Bandwidth Adaptive & Error Resilient Regenerating Codes

Kaveh Mahdaviani, Ashish Khisti
ECE Dept., University of Toronto
Toronto, ON M5S3G4, Canada
Email: {kaveh, akhisti}@comm.utoronto.ca

Soheil Mohajer
ECE Dept., University of Minnesota
Minneapolis, MN 55404, USA
Email: soheil@umn.edu

***Abstract*—To be considered for the 2016 IEEE Jack Keil Wolf ISIT Student Paper Award. Regenerating codes are efficient methods for distributed storage in practical networks where node failures are common. They guarantee low cost data reconstruction and repair through accessing only a predefined number of arbitrary chosen storage nodes in the network. In this work we study the fundamental limits of required total repair bandwidth and the storage capacity of these codes under the assumption that *i*) both data reconstruction and repair are resilient to the presence of a certain number of erroneous nodes in the network and *ii*) the number of helper nodes in every repair is not fixed, but is a flexible parameter that can be selected during the run-time. We focus on the minimum repair bandwidth point in this work, propose the associated coding scheme to posses both these extra properties, and prove its optimality.**

I. Introduction

The urge for efficient, flexible and robust distributed storage systems is now not a surprise any more. Everyday, huge quantities of data are being produced and converted to the digital format from scientific, commercial or entertainment industry sources. Having to deal with "big data" in many applications has driven several traditional methods of handling data, such as storing the whole data on a single disk where it is needed to be processed, impractical. Let alone backing up and sharing by duplication. Hence, the demand for distributed storage systems, as a natural solution to these basic problems associated with the huge data, is inevitable.

As the number of storage devices increase in the distributed storage solution, the frequency and likeliness of storage failure also scales, which turns partial data loss into a norm rather than a rare event. As a result, in order to avoid loosing the source data in distributed storage systems, storing redundant data have been considered. More precisely, a distributed storage system needs to store the original data such that it is prepared for losing a part of the stored data and yet being able to recover the whole original data.

An important feature of a distributed storage system is its ability to *repair* itself after a node, or part of the stored data, either temporary or permanently, becomes unavailable. In the absence of any repair mechanism, as storage nodes become unavailable in a distributed storage system over time, the available stored segments of data reduces and eventually the remaining stored data will not be sufficient for reconstructing the original data any more. Therefore, it is natural to consider a mechanism to replace the failed storage nodes by new nodes, and store appropriate data on the replacement nodes such that the whole system maintain its functionality over time. Such a procedure is referred to as the repair process.

Dimakis *et. al.* [1] formulated the problem of code design for distributed storage of a large data in a network of n nodes each of *per node storage capacity* α. In their formulation the repair procedure should be performed by only accessing a subset of size d of the available storage nodes, namely *helpers*, and downloading only β *repair data* from each of them. Moreover, the original source data store in the system is required to be reconstructible through accessing the data stored only on a subset of size $k \leq d$ of the nodes. Such family of codes is hence named *regenerating* codes. They showed that there is a trade-off between the per node storage α and the amount of data required for repair, namely the *total repair bandwidth* denoted by $\gamma = d\beta$. Regenerating codes realizing the two extreme points of this trade-off which refer to the minimum repair bandwidth and minimum per node storage are named as the MBR and MSR codes.

Since then a significant interest has been attracted to the design and analysis of regenerating codes. In particular a specific class of regenerating codes named the *exact repair regenerating codes*, which are capable of performing the repair process such that the data stored in the replacement node is exactly the same as the data that was stored in the failed node. Exact repair regenerating codes are practically much more appealing, e.g., for having invariant encoding and decoding mechanisms and more importantly capability to be tuned as systematic codes. A regenerating code which is not capable of exact repair is referred to as *functional repair*.

In this work we consider two simultaneous extensions to the original work of Dimakis *et. al.* [1]. One extra feature that we consider in our setup is that the number of helpers chosen for repair can be adaptively selected, which allows for run-time optimization according to the dynamic state of the system. We refer to this property as *bandwidth adaptivity*. Such flexibility adds a notable robustness to distributed storage systems with time-varying conditions such as peer-to-peer distributed storage systems where storage nodes join and leave the system frequently. It is also an important feature for systems with significant load unbalance or heterogeneous connectivity where the number of available nodes for repair

vary for different repair cases.

While the importance of this feature has been recently addressed by other researchers [2]–[4], no practical exact repair coding scheme is yet known to be introduced for such a setup. In [2] a coding scheme is proposed which performs exact repair with bandwidth adaptivity based on interference alignment for the MSR mode. However the optimality of this scheme is achievable in asymptotically large per node storage capacity and repair bandwidth. The other results have only considered the functional repair, while [3] investigates the gain of coordination in simultaneous repair of multiple failures, and [4] addresses the upper bound on the storage capacity and gain in *Mean Time to Lose Data* in presence of bandwidth adaptivity.

Besides the data loss due to storage node failure, one other practical issue that affects the reliability of data recovery in distributed storage systems is the presence of errors. In our model we also encounter the presence of error in the system by adopting a limited power adversarial intruder model, which can compromise up to a fixed number $b < k/2$ of nodes as described in [5], [6]. Such intruder is considered to be omniscient, i.e. knows the original data stored in the system and the coding scheme, and can control the data stored in, and being transferred by no more than b nodes under his control.

In [5] an upper bound for the capacity of distributed storage systems is presented in the presence of different intruders including the limited power omniscient intruder described above. Exact repair coding schemes are also proposed for MBR and MSR modes in [5], [6] to achieve this upper bound. However, none of these results consider bandwidth adaptivity.

In this work we focus on the natural extension of MBR mode with bandwidth adaptivity and error resiliency, and present a coding scheme which we show is optimal. To the best of our knowledge, this work is the first non-asymptotic exact repair code construction for such a setting. The main contributions of this work are explained in Section III, after formally defining the setup in the next section.

II. Model

In this section we will briefly introduce a setup for the distributed storage system and the coding scheme of our interest. This model is a modified version of the original setup considered in [1].

A Bandwidth adaptive and error resilient (BAER) distributed storage system has a predefined Galois filed alphabet, $\mathbb{F}_q$ of size q, such that hereafter in this work we assume all the symbols stored or transmitted through the network are elements of $\mathbb{F}_q$.

Definition 1 (BAER Regenerating Code)**.** *Consider the set of parameters α, n, $d_{\min}$, $d_{\max}$, k, b, and a* total repair bandwidth function $\gamma : \{d_{\min}, \cdots, d_{\max}\} \to [\alpha, \infty)$. *A BAER regenerating code $\mathcal{C}(\alpha, \gamma(\cdot), n, d_{\min}, d_{\max}, k, b)$ is a regenerating code with per node storage capacity α, which is capable of performing any sequence of repair and data reconstruction processes in arbitrary order when up to b out of n nodes are allowed to be compromised and provide adversarial data in each process. Moreover, in any repair process the number of helpers, d, can be chosen arbitrarily such that $d_{\min} \leq d \leq d_{\max}$. The choice of helper nodes is also arbitrary and each of the chosen helpers then provide $\gamma(d)/d$ repair symbols. Similarly, in any download process the data collector accesses any arbitrary set of k nodes and downloads α symbols from each.*

Remark 1. *Note that when erroneous nodes are present in the system the repair process should prevent the propagation of the errors. In the other words, the repair of a non-compromised node should replace that with a node that stores non-compromised data, which is the data that the coding scheme would have stored in the replacement node if there exists no compromised in the whole network at all.*

Definition 2 (Storage Capacity, Optimal Codes & Modes)**.** *For the set of parameters α, n, $d_{\min}$, $d_{\max}$, k, b, and a given function $\gamma : \{d_{\min}, \cdots, d_{\max}\} \to [\alpha, \infty)$, the* storage capacity *of a BAER distributed storage system is the maximum size of the file that could be stored in a network of n storage nodes with per node storage capacity α, using a BAER regenerating code $\mathcal{C}(\alpha, \gamma(\cdot), n, d_{\min}, d_{\max}, k, b)$. We will denote the storage capacity of such a system by $F(\alpha, \gamma(\cdot), n, d_{\min}, d_{\max}, k, b)$, or simply F whenever the parameters of the system could be inferred from the context. F is then the size of the largest file that could be stored in the distributed storage systems with the associated parameters. Moreover, for any choice of parameters, the BAER codes that realize the storage capacity are referred to as an* optimal BAER codes*, and the pair of $(F, \gamma(\cdot))$ are referred to as a mode.*

Since there are many parameters involved in the presented setting, to focus on the trade-off between the per node storage α, total repair bandwidth function $\gamma(\cdot)$, error resiliency b, and total storage capacity F, hereafter we will consider α, n, $d_{\min}$, $d_{\max}$, k, b to be fixed and mainly focus on exploring the tension between F and $\gamma(\cdot)$.

Remark 2. *It is obvious that scaling the per node storage capacity α and we can always scale the total repair bandwidth function and the storage capacity with the same factor. Moreover, if part of the data stored in a single node is a function of the rest of the data stored on the same node (per node redundancy), it is always possible to remove the redundant part and scale everything to increase the storage capacity in the system. Hence, through this work we always assume the data stored in each node does not include per node redundancy.*

Definition 3 (MBR Mode for BAER Regenerating Codes)**.** *Assuming that there is no per node redundancy in the distributed storage system, the MBR BAER mode is the pair of $(F_{MBR}, \gamma_{MBR}(\cdot))$, such that $\gamma_{MBR}(d)$ is the minimum possible $\gamma(d)$ for all values of $d_{\min} \leq d \leq d_{\max}$, for which the storage capacity F is positive.*

Definition 4 ($\Gamma(F)$)**.** *For any specific choice of parameters*

α, n, $d_{\min}$, $d_{\max}$, k, b, there may exist a family of total repair bandwidth functions which all have the same storage capacity F. We use the notation $\Gamma(F)$, to refer to the set of all functions $\gamma : \{d_{\min}, \cdots, d_{\max}\} \to [\alpha, \infty)$, such that $F(\alpha, \gamma(\cdot), n, d_{\min}, d_{\max}, k, b) = F$.

In another perspective to the BAER setup with a given set of parameters α, n, $d_{\min}$, $d_{\max}$, k, b, one might fix storage capacity F, and search among the feasible set of total repair bandwidth functions $\gamma(\cdot)$, i.e., $\Gamma(F)$ for the best choices. While optimal BAER codes correspond to the Pareto optimal functions $\gamma(\cdot) \in \Gamma(F)$, in this work we consider the strongest definition for optimality as will be introduced in the following definitions. Surprisingly, we will show that such strong optimality is achievable for the MBR BAER mode.

Definition 5 (Universally Optimal BAER Regenerating Code)**.** *Let $\gamma_F^*(\cdot)$, be the point-wise minimum of all the $\gamma(\cdot)$ functions in $\Gamma(F)$ defined as*

$$\gamma_F^*(d) = \min_{\gamma(\cdot)\in\Gamma(F)} \gamma(d), \quad \forall d,\ d_{\min} \le d \le d_{\max}.$$

For any storage capacity F, and fixed set of parameters α, n, $d_{\min}$, $d_{\max}$, k, b the universally optimal *BAER regenerating code is the optimal BAER code which has the minimal total repair bandwidth function $\gamma_F^*(\cdot)$.*

Note that the this definition does not guarantee the existence of universally optimal BEAR codes, since the $\gamma_F^*(\cdot)$ might not be simultaneously achievable by a single BEAR code for all feasible values of d.

III. Main Result

Here we briefly summarize the results and main contributions. In this work we focus on the MBR mode. Considering the set of parameters α, n, $d_{\min}$, $d_{\max}$, k, b, we first characterize the minimum total repair bandwidth such that the storage capacity is positive, and hence define the MBR mode for the described setting. The following theorem describes this result.

Theorem 1. *Consider a BEAR distributed storage system with parameters α, n, $d_{\min}$, $d_{\max}$, k, b, such that $d_{\min} \ge k > 2b$. Assuming zero per node redundancy, the minimal total repair bandwidth function is given by*

$$\gamma_{MBR}(d) = \frac{\alpha d}{d-2b}, \quad \forall d,\ d_{\min} \le d \le d_{\max}. \tag{1}$$

We also determine the total storage capacity of a BAER distributed storage system in the MBR mode, and provide an upper bound for the total storage capacity of the general mode. We summarize these results in the following theorem.

Theorem 2. *The storage capacity of a BAER distributed storage system in the MBR mode, i.e. with the total repair bandwidth function $\gamma_{MBR}(\cdot)$ as introduced in Theorem 1, is given by*

$$\begin{aligned} F_{MBR} &= \sum_{j=2b}^{k-1} (d_{\min}-j)\frac{\alpha}{(d_{\min}-2b)} \\ &= \frac{\alpha}{d_{\min}-2b}(k-2b)\left(d_{\min}-b-\frac{k-1}{2}\right). \end{aligned} \tag{2}$$

Moreover, let $D = \{d_{\min}, \cdots, d_{\max}\}$. The following upper bound holds for the total storage capacity of any BAER distributed storage system.

$$F \le \sum_{j=0}^{k-2b-1} \min\left(\alpha, \min_{d\in D}\left((d-2b-j)\frac{\gamma(d)}{d-2b}\right)\right).$$

Note that the results in the above theorems holds for the exact repair as well as for the functional repair.

Finally, another main result we introduce in this work is to present the first non-MSR exact repair bandwidth adaptive regenerating code. More specifically, we show that universally optimal exact repair BAER code exists for the MBR mode by providing an explicit code construction. This coding scheme is the first MBR exact repair bandwidth adaptive regenerating code. The only other bandwidth adaptive exact repair regenerating code construction presented so far is the MSR code presented in [2]. The MBR code construction presented in this work also provides error resiliency.

IV. Coding Scheme

In this section we introduce an MBR BAER regenerating code $\mathcal{C}(\alpha, \gamma(\cdot), n, d_{\min}, d_{\max}, k, b)$, for $k, d_{\min} > 2b$, and

$$\alpha = \prod_{d=d_{\min}}^{d_{\max}} (d-2b),$$

which achieves the total storage capacity F_{MBR} of (2). We will also show that the presented coding scheme is universally optimal for MBR mode.

In order to achieve a BAER MBR regenerating code, we start from an exact repair MBR regenerating code construction introduced by Rashmi *et. al.* named *Product Matrix* MBR codes [7]. Indeed, our code construction could be considered as a generalization of the MBR Product Matrix codes in which we use the Product Matrix codes as basic components. For the rest of this section let $\vec{s} = (s_1, \cdots, s_{F_{\text{MBR}}})$ denote the source data symbols, where F_{MBR} is given in (2).

A. Encoding for Storage

Let $\underline{d} = d_{\min} - 2b$, and $\underline{k} = k - 2b$, and also let O be a $\underline{d} \times \underline{d}$ zero matrix. The first step in the encoding process is to arrange the source data symbols in the form of an *data matrix* $M_{\alpha\times\underline{d}}$, which is a block diagonal matrix consisting of

$z = \alpha/(\underline{d})$ submatrices $M_1, \cdots, M_z$ as the diagonal blocks in the following form.

$$M = \begin{bmatrix} M_1 & O & \cdots & O \\ O & M_2 & \cdots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & \cdots & O & M_z \end{bmatrix}. \tag{3}$$

Moreover each of the submatrices $M_i,\ i \in \{1, \cdots, z\}$ is a symmetric matrix satisfying the structural properties of a Product Matrix MBR code for parameters $k = \underline{k}$, and $d = \underline{d}$. In other words

$$M_i = \begin{bmatrix} N_i & L_i \\ L_i^\intercal & O' \end{bmatrix}, \quad i \in \{1, \cdots, z\},$$

where N_i is a symmetric $\underline{k} \times \underline{k}$ matrix, and L_i is an arbitrary $\underline{k} \times (d_{\min} - k)$ matrix, and finally O' is a $(\underline{d} - \underline{k}) \times (\underline{d} - \underline{k})$ zero matrix.

Similar to the original Product Matrix MBR codes the encoding for storage over each node is performed using a *coefficient vector*. The coefficient vector in our construction is constructed based on a Vandermonde matrix $\Psi_{zn \times \underline{d}}$,

$$\Psi = \begin{bmatrix} 1 & e_1 & e_1^2 & \cdots & e_1^{(\underline{d}-1)} \\ 1 & e_2 & e_2^2 & \cdots & e_2^{(\underline{d}-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e_{zn} & e_{zn}^2 & \cdots & e_{zn}^{(\underline{d}-1)} \end{bmatrix}, \tag{4}$$

where, $e_i,\ i \in \{1, \cdots, z\}$ are distinct non-zero elements of $\mathbb{F}_q$. In particular, denoting the j^{th} row of Ψ by $\vec{\psi}_j,\ 1 \le j \le zn$, we define the vector of encoded symbols to be stored on node $\ell \in \{1, \cdots, n\}$, denoted by $\vec{x}_\ell$ as follows;

$$\vec{x}_\ell = (\vec{\psi}_{(\ell-1)z+1}, \cdots, \vec{\psi}_{\ell z})M = (\vec{\psi}_{(\ell-1)z+1}M_1, \cdots, \vec{\psi}_{\ell z}M_z).$$

Note that $\vec{x}_\ell$ is an α dimensional vector and hence the per node storage capacity is satisfied.

B. Encoding for Repair and Reconstruction

Repair: In order to perform the encoding for a repair process, we require to have a matrix $\Omega \in \mathbb{F}_q^{z \times z}$, where columns of Ω are rows a Vandermonde matrix as bellow,

$$\Omega = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ e'_1 & e'_2 & \cdots & e'_z \\ e'^2_1 & e'^2_2 & \cdots & e'^2_z \\ \vdots & \vdots & \ddots & \vdots \\ e'^{(z-1)}_1 & e'^{(z-1)}_2 & \cdots & e'^{(z-1)}_z \end{bmatrix}, \tag{5}$$

where, $e'_i,\ i \in \{1, \cdots, z\}$ are distinct non-zero elements of $\mathbb{F}_q$. We assume the choice of matrix Ω, is known to all the nodes in the network. Moreover we will represent the i^{th} row of this matrix by $\vec{\omega}_i$. This matrix Ω will be used to adjust the dimension of the vector of repair symbols provided by each helper based on the selected parameter d.

When a node f in the network fails, we can choose any number d of other nodes as helpers such that $2b < k \le d_{\min} \le d \le d_{\max}$. To describe the encoding process for repair at helper node j_ℓ, we define the following notations.

$$\Omega_d = \begin{bmatrix} \vec{\omega}_1 \\ \vdots \\ \vec{\omega}_{\alpha/(d-2b)} \end{bmatrix}.$$

Note that $\Omega_d^\intercal,\ \forall d \in \{1, \cdots, z\}$ is then also a Vandermonde matrix. Moreover, for any node $i \in \{1, \cdots, n\}$ in the network, the matrix Φ_i is a $\alpha \times z$ block-diagonal matrix with $\underline{d} \times 1$ diagonal blocks $\vec{\psi}_{(i-1)z+1}^\intercal, \cdots, \vec{\psi}_{iz}^\intercal$.

Each helper node $j_\ell,\ \ell \in \{1, \cdots, d\}$ then produces a repair vector $\vec{r}_{j_\ell, f}$ for the replacement node f as

$$\vec{r}_{j_\ell, f} = \vec{x}_{j_\ell}\Phi_f\Omega_d^\intercal. \tag{6}$$

Note that this will be a row vector of length $\alpha/(d-2b)$.

Reconstruction: In the data reconstruction, there is no encoding and the data collector just downloads the stored data $\vec{x}_{j_\ell}$, for $\ell \in \{1, \cdots, k\}$, where $j_\ell,\ \ell \in \{1, \cdots, k\}$ is the index of the ℓ^{th} selected node.

C. Decoding for Repair and Reconstruction

The decoding procedure for both data reconstruction as well as the repair is performed using a scheme which will be referred to as the *"Test-group decoding"*. This decoding scheme enables the decoder to both recover the required data, and simultaneously authenticate the ingenuity of the recovered message. In this subsection we first describe the Test-group decoding for the repair and reconstruction processes and then prove that this decoder never fails given that the omniscient intruder could not compromise more than b nodes.

Repair: The Test-group decoding is an iterative procedure. In the repair process, the decoding algorithm receives the index of the failed node f, the chosen parameter d, the set of selected helper nodes $\{j_1, \cdots, j_d\}$ of size d, and the their provided repair data $\vec{r}_{j_\ell},\ \ell \in \{1, \cdots, d\}$, as the input. Each iteration then starts by selecting one of the $\binom{d}{d-b}$ subsets of the helper nodes of size $d-b$, which has not been used in the previous iterations. We will refer to the selected subset at the current iteration as the selected *test-group*, and denote it by $\mathcal{T}$. Once the test-group is selected the decoder calculates one *guess* for the lost data based every subset of size $d-2b$ of the test-group.

To describe the process of calculating the guesses, let's denote one arbitrary subset $\mathcal{H} \subset \mathcal{T}$ of size $d-2b$, and its corresponding guess as $\vec{\hat{x}}_{\mathcal{H},f}$. The decoder first creates a vector $\vec{\rho}_{\mathcal{H},f}$ by concatenating all the $\vec{r}_{i_\ell,f}$ row vectors for all $i_\ell \in \mathcal{H}$, as defined in (6), based on a predefined order. Hence, $\vec{\rho}_{\mathcal{H},f}$ will be a row vector of dimension α. Now note that for each $i_\ell \in \mathcal{H}$, assuming it is not a compromised node, we have

$$\begin{aligned} \vec{r}_{i_\ell,f} &= \left[\vec{\psi}_{(i_\ell-1)z+1}M_1\vec{\psi}_{(f-1)z+1}^\intercal, \cdots, \vec{\psi}_{i_\ell z}M_z\vec{\psi}_{fz}^\intercal\right]\Omega_d^\intercal \\ &= \left[\vec{\psi}_{(f-1)z+1}M_1\vec{\psi}_{(i_\ell-1)z+1}^\intercal, \cdots, \vec{\psi}_{fz}M_z\vec{\psi}_{i_\ell z}^\intercal\right]\Omega_d^\intercal, \end{aligned}$$

and hence, assuming all nodes in $\mathcal{H}$ are providing genuine repair data, we have

$$\vec{\rho}_{\mathcal{H},f} = \vec{x}_f\left[\Phi_{i_1}\Omega_d^\intercal, \cdots, \Phi_{i_{d-2b}}\Omega_d^\intercal\right].$$

Therefore defining

$$\Theta_{\mathcal{H}} = \left[\Phi_{i_1}\Omega_d^{\intercal}, \cdots, \Phi_{i_{d-2b}}\Omega_d^{\intercal}\right] = \left[\Phi_{i_1}, \cdots, \Phi_{i_{d-2b}}\right]\left(\Omega_d^{\intercal} \otimes I_{(d-2b)}\right), \quad (7)$$

where $I_{(d-2b)}$ is the identity matrix of size $(d-2b)$, and $\otimes$ represents the Kronecker product.

Assuming all the repair data provided by the helper nodes in $\mathcal{H}$ is genuine we have

$$\vec{x}_f = \vec{\rho}_{\mathcal{H},f}\Theta_{\mathcal{H}}^{(-1)},$$

given that the matrix $\Theta_{\mathcal{H}}^{(-1)}$ exists. As a result, the decoder will be able to produce a guess $\vec{\hat{x}}_{\mathcal{H},f}$ for any subset of size $d-2b$ of the Test-group, $\mathcal{H} \subset \mathcal{T}$ if the corresponding matrix $\Theta_{\mathcal{H}}$ is nun-singular. The following lemma guarantees that this condition always holds and the Test-group decoder is able to produce the guesses.

Lemma 1. *For any parameter $d_{\min} \leq d \leq d_{\max}$, and any subset $\mathcal{H} \subset \mathcal{T}$ of size $d-2b$ of a Test-group $\mathcal{T}$, consisting of $d-b$ helpers, the matrix $Theta_{\mathcal{H}}$, defined in (7), is invertible.*

For the proof of this lemma please see the Appendix A.

Once all the guesses are calculated for a test-group, the decoder proceeds by checking the consistency of the guesses. In other words each guess $\vec{\hat{x}}_{\mathcal{H},f}$ will be equal to the lost data $\vec{x}_f$, if the corresponding subset of helpers, $\mathcal{H}$, are all providing genuine repair data, and hence, all the guesses in a test-group $\mathcal{T}$ will be the same if all the helpers in $\mathcal{T}$ are providing genuine repair data. The decoder, then will stop whenever it finds a consistent test-group and outputs the consistent guess as the decoded data. The following lemma guarantees this procedure will always succeed.

Lemma 2. *Assuming the maximum number of compromised nodes is b, if all the guesses produced in the Test-group decoding for a test-group $\mathcal{T}$ is consistent, then all of them are correct. Moreover, the Test-group decoding will always find a consistent test-group.*

Proof. First note that any test-group $\mathcal{T}$, consists of $d-b$ nodes and the decoder produces a guess based on any subset of $\mathcal{H}$ of size $d-2b$ in $\mathcal{T}$. Since the maximum number of compromised nodes is b, then at least one of the subsets in any test-group is guaranteed to be totally non-compromised. Therefore, at least one of the guesses in each test-group is genuine and if all the guesses in the test-group are consistent then all of them should be genuine.

Now to prove that the Test-group decoder always finds a consistent test-group, simply note that for any specific choice of b compromised nodes, there exists at least one subset of size $d-b$, in any set of d selected helpers, which does not contain any compromised node. Since the Test-group decoder checks all possible choices of test-groups, it always finds a consistent one. □

To summarize, the Test-group decoding for the repair is described bellow. It receives the set of d selected helpers, index of failed node f, and the maximum possible number of compromised nodes b, as its inputs.

Algorithm 1 Test-group decoding for repair

```
Consistency ← False
while ¬(Consistency) do
    T ← A new subset of helpers of size d − b
    for each subset H ⊂ T do
        Calculate ρ⃗_{H,f} and Θ_{H,f}
        x⃗̂_{H,f} ← ρ⃗_{H,f} Θ^{(−1)}_{H,f}
    end for
    if ∀H ⊂ T  x⃗̂_{H,f} vectors are the same then
        Consistency ← True
        Output ← x⃗̂_{H,f} for some H ⊂ consistent T
    end if
end while
```

Reconstruction: In the case of data reconstruction, again the Test-group decoder will iteratively select a test-group $\mathcal{T}$ of size $k-b$, which has not been used before. Then for any subset $\mathcal{H} \subset \mathcal{T}$ of size $k-2b$, the decoder calculates guesses $\hat{M}_{\mathcal{H}}$, and checks f all the calculated guesses match in the current test-group. If there is a non-consistency, the decoder ends this iteration and starts the next iteration by selecting a new test-group until it finds a consistent one. A guess in a consistent test-group will be considered as the output.

To describe the process of calculating guesses let's define the following notations for any subset of selected nodes $\mathcal{H} = \{i_1, \cdots, i_{k-2b}\}$.

$$X_{\mathcal{H}} = \begin{bmatrix} \vec{x}_{i_1} \\ \vdots \\ \vec{x}_{i_{k-2b}} \end{bmatrix}, \quad \Phi_{\mathcal{H}} = \begin{bmatrix} \vec{\psi}_{(i_1-1)z+1}, \cdots, \vec{\psi}_{i_1 z} \\ \vdots \\ \vec{\psi}_{(i_{k-2b}-1)z+1}, \cdots, \vec{\psi}_{i_{k-2b} z} \end{bmatrix}.$$

Again assuming the selected subset $\mathcal{H}$ provide genuine data we have,

$$X_{\mathcal{H}} = \Phi_{\mathcal{H}} M.$$

Therefore, to calculate a guess based on any selected subset $\mathcal{H} \subset \mathcal{T}$, we have,

$$\hat{M}_{\mathcal{H}} = \left(\Phi_{\mathcal{H}}^{\intercal}\Phi_{\mathcal{H}}\right)^{(-1)}\Phi_{\mathcal{H}}^{\intercal}X_{\mathcal{H}}.$$

A similar discussion as in Lemma 2 shows that the Test-group decoding will always find a consistent test-group in the data reconstruction and any guess in a consistent test-group is correct.

To summarize, the test group decoding for data reconstruction is given bellow. It receives the set of k selected nodes and the maximum number of compromised nodes b, as the input.

This ends the description of the coding scheme. Now based on the proposed coding scheme we conclude the following corollary.

Corollary 1. *For*

$$\gamma(d) = \frac{\alpha d}{d-2b}, \quad (8)$$

Algorithm 2 Test-group decoding for data reconstruction

Consistency $\leftarrow$ False
while $\neg$(Consistency) **do**
  $\mathcal{T} \leftarrow$ A new subset of helpers of size $k-b$
  for each subset $\mathcal{H} \subset \mathcal{T}$ **do**
    Calculate $X_{\mathcal{H}}$, and $\Phi_{\mathcal{H}}$
    $\hat{M}_{\mathcal{H}} = \left(\Phi_{\mathcal{H}}^{\intercal}\Phi_{\mathcal{H}}\right)^{(-1)}\Phi_{\mathcal{H}}^{\intercal}X_{\mathcal{H}}$
  end for
  if $\forall \mathcal{H} \subset \mathcal{T}$ $\hat{M}_{\mathcal{H}}$ vectors are the same **then**
    Consistency $\leftarrow$ True
    Output $\leftarrow \hat{M}_{\mathcal{H}}$ for some $\mathcal{H} \subset$ consistent $\mathcal{T}$
  end if
end while

the storage capacity for a BAER distributed storage system with parameters n, α, b, $d_{\min} \geq k > 2b$, $d_{\max}$, $\gamma(\cdot)$ is lower bounded as follows

$$F \geq \frac{\alpha(k-2b)}{d_{\min}-2b}\left(d_{\min}-b-\frac{(k-1)}{2}\right). \tag{9}$$

Proof. The proof for (8) follows directly from the definition of the repair procedure in the proposed coding scheme. The achievable storage capacity, F, of the proposed coding scheme is equal to the number of independent elements of matrix M. According to the structure of the matrix M in (3), this quantity is z times the total storage capacity of an MBR Product Matrix code with parameters $\underline{k}$, and $\underline{d}$. Hence we have

$$F = z\left(\frac{\underline{k}(\underline{k}+1)}{2}+\underline{k}(\underline{d}-\underline{k})\right) = \frac{\alpha\underline{k}}{\underline{d}}\left(\underline{d}-\frac{(\underline{k}-1)}{2}\right).$$

Replacing $\underline{d} = d_{\min}-2b$, and $\underline{k} = k-2b$, we get the lower bound in (9). □

V. An Upper Bound on the Capacity of BAER Codes

In the proofs of this section we will use a lemma proved in [5] for the conventional regenerating codes. We restate the lemma in the BAER setting below while the proof follows similarly as provided in [5].

Lemma 3. *In a BAER code $\mathcal{C}(\alpha, \gamma(\cdot), n, d_{\min}, d_{\max}, k, b)$, in any data reconstruction, the data provided by any subset of size $k-2b$ of the k selected nodes should be enough for uniquely decoding the source data stored in the system. Moreover, in any repair, the repair data provided by any subset of size $d-2b$ of the d selected helpers should be enough to uniquely decode the lost data.*

Proof. The proof is the same for both data reconstruction and repair and is through proof by contradiction. We will use the notation a to refer to either d or k for the repair and data reconstruction respectively. Consider a scenario (either a repair or reconstruction) in which a set of a nodes are selected in the network to provide data. Also, assume the message (either the source data stored in the network or the data to be stored in the replacement node) is m_1. Regardless of the scenario we can always assume the message to be recovered belongs to set $\mathcal{M}$ of all possible messages. Moreover, let's denote the data provided by any subset $\mathcal{L}$ of the selected set of nodes by $\vec{y}_{\mathcal{L}}(m)$, where $m \in \mathcal{M}$ is the message to be recovered. Now let us assume there exists a subset of selected nodes $\mathcal{L}^*$ such that $|\mathcal{L}^*| = a-2b$ and for some different messages $m_2 \in \mathcal{M}$ we have $\vec{y}_{\mathcal{L}^*}(m_1) = \vec{y}_{\mathcal{L}^*}(m_2)$. If an intruder has the control of a subset $\mathcal{L}'$ of size b among the remaining $2b$ nodes, and sets $\vec{y}_{\mathcal{L}'}(m_2)$ to be the provided data from nodes under his control then the receiver will have no guarantee to recover the genuine message m_1. □

Having this lemma along with Corollary 1, we are now ready to prove Theorem 1.

Proof of Theorem 1. First note that the data stored in a single node does not have any redundancy. In other words if some part of the data stored in a single node is a function of the rest of the data we can improve the storage-bandwidth trade-off in the whole system by simply removing the redundant part from each node. Hence, α is an information theoretic lower bound on the repair bandwidth required for any repair decoding trial. In specific as Lemma 3 asserts, in any BAER code, the sum of repair bandwidth provided by any subset of helpers of size $d-2b$ should be at least α. As a result for any BAER code with positive storage capacity, and any feasible d, $d_{\min} \leq d \leq d_{\max}$ we have

$$\frac{\gamma(d)}{d}(d-2b) \geq \alpha.$$

However, since (8) assures this is achievable by a single code for all feasible choices of d, we will then have (1) of Theorem 1. □

Remark 3. *Note that Theorem 1, introduces limits for $d_{\min}$, and k in an MBR BAER code. The lower limit $2b$ for k, $d_{\min}$ could be justified using Lemma 3. Since for the choice of $d < 2b$, or $k < 2b$ any subset of helpers of size $d-2b$, or $k-2b$ is empty and hence the storage capacity of the BAER code supporting such a d or k is zero (trivial code).*

Lemma 4. *For any BAER code $\mathcal{C}(\alpha, \gamma(\cdot), n, d_{\min}, d_{\max}, k, b)$, let $D = \{d_{\min}, \cdots, d_{\max}\}$, then the total storage capacity F is upper bounded as follows*

$$F \leq \sum_{i=0}^{k-2b-1} \min\left(\alpha, \min_{d\in D}\left((d-2b-i)\frac{\gamma(d)}{d-2b}\right)\right). \tag{10}$$

In specific, for the MBR case we have,

$$F_{MBR} \leq \frac{\alpha(k-2b)}{d_{\min}-2b}\left(d_{\min}-b-\frac{k-1}{2}\right). \tag{11}$$

Please see Appendix B for the proof of this Lemma.

Finally the proof of Theorem 2 simply follows from (9) in Corollary 1, and Lemma 4.

VI. Conclusion

We considered a modified setup for the regenerating codes in which error resiliency and bandwidth adaptivity (BAER) are required to be satisfied simultaneously, and studied the storage-bandwidth trade-off in the modified BAER setup for regenerating codes. Focusing on the minimum repair bandwidth point, we derived the total repair bandwidth function in the bandwidth adaptive scheme along with the corresponding storage capacity through proposing an exact repair coding scheme, and providing the converse proofs. We showed that for the MBR mode, optimality is achievable in strongest form (i.e., point-wise rather than Pareto optimality). We also presented an upper bound on the storage capacity of the BAER setup for the general case.

Appendix A
Proof of Lemma 1

Proof. Note that $\Theta_{\mathcal{H}}$ is an $\alpha \times \alpha$ square matrix. We will prove this lemma by showing that the determinant of this matrix, denoted by $\det(\Theta_{\mathcal{H}})$, is non-zero. From (7), we have

$$\det(\Theta_{\mathcal{H}}) = \left|\left[\Phi_{i_1}, \cdots, \Phi_{i_{d-2b}}\right]\left(\Omega_d^{\intercal} \otimes I_{(d-2b)}\right)\right|.$$

Note that we can perform a column permutation on the matrix $\left[\Phi_{i_1}, \cdots, \Phi_{i_{d-2b}}\right]$, and simultaneously perform the same permutation on the rows of matrix $\left(\Omega_d^{\intercal} \otimes I_{(d-2b)}\right)$, and hence keep the product untouched. We define the column permutation on the former matrix to shift $z(d-d_{\min})$ of the columns to the right-most part of the matrix. The set of columns to be shifted to the right are selected such that; 1) the number of selected columns from each of the $\Phi_{i_\ell},\ \ell \in \{1, \cdots, d-2b\}$ blocks is exactly

$$\frac{z(d-d_{\min})}{d-2b}, \tag{12}$$

2) the shifted columns are selected evenly from different positions in the $\Phi_{i_\ell},\ \ell \in \{1, \cdots, d-2b\}$ blocks. In other words, for each $j \in \{1, \cdots, z\}$, the number of $\Phi_{i_\ell},\ \ell \in \{1, \cdots, d-2b\}$ blocks whose j^{th} column is selected and shifted to the right-most part of the matrix is exactly $d-d_{\min}$.

The resulting matrix after this column permutation looks like $[A, A']$, where A is a $\alpha \times \alpha$ left submatrix. In A columns could be partitioned into $d-2b$ groups such that in each group the non-zero segment of the columns are located in the same positions and form a full-rank transposed Vandermonde submatrix. Therefore, $A_{\alpha\times\alpha}$ is full-rank.

It is easy to check that performing the corresponding permutation on the rows of matrix $\left(\Omega_d^{\intercal} \otimes I_{(d-2b)}\right)$, will shift exactly

$$\frac{z(d-d_{\min})}{d-2b} \tag{13}$$

rows from each of the $\Omega_d^{\intercal}$ diagonal blocks to the bottom of the matrix, and hence in the resulting matrix the top α rows form a block diagonal matrix with $z \times z$ square diagonal blocks. Moreover, since $\Omega_d^{\intercal}$ in a Vandermonde matrix, the remaining $z \times z$ diagonal blocks are all full-rank. To summarize, after the permutation we have,

$$\det(\Theta_{\mathcal{H}}) = \left|\left[A_{\alpha\times\alpha}, A'_{\alpha\times\alpha\left(\frac{d}{d_{\min}}-1\right)}\right]\begin{bmatrix} B_{\alpha\times\alpha} \\ B'_{\alpha\left(\frac{d}{d_{\min}}-1\right)\times\alpha}\end{bmatrix}\right|.$$

Since A is full-rank then we can easily use elementary column operations to transform all the columns in the submatrix A' to zero columns. Again, by applying the inverse of these elementary operations to the rows of $[B^{\intercal}, B'^{\intercal}]^{\intercal}$, the product will remain untouched. Moreover, such elementary operations will not make any changes to the rows of the B submatrix, and as a result we have

$$\begin{aligned}\det(\Theta_{\mathcal{H}}) &= \left|\left[A_{\alpha\times\alpha}, O_{\alpha\times\alpha\left(\frac{d}{d_{\min}}-1\right)}\right]\begin{bmatrix} B_{\alpha\times\alpha} \\ B''_{\alpha\left(\frac{d}{d_{\min}}-1\right)\times\alpha}\end{bmatrix}\right| \\ &= \det(A)\det(B) \neq 0.\end{aligned}$$

□

Appendix B
Proof of Lemma 4

Proof. The proof follows ideas similar to [1], [5], and [4]. However, to derive an upper bound on the capacity of a BAER setting, we introduce a *genie-aided* version of this code. Then we derive the upper bound on the capacity F, by finding an appropriate cut-set in the *information-flow graph* corresponding to the genie-aided version.

In the genie-aided version of $\mathcal{C}(\alpha, \gamma(\cdot), n, d_{\min}, d_{\max}, k, b)$, when we select the set of d helper nodes for a repair, the genie identifies a subset of size $d-2b$ of the selected helpers as genuine helpers, and we will only receive repair data from them. Similarly, in the download process after choosing the set of k nodes, the genie identifies a subset of size $k-2b$ of genuine nodes among them, and the data collector only receives data from this subset. From Lemma 3 we know that limiting the connections in the genie-aided version will not reduce the storage capacity. Hence, the storage capacity of the genie-aided version is an upper bound for F; the storage capacity of the original setting.

To derive an upper bound on the storage capacity of the genie-aided version, we will consider the *information-flow graph* as introduced in [1]. The information-flow graph is a *directed acyclic graph* (DAG) model to represent the flow of information during a sequence of repairs and data reconstructions in the network. The source of information is represented as a single node which has only out-going edges, and any data collector is represented as a single node which only has incoming edges. Every storage node ℓ, which has once been used in the network, is represented by a pair of nodes ℓ_{in}, and ℓ_{out} in the DAG such that an edge with capacity α takes the floe of information from ℓ_{in} to ℓ_{out}. This edge represents the per node storage capacity constraint for node ℓ, hence we refer to such edges as *storage edges*. In addition to storage edges there are three other types of edges in the information-flow graph, namely the *download edges*, the *repair edges*, and the *source edges*. Download edges have capacity α and take information

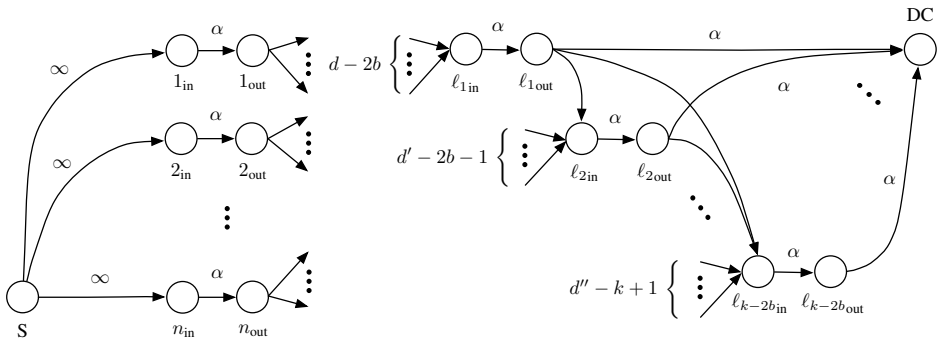

Fig. 1. The information-flow graph of the genie-aided version of a BAER code $\mathcal{C}(\alpha, \gamma(\cdot), n, d_{\min}, d_{\max}, k, b)$.

flow from the any node ℓ_{out} to a data collector node if ℓ is among the genuine selected nodes for the data collector. Repair edges take information flow from a node ℓ_{out} to a node ℓ'_{in} in the information-flow graph if ℓ' is a replacement node in the distributed storage network, and ℓ is one of the selected genuine helpers for the repair. The capacity of repair edges in the information-flow graph is then $\gamma(d)/(d-2b)$, for the chosen parameter d in the corresponding repair. Finally, we also consider a set of n source edges with infinite capacity, taking information flow from the source node to the input node of initial n storage nodes in the network. Figure 1 depicts one example of an information-flow graph.

Corresponding to any specific series of repair and data reconstruction processes, there exists a specific information-flow graph. Any cut-set in the DAG model for an information-flow graph consists of a set of edges such that removing them remains no path from the source to the data collector. As a results the sum capacity of all the edges in a cut-set provides an upper bound on the capacity of information which could be stored in the corresponding distributed storage network and restored by a data collector after the sequence of repairs associated to the information-flow graph is performed. We are going to consider the information-flow graph depicted in figure 1.

As depicted n the figure in our scenario a data collector is downloading the data stored in the network by accessing a set of $k-2b$ nodes in the genie-aided setting. These nodes are indexed as $\ell_1, \cdots, \ell_{k-2b}$, and each of them is a replacement node. We also assume for any $i \in \{1, \cdots, k-2b\}$, the repair for node ℓ_i is performed after the repair for any node $\ell_j,\ j < i$, and all of the nodes $\ell_j,\ j \in \{1, \cdots, i-1\}$ are used as genuine helpers in the repair of the node ℓ_i.

Initiating with an empty cut-set, for any $i \in \{1, \cdots, k-2b\}$, we will compare the capacity of the storage edge of node i with the sum of the capacities of the set of repair edges which take information flows from the helper nodes not in $\{\ell_1, \cdots, \ell_{i-1}\}$, and whichever is smaller its corresponding edges will be added to the cut-set. Let $D = \{d_{\min}, \cdots, d_{\max}\}$. The cut-set achieved by this scheme then results in the following upper bound on the total storage capacity.

$$F \leq \sum_{j=0}^{k-2b-1} \min\left(\alpha, \min_{d\in D}\left((d-2b-j)\frac{\gamma(d)}{d-2b}\right)\right).$$

In the case of the MBR mode however, we know from Theorem 1,

$$\gamma_{\text{MBR}}(d) = \frac{\alpha d}{d-2b}.$$

As a result we get

$$\begin{aligned} F_{\text{MBR}} &\leq \sum_{i=0}^{k-2b-1} \min\left(\alpha, (d_{\min}-2b-i)\frac{\alpha}{d_{\min}-2b}\right). \\ &= \frac{\alpha(k-2b)}{d_{\min}-2b}\left(d_{\min} - b - \frac{k-1}{2}\right). \end{aligned}$$

□

REFERENCES

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

[2] V. Cadambe, S. Jafar, H. Maleki, K. Ramchandran, and C. Suh, "Asymptotic interference alignment for optimal repair of mds codes in distributed storage," *IEEE Transactions on Information Theory*, vol. 59, no. 5, pp. 2974–2987, May 2013.

[3] A.-M. Kermarrec, N. L. Scouarnec, and G. Straub, "Repairing multiple failures with coordinated and adaptive regenerating codes," in *Proc. IEEE International Symposium on Network Coding (NetCod)*, Beijing, China, July 2011, pp. 1–6.

[4] V. Aggarwal, C. Tian, V. A. Vaishampayan, and Y.-F. R. Chen, "Distributed data storage systems with opportunistic repair," in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, Toronto, Canada, April–May 2014, pp. 1833–1841.

[5] S. Pawar, S. E. Rouayheb, and K. Ramchandran, "Securing dynamic distributed storage systems against eavesdropping and adversarial attacks," *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6734–6753, October 2011.

[6] K. V. Rashmi, N. B. Shah, K. Ramchandran, and P. V. Kumar, "Regenerating codes for errors and erasures in distributed storage," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, Cambridge, MA, USA, July 2012, pp. 1202–1206.

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