Multi-Rack Distributed Data Storage Networks

Ali Tebbi, Terence H. Chan, and Chi Wan Sung, *Senior Member, IEEE*

***Abstract*— The majority of works in distributed storage networks assume a simple network model with a collection of identical storage nodes with the same communication cost between the nodes. In this paper, we consider a realistic multi-rack distributed data storage network and present a code design framework for this model. Considering the cheaper data transmission within the racks, our code construction method is able to locally repair the nodes failure within the same rack by using only the survived nodes in the same rack. However, in the case of severe failure patterns when the information content of the survived nodes is not sufficient to repair the failures, other racks will participate in the repair process. By employing the criteria of our multi-rack storage code, we establish a linear programming bound on the size of the code in order to maximize the code rate.**

***Index Terms*— Multi-rack storage network, repair process, linear programming, symmetry.**

I. Introduction

MOST of the existing distributed storage network models assume a 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 [2]–[6]. However, this model cannot perfectly represent the real world storage networks. In reality, a typical data centre can easily house hundreds of racks each of which contains numerous storage disks [7]. While all storage nodes (or disks here) in the network 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 [8]. It is reported that in practical networks the inter-rack communication cost is typically 5 to 20 times higher than the intra-rack transmission cost [9].

A common approach of storing data in multi-rack storage networks is storing each encoded symbol of a data block in distinct nodes located in distinct racks [10], [11]. Consequently, repairing any node failure requires transferring data from survived nodes across the racks. Due to the large amount of data which is required to be communicated across racks during a failure repair process, this approach could be highly costly [12].

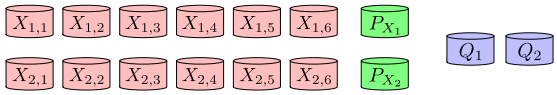

Fig. 1. A (16,12) pyramid code used in Windows Azur storage.

While *Regenerating Codes* [2], [13], [14] were introduced to reduce the repair bandwidth, *Locally Repairable Codes (LRC)* [6], [15]–[21] were introduced in order to optimize the node failure repair by involving a small group of helper nodes. It is worth to mention that any storage code, specially locally repairable codes which are designed for generic model of distributed storage networks can also be used for the multi-rack networks. However, these coding schemes are not able to provide optimal methods to repair the failures in the network since they do not take into consideration the different transmission cost between the nodes in the multi-rack storage networks. For instance, consider the locally repairable code known as *Pyramid Code* [21] which is used in Windows Azur storage [10]. This pyramid code, as illustrated in Figure 1, is a $(n = 16, k = 12, r = 6)$ locally repairable code with two *local repair groups* of size $r = 6$. The local repair groups are $X_{1,1} - X_{1,6}$ with its local parity node P_{X_1} and $X_{2,1} - X_{2,6}$ with its local parity node P_{X_2}. Parity nodes Q_1 and Q_2 are global parities. Assume each repair group (and one of the global parity nodes to keep the load balanced) is stored in a separate rack.

Any node failure can be repaired inside the rack by the $r = 6$ surviving nodes. However, if there exists multiple failures within a rack, the content of the helper nodes from the other rack needs to be transmitted across to the failed rack. More precisely, at least 6 symbols need to be transferred across the racks in order to repair a single failure. This will impose a high repair bandwidth to the storage network.

In this paper we introduce a realistic multi-rack storage network which represents the real data storage networks more generally and practically. We focus on a storage code-design framework, specifically tailored for multi-rack data storage networks and their requirements. Our storage network model depicted in Figure 2 consists of M racks each of which contains N storage nodes. We will assume that each rack has a *Processing Unit* (PU) which is directly connected to all storage nodes in the same rack.

Manuscript received September 26, 2017; revised March 4, 2019; accepted May 30, 2019. Date of publication July 9, 2019; date of current version September 13, 2019. This work was supported in part by a grant from the Australian Research Council through the Discovery Project under Grant DP150103658 and in part by the Research Grants Council of the Hong Kong Special Administrative Region, China, under Project CityU 11205318. This paper was partially presented at the 2014 Information Theory Workshop [1].

A. Tebbi and T. H. Chan are with the Institute for Telecommunications Research, University of South Australia, Adelaide, SA 5095, Australia (e-mail: ali.tebbi@unisa.edu.au; terence.chan@unisa.edu.au).

C. W. Sung is with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong (e-mail: albert.sung@cityu.edu.hk).

Communicated by K. Narayanan, Associate Editor for Coding Techniques.

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

Digital Object Identifier 10.1109/TIT.2019.2927565

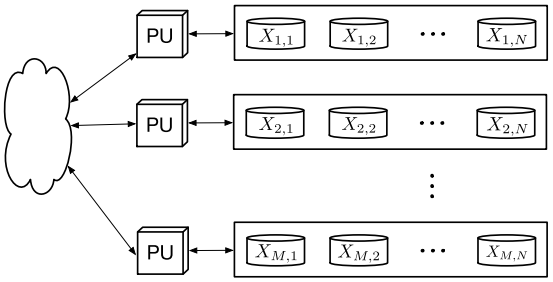

Fig. 2. The multi-rack distributed storage network. Each rack is equipped with a processing unit (PU) which is responsible for computations and intra-rack and inter-rack communication.

It is worth mentioning that in practical data centres all servers in a rack are connected to an in-rack switch which is called Top-of-Rack (ToR) switch. The ToR switch is responsible for the intra-rack communication while one or more servers in the rack could be used as compute nodes [22]. This architecture can be viewed as the Processing Unit of the rack. The rack processing unit is responsible for both computation on the stored data and communication between the nodes in the rack. Moreover, the processing units of racks can communicate to each other in order to transmit data from one rack to another (via *aggregation switches* [22]). In other words, storage nodes in two different racks can only communicate via their respective processing units. It is very common in realistic systems that the communication cost between the storage nodes within a rack (via its PU) is much lower than the communication cost between two different processing units (and hence located in two different racks) [8]. Therefore, it is desirable and in fact critical that a failed node could be repaired by only the survived nodes within the same rack in order to keep the repair cost low. We further assume that the system bottleneck is at the PU of each rack. Therefore, in our code-design framework, the focus is to design distributed storage codes that minimize the communication costs between nodes and the processing unit of the rack. It is only for some occasional severe failure patterns that it will require nodes from other racks to assist in the repair process.

Our multi-rack storage code is defined with three parity check matrices $\mathbf{H}$, $\mathbf{K}$, and $\mathbf{G}$. Matrix $\mathbf{H}$ determines the intra-rack repair groups such that any failed node in any of the racks can be repaired by at most r_1 surviving nodes in the same rack. We derive the conditions on $\mathbf{H}$ such that the intra-rack local repair can still be successful even in the presence of multiple failures in the rack. However, as mentioned earlier, in the case of severe failure patterns where the intra-rack repair fails, matrices $\mathbf{H}$ and $\mathbf{K}$ determine a group of helper nodes from the same rack of the failure and the other racks in order to proceed with the inter-rack repair. Moreover, in our coding scheme, parity matrix $\mathbf{G}$ determines the group of helper racks during an inter-rack repair. We show that $\mathbf{G}$ can be designed separately as the parity check matrix of a locally repairable code such that only a small group of racks participate in the inter-rack repair process.

In general, existing locally repairable coding schemes are not suitable candidates for practical multi-rack distributed storage networks. Assume that a repair group of an LRC is considered to be a rack. In the case that a local repair fails, the network will need other repair groups (racks) to help for the failure repair. This assumption is costly due to the geographically distributed nature of the storage networks. However, in our coding scheme, it is only in occasional sever failure patterns that the other racks are needed to help repairing the failure. Moreover, the existing LRC schemes do not take into account the different communication cost between the nodes. A major advantage of our coding scheme compared to similar schemes, such as LRCs, is the concept of the rack processing unit. In contrast to LRCs, our coding scheme enables the helper racks to only transmit a linear combination of the helper nodes' content to the failed rack via their processing unit. This approach optimizes the inter-rack repair bandwidth and significantly reduces the inter-rack communication cost.

A. Related Work

A regenerating code [2] is proposed in [23] in order to minimize the across rack repair bandwidth. In this coding scheme, each rack stores multiple encoded symbols (rather than one symbol [11]) of a data block in distinct storage nodes. To repair a failed node, first a regeneration will be occurred within each rack (i.e., one of nodes in each rack collects the encoded data from all nodes in the rack and re-encodes it) and then the regenerated data from each rack will be transferred to the failed rack to regenerate the content of the failed node. Recently, heterogeneous distributed storage networks (including the multi-rack models) has received a fair amount of attention [24]–[28] due to the heterogeneous nature of the practical storage networks and their various applications such as hybrid storage systems [29], video-on-demand systems [30], and heterogeneous wireless networks [31]. A heterogeneous model for distributed storage networks is introduced in [28] where a static classification of the storage nodes is proposed. In this model the storage nodes are partitioned into two groups with "cheap" and "expensive" bandwidth. In other word, the data download cost, to repair a failure, from the nodes in the "cheap bandwidth" group will be lower than the data download cost from the "expensive bandwidth" group. The model in [28] partially addressed the issues that the communication costs among nodes are not all equal where the download cost from one of the groups is always cheaper than the other group. However, this model does not fit well into a multi-rack model, where the transmission cost should depend on both 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 [25], in which the authors considered a two-rack model. In their model, the communication cost between the nodes in the same rack is smaller than between two different racks. Therefore, the main difference of this model compared to the one in [28] is that the classification of the storage nodes depends on the location of the failed node. More precisely, the data download cost from the nodes in the same rack (i.e., group) of the failure is lower (i.e., cheap bandwidth) than the download cost from the other group (i.e., expensive bandwidth). As such, it is desirable that more

data should be transmitted by nodes in the same rack of the failed node during the repair process. Using an information flow graph, [25] derives the trade-off between the storage cost and repair cost by identifying that if certain choice of parameters are achievable or not. The trade-off in [28] and [25] is asymptotic (without restriction on the alphabet size) and functional repair is always assumed (i.e., the failed node is not required to recover exactly what it was previously storing, as long as the whole storage system is still robust after repair).

A non-homogenous storage system is considered in [27] where there exists a super node in the network with higher storage capacity, reliability and availability probability than the other nodes. It has been shown that this model can achieve the optimal bandwidth-storage trade-off bound in [2] with a smaller file and alphabet size than the traditional homogeneous storage network in [32].

The Data retrieval problem in heterogeneous storage systems is studied in [33]. In this model it is assumed that each node has a different storage size where any amount of encoded data can be stored in each node such that the total allocated storage remains less than a threshold. The optimal allocation to retrieve the original data is studied such that the data collector can access to only a random group of nodes. A combination of the repair problem with data allocation is investigated in [34] and [35]. In these works a general model of a heterogeneous storage network is considered where each node has a different storage and download cost. The amount of data allocated to each storage node and the amount of data to be downloaded from each survived node to repair a failure has been investigated using the information flow graph to minimize the storage and repair cost and establishing a storage-repair trade-off.

The capacity of heterogenous storage networks is studied in [36]. The proposed network in this work consists of storage nodes with different storage capacities and repair bandwidths. It is assumed that the repair bandwidth of each node depends on the repair group that the helper nodes belong to. The functional repair of node failures is assumed and the capacity of this network as the maximum amount of stored information in order to reach a level of reliability is studied.

Block Failure Resilient (BFR) codes are studied in [37]. The authors consider a distributed storage network with a single failure domain [38] where the storage nodes are divided in blocks (e.g. racks). The failure of a block will result in unavailability of the nodes in that block. Consider a storage network with n nodes and b blocks where each block contains $\frac{n}{b}$ nodes. BFR codes relax the node-repairability and data-reconstruction constraints of the regenerating codes such that any failed node within a block can be repaired by contacting any d_r nodes of any $b_r = b-\sigma$ available blocks (i.e., $d = d_r b_r$ nodes in total). Moreover, the original data can be retrieved by contacting any k_c nodes of any $b_c = b - \rho$ available blocks (i.e., $k = k_c b_c$ nodes in total) where ρ is the resilience parameter. For such a relaxation, similar to the regenerating codes, the storage per node and repair bandwidth trade-off is derived. Locally repairable BFR codes are also introduced in [37] such that a failed node can be repaired by contacting the nodes of a local group of blocks (e.g., cluster). One of the main differences of the network model in [37] with our model is that it is assumed that always during the repair process of a failed node, the other nodes of the same rack are also unavailable (i.e., single failure domain) and they are not able to contribute in the repair process. In our model, we assume that in non-severe failure patterns, the available nodes of the rack can locally repair the failed node.

A similar network model to our work is considered in [39] where the network consists of n clusters (e.g., racks) each of them stores m nodes. The network is fully connected such that the nodes within a cluster are connected via an intra-cluster link and the clusters are connected via an inter-cluster link. The proposed coding scheme is a generalization of the regenerating codes [2] where a file of size B symbols is encoded into $nm\alpha$ symbols and stored across nm nodes in the network such that each node stores α symbols. In order to repair a failed node, β symbols will be downloaded each from any subset of d clusters. These β symbols are a function of the content (at most γ', $\gamma' \leq \alpha$ symbols) of at most ℓ' nodes in each helper cluster. Moreover, the content (at most γ, $\gamma \leq \alpha$ symbols) of ℓ local helper nodes will be downloaded to contribute in the repair process. Utilizing the information flow graph under the functional repair settings, an upper bound on the file size B is derived. For fixed values of B, the bound gives the trade-off between storage and inter-cluster bandwidth. A lower bound on intra-cluster bandwidth γ is also obtained. Unlike our coding scheme where the inter-rack repair happens only in the case of severe failure patterns, a failed node in [39] always is repaired by the help of a group of d clusters (racks). Note that, all the bounds obtained in the aforementioned papers [37], [39] are based on the information flow graph under functional repair settings.

The capacity of clustered storage systems is investigated in [40]. The proposed network model consists of n storage node distributed over L clusters each of which contains $n_I = \frac{n}{L}$ nodes each with storage size α. A failed node is regenerated by downloading β_I symbols each from d_I nodes within the same rack and β_c symbols each from d_c nodes from each cluster. It is assumed that during the repair process, all other nodes are available and will be contacted (i.e., $d_I = n_I - 1$ and $d_c = n - n_I$). Also $\beta_I \geq \beta_c$, due to the lower inter-cluster communication bandwidth compared to the intra-cluster. Employing the information flow graph, the storage capacity of this network is obtained in terms of the node storage size α, intra-cluster repair bandwidth $\gamma_I = d_I\beta_I$, and inter-cluster repair bandwidth $\gamma_c = d_c\beta_c$. Note that since the coding scheme is based on regenerating codes, in order to minimize the repair bandwidth all $n-1$ nodes need to help to repair the failure.

The availability of clustered storage networks is studied in [41]. The aim in this work is to partition n storage nodes of the network into s clusters of size d (there could be an extra cluster of size $< d$) such that any failed node in a cluster can be repaired by any of the remaining clusters (except the last cluster with less storage nodes) as its repair group. Then, the network is said to have availability $s-1$. The objective is achieving high availability and low

repair bandwidth. The storage per node vs repair bandwidth trade-off is characterized following the network information flow graph under the functional and exact repair settings. Some class of codes are also proposed to minimize the exact repair bandwidth.

The notion of codes with *hierarchical locality* has been studied in [42]. Codes with hierarchical locality are an extension on the codes with (r, δ)-locality which are introduced in [5] such that any code symbol can be recovered locally by at most r other symbols even in the presence of an additional $(\delta - 2)$ erasures. A h-level hierarchical code is an $[n, k, d]$ linear code $\mathcal{C}$ with locality parameters $[(r_1, \delta_1), (r_2, \delta_2), \ldots, (r_h, \delta_h)]$ where depending on the number of the failures (i.e., $\delta_i - 1$), there exists a punctured code C_i with locality parameters (r_i, δ_i) that can repair the failures.

The fact that coding at large lengths allows better error-tolerance for a given overhead, motivated the work in [43]. One of the main challenges in the storage networks with large length codes is correlated failures which could happen due to e.g. a rack failure, a data centre failure, or failure of a power source shared by a group of servers (i.e., single failure domain [38]). This work views the code design for a distributed storage network as a two step process of 1) picking a topology and 2) optimizing encoding/decoding efficiency and maximizing reliability. The authors consider a simple grid-like topology (which is also extendable to the multi-rack storage networks) where each row and column of coded symbols has a bunch of parity equations and there are some global parity equations that depend on all symbols (i.e., tensor products of row and column codes, augmented with global parity equations). A lower bound on the field size of the Maximally Recoverable codes is obtained and the correctable erasure patterns by these codes are characterized. An asymptotically optimal family of Maximally Recoverable codes for one basic topology is also proposed.

Despite of the applications of the works in [42] and [43] in multi-rack storage networks, neither of them propose a general code design framework which is specifically tailored for multi-rack distributed storage network considering various network parameters such as different intra-rack and inter-rack communication cost. For example, assume that a storage code with hierarchical locality is employed in a multi-rack storage network. Depending on the failure pattern, it would need all the racks to be available to repair a failure which is not a practical assumption due to geographically distributed nature of the network. Moreover, in [42] and generally other LRC schemes in the literature, it is assumed that during a repair process, the content of each helper node will be transmitted separately to the failed node's replacement (newcomer) in order to recover the lost data. In a multi-rack storage network, this will impose a high inter-rack repair bandwidth to the network due to the high inter-rack communication cost.

B. Contributions and Organization

The main contributions of this paper are:

- A code-design framework for multi-rack storage networks: we propose a general coding scheme for multi-rack distributed storage networks. Our proposed scheme is defined by three parity check matrices **H**, **K**, and **G**. This coding scheme is able to locally repair any node failure within the rack by using matrix **H** in order to minimize the repair cost. Moreover, in the case of severe failure patterns that the failures cannot be repaired only by the survived nodes inside the rack, by using matrix **K**, our scheme is able to engage some of the nodes in other racks in the repair process. The helper racks will be determined by matrix **G**.
- Establishing linear programming bounds on the code size: we show that maximizing the rate of the multi-rack storage code is equivalent to maximizing the code size. We establish a linear programming problem on the code size based on the definition and criteria of our multi-rack storage code. The maximum size of the code in turn will determine the optimal size of the parity check matrices **H** and **K**.

This paper is extended from our earlier work on multi-rack distributed storage codes [1] which is presented in IEEE Information Theory workshop (ITW 2014). The rest of this paper is organized as follows. In Section II we present the code-design framework for multi-rack storage networks and give a detailed description of its criteria and the failures repair processes. We also derive the code rate in this section. Then, In Section III, we establish a linear programming problem to upper bound the code size. Moreover, in this section, we exploit symmetry in our code in order to reduce the complexity of the problem. The paper is concluded in Section IV.

II. Multi-Rack Storage Code – Design Framework

In this section, we first introduce our system model and multi-rack storage code which is defined by three parity check matrices **H**, **K**, and **G**. We then describe the intra-rack repair process and show how the failures can be repaired only by the surviving nodes inside the rack using the parity check matrix **H**. The inter-rack repair process will be described afterwards where we show how a failure can be repaired by the surviving nodes inside the rack and the nodes in helper racks when the intra-rack repair fails. Finally, we present the rate of the multi-rack storage code which will be used in the next section to establish an upper bound on the code size.

Consider the rack model storage network depicted in Figure 2. This multi-rack data storage network consists of M racks each of which contains N storage nodes (or storage disks). We will represent each node as

$$\left(X_{m,n},\ \forall m \in \mathcal{M} \text{ and } \forall n \in \mathcal{N}\right)$$

where $\mathcal{M} = \{1, \ldots, M\}$, $\mathcal{N} = \{1, \ldots, N\}$, and $X_{m,n}$ is referred to the nth node in the mth rack. Abusing notations, $X_{m,n}$ will also be referred to the content stored at that particular storage node. We define

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

whose entries are from $\mathbb{F}_q$. Particularly, $X_{m,*}$ is the vector of encoded data stored in the rack m. Collecting all the stored

contents from each rack, we have

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

In this paper, we assume that each rack has a processing unit, which is responsible for all computations required in nodes repair. In other words, contents stored in a failed node will be regenerated in the processing unit, before sending all the regenerated content to the failed node (or its replica).

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

$$\mathbf{H}X^\top = \mathbf{0} \tag{2}$$

$$\mathbf{K}X^\top\mathbf{G}^\top = \mathbf{0}. \tag{3}$$

We will call $\mathbf{H}, \mathbf{K}$ *respectively the intra-rack and inter-rack parity matrices. The matrix* $\mathbf{G}$ *will be called helper-rack parity check matrix.*

Later in Section III we will show that maximizing the code rate is equivalent to maximizing the size of the code which in turn can determine the optimal value of S_1 and S_2. Moreover, we will show that the value of L is only dependent on the network topology and can be chosen separately from S_1 and S_2.

In multi-rack storage code, it is expected that most of the node failures should be recovered and repaired locally within their own racks. However, in the special case where local repair is not possible, redundancies added among rack will be used in the recovery. As there is a much lower probability that nodes in a rack cannot be recovered locally within the rack, this paper focuses on the special case where only one rack has node failures (or that all failed nodes in other racks can be completely repaired locally).

We now consider the first case where failures in a rack can be repaired by using only nodes within the rack.

A. Intra-Rack Repair

In this subsection, we will describe how to repair nodes locally within a rack. Assume without loss of generality that rack 1 fails (i.e., a group of nodes fails inside the rack). Let γ be the index set for the nodes in rack 1 that fail. In other words, the values of $\{X_{1,n}, n \in \gamma\}$ (i.e., the node content) are unknown to the processing unit in rack 1. Let

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}$$

where

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

In other words, $\mathbf{x}$ 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. Let $\bar{\gamma}$ be the complement set of γ. Therefore, $\bar{\gamma}$ will be the set of survived nodes in the rack 1. Consequently, $\mathbf{I}^{\bar{\gamma}} X_{1,*}^\top = \mathbf{x}$. Recall that $\mathbf{H}X_{1,*}^\top = \mathbf{0}$. Therefore, rack 1 can repair ALL its failed nodes by the local rack survived nodes if and only if the following system of linear equations

$$\begin{cases} \mathbf{I}^{\bar{\gamma}} X_{1,*}^\top = \mathbf{x} \\ \mathbf{H}X_{1,*}^\top = \mathbf{0} \end{cases} \tag{4}$$

has a unique solution. For notation simplicity, we will use $\langle \mathbf{I}^\beta, \mathbf{H} \rangle$ to denote the vector space spanned by rows of $\mathbf{I}^\beta$ and $\mathbf{H}$. The set of linear equations in (4) has a unique solution if and only if

$$\dim\langle \mathbf{I}^{\bar{\gamma}}, \mathbf{H} \rangle = N. \tag{5}$$

Let γ_o be the smallest set such that $\dim\langle \mathbf{I}^{\bar{\gamma_o}}, \mathbf{H} \rangle < N$. We will denote its size $|\gamma_o|$ as $\text{Dist}(\mathbf{H})$. By definition, if $|\gamma| < \text{Dist}(\mathbf{H})$, then it is sufficient to use intra-rack repair to repair all failed nodes.

Remark 1. *It is well known that* $\text{Dist}(\mathbf{H})$ *is equal to the minimum distance of a linear code defined by the parity check matrix* $\mathbf{H}$.

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*

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

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

Remark 2. *As we shall see,* $\Omega(\mathbf{H}, \mathbf{r}, j)$ *plays a fundamental role in determining whether failures in a rack can be repaired or not. Specifically,* $\Omega(\mathbf{H}, j)$ *contains all intra-rack repair groups for* $X_{1,j}$. *If there exists a set (or group)* $\beta \in \Omega(\mathbf{H}, j)$ *such that all* $X_{1,\ell}$ *are survived for all* $\ell \in \beta$, *then the failed node* $X_{1,j}$ *can be repaired by using only* $X_{1,\ell}$ *for all* $\ell \in \beta$. *The general case where* $\mathbf{r}$ *is non-zero vector will be used in the inter-rack repair and will be explained soon.*

Example 1. *Suppose* $\mathbf{H}$ *is the intra-rack parity check matrix and is given by*

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6}$$

From Definition 3, $\Omega(\mathbf{H}, 1)$ *will be given by*

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

where the entries are the index set of a group of nodes in each rack. Each subset in $\Omega(\mathbf{H}, 1)$ *denotes a intra-rack repair group for repairing* $X_{1,1}$ *(or* $X_{m,1}$ *in general).*

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

$$\mathbf{e}_j = \mathbf{y}\mathbf{H} + a\mathbf{r} + \mathbf{y}'\mathbf{I}^{\beta} \tag{7}$$

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

$$e_{j,\ell} = \begin{cases} 1 & \text{if } \ell = j \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Conversely, if there exist vectors $\mathbf{y}$*,* $\mathbf{y}'$ *and* $a \in \mathbb{F}_q$ *such that* (7) *holds, then there exists* $\alpha \subseteq \beta$ *such that* $\alpha \in \Omega(\mathbf{H}, \mathbf{r}, j)$*.*

Proof. Since $\beta \in \Omega(\mathbf{H}, \mathbf{r}, j)$, then there exists $\mathbf{u} = [u_1, \ldots, u_N]$ such that 1) $\mathbf{u} = \mathbf{y}\mathbf{H} + a\mathbf{r}$ for some vector $\mathbf{y}$ and $a \in \mathbb{F}_q$, 2) $u_j = 1$ and 3) $\lambda(\mathbf{u}) \setminus \{j\} = \beta$. Let $\mathbf{v} = -\mathbf{u}\mathbf{I}^{\beta}$. Since $\lambda(\mathbf{u}) \setminus \{j\} = \beta$, $\mathbf{v} = -\mathbf{u} + \mathbf{e}_j$. Hence,

$$\begin{aligned} \mathbf{e}_j &= \mathbf{v} + \mathbf{u} \\ &= \mathbf{y}\mathbf{H} + a\mathbf{r} + \mathbf{v} \\ &= \mathbf{y}\mathbf{H} + a\mathbf{r} - \mathbf{u}\mathbf{I}^{\beta}. \end{aligned}$$

The lemma thus follows by letting $\mathbf{y}' = -\mathbf{u}$. The proof of the converse is straightforward and is omitted. □

Based on Lemma 1, the following theorem specifies conditions for intra-rack repair.

Theorem 1 (Intra-rack Repair)**.** *Suppose node* j *fails in rack* $m = 1$*. Let* γ_j *be the index set for all failed nodes*[1] *(hence,* $j \in \gamma_j$*). If* $\beta_j \subseteq \{1, \ldots, N\}$ *satisfies the following two criteria,*

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

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

$$X_{1,j} = \sum_{n \in \beta_j} c_{j,n} X_{1,n}. \tag{9}$$

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

$$\mathbf{e}_j = \mathbf{y}\mathbf{H} + \mathbf{y}'\mathbf{I}^{\beta_j}. \tag{10}$$

Hence,

$$\begin{aligned} \mathbf{e}_j X_{1,*}^{\top} &= (\mathbf{y}\mathbf{H} + \mathbf{y}'\mathbf{I}^{\beta_j}) X_{1,*}^{\top} && (11) \\ &= \mathbf{y}\mathbf{H} X_{1,*}^{\top} + \mathbf{y}'\mathbf{I}^{\beta_j} X_{1,*}^{\top} && (12) \\ &= \mathbf{y}'\mathbf{I}^{\beta_j} X_{1,*}^{\top}, && (13) \end{aligned}$$

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

$$[c_{j,1}, \ldots, c_{j,N}] = \mathbf{y}'\mathbf{I}^{\beta_j}. \tag{14}$$

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 the theorem. □

Equation (9) essentially defines how to regenerate the content of a failed node $X_{1,j}$ from $X_{1,n}$ for $n \in \beta_j$ (i.e., the nodes in its repair group). In other words, node $X_{1,j}$ is a linear combination of the nodes in its repair group where the coefficients are $c_{j,n}$. 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 (9). 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.

[1] γ_j can be interpreted as the set of failed nodes at the moment when the node j is being repaired.

Example 2. *Let* $\mathbf{H}$ *be the intra-rack parity check matrix over* $\mathbb{F}_3$ *such that*

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 \end{bmatrix}.$$

Assume nodes $X_{1,1}$ *and* $X_{1,2}$ *are failed. Thus, the failure pattern will be* $\gamma = \{1, 2\}$*. Suppose we want to repair node* $X_{1,1}$*. The repair groups of node* $j = 1$ *is given by*

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

A repair group $\beta_1 \in \Omega(\mathbf{H}, 1)$ *is eligible for intra-rack repair process such that* $\beta_1 \cap \gamma = \emptyset$*. Therefore,* $\beta_1 = \{3, 4\}$*. Moreover, we choose* $\mathbf{y} = [0\ 1]$*. Then,* $\mathbf{u} = \mathbf{y}\mathbf{H} = [1\ 0\ 1\ 2]$ *and* $\mathbf{y}' = -\mathbf{u} = [-1\ 0\ -1\ -2]$ *satisfying the conditions in Lemma 1. Therefore, the repair coefficients vector in* (14) *is given by*

$$\mathbf{y}'\mathbf{I}^{\beta_1} = [0\ \ 0\ \ -1\ \ -2].$$

Consequently,

$$X_{1,1} = -X_{1,3} - 2X_{1,4}.$$

The remaining failure $X_{1,2}$ *can also be repaired by the same procedure.*

B. Inter-Rack Repair

Communications across racks in a multi-rack storage network are in general more expensive. Consider the extreme case where each rack physically represents a data center, each of which is geographically distant from each other. In this case, data transmission across long distance is clearly more expensive than transmission within each rack. Therefore, it is often desirable to design codes such that more repairs can be done locally within racks. However, in some rare cases (e.g., burst failure within a rack), nodes failure cannot be repaired locally. For example, this may occur when node $X_{m,j}$ fails and for all $\beta \in \Omega(\mathbf{H}, j)$, there is at least another node $X_{m,k}$ for $k \in \beta$ which also fails. When intra-rack repair fails, inter-rack repair can be done. The idea is described below.

Let $\mathbf{h} = (h_1, \ldots, h_N)$, $\mathbf{k} = (k_1, \ldots, k_N)$ and $\mathbf{g} = (g_1, \ldots, g_M)$ be respectively vectors spanned by the rows of the matrices $\mathbf{H}$, $\mathbf{K}$ and $\mathbf{G}$. Then, it can be verified directly from (2) and (3) that

$$\mathbf{h} X_{m,*}^{\top} = 0 \tag{15}$$

$$\mathbf{k} X_{m,*}^{\top} = -g_m^{-1} \sum_{i \in \tau \setminus \{m\}} \mathbf{k}\, g_i X_{i,*}^{\top} \tag{16}$$

where $\tau = \{i \in \mathcal{M} : g_i \neq 0\}$ and is assumed to contain m. Suppose $\beta = \lambda(\mathbf{h} + \mathbf{k})$ and $j \in \beta$. Then

$$(h_j + k_j)X_{m,j} = -\sum_{i \in \beta \setminus \{j\}} (h_i + k_i)X_{m,i} - g_m^{-1} \sum_{i \in \tau \setminus \{m\}} \mathbf{k} g_i X_{i,*}^{\top} \tag{17}$$

can be used to recover $X_{m,j}$. Equation (17) consequently defines the across rack repairs. To be more precise, in order to repair the failed node $X_{m,j}$, one would need 1) code symbols $X_{m,i}$ from the failing rack m for $i \in \beta \setminus \{j\}$, and 2) code symbols $X_{i,\ell}$ from rack i for $i \in \tau \setminus \{m\}$ and $\ell \in \{i \in \mathcal{N} : k_i \neq 0\}$. In other words, to repair a failed node $X_{m,j}$ a group of helper racks τ are identified by parity matrix $\mathbf{G}$. Also, a group of helper nodes in each helper rack is identified by parity matrix $\mathbf{K}$. The helper nodes in each helper rack will send their content to the rack process unit. Each helper rack process unit calculates a linear combination of the helper nodes content and send it to the process unit of the failed rack m. The process unit of the failed rack m calculates the sum of this information received from helper racks. A group of survived nodes from the failed rack which are specified by $\mathbf{H}$ and $\mathbf{K}$ send their content to the rack process unit. The process unit then calculates a linear combination of the information from these nodes and adds it to the information from the helper racks. This results in the information content of the failed node $X_{m,j}$.

Theorem 2. *Suppose node j fails in rack $m = 1$. Let γ_j be the index set for all failed nodes (hence, $j \in \gamma_j$). If $(\beta_j, \mu_j, \mathbf{r}_j, \tau)$ satisfies the following criteria,*

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

then there exists $c_{j,n}$ for $n \in \beta_j$ and $d_{j,m,s}$ for $m \in \tau, s \in \mu_j$ such that

$$X_{1,j} = \sum_{m \in \tau} \left(\sum_{s \in \mu_j} d_{j,m,s} X_{m,s} \right) + \sum_{n \in \beta_j} c_{j,n} X_{1,n}. \tag{18}$$

The proof of Theorem 2 is given in Appendix A.

Remark 3. *The interpretation of the theorem is as follows: The support of $\mathbf{r}_j \in \langle \mathbf{K} \rangle$ corresponds to index of nodes in the "helper racks". Clearly, the smaller is the support the better, in order to minimize transmission cost. However, we would also point out that the transmission costs required to transmit across racks does not depend on the support size of $\mathbf{r}_j$. More precisely, for each helper rack, only the sum $\sum_{s \in \mu_j} d_{j,m,s} X_{m,s}$ is required to be transmitted, instead of specific individual $X_{m,s}$. On the other hand, the set β_j denotes the set of nodes which can be used to repair $X_{1,j}$. Consequently, β_j and γ_j (index set for the failed nodes in rack 1) must be disjoint. Finally, τ is the index set of the helper racks. Note also that Theorem 2 reduces to Theorem 1 if $\tau = \mu_j = \emptyset$ and $\mathbf{r}_j$ is the zero vector.*

Remark 4. *As a consequence of Theorem 2, The processing unit in rack m where $m \in \tau$, will retrieve $|\mu_j|$ symbols. The processing unit of rack 1, will need to retrieve $|\beta_j|$ symbols within the rack. Also, one symbol transmission is needed for the processing unit of rack 1 to send the recovered symbol back to the failed storage node $X_{1,j}$. Finally, each helper rack indexed in τ will transmit 1 symbol to the processing unit of rack 1. Summing up all these transmissions, there are in total 1) $|\beta_j| + |\mu_j||\tau| + 1$ symbol transmission within racks, and 2) $|\tau|$ symbol transmissions across racks.*

Example 3. *Consider a rack model storage network with $M = 5$ racks, each of which contains $N = 8$ storage nodes. Suppose the parity check matrices are as follows:*

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

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

and

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

Note that the code is over $GF(2)$. Suppose node $X_{1,1}$ fails. Then by Definition 3, it can be verified that

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

In particular, any subset of nodes in rack 1 indexed by $\Omega(\mathbf{H}, 1)$ can be used to repair $X_{1,1}$.

Now, suppose that the following nodes $\{X_{1,1}, X_{1,2}, X_{1,4}, X_{1,6}\}$ failed in rack 1 (i.e., $\gamma = \{1, 2, 4, 6\}$). In this case, $X_{1,1}$ cannot be repaired via intra-rack repair since there exist no intra-rack repair group $\beta_1 \in \Omega(\mathbf{H}, 1)$ such that $\beta_1 \cap \gamma = \emptyset$. Therefore, inter-rack repair is needed. Let $\mathbf{r}_1 = [0\ 1\ 1\ 0\ 1\ 0\ 1\ 1]$ and hence $\mu_1 = \{2, 3, 5, 7, 8\}$ following from criterion 1 and 2, respectively. Moreover, let $\beta_1 = \{7, 8\}$ and $\tau = \{2, 3, 4\}$ following from criteria 3–5, respectively. Note that, τ, μ_1, and β_1 indicate the group of helper racks, the group of helper nodes in the helper racks, and a repair group in rack 1 which will participate in repairing the failed node $X_{1,1}$.

Choose $\mathbf{y} = [1\ 0\ 0\ 0]$, $\mathbf{z} = [1\ 0\ 0]$, $\mathbf{z}' = [0\ 1\ 1\ 1\ 0]$, $\mathbf{y}' = [0\ 0\ 0\ 0\ 0\ 0\ 1\ 1]$, and $a = 1$ such that

$$\mathbf{e}_1 = \mathbf{y}\mathbf{H} + \mathbf{y}'\mathbf{I}^{\beta_1} + a\mathbf{r}_1, \tag{19}$$

$$\mathbf{f}_1 = \mathbf{z}\mathbf{G} + \mathbf{z}'\mathbf{I}^{\tau}, \tag{20}$$

are satisfied where (19) and (20) follow from Lemma 1, and $\mathbf{f}_1$ is defined in (60). Following from (58) and (69), we have

$$[c_{1,1}, \ldots, c_{1,8}] = [\ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1\ ],$$

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* (18) *such that*

$$X_{1,1} = \sum_{m\in\{2,3,4\}} \left(\sum_{s\in\{2,3,5,7,8\}} d_{1,m,s} X_{m,s} \right) + \sum_{n\in\{7,8\}} c_{1,n} X_{1,n}. \tag{21}$$

In this example a total number of 18 symbol transmissions within the racks and 3 *symbol transmissions across the racks are needed to repair the failed node* $X_{1,1}$.

When we need to repair multiple failed nodes (say $|\gamma_j|$ of them) via inter-rack repair, the cost is not simply $|\gamma_j|$ times: First, it is possible that a transmission from inter-rack can be used to repair for more than one node. Second, once inter-rack repair has been achieved, nodes which are previously not repairable via intra-rack repair may become repairable. To be more precise, suppose nodes $(X_{1,j}, j \in \gamma)$ fail where $|\gamma| \geq \text{Dist}(\mathbf{H})$. In this case, nodes failure may not be able to be recovered merely via intra-rack repairs. Let $\alpha \subseteq \gamma$ be of size $\text{Dist}(\mathbf{H}) - 1$. In that case, in the worst case scenario, one can at least aim to recover variables $X_{1,j}$ for $j \in \gamma \setminus \alpha$ via inter-rack repair first. Once this is achieved, the remain nodes $(X_{1,j}, j \in \alpha)$ can be recovered via intra-rack repair. Following the idea, the following theorem gives an upper bound on the repair transmission cost.

Theorem 3 (Upper bound on transmission costs)**.** *Let* γ *be the index set of all failed nodes in rack* 1. *Suppose that 1) for any* $j \in \gamma \setminus \alpha$, *there exists* $(\beta_j, \mu_j, \mathbf{r}_j, \tau)$ *satisfying the criteria in Theorem 6 where* $\gamma_j \triangleq \gamma$ *and 2) for any* $j \in \alpha$, *there exists* β_j *satisfying the criteria in Theorem 1 where* $\gamma_j \triangleq \alpha$. *Then the required total transmissions within a rack* θ_{intra} *and across racks* θ_{inter} *are respectively upper bounded by*

$$\theta_{intra} \leq |\tau| \left| \bigcup_{j\in\gamma\setminus\alpha} \mu_j \right| + \left| \bigcup_{j\in\gamma} \beta_j \right| + |\gamma| \tag{22}$$

$$\theta_{inter} \leq |\tau| \dim\langle \mathbf{r}_j, j \in \gamma \setminus \alpha \rangle. \tag{23}$$

The proof of Theorem 3 is given in Appendix B.

C. Code Rate

In this subsection we derive the rate of our proposed code for multi-rack storage networks. The code rate will later be employed to establish the upper bound of the code size in Section III. The following theorem gives the rate of the multi-rack storage 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 multi-rack storage code* $(\mathbf{H}, \mathbf{K}, \mathbf{G})$ *is lower bounded by*

$$R \geq \frac{MN - MS_1 - LS_2}{MN}. \tag{24}$$

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

The proof of Theorem 4 is given in Appendix C.

III. Bounds

In this section, we first derive the relations between the code rate, code size, and the size of the parity check matrices. We show that under some constraints, maximizing the code rate is equivalent to maximizing the size of the code which in turn can determine the optimum size of the parity check matrices. We define the multi-rack storage network parameters such as intra- and inter-rack resilience and locality in order to establish a *Linear Programming (LP)* problem to maximize the size of the multi-rack storage code. Then, the symmetries in the problem will be used to significantly reduce the complexity of the LP problem.

A. Linear Programming Bound

In the previous section, we introduced a class of storage codes called multi-rack storage codes and explained how to repair nodes failure via intra-rack or inter-rack repairs. In this section, we will develop bounds for this class of codes. Recall our code construction and definition. We will notice the following:

1) The intra-rack parity check matrix $\mathbf{H}$ or more precisely the support of the dual codewords spanned by the rows of $\mathbf{H}$ determines how failed nodes can be repaired locally within a rack. Alternatively, the dual codewords spanned by intra-rack parity check matrix $\mathbf{H}$ and inter-rack parity check matrix $\mathbf{K}$ together defines the inter-rack repair process.
2) The helper-rack parity check matrix $\mathbf{G}$ specifies which racks can be used in the inter-rack repair process. Naturally, one would prefer to involve only a small number of racks to minimize the inter-rack transmission cost.

Assuming without loss of generality that all rows of $\mathbf{H}$ and $\mathbf{K}$ are independent and that $\mathbf{G}$ is full rank, Theorem 4 shows that

$$R(\Lambda_{\mathcal{C}}) = \frac{MN - MS_1 - LS_2}{MN}, \tag{25}$$

or equivalently,

$$R(\Lambda_{\mathcal{C}}) = \frac{N - S_1 - S_2}{N} + \frac{S_2(M - L)}{MN}. \tag{26}$$

The rate of the multi-rack storage code is essentially determined by the size of the matrices $\mathbf{H}, \mathbf{K}$ and $\mathbf{G}$.

Understanding their roles, we can immediately recognize that one can separately design $\mathbf{G}$ and $(\mathbf{H}, \mathbf{K})$. The design of $\mathbf{G}$ will affect the number of helper racks needed in inter-rack repair. In fact, it is very similar to the design of locally repairable codes. The idea is to design a linear code (specified

by the parity check matrix $\mathbf{G}$) such that for any $m \in \mathcal{M}$, there are dual codewords $\mathbf{g}$ with a small support containing m. If the racks are geographically separated and connected to a network, the design of $\mathbf{G}$ may take into account the network topology and the costs of the transmission links. For example, an algorithm for the design of linear binary locally repairable codes over a network can be found in [16]. There are also previous works including our own work in [19] and [18] which discuss the design and bounds for locally repairable codes. On the other hand, the design of $\mathbf{H}$ and $\mathbf{K}$ will affect the code's ability in intra-rack and inter-rack repair. The focus of the remaining paper is on understanding the fundamental limits of the best design of these two matrices.

Separating the design of $(\mathbf{H}, \mathbf{K})$ from $\mathbf{G}$, we can simply consider a simple special case where there are only two racks (i.e., $M = 2$) and that

$$\mathbf{G} = [1, -1].$$

Assume without loss of generality, we may characterize our multi-rack storage code via the following parity-check equations:

$$\mathbf{H}X_{1,*}^{\top} = \mathbf{0} \tag{27}$$

$$\mathbf{H}X_{2,*}^{\top} = \mathbf{0} \tag{28}$$

$$\mathbf{K}X_{1,*}^{\top} = \mathbf{K}X_{2,*}^{\top}. \tag{29}$$

As such, we will simply refer a multi-rack storage code as $(\mathbf{H}, \mathbf{K})$.

Definition 4. *We call a multi-rack storage code* $(\mathbf{H}, \mathbf{K})$ *as a* $(\delta_1, \Gamma_1, r_1, \delta_2, \Gamma_2, r_2, a)$ *linear multi-rack storage code if it satisfies the following conditions*:

1) *(Intra-rack resilience) Any* δ_1 *node failures in a rack can be repaired via intra-rack repair;*
2) *(Intra-rack locality) For any* $\Gamma_1 + 1$ *node failure pattern in a rack, each node can be repaired via intra-rack repair, involving at most* r_1 *surviving nodes;*
3) *(Inter-rack resilience) Any* δ_2 *node failures in a rack can be repaired via inter-rack repair.*
4) *(Inter-rack locality) For any* $\Gamma_2 + 1$ *node failure pattern in a rack, each node can be repaired via inter-rack repair such that involving* i*) at most* r_2 *surviving nodes in the failing rack and* ii*) at most* a *nodes from each helper rack.*

The definition for $(\delta_1, r_1, \Gamma_1, r_2, \Gamma_2, a, \delta_2)$ linear multi-rack storage code can be made more precise via the use of support enumerators, to be described as follows.

To simplify our notation, we may use $\mathbf{x}$ and $\mathbf{y}$ instead of $X_{1,*}^{\top}$ and $X_{2,*}^{\top}$. Let

$$\mathcal{C} = \{(\mathbf{x}, \mathbf{y}) : \mathbf{H}\mathbf{x} = \mathbf{H}\mathbf{y} = \mathbf{0},\ \mathbf{K}\mathbf{x} = \mathbf{K}\mathbf{y}\}. \tag{30}$$

We call $\mathcal{C}$ the codebook. Clearly, the design of $\mathcal{C}$ and the design of $(\mathbf{H}, \mathbf{K})$ are equivalent.

Definition 5 (Support)**.** *Consider any codeword* $(\mathbf{x}, \mathbf{y}) \in \mathcal{C}$ *where* $\mathbf{x} = (x_1, \ldots, x_N)$ *and* $\mathbf{y} = (y_1, \ldots, y_N)$*. Its "support"* $\lambda(\mathbf{x}, \mathbf{y})$ *is a tuple* $(\mathbf{w}, \mathbf{s})$ *such that* $\mathbf{w} = (w_1, \ldots, w_N)$ *and* $\mathbf{s} = (s_1, \ldots, s_N)$*, where*

$$w_i = \begin{cases} 1 & \text{if } x_i \neq 0 \\ 0 & \text{if } x_i = 0 \end{cases} \tag{31}$$

$$s_i = \begin{cases} 1 & \text{if } y_i \neq 0 \\ 0 & \text{if } y_i = 0, \end{cases} \tag{32}$$

for all $i = 1, \ldots, N$*. For notation simplicity, we will simply denote that*

$$\lambda(\mathbf{x}, \mathbf{y}) = (\mathbf{w}, \mathbf{s}).$$

Remark 5. *While* $\mathbf{w}$ *and* $\mathbf{s}$ *are subsets of* $\mathcal{N}$*, it is sometimes simpler and more practical to represent them as vectors, as in* (31) *and* (32)*.*

Definition 6 (Support enumerator)**.** *The enumerator function of the code* $\Lambda_{\mathcal{C}}(\mathbf{w}, \mathbf{s})$ *is defined as*

$$\Lambda_{\mathcal{C}}(\mathbf{w}, \mathbf{s}) \triangleq |\{(\mathbf{x}, \mathbf{y}) \in \mathcal{C} : \lambda(\mathbf{x}, \mathbf{y}) = (\mathbf{w}, \mathbf{s})\}| \tag{33}$$

for all $\mathbf{w}, \mathbf{s} \subseteq \mathcal{N}$*.*

The below theorem gives properties of a multi-rack storage code. As we shall see, these properties will form constraints in our linear programming bound.

Theorem 5. *For any* $(\delta_1, \Gamma_1, r_1, \delta_2, \Gamma_2, r_2, a)$ *multi-rack storage code* $\mathcal{C}$*, the support enumerators of* $\mathcal{C}$ *and its dual* $\mathcal{C}^{\perp}$ *satisfy the following properties*:

1) ***Dual codeword support enumerator:***

$$\Lambda_{\mathcal{C}^{\perp}}(\mathbf{w}, \mathbf{s}) = \frac{1}{|\mathcal{C}|} \sum_{\mathbf{w}', \mathbf{s}' \subseteq \mathcal{N}} \Lambda_{\mathcal{C}}(\mathbf{w}', \mathbf{s}') \prod_{j \in \mathcal{N}} \kappa_q(w_j', w_j) \kappa_q(s_j', s_j), \tag{34}$$

where

$$\kappa_q(u, v) = \begin{cases} 1 & \text{if } v = 0 \\ q - 1 & \text{if } u = 0 \text{ and } v = 1 \\ -1 & \text{otherwise.} \end{cases}$$

2) ***Symmetry:*** *For all* $\mathbf{w}, \mathbf{s} \subseteq \mathcal{N}$*,*

$$\Lambda_{\mathcal{C}}(\mathbf{w}, \mathbf{s}) = \Lambda_{\mathcal{C}}(\mathbf{s}, \mathbf{w}). \tag{35}$$

3) ***Intra-rack resilience:*** *For all* $\mathbf{w} \subseteq \mathcal{N}$ *such that* $1 \leq |\mathbf{w}| \leq \delta_1$

$$\Lambda_{\mathcal{C}}(\mathbf{w}, \mathbf{s}) = 0 \tag{36}$$

4) ***Intra-rack locality:*** *For any* $(i, \gamma) \in \Phi(\Gamma_1) \triangleq \{(i, \gamma) : i \in \mathcal{N}, i \notin \gamma$ *and* $|\gamma| = \Gamma_1\}$*,*

$$\sum_{\mathbf{w} \in \Theta_1(i, \gamma, r_1)} \Lambda_{\mathcal{C}^{\perp}}(\mathbf{w}, \emptyset) \geq (q - 1), \tag{37}$$

where $\Theta_1(i, \gamma, r_1) \triangleq \{\mathbf{w} : i \in \mathbf{w}, \mathbf{w} \cap \gamma = \emptyset$ *and* $|\mathbf{w}| \leq r_1 + 1\}$*.*

5) ***Inter-rack resilience:*** *For all* $\mathbf{w} \subseteq \mathcal{N}$ *such that* $1 \leq |\mathbf{w}| \leq \delta_2$

$$\Lambda_{\mathcal{C}}(\mathbf{w}, \emptyset) = 0 \tag{38}$$

6) ***Inter-rack locality:*** *For any* $(i,\gamma) \in \Phi(\Gamma_2) \triangleq \{(i,\gamma) : i \in \mathcal{N}, i \notin \gamma$ *and* $|\gamma| = \Gamma_2\}$,

$$\sum_{(\mathbf{w},\mathbf{s})\in\Theta_2(i,\gamma,r_2,a)} \Lambda_{\mathcal{C}^\perp}(\mathbf{w},\mathbf{s}) \geq (q-1), \tag{39}$$

where $\Theta_2(i,\gamma,r_2,a) \triangleq \{(\mathbf{w},\mathbf{s}) : i \in \mathbf{w}, \mathbf{w}\cap\gamma = \emptyset, |\mathbf{w}| \leq r_2+1$ *and* $|\mathbf{s}| \leq a\}$.

The proof of Theorem 5 is given in Appendix E.

Lemma 2. *Consider a multi-rack storage code* $(\mathbf{H},\mathbf{K},\mathbf{G})$. *Suppose the dimensions of the matrices* $\mathbf{H},\mathbf{K},\mathbf{G}$ *are respectively* $S_1\times N$, $S_2\times N$, *and* $L\times M$. *Then*

$$N - S_1 - S_2 = \log_q \sum_{\mathbf{w}\subseteq\mathcal{N}} \Lambda_{\mathcal{C}}(\mathbf{w},\emptyset) \tag{40}$$

and

$$S_2 = \log_q \sum_{\mathbf{w},\mathbf{s}\subseteq\mathcal{N}} \Lambda_{\mathcal{C}}(\mathbf{w},\mathbf{s}) - 2\log_q \sum_{\mathbf{w}\subseteq\mathcal{N}} \Lambda_{\mathcal{C}}(\mathbf{w},\emptyset). \tag{41}$$

Hence, the rate of the code is

$$R(\Lambda_{\mathcal{C}}) = \frac{\log_q \sum_{\mathbf{w}\subseteq\mathcal{N}} \Lambda_{\mathcal{C}}(\mathbf{w},\emptyset)}{N} + \frac{M-L}{MN}\left(\log_q \sum_{\mathbf{w},\mathbf{s}\subseteq\mathcal{N}} \Lambda_{\mathcal{C}}(\mathbf{w},\mathbf{s}) - 2\log_q \sum_{\mathbf{w}\subseteq\mathcal{N}} \Lambda_{\mathcal{C}}(\mathbf{w},\emptyset)\right). \tag{42}$$

Proof. First, it is clear that $\sum_{\mathbf{w}\subseteq\mathcal{N}} \Lambda_{\mathcal{C}}(\mathbf{w},\emptyset)$ is equal to the size of the following set

$$|\{(\mathbf{x},\mathbf{0}) : \mathbf{Hx} = \mathbf{0},\ \mathbf{Kx} = \mathbf{0}\}|$$

As the dimensions of $\mathbf{x},\mathbf{H}$ and $\mathbf{K}$ are respectively $N\times 1$, $S_1\times N$ and $S_2\times N$, the size of the set is obviously $q^{N-S_1-S_2}$ leading to (40)

Since $\mathbf{H}$ and $\mathbf{K}$ are respectively $S_1\times N$ and $S_2\times N$ parity matrices of the code $\mathcal{C}$ in (30), i) the total number of parity equations is $2S_1+S_2$, and ii) the length of codeword $(\mathbf{x},\mathbf{y})$ is $2N$. Hence, the total number of codewords satisfying (30) is

$$\sum_{\mathbf{w},\mathbf{s}\subseteq\mathcal{N}} \Lambda_{\mathcal{C}}(\mathbf{w},\mathbf{s}) = q^{2N-2S_1-S_2}.$$

Together with (40), we have (41) and (42). □

Remark 6. *According to* (42)*, the rate of the storage code is clearly nonlinear, with respect to the support enumerator* $\Lambda_{\mathcal{C}}$. *However, if we fix*

$$\sum_{\mathbf{w}\subseteq\mathcal{N}} \Lambda_{\mathcal{C}}(\mathbf{w},\emptyset) = O_1,$$

then maximizing $R(\Lambda_{\mathcal{C}})$ *is equivalent to maximizing* $\sum_{\mathbf{w},\mathbf{s}\subseteq\mathcal{N}} \Lambda_{\mathcal{C}}(\mathbf{w},\mathbf{s})$.

Theorem 6 (Upper bound)**.** *Consider fixed* N, M *and* L *and* a $(\delta_1, r_1, \Gamma_1, \delta_2, r_2, \Gamma_2, a)$ *multi-rack storage code* $\mathcal{C}$ *such that*

$$\sum_{\mathbf{w}\subseteq\mathcal{N}} \Lambda_{\mathcal{C}}(\mathbf{w},\emptyset) = O_1.$$

Let O^* *be the maximum of the following linear programming problem.*

Linear Programming Problem (LP1)

Maximize $\sum_{\mathbf{w},\mathbf{s}\subseteq\mathcal{N}} A_{\mathbf{w},\mathbf{s}}$

subject to

$$A_{\mathbf{w},\mathbf{s}} \geq 0, \quad \forall \mathbf{w},\mathbf{s}\subseteq\mathcal{N} \tag{C1}$$

$$A_{\mathbf{w},\mathbf{s}} = A_{\mathbf{s},\mathbf{w}} \tag{C2}$$

$$C_{\mathbf{w},\mathbf{s}} = \sum_{\mathbf{w}',\mathbf{s}'\subseteq\mathcal{N}} A_{\mathbf{w}',\mathbf{s}'} \prod_{j\in\mathcal{N}} \kappa_q(w'_j, w_j)\kappa_q(s'_j, s_j), \quad \forall \mathbf{w},\mathbf{s} \tag{C3}$$

$$C_{\mathbf{w},\mathbf{s}} \geq 0, \quad \forall \mathbf{w},\mathbf{s}\subseteq\mathcal{N} \tag{C4}$$

$$A_{\emptyset,\emptyset} = 1 \tag{C5}$$

$$A_{\mathbf{w},\mathbf{s}} = 0, \quad \forall 1 \leq |\mathbf{w}| \leq \delta_1 \tag{C6}$$

$$A_{\mathbf{w},\emptyset} = 0, \quad \forall 1 \leq |\mathbf{w}| \leq \delta_2 \tag{C7}$$

$$\sum_{\mathbf{w}\in\Theta_1(i,\gamma,r_1)} C_{\mathbf{w},\emptyset} \geq (q-1)\sum_{\mathbf{w},\mathbf{s}} A_{\mathbf{w},\mathbf{s}}, \quad \forall (i,\gamma)\in\Phi(\Gamma_1) \tag{C8}$$

$$\sum_{(\mathbf{w},\mathbf{s})\in\Theta_2(i,\gamma,r_2,a)} C_{\mathbf{w},\mathbf{s}} \geq (q-1)\sum_{\mathbf{w},\mathbf{s}} A_{\mathbf{w},\mathbf{s}}, \quad \forall (i,\gamma)\in\Phi(\Gamma_2) \tag{C9}$$

$$\sum_{\mathbf{w}\subseteq\mathcal{N}} A_{\mathbf{w},\emptyset} = O_1. \tag{C10}$$

Then $R(\Lambda_{\mathcal{C}})$ *is upper bound by*

$$\frac{\log_q O_1}{N} + \frac{M-L}{MN}\left(\log_q O^* - 2\log_q O_1\right).$$

Proof of Theorem 6. We define

$$A_{\mathbf{w},\mathbf{s}} \triangleq \Lambda_{\mathcal{C}}(\mathbf{w},\mathbf{s})$$
$$C_{\mathbf{w},\mathbf{s}} \triangleq |\mathcal{C}|\Lambda_{\mathcal{C}^\perp}(\mathbf{w},\mathbf{s}).$$

As mentioned earlier, maximizing the code rate is equivalent to maximize the code size which is the objective function of the linear programming problem LP1. The first constraint of the optimization problem LP1 follows from the fact that the number of codewords are non-negative. The second constraint follows from the symmetry property of the code. The constraint (C3) follows from the dual code support enumerator property (MacWilliam's identity) in Theorem 5. Constraint (C4) follows from the fact that the number of dual codewords are non-negative. Constriant (C5) follows from the fact that there exists only one zero codeword in code $\mathcal{C}$. Constraints (C6)–(C9) follow from the properties 3–6 in Theorem 5. □

Remark 7. *Strictly speaking, to optimize* $R(\Lambda_{\mathcal{C}})$*, one also needs to optimize the choice of* O_1*, which is generally unknown. However, as* $O_1 \in \{q^i,\ i = 0,\ldots,N\}$. *Hence, we have*

$$R(\Lambda_{\mathcal{C}}) \leq \max_{i=0,\ldots,N} \frac{i}{N} + \frac{M-L}{MN}\left(\log_q O^*(i) - 2\ i\right)$$

where $O^*(i)$ *is the maximum of (LP1) when* $O_1 = q^i$.

B. Bound Simplification via Symmetry

The complexity (in terms of the number of variables and the number of constraints) of the linear programming problem LP1 in Theorem 6 will increase exponentially with the number of storage nodes N in each rack. However, if we notice the LP1 carefully, we can observe that the problem itself has much symmetries in it such that exploiting this inherent symmetry can significantly reduce the problem complexity.

Let $S_{\mathcal{N}}$ be the symmetric group on $\mathcal{N}$, whose elements are all the permutations of the elements in $\mathcal{N}$ which are treated as bijective functions from the set of symbols to itself. Clearly, $|S_{\mathcal{N}}| = N!$. The variables in the optimization problem LP1 are

$$(A_{\mathbf{w},\mathbf{s}}, C_{\mathbf{w},\mathbf{s}},\ \mathbf{w},\mathbf{s} \subseteq \mathcal{N}).$$

Let σ be a permutation on $\mathcal{N}$ such that $\sigma \in S_{\mathcal{N}}$. For each $\mathbf{w} \subseteq \mathcal{N}$, we extend the mapping σ by defining

$$\sigma(\mathbf{w}) \triangleq \{\sigma(i) :\ i \in \mathbf{w}\}.$$

Due to the symmetries, we have the following proposition.

Proposition 1. *Suppose* $(a_{\mathbf{w},\mathbf{s}}, c_{\mathbf{w},\mathbf{s}} :\ \mathbf{w},\mathbf{s} \subseteq \mathcal{N})$ *satisfies all the constraints in the linear programming problem LP1 in Theorem 6. For any* $\sigma \in S_{\mathcal{N}}$, *let*

$$a^{(\sigma)}_{\mathbf{w},\mathbf{s}} = a_{\sigma(\mathbf{w}),\sigma(\mathbf{s})} \tag{43}$$

$$c^{(\sigma)}_{\mathbf{w},\mathbf{s}} = c_{\sigma(\mathbf{w}),\sigma(\mathbf{s})}. \tag{44}$$

Then $(a^{(\sigma)}_{\mathbf{w},\mathbf{s}}, c^{(\sigma)}_{\mathbf{w},\mathbf{s}} :\ \mathbf{w},\mathbf{s} \subseteq \mathcal{N})$ *also satisfies the constraints in LP1, with the same values in the objective function. In other words,*

$$\sum_{\mathbf{w},\mathbf{s}\subseteq\mathcal{N}} a^{(\sigma)}_{\mathbf{w},\mathbf{s}} = \sum_{\mathbf{w},\mathbf{s}\subseteq\mathcal{N}} a_{\mathbf{w},\mathbf{s}}.$$

Proof. The proposition follows directly from the symmetry in the constraint and optimizing function. □

As the feasible region in the linear programming problem (LP1) is convex, we have the following corollary.

Corollary 1. *Let*

$$a^{*}_{\mathbf{w},\mathbf{s}} = \frac{1}{|S_{\mathcal{N}}|}\sum_{\sigma\in S_{\mathcal{N}}} a^{(\sigma)}_{\mathbf{w},\mathbf{s}} \tag{45}$$

$$c^{*}_{\mathbf{w},\mathbf{s}} = \frac{1}{|S_{\mathcal{N}}|}\sum_{\sigma\in S_{\mathcal{N}}} c^{(\sigma)}_{\mathbf{w},\mathbf{s}} \tag{46}$$

Then $(a^{*}_{\mathbf{w},\mathbf{s}}, c^{*}_{\mathbf{w},\mathbf{s}} :\ \mathbf{w},\mathbf{s} \subseteq \mathcal{N})$ *also satisfies the constraints in (LP1) and*

$$\sum_{\mathbf{w},\mathbf{s}\subseteq\mathcal{N}} a^{*}_{\mathbf{w},\mathbf{s}} = \sum_{\mathbf{w},\mathbf{s}\subseteq\mathcal{N}} a_{\mathbf{w},\mathbf{s}}. \tag{47}$$

Proof. From Proposition 1, for any feasible solution $(a_{\mathbf{w},\mathbf{s}}, c_{\mathbf{w},\mathbf{s}} : \mathbf{w},\mathbf{s} \subseteq \mathcal{N})$, there exist $|S_{\mathcal{N}}|$ other feasible solution $(a^{(\sigma)}_{\mathbf{w},\mathbf{s}}, c^{(\sigma)}_{\mathbf{w},\mathbf{s}} :\ \mathbf{w},\mathbf{s} \subseteq \mathcal{N}, \sigma \in S_{\mathcal{N}})$. Since $(a^{*}_{\mathbf{w},\mathbf{s}}, c^{*}_{\mathbf{w},\mathbf{s}})$ is the convex linear combination of all these feasible solutions, it also satisfies the constraints of LP1. From (45)

$$\sum_{\mathbf{w},\mathbf{s}\subseteq\mathcal{N}} a^{*}_{\mathbf{w},\mathbf{s}} = \frac{1}{|S_{\mathcal{N}}|}\sum_{\mathbf{w},\mathbf{s}\subseteq\mathcal{N}}\sum_{\sigma\in S_{\mathcal{N}}} a^{(\sigma)}_{\mathbf{w},\mathbf{s}}.$$

Moreover, from Proposition 1

$$\sum_{\mathbf{w},\mathbf{s}\subseteq\mathcal{N}}\sum_{\sigma\in S_{\mathcal{N}}} a^{(\sigma)}_{\mathbf{w},\mathbf{s}} = |S_{\mathcal{N}}|\sum_{\mathbf{w},\mathbf{s}\subseteq\mathcal{N}} a_{\mathbf{w},\mathbf{s}}.$$

The corollary then follows. □

By Corollary 1, it is sufficient to consider only "symmetric" feasible solutions of the form $(a^{*}_{\mathbf{w},\mathbf{s}}, c^{*}_{\mathbf{w},\mathbf{s}} :\ \mathbf{w},\mathbf{s} \subseteq \mathcal{N})$. in LP1. In other words, one can impose additional symmetric constraint to LP1 without affecting the value of the objective function.

One important benefit for considering $(a^{*}_{\mathbf{w},\mathbf{s}}, c^{*}_{\mathbf{w},\mathbf{s}} :\ \mathbf{w},\mathbf{s} \subseteq \mathcal{N})$. is that many terms in the linear programming bound can become alike (and hence can be grouped together).

Proposition 2 (Grouping alike terms)**.** *Suppose* $\mathbf{w},\mathbf{s},\mathbf{w}',\mathbf{s}' \subseteq \mathcal{N}$ *such that*

$$|\mathbf{w}\setminus\mathbf{s}| = |\mathbf{w}'\setminus\mathbf{s}'| \tag{48}$$

$$|\mathbf{w}\cap\mathbf{s}| = |\mathbf{w}'\cap\mathbf{s}'| \tag{49}$$

$$|\mathbf{s}\setminus\mathbf{w}| = |\mathbf{s}'\setminus\mathbf{w}'|. \tag{50}$$

Then $a^{*}_{\mathbf{w},\mathbf{s}} = a^{*}_{\mathbf{w}',\mathbf{s}'}$ *and* $c^{*}_{\mathbf{w},\mathbf{s}} = c^{*}_{\mathbf{w}',\mathbf{s}'}$.

The proof of Proposition 2 is given in Appendix D.

Due to Proposition 2, we can impose the following additional constraint on (LP1)

$$A_{\mathbf{w},\mathbf{s}} = A_{\mathbf{w}',\mathbf{s}'} \tag{51}$$

$$C_{\mathbf{w},\mathbf{s}} = C_{\mathbf{w}',\mathbf{s}'} \tag{52}$$

for all $\mathbf{w},\mathbf{s}$ satisfying (48) - (50). As many of these variables are now the same, one can greatly reduce the number of variables in (LP1).

Theorem 7 (Simplified LP Bound)**.** *The maximum in (LP1) is the same as the maximum of the following linear programming problem:*

Reduced Linear Programming Problem (LP2)

maximize $\displaystyle\sum_{d,e,f}\binom{N}{d,e,f}X_{d,e,f}$

subject to

$$X_{d,e,f} \geq 0, \qquad \forall d,e,f \tag{D1}$$

$$X_{d,e,f} = X_{f,e,d}, \qquad \forall d,e,f \tag{D2}$$

$$Y_{d,e,f} = \sum_{d',e',f'} \Delta_1(d,e,f,d',e',f')X_{d',e',f'}, \quad \forall d,e,f \tag{D3}$$

$$Y_{d,e,f} \geq 0, \qquad \forall d,e,f \tag{D4}$$

$$X_{0,0,0} = 1 \tag{D5}$$

$$X_{d,e,f} = 0, \qquad \forall 1 \leq d+e \leq \delta_1 \tag{D6}$$

$$X_{d,0,0} = 0, \qquad \forall 1 \leq d \leq \delta_2 \tag{D7}$$

$$\sum_{d=2}^{r_1+1}\Delta_2(d)Y_{d,0,0} \geq (q-1)\sum_{d,e,f}\binom{N}{d,e,f}X_{d,e,f}, \tag{D8}$$

$$\sum_{d+e\le r_2+1, e+f\le a} \Delta_3(d,e,f)Y_{d,e,f} \ge (q-1)\sum_{d,e,f}\binom{N}{d,e,f}X_{d,e,f}, \tag{D9}$$

$$\sum_{d} X_{d,0,0} = O_1 \tag{D10}$$

where $\Delta_1(d,e,f,d',e',f')$, $\Delta_2(d)$ *and* $\Delta_3(d,e,f)$ *are respectively defined as in* (104), (87) *and* (91). *Here,* (d,e,f) *are tuples such that* d,e,f *are nonnegative integers with a total sum no more than* N.

The proof of Theorem 7 is given in Appendix F.

Remark 8. *In (LP1), the number of variables and constraints grows exponentially with* N *– the number of nodes in a rack. Such exponential growth makes (LP1) practically infeasible to solve for moderate* N. *However, via reduction by symmetry, the number of variables in (LP2) has greatly reduced to* $2\binom{N+3}{3}+1$ *while the number of constraints to* $\frac{7}{2}\binom{N+3}{3}+5+\delta_1+\delta_2$. *Clearly, the reduction is significant.*

Example 4. *We consider a very simple setup to show how some of the constraints in linear programming bound in (LP2) are calculated. Let,* $N=8$ *is the number of the storage nodes in each rack. As mentioned earlier in Section III-A parity matrix* $\mathbf{G}$ *can be design separately, hence the number of the racks* M *will not affect the LP bound. Moreover, let* $\Gamma_1=2$, $\delta_1=3$, $r_1=3$, $\Gamma_2=4$, $\delta_2=6$, $r_2=1$, *and* $a=3$. *Then, we have* $(0\le d+e+f\le 8, 0\le d,e,f\le 8)$ *and* $(0\le d'+e'+f'\le 8, 0\le d',e',f'\le 8)$. *Here, we show how to calculate* $\Delta_1(d,e,f,d',e',f')$, $\Delta_2(d)$, *and* $\Delta_3(d,e,f)$. *Following* (87) *and* (91), *respectively,*

$$\Delta_2(d)=\binom{5}{d-1},$$

$$\Delta_3(d,e,f)=\binom{3}{e-1}\binom{3-e}{d}\binom{8-e-d}{f}+\binom{3}{e}\binom{3-e}{d-1}\binom{8-e-d}{f}.$$

In order to calculate $\Delta_1(d,e,f,d',e',f')$ *from* (104), *we need to find all possible values of* $(\zeta_{i,j}, 1\le i,j\le 4)$ *for each fixed tuples of* (d,e,f) *and all tuples of* (d',e',f') *such that* $d=\sum_j \zeta_{1,j}$, $e=\sum_j \zeta_{2,j}$, $f=\sum_j \zeta_{3,j}$, $d'=\sum_j \zeta_{j,1}$, $e'=\sum_j \zeta_{j,2}$, $f'=\sum_j \zeta_{j,3}$. *Then* $U(\zeta)$, $\sigma_1(\zeta)$, *and* $\sigma_2(\zeta)$ *can be calculated from* (98), (99), *and* (103), *respectively. For instance, let* $(d,e,f)=(5,2,1)$. *Then for a tuple* $(d',e',f')=(3,1,0)$, *one possible combinations for* $\zeta_{i,j}$ *will be* $(\zeta_{1,1}=2,\zeta_{1,2}=0,\zeta_{1,3}=0,\zeta_{1,4}=3,\zeta_{2,1}=1,\zeta_{2,2}=0,\zeta_{2,3}=0,\zeta_{2,4}=1,\zeta_{3,1}=0,\zeta_{3,2}=1,\zeta_{3,3}=0,\zeta_{3,4}=0,\zeta_{4,1}=0,\zeta_{4,2}=0,\zeta_{4,3}=0,\zeta_{4,4}=0)$. *Therefore,*

$$\sigma_1(\zeta)=6,\quad \sigma_2(\zeta)=4,$$
$$U(\zeta)=\binom{5}{2,0,0,3}\binom{2}{1,0,0,1}\binom{1}{0,1,0,0}\binom{0}{0,0,0,0}=20.$$

Another possible combination will be $(\zeta_{1,1}=1,\zeta_{1,2}=0,\zeta_{1,3}=0,\zeta_{1,4}=4,\zeta_{2,1}=1,\zeta_{2,2}=1,\zeta_{2,3}=0,\zeta_{2,4}=0,\zeta_{3,1}=1,\zeta_{3,2}=0,\zeta_{3,3}=0,\zeta_{3,4}=0,\zeta_{4,1}=0,\zeta_{4,2}=0,\zeta_{4,3}=0,\zeta_{4,4}=0)$. *Therefore,*

$$\sigma_1(\zeta)=6,\quad \sigma_2(\zeta)=4,$$
$$U(\zeta)=\binom{5}{1,0,0,4}\binom{2}{1,1,0,0}\binom{1}{1,0,0,0}\binom{0}{0,0,0,0}=10.$$

	$d'=3$ $e'=1$	$f'=0$	$N-d'-e'-f'=4$
$d=5$	2 1 $\zeta_{1,1}$	0 0 $\zeta_{1,2}$	0 0 $\zeta_{1,3}$	3 4 $\zeta_{1,4}$
$e=2$	1 1 $\zeta_{2,1}$	0 1 $\zeta_{2,2}$	0 0 $\zeta_{2,3}$	1 0 $\zeta_{2,4}$
$f=1$	0 1 $\zeta_{3,1}$	1 0 $\zeta_{3,2}$	0 0 $\zeta_{3,3}$	0 0 $\zeta_{3,4}$
$N-d-e-f=0$	0 0 $\zeta_{4,1}$	0 0 $\zeta_{4,2}$	0 0 $\zeta_{4,3}$	0 0 $\zeta_{4,4}$

Fig. 3. An illustration on selecting different combinations for $\zeta_{i,j}$s.

As mentioned earlier, all values of $\zeta_{i,j}$ *must be calculated (this can be done by a computer simulation code e.g. in MATLAB) for all* (d,e,f) *and* (d',e',f') *in order to calculate* $\Delta_1(d,e,f,d',e',f')$. *Figure 3 illustrates of selecting the different combinations for* $\zeta_{i,j}$s. *The numbers in red and green are the two aforementioned different possible combinations for* $\zeta_{i,j}$s. *The combinations will be chosen such that the sum of the rows and columns satisfy the values of* (d,e,f) *and* (d',e',f').

IV. Conclusion

In this paper we introduced a code-design framework for multi-rack distributed data storage networks. In this model the encoded data is stored in storage nodes distributed over multiple racks. Each rack has a process unit which is responsible for all calculations and transmissions inside the rack. Practically, the cost of data transmission within a rack is much less than the data transmission across the racks. Therefore, it will be a significant reduction in the repair bandwidth if the node failures are repaired locally inside the racks. Our multi-rack distributed storage code is defined with three parity check matrices $\mathbf{H}$, $\mathbf{K}$, and $\mathbf{G}$. We proposed a code-design framework for multi-rack storage networks which is able to locally repair the node failures within the rack using the parity check matrix $\mathbf{H}$ in order to minimise the repair cost. However, under the severe failure circumstances where the failures are not repairable by only the survived nodes inside the rack, our coding scheme is capable of engaging the storage nodes in other racks in helping the surviving nodes inside the failed rack to repair the failures where the parity check matrices $\mathbf{G}$ and $\mathbf{K}$ determine the helper racks and the helper nodes, respectively. We justify that the parity matrix $\mathbf{G}$ can be designed separately from the matrices $\mathbf{H}$ and $\mathbf{K}$ where indeed it can be designed as the parity check matrix of a locally repairable code such that only a small group of racks need to participate in an inter-rack repair. We established the relation between the rate and the size of the multi-rack storage code and showed that maximizing the rate of our multi-rack storage code is

equivalent to maximizing the code size. In order to maximize the code size, we established a linear programming problem based on the code-design framework criteria. These bounds characterize the trade-off between different code parameters such as inter- and intra-rack resilience and locality. We also exploited symmetries in the linear programming problem for simplifying and significant reductions in its complexity.

Appendix A
Proof of Theorem 2

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

$$\mathbf{r}_j = \mathbf{u}\mathbf{K}. \tag{53}$$

In other words, $\mathbf{r}_j$ is a vector from row space of $\mathbf{K}$. From Lemma 1, as $\beta_j \in \Omega(\mathbf{H}, \mathbf{r}_j, j)$ by Criterion 3, there exist row vectors $\mathbf{y}$, $\mathbf{y}'$ and $a \in \mathbb{F}_q$ such that

$$\mathbf{e}_j = \mathbf{y}\mathbf{H} + \mathbf{y}'\mathbf{I}^{\beta_j} + a\mathbf{r}_j. \tag{54}$$

Now, notice

$$\begin{aligned} X_{1,j} &= \mathbf{e}_j X_{1,*}^\top && (55)\\ &= (\mathbf{y}\mathbf{H} + \mathbf{y}'\mathbf{I}^{\beta_j} + a\mathbf{r}_j)X_{1,*}^\top && (56)\\ &= \mathbf{y}'\mathbf{I}^{\beta_j} X_{1,*}^\top + a\mathbf{r}_j X_{1,*}^\top && (57)\end{aligned}$$

where the last equality follows from (2). Let

$$[c_{j,1}, \ldots, c_{j,N}] = \mathbf{y}'\mathbf{I}^{\beta_j}. \tag{58}$$

Then

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

Consequently,

$$X_{1,j} = \sum_{n\in\beta_j} c_{j,n} X_{1,n} + a\mathbf{r}_j X_{1,*}^\top. \tag{59}$$

Let $\mathbf{f}_\ell = [f_1, \ldots, f_M]$, $\ell \in \{1, \ldots, M\}$, be a length M row vector such that

$$f_i = \begin{cases} 1 & \text{if } i = \ell \\ 0 & \text{otherwise.} \end{cases} \tag{60}$$

Then

$$X_{1,*} = \mathbf{f}_1 X$$

and hence

$$\begin{aligned} a\mathbf{r}_j X_{1,*}^\top &= aX_{1,*}\mathbf{r}_j^\top && (61)\\ &= a\mathbf{f}_1 X \mathbf{r}_j^\top. && (62)\end{aligned}$$

From Lemma 1, since $\tau \in \Omega(\mathbf{G}, 1)$, there exist vectors $\mathbf{z}$ and $\mathbf{z}'$ such that

$$\mathbf{f}_1 = \mathbf{z}\mathbf{G} + \mathbf{z}'\mathbf{I}^\tau. \tag{63}$$

Now, notice that $\mathbf{I}^\tau$ is a $M \times M$ matrix over $\mathbb{F}_q$, as $\mathbf{G}$ has only M columns. Consequently,

$$\begin{aligned} a\mathbf{r}_j X_{1,*}^\top &= a(\mathbf{z}\mathbf{G} + \mathbf{z}'\mathbf{I}^\tau)X\mathbf{r}_j^\top && (64)\\ &= a\mathbf{z}'\mathbf{I}^\tau X\mathbf{r}_j^\top && (65)\end{aligned}$$

where the last equality follows from that

$$\begin{aligned} \mathbf{z}\mathbf{G}X\mathbf{r}_j^\top &= \mathbf{z}\mathbf{G}X\mathbf{K}^\top\mathbf{u}^\top && (66)\\ &= \mathbf{z}\left(\mathbf{G}X\mathbf{K}^\top\right)\mathbf{u}^\top && (67)\\ &= 0, && (68)\end{aligned}$$

where the last equality follows from (3).

Let the matrix

$$\begin{aligned} \begin{bmatrix} d_{j,1,1} & \cdots & d_{j,1,N} \\ \vdots & \ddots & \vdots \\ d_{j,M,1} & \cdots, & d_{j,M,N} \end{bmatrix} &= a(\mathbf{z}'\mathbf{I}^\tau)^\top \mathbf{r}_j && (69)\\ &= a\mathbf{I}^\tau(\mathbf{z}')^\top \mathbf{r}_j, && (70)\end{aligned}$$

as $\mathbf{I}^\tau$ is a diagonal matrix. If $\mathbf{r}_j = [r_{j,1}, \ldots, r_{j,N}]$ and $\mathbf{z}'\mathbf{I}^\tau \triangleq \mathbf{v} \triangleq [v_1, \ldots, v_M]$, then it can be verified easily that

1) $d_{j,m,n} = av_m r_n$ and hence $d_{j,m,n} = 0$ if either $m \notin \tau$ or $n \notin \mu_j$.
2)

$$a\mathbf{r}_j X_{1,*}^\top = \sum_{m\in\tau}\left(\sum_{n\in\mu_j} d_{j,m,n} X_{m,n}\right). \tag{71}$$

Thus the theorem is then proved.

Appendix B
Proof of Theorem 3

First, we will consider intra-rack transmissions. In each helper rack (say $m \in \tau$) involved in the inter-rack repair, $|\mu_j|$ nodes in it will need to transmit its data ($X_{m,s}$ where $s \in \mu_j$) to the processing unit in the helper rack. So, the set of symbols sent from the nodes to the processing unit is $\bigcup_{j\in\gamma\setminus\alpha}\mu_j$. Consequently, the total number of symbols sent to the processing units of helper rack is equal to $|\tau|\,|\bigcup_{j\in\gamma\setminus\alpha}\mu_j|$ where $|\tau|$ is the number of helper racks. That explains the first term in LHS of (22). Next, at each helper rack m, it will need to transmit

$$\{\mathbf{r}_j X_{m,*} : j \in \gamma \setminus \alpha\}$$

to the failing rack. As these symbols may be linearly dependent, the actual number of symbols that really needed to be transmitted is only $\dim\langle \mathbf{r}_j, j \in \gamma \setminus \alpha\rangle$. Therefore, the total number of inter-rack transmission is $|\tau|\dim\langle \mathbf{r}_j, j \in \gamma \setminus \alpha\rangle$. This explains the upper bound on (23).

After receiving the transmissions from the helper racks, the processing unit of the failed rack can now aim to recover the failed nodes. For each $j \in \gamma$, it requires transmission from nodes in the set β_j. Therefore, the number of intra-rack transmissions in rack 1 from nodes to the processing unit is $|\cup_{j\in\gamma}\beta_j|$. This corresponds to the second term in RHS of (22). Finally, receiving all the symbols, the processing nodes can recover the contents of all the failed nodes. It requires $|\gamma|$ intra-rack transmissions from the processing unit of the failed rack to the failed nodes for recovery, explaining the last term of LHS of (22). The theorem thus proved.

APPENDIX C
PROOF OF THEOREM 4

According to (2) and (3), the total number of parity check equations is at most $MS_1 + LS_2$ while the number of variables is MN. Therefore, the rate of the code is at least

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

Now, Consider the following set

$$\mathcal{S}_m(\mathbf{b}_m) = \left\{ X_{m,*}^\top : \mathbf{H}X_{m,*}^\top = \mathbf{0} \text{ and } \mathbf{K}X_{m,*}^\top = \mathbf{b}_m \right\}.$$

In other words, $\mathcal{S}_m(\mathbf{b}_m)$ is the set of solutions for the system of linear equations specified by parity check matrices $\mathbf{H}$ and $\mathbf{K}$ in (2) and (3), respectively. From linear algebra, the number of these solutions for any $\mathbf{b}_m$ is given by

$$|\mathcal{S}_m(\mathbf{b}_m)| = q^{N-S_1-S_2},$$

where q is the field size. Similarly, let

$$S(\mathbf{B}) = \left\{ X : \mathbf{H}X^\top = \mathbf{0} \text{ and } \mathbf{K}X^\top = \mathbf{B} \right\}. \tag{72}$$

where $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_M]$. Then $|S(\mathbf{B})| = q^{M(N-S_1-S_2)}$ for any $\mathbf{B}$.

Now, consider the set

$$\mathcal{S} \triangleq \{X : X \text{ satisfies (2) and (3)}\}.$$

Then it is clear that

$$\begin{aligned} |\mathcal{S}| &= \sum_{\mathbf{B}:\, \mathbf{B}\mathbf{G}^\top = \mathbf{0}} |S(\mathbf{B})| \\ &= |\Delta| q^{M(N-S_1-S_2)} \end{aligned}$$

where $\Delta = \left\{ \mathbf{B} : \mathbf{B}\mathbf{G}^\top = \mathbf{0} \right\}$.

As the rank of $\mathbf{G}$ is L and $\mathbf{B}$ is a matrix of size $S_2 \times M$, $|\Delta| = q^{(M-L)S_2}$. Consequently,

$$|\mathcal{S}| = q^{(M-L)S_2} q^{M(N-S_1-S_2)} \tag{73}$$

$$= q^{MN-MS_1-LS_2}. \tag{74}$$

The theorem then follows.

APPENDIX D
PROOF OF PROPOSITION 2

By definitions,

$$a^*_{\mathbf{w},\mathbf{s}} = \frac{1}{|S_\mathcal{N}|} \sum_{\sigma \in S_\mathcal{N}} a^{(\sigma)}_{\mathbf{w},\mathbf{s}} \tag{75}$$

$$a^*_{\mathbf{w}',\mathbf{s}'} = \frac{1}{|S_\mathcal{N}|} \sum_{\sigma \in S_\mathcal{N}} a^{(\sigma)}_{\mathbf{w}',\mathbf{s}'}. \tag{76}$$

Now, by (48)–(50), there exists a permutation $\mu \in S_\mathcal{N}$ such that $\mathbf{w}' = \mu(\mathbf{w})$ and $\mathbf{s}' = \mu(\mathbf{s})$. Hence, for any permutation $\sigma \in S_\mathcal{N}$, we have

$$\sigma(\mathbf{w}') = \sigma(\mu(\mathbf{w})) = (\sigma \circ \mu)(\mathbf{w}) \tag{77}$$

Similarly, we have $\sigma(\mathbf{s}') = (\sigma \circ \mu)(\mathbf{s})$. Consequently,

$$a^*_{\mathbf{w}',\mathbf{s}'} = \frac{1}{|S_\mathcal{N}|} \sum_{\sigma \in S_\mathcal{N}} a^{(\sigma)}_{\mathbf{w}',\mathbf{s}'} \tag{78}$$

$$= \frac{1}{|S_\mathcal{N}|} \sum_{\sigma \in S_\mathcal{N}} a_{\sigma(\mathbf{w}'),\sigma(\mathbf{s}')} \tag{79}$$

$$= \frac{1}{|S_\mathcal{N}|} \sum_{\sigma \in S_\mathcal{N}} a_{(\sigma\circ\mu)(\mathbf{w}),(\sigma\circ\mu)(\mathbf{s})} \tag{80}$$

$$= \frac{1}{|S_\mathcal{N}|} \sum_{\sigma \in S_\mathcal{N}} a_{\sigma(\mathbf{w}),\sigma(\mathbf{s})} \tag{81}$$

$$= a^*_{\mathbf{w},\mathbf{s}} \tag{82}$$

where the second last equality follows from the fact that $S_\mathcal{N}$ is a group and hence

$$\{\sigma \circ \mu : \sigma \in S_\mathcal{N}\} = \{\sigma : \sigma \in S_\mathcal{N}\}.$$

Similarly, we can also prove that $c^*_{\mathbf{w},\mathbf{s}} = c^*_{\mathbf{w}',\mathbf{s}'}$.

APPENDIX E
PROOF OF THEOREM 5

Recalling our assumptions in Section III, any codeword from the rack model storage code $\mathcal{C}$ is in the form of $(\mathbf{x}, \mathbf{y}) \in \mathcal{C}$ satisfying (30). The support enumerator of the dual code $\mathcal{C}^\perp$ in (34) follows from the MacWilliam's identity [44] . Equation (35) follows directly from definition in (30). From the intra-rack resilience property in Definition 4, any δ_1 simultaneous failures in a rack can be repaired via intra-rack repairs. Now consider $\mathbf{w} \subseteq \mathcal{N}$ such that $1 \le |\mathbf{w}| \le \delta_1$. Suppose to the contrary that (36) does not hold, i.e., $\Lambda_\mathcal{C}(\mathbf{w}, \mathbf{s}) \ge 1$ for some $\mathbf{s} \subseteq \mathcal{N}$. Assume without loss of generality that $\mathbf{w} = \{1, \ldots, |\mathbf{w}|\}$. Then there exists a codeword $\mathbf{c} = [x_1, \ldots, x_N, y_1, \ldots y_N]$ such that

1) $[x_1, \ldots, x_N] \neq [0, \ldots, 0]$ and hence $[x_1, \ldots, x_N, y_1, \ldots y_N] \neq [0, \ldots, 0]$
2) $x_i = 0$ for all $N \ge i \ge |\mathbf{w}| + 1$

Hence, if the nodes $(x_1, \ldots, x_{\delta_1})$ fail, then it is impossible to perfectly recover x_i for all $N \ge i \ge |\mathbf{w}| + 1$ since the failing rack cannot distinguish whether the stored symbol was $[x_1, \ldots, x_N]$ or $[0, \ldots, 0]$. Therefore, we prove (36).

Follows from the inter-rack resilience property in Definition 4, any δ_2 simultaneous failures in one rack can be recovered by the survived nodes in the same rack and all the nodes in the other rack. Again, suppose to the contrary that $\Lambda_\mathcal{C}(\mathbf{w}, \emptyset) \ge 1$ for some $\mathbf{w} \subseteq \mathcal{N}$ such that $1 \le |\mathbf{w}| \le \delta_2$. Assume without loss of generality that $\mathbf{w} = \{1, \ldots, |\mathbf{w}|\}$. Then there exists a codeword $\mathbf{c} = [x_1, \ldots, x_N, 0, \ldots 0]$ such that

1) $[x_1, \ldots, x_N] \neq [0, \ldots, 0]$ and hence $[x_1, \ldots, x_N, 0, \ldots, 0] \neq [0, \ldots, 0]$
2) $x_i = 0$ for all $N \ge i \ge |\mathbf{w}| + 1$

Hence, if nodes $x_1, \ldots, x_{\delta_2}$ fail, then it is impossible to perfectly recover x_i for all $N \ge i \ge |\mathbf{w}| + 1$. Thus, we prove (38).

From the intra-rack locality property, if the number of failures in a rack is at most $\Gamma_1 + 1$, then each node can be repaired via intra-rack repair, involving at most r_1 surviving

nodes. Now, suppose that the set of failing nodes is $\gamma \cup \{i\}$ where $|\gamma| = \Gamma_1$. In order to repair the failed nodes x_i via intra-rack repair, there must exists a dual codeword $(\mathbf{f}', \mathbf{0}) \in \mathcal{C}^\perp$ whose support is $(\mathbf{w}', \emptyset)$ for some $\mathbf{w}' \in \Theta_1(i, \gamma, r_1)$. Hence,

$$\sum_{\mathbf{w} \in \Theta_1(i,\gamma,r_1)} \Lambda_{\mathcal{C}^\perp}(\mathbf{w}, \emptyset) \geq 1 \tag{83}$$

In fact, since supports of $(\mathbf{f}', \mathbf{0})$ and $(c \cdot \mathbf{f}', \mathbf{0})$ are the same for all non-zero $c \in \mathbb{F}_q$, we have

$$\sum_{\mathbf{w} \in \Theta_1(i,\gamma,r_1)} \Lambda_{\mathcal{C}^\perp}(\mathbf{w}, \emptyset) \geq q - 1 \tag{84}$$

and (37) is proved.

Finally, from the inter-rack locality property, if the number of failures in a rack is at most $\Gamma_2 + 1$, then each node can be repaired via inter-rack repair such that involving i) at most r_2 surviving nodes in the failing rack and ii) at most a nodes from each helper rack. Now, suppose that the set of failing nodes is $\gamma \cup \{i\}$ where $|\gamma| = \Gamma_2$. In order to repair the failed nodes x_i via intra-rack repair, there must exists a dual codeword $(\mathbf{f}', \mathbf{g}') \in \mathcal{C}^\perp$ whose support is $(\mathbf{w}', \mathbf{s}') \in \Theta_2(i, \gamma, r_2, a)$. Again, as support of $(\mathbf{f}', \mathbf{g}')$ and $(c\mathbf{f}', c\mathbf{g}')$ for all non-zero $c \in \mathbb{F}_q$, (39) thus follows.

Appendix F
Proof of Theorem 7

The simplified bound is obtained by respectively replacing $A_{\mathbf{w},\mathbf{s}}$ and $C_{\mathbf{w},\mathbf{s}}$ with newly introduced variables $X_{d,e,f}$ and $Y_{d,e,f}$ where

$$d = |\mathbf{w} \setminus \mathbf{s}|, \quad e = |\mathbf{w} \cap \mathbf{s}|, \text{ and } f = |\mathbf{s} \setminus \mathbf{w}|. \tag{85}$$

The replacement is possible, due to Proposition 2.

With the new notations, some terms will become alike and can be grouped together. For example, the term $\sum_{\mathbf{w},\mathbf{s} \subseteq \mathcal{N}} A_{\mathbf{w},\mathbf{s}}$ can be replaced by $\sum_{d,e,f} \binom{N}{d,e,f} X_{d,e,f}$. Moreover, some constraints will become equivalent and hence can be omitted. Most constraints in (LP1) can be rewritten directly. The more complicated ones are (C3), (C8) and (C9). In the following, we will illustrate how to simplify and rewrite them.

We now first "reduce" constraint (C8). The key idea of reduction is described as follows: First, by Proposition 2, $C_{\mathbf{w},\emptyset} = C_{\mathbf{w}',\emptyset}$ if $|\mathbf{w}| = |\mathbf{w}'|$. In fact, by (85),

$$C_{\mathbf{w},\emptyset} = Y_{|\mathbf{w}|,0,0}$$

Consider any given $(i, \gamma) \in \Phi(\Gamma_1)$. Recall that

$$\Theta_1(i, \gamma, r_1) \triangleq \{\mathbf{w} : i \in \mathbf{w}, \mathbf{w} \cap \gamma = \emptyset \text{ and } |\mathbf{w}| \leq r_1 + 1\}. \tag{86}$$

Let $\Delta_2(d) = |\{\mathbf{w} : \mathbf{w} \in \Theta_1(i, \gamma, r_1) \text{ and } |\mathbf{w}| = d\}|$. By direct counting, we can prove that

$$\Delta_2(d) = \binom{N - \Gamma_1 - 1}{d - 1} \tag{87}$$

Note that $\Delta_2(d)$ is independent of the choice of i and γ. Then, the collection of constraint (C8) is reduced to one single constraint

$$\sum_{d=2}^{r_1+1} \Delta_2(d) Y_{d,0,0} \geq (q-1) \sum_{d,e,f} \binom{N}{d,e,f} X_{d,e,f}. \tag{88}$$

The reduction of constraint (C9) is very similar to that of (C8). Again, consider any given $(i, \gamma) \in \Omega(\Gamma_2)$. Recall that

$$\Theta_2(i, \gamma, r_2, a) \triangleq \{(\mathbf{w}, \mathbf{s}) : i \in \mathbf{w}, \mathbf{w} \cap \gamma = \emptyset, |\mathbf{w}| \leq r_2 + 1 \text{ and } |\mathbf{s}| \leq a\}. \tag{89}$$

Let

$$\Pi(d, e, f) = \left\{ (\mathbf{w}, \mathbf{s}) : \begin{array}{c} (\mathbf{w}, \mathbf{s}) \in \Theta_2(i, \gamma, r_2, a) \\ \text{and } \Upsilon(\mathbf{w}, \mathbf{s}) = (d, e, f) \end{array} \right\},$$

where $\Upsilon(\mathbf{w}, \mathbf{s}) = (|\mathbf{w} \setminus \mathbf{s}|, |\mathbf{w} \cap \mathbf{s}|, |\mathbf{s} \setminus \mathbf{w}|)$. Hence, the constraint (C9) can be rewritten as

$$\sum_{d+e \leq r_2+1, e+f \leq a} \Delta_3(d, e, f) Y_{d,e,f} \geq T(q-1) \sum_{d,e,f} \binom{N}{d,e,f} X_{d,e,f}, \tag{90}$$

where $\Delta_3(d, e, f) = |\Pi(d, e, f)|$.

Finally, it remains to determine $\Delta_3(d, e, f)$. Partition $\Pi(d, e, f)$ into two sets

$$\Pi^{(1)}(d, e, f) \triangleq \{(\mathbf{w}, \mathbf{s}) \in \Pi(d, e, f) \text{ and } i \in \mathbf{s}\}$$

and

$$\Pi^{(2)}(d, e, f) \triangleq \{(\mathbf{w}, \mathbf{s}) \in \Pi(d, e, f) \text{ and } i \notin \mathbf{s}\}$$

Hence,

$$\Delta_3(d, e, f) = |\Pi^{(1)}(d, e, f)| + |\Pi^{(2)}(d, e, f)| \tag{91}$$

where, by direct counting,

$$|\Pi^{(1)}(d, e, f)| = \binom{N - \Gamma_2 - 1}{e - 1} \binom{N - \Gamma_2 - e - 1}{d} \binom{N - e - d}{f}, \tag{92}$$

$$|\Pi^{(2)}(d, e, f)| = \binom{N - \Gamma_2 - 1}{e} \binom{N - \Gamma_2 - e - 1}{d - 1} \binom{N - e - d}{f}. \tag{93}$$

Finally, we will consider the reduction of (C3). The approach is similar, despite that the derivation is much more tedious. First, we fix a given $(\mathbf{w}, \mathbf{s})$. Consider the term

$$A_{\mathbf{w}',\mathbf{s}'} \prod_{j \in \mathcal{N}} \kappa_q(w'_j, w_j) \kappa_q(s'_j, s_j)$$

in (C3). Let

$$\Xi_1(\mathbf{w}, \mathbf{s}) = \{i \in \mathcal{N} : i \in \mathbf{w} \setminus \mathbf{s}\} \tag{94}$$

$$\Xi_2(\mathbf{w}, \mathbf{s}) = \{i \in \mathcal{N} : i \in \mathbf{w} \cap \mathbf{s}\} \tag{95}$$

$$\Xi_3(\mathbf{w}, \mathbf{s}) = \{i \in \mathcal{N} : i \in \mathbf{s} \setminus \mathbf{w}\} \tag{96}$$

$$\Xi_4(\mathbf{w}, \mathbf{s}) = \{i \in \mathcal{N} : i \in \mathcal{N} \setminus (\mathbf{w} \cup \mathbf{s})\}. \tag{97}$$

and $\xi_{i,j}(\mathbf{w}, \mathbf{s}, \mathbf{w}', \mathbf{s}') = |\Xi_i(\mathbf{w}, \mathbf{s}) \cap \Xi_j(\mathbf{w}', \mathbf{s}')|$.

Suppose $\xi_{i,j}(\mathbf{w}, \mathbf{s}, \mathbf{w}', \mathbf{s}') = \zeta_{i,j}$ for all $1 \le i, j \le 4$. Let $\zeta = (\zeta_{i,j},\ 1 \le i, j \le 4)$. Then

1) $\sigma_1(\zeta) = |\mathbf{w} \setminus \mathbf{w}'| + |\mathbf{s} \setminus \mathbf{s}'|$ where

$$\sigma_1(\zeta) \triangleq \zeta_{1,3} + \zeta_{1,4} + \zeta_{2,3} + \zeta_{2,4} + \zeta_{2,1} + \zeta_{2,4} + \zeta_{3,1} + \zeta_{3,4}. \tag{98}$$

Similarly, $\sigma_2(\zeta) = |\mathbf{w} \cap \mathbf{w}'| + |\mathbf{s} \cap \mathbf{s}'|$ where

$$\sigma_2(\zeta) \triangleq \zeta_{1,1} + \zeta_{1,2} + \zeta_{2,1} + \zeta_{2,2} + \zeta_{2,2} + \zeta_{2,3} + \zeta_{3,2} + \zeta_{3,3}. \tag{99}$$

2) $\prod_{j \in \mathcal{N}} \kappa_q(w'_j, w_j)\kappa_q(s'_j, s_j) = (q-1)^{\sigma_1(\zeta)}(-1)^{\sigma_2(\zeta)}$.

3) $A_{\mathbf{w}',\mathbf{s}'} = X_{d',e',f'}$ where

$$d' = \zeta_{1,1} + \zeta_{2,1} + \zeta_{3,1} + \zeta_{4,1} \tag{100}$$

$$e' = \zeta_{1,2} + \zeta_{2,2} + \zeta_{3,2} + \zeta_{4,2} \tag{101}$$

$$f' = \zeta_{1,3} + \zeta_{2,3} + \zeta_{3,3} + \zeta_{4,3} \tag{102}$$

4) Fix $\mathbf{w}, \mathbf{s}$. Let

$$U(\zeta) = \left|\{(\mathbf{w}', \mathbf{s}') :\ \xi_{i,j}(\mathbf{w}, \mathbf{s}, \mathbf{w}', \mathbf{s}') = \zeta_{i,j},\ 1 \le i, j \le 4\}\right|.$$

Then

$$U(\zeta) = \binom{\sum_{j=1}^4 \zeta_{1j}}{\zeta_{1j},\ j = 1,2,3,4}\binom{\sum_{j=1}^4 \zeta_{2j}}{\zeta_{2j},\ j = 1,2,3,4} \times \binom{\sum_{j=1}^4 \zeta_{3j}}{\zeta_{3j},\ j = 1,2,3,4}\binom{\sum_{j=1}^4 \zeta_{4j}}{\zeta_{4j},\ j = 1,2,3,4}. \tag{103}$$

5) For any (d, e, f, d', e', f'), let

$$\Delta_1(d, e, f, d', e', f') = \sum_{\zeta:\zeta^{(1)}=(d,e,f),\zeta^{(2)}=(d',e',f')} U(\zeta)(q-1)^{\sigma_1(\zeta)}(-1)^{\sigma_2(\zeta)}, \tag{104}$$

where

$$\zeta^{(1)} = \left(\sum_j \zeta_{1,j}, \sum_j \zeta_{2,j}, \sum_j \zeta_{3,j}\right)$$

$$\zeta^{(2)} = \left(\sum_j \zeta_{j,1}, \sum_j \zeta_{j,2}, \sum_j \zeta_{j,3}\right).$$

Grouping all the like terms, the constraint (C3) can be rewritten as in (D3).

REFERENCES

[1] M. A. Tebbi, T. H. Chan, and C. W. Sung, "A code design framework for multi-rack distributed storage," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2014, pp. 55–59.

[2] 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.

[3] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Distributed storage codes with repair-by-transfer and nonachievability of interior points on the storage-bandwidth tradeoff," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1837–1852, Mar. 2012.

[4] 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.

[5] G. M. Kamath, N. Prakash, V. Lalitha, and P. V. Kumar, "Codes with local regeneration and erasure correction," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 4637–4660, Aug. 2014.

[6] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Cambridge, MA, USA, Jul. 2012, pp. 2771–2775.

[7] T. C. Jepsen, *Distributed Storage Networks: Architecture, Protocols and Management*. Hoboken, NJ, USA: Wiley, 2003.

[8] G. Ananthanarayanan, S. S. A. Ghodsi, and I. Stoica, "Disk-locality in datacenter computing considered irrelevant," in *Proc. USENIX HotOS*, 2011, p. 12.

[9] F. Ahmad, S. T. Chakradhar, A. Raghunathan, and T. N. Vijaykumar, "Shufflewatcher: Shuffle-aware scheduling in multi-tenant mapreduce clusters," in *Proc. USENIX ATC USENIX Annu. Tech. Conf.*, Philadelphia, PA, USA, Jun. 2014, pp. 1–13.

[10] C. Huang *et al.*, "Erasure coding in windows azure storage," in *Proc. USENIX ACT*, 2012, pp. 15–26.

[11] M. Sathiamoorthy *et al.*, "Xoring elephants: Novel erasure codes for big data," *Proc. VLDB Endowment*, vol. 6, no. 5, pp. 325–336, 2013.

[12] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. ACM IMC*, 2010, pp. 267–280.

[13] D. S. Papailiopoulos, J. Luo, A. G. Dimakis, C. Huang, and J. Li, "Simple regenerating codes: Network coding for cloud storage," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2801–2805.

[14] 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 Annu. Allerton Conf. Commun., Control, Comput. Urbana-Champaign,*, Sep. 2009, pp. 1243–1249.

[15] 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, Aug. 2012.

[16] Q. Yu, C. W. Sung, and T. H. Chan, "Locally repairable codes over a network," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2014, pp. 70–74.

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

[18] M. A. Tebbi, T. H. Chan, and C. W. Sung, "Linear programming bounds for robust locally repairable storage codes," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2014, pp. 50–54.

[19] T. H. Chan, M. A. Tebbi, and C. W. Sung, "Linear programming bounds for storage codes," in *Proc. 9th Int. Conf. Inf., Commun., Signal Process. (ICICS)*, Dec. 2013, pp. 1–5.

[20] A. S. Rawat, A. Mazumdar, and S. Vishwanath, "On cooperative local repair in distributed storage," in *Proc. 48th Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2014, pp. 1–5.

[21] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," Microsoft Res., Redmond, WA, USA, Tech. Rep. MSR-TR-2007-25, Mar. 2007.

[22] Cisco Validated Design I, *Cisco Data Center Infrastructure 2.5 Design Guide*. San Jose, CA, USA: Cisco Systems, 2007.

[23] Y. Hu, P. P. C. Lee, and X. Zhang, "Double regenerating codes for hierarchical data centers," in *Proc. IEEE Int. Symp. Inf. Theory*, Barcelona, Spain, Jul. 2016, pp. 245–249.

[24] B. Gaston, J. Pujol, and M. Villanueva, "A realistic distributed storage system: The rack model," Feb. 2013. *arXiv:1302.5657*. [Online]. Available: https://arxiv.org/abs/1302.5657

[25] B. Gastón, J. Pujol, and M. Villanueva, "A realistic distributed storage system that minimizes data storage and repair bandwidth," in *Proc. Data Compress. Conf.*, Mar. 2013, p. 491.

[26] J. Pernas, C. Yuen, B. Gastón, and J. Pujol, "Non-homogeneous two-rack model for distributed storage systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2013, pp. 1237–1241.

[27] V. T. Van, C. Yuen, and J. Li, "Non-homogeneous distributed storage systems," in *Proc. 50th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Oct. 2012, pp. 1133–1140.

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

[29] J. Kubiatowicz *et al.*, "Oceanstore: An architecture for global-scale persistent storage," in *Proc. 9th Int. Conf. Architectural Support Programm. Lang. Oper. Syst.*, Boston, MA, USA, Nov. 2000, pp. 190–201.

[30] H. Zhang, M. Chen, A. Parek, and K. Ramchandran, "A distributed multichannel demand-adaptive P2P VoD system with optimized caching and neighbor-selection," *Proc. SPIE*, vol. 8135, Sep. 2011.

[31] N. Golrezaei, A. G. Dimakis, and A. F. Molisch, "Wireless device-to-device communications with distributed caching," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2012, pp. 2781–2785.
[32] D. S. Papailiopoulos, A. G. Dimakis, and V. R. Cadambe, "Repair optimal erasure codes through Hadamard designs," in *Proc. 49th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2011, pp. 1382–1389.
[33] D. Leong, A. G. Dimakis, and T. Ho, "Distributed storage allocations," *IEEE Trans. Inf. Theory*, vol. 58, no. 7, pp. 4733–4752, Jul. 2012.
[34] Q. Yu, K. W. Shum, and C. W. Sung, "Minimization of storage cost in distributed storage systems with repair consideration," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Houston, TX, USA, Dec. 2011, pp. 1–5.
[35] Q. Yu, K. W. Shum, and C. W. Sung, "Tradeoff between storage cost and repair cost in heterogeneous distributed storage systems," *Trans. Emerg. Telecommun. Technol.*, vol. 26, no. 10, pp. 1201–1211, Oct. 2015.
[36] T. Ernvall, S. El Rouayheb, C. Hollanti, and H. V. Poor, "Capacity and security of heterogeneous distributed storage systems," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 2701–2709, Dec. 2013.
[37] G. Calis and O. O. Koyluoglu, "Architecture-aware coding for distributed storage: Repairable block failure resilient codes," Feb. 2017, *arXiv:1605.04989*. [Online]. Available: https://arxiv.org/abs/1605.04989
[38] D. Ford *et al.*, "Availability in globally distributed storage systems," in *Proc. 9th USENIX Symp. Operating Syst. Design Implement.*, Vancouver, BC, Canada, Oct. 2010, pp. 1–14.
[39] N. Prakash, V. Abdrashitov, and M. Médard, "The storage versus repair-bandwidth trade-off for clustered storage systems," *IEEE Trans. Inf. Theory*, vol. 64, no. 8, pp. 5783–5805, Aug. 2018.
[40] J. yong Sohn, B. Choi, S. W. Yoon, and J. Moon, "Capacity of clustered distributed storage," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, France, May 2017, pp. 1–7.
[41] S. Sahraei and M. Gastpar, "Increasing availability in distributed storage systems via clustering," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 1705–1709.
[42] B. Sasidharan, G. K. Agarwal, and P. V. Kumar, "Codes with hierarchical locality," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2015, pp. 1257–1261.
[43] P. Gopalan, G. Hu, S. Kopparty, S. Saraf, C. Wang, and S. Yekhanin, "Maximally recoverable codes for grid-like topologies," in *Proc. 28th Annu. ACM-SIAM Symp. Discrete Algorithms*, Barcelona, Spain, Jan. 2017, pp. 2092–2108.
[44] F. J. Macwilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North Holland, 1977.

Ali Tebbi received his B.Sc. degree in Electrical Engineering from University of Tabriz, Tabriz, Iran in 2007, his M.Sc. degree in Telecommunication Systems from K. N. Toosi University of Technology, Tehran, Iran in 2011, and his Ph.D. degree in Telecommunications from University of South Australia, Australia in 2016. In 2017 he was a post-doctoral fellow at the City University of Hong Kong and a visiting researcher at the Institute of Network Coding, The Chinese University of Hong Kong. In 2018 he was a research associate at the Institute for Telecommunications Research, University of South Australia. He is currently with the School of Information Technology and Mathematical Sciences, University of South Australia. His main research interests include distributed storage, network information Theory, coding theory, distributed computing, and coded cache networks.

Terence H. Chan received his B.Sc (Math), Master's and Ph.D. degrees in Information Engineering in 1996, 1998 and 2000 respectively, all from The Chinese University of Hong Kong. In 2001, he was a visiting assistant professor in the Department of Information Engineering at the same university. From February 2002 to June 2004, he was a Post-doctoral Fellow at the Department of Electrical and Computer Engineering at the University of Toronto. He was an assistant professor in University of Regina from 2004–2006. He is currently an Associate Professor in Institute for Telecommunications Research at University of South Australia.

Chi Wan Sung (M'98–SM'16) was born in Hong Kong. He received the B.Eng., M.Phil., and Ph.D. degrees in information engineering from The Chinese University of Hong Kong in 1993, 1995, and 1998, respectively. After graduation, he was an Assistant Professor at The Chinese University of Hong Kong. He joined the faculty at the City University of Hong Kong in 2000, and is now an Associate Professor with the Department of Electronic Engineering. His research interest is on coding, communications, and networking, with emphasis on algorithm design and complexity analysis. He was an Associate Editor of the Transactions on Emerging Telecommunications Technologies from 2013 to 2016, and is currently on the editorial boards of the *ETRI Journal* and *Electronics Letters*.