{\Lambda_i}) = \underline{p}_{\Lambda_{d+1}} \quad (24)$$

i.e

$$\sum_{i=1}^d \delta_i a_i \underline{p}_{\Lambda_i} = \underline{p}_{\Lambda_{d+1}} - \sum_{i=1}^d \delta_i b_i \underline{u}_{\Lambda_i} \quad (25)$$

Equation (25) is a set of k linear equations in $d (= k + 1)$ unknowns which can be easily solved in \mathbb{F}_q . Since none of the δ_i ($i = 1, \dots, d$) are zero, the particular choice of $\underline{p}_{\Lambda_i}$'s used guarantees a solution for a_i ($i = 1, \dots, d$). Hence, regeneration of any node using any d other nodes is achieved.

D. Analysis of the Code

The minimum field size required is the minimum field size required to construct a $[n, k]$ MDS code. If we use a Reed-Solomon code, the minimum field size required for our construction is n .

Since the code is explicit, the system requires low maintenance.

REFERENCES

- [1] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright and K. Ramchandran, "Network Coding for distributed storage systems," *IEEE Proc. INFOCOM*, (Anchorage, Alaska), May 2007.
- [2] Y. Wu, A. G. Dimakis and K. Ramchandran, "Deterministic Regenerating codes for Distributed Storage," *Proc. Allerton Conf.*, Sep. 2007.
- [3] K. V. Rashmi, N. B. Shah, P. V. Kumar and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *Proc. Allerton Conference on Control, Computing and Communication*, Urbana-Champaign, September 2009.
- [4] Y. Wu and A. Dimakis, "Reducing Repair Traffic for Erasure Coding-Based Storage via Interference Alignment," *Proc. ISIT*, Jul. 2009.
- [5] N. B. Shah, K. V. Rashmi, P. V. Kumar and K. Ramchandran, "Explicit codes minimizing repair bandwidth for distributed storage," in *Proc. Information Theory Workshop*, Cairo, January 2010. Available online at [arXiv:0908.2984](https://arxiv.org/abs/0908.2984).
- [6] D. Cullina, A. G. Dimakis and Tracey Ho, "Searching for Minimum Storage Regenerating Codes," in *Proc. Allerton Conference on Control, Computing and Communication*, Urbana-Champaign, September 2009.
- [7] Dennis S. Bernstein, *Matrix mathematics: Theory, facts, and formulas with application to linear systems theory*, Princeton University Press, Princeton, NJ, p.119, 2005.
- [8] Changho Suh and K. Ramchandran, "Exact Regeneration Codes for Distributed Storage Repair Using Interference Alignment," submitted to *ISIT*, June 2010. Available online at [arXiv:1001.0107](https://arxiv.org/abs/1001.0107).