Locally Repairable Codes

Dimitris S. Papailiopoulos and Alexandros G. Dimakis
The University of Texas at Austin
dimitris@utexas.edu, dimakis@austin.utexas.edu

March 15, 2013

Abstract

Distributed storage systems for large scale applications typically use replication for reliability. Recently, erasure codes were used to reduce the large storage overhead, while increasing data reliability. A main limitation of off-the-shelf erasure codes is their high-repair cost when nodes fail in the storage system. A major open problem in this area has been the design of codes that *i)* are repair efficient and *ii)* achieve arbitrarily high data rates.

In this paper, we explore the repair metric of *locality*, which corresponds to 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 the storage capacity of each node. We study optimal *locally repairable codes* (LRCs), codes that are shown to achieve this trade-off. The achievability proof uses a locality aware flow-graph gadget which leads to a randomized code construction. Finally, we present an optimal and explicit LRC construction that achieves arbitrarily high data-rates. Our locality optimal code is based on simple combinations of Reed-Solomon coded blocks. Moreover, it is shown to achieve very low repair-bandwidth.

1 Introduction

Traditional architectures for large-scale storage rely on systems that provide reliability through block replication. The major disadvantage of replication is the large storage overhead. As the amount of stored data is growing faster than hardware infrastructure, this becomes a major data center cost bottleneck.

Erasure coding techniques achieve higher data reliability with considerably smaller storage overhead [2]. For that reason various erasure codes are currently implemented and deployed in production storage clusters. Applications where coding techniques are being currently deployed include cloud storage systems like Windows Azure [3], big data analytics clusters (*e.g.*, the Facebook Analytics Hadoop cluster [4]), archival storage systems, and peer-to-peer storage systems like Cleversafe and Wuala.

However, classical erasure codes (such as Reed-Solomon) are highly suboptimal for distributed storage settings [5]. For example, the Facebook analytics Hadoop cluster discussed in [4], deployed

Parts of this work were presented in [1].

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

A significant portion of this work has been completed while the authors were with the department of Electrical Engineering at the University of Southern California.

Reed-Solomon encoding for 8% of the stored data. This 8% of the stored data was reported to generate repair traffic approximately equal to 20% of the total network traffic. This repair inefficacy is currently the main bottleneck that limits the code deployment in storage systems.

Three major repair cost metrics have been identified in the recent literature: *i)* the number of bits communicated in the network, also known as the *repair-bandwidth* [5–10], *ii)* the number of bits read during each repair, the *disk-I/O* [8, 11], and *iii)* more recently the number of nodes that participate in the repair process, also known as, *repair locality*. Each of these metrics is more relevant for different systems and their fundamental limits are not completely understood. In this work, we focus on the metric of repair locality, one that seems most relevant for single-location high-connectivity storage clusters.

Locality was identified as a good metric independently by Gopalan *et al.* [12], Oggier *et al.* [13], and Papailiopoulos *et al.* [14]. Consider a code of length n, with k information symbols. A symbol i has locality r_i, if it can be reconstructed by accessing at most r_i other symbols in the code. For example, in an (n, k) maximum-distance separable (MDS) code, every symbol has trivial locality k. We will say that a systematic code has *information-symbol locality* r, if all the k information symbols have locality r. Similarly, a code has *all-symbol locality* r, if all n symbols have locality r. Codes with good locality properties were initially studied in [15, 16].

In [12], a trade-off between code distance, *i.e.*, reliability, and information-symbol locality was derived for scalar linear codes. Bounds on the code-distance for a given locality and code constructions were also derived and generalized in parallel and subsequent works [17–20]. Some works extend the designs and theoretic bounds to the case where repair bandwidth and locality are jointly optimized, under multiple local failures [18, 19], and under security constraints [19].

Our Contribution: In this work, we provide the first information theoretic bounds and explicit constructions for codes with all-symbol locality. We refer to a code with all-symbol locality as a locally repairable code (LRC). An LRC with locality r is a code where any of its n coded symbols can be reconstructed by accessing and processing at most r other symbols. We present a universal trade-off that binds together the code distance d, the locality r, and the storage capacity of each node α. Our trade-off is information theoretic and covers all codes, linear or nonlinear, and looks as follows:

Theorem 1. *Let a code with all-symbol locality r that encodes a file of size M in n symbols, each of size α. Then,*

$$d \leq n - \left\lceil \frac{M}{\alpha} \right\rceil - \left\lceil \frac{M}{r\alpha} \right\rceil + 2.$$

Our trade-off is derived using a converse and an achievability scheme. We use counting techniques on entropies to provide the converse. We prove the achievability using a novel information flow-graph gadget. In constrast to [5], the flow-graph that we construct is finite, locality aware, and simpler to analyze. Using random linear network coding (RLNC) arguments on this flow-graph [21], we construct randomized vector linear LRCs that achieve the trade-off, when $(r + 1)|n$.

Finally, we explicitly construct optimal LRCs for the operational point of distance d, where any k coded symbols suffice to recover the file of size M.Our designs are vector linear, work for any n, k, r such that $(r + 1)|n$, and each symbol stored requires only $r \log(n)$ bits in its representation. We show that these codes not only have optimal locality, but have simple repairs based on XORs, and attain repair bandwidth that can in some cases come close to the optimal bounds of [5].

The remainder of this paper is organized as follows. In Section II, we provide the dinstance bound of a code with all symbol locality. In Section III, we prove that this bound is achievable using randomized vector codes. In Section IV, we provide an explicit LRC construction, and discuss its properties. In our appendices, we have our proofs for our more technical lemmas and theorems of this work.

2 Locally Repairable Codes: Bounds

In this section we present our information theoretic bounds for LRCs.

2.1 Code distance through entropy

In the following, we use entropy to characterize the distance of a code. The use of entropy, will in turn ensure that our bounds are universal, and hold for any code. We will use the following definitions to establish a trade-off that binds together the metric of locality r, the code distance d, and the storage capacity α spent for each coded symbol, or node.

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

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

where X_is can be viewed as k source symbols, over some domain $\mathcal{X}$, 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 $G : \mathcal{X}^k \mapsto \mathcal{Y}^n$ that maps k symbols from the $\mathcal{X}$ domain to n coded symbols in a codomain $\mathcal{Y}$,

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

where each encoded symbol has entropy

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

for all $i \in [n]$. The generator function G defines a code $\mathcal{C}$. The rate of the code is the ratio of the total source entropy to the aggregate entropy of the stored information

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

with equality when $\alpha = \frac{M}{k}$ and the source symbols are independent.

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

$$d = \min_{H(\{Y_1, \dots, Y_n\} \setminus \mathcal{E}) < M} |\mathcal{E}|,$$

where $\mathcal{E} \in 2^{\{Y_1, \dots, Y_n\}}$ and $2^{\{Y_1, \dots, Y_n\}}$ is the power set of the symbols 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 symbol 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 symbols 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.*

We continue by explicitly defining repair locality.

Definition 2 (Repair Locality). *A coded symbol 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 symbols that can reconstruct Y_i and f_i is some function on these r coded symbols.*

In [12], Gopalan *et al.* show that for length n *scalar linear codes*, where G is a linear function on $\mathbf{x}$, each coded symbol Y_i, $i \in [n]$, has entropy $\alpha = \frac{M}{k}$, and we require information-symbol locality r, then the minimum code distance is bounded as

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

The above bound states that locality takes its toll on code distance: locality r reduces code distance by $\frac{k}{r} - 1$. Observe that (n,k)-MDS codes have both the maximum possible distance $n - k + 1$ and the *worst possible* locality $r = k$.

In the following, we show that the above distance penalty can neither be avoided for the case of all-symbol locality. However, if we allow a little more storage than $\alpha = \frac{M}{k}$, we see that we can obtain much higher distance. We proceed by deriving an information theoretic trade-off between all code parameters n, r, d, M, α.

Before we proceed, to simplify calculations, we denote the entropy of each symbol (or the storage capacity of each node), without loss of generality, as

$$\alpha = (1 + \epsilon)\frac{M}{k}$$

where $\epsilon \geq 0$. Moreover, to have a more compact discussion, we also introduce the following notation.

Definition 3. *We denote by $\mathcal{C}(n, r, d, M, \alpha)$ a code that takes a file of size M and encodes it in n symbols, where each symbol has size α. The code has all-symbol locality r, i.e., each symbol can be reconstructed by r other symbols. The code has distance d, i.e., the file of size M can be reconstructed from any collection of $n - d + 1$ symbols. The rate of this code is $R = \frac{M}{n\alpha}$.*

2.2 An information theoretic trade-off between locality, distance, and storage

In this section, we provide an information theoretic bound for locally repairable codes. More precisely, we answer the question: assuming that a code has locality r, what is the maximum distance d it could have? We provide a universal upper bound on the minimum distance of a code of length n, with all-symbol locality r, where each coded symbol has size α. We do so by an algorithmic proof, in a very similar manner to [12]. We will see that deriving such a distance bound, reduces to finding the cardinality of the largest set $\mathcal{S}$ of coded symbols whose entropy is less than M.

In our proof, the only structural property of a code that we use is the fact that, since every symbol has locality r, there are $r + 1$ symbols whose entropy cannot be of maximal value. Specifically, if a code has locality r, then for each of its coded symbols, say Y_i, there exist at most other r coded symbols $Y_{\mathcal{R}(i)}$ that can reconstruct Y_i, for $i \in [n]$. Then, the coded symbols indexed by $\Gamma(i) = \{i, \mathcal{R}(i)\}$ from an $(r + 1)$-group, that has the property

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

for all $i \in [n]$; the above comes due to the functional dependencies induced by locality.

Theorem 1. *A code $\mathcal{C}(n,r,d,M,\alpha)$ has minimum distance d that is bounded as*

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

Proof. The proof can be found in the Appendix. □

Corollary 1. *In terms of achieving the aforementioned code distance bound, non-overlapping $(r+1)$-groups is one of the (possibly many) optimal arrangements of repair groups.*

Remark 2. *Observe that (n,k)-maximum-distance separable (MDS) codes corresponds to the operational point where $d_{MDS} = n-k+1$ and $\alpha_{MDS} = \frac{M}{k}$. An (n,k)-MDS code has the worst possible locality $r_{MDS} = k$. Moreover, observe that we can get distance as high as that of an MDS code, if we are willing to spend more on storage,. i.e., if we have $\alpha = \left(1+\frac{1}{r}\right)\alpha_{MDS}$.*

Remark 3. *In the above bound, if we set $\epsilon = 0$, we get the same bound as [12]. The distance bound in [12] was derived for linear scalar codes, where $\alpha = \frac{M}{k}$, and only for information-symbol locality. Our theorem states that the same bound applies to vector linear, or nonlinear codes, and all-symbol locality.*

In the following, we prove that the above is tight whenever $(r+1)|n$. This does not rule out that the bound is tight in general, but for simplicity we only consider the case where we can divide all symbols in non-overlapping repair groups of $r+1$ symbols.

3 Achievability of the Bound: Random LRCs

In this section, we show that the bound of Theorem 1 is achievable.

Theorem 2. *Let $(r+1)|n$. Then, if there exist locally repairable codes $\mathcal{C}(n,r,d,M,\alpha)$, over $\mathbb{F}_{2^n}$, with minimum distance $d = n - \lceil \frac{M}{a} \rceil - \lceil \frac{M}{ra} \rceil - 1$.*

We first provide a quick sketch of the proof and then proceed with a more technical discussion.

3.1 Proof architecture

Our achievability proof is based on two key components: a novel information flow-graph and random linear network coding (RLNC) techniques. The key fact is that the existence of a code can be recast in testing wether a directed network of nodes can have a feasible multicast, where all destination nodes can decode all source symbols. We first construct an information flow-graph $\mathcal{G}(n,r,d,M,\alpha)$, that depends on the code parameters and is locality aware. This means that locality is embedded as a combinatorial property of the flow-graph. Then, we recast the problem of existence of a $\mathcal{C}(n,r,d,M,\alpha)$ code that satisfies the bound in Theorem 1, as a feasibility of a multicast session on $\mathcal{G}(n,r,d,M,\alpha)$. We obtain a cut-set analysis for $\mathcal{G}(n,r,d,M,\alpha)$, that directly ties the distance bound of $\mathcal{C}(n,r,d,M,\alpha)$ with the minimum cut of $\mathcal{G}(n,r,d,M,\alpha)$. We continue to show using RLNC theory, that if the cut of $\mathcal{G}(n,r,d,M,\alpha)$ is large enough, then the multicast session is possible. This will then directly imply the existence of $\mathcal{C}(n,r,d,M,\alpha)$.

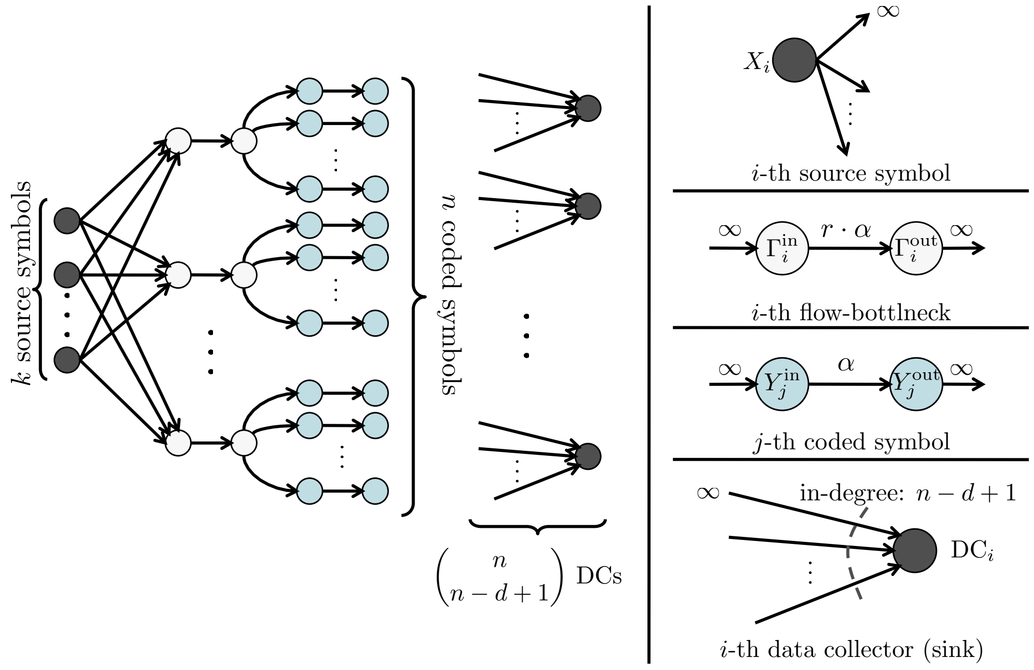

Figure 1: We sketch the directed acyclic information flow-graph $\mathcal{G}(n,r,d,M,\alpha)$. The left-most black vertices correpond to the k sources, X_i, each of which has entropy $\frac{M}{k}$. The $\frac{n}{r+1}$ grey vertices correspond to nodes that bottleneck the in-flow within a repair group.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

3.2 Proof of Theorem 2

In Fig. 1, we show the general flow-graph that we use in our proof. We refer to this *directed* graph as $\mathcal{G}(n,r,d,M,\alpha)$ with vertex set

$$\begin{aligned}\mathcal{V} = \big\{&\{X_i; i\in[k]\},\ \left\{\Gamma_j^{\text{in}}, \Gamma_j^{\text{out}}; j\in[n]\right\},\\ &\left\{Y_j^{\text{in}}, Y_j^{\text{out}}; j\in[n]\right\}, \{\text{DC}_l; \forall l\in[N]\}\big\}.\end{aligned}$$

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

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

The network that is defined by $\mathcal{G}(n,r,d,M,\alpha)$ has k sources and $N=\binom{n}{n-d+1}$ sinks, the Data Collectors (DCs). The vertices $\{X_i; i\in[k]\}$ correspond to the k file symbols and $\left\{Y_j^{\text{out}}; j\in[n]\right\}$ correspond to the n coded symbols. The edge capacity between the in- and out- Y_i vertices corresponds to the entropy of a single coded symbol. In the network described by Fig. 1, we have functions of the file symbols traveling along the edges of this graph towards the DCs (sinks).

Each DC has in-degree $n-d+1$, with incident vertices originating from $n-d+1$ coded symbol nodes. In this network, we wish all DCs to decode all source symbols. The intermediate function of the source symbols that the node Y_i^{in} transmits to Y_i^{out}, corresponds to the encoding map of a code $\mathcal{C}$ that maps the source symbols $X_1, \ldots, X_k$ to the coded symbol Y_i. This means that a DC has access to a subset of $n-d+1$ coded symbols. If this subset is sufficient to reconstruct all X_is, then this DC can successfully decode all source symbols. If all N DCs can decode all k source symbols then, the code that "sits" on the edges between the Y_i^{in} and Y_i^{out}, has minimum distance at least d.

The fundamental difference with the flow-graph of [5] is the additional flow constraints that need to be enforced due to the repair locality assumptions: the $r+1$ coded symbols (nodes) in an $(r+1)$-group have joint flow (entropy) at most $r\alpha$, instead of $(r+1)\alpha$. To enforce this constraint, we create non-overlapping groups of $r+1$ coded symbols (hence the $(r+1)|n$ assumption). Each repair group, requires an in-flow bottlneck that restricts the joint entropy to be at most $r\alpha$.

In Fig. 2, we show the part of the flow-graph that enforces the bottleneck. For a group $\Gamma(i)$,

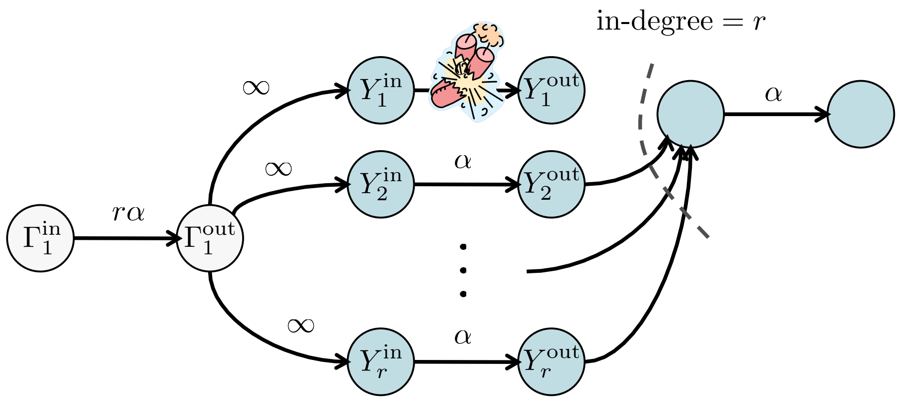

Figure 2: The flow bottlenecks of an $(r+1)$-group of symbols. Observe that when an symbol is lost, a new node can join the system and download everything from the remaining nodes in its group.

$i \in \left[\frac{n}{r+1}\right]$, we have a node Γ_i^{in} that receives flow from the sources and is connected with an edge of capacity $r\alpha$ to a new node Γ_i^{out}. The latter connects to the $r+1$ symbols of the i-th group. When a block is lost, the functional dependence among the symbols of that group allows a newcomer node (as shown in the figure) to compute a function on the remaining r symbols and reconstruct what was lost. That is, the newcomer can connect to all remaining r nodes and download all of their contents of size $r\alpha$. Operationally, the newcomer can be seen as a local DC that has access to r local nodes. Since the total flow within that group is $r\alpha$, the newcomer can reconstruct (if appropriate local functions are picked) all packets within that group, hence its in-flow is sufficient to reconstruct what was lost.[1]

Now, a necessary condition for a DC to be able to reconstruct all source information, is that the minimum cut between the sources and that DC has to be at least M. Hence, requiring all DCs to be able decode all source symbols, implies the necessary requirement that all source-to-sink ($s-t$) cuts between the file symbols X_i and the DCs have to be at least equal to M, i.e., the size of the file. We now establish that when d is consistent with the bound of Theorem 1, then the minimum of all $s-t$ cuts in $\mathcal{G}(n, r, d, M, \alpha)$ are at least as large as the file size M.

[1]An example of local functions that are good for the purpose of local repairs are simple $(r+1, r)$-MDS codes: any r symbols can reconstruct all in-flowing source functions.

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

Proof. The proof can be found in the Appendix. □

The above can be combined with the following result from [21].

Lemma 2 (RLNC). *Let a network with k sources and N destinations, where the minimum $s - t$ cut of the network is larger than the size of the source symbols. Then, if the nodes in the network transmit random linear combinations of their inputs, over a field of size $q > N$, the probability of successful decoding at all destination nodes is nonzero. That is, when $q > N$ there exists a multicasting solution that achieves the min-cut capacity of the network.*

A successful multicast session on $\mathcal{G}(n, r, d, M, \alpha)$, means that all source symbols can be sent through the network, so that the DCs receive functions of them that are sufficient to decode the source. A successful multicast session on $\mathcal{G}(n, r, d, M, \alpha)$ determines all local functions of in-flow symbols that each node in the network transmits to its incident vertices. The particular functions that travel between the in and out Y_i nodes is exactly a code construction $\mathcal{C}(n, r, d, M, \alpha)$ that has distance d, if the multicast on $\mathcal{G}(n, r, d, M, \alpha)$ is successful. This means that constructing successful multicast sessions on $\mathcal{G}(n, r, d, M, \alpha)$ where all DCs decode all sources, automatically constructs a code $\mathcal{G}(n, r, d, M, \alpha)$.

We can now combine the above with the fact that there exists a RLNC solution that succeeds when $q > N$, i.e., when

$$q > \binom{n}{N} = \binom{n}{n-d+1}. \tag{2}$$

For simplicity we can use used the upper bound $\binom{n}{N} < 2^n, \forall N \leq n$, for the binomial coefficient and then obtain Theorem 2. This concludes our proof.

4 Locally Repairable Codes: Explicit Constructions

In this section, we provide an explicit LRC family for the operational point on the distance trade-off, where any k subsets of coded nodes can reconstruct all k file symbols, i.e., when $d = n - k + 1$. For this regime we see that the excess in storage that we need to pay is $\epsilon = \frac{1}{r}$. Our codes apart from being locality optimal, they have the following significant benefits:

1. They achieve arbitrarily high data rates.
2. They can be constructed using Reed-Solomon encoded blocks.
3. The repair of a lost node requires downloading blocks and XORing them at a destination node.
4. their vector size is r.
5. they achieve non-trivially low repair bandwidth.

4.1 Code Construction

Let a file $\mathbf{x}$, of size $M = rk$, that is subpacketized in r parts,

$$\mathbf{x} = \left[\mathbf{x}^{(1)} \dots \mathbf{x}^{(r)}\right],$$

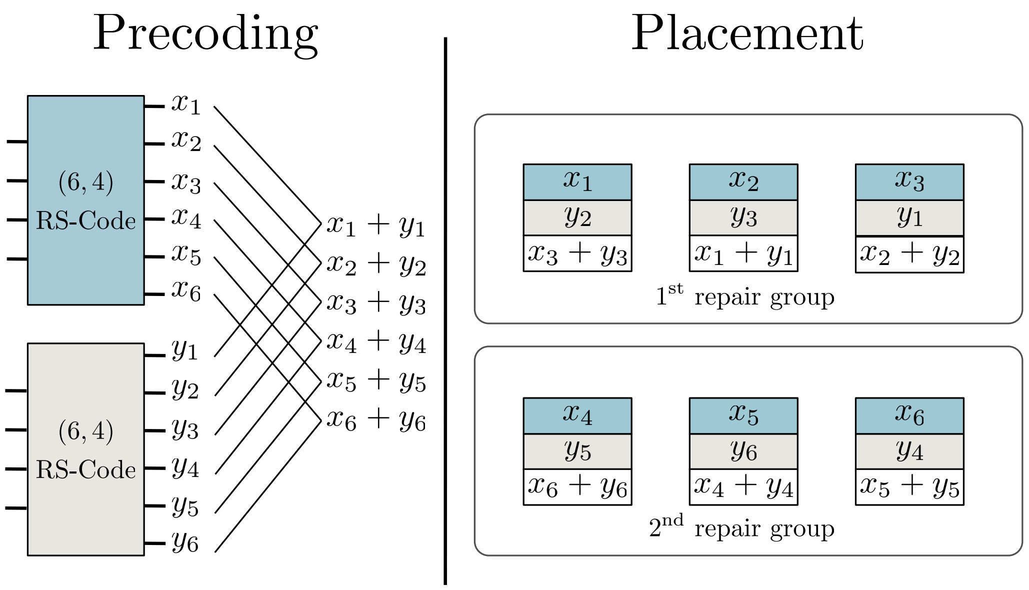

Figure 3: A $(6,4)$-LRC with locality 2. Observe that we store 3 blocks per coded symbol, instead of $\frac{M}{k} = 2$. This allows us to have simple repairability with locality 2. The distance of this code is guaranteed through the 2 $(6,4)$-MDS pre-codes.

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 and $n \times k$ 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

$$\mathbf{s} = \sum_{i=1}^{r} \mathbf{y}^{(i)}.$$

Observe that when $\mathbb{F}$ is a binary field extension, then the above sum can simply restated as XORing the $\mathbf{y}^{(i)}$ vectors.

The above *pre-coding* process yields a total of rn coded blocks, the $\mathbf{y}^{(i)}$ vectors, and n parity blocks, the $\mathbf{s}$ vector. That is we have an aggregate of $(r+1)n$ blocks available to place in n nodes: $r+1$ blocks per node. Hence, in our code $\epsilon = \frac{1}{r}$ and each node expends

$$\alpha = \frac{M}{k} + \frac{1}{r}\frac{M}{k} = r + 1 \text{ (coded blocks)}$$

in storage capacity. In Appendix C, we show distance optimality of our codes, when $(r+1) \nmid k$.

In Table I, we state the circular placement of symbols in nodes of the first $(r+1)$-group . There are three key properties that are obeyed by our placement:

1. each node contains r coded blocks coming from different $\mathbf{y}^{(l)}$ coded vectors and 1 additional parity symbol,

2. the blocks in the $r+1$ nodes of the i-th $(r+1)$-group, have indices that appear only in that specific repair group, and

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

Table 1: Code construction

3. the blocks of each row have indices that obey a circular pattern, i.e., the first row of symbols has indices $\{1,2,\dots,r+1\}$ the second $\{2,3,\dots,r+1,1\}$, and so on.

In Fig. 3, we show an LRC of the above construction with $M=12$, $\alpha=3$, $n=6$ and $k=4$, that has locality 2.

4.2 Repairing Lost Nodes

Here, we see that the repair of each lost node requires contacting r nodes, i.e., the locality of the code is r. Without loss of generality, we consider the repair of a node in the first repair group of $r+1$ nodes. This is sufficient since the nodes across different repair groups follow the same placement.

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. When for example the first node fails, then $y_1^{(1)}$, the symbol of the first row, is regenerated by downloading s_1 from the second node, $y_1^{(r+1)}$ from the third, and so on. Once all these symbols are downloaded a simple XOR of all of them is exactly equal to $y_1^{(1)}$. In the same manner, for each node, in each repair group when we need to reconstruct a lost block, we first download the r remaining blocks of the same index and XOR them together to regenerate the desired lost block. Since each block can be reconstructed by contacting r other blocks, and since the repair is confined within the group of r remaining nodes in a repair group, the code has locality r.

In Fig. 4, we show how repair is performed for the code construction presented in Fig. 3.

4.3 Code Rate

The effective coding rate of our LRC is

$$R=\frac{\text{size of useful information}}{\text{total storage spent}}=\frac{M}{n\cdot\alpha}=\frac{r}{r+1}\frac{k}{n}.$$

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 loss in rate is incurred due to the use of the extra XOR stripe of blocks, that is used for efficient and local repairs. 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.

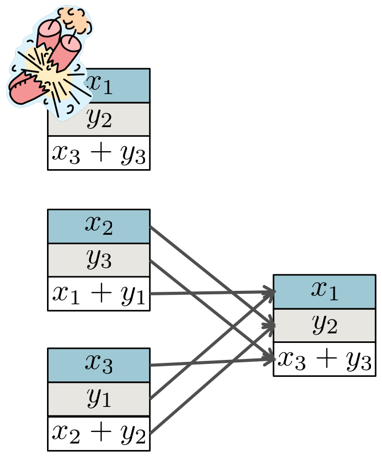

Figure 4: We show an example of a failed node repair. The repair locality here is 2 since 2 remaining nodes are involved in reconstructing the lost information of the first node, or the first coded symbol. Observe that we repair by only transferring blocks: no block combinations are need to be performed at the sender nodes. Once the blocks are transferred to a newcomer, a simple XOR suffices for reconstruction.

4.4 Distance

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. In Fig. 5, we give an example for the code of Fig. 3.

4.5 Repair bandwidth

In this subsection, we compare the repair bandwidth of the codes presented above, to the repair-BW optimal code, i.e., regenerating codes. We see that although our codes are not optimized for the metric repair-BW, they achieve very good values, that in cases come close to the optimal bounds of [5].

Observe that in our construction, the repair bandwidth communicated for a single node failure is equal to r times the contents of a single node. That is,

$$\gamma_{\text{LRC}} = r \cdot \alpha_{\text{LRC}} = (1 + r)\frac{M}{k}$$

which is vanishing when k, n grow to infinity. From, the work of [5] we know that the optimal repair bandwidth bounds for the case of $d = n - k + 1$. Regenerating codes (RCs) are codes that achieve the optimal trade-off of [5] that has been proven to be exact for specific points, under the exact repair model. We would like to note that a major deference between RCs and our LRCs, is the fact that in RCs a lost node can be repaired by contacting multiple (sometimes exponentially many in n) subsets of remaining nodes; in our LRC constructions we only have *a single* subset of r nodes that can reconstruct the file. This appears to make a great difference in the achieved locality and repair bandwidth.

In Fig. 6, we show how our codes compare with the RC trade-off, for $n = 14 \cdot i, k = 10 \cdot i$ and $i = \{1, 10, 100, 10000\}$. We plot our LRC repair-BW for different values of $\epsilon = \frac{1}{r}$, for values of r varying

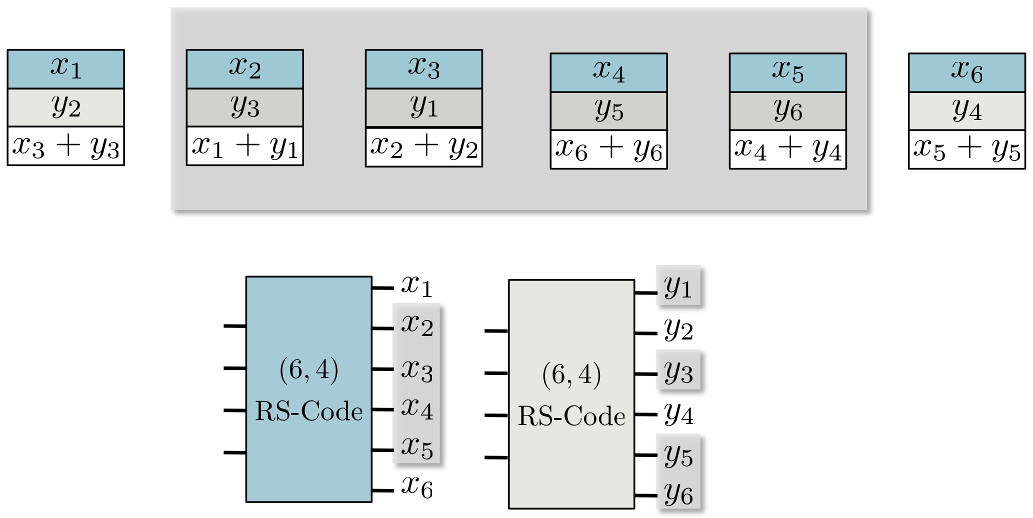

Figure 5: We show how the file can be reconstructed by contacting k nodes. Observe that by accessing k nodes, a DC has access to k blocks from the first MDS code and k blocks from the second. Since, the pre-codes are MDS, it means that any k blocks from each of the two coded blocks suffice to reconstruct both file parts.

from 1 to $k-1$, which correspond to different values of α, and hence rate. The RC repair-BW curve is parameterized by the number of nodes that are accessed during a repair. In Fig. 6, we slightly overload notation and refer to this as r (although RCs have minimum locality k). We set the RC r to its two extreme values, k and $n-1$, and then to an intermediate value of $\lceil \frac{k+n-1}{2} \rceil$. The reason why the curves start and stop at different values of the code rate, is due to the fact that they have different feasible operating points.

We see that as the coding parameters n and k grow larger, for a fixed ratio $\frac{k}{n}$, the repair BW performance of our codes come very close to the $r=k$ case of RCs. This is very interesting since for RCs there have been constructions only for the extremal points of the trade-off (MBR and MSR points). The intermediate points of trade-off are known to not exactly achievable, in general, and no achievability schemes are given for points that are slightly off the trade-off [22]. Therefore, LRCs add an interesting point in the repair-BW trade-off.

5 Conclusions

In this work, we have presented *locally repairable codes*, a new family of repair efficient codes that optimize the metric of locality. We analyze what is the best possible reliability of these codes, given the requirement that each coded symbol can be reconstructed by r other symbols in the code. We provide an information theoretic bound that ties together the code distance, the locality, and the storage cost of a code. We prove that this bound is achievable using vector linear codes. Eventually, we give an explicit construction of LRCs for when we require that any k nodes can recover the file. We show how this explicit construction not only has optimal locality, but also requires small field size, admits very simple XOR repairs, and achieves good repair bandwidth.

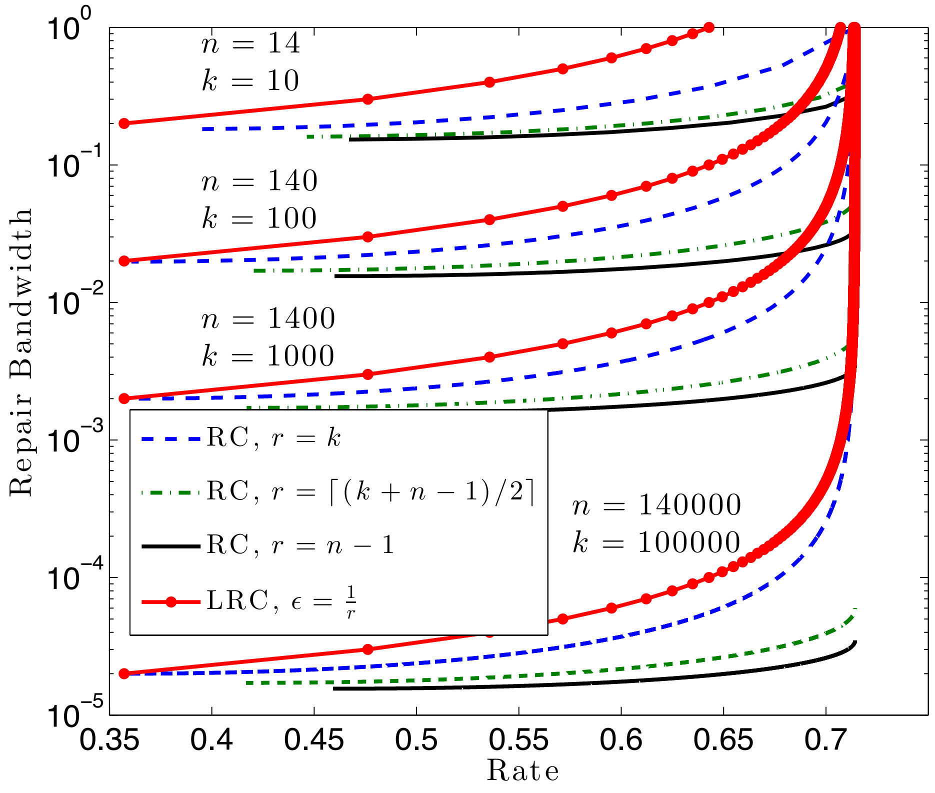

Figure 6: We plot the repair BW for a single node failure versus the rate of each code. We consider the LRCs that we construct in this section, and regenerating codes for $r = k, \left\lceil \frac{k+n-1}{2} \right\rceil, n-1$. On the LRC curves, each dot corresponds to a different point of locality, with $r = 1$ being the left most, and $r = k-1$ being the right most.

References

[1] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," in *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pp. 2771–2775, IEEE, 2012.

[2] H. Weatherspoon and J. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," *Peer-to-Peer Systems*, pp. 328–337, 2002.

[3] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in windows azure storage," in *USENIX Annual Technical Conference (USENIX ATC)*, 2012.

[4] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "XORing elephants: Novel erasure codes for big data," *Proceedings of the VLDB Endowment (to appear)*, 2013.

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

[6] K. V. 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," *Information Theory, IEEE Transactions on*, vol. 57, no. 8, pp. 5227–5239, 2011.

[7] C. Suh and K. Ramchandran, "Exact-repair mds code construction using interference alignment," *Information Theory, IEEE Transactions on*, vol. 57, no. 3, pp. 1425–1442, 2011.

[8] I. Tamo, Z. Wang, and J. Bruck, "Mds array codes with optimal rebuilding," in *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pp. 1240–1244, IEEE, 2011.

[9] V. R. Cadambe, C. Huang, S. A. Jafar, and J. Li, "Optimal repair of mds codes in distributed storage via subspace interference alignment," *arXiv preprint arXiv:1106.1250*, 2011.

[10] D. S. Papailiopoulos, A. G. Dimakis, and V. R. Cadambe, "Repair optimal erasure codes through hadamard designs," in *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, pp. 1382–1389, IEEE, 2011.

[11] O. Khan, R. Burns, J. Plank, and C. Huang, "In search of i/o-optimal recovery from disk failures," in *Proceedings of the 3rd USENIX conference on Hot topics in storage and file systems*, pp. 6–6, USENIX Association, 2011.

[12] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *Information Theory, IEEE Transactions on*, vol. 58, no. 11, pp. 6925–6934, 2011.

[13] F. Oggier and A. Datta, "Self-repairing homomorphic codes for distributed storage systems," in *INFOCOM, 2011 Proceedings IEEE*, pp. 1215–1223, IEEE, 2011.

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

[15] J. Han and L. A. Lastras-Montano, "Reliable memories with subline accesses," in *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, pp. 2531–2535, IEEE, 2007.

[16] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," in *Network Computing and Applications, 2007. NCA 2007. Sixth IEEE International Symposium on*, pp. 79–86, IEEE, 2007.

[17] N. Prakash, G. M. Kamath, V. Lalitha, and P. V. Kumar, "Optimal linear codes with a local-error-correction property," in *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pp. 2776–2780, IEEE, 2012.

[18] G. M. Kamath, N. Prakash, V. Lalitha, and P. V. Kumar, "Codes with local regeneration," *arXiv preprint arXiv:1211.1932*, 2012.

[19] A. Rawat, O. Koyluoglu, N. Silberstein, and S. Vishwanath, "Optimal locally repairable and secure codes for distributed storage systems," *arXiv preprint arXiv:1210.6954*, 2012.

[20] I. Tamo, D. S. Papailiopoulos, and A. G. Dimakis, "Optimal locally repairable codes and connections to matroid theory," *arXiv preprint arXiv:1301.7693*, 2013.

[21] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *Information Theory, IEEE Transactions on*, vol. 52, no. 10, pp. 4413–4430, 2006.

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

A. Proof of the distance bound

Proof. In this proof, we use similar proving techniques that can be found in [12]. Here, we aim to lower bound the cardinality of a set $\mathcal{S}$ that includes the minimum number of coded symbols whose entropy is less than the filesize M. This bound results in an upper bound on the minimum code distance d, since $d \leq n - |\mathcal{S}|$. We start by building a set $\mathcal{S}$ in steps and denote the collection of coded symbols at each step as $\mathcal{S}_i$. At each step i we denote the difference in cardinality of $\mathcal{S}_i$ ans $\mathcal{S}_{i-1}$ as

$$s_i = |\mathcal{S}_i| - |\mathcal{S}_{i-1}| \tag{3}$$

and the difference in entropy as

$$h_i = H(\mathcal{S}_i) - H(\mathcal{S}_{i-1}). \tag{4}$$

The algorithm that builds the set follows in Fig. 7.

```
step
1    Set $\mathcal{S}_0 = \emptyset$ and $i = 1$
2    WHILE $H(\mathcal{S}_{i-1}) < M$
3        Pick a coded block $Y_j \notin \mathcal{S}_{i-1}$
4        IF $H(\mathcal{S}_{i-1} \cup \{Y_{\Gamma(j)}\}) < M$
5            set $\mathcal{S}_i = \mathcal{S}_{i-1} \cup Y_{\Gamma(j)}$
6        ELSE IF $H(\mathcal{S}_{i-1} \cup \{Y_{\Gamma(j)}\}) \geq M$
7            pick $Y_S \subset Y_{\Gamma(j)}$ s.t. $H(Y_S \cup \mathcal{S}_{i-1}) < M$
8            set $\mathcal{S}_i = \mathcal{S}_{i-1} \cup Y_S$
9        $i = i + 1$
```

Figure 7: The algorithm that builds set $\mathcal{S}$

At each step (depending on the possibility that two $(r+1)$-groups overlap) the difference in cardinalities s_i is bounded as

$$1 \leq s_i \leq r + 1, \tag{5}$$

that is $s_i = r + 1 - p$, where $p = \left|\{Y_{\Gamma(j)}\} \cap \mathcal{S}_{i-1}\right|$. There exist two lines in the algorithm at which it can terminate. First, there is the case where the set $\mathcal{S}_l$ of the last step is generated by line 5. For this case we can also bound the entropy difference at each step as

$$h_i \leq (s_i - 1)\alpha, \tag{6}$$

which comes from the fact that, at least one coded variable in $\{Y_j\} \cup Y_{\mathcal{R}(j)}$ is a function of variables in $\mathcal{S}_{i-1} \cup Y_{\mathcal{R}(j)}$. Hence, we get the bound

$$\begin{aligned} |\mathcal{S}_l| &= \sum_{i=1}^{l} s_i \geq \sum_{i=1}^{l} \left(\frac{h_i}{\alpha} + 1\right) = l + \frac{1}{\alpha}\sum_{i=1}^{l} h_i \\ &= l + \frac{1}{\alpha}\left(\left\lceil \frac{M}{a} \right\rceil \alpha - \alpha\right) = \left\lceil \frac{M}{a} \right\rceil - 1 + l. \end{aligned} \tag{7}$$

We now have to bound l. The fastest (i.e. the smallest l) we can reach M is by collecting $r+1$ sets that have entropy $r\alpha$: if we collect $(r+1)$-groups, each having entropy $r+1$, we stop just before $\mathcal{S}_l$ has entropy M, that is

$$\begin{aligned} lr\alpha < M \Leftrightarrow l &< \frac{M}{r\alpha} \\ \Leftrightarrow l &= \left\lceil \frac{M}{r\alpha} \right\rceil - 1. \end{aligned} \tag{8}$$

Hence,

$$|\mathcal{S}_l| \geq \left\lceil \frac{M}{a} \right\rceil - 1 + l \geq \left\lceil \frac{M}{a} \right\rceil - 1 + \left\lceil \frac{M}{r\alpha} \right\rceil - 1, \tag{9}$$

in which case the distance of the code d is bounded as

$$d \leq n - \left\lceil \frac{M}{\alpha} \right\rceil - \left\lceil \frac{M}{r\alpha} \right\rceil + 2. \tag{10}$$

The second case, is the one where we reach line 7 of our algorithm. Depending on what the remainder of the division of M by $r\alpha$ is, we are left with some entropy that needs to be covered by at most $r-1$ additional nodes not in $\mathcal{S}_l$. The entropy not covered by the set $\mathcal{S}_l$ is

$$M - lr\alpha = M - \left\lceil \frac{M}{r\alpha} \right\rceil r\alpha - r\alpha. \tag{11}$$

To cover that we need an additional number of nodes that is equal to

$$s = \left\lceil \frac{M - lr\alpha}{\alpha} \right\rceil = \left\lceil \frac{M}{\alpha} \right\rceil - lr = \left\lceil \frac{M}{\alpha} \right\rceil - \left(\left\lceil \frac{M}{r\alpha} \right\rceil - 1 \right) r. \tag{12}$$

Hence, our final set $\mathcal{S}_l$ has size

$$\begin{aligned} |\mathcal{S}| &= |\mathcal{S}_l| + s - 1 \\ &= l(r+1) + s - 1 \\ &= \left(\left\lceil \frac{M}{r\alpha} \right\rceil - 1 \right)(r+1) + \left\lceil \frac{M}{\alpha} \right\rceil - \left(\left\lceil \frac{M}{r\alpha} \right\rceil - 1 \right) r - 1 \\ &= \left\lceil \frac{M}{r\alpha} \right\rceil (r+1) - r - 1 + \left\lceil \frac{M}{\alpha} \right\rceil - \left\lceil \frac{M}{r\alpha} \right\rceil r + r - 1 \\ &= \left\lceil \frac{M}{\alpha} \right\rceil + \left\lceil \frac{M}{r\alpha} \right\rceil - 2 \\ &= \left\lceil \frac{k}{1+\epsilon} \right\rceil + \left\lceil \frac{k}{r(1+\epsilon)} \right\rceil - 2. \end{aligned} \tag{13}$$

Again, due to the fact that the distance is bounded by $n - |\mathcal{S}|$ we have

$$d \leq n - \left\lceil \frac{M}{\alpha} \right\rceil - \left\lceil \frac{M}{r\alpha} \right\rceil + 2. \tag{14}$$

□

B. Proof of the min-cut of the flow-graph

Proof. It should be clear that the "worst" DC, i.e., the one that receives the least amount of flow, is a DC that connects to all $r+1$ nodes of as many $(r+1)$-groups as possible. In other words, the worst setup for a DC is to lose α flow per $r+1$ nodes that it connects to. Since the total number of nodes that a DC connects to is $n-d+1$ the number of groups it can entirely cover is

$$\left\lfloor \frac{n-d+1}{r+1} \right\rfloor \tag{15}$$

which will supply flow equal to $\left\lfloor \frac{n-d+1}{r+1} \right\rfloor r\alpha$. The remaining flow will be supplied by some other $n-d+1-(r+1)\left\lfloor \frac{n-d+1}{r+1} \right\rfloor$ nodes. Therefore, we have that the worst DC collects flow that is equal to

$$\left\lfloor \frac{n-d+1}{r+1} \right\rfloor r\alpha + \left(n-d+1-(r+1)\left\lfloor \frac{n-d+1}{r+1} \right\rfloor\right)\alpha, \tag{16}$$

hence, we obtain the following minimum cut bound

$$\begin{aligned}
&\min_i \min \text{cut}(\text{source}, \text{DC}_i) \\
&\geq \left\lfloor \frac{n-d+1}{r+1} \right\rfloor r\alpha + \left(n-d+1-(r+1)\left\lfloor \frac{n-d+1}{r+1} \right\rfloor\right)\alpha \\
&= \left(n-d+1-\left\lfloor \frac{n-d+1}{r+1} \right\rfloor\right)\alpha \\
&= \left(\left\lceil \frac{M}{\alpha} \right\rceil + \underbrace{\left\lceil \frac{M}{r\alpha} \right\rceil - 1 - \left\lfloor \frac{\left\lceil \frac{M}{\alpha} \right\rceil + \left\lceil \frac{M}{r\alpha} \right\rceil - 1}{r+1} \right\rfloor}_{A}\right)\alpha.
\end{aligned} \tag{17}$$

It suffices to show that $A \geq 0$, since $\left\lceil \frac{M}{\alpha} \right\rceil \alpha \geq M$. We proceed by checking if $A \geq 0$

$$\begin{aligned}
A = \left\lceil \frac{M}{r\alpha} \right\rceil - 1 - \left\lfloor \frac{\left\lceil \frac{M}{\alpha} \right\rceil + \left\lceil \frac{M}{r\alpha} \right\rceil - 1}{r+1} \right\rfloor &\geq 0 \\
\Leftrightarrow \left\lceil \frac{M}{r\alpha} \right\rceil - \left\lfloor \frac{\left\lceil \frac{M}{\alpha} \right\rceil + \left\lceil \frac{M}{r\alpha} \right\rceil - 1}{r+1} \right\rfloor &\geq 1 \\
\Leftrightarrow \left\lceil \left\lceil \frac{M}{r\alpha} \right\rceil - \frac{\left\lceil \frac{M}{\alpha} \right\rceil + \left\lceil \frac{M}{r\alpha} \right\rceil - 1}{r+1} \right\rceil &\geq 1 \\
\Leftrightarrow \left\lceil \frac{r}{r+1} \left\lceil \frac{M}{r\alpha} \right\rceil - \frac{\left\lceil \frac{M}{\alpha} \right\rceil - 1}{r+1} \right\rceil &\geq 1 \\
\Leftrightarrow \left\lceil \frac{r\left\lceil \frac{M}{r\alpha} \right\rceil - \left\lceil \frac{M}{\alpha} \right\rceil + 1}{r+1} \right\rceil &\geq 1,
\end{aligned} \tag{18}$$

which is true if and only if

$$\begin{aligned}
\frac{r\left\lceil \frac{M}{r\alpha} \right\rceil - \left\lceil \frac{M}{\alpha} \right\rceil + 1}{r+1} &> 0 \\
\Leftrightarrow r\left\lceil \frac{M}{r\alpha} \right\rceil > \left\lceil \frac{M}{\alpha} \right\rceil - 1 &= \left\lfloor \frac{M-1}{\alpha} \right\rfloor.
\end{aligned} \tag{19}$$

Observe that

$$r\left\lceil \frac{M}{r\alpha} \right\rceil \geq \frac{M}{\alpha} > \frac{M-1}{\alpha} \geq \left\lfloor \frac{M-1}{\alpha} \right\rfloor, \tag{20}$$

hence, indeed $A \geq 0$. Therefore, we obtain

$$\min_i \min \text{cut}(\text{s}, \text{DC}_i) \geq \left\lceil \frac{M}{\alpha} \right\rceil \alpha + A \geq \left\lceil \frac{M}{\alpha} \right\rceil \alpha \geq M, \tag{21}$$

that is the mincut is at least as much as the filesize. □

C. The $d = n - k + 1$ operational point

We first calculate the minimum storage overhead that allows the property that the file can be reconstructed by any k coded symbols. This overhead ϵ is the minimum one that satisfies the equation

$$\begin{aligned} n-k+1 &= n - \left\lceil \frac{M}{a} \right\rceil - \left\lceil \frac{M}{ra} \right\rceil + 2 \\ &= n - \left\lceil \frac{k}{1+\epsilon} \right\rceil - \left\lceil \frac{k}{r(1+\epsilon)} \right\rceil + 2. \end{aligned} \tag{22}$$

Therefore, the minimum storage overhead for erasure distance can be found through the following optimization

$$\begin{aligned} &\min \epsilon \\ \text{subject to: } &\left\lceil \frac{k}{1+\epsilon} \right\rceil + \left\lceil \frac{k}{r(1+\epsilon)} \right\rceil = k+1 \\ &\epsilon \geq 0. \end{aligned} \tag{23}$$

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 the range of ϵ values that satisfy the equation

$$n-k+1 = n - \left\lceil \frac{k}{1+\epsilon} \right\rceil - \left\lceil \frac{k}{r(1+\epsilon)} \right\rceil + 2. \tag{24}$$

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. We start with (24) and obtain

$$\begin{aligned} n-k+1 &= n - \left\lceil \frac{k}{1+\epsilon} \right\rceil - \left\lceil \frac{k}{r(1+\epsilon)} \right\rceil + 2 \\ \Leftrightarrow \left\lceil \frac{k}{1+\epsilon} \right\rceil + \left\lceil \frac{k}{r(1+\epsilon)} \right\rceil &= k+1. \end{aligned} \tag{25}$$

We use the following fact for two real numbers x and y

$$\lceil x+y \rceil \leq \lceil x \rceil + \lceil y \rceil \leq \lceil x+y \rceil + 1, \tag{26}$$

which implies that

$$\lceil x \rceil + \lceil y \rceil \in \{\lceil x+y \rceil, \lceil x+y \rceil + 1\}, \tag{27}$$

since $\lceil x+y \rceil$ and $\lceil x+y \rceil + 1$ are two consecutive integers. Due to (26) and (27) we have that

$$\left\lceil \frac{k}{1+\epsilon} \right\rceil + \left\lceil \frac{k}{r(1+\epsilon)} \right\rceil \in \left\{ \left\lceil \frac{r+1}{r} \frac{k}{1+\epsilon} \right\rceil, \left\lceil \frac{r+1}{r} \frac{k}{1+\epsilon} \right\rceil + 1 \right\}, \tag{28}$$

depending on k and r and ϵ. Then, we consider two ranges of values ϵ that satisfy the following two equations

$$\left\lceil \frac{r+1}{r} \frac{k}{1+\epsilon_1} \right\rceil = k+1 \tag{29}$$

$$\text{and } \left\lceil \frac{r+1}{r} \frac{k}{1+\epsilon_2} \right\rceil + 1 = k+1. \tag{30}$$

We start with (29) and obtain

$$\left\lceil \frac{r+1}{r} \frac{k}{1+\epsilon_1} \right\rceil = k+1 \Leftrightarrow \left\lceil \frac{(r+1)k}{r(1+\epsilon_1)} - k \right\rceil = 1$$
$$\Leftrightarrow \left\lceil \frac{(r+1)k}{r(1+\epsilon_1)} - \frac{r(1+\epsilon_1)k}{r(1+\epsilon_1)} \right\rceil = 1,$$

which is equivalent to

$$\begin{aligned} & 0 < \frac{(r+1)k}{r(1+\epsilon_1)} - \frac{r(1+\epsilon_1)k}{r(1+\epsilon_1)} \leq 1 \\ \Leftrightarrow & 0 < \frac{(r+1)k - r(1+\epsilon_1)k}{r(1+\epsilon_1)} \leq 1 \\ \Leftrightarrow & 0 < \frac{rk + k - rk - r\epsilon_1 k}{r(1+\epsilon_1)} \leq 1 \\ \Leftrightarrow & 0 < \frac{k - kr\epsilon_1}{r(1+\epsilon_1)} \leq 1 \\ \Leftrightarrow & 0 \overset{(a)}{<} k - kr\epsilon_1 \overset{(b)}{\leq} r(1+\epsilon_1). \end{aligned}$$

By (a) we obtain that $\epsilon_1 < \frac{1}{r}$ and by (b) we get

$$\begin{aligned} & k - kr\epsilon_1 \leq r(1+\epsilon_1) \Leftrightarrow k - r \leq kr\epsilon_1 + r\epsilon_1 \\ \Leftrightarrow & k - r \leq \epsilon_1 r(k+1)\epsilon_1 \\ & \Leftrightarrow \epsilon_1 \geq \frac{k-r}{r(k+1)} = \frac{k-r}{r(k+1)} = \frac{1}{r} - \frac{r+1}{r} \frac{1}{k+1}. \end{aligned}$$

Hence, the range of ϵ_1 is

$$\epsilon_1 \in \left[\frac{1}{r} - \frac{r+1}{r} \frac{1}{k+1}, \frac{1}{r} \right). \tag{31}$$

We continue with (30)

$$\left\lceil \frac{r+1}{r} \frac{k}{1+\epsilon_2} \right\rceil = k \Leftrightarrow \left\lceil \frac{(r+1)k}{r(1+\epsilon_2)} - \frac{r(1+\epsilon_2)k}{r(1+\epsilon_2)} \right\rceil = 0,$$

which is equivalent to

$$\begin{aligned}
& -1 < \frac{(r+1)k}{r(1+\epsilon_2)} - \frac{r(1+\epsilon_2)k}{r(1+\epsilon_2)} \leq 0 \\
\Leftrightarrow & -1 < \frac{(r+1)k - r(1+\epsilon_2)k}{r(1+\epsilon_2)} \leq 0 \\
\Leftrightarrow & -1 < \frac{rk + k - rk - r\epsilon_2 k}{r(1+\epsilon_2)} \leq 0 \\
\Leftrightarrow & -1 < \frac{k - kr\epsilon_2}{r(1+\epsilon_2)} \leq 0 \\
\Leftrightarrow & -r(1+\epsilon_2) \overset{(c)}{<} k - kr\epsilon_2 \overset{(d)}{\leq} 0.
\end{aligned}$$

By (c) we obtain

$$\begin{aligned}
& k - kr\epsilon_2 > -r(1+\epsilon_2) \Leftrightarrow kr\epsilon_2 - r\epsilon_2 < k + r \\
\Leftrightarrow & \epsilon_2 < \frac{k+r}{r(k-1)} = \frac{1}{r} + \frac{r+1}{r}\frac{1}{k-1}
\end{aligned}$$

and by (d) we get $\epsilon_2 \geq \frac{1}{r}$. Hence, (29) yields

$$\epsilon_2 \in \left[\frac{1}{r}, \frac{1}{r} + \frac{r+1}{r}\frac{1}{k-1}\right). \tag{32}$$

Observe that if we set $\epsilon = \frac{1}{r}$, the code can tolerate $n-k+1$ failures when $(r+1) \nmid k$ and at most $n-k+2$ failures, when $(r+1)|k$, i.e.,

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

and

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

that is $\epsilon = \frac{1}{r}$, is always a feasible solution for erasure distance. Hence, the optimal value of ϵ lies somewhere in the range

$$\epsilon_{\min} \in \left[\frac{1}{r} - \frac{r+1}{r}\frac{1}{k+1}, \frac{1}{r}\right]. \tag{33}$$

$\square$

Remark 4. *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.*