# Threshold-based File Maintenance Strategies for Mobile Cloud Storage Systems

Swetha Shivaramaiah, Gokhan Calis, O. Ozan Koyluoglu, and Loukas Lazos
Department of Electrical and Computer Engineering, The University of Arizona
Email: {sshivaramaiah, gcalis, ozan, llazos}@email.arizona.edu

***Abstract*—We study the data reliability problem for a community of devices forming a mobile cloud storage system. We consider the application of regenerating codes for maintaining a file within a geographically-limited area. Such codes require lower bandwidth to regenerate lost data fragments compared to file replication or reconstruction. We investigate threshold-based repair strategies where data repair is initiated after a threshold number of data fragments have been lost due to node mobility. We show that at a low departure-to-repair rate regime, a *lazy repair* strategy in which repairs are initiated after several nodes have left the system outperforms *eager repair* in which repairs are initiated after a single departure. This optimality is reversed when nodes are highly mobile. We further compare distributed and centralized repair strategies and derive the optimal repair threshold for minimizing the average repair cost per unit of time, as a function of underlying code parameters.**

***Index Terms*—Distributed storage, regenerating codes, mobile cloud, data reliability.**

## I. Introduction

Local caching and content distribution from a community of mobile devices has been proposed as an alternative architecture to traditional centralized storage [1]–[3]. The so-called mobile cloud storage systems reduce the traffic load of the already over-burdened infrastructure network and improve content availability in the event of network outages. Also, the cloud storage system can be used for applications like cloud computing, mobile health care and mobile learning. In a mobile cloud storage scenario, a file $\mathcal{F}$ is stored within a geographically-limited area $\mathcal{A}$ by a community of mobile devices. A user within $\mathcal{A}$ can download $\mathcal{F}$ from the community of mobile devices, without accessing the network infrastructure.

However, content storage at mobile devices leads to frequent data loss due to mobility. Since not all storage systems have the same mobility, one may need to design the system according to the mobility of the devices within. For example, cloud storage system used for local caching may have a lower mobility compared to one used for applications involving mobile users. That's because devices in local caching systems are mostly stable whereas devices in applications involving mobile users in a wireless network may move in and out of the network more freely. When a mobile device storing $\mathcal{F}$ or any fragment of $\mathcal{F}$ exits $\mathcal{A}$, the stored data is lost. To deal with such losses, redundancy is introduced in the form of data replication or coding [4], [5]. In replication storage, copies of $\mathcal{F}$ are stored at multiple devices within the community. More sophisticated coding schemes such as erasure coding achieve the same reliability at lower storage overhead [6], [7]. Despite the application of coding, a stored file $\mathcal{F}$ will eventually be lost when a certain number of mobile devices (storage nodes) depart from $\mathcal{A}$. To maintain $\mathcal{F}$ over long time periods, the mobile cloud system must be capable of recovering the lost data. A repair scenario is shown in Fig. 1. Lost data is recovered by downloading fragments from the storage nodes that remain within $\mathcal{A}$. The amount of data downloaded for repair is referred to as the *repair bandwidth*. For mobile communities, the repair bandwidth can be significant.

This work is supported in part by the National Science Foundation under Grants No. 1563622 and No. 1409172.

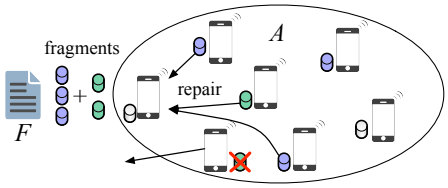



Fig. 1: File maintenance in a mobile cloud storage system.

The file maintenance problem for distributed storage systems has been primarily studied assuming that erasure codes are applied for redundancy [6], [8]. However, erasure codes may not be bandwidth-efficient. The repair bandwidth can be reduced by applying regenerating codes, which allow fragment recovery without file reconstruction (see [9]–[12] and references therein). Although regenerating codes lower the repair bandwidth (per single node repair), the design of an efficient repair strategy for a mobile cloud storage system involves cost optimizations with respect to many parameters, including the code redundancy factor, the device departure and fragment repair rates, the threshold for initiating repair operations, and the available communication bandwidth. In this paper, *we study the problem of minimizing the file maintenance cost, as a function of the network dynamics, the code parameters and the communication model for repairing lost data fragments*. Specifically, we make the following contributions.

- We focus on threshold-based file maintenance strategies, in which repairs are initiated when a threshold number of fragments is lost. We analyze two communication models, namely *distributed repair* and *centralized repair*. In distributed repair, the new storage nodes independently download data from existing nodes to recover lost fragments. In centralized repair, a *leader* node first recovers $\mathcal{F}$ via reconstruction, before regenerating and distributing the repaired fragments to new storage nodes.

- We derive the *optimal repair threshold* that minimizes the average repair cost per unit of time for each communication model. Our results show that no one strategy is optimal for all possible system configurations and mobility patterns. At the low mobility-to-repair rate regime, repairing at the regeneration threshold yields the optimal strategy. On the other hand, at the high mobility-to-repair rate regime, regenerating after a single fragment loss minimizes the average repair cost per unit of time.

Although we present our analysis in the context of mobile storage systems, we emphasize that our work is applicable in any distributed storage system where fragment loss can occur. This includes popular wired distributed storage architectures such as HDFS [13], in which fragment loss occurs due to server failures. Similar to a mobile storage cloud system, the optimal threshold repair strategy for the wired domain would depend on the fragment loss rate (server failure or unavailability rate) and the various system parameters.

## II. Related Work

In reliable storage systems, information is replicated or coded such that the original content can be recovered if some limited fraction of the stored data is lost. Replication is the most intuitive way to introduce redundancy. This method refers to the maintenance of verbatim copies of the same file $\mathcal{F}$. Although replication is easy to implement, it suffers from high storage and repair overhead.

Erasure codes incur less storage overhead compared to replication while maintaining the same degree of reliability. In particular, Maximum Distance Separable (MDS) codes achieve the optimal tradeoff between failure tolerance and storage overhead [14], [15]. An $(n, k)$ MDS code encodes $k$ data chunks to $n$ fragments and can tolerate up to $n - k$ fragment losses. Any $k$ encoded fragments can be used to reconstruct $\mathcal{F}$. Fig. 2(a) shows the encoding process for a file $\mathcal{F}$ of size 4MB using a (4, 2) erasure code. File $\mathcal{F}$ is split into $k = 2$ chunks $A$ and $B$, each of size 2MB. The two chunks are then encoded into $n = 4$ fragments. The repair bandwidth for this scheme equals the size of the original file. Reed-Solomon codes are a classical example of MDS codes and are deployed in many existing storage systems (e.g. [7], [16]–[18]).

Although erasure codes offer significant savings in storage, their repair bandwidth is suboptimal. Regenerating codes trade repair bandwidth for storage and can recover lost fragments without reconstructing the entire file. They were initially investigated in the seminal work of Dimakis *et al.* [9], which focuses on the following setup. A file $\mathcal{F}$ of size $\mathcal{M}$ symbols is encoded into $n$ fragments, each of size $\alpha$ symbols, such that (i) the file can be reconstructed from any $k$ fragments, and (ii) a lost fragment can be repaired by downloading $\beta \leq \alpha$ symbols from any $d \geq k$ fragments, resulting in a repair bandwidth of $\gamma = d\beta$. Dimakis *et al.* characterized the tradeoff between per node storage ($\alpha$) and repair bandwidth ($\gamma$) [9].

Fig. 2(b) shows an example of a $(n, k, d, \alpha, \beta) = (4, 2, 3, 2, 1)$ regenerating code. Here, the file $\mathcal{F}$ is split into $k = 2$ chunks each of size $\alpha = 2$MB. The chunks are encoded in $n = 4$ fragments, with each fragment being 2MB. A failed node in this scenario can be regenerated by retrieving fragments of size $\beta = 1$MB from $d = 3$ surviving nodes. This yields a repair bandwidth of $d\beta = 3$MB which is less that $k\alpha = 4$MB. Note, however, that regeneration can be applied only if at least $d$ fragments are available. If fewer than $d$ but more than $k$ fragments remain available, the lost fragments can only be repaired through file reconstruction.

In the context of mobile cloud systems, Pääkkönen *et al.* considered a wireless device-to-device network used for distributed storage [19]. The authors showed the energy consumption for maintaining data using regenerating codes is lower compared to retrieving a lost file from a remote source. This result holds if the per-bit energy cost for communication between the mobile devices is lower than the cost for communicating with the remote source.

In a follow-up work, Pääkkönen *et al.* compared replication with regeneration for a similar wireless P2P storage system [2]. They derived closed-form expressions for the expected total energy cost of file retrieval using replication and regeneration. They showed that the expected total cost of 2-replication is lower than the cost of regeneration. However, only an eager repair strategy was considered in the analysis. Moreover, the advantages of regeneration were not fully exploited by considering codes with different parameters. Pääkkönen *et al.* also addressed the problem of tolerating multiple simultaneous failures [20]. They investigated the energy overhead of regenerating codes in a cellular network. They showed that large energy gains can be obtained by employing regenerating codes. These gains depend on the file popularity. The authors provided decision rules for choosing between simple caching, replication, MSR and MBR codes, based on numerical results on certain application scenarios. In our work, we analytically provide decision rules to choose optimal repair strategies that minimize the repair bandwidth per unit of time.

Pedersen *et al.* recently studied the cost of content caching on mobile devices using erasure codes [21]. They derived analytical expressions for the cost of content download and repair bandwidth as a function of the repair interval. These expressions were used to evaluate the communication cost of distributed storage for MDS codes, regenerating codes, and locally repairable codes. Their results show that in high churn, distributed storage can reduce the communication cost compared to downloading from a base station. They conclude that MDS codes are the best performers in this setup.

## III. System Model

### A. Network Model

We consider a distributed storage system (DSS) consisting of mobile storage nodes that enter and exit a geographically-limited area $\mathcal{A}$. When a node departs from $\mathcal{A}$, its data is lost. As we are interested in the system performance due to network dynamics, we do not consider data loss due to hardware failures. Such failures occurs orders of magnitude less frequently than node departures. Following the network dynamics model of prior works [8], [19], we model the time $X_i$ spent by each

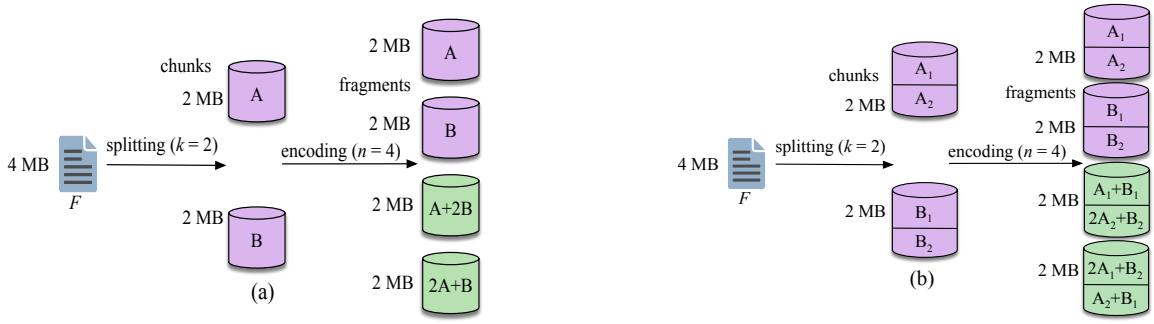



Fig. 2: Storage of $F$ using (a) a $(n = 4, k = 2)$ erasure code and, (b) a $(n = 4, k = 2, d = 3, \alpha = 2, \beta = 1)$ regenerating code.

node within $\mathcal{A}$ as an exponentially distributed random variable with parameter $\lambda$ (i.e., $X_i \sim \text{Exp}(\lambda),\ \forall i$). Random variables $\{X_i\}$ are assumed independent and identically distributed.

The repair time is modeled by an exponentially distributed random variable with parameter $\mu$. For ease of analysis, we assume that $\mu$ is independent of the number of fragments that need to be repaired. This model corresponds to independent repairs that proceed in parallel at different nodes once the repair process is initiated and it corresponds to the distributed nature of the mobile cloud DSS. Finally, we define $\rho = \frac{\lambda}{\mu}$ as the ratio of the departure-to-repair rate.

### B. Storage Model

A file $\mathcal{F}$ of size $\mathcal{M}$ bits is stored in $n$ storage nodes using a regenerating code with parameters $(n, k, d, \alpha, \beta)$ (see Fig. 2(b)). We focus on the two most popular types of regenerating codes, namely Minimum Storage Regenerating (MSR) codes and Minimum Bandwidth Regenerating (MBR) codes. These two classes of codes operate at the end points of the tradeoff between per node storage and repair bandwidth, as introduced in [9]. MSR codes achieve minimum storage by setting $\alpha = \mathcal{M}/k$ and minimize the repair bandwidth under this constraint. Their operating point is given by:

$$(\alpha_{\text{MSR}}, \gamma_{\text{MSR}}) = \left( \frac{\mathcal{M}}{k}, \frac{\mathcal{M}d}{k\,(d-k+1)} \right). \tag{1}$$

Note that, for MSR codes, $\alpha_{\text{MSR}} \leq \gamma_{\text{MSR}}$ and hence, the per-node storage is smaller than the repair bandwidth. MBR codes, on the other hand, minimize the repair bandwidth (achieved when $\gamma = \alpha$), and operate at:

$$(\alpha_{\text{MBR}}, \gamma_{\text{MBR}}) = \left( \frac{2\mathcal{M}d}{2kd - k^2 + k}, \frac{2\mathcal{M}d}{2kd - k^2 + k} \right). \tag{2}$$

Instances of these codes can be found in [10]–[12].

### C. File Repair Model

In our model, the system continuously monitors the redundancy level and initiates a repair when $\tau$ nodes are left within $\mathcal{A}$. The determination of $\tau$, the type of repair (regeneration, reconstruction, or both) and the communication model for fragment retrieval (centralized or distributed) form a *file maintenance strategy*. We note that the practical implementation details of the redundancy monitoring mechanism and of the communication protocols for retrieving various fragments are beyond the scope of the present work. We focus on the theoretical aspects of the maintenance process. Since repairs are initiated only when the number of remaining nodes reaches threshold $\tau$, a repair strategy can be viewed as an i.i.d. system recovery process occurring every $\Delta$ seconds, where $\Delta$ is a random variable denoting the time elapsed between two instances of a fully repaired system. For this recovery process, we define the following costs.

**Definition 1** (Repair cost $c(\tau)$)**.** *The number of bits $c(\tau)$ that must be downloaded from the $\tau$ remaining nodes to restore $n$ fragments in $\mathcal{A}$, when $n - \tau$ nodes have departed $\mathcal{A}$.*

**Definition 2** (Average repair cost per unit of time $r(\tau)$)**.** *The average cost per unit of time for maintaining $n$ fragments in $\mathcal{A}$, defined as $c(\tau)$ over the average time between two instances of a fully repaired system, i.e., $E[\Delta]$, with $n$ fragments ($r(\tau)$ is measured in bits per unit of time).*

We determine the optimal file maintenance strategy for different node departure rates, code parameters, and communication models for fragment retrieval.

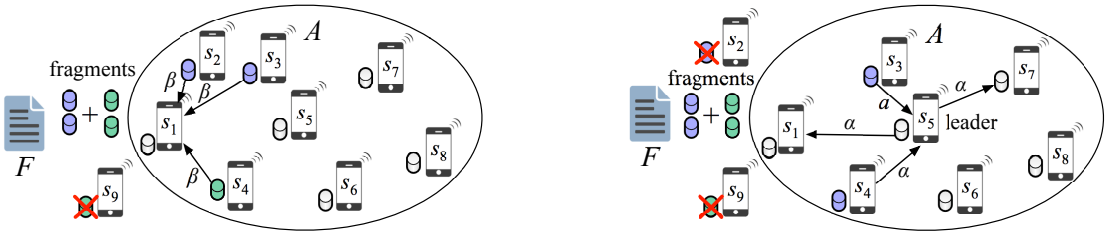
## IV. File Maintenance Strategies

Let $\tau$ denote the number of nodes remaining within $\mathcal{A}$ after the departure of $n - \tau$ nodes. We focus on determining the optimal repair threshold $\tau^*$, which minimizes the average repair cost per unit of time. We first compare the distributed repair strategy with centralized repair strategy.

### A. Distributed Repair

In distributed repair, nodes recover lost fragments by independently downloading relevant symbols from other nodes. The repair process is initiated when $\tau$ nodes remain within $\mathcal{A}$, where $k \leq \tau < n - 1$ (when $\tau < k$, the data is irrecoverably lost). If $\tau \geq d$, fragment recovery can be performed through regeneration. Each of the $n - \tau$ replacement nodes downloads $\beta$ symbols from $d$ storage nodes and independently regenerates a lost fragment. Fig. 3(a) demonstrates the distributed repair process for a file $\mathcal{F}$ stored with a $(n = 4, k = 2, d = 3, \alpha = 2, \beta = 1)$ regenerating code. One fragment of $\mathcal{F}$ is lost because node $s_9$ departed from $\mathcal{A}$. The lost fragment is regenerated at $s_1$ by independently downloading $\beta = 1$ symbol from three nodes. The total repair bandwidth is equal to 3 symbols.

If $\tau < d$, regeneration cannot be directly applied. To reduce the repair cost, we consider a hybrid scheme consisting of




(a) distributed repair (b) centralized repair

Fig. 3: (a) Distributed repair: nodes independently regenerate a lost fragment by obtaining symbols from other nodes, (b) centralized repair: a leader node reconstructs $\mathcal{F}$ and distributes lost fragments to new nodes.

regeneration and reconstruction. First, $d-\tau$ nodes are repaired by downloading $\alpha$ symbols from $k$ nodes and reconstructing $\mathcal{F}$. When $d$ fragments become available, regeneration is applied to repair the remaining $n-d$ nodes. Accordingly, the repair cost is expressed by:

$$c_D(\tau) = \begin{cases} k\alpha(d-\tau)+\gamma(n-d), & \text{if } \tau < d \\ \gamma(n-\tau), & \text{if } \tau \geq d. \end{cases} \quad (3)$$

The subscript $D$ in $c_D(\tau)$ is used to denote the cost of distributed repair and $\gamma$ denotes the regeneration cost of a single fragment which depends on the underlying regeneration code (see eqs. (1) and (2) for MSR and MBR codes, respectively). From (3), it is evident that $c_D(\tau)$ monotonically decreases with $\tau$. Moreover, the rate of cost change (with respect to $\tau$) is higher when $\tau < d$. To determine the optimal threshold $\tau^*$, we are interested in minimizing $r_D(\tau)$, which captures the repair cost for maintaining $n$ fragments per unit of time.

To calculate $r_D(\tau)$, we use the continuous-time Markov chain (CTMC) model shown in Fig. 4. This model captures the periodic repair process when node departures occur independently, the time spent by each node in $\mathcal{A}$ is exponentially distributed with parameter $\lambda$, and the system recovery process is exponentially distributed with parameter $\mu$.

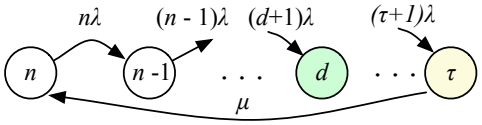



Fig. 4: Markov chain for a threshold-based file maintenance.

The CTMC consists of $n-\tau+1$ states representing the number of fragments that remain within $\mathcal{A}$ after each node departure, until a repair at state $\tau$ is initiated. Note that we have omitted states after $\tau$ in the CTMC model, because we are interested in optimizing the periodic cost of repairing the DSS at threshold $\tau$. Moreover, the transition probability to state $\tau-1$ is negligible for most realistic scenarios in which $\mu \gg \tau\lambda$. For cases when $\mu \not\gg \tau\lambda$, we compute the mean time it takes to depart from the optimal repair strategy of repairing at state $\tau$ and interpret this event as a form of system error which leads to data loss (see Section V-C).

For the CTMC in Fig 4, the departure rate from a state $i$ equals the node departure rate $\lambda$, times the number of nodes which store fragments at state $i$. When the repair process is initiated, the system transitions from state $\tau$ to state $n$ because all fragment repairs nodes proceed in parallel. For the CTMC, we define the expected average cost $r_D(\tau)$ per unit of time as

$$r_D(\tau) = \frac{c_D(\tau)}{\mathrm{E}[\Delta]}, \quad (4)$$

where $E[\Delta]$ is the average time between two transitions through the $n^{th}$ state in the periodic repair process[1]. For $\Delta$,

$$\Delta = T_n + T_{n-1} + \cdots + T_{\tau+1} + T_\tau, \quad (5)$$

where $T_i$ denotes the time that the system stays at state $i$ (inter-departure time) and $T_\tau$ is the expected time for completing repairs so that $n-\tau$ fragments are recovered (return to state $n$). The random variables $T_i$ are independent and exponentially distributed with parameter $i\lambda$, whereas $T_\tau$ is exponentially distributed with parameter $\mu$. In particular, $\mathrm{E}[T_i] = \frac{1}{i\lambda}$ and $\mathrm{E}[T_\tau] = \frac{1}{\mu}$. Therefore, $\mathrm{E}[\Delta]$ is the sum expectation of independent exponential random variables.

$$\mathrm{E}[\Delta] = \sum_{i=\tau+1}^{n} \frac{1}{i\lambda} + \frac{1}{\mu} = \frac{H_{n,\tau}}{\lambda} + \frac{1}{\mu}, \quad (6)$$

where $H_{n,\tau} = \sum_{i=\tau}^{n} \frac{1}{i}$. Combining (4) and (6), we obtain the average repair cost per unit of time as follows.

$$r_D(\tau) = \frac{c_D(\tau)}{\mathrm{E}[\Delta]} = \begin{cases} \frac{\lambda\mu(k\alpha(d-\tau)+\gamma(n-d))}{\mu H_{n,\tau}+\lambda}, & \text{if } \tau < d \\ \frac{\lambda\mu(\gamma(n-\tau))}{\mu H_{n,\tau}+\lambda}, & \text{if } t \geq d. \end{cases} \quad (7)$$

We use (7) to determine the optimal threshold $\tau^*$ which minimizes $r_D(\tau)$. This is given by Propositions 1 and 2.

**Proposition 1.** *For regeneration ($d \leq \tau \leq n-1$), the optimal repair threshold $\tau^*$ is given by*

$$\tau^* = \begin{cases} d, & \rho \leq \frac{H_{n-1,d}}{n-d-1} - \frac{1}{n} \\ n-1, & \textit{otherwise.} \end{cases} \quad (8)$$

*Proof.* Due to space limitations, the proof is provided in our technical report [22]. □

Proposition 1 determines the $\rho$ regime for which repairs at $\tau = d$, an instance of *lazy repair*, is more efficient than initiating repairs at $\tau = n-1$, referred to as *eager repair*. In

[1]The alternative definition of $r_D(\tau) = E\left[\frac{c_D(\tau)}{\Delta}\right]$ is not useful because the expectation is infinite. This is due to the infinitesimally small values that can be obtained by $\Delta$, whereas $c_D(\tau)$ remains lower bounded.

the following Lemma, we show that there is always a positive $\rho$ for which lazy repair is more efficient that eager repair.

**Lemma 1.** *There is always some $\rho > 0$ for which lazy repair ($\tau^* = d$) is more efficient than eager repair ($\tau = n-1$), independent of the code parameters used for regeneration.*

*Proof.* Due to space limitations, the proof is provided in our technical report [22]. □

We now examine if there is a $\rho$ regime for which the hybrid scheme, i.e., reconstruction plus regeneration results in a lower expected cost per unit of time compared to regeneration only. This rate regime is given by the following proposition.

**Proposition 2.** *For regeneration plus reconstruction ($k \leq \tau \leq d$), the optimal repair threshold $\tau^*$ is given by*

$$\tau^* = \begin{cases} k, & \rho \leq \frac{\gamma(n-d)H_{d,k}}{k\alpha(d-k)} - H_{n,d} \\ d, & \textit{otherwise.} \end{cases} \tag{9}$$

*Proof.* Due to space limitations, the proof is provided in our technical report [22]. □

Similar to Lemma 1, we investigate if the highest departure-to-repair rate for which reconstruction at $k$ is more efficient than regeneration is always positive independent of the code parameters. Unlike the case of Lemma 1, we show that for a certain relationship between $n, k, \gamma$, and $\alpha$, regeneration is strictly more efficient than regeneration plus reconstruction, independent of $\rho$. For any other code parameters, the most efficient strategy depends on $\rho$.

**Lemma 2.** *For any departure-to-repair ratio $\rho$, regeneration is strictly more efficient than regeneration plus reconstruction for codes satisfying $n\gamma < k^2\alpha$.*

*Proof.* Due to space limitations, the proof is provided in our technical report [22]. □

We further explore the condition in Lemma 2 for MSR and MBR codes. For MSR codes, we obtain that $dn < k^2(d-k+1)$ by substituting the operation points of MSR from (1). Similarly, for MBR codes, we obtain that $n < k^2$ by substituting the operation points of MBR from (2). Note that Lemma 2 does not enumerate all possible codes for which regeneration is strictly more efficient than regeneration plus reconstruction for any $\lambda$. This is because we have used bounds on the harmonic function to derive the analytic formulas. Numerical bounds could provide a more accurate range of code parameters for which Lemma 2 is true.

### B. Centralized Repair

In the centralized strategy, repairs are performed by a *leader node* in two stages. In the first stage, the leader node downloads $\alpha$ symbols from $k$ nodes and reconstructs $\mathcal{F}$. In the second stage, the leader node transmits $\alpha$ bits to each of the remaining $(n-\tau-1)$ nodes to restore the remaining $(n-\tau-1)$ fragments. Fig. 3(b) shows an example of centralized repair for a $(n=4, k=2, d=3, \alpha=2, \beta=1)$ regenerating code. Nodes $s_2$ and $s_9$ have departed from area $\mathcal{A}$, leading to the loss of their respective fragments. Node $s_5$, who acts as a leader, downloads $\alpha = 2$ symbols from $k = 2$ other nodes to reconstruct $\mathcal{F}$. It then distributes $\alpha = 2$ symbols to $s_1$ and $s_7$ to restore the system reliability. The repair cost of centralized repair is given by:

$$c_C(\tau) = \alpha\left(k + n - \tau - 1\right). \tag{10}$$

In (10), the subscript $C$ in $c_C(\tau)$ is used to denote the cost of centralized repair. The node departure process does not vary with the repair strategy. Therefore, the same CTMC model shown in Fig. 4 applies for the centralized repair. According to (4), the average repair cost $r_C(\tau)$ is given by:

$$r_C(\tau) = \frac{c_C(\tau)}{\mathrm{E}[\Delta]} = \frac{\lambda\mu\alpha(k+n-\tau-1)}{\mu H_{n,\tau} + \lambda}. \tag{11}$$

The optimal threshold $\tau^*$ which minimizes $r(\tau)$ is obtained in Proposition 3.

**Proposition 3.** *The optimal repair threshold $\tau^*$ which minimizes $r(\tau)$ for centralized repair is given by*

$$\tau^* = \begin{cases} k, & \rho \leq \frac{kH_{n-1,k}}{n-k-1} - \frac{1}{n} \\ n-1, & \textit{otherwise} \end{cases} \tag{12}$$

*Proof.* Due to space limitations, the proof is provided in our technical report [22]. □

Using Proposition 3, we can determine the optimal repair strategy for any $\rho$, when centralized repair is employed. We note that according to Lemma 1, the value $\frac{kH_{n-1,k}}{(n-k-1)} - \frac{1}{n}$ is strictly positive for any code parameters. Therefore, there is always a departure-to-repair ratio for which lazy repair is more efficient than eager repair, independent of the code used for regeneration and reconstruction.

## V. Analysis of Maintenance Strategies

In this section, we characterize the $\rho$ regime for which lazy repair is more cost-efficient than eager repair. Moreover, we determine the optimal repair strategy (decentralized vs. centralized) as a function of the code parameters, when the departure and repair rates are fixed. To ease the reader to our analysis, we summarize the cost of repair in Table I.

### A. Eager vs. Lazy Repair

According to the results of Propositions 1, 2, and 3, we classify the departure-to-repair ratios into a *low departure-to-repair rate regime* ($\rho_{low}$) and a *high departure-to-repair rate regime* ($\rho_{high}$). The two regimes are defined by finding the lowest and highest rates, based on the bounds stated in the three propositions.

$$\rho_{\text{low}} = \min\Big\{ \frac{H_{n-1,d}}{n-d-1} - \frac{1}{n}, \frac{\gamma(n-d)H_{d,k}}{k\alpha(d-k)} - H_{n,d}, \frac{kH_{n-1,k}}{(n-k-1)} - \frac{1}{n}\Big\}. \tag{13}$$

$$\rho_{\text{high}} = \max\Big\{ \frac{H_{n-1,d}}{n-d-1} - \frac{1}{n}, \frac{\gamma(n-d)H_{d,k}}{k\alpha(d-k)} - H_{n,d}, \frac{kH_{n-1,k}}{(n-k-1)} - \frac{1}{n}\Big\}. \tag{14}$$

TABLE I: Cost comparison of repair strategies at different thresholds.

| | Distributed Repair | | | Centralized Repair | |
|---|---|---|---|---|---|
| | Regeneration | | Regeneration + Reconstruction | Reconstruction | |
| Code | $r_D(n-1)$ | $r_D(d)$ | $r_D(k)$ | $r_C(n-1)$ | $r_C(k)$ |
| MSR | $\frac{n\mathcal{M}d\lambda\mu}{k(d-k+1)(\mu+n\lambda)}$ | $\frac{\mathcal{M}(n-d)d\lambda\mu}{k(d-k+1)(\lambda+\mu H_{n,d})}$ | $\frac{\mathcal{M}[k(d-k+1)(d-k)+d(n-d)]\lambda\mu}{k(d-k+1)(\lambda+\mu H_{n,k})}$ | $\frac{n\mathcal{M}\lambda\mu}{\mu+n\lambda}$ | $\frac{(n-1)\mathcal{M}\lambda\mu}{k(\lambda+\mu H_{n,k})}$ |
| MBR | $\frac{2n\mathcal{M}d\lambda\mu}{k(2d-k+1)(\mu+n\lambda)}$ | $\frac{2\mathcal{M}(n-d)d\lambda\mu}{k(2d-k+1)(\lambda+H_{n,d})}$ | $\frac{2\mathcal{M}d(n+kd-k^2-d)\lambda\mu}{k(2d-k+1)(\lambda+H_{n,k})}$ | $\frac{2n\mathcal{M}d\lambda\mu}{(2d-k+1)(\mu+n\lambda)}$ | $\frac{2(n-1)\mathcal{M}d\lambda\mu}{k(2d-k+1)(\lambda+\mu H_{n,k})}$ |

Noting that $\frac{H_{n-1,d}}{n-d-1}-\frac{1}{n}<\frac{kH_{n-1,k}}{(n-k-1)}-\frac{1}{n}$ for $k<d$, the two regime expressions can be simplified to

$$\rho_{\text{low}}=\min\left\{\frac{H_{n-1,d}}{n-d-1}-\frac{1}{n},\frac{\gamma(n-d)H_{d,k}}{k\alpha(d-k)}-H_{n,d}\right\}.$$

$$\rho_{\text{high}}=\max\left\{\frac{\gamma(n-d)H_{d,k}}{k\alpha(d-k)}-H_{n,d},\frac{kH_{n-1,k}}{(n-k-1)}-\frac{1}{n}\right\}.$$

For any $\rho\leq\rho_{\text{low}}$, the repair cost per unit of time is minimized when lazy repair is applied since that choice of $\rho$ would be lower than the bounds found in (8),(9) and (12) and the corresponding repair thresholds are the lowest possible. On the other hand, for any $\rho\geq\rho_{\text{high}}$, eager repair (i.e., repair at $\tau^*=n-1$) yields the lowest $r(\tau)$. These findings hold for both distributed and centralized repair. If the departure-to-repair rates do not lie in either of the $\rho$ regimes, then the optimal repair policy (eager vs. lazy) depends on the relationship of the code parameters and the repair strategy (centralized or distributed).

### B. Centralized vs. Distributed Repair

We now fix the departure rate $\lambda$ and repair rate $\mu$ to compare the repair cost of centralized vs. distributed repair per unit of time, as a function of the code parameters. Specifically, we determine relationships between $n,k,d$ and the code type (MSR vs. MBR) for which an optimal strategy can be derived. Our results are stated in the following two propositions.

**Proposition 4.** *For $d\leq\tau^*\leq n-1$, using MBR codes and distributed repair minimizes the average repair cost per unit of time, if $d>\frac{n+k-1}{3}$.*

*Proof.* Due to space limitations, the proof is provided in our technical report [22]. □

We now prove that if $\tau^*$ lies between $k$ and $d$, using MSR codes with centralized repair is optimal.

**Proposition 5.** *For $k\leq\tau^*<d$, the optimal repair strategy is given by centralized repair with MSR codes.*

*Proof.* Due to space limitations, the proof is provided in our technical report [22]. □

### C. Mean Time to Data Loss for Periodic Repairs

We now examine the *Mean Time to Data Loss (MTTDL)* for the periodic threshold repair process. For our purposes, we consider that data is lost if the DSS transitions from state $\tau$ to state $\tau-1$ instead of state $n$. That is, if a node leaves the system before repairs are completed when initiated at state $\tau$, the repair process is abandoned and the system eventually reaches state $k-1$, at which data is lost. In this case, the file $F$ is reinstated at the mobile nodes by a central entity. Note that when $\tau>k$ repairs could be re-initiated at state $\tau-1$, because at least $k$ fragments remain available. We opted not to consider this option for the MTTDL calculation to capture the periodic nature of the threshold repair strategy. The MTTDL reflects the period of time at which the DSS oscillates between states $n$ and $\tau$. The time to reach state $k-1$ assuming no repairs are attempted after state $\tau$ is given by:

**Proposition 6.** *For a threshold-based repair strategy attempting regeneration at state $\tau$, the MTTDL is given by*

$$MTTDL=\sum_{i=1}^{\infty}\Big(\frac{iH_{n,\tau}}{\lambda}+\frac{i-1}{\mu}+\frac{H_{\tau,k-1}}{\lambda}\Big)(1-p)^{(i-1)}p, \quad (15)$$

*where $p=\frac{\tau\lambda}{\tau\lambda+\mu}$.*

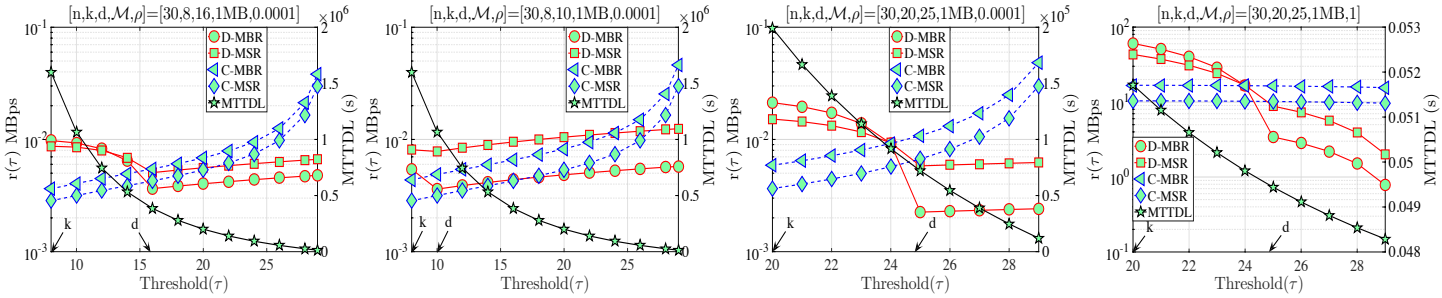
*Proof.* Due to space limitations, the proof is provided in our technical report [22]. □

The MTTDL is a decreasing function of $\tau$. This is intuitive considering that the rate $\lambda\tau$ of departing from state $\tau$ decreases with $\tau$, making it more probable to return to state $n$. Moreover, the average time it takes to reach state $\tau$ from state $n$ increases with $\tau$. This indicates that the periodic repair of the DSS will on average have last longer if a lazy repair strategy is adopted.

### D. Numerical Examples

In this section, we validate our theoretical results be providing numerical examples. Fig. 5(a) shows $r(\tau)$ when $d>\frac{n+k-1}{3}$ and $\rho=10^{-4}$ . According to Proposition 4, for this combination of code parameters, a distributed repair strategy with MBR codes (D-MBR) achieves the minimum $r(\tau)$ for all $d\leq\tau^*\leq n-1$. The minimum occurs at $\tau^*=d$. Moreover, according to Proposition 5, centralized MSR codes (C-MSR) minimize $r(\tau)$ for $k\leq\tau<d$. This is verified in all plots of Fig. 5, for which the cost is minimized by the C-MSR strategy when $\tau^*=k$, if $\tau<d$. In Fig. 5(b), we show $r(\tau)$ when $d<\frac{n+k-1}{3}$ and $\rho=10^{-4}$. For this case, there is no one scheme with optimal cost for any value of $d\leq\tau\leq n-1$. For $\tau>16$, D-MBR is optimal, whereas for $10\leq\tau\leq15$, C-MSR becomes optimal. C-MSR achieves the lowest overall cost at $\tau=k$.

We also studied the impact of $\rho$, when the code parameters are fixed to $(n=30,k=20,d=25)$. Fig. 5(c) shows the average cost per unit of time ($r(\tau)$) when $\rho=10^{-4}$. For this $\rho$ regime, a lazy repair strategy with $\tau^*=d$ minimizes $r(\tau)$, with D-MBR codes achieving the lowest cost. On the other hand, eager repair becomes optimal for any $\rho>\rho_{\text{high}}$. This is observed in Fig. 5(d), in which the value of $\rho$ has




(a) (b) (c) (d)

Fig. 5: Cost $r(\tau)$ vs. repair threshold ($\tau$) for: (a) $d > \frac{n+k-1}{3}$, (b) $d < \frac{n+k-1}{3}$ >, (c) $\rho = 10^{-4}$, (d) $\rho = 1$.

been increased to one. D-MBR codes still remain the optimal option, however, the optimal repair threshold is now shifted to $\tau^* = n - 1$. Note that at the high $\rho$ regime, all codes exhibit the same behavior. The average cost per unit of time becomes a decreasing function of $\tau$.

Finally, on the right $y$-axis of the plots in Fig. 5, we show the MTTDL values for the given set of parameters. As expected, the MTTDL is an decreasing function of $\tau$ due to the corresponding increase in departure rate from state $\tau$ with the value of $\tau$. The MTTDL becomes impractical in the high $\rho$ regime, because nodes frequently leave area $\mathcal{A}$ before repairs can be completed.

## VI. Conclusion and Future Work

We analyzed threshold-based repair strategies for maintaining files in mobile cloud storage systems. We derived the optimal repair thresholds for both distributed and centralized repair schemes under fragment regeneration and/or reconstruction. Our results show that optimal thresholds are dependent on system configurations, the underlying code parameters and mobility-to-repair rate ratio. For high mobility-to-repair scenarios, eager repair minimizes the average repair cost per unit of time. Under low mobility-to-repair ratio, lazy repair is optimal in terms of average repair cost.

As part of our future work, we will consider a more advanced repair model in which fragment repairs occur under a fixed bandwidth constraint. This assumption makes the repair rate $\mu$ dependent on the repair threshold $\tau$. Moreover, we will investigate a sequential repair model in which the system may oscillate between states $k$ and $n$, once repairs are initiated. Finally, we will generalize our model to the case where repairs are initiated at a every state with some probability and study the cost vs. MTTDL tradeoff.

## References


[1] N. Golrezaei, A. G. Dimakis, and A. F. Molisch, "Device-to-device collaboration through distributed storage," in *Proc. GLOBECOM Conference*. IEEE, 2012.

[2] J. Pääkkönen, C. Hollanti, and O. Tirkkonen, "Device-to-device data storage for mobile cellular systems," in *Proc. GLOBECOM Workshops*, 2013, pp. 671–676.

[3] N. Golrezaei, A. G. Dimakis, and A. F. Molisch, "Scaling behavior for device-to-device communications with distributed caching," *IEEE Trans. on Information Theory*, vol. 60, no. 7, pp. 4286–4298, 2014.

[4] R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G. M. Voelker, "Total recall: system support for automated availability management." in *Proc. NSDI Conference*, vol. 4, 2004, pp. 25–25.

[5] F. Dabek, J. Li, E. Sit, J. Robertson, M. F. Kaashoek, and R. Morris, "Designing a DHT for low latency and high throughput," in *Proceedings of NDSI Symposium*, 2004, pp. 7–7.

[6] Y. Hu, Y. Xu, X. Wang, C. Zhan, and P. Li, "Cooperative recovery of distributed storage systems from multiple losses with network coding," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 2, pp. 268–276, 2010.

[7] H. Weatherspoon and J. D. Kubiatowics, "Erasure coding vs. replication: A quantitive comparison," in *Proc. 1st International Workshop on Peer-to-Peer Systems*, 2002, pp. 328–337.

[8] F. Giroire, J. Monteiro, and S. Pérennes, "Peer-to-peer storage systems: a practical guideline to be lazy," in *Proc. GLOBECOM Conference*, 2010, pp. 1–6.

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

[10] K. Rashmi, N. Shah, and P. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Trans. on Information Theory*, vol. 57, no. 8, pp. 5227 –5239, Aug. 2011.

[11] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: MDS array codes with optimal rebuilding," *IEEE Trans. on Information Theory*, vol. 59, no. 3, pp. 1597–1616, March 2013.

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

[13] D. Borthakur, "HDFS architecture guide," http://hadoop.apache.org/common/docs/current/hdfs_design.pdf, 2008.

[14] R. C. Singleton, "Maximum distance q -nary codes," *IEEE Trans. on Information Theory*, vol. 10, no. 2, pp. 116–118, 1964.

[15] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures," *IEEE Trans. on Computers*, vol. 44, no. 2, pp. 192–202, Feb. 1995.

[16] C. Huang and L. Xu, "STAR: An efficient coding scheme for correcting triple storage node failures," in *Proceedings of the USENIX Conference on File and Storage Technologies*, 2005.

[17] B. Calder, Wang *et al.*, "Windows Azure Storage: A highly available cloud storage service with strong consistency," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, 2011, pp. 143–157.

[18] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems." in *Proceedings of the OSDI Conference*, 2010, pp. 61–74.

[19] J. Pääkkönen, P. Dharmawansa, C. Hollanti, and O. Tirkkonen, "Distributed storage for proximity based services," in *Communication Technologies Workshop (Swe-CTW), 2012 Swedish*. IEEE, 2012, pp. 30–35.

[20] J. Pääkkönen, C. Hollanti, and O. Tirkkonen, "Device-to-device data storage with regenerating codes," in *Multiple Access Communications*. Springer, 2015, pp. 57–69.

[21] J. Pedersen, I. Andriyanova, F. Brännström *et al.*, "Distributed storage in mobile wireless networks with device-to-device communication," *arXiv preprint arXiv:1601.00397*, 2016.

[22] S. Shivaramaiah, G. Calis, O. O. Koyluoglu, and L. Lazos, "Repair strategies for storage on mobile clouds," 2016.