A Flexible Class of Regenerating Codes for Distributed Storage

Nihar B. Shah, K. V. Rashmi, P. Vijay Kumar

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

***Abstract*—In the distributed storage setting introduced by Dimakis et al., B units of data are stored across n nodes in the network in such a way that the data can be recovered by connecting to any k nodes. Additionally one can repair a failed node by connecting to any d nodes while downloading at most β units of data from each node.**

In this paper, we introduce a flexible framework in which the data can be recovered by connecting to any number of nodes as long as the total amount of data downloaded is at least B. Similarly, regeneration of a failed node is possible if the new node connects to the network using links whose individual capacity is bounded above by $\beta_{\max}$ and whose sum capacity equals or exceeds a predetermined parameter γ. In this flexible setting, we obtain the cut-set lower bound on the repair bandwidth along with a constructive proof for the existence of codes meeting this bound for all values of the parameters. An explicit code construction is provided which is optimal in certain parameter regimes.

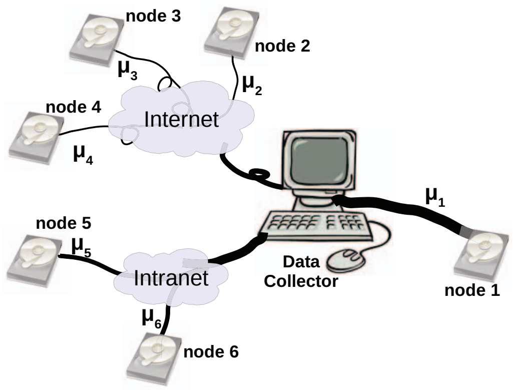

Fig. 1. An example of flexible regeneration where the DC connects to different storage nodes via different types of networks, and hence would like to download the amount of data from a node that is proportional to its link speed μ_i to that node.

I. Introduction

Erasure codes like Reed Solomon (RS) codes can be used in distributed storage systems to provide resilience towards individual node failures while minimizing the storage overhead. Let the total data to be stored be of size B symbols over a finite field $\mathbb{F}_q$ of size q. Let n be the total number of nodes in the system each having a capacity to store α symbols. It is required that an end-user i.e a data collector (DC) be able to recover all the data by connecting to a subset of the nodes. This operation is termed *reconstruction*. Upon failure of a node, a self-sustaining data storage system must posses the ability to repair the failed node by downloading data from other existing nodes. This operation is termed *regeneration* and the amount of data download is termed *repair bandwidth*. RS codes require the entire data to be downloaded to repair a failed node. Since each node stores only a part of the total data, downloading B symbols to repair a failed node is wasteful and raises the question as to whether there is a better option. Such an option is provided by *regenerating codes* introduced by Dimakis et al. [1], [2]. The authors consider a setting where a DC can connect to any k nodes and download the α data symbols stored in each of them for reconstruction. On failure of a node, the new node connects to any d nodes and downloads exactly β symbols from each of them. The authors move on to obtain a cut-set based lower bound for the repair bandwidth. The work in this paper draws inspiration from the tools used in [1], [2]. Other works in the past, including [5]–[7], [9] obtain schemes and explicit codes for such a system setup.

In a practical scenario, the storage nodes may be spread out geographically, say over the internet, and may have routes of different capacities between them. The assumption in the original setup, of the DC connecting to only k nodes and a new node connecting to only d nodes with downloading β symbols from each is very restrictive in nature. It is desirable to enable the DC or the new node to make use of parallel downloads according to the availability of other storage nodes in its vicinity and this can greatly reduce the total download time [3], [4]. Also, the links from the DC or the new node to other nodes may not be symmetric in general; some links may have higher capacities (or lower RTTs) than the other links at a given instant depending on the topology of the network and the prevailing traffic in different parts of the network. This is illustrated in Figure 1. Hence, freedom to download different amounts of data from the different nodes helps in reducing the net download time and traffic congestion. Such a system will also be highly conducive for load-balancing across the nodes in the network.

In this paper, we introduce a framework for flexible distributed storage systems and obtain a lower bound on the repair bandwidth for such systems. We call codes achieving this lower bound as *Flexible Regenerating Codes*. We also provide an explicit code construction of flexible regenerating codes for certain parameter regimes.

The rest of the paper is organized a follows. A formal description of flexible regenerating codes is given in Section II. A lower bound on the total amount of download required to regenerate a failed node is derived in Section III. A constructive proof for existence of linear codes which achieve the lower bound for all values of the system parameters is provided in Section IV. In Section V explicit code construction is described with the help of an example.

II. Flexible Regenerating Codes

In this section, we introduce a flexible framework for distributed storage systems. Data is stored in a distributed manner across n storage nodes, each having a capacity to store α symbols. A DC downloads $\mu_1, \ldots, \mu_n$ symbols from nodes $1, \ldots, n$ respectively. The DC should be able to recover back the entire data for any choice of $\mu_i, \;\; i = 1, \ldots, n$ satisfying

$$\sum_{i=1}^{n} \mu_i \geq B, \qquad 0 \leq \mu_i \leq \alpha \;. \tag{1}$$

We call this *Flexible Reconstruction*.

Compared to the original setup of regenerating codes, where the DC is restricted to connect to k nodes, this framework provides a great deal of flexibility to the DC to choose the link capacities with which it wants to connect to each of the nodes. This choice could be based on the network conditions at that instant, and the DC can even possibly connect to all the n nodes.

When a storage node ℓ fails, it is replaced by a new node which downloads β_i symbols from node $i, \quad \forall \; i = 1, \ldots, n, \quad i \neq \ell$ as long as

$$\sum_{i=1 \; (i \neq \ell)}^{n} \beta_i \geq \gamma, \qquad 0 \leq \beta_i \leq \beta_{\max} \tag{2}$$

for some value γ, the repair bandwidth. Here $\beta_{\max}$ is a constant satisfying

$$0 \leq \beta_{\max} \leq \alpha \;. \tag{3}$$

The new node along with the existing nodes should satisfy the flexible reconstruction property and should be able to participate in the regeneration of any other failed node in the future.

The parameter $\beta_{\max}$ puts a cap on the maximum amount of data that the new node can download from an existing node. The most general setting would be to allow the new node to download any amount of data from the existing nodes, i.e., choosing $\beta_{max} = \alpha$. However, as it will be shown in Section III-C, this is not a wise choice and it results in new node having to download the entire file.

Again, in this flexible framework, the new node has the freedom to choose the link capacities to each node. Unlike in the original regenerating code setup, the new node is not constrained to download equal amounts of data from each node it connects to, and can download non-uniformly depending on the prevailing network conditions. We term this *Flexible Regeneration*.

Any code satisfying the flexible reconstruction and flexible regeneration properties is called a *Flexible Regenerating Code*.

Note that for any storage system to be feasible, we need $\alpha \geq \frac{B}{n}$. We assume throughout that this condition is satisfied. We also assume that all system parameters are non-negative integers.

III. Lower bound on the Repair Bandwidth

A. Information Flow Network

In this section we provide a lower bound on the repair bandwidth required to maintain a flexible distributed storage system. As in [1], we model the distributed storage system as an *information flow network*. In such a network, each storage node is modeled in the form of two nodes - an *in* node and an *out* node and a link of capacity α connecting the two. This captures the constraint that each node can store only α symbols. Figure 2 gives an example of such a network where S is the source producing data at the rate of B symbols per unit time (the data file). The source connects to the n nodes with links having capacities of α symbols each.

On failure of a storage node, say node ℓ, it is replaced by a new node by connecting nodes out(j), $j \in \{1, \ldots, n\}, \; j \neq \ell$, to in($\ell$), with links of capacities β_j, satisfying equation (2). Thus the network evolves through an infinite chain of failures and regenerations. For every instantiation of the network, there can be a different sequence of failures and regenerations with different sets of $\{\beta_j\}$ for each regeneration, and all these instantiations have to be satisfied by a flexible regenerating code.

For reconstruction, at any stage of the network evolution, a DC (sink) can connect to the n existing nodes. This is represented by links of capacities μ_j from the *out* nodes $j \; (= 1, \ldots, n)$ to the sink, satisfying equation (1). Each DC can connect to the storage nodes with a different set of $\{\mu_j\}$.

A lower bound on the repair bandwidth is obtained by bounding the maximum flow in this network.

B. Set of Cuts

We now demonstrate a set of cuts which give an upper bound on the flow B, and equivalently a lower bound on the total repair bandwidth γ. Any cut C partitions the set of nodes V in the network into V_C and V_C^c where $S \in V_C$ and $DC \in V_C^c$.

Consider the set of cuts where V_C^c contains only the DC along with the *out* parts of some r of the n existing storage nodes. Since the network considered is a directed acyclic graph, nodes in the network can be topologically ordered, as illustrated in Figure 2.

Consider the first storage node in V_C^c in the topological ordering. Call it node Λ_1. Since out(Λ_1) $\in V_C^c$, the cut crosses either the α capacity link between in(Λ_1) and out(Λ_1) or the set of links with total capacity γ entering in(Λ_1). We take the cut across the minimum of the two contributing $\min{(\alpha, \gamma)}$ to the value of the cut.

Consider the next storage node in V_C^c in the topological ordering and call it node Λ_2. To decrease the value of the cut, we assume that there is a link of value $\beta_{\max}$ from out(Λ_1) to in(Λ_2) which will not be a part of the cut. Again, the cut will cross either the α capacity link between in(Λ_2) and out(Λ_2) or the set of links with total capacity $(\gamma - \beta_{\max})^+$ entering in(Λ_2) from nodes in V_C. Again, we take the cut across the minimum of the two contributing $\min{(\alpha, (\gamma - \beta_{\max})^+)}$ to the value of the cut.

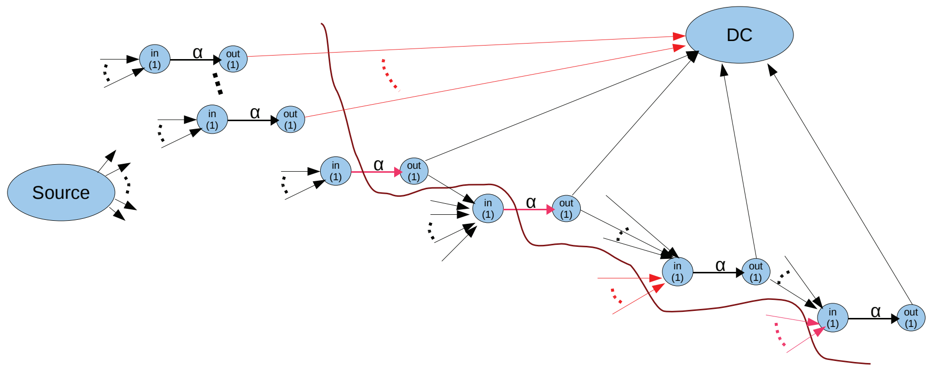

Fig. 2. An information flow network. The cut separates the source from a particular DC. The edges marked in red are the edges crossing the cut.

In general, node Λ_j, $1 \le j \le r$ in V_C^c has links with capacities $\beta_{\max}$ each from out(Λ_i), $i = 1, \ldots, j-1$ which will not be a part of the cut. Hence we take the cut across α or $(\gamma - (j-1)\beta_{\max})^+$, whichever is less. The DC connects to these r nodes and downloads α symbols each. The rest $(B - r\alpha)^+$ needs to come from nodes in V_C, and hence will be a part of the cut. Thus the value of the cut is

$$\sum_{j=0}^{r-1} \min\left(\alpha, (\gamma - j\beta_{\max})^+\right) + (B - r\alpha)^+ \tag{4}$$

The file size B has to be smaller than any cut and hence

$$B \le \min_{0 \le r \le n} \left\{ \sum_{j=0}^{r-1} \min\left(\alpha, (\gamma - j\beta_{\max})^+\right) + (B - r\alpha)^+ \right\} \tag{5}$$

Lemma 1: A cut-set lower bound on the repair bandwidth γ is given by

$$\gamma \ge \max(\alpha - \beta_{\max}, B \bmod \alpha) + s\beta_{\max} \tag{6}$$

Proof:
Define $s = \lfloor B/\alpha \rfloor$. (7)

For $r = s$, the equation (5) reduces to

$$\begin{aligned} B &\le \sum_{j=0}^{s-1} \min\left(\alpha, (\gamma - j\beta_{\max})^+\right) + (B - s\alpha) && (8)\\ s\alpha &\le \sum_{j=0}^{s-1} \min\left(\alpha, (\gamma - j\beta_{\max})^+\right) && (9)\\ &\le s\alpha && (10) \end{aligned}$$

Thus, $(\gamma - j\beta_{\max})^+ \ge \alpha \quad \forall\, j \in \{0, \ldots, s-1\}$ (11)

which gives a lower bound on the repair bandwidth as

$$\gamma \ge \alpha + (s-1)\beta_{\max} \tag{12}$$

For $r = s + 1$, equation (5) gives

$$\begin{aligned} B &\le \sum_{j=0}^{s} \min\left(\alpha, (\gamma - j\beta_{\max})^+\right) && (13)\\ &= s\alpha + \min\left(\alpha, (\gamma - s\beta_{\max})^+\right) && (14) \end{aligned}$$

where (14) is due to (12). This evaluates to

$$\gamma \ge B - s\alpha + s\beta_{\max} \tag{15}$$

Combining (12) and (15) we get

$$\gamma \ge \max(\alpha - \beta_{\max}, B \bmod \alpha) + s\beta_{\max}. \tag{16}$$

■

C. *Complete Flexibility ($\beta_{\max} = \alpha$) ?*

An obvious question in the flexible framework is, how much can we optimize the repair bandwidth if we give complete freedom to the new node replacing a failed node, i.e. allowing $\beta_{\max} = \alpha$. The answer to this is obtained by substituting $\beta_{\max} = \alpha$ in equation (16). This gives $\gamma \ge B$, i.e., the repair bandwidth is equal to the size of the entire file.

IV. Achievability

In this section, we prove the existence of a linear flexible regenerating code which meets the lower bound on the repair bandwidth given by Lemma 1. We prove the existence of a linear code where any DC connecting to node i with a link of capacity μ_i, $\forall i = 1, \ldots, n$ with

$$\sum_{i=1}^{n} \mu_i = B, \qquad 0 \le \mu_i \le \alpha \tag{17}$$

can recover the data, and any failed node ℓ can be regenerated by downloading β_i symbols from node i $(i = 1, \ldots, n, i \ne \ell)$ with

$$\sum_{i=1\ (i \ne \ell)}^{n} \beta_i = \gamma, \quad 0 \le \beta_i \le \beta_{\max} \tag{18}$$

where γ meets the lower bound on the repair bandwidth given by Lemma 1. Then, it is clear that any DC with $\sum_{i=1}^{n} \mu_i > B$ can recover the data, and any failed node with $\sum_{i=1\ (i \ne \ell)}^{n} \beta_i > \gamma$ can be regenerated.

Define a vector $\underline{f}$ of length B, consisting of the source symbols. Each source symbol can independently take values from $\mathbb{F}_q$, a finite field of size q. Any stored symbol is written as $\underline{u}^t \underline{f}$ for some B-length vector $\underline{u}$ which corresponds to the global kernel of this stored symbol. These global kernels for the stored symbols define the code, and the actual symbols stored depend on the instantiation of $\underline{f}$. Since a node stores α symbols, it can be considered as storing α vectors of the code, and hence can be represented by a $\alpha \times B$ matrix. We will say that the node *stores* this matrix.

Lemma 2: For any set of $\mu_i,\ i=1,\dots,n$ and $\beta_i,\ i=2,\dots,n$ satisfying the conditions in equations (17) and (18) with $\ell=1$,

$$\sum_{i=2}^{n}\min(\alpha,\mu_i+\beta_i)\geq B \tag{19}$$

Proof (Sketch): It can be shown that the given term attains its minimum when $\mu_i (i=1,\dots,n)$ and $\beta_i (i=2,\dots,n)$ have the most skewed distribution, i.e. $\mu_i=\alpha$ for $i=1,\dots,s$, $\mu_{s+1}=B \mod \alpha$ and $\mu_i=0$ elsewhere; $\beta_i=\beta_{\max}$ for $i=2,\dots,\left\lfloor\frac{\gamma}{\beta_{\max}}\right\rfloor$, $\beta_i=\gamma \mod \beta_{\max}$ for $i=\left\lfloor\frac{\gamma}{\beta_{\max}}\right\rfloor+1$, and zeros elsewhere. For these values of μ_i and β_i, using the conditions given by equations (17) and (18), it can be shown that, this term is lower bounded by B. ∎

The following Lemmas show that given a system which can achieve flexible reconstruction at a particular stage, then upon failure of a node, it can be regenerated such that the system retains the flexible reconstruction property.

Lemma 3: Suppose flexible reconstruction is satisfied for all DCs in the present stage. Suppose node ℓ fails and is replaced by a new node. Given a particular DC, in the next stage, i.e. after ℓ is regenerated, connecting to node i with a link of capacity $\mu_i,\ \forall i=1,\dots,n$, satisfying constraints given in equation (17), the new node can download β_i symbols from node i $(i=1,\dots,n,\ \ i\neq\ell)$ satisfying the constraints given in (18) and store α symbols such that this DC is satisfied.

Proof: The main idea is to show that the given DC (after regeneration of node ℓ) is equivalent to some DC connecting to the nodes in the present stage (before failure of node ℓ), satisfying (17). As flexible reconstruction is satisfied for all DCs in the present stage, this would imply that the given DC will also be satisfied. Without loss of generality assume that the first node fails and is regenerated i.e. $\ell=1$.

Since μ_i and β_i satisfy the conditions of Lemma 2,

$$\sum_{i=2}^{n}\min(\alpha,\mu_i+\beta_i)\geq B\ . \tag{20}$$

Now, reduce the values of β_i to $\beta_i',\ \forall i=2,\dots,n$ such that equality is attained above, i.e.

$$\sum_{i=2}^{n}\min(\alpha,\mu_i+\beta_i')=B \tag{21}$$

Thus we have β_i' point-wise lesser than β_i

$$0\leq\beta_i'\leq\beta_i\quad\forall i=2,\dots,n \tag{22}$$

Consider a virtual DC in the present stage connecting to node i with links of capacity $\tilde{\mu}_i,\ \forall i=2,\dots,n$ given by

$$\tilde{\mu}_i = \begin{cases} 0 & i=1\\ \min(\alpha,\mu_i+\beta_i') & i=2,\dots,n\end{cases} \tag{23}$$

This is a valid DC since $0\leq\tilde{\mu}_i\leq\alpha\quad\forall i$ and

$$\sum_{i=1}^{n}\tilde{\mu}_i=\sum_{i=2}^{n}\min(\alpha,\mu_i+\beta_i')=B \tag{24}$$

Consider each node passing μ_i out of the $\tilde{\mu}_i$ symbols directly to the DC and the remaining $(\tilde{\mu}_i-\mu_i)$ symbols via the new node. In the real scenario, this means that the new node downloads the $(\tilde{\mu}_i-\mu_i)$ symbols which are flowing through the new node in the virtual case, from each existing node i. This is a valid regeneration process since for all $i=2,\dots,n$,

$$\begin{aligned}(\tilde{\mu}_i-\mu_i) &= \min(\alpha,\mu_i+\beta_i')-\mu_i && (25)\\ &\leq \beta_i' && (26)\\ &\leq \beta_i && (27)\end{aligned}$$

Also, the number of symbols downloaded by the virtual DC through the new node is

$$\begin{aligned}\sum_{i=2}^{n}(\tilde{\mu}_i-\mu_i) &= \sum_{i=2}^{n}\tilde{\mu}_i-\sum_{i=2}^{n}\mu_i && (28)\\ &= B-(B-\mu_1) && (29)\\ &= \mu_1 && (30)\end{aligned}$$

Thus the given DC becomes equivalent to the virtual DC. Since flexible reconstruction is satisfied for any DC in the present stage, the virtual DC can recover all the data. This implies that the given DC can also recover all the data. ∎

Lemma 4: Suppose flexible reconstruction is satisfied for all DCs at the present stage. Suppose node ℓ fails and is replaced by a new node. The new node can download β_i symbols from node i $(i=1,\dots,n,\ \ i\neq\ell)$ satisfying the constraints given in (18) and store α symbols such that all DCs satisfying (17) are simultaneously satisfied, provided the field size is large enough.

Proof: Without loss of generality assume $\ell=1$. Let $\mathbf{G}^{(1)},\cdots,\mathbf{G}^{(n)}$ be the node matrices at the present stage where flexible reconstruction is satisfied for all DCs. Let $\tilde{\mathbf{G}}^{(1)}$ be the matrix stored in the new node replacing node 1.

The new node downloads β_i symbols from node i $(i=2,\dots,n)$ and stores α linear combinations of the symbols downloaded. Thus,

$$\tilde{\mathbf{G}}^{(1)}=\mathbf{Z}\begin{bmatrix}\mathbf{V}^{(2)}\mathbf{G}^{(2)}\\ \mathbf{V}^{(3)}\mathbf{G}^{(3)}\\ \vdots\\ \mathbf{V}^{(n)}\mathbf{G}^{(n)}\end{bmatrix} \tag{31}$$

where $\mathbf{V}^{(i)}$ is $\beta_i\times\alpha$ matrix representing the linear combinations used by node i to compute the β_i symbols that it passes to the new node. $\mathbf{Z}$ is $\alpha\times\gamma$ matrix representing the linear transformation that the new node performs on the downloaded symbols to compute the α symbols that it stores.

Consider a DC Δ connecting to the nodes (after regeneration of node 1) with link capacities satisfying equation (17). Every node i uses a $\mu_i\times B$ matrix $\mathbf{U}_\Delta^{(i)}$ to compute the linear combinations to be passed to this DC. Thus, for the DC to be able to recover the data, we need

$$\mathcal{P}_\Delta=\det\begin{bmatrix}\mathbf{U}_\Delta^{(1)}\tilde{\mathbf{G}}^{(1)}\\ \mathbf{U}_\Delta^{(2)}\mathbf{G}^{(2)}\\ \vdots\\ \mathbf{U}_\Delta^{(n)}\mathbf{G}^{(n)}\end{bmatrix}\neq 0 \tag{32}$$

The above determinant can also be viewed as a polynomial $\mathcal{P}_\Delta$ in $\mathbb{F}_q$ with entries of the matrices $\mathbf{U}_\Delta^{(i)}\ \ (i=1,\dots,n)$, $\mathbf{V}^{(i)}(i=2,\dots,n)$ and $\mathbf{Z}$ as variables. By Lemma 3 we know that the DC Δ can be satisfied, i.e. there exist values of the variables such that the above determinant is non-zero. Thus the polynomial in (32) is a non-zero polynomial.

For all DCs to be satisfied simultaneously, we need

$$\prod_{\Delta:\text{over all DCs}} \mathcal{P}_\Delta \neq 0 \tag{33}$$

This product is itself a polynomial with variables being entries of the matrices $\mathbf{U}_\Delta^{(i)}$ $(i = 1, \ldots, n,$ all DCs $\Delta)$, $\mathbf{V}^{(i)}$ $(i = 2, \ldots, n)$ and $\mathbf{Z}$. Since each polynomial in this product is non-zero, the product polynomial is also non-zero. Hence, the Schwartz-Zippel Lemma implies that there is an assignment to variables such that equation (33) is satisfied, provided the field size is large enough. ■

Theorem 5 (Existence of Flexible Regenerating Codes): Given any set of system parameters $(n,\ B,\ \alpha,\ \beta_{max})$, there exists a linear flexible regenerating code satisfying the lower bound on the repair bandwidth γ provided that the size of the finite field is large enough.

Proof: The proof is by induction. Initialize the $n\alpha$ symbols in the nodes with an $[n\alpha, B]$-MDS code. This clearly satisfies the flexible reconstruction property. Lemma 4 implies that when a node fails, it can be regenerated such that flexible reconstruction property is retained. Hence the code maintains flexible reconstruction property after any number of node regenerations if the field size is large enough. ■

Comparison of the bound with the original regenerating codes setup: For the parameters corresponding to the MSR point in the original setup, the repair bandwidths required are identical. Otherwise, the repair bandwidth for the flexible case is higher, since a system performing flexible reconstruction (regeneration) can also perform the original reconstruction (regeneration). Details are omitted due to lack of space.

V. An Explicit Code

In this section, we provide an explicit construction for flexible regenerating codes, which draws ideas from our previous work in [8]. Due to lack of space, only an example is provided which illustrates all the key features of the explicit code.

Consider the parameters $n = 6, \alpha = 4, \beta_{\max} = 2, B = 12$. This gives $s = 3$ and a lower bound on the repair bandwidth as $\gamma \geq 8$. Divide the $B = 12$ data symbols into $\alpha = 4$ sets, represented by the vectors $\underline{f}_1, \underline{g}_1, \underline{f}_2$ and $\underline{g}_2$, each of length 3. Let $\underline{v}^{(i)}$ $(i = 1, \ldots, 6)$ be 6 vectors of length $s = 3$, forming an 3-dimensional MDS code over $\mathbb{F}_q$. Also, for $i = 1, \ldots, 6$ let $\underline{z}_1^{(i)}$ and $\underline{z}_2^{(i)}$ be arbitrary vectors of length 3.

Code: Node i, $(i = 1, \ldots, 6)$ stores the following 4 symbols, one symbol corresponding to each of the 4 sets:

Vector (set)	Symbol stored
$\underline{f}_1$	$\underline{f}_1^t\underline{v}^{(i)}$
$\underline{g}_1$	$\underline{g}_1^t\underline{v}^{(i)} + \underline{f}_1^t\underline{z}_1^{(i)}$
$\underline{f}_2$	$\underline{f}_2^t\underline{v}^{(i)}$
$\underline{g}_2$	$\underline{g}_2^t\underline{v}^{(i)} + \underline{f}_2^t\underline{z}_2^{(i)}$

Flexible Reconstruction: Suppose a DC connects to the 6 nodes with link capacities $\underline{\mu} = [3, 1, 1, 1, 2, 4]$. DC needs 3 symbols from each of the 4 sets. Consider node i passing μ_i symbols corresponding to the sets

$$\left(\sum_{j=1}^{i-1}\mu_j + 1 \text{ to } \sum_{j=1}^{i-1}\mu_j + \mu_i\right) \bmod B\ .$$

In the e.g., the symbols passed by the nodes to the DC are

Node	Symbols passed
1	$\underline{f}_1^t\underline{v}^{(1)},\ \underline{g}_1^t\underline{v}^{(1)} + \underline{f}_1^t\underline{z}_1^{(1)},\ \underline{f}_2^t\underline{v}^{(1)}$
2	$\underline{g}_2^t\underline{v}^{(2)} + \underline{f}_2^t\underline{z}_2^{(2)}$
3	$\underline{f}_1^t\underline{v}^{(3)}$
4	$\underline{g}_1^t\underline{v}^{(4)} + \underline{f}_1^t\underline{z}_1^{(4)}$
5	$\underline{f}_2^t\underline{v}^{(5)},\ \underline{g}_2^t\underline{v}^{(5)} + \underline{f}_2^t\underline{z}_2^{(5)}$
6	$\underline{f}_1^t\underline{v}^{(6)},\ \underline{g}_1^t\underline{v}^{(6)} + \underline{f}_1^t\underline{z}_1^{(6)},\ \underline{f}_2^t\underline{v}^{(6)},\ \underline{g}_2^t\underline{v}^{(6)} + \underline{f}_2^t\underline{z}_2^{(6)}$

The DC can use these symbols to decode $\underline{f}_1$ and $\underline{f}_2$, subtract out the terms $\underline{f}_j^t\underline{z}_j^{(i)}$ from the other symbols, and then decode $\underline{g}_1$ and $\underline{g}_2$.

Flexible Regeneration: Suppose node 1 fails. The new node replacing it can download at most $\beta_{\max} = 2$ symbols from any existing node, while downloading $\gamma = 8$ symbols in total. Hence, it can obtain 4 symbols which are linear combinations of $\underline{f}_1$ and $\underline{g}_1$, and the remaining 4 as linear combinations of $\underline{f}_2$ and $\underline{g}_2$. The existing nodes can pass these linear combinations in such a way that the first 4 symbols can be combined to obtain $\underline{f}_1^t\underline{v}^{(1)}$ and $\underline{g}_1^t\underline{v}^{(1)} + \underline{f}_1^t\underline{\tilde{z}}_1^{(1)}$, and the remaining 4 symbols can be combined to obtain $\underline{f}_2^t\underline{v}^{(1)}$ and $\underline{g}_2^t\underline{v}^{(1)} + \underline{f}_2^t\underline{\tilde{z}}_2^{(1)}$. Here $\underline{\tilde{z}}_1^{(1)}$ and $\underline{\tilde{z}}_2^{(1)}$ are not constrained to be equal to $\underline{z}_2^{(1)}$ and $\underline{z}_2^{(1)}$ since flexible reconstruction and regeneration operations are carried out irrespective of the values of these vectors.

Repair Bandwidth: The general form of this code can perform flexible reconstruction and flexible regeneration for any set of parameters $(n, B, \alpha, \beta_{\max})$ provided $\beta_{\max} \leq \lceil \alpha/2 \rceil$. The repair bandwidth for this code is given by

$$\gamma = (s+1)\left\lfloor \frac{\alpha}{2} \right\rfloor + s(\alpha \bmod 2) \tag{34}$$

This meets the cut-set bound for the repair bandwidth when we allow maximum flexibility for regeneration, i.e. the code is optimal for any $(n, B, \alpha, \beta_{\max} = \lceil \frac{\alpha}{2} \rceil)$ when B is a multiple of α.

References

[1] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright and K. Ramchandran, "Network Coding for Distributed Storage Systems," in *Proc. INFOCOM*, Anchorage, Alaska, May 2007.

[2] Y. Wu, A. G. Dimakis and K. Ramchandran, "Deterministic Regenerating codes for distributed storage," in *Proc. Allerton Conf.*, Urbana-Champaign, September 2007.

[3] P. Rodriguez and E. W. Biersack, "Dynamic parallel access to replicated content in the internet," *IEEE/ACM Transactions on Networking*, v. 10, p. 455-465, Aug 2002.

[4] Z. Xu, L. Xianliang, H. Mengshu and Z. Chuan, "A speed-based adaptive dynamic parallel downloading technique," *ACM SIGOPS Operating Systems Review*, v. 10, p. 63-69, Jan. 2005.

[5] Y. Wu and A. Dimakis, Reducing repair traffic for erasure coding-based storage via interference alignment, in *Proc. ISIT*, Seoul, Korea, July 2009.

[6] K. V. Rashmi, N. B. Shah, P. V. Kumar and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *Proc. Allerton Conf.*, Urbana-Champaign, Sep. 2009.

[7] —–,"Explicit codes minimizing repair bandwidth for distributed storage," in *Proc. Information Theory Workshop,*, Cairo, January 2010.

[8] —–, "Explicit Codes Uniformly Reducing Repair Bandwidth in Distributed Storage," in *Proc. NCC,* Chennai, India, Jan. 2010.

[9] D. Cullina, A. G. Dimakis and T. Ho, "Searching for Minimum Storage Regenerating Codes," in *Proc. Allerton Conf.,* Urbana-Champaign, Sep. 2009.