A Code Design Framework for Multi-Rack Distributed Storage

M. Ali Tebbi[†], Terence H. Chan[†], Chi Wan Sung[‡]
[†]Institute for Telecommunications Research, University of South Australia
Email: {ali.tebbi, terence.chan}@unisa.edu.au
[‡]Department of Electronic Engineering, City University of Hong Kong
Email: albert.sung@cityu.edu.hk

***Abstract*—In practical distributed storage networks, data centres house hundreds of racks, each of which contains several storage nodes. However, the majority of works in distributed storage assume a simple network model with a collection of identical storage nodes with same communication cost between the nodes. In this paper, we consider a more realistic rack model of storage network and present a code design framework for this model. Using our code construction method, node failures within a rack can be repaired locally by survived nodes in the same rack or by the other survived racks when the information content of the same rack is not sufficient to repair the failed nodes.**

I. Introduction

The amount of data being created in modern information technological infrastructures is growing at an exponentially rapid rate. These created data need to be stored, analysed and managed efficiently in data centres. However, storage nodes (e.g., a storage disk or tape) are prone to failures, which may be due to malicious attacks, nature disasters (such as fire or earthquake), hardware failures or even software glitches.

When these failures occur, data stored in the nodes may be lost. In order to store the data reliably, one must deploy data redundancy across the storage network. Data replication or mirroring is the simplest scheme which is often used, either completely or partially in conjunction with other data protection methods [1], [2]. In replication method, multiple copies of the same data are simply stored in different storage nodes. Therefore, data recovery from failures is also simple and efficient due to the direct copying of lost data from survived nodes which holds the replicas. However, this method suffers from the huge storage cost. To reduce the storage cost, another common approach is using error correcting codes (in particularly the maximum distance separable codes such as Reed-Solomon Codes) [3]. To store data using erasure codes, a data block consisting of a symbols will be encoded into b (where $b > a$) symbols. Each of the b symbols will be stored in one of the storage nodes. The objective is that as long as there are at least a surviving nodes, the original data can still be retrieved. The merit of such an erasure coding based scheme is its high storage efficiency (or equivalently, low storage cost). However, the cost to repair any failed node (measured by the amount of data being transferred during the repair process) can be quite high. In fact, it might require the reconstruction of the whole data block (even the goal is to recover only one of the symbols in the block).

This work was partially supported by a grant from Australian Research Council (DP1094571) and the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-02/08).

There is a fundamental tradeoff between the storage cost and the repair cost in a distributed storage system. In [4], Dimakis *et al.* proposed "regenerating codes" which aim to achieve the optimal tradeoffs between the two costs. One of the main observations is that repair cost can be greatly reduced if more helper nodes (or the surviving storage nodes) can participate in the repair process. In [4] (and many subsequent works thereafter), the cost to setting up connections between storage nodes to the failed node is basically ignored. This cost can in fact be significant in some cases. In this context, locally repairable codes were proposed in [5]–[8] which aim to minimise this cost (or equivalently, the number of nodes required in the repair process).

A majority of existing distributed storage network models assume very simple structure that the network itself is viewed as a collection of "identical" storage nodes and that the transmission cost between any two nodes are identical. However, in practice, this model is rarely close to the truth. A typical data centre can easily house hundreds of racks, each containing numerous storage disks. While all storage nodes (or disks here) can communicate with each other, the transmission costs in terms of latency or overheads can differ vastly. For example, the transmission latency between storage disks in the same rack is usually much smaller, when compared with the case that both disks are not in the same rack.

Recently, design for heterogeneous distributed storage (including the multi-rack models) has received a fair amount of attention. In [9], a storage network is considered in which storage nodes are divided into two groups with different communication costs. The model in [9] partially addressed the issues that the communication costs among nodes are not all equal. However, it does not fit well into a multi-rack model, where the transmission cost should depend both on where the transmitting and the receiving (or the failed) storage nodes are located.

A more realistic rack model of a distributed storage network has been investigated in [10], in which the authors considered a two-rack model. In their model, the communication cost between the nodes in the same rack is much smaller than

978-1-4799-5999-0/14/$31.00 ©2014 IEEE

between two different racks. As such, it is desirable that more data should be transmitted by nodes in the same rack of the failed node. Using information flow graph, [10] identifies if certain choice of parameters are achievable or not, leading to the characterisation of the tradeoff between storage cost and repair cost.

In the above works, information flow graph is employed to characterise the fundamental tradeoff between various system parameters (such as storage cost and repair cost). The tradeoff is asymptotic (without restriction on the size of the symbol alphabet) and functional repair is always assumed (i.e., the failed node is not required to recover exactly what it previously stored, as long as the whole storage system is still robust after repair). In this paper, we instead focus on a storage code construction framework, specifically tailored for multi-rack data storage systems. Our codes are based on the use of locally repairable codes and exact repair is assumed (i.e., any data lost in a node will be recovered exactly).

II. Distributed storage codes for racks

In this paper, we will provide a framework for constructing storage codes for multi-rack distributed storages. Consider a multi-rack data storage network with M racks each of which contains N storage nodes (or storage disks). We will call these nodes respectively

$$\left(X_{m,n},\ \forall m \in \{1,2,\ldots,M\} \text{ and } \forall n \in \{1,2,\ldots,N\}\right).$$

Abusing notations, $X_{m,n}$ will also be referred to the content stored at that particular storage node.

In this paper, we will assume that each rack has a processing unit which is directly connected to all the storage nodes in the same rack. Storage nodes in two different racks can only communicate via their respective processing units which are responsible for all the computation and communications functionalities. We further assume that the system bottleneck is at the processing unit in each rack. Therefore, in our code construction, the focus is to design distributed storage codes that minimises the communication costs to the processing units.

It is very common in realistic systems that the communication cost from a storage node to a processing unit in the same rack is much lower than the communication cost between two different processing units (and hence located in two different racks). Therefore, it is desirable and in fact critical that a failed node can be repaired "locally" using only the survived nodes within the same rack in order to keep the repair cost low. It is only for some occasional failure patterns that it will require nodes from other racks to assist in the repair process.

A. Code construction

Let $\mathbf{X}$ be an $M \times N$ matrix

$$\mathbf{X} = \begin{bmatrix} X_{1,1} & \cdots & X_{1,N} \\ \vdots & \ddots & \vdots \\ X_{M,1} & \cdots & X_{M,N} \end{bmatrix} \tag{1}$$

whose entries are from $GF(q)$. Define

$$X_{m,*} \triangleq [X_{m,1}, \ldots, X_{m,N}].$$

Here, $X_{m,n}$ is the data stored at the nth storage node in the mth rack. Thus, rack m will store the vector of symbols $X_{m,*}$. In order to repair any failures, redundancies are introduced to the system. We are considering only linear codes in this paper and that $\mathbf{X}$ must satisfying some parity check equations.

Before we present the general coding framework, we first motivate our work via a very simple example. Suppose that we require $\mathbf{X}$ to satisfy the following parity check equations.

$$\mathbf{H}X_{m,*}^\top = \mathbf{0} \quad \forall m = 1, \ldots, M \tag{2}$$

$$\sum_{m=1}^{M} \mathbf{K}X_{m,*}^\top = \mathbf{0} \tag{3}$$

where $\mathbf{H}$ and $\mathbf{K}$ are respectively $S_1 \times N$ and $S_2 \times N$ matrices.

Assume without loss of generality that some of the nodes in rack 1 have failed. Let γ be the set of failed nodes in rack 1. Let

$$b_n = X_{1,n}$$

for $n \notin \gamma$. One can now repair the failed nodes by solving the following system of linear equations

$$\mathbf{H}X_{1,*}^\top = \mathbf{0} \tag{4}$$

$$X_{1,n} = b_n, \quad \forall n \notin \gamma. \tag{5}$$

In some cases, the failure pattern γ is so severe that the above system of linear equations have no unique solution, then we can use other racks for assistance. Notice that from (3), we have

$$\mathbf{K}X_{1,*}^\top = -\sum_{m=2}^{M} \mathbf{K}X_{m,*}^\top. \tag{6}$$

We can exploit this equation in the repair process. The idea is as follows. We will assume that there are no failures in racks 2 to M. Our assumption is motivated by the observation that most racks can repair its own failures or errors locally within the rack and that there is a much lower probability that there are more than one racks whose failures cannot be repaired locally. Therefore, we only focus on the special case when there is only one rack whose failures cannot be repaired locally. In this case, we assume either other racks have no failures or have been repaired its own failures locally.

Under our assumption, for $m = 2, \cdots M$, the processing unit in rack m will compute the vector

$$\mathbf{c}_m \triangleq \mathbf{K}X_{m,*}^\top$$

and will send it to the processing unit in rack 1. With this extra information, the processing unit in rack 1 can try to repair its failed nodes by solving the following system of equations

$$\mathbf{H}X_{1,*}^\top = \mathbf{0} \tag{7}$$

$$X_{1,n} = b_n, \quad \forall n \notin \gamma \tag{8}$$

$$\mathbf{K}X_{1,*}^\top = -\sum_{m=2}^{M} \mathbf{c}_m. \tag{9}$$

The above example illustrates the basic idea of our coding scheme. However, using the above code, when failures cannot be repaired within its own rack, all other racks must be available in the repair process. In some scenarios, this is not feasible or desirable – if one of the racks is not available, then the repair process may not continue. In the following, we will present a more general coding framework which requires only a subset of racks to participate in the repair process.

Definition 1 (Multi-rack storage codes)**.** *Consider three parity check matrices* $\mathbf{H}$*,* $\mathbf{K}$ *and* $\mathbf{G}$ *over* $GF(q)$ *of respectively sizes* $S_1 \times N$*,* $S_2 \times N$ *and* $L \times M$*. The three matrices induce a storage code such that* $\mathbf{X}$ *must satisfy the following parity-check equations*

$$\mathbf{H}\mathbf{X}^\top = \mathbf{0} \tag{10}$$

$$\mathbf{K}\mathbf{X}^\top\mathbf{G}^\top = \mathbf{0}. \tag{11}$$

B. Repair Process

Now, we will explain how repair should be performed. Our proposed code supports two ways to repair. If the failure pattern (i.e., the subset of nodes that fail) in a rack is not severe, nodes will be repaired locally. Otherwise, nodes in other racks will participate in the repair process.

Assume again without loss of generality that rack 1 fails and that all other racks have no failures (or have already been repaired). Let γ be the index set for the nodes in rack 1 that fail. In other words, the values of $\{X_{1,n}, n \in \gamma\}$ are unknown to the processing unit in rack 1. Let

$$\mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix}$$

where

$$b_n = \begin{cases} X_{1,n} & \text{if } n \notin \gamma \\ 0 & \text{otherwise.} \end{cases}$$

In other words, $\mathbf{b}$ is obtained from $X_{1,*}^\top$ by replacing $X_{1,n}$ with 0 for all $n \in \gamma$. Define $\mathbf{I}_N^\beta$ as an $N \times N$ diagonal matrix such that its (n,n)th entry is 1 if $n \in \beta$ and is 0 otherwise. For simplicity, we will drop the subscript N if it is understood from the context. Then, it can be verified easily that

$$\mathbf{I}^{\bar{\gamma}} X_{1,*}^\top = \mathbf{b} \tag{12}$$

where $\bar{\gamma}$ is the complement set of γ.

Recall that

$$\mathbf{H}X_{1,*}^\top = \mathbf{0}$$

Therefore, rack 1 can repair its failed nodes locally if the following system of linear equations

$$\begin{cases} \mathbf{I}^{\bar{\gamma}} X_{1,*}^\top = \mathbf{b} \\ \mathbf{H}X_{1,*}^\top = \mathbf{0} \end{cases} \tag{13}$$

has a unique solution.

For notation simplicity, we will use $\langle \mathbf{I}^\gamma, \mathbf{H} \rangle$ to denote the vector space spanned by rows of $\mathbf{I}^\gamma$ and $\mathbf{H}$. It is clear that (13) has a unique solution if and only if

$$\dim\langle \mathbf{I}^{\bar{\gamma}}, \mathbf{H} \rangle = N. \tag{14}$$

Before we continue, we will introduce the following definitions and intermediate results.

Definition 2 (support)**.** *The support* $\lambda(\mathbf{v})$ *of a vector* $\mathbf{v} = [v_1, v_2, \ldots, v_N]$ *is a subset of* $\{1, 2, \ldots, N\}$ *such that* $i \in \lambda(\mathbf{v})$ *if and only if* $v_i \neq 0$*,* $\forall i \in \{1, 2, \ldots, N\}$*.*

Definition 3. *Consider any matrix* $\mathbf{H}$ *and vector* $\mathbf{r}$ *(such that both have* N *columns). For any* $j = 1, \ldots, N$*, let*

$$\Omega(\mathbf{H}, \mathbf{r}, j) = \{\lambda(\mathbf{h}) \setminus j \mid \mathbf{h} \in \langle \mathbf{H}, \mathbf{r} \rangle \text{ and } j \in \lambda(\mathbf{h})\}.$$

If $\mathbf{r}$ *is the zero vector, we will simply denote* $\Omega(\mathbf{H}, \mathbf{r}, j)$ *by* $\Omega(\mathbf{H}, j)$*.*

Lemma 1. *If* $\beta \in \Omega(\mathbf{H}, \mathbf{r}, j)$*, then there exists vectors* $\mathbf{y}$*,* $\mathbf{y}'$ *and* $a \in GF(q)$ *such that*

$$\mathbf{e}_j = \mathbf{y}\mathbf{H} + a\mathbf{r} + \mathbf{y}'\mathbf{I}^\beta \tag{15}$$

where $\mathbf{e}_j = [e_{j,1}, \ldots, e_{j,N}]$ *is a length* N *row vector such that*

$$e_{j,\ell} = \begin{cases} 1 & \text{if } \ell = j \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

While (14) guarantees that the failure pattern γ can be repaired within rack 1, it does not explicitly specify how the failures in the rack can be repaired and what the cost is. In the following, we will explain how this can be achieved. To illustrate the idea, let us consider the repair of a specific storage node. Assume without loss of generality that $j \in \gamma$.

Theorem 1. *Let* γ *be the index set for all failed nodes in rack 1 and* $j \in \gamma$*. If* $\beta_j \subseteq \{1, \ldots, N\}$ *satisfies the following two criteria,*

1) $\beta_j \in \Omega(\mathbf{H}, j)$*, and*
2) $\beta_j \cap \gamma = \emptyset$*,*

then there exists $c_{j,n}$ *for* $n \in \beta_j$ *such that*

$$X_{1,j} = \sum_{n \in \beta_j} c_{j,n} X_{1,n}. \tag{17}$$

Proof: By Lemma 1 and criterion 1), there exists $\mathbf{y}$ and $\mathbf{y}'$ such that

$$\mathbf{e}_j = \mathbf{y}\mathbf{H} + \mathbf{y}'\mathbf{I}^{\beta_j} \tag{18}$$

Hence,

$$\mathbf{e}_j X_{1,*}^\top = (\mathbf{y}\mathbf{H} + \mathbf{y}'\mathbf{I}^{\beta_j}) X_{1,*}^\top \tag{19}$$

$$= \mathbf{y}\mathbf{H}X_{1,*}^\top + \mathbf{y}'\mathbf{I}^{\beta_j} X_{1,*}^\top \tag{20}$$

$$= \mathbf{y}'\mathbf{I}^{\beta_j} X_{1,*}^\top \tag{21}$$

where the last equality follows from (2). Finally, let

$$[c_{j,1}, \ldots, c_{j,N}] = \mathbf{y}'\mathbf{I}^{\beta_j}.$$

As the columns of $\mathbf{I}^{\beta_j}$ indexed by $\bar{\beta}_j$ are zero, $c_{j,n} = 0$ if $n \notin \beta_j$. Therefore, we prove our theorem. ∎

Equation (17) essentially defines how to regenerate $X_{1,j}$ from $X_{1,n}$ for $n \in \beta_j$. In this case, $|\beta_j|$ symbols are transmitted to the processing unit in rack 1, which can then repair the failed node $X_{1,j}$ by (17). Clearly, the choice of β_j will affect the repair cost. It is always desirable to pick β_j such that its size is as small as possible.

In some rare cases, the failed node $X_{1,j}$ cannot be repaired completely locally in rack 1. This is equivalent to the scenario where (14) is not satisfied. In this case, other racks must participate in the repair process. To describe how this is performed, we consider again the repair of the node $X_{1,j}$ as follows.

Theorem 2. *Suppose that the node $X_{1,j}$ (i.e., the jth node in rack 1) fails. If $(\beta_j, \mu_j, \mathbf{r}_j, \tau)$ satisfies the following criteria,*

1) $\mathbf{r}_j \in \langle \mathbf{K} \rangle$
2) $\mu_j = \{n \in \{1, \ldots, N\} : r_{j,n} \neq 0\}$
3) $\beta_j \in \Omega(\mathbf{H}, \mathbf{r}_j, j)$, *and*
4) $\beta_j \cap \gamma = \emptyset$,
5) $\tau \subseteq \{1, \ldots, M\} \in \Omega(\mathbf{G}, 1)$

then there exists $c_{j,n}$ for $n \in \beta_j$ such that

$$X_{1,j} = \sum_{m \in \tau} \left(\sum_{s \in \mu_j} d_{j,m,s} X_{m,s} \right) + \sum_{n \in \beta_j} c_{j,n} X_{1,n}. \quad (22)$$

Remark: Theorem 2 implies Theorem 1 if $\tau = \mu_j = \emptyset$ and $\mathbf{r}_j$ is the zero vector.

Proof of Theorem 2: By Criterion 1), there exists a row vector $\mathbf{u}$ of length S_2 such that

$$\mathbf{r}_j = \mathbf{u}\mathbf{K}. \quad (23)$$

In addition, as $\beta_j \in \Omega(\mathbf{H}, \mathbf{r}_j, j)$ by Criterion 3, there exists row vectors $\mathbf{y}$, $\mathbf{y}'$ and $a \in GF(q)$ such that

$$\mathbf{e}_j = \mathbf{y}\mathbf{H} + \mathbf{y}'\mathbf{I}^{\beta_j} + a\mathbf{r}_j \quad (24)$$

Now, notice

$$\begin{aligned} X_{1,j} &= \mathbf{e}_j X_{1,*}^\top && (25)\\ &= (\mathbf{y}\mathbf{H} + \mathbf{y}'\mathbf{I}^{\beta_j} + a\mathbf{r}_j) X_{1,*}^\top && (26)\\ &= \mathbf{y}'\mathbf{I}^{\beta_j} X_{1,*}^\top + a\mathbf{r}_j X_{1,*}^\top && (27) \end{aligned}$$

where the last equality follows from (2).

Let

$$[c_{j,1}, \ldots, c_{j,N}] = \mathbf{y}'\mathbf{I}^{\beta_j}. \quad (28)$$

Then

$$\mathbf{y}'\mathbf{I}^{\beta_j} X_{1,*}^\top = \sum_{n \in \beta_j} c_{j,n} X_{1,n}.$$

Consequently,

$$X_{1,j} = \sum_{n \in \beta_j} c_{j,n} X_{1,n} + a\mathbf{r}_j X_{1,*}^\top \quad (29)$$

.

Let $\mathbf{f} = [f_1, \ldots, f_M]$ be a length M row vector such that

$$f_\ell = \begin{cases} 1 & \text{if } \ell = 1 \\ 0 & \text{otherwise.} \end{cases} \quad (30)$$

Then

$$X_{1,*} = \mathbf{f}\mathbf{X}$$

and hence

$$\begin{aligned} a\mathbf{r}_j X_{1,*}^\top &= a X_{1,*} \mathbf{r}_j^\top && (31)\\ &= a\mathbf{f}\mathbf{X}\mathbf{r}_j^\top. && (32) \end{aligned}$$

Since $\tau \in \Omega(\mathbf{G}, 1)$, there exists vectors $\mathbf{z}$ and $\mathbf{z}'$ such that

$$\mathbf{f} = \mathbf{z}\mathbf{G} + \mathbf{z}'\mathbf{I}^\tau. \quad (33)$$

Now, notice that $\mathbf{I}^\tau$ is a $M \times M$ matrix over $GF(q)$, as $\mathbf{G}$ has only M columns. Consequently,

$$\begin{aligned} a\mathbf{r}_j X_{1,*}^\top &= a(\mathbf{z}\mathbf{G} + \mathbf{z}'\mathbf{I}^\tau)\mathbf{X}\mathbf{r}_j^\top && (34)\\ &= a\mathbf{z}'\mathbf{I}^\tau \mathbf{X}\mathbf{r}_j^\top && (35) \end{aligned}$$

where the last equality follows from that

$$\begin{aligned} \mathbf{z}\mathbf{G}\mathbf{X}\mathbf{r}_j^\top &= \mathbf{z}\mathbf{G}\mathbf{X}\mathbf{K}^\top\mathbf{u}^\top && (36)\\ &= \mathbf{z}\left(\mathbf{G}\mathbf{X}\mathbf{K}^\top\right)\mathbf{u}^\top && (37)\\ &= 0. && (38) \end{aligned}$$

Let the matrix

$$\begin{aligned} \begin{bmatrix} d_{j,1,1} & \cdots & d_{j,1,N} \\ \vdots & \ddots & \vdots \\ d_{j,M,1} & \cdots, & d_{j,M,N} \end{bmatrix} &= a(\mathbf{z}'\mathbf{I}^\tau)^\top \mathbf{r}_j && (39)\\ &= a\mathbf{I}^\tau(\mathbf{z}')^\top \mathbf{r}_j && (40) \end{aligned}$$

as $\mathbf{I}^\tau$ is a diagonal matrix.

Finally, notice that only those rows and columns of $\mathbf{I}^\tau$ indexed by τ are nonzero and $r_{j,n}$ can be nonzero only if $n \in \mu_j$. Therefore, $d_{j,m,n}$ will be zero if either $m \notin \tau$ or $n \notin \mu_j$. Hence,

$$a\mathbf{r}_j X_{1,*}^\top = \sum_{m \in \tau} \left(\sum_{n \in \mu_j} d_{j,m,n} X_{m,n} \right). \quad (41)$$

And thus the theorem is then proved. ∎

Remark: Again, (22) can be interpreted as a repair procedure for $X_{1,j}$. The processing unit in rack m where $m \in \tau$, will retrieve $|\mu_j|$ symbols. The processing unit in rack 1, will need to retrieve $|\beta_j|$ symbols within the rack. Also, one symbol transmission is needed for the processing unit to send the recovered symbol back to the failed storage node. Finally, each rack indexed by τ will transmit 1 symbol to the processing unit in rack 1. Summing up all these transmissions, there are in total

$$|\beta_j| + |\mu_j||\tau| + 1$$

symbol transmission within racks and $|\tau|$ symbol transmissions across racks. When we need to repair multiple storage nodes (say $|\gamma|$ of them), the repair cost is not simply $|\gamma|$

times the above cost. This is because cost can be reduced if transmission of the same symbols occur.

Theorem 3. *Suppose the parameter $(\beta_j, \mu_j, \mathbf{r}_j, \tau)$ is selected in the repair for the node $X_{1,j}$ for $j \in \gamma$. Then the required total transmissions within the rack is equal to*

$$|\cup_{j\in\gamma}\beta_j| + |\tau||\cup_{j\in\gamma}\mu_j| + |\gamma|$$

and the required transmissions across the racks is

$$|\tau|\dim\langle \mathbf{r}_j, j\in\gamma\rangle.$$

Finally, we will derive the rate of the code.

Theorem 4. *Let $\mathbf{H}$, $\mathbf{K}$, and $\mathbf{G}$ be respectively $S_1\times N$, $S_2\times N$, and $L\times M$ matrices. Then the rate of the generalised rack model code is*

$$R \geq \frac{MN - MS_1 - LS_2}{MN}.$$

Equality holds if rows in $\mathbf{H}$ and $\mathbf{K}$ are linearly independent, and $\mathbf{G}$ is a full rank matrix.

III. Example

In this section we will provide an explicit binary code example for a multiple-rack distributed storage network. Consider a storage network with $M = 5$ racks, each of which contains $N = 8$ storage nodes. Then, the codeword matrix $\mathbf{X}$ in (1) will be a 5×8 matrix. Eeach row $X_{m,*}$ is the stored data in rack m for $m = 1, 2, \ldots, 5$. Here, the codeword matrix $\mathbf{X}$ must satisfy (10) and (11) where the parity check matrices are respectively

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{K} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix},$$

and

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Suppose the node $X_{1,1}$ fails. Then by Definition 3,

$$\Omega(\mathbf{H}, 1) = \Big\{\{3,4,8\},\{2,7,8\},\{2,4,6\},\{3,6,7\},\{2,3,5\},\{4,5,7\},\{5,6,8\},\{2,3,4,5,6,7,8\}\Big\}.$$

According to our definition, each set in $\Omega(\mathbf{H}, 1)$ indicates a group of nodes which can be used to repair $X_{1,1}$ locally within the rack.

Now, assume that nodes $X_{1,1}, X_{1,2}, X_{1,4}, X_{1,6}$ in rack 1 fail simultaneously. In this case, $X_{1,1}$ cannot be repaired locally within the rack and must be repaired with the assistance from other racks. Let

$$\mathbf{r}_1 = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

It can be verified directly that $\mathbf{r}_1$ satisfies the criterion 1) in Theorem 2. Then, μ_1 (the support of $\mathbf{r}_1$) is the set $\{2,3,5,7,8\}$. Let $\beta_1 = \{7,8\}$ and $\tau = \{2,3,4\}$. Then they satisfy the criteria 3), 4), and 5). Note that, τ, μ_1, and β_1 indicate the group of racks, group of nodes in the survived racks, and the group of nodes in rack 1 which will participate in repairing the failed node $X_{1,1}$. By choosing the parameters $(\beta_1, \mu_1, \mathbf{r}_1, \tau)$ as above and the proper values for $\mathbf{y}$, $\mathbf{y}'$, a, $\mathbf{z}$, and $\mathbf{z}'$ we have

$$[c_{1,1}, \ldots, c_{1,8}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix},$$

and

$$\begin{bmatrix} d_{1,1,1} & \cdots & d_{1,1,8} \\ \vdots & \ddots & \vdots \\ d_{1,5,1} & \cdots, & d_{1,5,8} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Now, the failed node $X_{1,1}$ can be recovered by (22) such that

$$X_{1,1} = \sum_{m\in\{2,3,4\}} \left(\sum_{s\in\{2,3,5,7,8\}} d_{1,m,s} X_{m,s} \right) + \sum_{n\in\{7,8\}} c_{1,n} X_{1,n}.$$

According to Theorem 3, the total number of 18 symbol transmissions within racks and 3 symbol transmissions across racks are needed to repair the failed node $X_{1,1}$.

References

[1] W. J. Bolosky, J. R. Douceur, D. ELY, and M. Theimer, "Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs," in *Proc. of Sigmetrics*, June 2000.

[2] P. Druschel and A. Rowstron, "Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility," in *Proc. of ACM SOSP*, 2001.

[3] H. Weatherspoon and J. D. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Proc. Int. Workshop Peer-to-Peer Syst.*, 2002.

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

[5] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6925–6934, Nov. 2012.

[6] T. H. Chan, M. A. Tebbi, and C. W. Sung, "Linear programming bounds for storage codes," in *9th International Conference on Information, Communication, and Signal Processing (ICICS 2013)*, Dec. 2013.

[7] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," in *proc. IEEE Int. Symp. Information Theory*, Cambridge, MA, July 2012, pp. 2771–2775.

[8] L. Pamies-Juarez, H. D. Hollmann, and F. Oggier, "Locally repairable codes with multiple repair alternatives," in *proc. IEEE Int. Symp. Information Theory*, 2013, pp. 892–896.

[9] S. Akhlaghi, A. Kiani, and M. R. Ghanavati, "Cost-bandwidth tradeoff in distributed storage systems," *Computer Communications*, vol. 33, no. 17, pp. 2105–2115, Nov. 2010.

[10] B. Gaston, J. Pujol, and M. Villanueva, "A realistic distributed storage system that minimizes data storage and repair bandwidth," in *Data Compression Conference (DCC), 2013*, March 2013, pp. 491–491.