Locally Repairable Codes

Dimitris S. Papailiopoulos and Alexandros G. Dimakis
University of Southern California
Email:{papailio, dimakis}@usc.edu

January 27, 2013

Abstract

One main challenge in the design of distributed storage codes is the *Exact Repair Problem*: if a node storing encoded information fails, to maintain the same level of reliability, we need to exactly regenerate what was lost in a new node. A major open problem in this area has been the design of codes that *i)* admit exact and low cost repair of nodes and *ii)* have arbitrarily high data rates.

In this paper, we are interested in the metric of *repair locality*, which corresponds to the the number of disk accesses required during a node repair. Under this metric we characterize an information theoretic trade-off that binds together locality, code distance, and storage cost per node. We introduce *Locally repairable codes* (LRCs) which are shown to achieve this tradeoff. The achievability proof uses a "locality aware" flow graph gadget which leads to a randomized code construction. We then present the *first* explicit construction of LRCs that can achieve arbitrarily high data-rates.

1 Introduction

Distributed and cloud storage systems have reached such a massive scale that recovery from failures is now part of regular operation rather than a rare exception. These large scale storage systems have to allow for high data availability and be able to tolerate multiple physical node failures to prevent data loss. These systems can achieve the targeted data availability and reliability requirements by introducing redundancy among the stored bits. Erasure coded storage systems achieve high reliability without requiring the increased storage cost that is associated with data replication [5]. Three application contexts where erasure coding techniques are being currently deployed or under investigation are Cloud storage systems like Facebook's Hadoop cluster, archival storage, and peer-to-peer storage systems like Cleversafe and Wuala (see e.g. [1,3])

A central issue that arises in coded storage is the *Repair Problem*: how to maintain the encoded representation when failures (node erasures) occur. To maintain the same redundancy when a storage node leaves the system, a *newcomer* node has to join the array, access some existing nodes, and exactly reproduce the lost contents. During this repair process, there are several metrics that can be optimized: the total information read from existing disks, the total number of bits communicated in the network [7–12] (called repair bandwidth [2]), or the total number of disks required for each repair [4,6]. Currently, the most well-understood metric is repair bandwidth that was characterized in [2]. A great variety of asymptotic and explicit repair bandwidth optimal code constructions were introduced in [1,3,7–12].

This research was supported in part by NSF Career Grant CCF-1055099 and research gifts by Intel and Microsoft Research.

It seems that for cloud storage applications, the main repair performance bottleneck is the disk I/O overhead. The disk I/O is proportional to the number of nodes r involved in the repair process of a failed node. This number r defines our metric of interest: *repair locality*.

Repair locality was identified as a good metric for repair cost independently by Gopalan *et al.* [14], Oggier *et al.* [6], and Papailiopoulos *et al.* [15]. Codes that have good locality properties where studied in [4,13–15,18]. In [14], a trade-off between locality and code distance, i.e., reliability, was defined for *scalar linear* codes. However, up to now there do not exist explicit and high rate codes optimized for locality and there is no universal approach (for both linear and nonlinear codes) that characterizes the information theoretic limits between repair locality r, code distance d, and storage per node α. In this paper we address this open problem.

Our Contribution: Let a file of size M that is cut in k pieces which are encoded in $n > k$ elements of size $\alpha = (1+\epsilon)\frac{M}{k}$. We establish an information theoretic tradeoff between the repair locality r, the code distance d, and the amount of storage spent per node α, for storage codes of length n. We derive our bounds using a characterization of the code distance d in terms of entropy. A new information flow graph is fundamental to our derivations. Using random linear network coding (RLNC) arguments on this flow graph [16], we show that *linear vector codes* suffice to achieve the trade-off. We call these optimal codes *locally repairable codes*.

Then, we focus on the operational point where any k coded elements can recover the file, i.e., $d = n - k + 1$. We construct the first explicit family of LRCs that have locality r at the cost of an excess storage overhead of $\epsilon = \frac{1}{r}$. This cost can be made asymptotically (in n, k) negligible when r is any sub-linear function of k such as $r = \log(k)$, or $r = \sqrt{k}$. Our designs are vector linear, work for any n, k, r and require finite field of order n. A general LRC construction for any feasible point of the tradeoff is left as an interesting open problem.

2 Repair Locality vs. Code Distance vs. Storage Capacity

2.1 Storage Code Distance Through Entropy

In the following, we see how we can use entropy on the coded elements of a storage code to make arguments on the code distance. This way, we aim to establish a universal information theoretic tradeoff of (linear or nonlinear) codes that binds together the metric of locality, the code distance, and the storage capacity spent for each coded element or node. We would like to note that in many points in this work, we use the phrase "coded element" instead of "node".

Let a file of size M be cut in k equally sized pieces

$$\mathbf{x} = [X_1 \dots X_k], \tag{1}$$

where X_is can be viewed as k source elements over some finite field $\mathbb{F}$ that are i.i.d. random variables each having entropy $H(X_i) = \frac{M}{k}$, for all $i \in [k]$, where $[N]$ denotes the set of integers $\{1, \dots, N\}$. Moreover, let an encoding (generator) function $G : \mathbb{F}^{1\times k} \mapsto \mathbb{F}^{1\times n}$ that takes as input the k elements and outputs n coded elements

$$G(\mathbf{x}) = \mathbf{y} = [Y_1 \dots Y_n], \tag{2}$$

where each encoded element (which can be also seen as a random variable) has entropy

$$H(Y_i) = \alpha \geq \frac{M}{k}, \tag{3}$$

for all $i \in [n]$. The generator function G defines a code $\mathcal{C}$. The rate of the code is the ratio of the aggregate useful information to the aggregate stored information, i.e., the entropy of the source elements to the sum of the entropy of each encoded element

$$R = \frac{H(X_1, \dots, X_k)}{\sum_{i=1}^{n} H(Y_i)} \leq \frac{k}{n}, \tag{4}$$

with equality when $\alpha = \frac{M}{k}$.

Definition 1 (Minimum Code Distance). *The minimum distance d of the code $\mathcal{C}$ is equal to the minimum number of erasures of elements in* $\mathbf{y}$ *after which the entropy of the non-erased variables is strictly less than M, that is,*

$$d = \min |\mathcal{E}|, \tag{5}$$

such that $H\left(\{Y_1, \dots, Y_n\} \backslash \mathcal{E}\right) < M$ and $\mathcal{E} \in 2^{\{Y_1,\dots,Y_n\}}$, where $2^{\{Y_1,\dots,Y_n\}}$ is the power set of the elements in $\{Y_1, \dots, Y_n\}$.

In other words, a code has minimum distance d, when there is "enough" entropy after any $d-1$ coded element erasures to reconstruct the file. The above definition can be restated in its "dual" form: the minimum distance d of the code $\mathcal{C}$ is equal to n minus the maximum number of non-erased coded elements in $\mathbf{y}$ that cannot reconstruct the file, that is, $d = n - \max_{H(\mathcal{S})<M} |\mathcal{S}|$, where $\mathcal{S} \in 2^{\{Y_1,\dots,Y_n\}}$.

Remark 1. *Observe that the above distance definition is universal in the sense that it applies to linear or nonlinear codes and is oblivious to any type of element subpacketization.*

We continue by explicitly defining repair locality.

Definition 2 (Repair Locality). *A coded element Y_i, $i \in [n]$, has repair locality r, if it is a function of r other coded variables $Y_i = f_i(Y_{\mathcal{R}(i)})$. The set $\mathcal{R}(i)$ indexes the smallest set of r coded elements that can reconstruct Y_i and f_i is some function on these r coded elements.*

In [14] Gopalan *et al.* show that for length n *scalar linear codes*, where G is a linear function on $\mathbf{x}$, each coded element Y_i, $i \in [n]$, has entropy $\alpha = \frac{M}{k}$, and locality r, then the minimum code distance is bounded as

$$d \leq n - k - \left\lceil \frac{k}{r} \right\rceil + 2. \tag{6}$$

Observe that according to the above bound, low-locality $r << k$ is penalizing minimum distance by a component of $\frac{k}{r}$. This distance, or reliability, penalty cannot be avoided for scalar codes. On the other hand, maximum reliability, i.e., $d = n - k + 1$, costs in locality. Indeed an (n,k)-Maximum-Distance Separable (MDS) code has both the maximum possible distance $n - k + 1$ and the *worst possible* locality $r = k$. However, for our purposes we would like locality to be low: either a constant, or a sub-linear function of k.

In the following, we derive an information theoretic tradeoff between locality r, distance d, and storage per node α. We see how the third parameter α can be used to define operational points of high distance and low locality. We will refer to codes that achieve this tradeoff as (n,k,r,d,α)-LRCs.

We will eventually present explicit LRCs for any n,k,r that have the "(n,k) erasure property", i.e., that any set of k coded elements has entropy at least M, which is equivalent to requiring distance $d = n - k + 1$. This operational point will require storage $\alpha = \frac{M}{k} + \frac{1}{r}\frac{M}{k}$.

2.2 An information theoretic (r, d, α) tradeoff

In the following we determine the information theoretic minimum distance of a code, where each coded element has entropy α and repair locality r. We do that by an algorithmic proof in the same manner as [14] that bounds the distance for any possible code $\mathcal{C}$. We give a lower bound over all codes, of the largest set $\mathcal{S}$ of coded elements whose entropy is less than M. To simplify calculations, we denote the storage of each node as $\alpha = (1+\epsilon)\frac{M}{k}$, where $\epsilon \geq 0$.

The only structural property of a code that we use in our proof, is the fact that there exist $(r+1)$ sized repair groups. For a code of length n, locality r, and for each of its coded elements, say Y_i, there exist at most other r coded elements $Y_{\mathcal{R}(i)}$ that can reconstruct Y_i, for $i \in [n]$. Then, the coded elements indexed by $\Gamma(i) = \{i, \mathcal{R}(i)\}$ from an $(r+1)$-group, that has the property

$$H(Y_{\Gamma(i)}) = H(Y_i, Y_{\mathcal{R}(i)}) = H(Y_{\mathcal{R}(i)}) \leq r\alpha, \tag{7}$$

for all $i \in [n]$, due to the functional dependencies induced by the locality property. To determine the upper bound on minimum distance of a $\mathcal{C}(n, r, d, \alpha)$ code, we construct the maximum set of nodes, or coded elements, $\mathcal{S}$ that have entropy less than the M.

Theorem 1. *For a code $\mathcal{C}(n, r, d, \alpha)$, the minimum distance is bounded as*

$$\begin{aligned} d &\leq n - \left\lceil \frac{M}{\alpha} \right\rceil - \left\lceil \frac{M}{r\alpha} \right\rceil + 2 \\ &= n - \left\lceil \frac{k}{1+\epsilon} \right\rceil - \left\lceil \frac{k}{r(1+\epsilon)} \right\rceil + 2. \end{aligned} \tag{8}$$

Proof. The proof can be found in the Appendix of the full-version of the manuscript [17]. □

Remark 2. *We would like to note that the main difference of our proving technique compared to the one in [14], is that it involves counting arguments on information flows, or entropies, instead of ranks of matrices.*

Corollary 1. *In terms of the code distance, non-overlapping $(r+1)$-groups are optimal.*

Observe that if we set $\epsilon = 0$ in the above bound, we get the same bound as [14]. This means that the bound derived in [14] applies to nonlinear codes as well.

In the following, for simplicity we will assume that $(r+1)|n$ and then prove that the above bound is achievable using information flows and random linear network coding techniques.

3 Achievability of the Bound: Random LRCs

In this section, we show that the bound of Theorem 1 is achievable using a random linear network coding (RLNC) scheme [16]. In our proof, we use a variant of the information flow graph of [2] and show that when the (n, k, r, d, α) parameters of an LRC agree with the bound in Theorem 1, then the min-cut of this flow graph is large enough for a specific multicast session to be feasible. The feasibility of this multicasting problem is shown to be equivalent to the existence of (n, k, r, d, α)-LRCs. More precisely we have the following theorem which we prove in this section.

Theorem 2. *Let $(r+1)|n$. Then, there exist vector linear codes (n, k, r, d, α)-LRCs over $\mathbb{F}_{2^n}$, that have minimum distance $d = n - \lceil \frac{M}{a} \rceil - \lceil \frac{M}{ra} \rceil - 1$.*

In the same manner as [2], the information flow graph is a directed network, where the k input elements correspond to k sources, the n coded elements are represented as intermediate nodes, and the sinks of the network are what we call Data Collectors (DC), each of which requires to decode all k source elements. The specifications of this network, such as the number and degree of nodes, the edge-capacities, and the cut-set bounds, are determined by the (n, k, r, d, α) parameters. Here, in contrast to the work in [2], we need to account for the locality properties of the code. By incorporating a subgraph that accounts for the dependencies among the coded elements of a repair group, we obtain our "locality aware" flow graph.

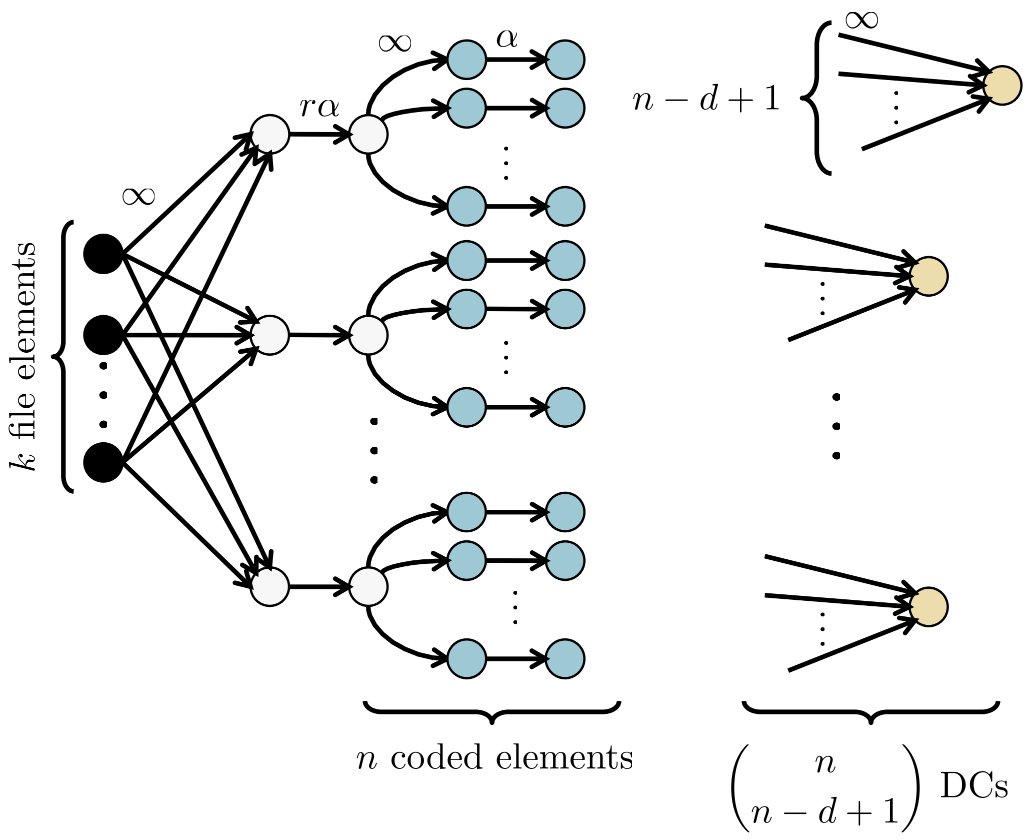

Figure 1: We sketch the directed acyclic information flow graph $\mathcal{G}(n, k, r, d, \alpha)$. The black vertices correpond to the k sources, each of which has entropy $\frac{M}{k}$. The $\frac{n}{r+1}$ white vertices correspond to nodes that bottleneck the in-flow within a repair group. The blue vertices correspond to the n nodes, or coded elements, and the yellow vertices are the DCs (sinks) of the network.

In Fig. 1, we show the general flow graph that we use in our proof. The network that is defined by the flow-graph has k sources and $N = \binom{n}{n-d+1}$ sinks (DCs). We refer to this *directed* graph as $\mathcal{G}(n, k, r, d, \alpha)$ with vertex set

$$\mathcal{V} = \Big\{ \{X_i; i \in [k]\},\ \Big\{\Gamma_j^{\text{in}}, \Gamma_j^{\text{out}}; j \in [n]\Big\}, \\ \Big\{Y_j^{\text{in}}, Y_j^{\text{out}}; j \in [n]\Big\}, \{\text{DC}_l; \forall l \in [N]\} \Big\}.$$

The directed edge set is implied by the following edge capacity function

$$c_e(v,u) = \begin{cases} \infty, (v,u) \in \left(\{X_i; i \in [k]\}, \left\{\Gamma_j^{\text{in}}; j \in \left[\frac{n}{r+1}\right]\right\}\right) \\ \quad \cup \left(\left\{\Gamma_j^{\text{out}}; j \in \left[\frac{n}{r+1}\right]\right\}, \left\{Y_j^{\text{in}}; j \in [n]\right\}\right) \\ \quad \cup \left(\left\{Y_j^{\text{out}} j \in [n]\right\}, \ \{\text{DC}_l; l \in [N]\}\right), \\ \alpha, (v,u) \in \left(\left\{Y_j^{\text{in}} j \in [n]\right\}, \ \left\{Y_j^{\text{out}} j \in [n]\right\}\right), \\ 0, \text{otherwise.} \end{cases}$$

The vertices $\{X_i; i \in [k]\}$ correspond to the k file elements and $\left\{Y_j^{\text{out}}; j \in [n]\right\}$ correspond to the coded elements. The edge capacity between the in- and out- Y_i vertices corresponds to the entropy of a single coded block. The fundamental difference with the flow graph of [2] is the additional flow constraints invoked by repair locality assumptions: coded elements (nodes) in an $(r+1)$-group have joint flow (entropy) at most $r\alpha$, instead of $(r+1)\alpha$. To enforce this constraint, we bottleneck the in-flow of each group by a node that restricts it to be at most $r\alpha$. In Fig. 2, we show the part of the flow graph that enforces the "bottleneck" induced by a repair group, where we consider the first group, without loss of generality. For a group $\Gamma(i)$, $i \in \left[\frac{n}{r+1}\right]$, we add node Γ_i^{in} that receives flow from the

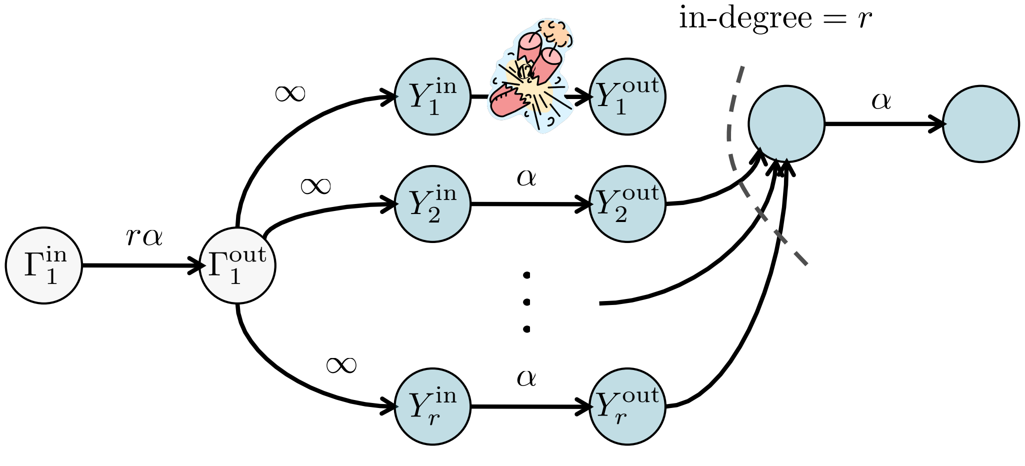

Figure 2: The flow bottlenecks of an $(r+1)$-group of elements. Observe that when an element is lost, a new node can join the system and download everything from the remaining nodes in its group.

sources and is connected with an edge of capacity $r\alpha$ to a new node Γ_i^{out}. The latter connects to the $r+1$ elements of the i-th group. We should note that when we are considering a specific group, it is implied that any block within that group can be repaired from the remaining r elements. When a block is lost, the functional dependence among the elements of that group allows a newcomer block to compute a function on the remaining r elements and reconstruct what was lost.

Linear combinations of the file elements travel along the edges of this graph towards the sinks, which we call Data Collectors (DCs). A DC needs to connect to as many coded elements as such that it can reconstruct the file. This is equivalent to requiring source-to-sink $(s-t)$ cuts between the file elements and the DCs that are at least equal to M, i.e., the file size. An $s-t$ cut in $\mathcal{G}(n,k,r,d,\alpha)$ determines the amount of flow, or entropy, that can travel from the source elements to the destinations. When d is consistent with the bound of Theorem 1, then the minimum of all the cuts is at least as much as the file size M.

Lemma 1. *The minimum source-DC cut in $\mathcal{G}(n, k, r, d, \alpha)$ is larger than or equal to M, when d is consistent Theorem 1.*

Proof. The proof can be found in the Appendix of the full-version of the manuscript [17]. □

Then, a successful multicast session on $\mathcal{G}(n, k, r, d, \alpha)$ can be interpreted as, and is equivalent to, all DCs decoding the file, i.e., all k source elements can be reconstructed at each sink using the received linear combinations. Interestingly, the linear combinations of the elements along the edges between the n node couples $(Y_i^{\text{in}}, Y_i^{\text{out}})$, are exactly the coding coefficients that need to be used by the n coded elements to achieve distance d. Hence, we will use the following lemma to prove the existence of codes.

Lemma 2 (RLNC)**.** *For a network with k sources and N destinations where η links transmit linear combination of inputs, the probability of success is at least $\left(1 - \frac{N}{q}\right)^{\eta}$*

We can now combine the above with the fact that there exists a RLNC that succeeds when $q > N$, i.e., when $q > \binom{n}{d} = \binom{n}{\lceil \frac{M}{\alpha} \rceil + \lceil \frac{M}{r\alpha} \rceil - 1}$, to obtain Theorem 2. For simplicity we used the upper bound 2^n for the binomial coefficient $\binom{n}{r}$.

4 Explicit LRC Constructions

4.1 The $d = n - k + 1$ operational point

In this section, we study the operational point of $d = n - k + 1$. We calculate the minimum storage overhead that allows the "any k property" and we construct explicit LRCs. Our codes are based on existing MDS codes, like Reed-Solomon (RS) codes, and the finite field order that we require is $q \geq n$.

We first solve for the storage overhead that is required to have distance $n - k + 1$. This overhead ϵ is the minimum one that satisfies the equation $d = n - \left\lceil \frac{k}{1+\epsilon} \right\rceil - \left\lceil \frac{k}{r(1+\epsilon)} \right\rceil + 2$, where $d = n - k + 1$. Therefore, the minimum storage overhead for erasure distance can be found through the following optimization

$$\begin{aligned} &\min \epsilon \\ \text{subject to: } &\left\lceil \frac{k}{1+\epsilon} \right\rceil + \left\lceil \frac{k}{r(1+\epsilon)} \right\rceil = k + 1 \\ &\epsilon \geq 0. \end{aligned} \tag{9}$$

Due to the ceiling function in the above constraint we do not obtain a closed form expression of the minimizer $\epsilon_{\min}$. However, we identify a potential range of ϵ values that satisfy the equation

$$n - k + 1 = n - \left\lceil \frac{k}{1+\epsilon} \right\rceil - \left\lceil \frac{k}{r(1+\epsilon)} \right\rceil + 2. \tag{10}$$

Lemma 3. *An LRC with node repair locality r and distance $n - k + 1$ requires an additional storage overhead that is equal to $\epsilon_{\min} = \frac{1}{r} - \delta_k$, where $\delta_k \in \left[0, \frac{r+1}{r} \frac{1}{k+1}\right]$.*

Proof. The proof can be found in the Appendix of the full-version of the manuscript [17]. □

Remark 3. *We can always perform a line search within $\left[\frac{1}{r} - \frac{r+1}{r}\frac{1}{k+1}, \frac{1}{r}\right]$ to find the minimum ϵ that satisfies the erasure distance.*

In the following we present LRCs with repair locality r for $\epsilon = \frac{1}{r}$. In the proof of Lemma 3, we show that when $\epsilon = \frac{1}{r}$, then $d = n - k + 1$ is the maximum possible distance when $(r + 1) \nmid k$. When $(r + 1) | k$ the optimal distance is $d = n - k + 2$.

4.2 Code Construction

The codes that follow are optimal, i.e., LRC, for all n, k, r, when $(r+1) \nmid k$ and $(r+1)|n$.

Let a file $\mathbf{x}$, of size $M = rk$, that is subpacketized in r parts, $\mathbf{x} = \left[\mathbf{x}^{(1)} \ldots \mathbf{x}^{(r)}\right]$, with each $\mathbf{x}^{(i)}$, $i \in [r]$, having size k. We encode each of the r file parts independently, into coded vectors $\mathbf{y}^{(i)}$ of length n, where $(r+1)|n$, using an outer (n,k) MDS code $\mathbf{y}^{(1)} = \mathbf{x}^{(1)}\mathbf{G}, \ \ldots, \ \mathbf{y}^{(r)} = \mathbf{x}^{(r)}\mathbf{G}$, where $\mathbf{G}$ is the $n \times k$ MDS generator matrix. As MDS pre-codes, we use (n,k)-RS codes that require $\mathbb{F}_q$, with $q \geq n$. We generate a single parity sum vector from all the coded vectors $\mathbf{s} = \sum_{i=1}^{r} \mathbf{y}^{(i)}$. This *precoding* process yields a total of rn coded blocks in the $\mathbf{y}^{(i)}$ vectors and n parity blocks in $\mathbf{s}$, i.e., an aggregate of $(r+1)n$ blocks available to place in n nodes. In our code, each node expends $\alpha = \frac{M}{k} + \frac{1}{r}\frac{M}{k} = r + 1$ (coded blocks) in storage capacity.

Below we state the circular placement of elements in nodes of the first $(r+1)$-group

	node 1	node 2	$\ldots$	node r	node $r+1$
blocks of $\mathbf{y}^{(1)}$	$y_1^{(1)}$	$y_2^{(1)}$	$\ldots$	$y_r^{(1)}$	$y_{r+1}^{(1)}$
blocks of $\mathbf{y}^{(2)}$	$y_2^{(2)}$	$y_3^{(2)}$	$\ldots$	$y_{r+1}^{(2)}$	$y_1^{(2)}$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
blocks of $\mathbf{y}^{(r)}$	$y_r^{(r)}$	$y_{r+1}^{(r)}$	$\ldots$	$y_{r-2}^{(r)}$	$y_{r-1}^{(r)}$
blocks of $\mathbf{s}$	s_{r+1}	s_1	$\ldots$	s_{r-1}	s_r

.

There are three key properties that are obeyed by our placement:
i) each node contains r coded blocks coming from different $\mathbf{y}^{(l)}$ coded vectors and 1 additional parity element,
ii) The blocks in the $r+1$ nodes of the i-th $(r+1)$-group, have indices that appear only in that specific repair group, $i \in \left[\frac{n}{r+1}\right]$, and
iii) the blocks of each "row" have indices that obey a circular pattern, i.e., the first row of elements has indices $\{1, 2, \ldots, r+1\}$ the second $\{2, 3, \ldots, r+1, 1\}$, and so on.

We follow the same placement for all $\frac{n}{r+1}$ groups. In Fig. 3, we show an LRC of the above construction with $n = 6$ and $k = 4$ that has locality 2.

4.3 Repairing Lost Nodes

We will concentrate on the repair of lost nodes in the first repair group of r nodes. This is sufficient since the block placement is identical across repair groups. The key observation is that each node within a repair group stores $r+1$ blocks of *distinct* indices: the $r+1$ blocks of a particular index are stored in $r+1$ distinct nodes within the repair group. Hence, when for example the first node fails, then element $y_1^{(1)}$ is regenerated by downloading s_1 from the second node in the group, $y_1^{(r+1)}$ from the third, $\ldots$, and $y_1^{(2)}$ from the last node in the group. A simple XOR of the above blocks suffices to reconstruct the lost block. Hence, for every lost block of a failed node, the r remaining blocks of the same index that are stored in the r remaining nodes of the repair group have to be XORed to regenerate what was lost.

Hence, a single block repair has locality r. Interestingly, the same applies to the repair locality of an entire node. For each lost block, r other coded blocks are used for regeneration, and all originate from the r remaining nodes. In Fig. 4, we show how repair is performed for the code construction presented in Fig. 3.

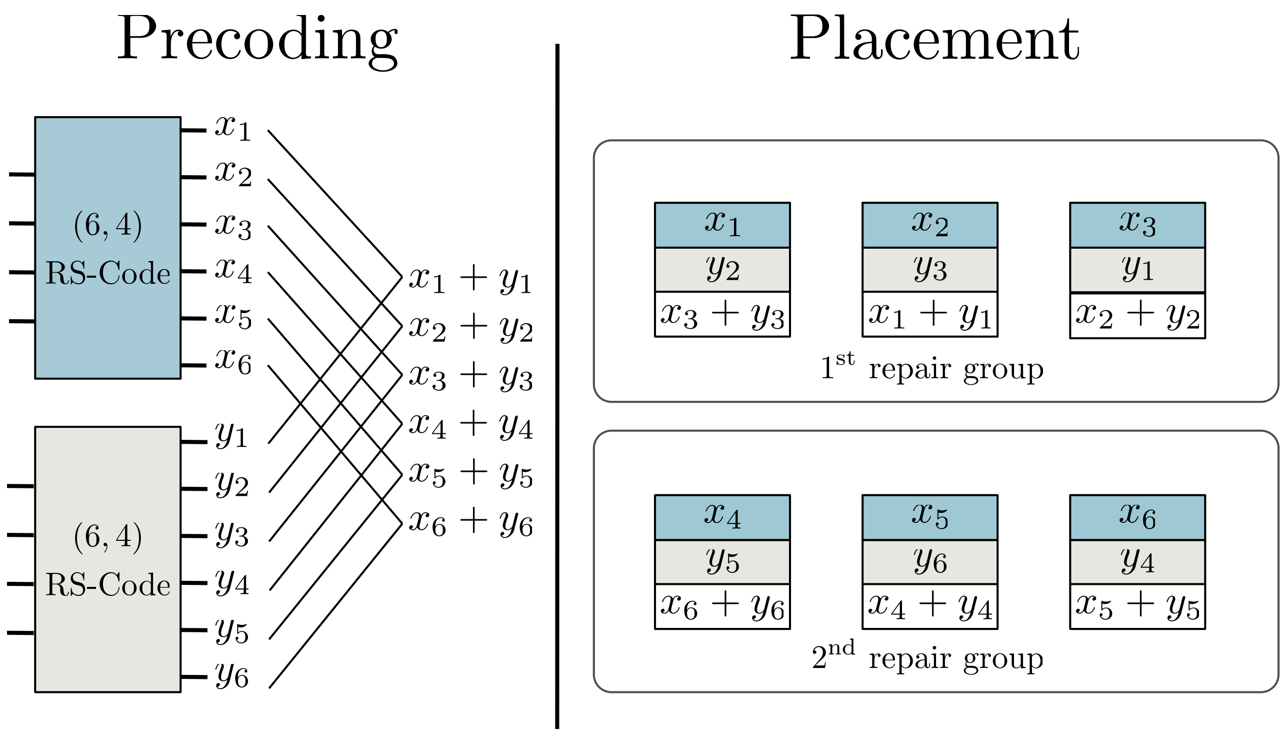

Figure 3: A $(6,4)$-LRC with locality 2. Observe that we spent 3 blocks per coded element, instead of $\frac{M}{k} = 2$. This allows us to have easy and repairability with locality 2. The distance of this code is guaranteed through the 2 $(6,4)$-MDS precodes.

4.4 Rate and Code Distance

The effective coding rate of the codes presented in this section is

$$R = \frac{\text{size of useful information}}{\text{total storage spent}} = \frac{r \cdot k}{(r+1) \cdot n}. \tag{11}$$

That is, the rate of the code is a fraction $\frac{r}{r+1}$ of the coding rate of an (n,k) MDS code, hence is always upper bounded by $\frac{r}{r+1}$. This is due to the extra storage overhead required to store the parity blocks s_i, $i \in [n]$.

Remark 4. *Observe that if we set the repair locality to $r = f(k)$ and f is a sub-linear function of k (i.e., $\log(k)$ or $\sqrt{k}$), then we obtain non-trivially low locality $r << k$, while the excess storage cost $\epsilon = \frac{1}{r}$ is vanishing when n, k grow.*

The distance of the presented code is (at least) $d = n - k + 1$ due to the MDS precodes that are used in its design: any k nodes in the system contain rk distinct coded pieces, k from each of the r file pieces. Hence, by performing erasure decoding on each of these r k-tuples of blocks, we can generate the r pieces of the file.

References

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

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

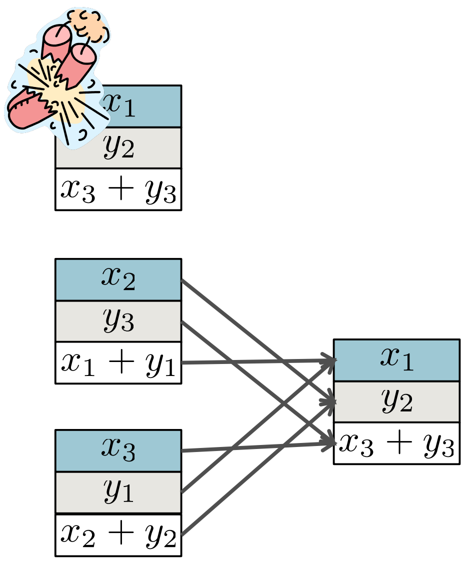

Figure 4: The repair of a node failure. The repair locality is 2 since 2 remaining nodes are involved in reconstructing the lost information of the first node, or the first coded element. Observe that we repair by only transferring blocks: no block combinations are need to be performed at the sender nodes.

[3] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," in *IEEE Proceedings*, vol. 99, pp. 476 – 489, Mar. 2011.

[4] O. Khan, R. Burns, J. Plank, and C. Huang, "In search of I/O-optimal recovery from disk failures," in *Hot Storage 2011, 3rd Workshop on Hot Topics in Storage and File Systems*, Portland, OR, Jun., 2011.

[5] H. Weatherspoon and J. D. Kubiatowicz, "Erasure coding vs. replication: a quantitative comparison," *in Proc. IPTPS*, 2002.

[6] F. Oggier and A. Datta, "Self-repairing homomorphic codes for distributed storage systems," in *Proc. IEEE Infocom 2011*, Shanghai, China, Apr. 2011.

[7] K.V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran "Explicit construction of optimal exact regenerating codes for distributed storage," In *Allerton Conf. on Control, Comp., and Comm.*, Urbana-Champaign, IL, September 2009.

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

[9] K. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," in *IEEE Trans. on Inform. Theory*, vol. 57, pp. 5227 – 5239, Jul. 2011.

[10] I. Tamo, Z. Wang, and J. Bruck "MDS Array Codes with Optimal Rebuilding," in *Proc. IEEE Intern. Symp. on Inform. Theory (ISIT) 2001*, St. Petersburg, Russia, Aug. 2011.

[11] V. R. Cadambe, C. Huang, S. A. Jafar, and J. Li, "Optimal repair of MDS codes in distributed storage via subspace interference alignment," *arxiv pre-print 2011*. Preprint available at http://arxiv.org/abs/1106.1250.

[12] D. S. Papailiopoulos, A. G. Dimakis, and V. R. Cadambe, "Repair optimal erasure codes through Hadamard designs," In *Allerton Conf. on Control, Comp., and Comm.*, Urbana-Champaign, IL, September 2011.

[13] C. Huang, M. Chen, and J. Li, "Pyramid codes: flexible schemes to trade space for access efficiency in reliable data storage systems", *in Proc. IEEE International Symposium on Network Computing and Applications (NCA 2007)*, Cambridge, MA, Jul. 2007.

[14] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, “On the locality of codeword elements,” Preprint available at http://arxiv.org/abs/1106.3625.

[15] D. S. Papailiopoulos, Jianqiang Luo, Alexandros G. Dimakis, C. Huang, and J. Li, “Simple regenerating codes: network coding for cloud storage”, in *IEEE International Conference on Computer Communications (Infocom) 2012, Miniconference*.

[16] T. Ho, R. Koetter, M. Médard, M. Effros, J. Shi, and D. Karger, ”A random linear network coding approach to multicast,” *IEEE Transactions on Information Theory*, vol. 52 (10), pp. 4413–4430, Oct. 2006.

[17] D. S. Papailiopoulos and A. G. Dimakis, “Locally repairable codes,” full version available at `http://tinyurl.com/82cucvd`

[18] J. Han and L. A. Lastras-Montaño, “Reliable memories with subline accesses” in *Proc. 2010 IEEE Int. Symp. on Inform. Theory (ISIT)*, pp. 2531–2535, 2007.