# Distributed Data Storage Systems with Opportunistic Repair

Vaneet Aggarwal, Chao Tian, Vinay A. Vaishampayan, and Yih-Farn R. Chen
AT&T Labs-Research, Florham Park, NJ 07932
email: {vaneet, tian, vinay, chen}@research.att.com

***Abstract*—The reliability of erasure-coded distributed storage systems, as measured by the mean time to data loss (MTTDL), depends on the repair bandwidth of the code. Repair-efficient codes provide reliability values several orders of magnitude better than conventional erasure codes. Current state of the art codes fix the number of helper nodes (nodes participating in repair) a priori. In practice, however, it is desirable to allow the number of helper nodes to be adaptively determined by the network traffic conditions. In this work, we propose an opportunistic repair framework to address this issue. It is shown that there exists a threshold on the storage overhead, below which such an opportunistic approach does not lose any efficiency from the optimal storage-repair-bandwidth tradeoff; i.e. it is possible to construct a code simultaneously optimal for different numbers of helper nodes. We further examine the benefits of such opportunistic codes, and derive the MTTDL improvement for two repair models: one with limited total repair bandwidth and the other with limited individual-node repair bandwidth. In both settings, we show orders of magnitude improvement in MTTDL. Finally, the proposed framework is examined in a network setting where a significant improvement in MTTDL is observed.**

## I. Introduction

Efficient data storage systems based on erasure codes have attracted much attention recently, because they are able to provide reliable data storage at a fraction of the cost of those based on simple data replication. In such systems, the data shares stored on a server or disk may be lost, either due to physical disk drive failures, or due to storage servers leaving the system in a dynamic setting. To guarantee high reliability, the lost data shares must be repaired and placed on other servers. The total amount of data that needs to be transferred during this repair phase should be minimized, both to reduce network traffic costs and to reduce repair time. Dimakis *et al.* [1] recently proposed a framework to investigate this tradeoff between the amount of storage at each node (*i.e.,* data storage) and the amount of data transfer for repair (*i.e.,* repair bandwidth).

In the setting considered in [1], there are $n$ nodes, data can be recovered from any $k$ nodes, and the lost data share needs to be regenerated using (certain information obtained from) $d \geq k$ helper nodes. It was shown that for any fixed $d > k$, there exists a natural tradeoff between the total amount of repair traffic bandwidth and the storage overhead; the two extreme points are referred to as the minimum storage regenerating (MSR) point and the the minimum bandwidth regenerating (MBR) point, respectively. In general, by using $d > k$ helper nodes, the total repair traffic can be reduced, compared to the naive (and currently common) practice of using only $k$ helper nodes.

There are few repair strategies considered in the literature. Functional repair assumes that the failed block is repaired with possibly different data than that of the failed node, as long as the repaired system maintains the code properties. The code construction for functional repair in [1] uses network coding [2], and a survey for using network coding in distributed storage can be found in [3]. Exact repair assumes that the failed disk is reconstructed exactly. Some of the works for exact repair can be seen in [4–11, 19]. In this paper, we will consider functional repair for distributed storage systems.

The number of helper nodes, $d$, is a design parameter in [1]; however, in practice it may not be desirable to fix $d$ a priori. As an example, consider the dynamic peer-to-peer distributed storage environment, where peers are geo-distributed in a wide area without a centralized control mechanism. The peers may choose to join and leave the system in a much less controlled manner than in a data center setting. One example of such a system is the Space Monkey project [15], and another is the open source peer-to-peer storage sharing solution built in Tahoe-LAFS [16]. In such systems, at the time of node repair it is desirable to utilize as many as possible helper nodes and achieve the most efficient repair, instead of accessing only a fixed subset of $d$ available peers. In other words, instead of designing a code for a single value of $d$, it would be more desirable for a code to be universally applicable for multiple values of $d$.

In this work, we address this problem and propose an erasure-coded system based on an opportunistic repair framework, where a single universal code is able to regenerate the share on the lost node by using any $d$ helper nodes, in a set $D$ of available choices of $d$ values. We address the issues of system design and then develop fundamental bounds on the performance that can be achieved. Specifically, our distributed storage system consists of a network of storage nodes, connected together by an IP network. In an effort to minimize the time to repair a failed node, our system must determine the bandwidths between the site to be repaired and all the potential sites that will participate in the repair. This problem becomes especially important when the underlying physical network topology is not known and when there is significant ambient traffic in the network. Thus our system includes a pre-repair probing step, whose objective is to determine the available bandwidth constraints, and then select

www.manaraa.com

a repair strategy to minimize the repair time.

We then investigate the tradeoff between storage and the repair bandwidths (one bandwidth for each $d$ value), and provide a characterization of the complete tradeoff region through an analysis technique similar to that used in [1]. A particularly interesting question arises as to whether there is a loss for any given $d \in D$ by taking this opportunistic approach, when compared to the case in [1] where the parameter $d$ is fixed a-priori. We find that there is a critical storage overhead threshold, below which there is no such loss; above this threshold, the universality requirement of the code indeed incurs a loss. In particular, the MSR points for individual $d$ values are simultaneously achievable in the opportunistic setting; this phenomenon for the special case of MSR codes has in fact been observed in an asymptotically optimal code construction [7].

The reliability of storage systems is usually a foremost concern to the service provider and the users. Data loss events can be extremely costly: consider for example the value of data in systems storing financial or medical records. For this reason, storage systems must be engineered such that the chance of an irrecoverable data loss is extremely low, perhaps on the order of one in 10 million objects per 10,000 years of operation - as is the case of the Amazon Glacier Storage Service. The reliability of distributed storage systems is usually measured using the mean time to data loss (MTTDL) of the system. In this paper, we analyze the system performance under two models, the first with limited total bandwidth to all the failed nodes, and the second with separate per user links for repair. We find the mean time to data loss for these two cases with the rate of failures and repairs being constant over time. For both of these models, we show that the MTTDL is improved by a multiplicative factor of $\frac{k^{n-k}}{\binom{n-1}{k-1}}$, when compared to that without opportunistic repair. This translates to a significant improvement, and is many orders of magnitude better, even for lower values of $n$ and $k$. For $n = 51$ and $k = 30$, this improvement is a multiplicative factor of around $10^{17}$, while even for the parameters chosen in Facebook HDFS ($n = 14$ and $k = 10$ ) [21], this improvement is a factor of around 14.

In practical systems, the bandwidths from all the nodes is not the same. Hence, there is an additional optimization step: choose a subset of nodes among the active ones to repair a failed node. Even with different bandwidths of different links among the nodes, we find that the opportunistic distributed storage system helps in repair time and hence the MTTDL increases by orders of magnitude.

The remainder of the paper is organized as follows. Section II gives the background and introduces the system model. Section III gives our results on functional repair, and the loss for an opportunistic system as compared to an optimized system for a single value of $d$. Section IV gives our results on the mean time to data loss of an opportunistic distributed storage system. Section V gives our results when the bandwidths between different links are random. Section VI concludes the paper.

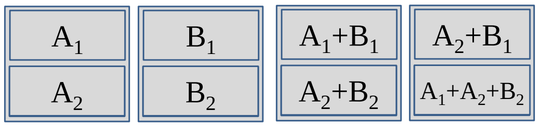

Fig. 1. A (4,2) MDS binary erasure code ( [19]). Each storage node (box) stores two blocks that are linear binary combinations of the original data blocks $A_1$, $A_2$, $B_1$ and $B_2$. In this example the total stored size is $\mathsf{M} = 4$ blocks.

## II. Background and System Model

### A. Distributed Storage Systems

A distributed storage system consists of multiple storage nodes that are connected in a network. The issue of code repair arises when a storage node of the system fails. Consider an example with $n = 4$ distributed nodes, which can be recovered from any $k = 2$ nodes as shown in Figure 1. If only one node fails and needs to be repaired, the conventional strategy is to get all the $M = 4$ blocks from two active nodes. However, as shown in [1], fewer than $M$ blocks are sufficient for repairing a failed node and in fact three blocks suffice in the above example. For example, to repair the first node, $B_1$, $A_1 + B_1$, and $A_2 + B_1$ are enough to get $A_1$ and $A_2$. Similarly, to repair the second node, $A_2$, $A_2 + B_2$ and $A_2 + B_1$ are sufficient to get $B_1$ and $B_2$. To repair the third node, $A_1 - A_2$, $-B_1 + B_2$ and $A_2 + B_1$ are sufficient to get $A_1 + B_1$ and $A_2 + B_2$. To repair the fourth node, $A_1$, $B_1 - B_2$ and $A_2 + B_2$ are sufficient to get $A_2 + B_1$ and $A_1 + A_2 + B_2$.

In this paper, we will consider a mode of repair called functional repair, in which the failed block may be repaired with possibly different data than that of the failed node, as long as the repaired system maintains the code properties (repair bandwidth and the ability to recover from $n - k$ erasures). The authors of [1] derive a tradeoff between the amount of storage at each node and the repair bandwidth for a given number of nodes $n$, number of nodes from which the data should be recovered $k$ and the number of nodes $d$ that can be accessed for repair. Let each node store $\alpha$ bits, let $\beta_d$ bits be downloaded from each of the $d$ nodes for repair and let the total data be $\mathsf{M}$ bits. It was shown in [1] that the optimal storage-repair-bandwidth tradeoff, i.e., $\alpha$ vs. $\beta_d$, satisfies

$$\sum_{i=0}^{k-1} \min(\alpha, (d-i)\beta_d) \geq \mathsf{M}. \tag{1}$$

For a given $\alpha$, the minimum $\beta_d$ satisfying the above is given as $\beta_d^*(\alpha)$. The codes associated with the two extremes of this tradeoff are referred to as minimum-storage regenerating (MSR) and minimum-bandwidth regenerating (MBR) codes. An MSR code has a minimum storage overhead requirement per node while an MBR point has the minimum repair bandwidth.

One particularly important observation in this example code is as follows: there are two possible choices of the value $d$, which is $d = 3$ (with repair bandwidth from each node as 1) and $d = 2$ (with repair bandwidth from each node as

www.manaraa.com

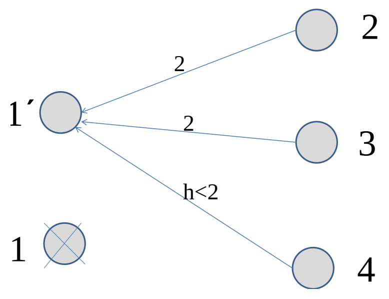

Fig. 2. Example network with less bandwidth from one node.

2). It is natural to ask whether this flexible choice of $d$ is generally available, and whether this flexibility incurs any loss of efficiency.

## B. Motivation Example in a Simple Network

Consider a file of size $\mathsf{M}$ is encoded into $n$ shares, which are placed on $n$ distinct nodes[1]. We assume that the $n$ nodes are connected via a network and that $R_{ij}$, $1 \le i, j \le n$, $i \ne j$ represents the bandwidth from node $i$ to node $j$. In a typical scenario the rates will obey a set of constraints and these constraints will have an impact on the time to repair a single failed node. As an illustration consider the network in Fig. 2, which shows a failed node (node 1), an $(n, k) = (4, 2)$ MDS code (such as a Reed-Solomon code), and the link bandwidths between the helper nodes and the node to be restored (node $1'$). Consider two scenarios, the first where $d = k = 2$, the second where $d = n - 1 = 3$. In the first case the repair time is proportional to $\mathsf{M}/4$ if the helper nodes are nodes 2 and 3. On the other hand if $d = 3$, then even though we need to download only $\mathsf{M}/4$ symbols from each helper node, the bottleneck link is the link from fourth node and thus the repair time is proportional to $\mathsf{M}/(4h)$. If $h > 1$, it is beneficial to use $d = 3$ while if $h < 1$, it is beneficial to use $d = 2$. This example illustrates the situations where the network topology may be unknown at encoding time, which may be changing due to link failures, upgrades, or, a network composed of mobile storage nodes. Clearly there are benefits if the value of $d$ and the set of helper nodes can be chosen at the time of repair.

Motivated by this observation, we assume a distributed storage system with probing, by which the number of nodes to access for repair can be decided. Using an active probe, we obtain the bandwidths between any pair of nodes which can be used to decide the number of nodes to access in order to repair a failed node. Thus, in practice, the value of $d$ is not fixed and hence the code design should work for multiple values of parameter $d$. We will consider bandwidths between multiple nodes for choosing the value of $d$ in Section V.

## C. Opportunistic Distributed Storage System

Assume that there are $n$ nodes and the total file that we need to store is of size $\mathsf{M}$, which can be reconstructed from $k$ nodes. The repair bandwidth is a function of how many nodes are used for repair. In this setting, we consider what happens for the region formed by the bandwidth to repair from multiple valued of nodes accessed, $d$. We assume that the code structure should remain the same and should allow for repair with varying values of $d \in D = \{d_1, d_2, \cdots\}$ such that each $d_i \ge k$. We denote the storage capacity at each node by $\alpha$ and the bandwidth from each node as $\beta_d$ when the content is accessed from $d$ nodes.

Thus, the distributed storage system with opportunistic repair works for different values of $d$. An opportunistic code can take advantage of varying number of failed nodes. For example, in the $n = 4$, $k = 2$ case in Figure 1, we note that the system can be repaired from $d = 2$ as well as $d = 3$ nodes. So, this code design works for all $k \le d \le n-1$. In this paper, we will find the parameters the codes have to satisfy so that they work for multiple values of $d$. It was noted in [7] that at the MSR point, there is no loss for opportunistic repair in the sense that at $\alpha_{MSR}$, the optimal values $\beta_d$ for the MSR point corresponding to $d$ are simultaneously achievable for all $d \ge k$. This result was further extended in [20] for adaptive repair at the MSR point where the failed nodes perform a coordinated repair. In this paper, however, we do not consider any coordination among the failed nodes. Further, we will consider the complete tradeoff region formed by the storage capacity and the repair bandwidths for different values of $d$.

## III. Results on Opportunistic Distributed Storage System

In this Section, we present the main theoretical results on opportunistic repair distributed storage systems.

As in storage systems with a fixed number of repair helper nodes, in opportunistic-repair distributed storage systems, there is also a fundamental tradeoff between the share size and the repair bandwidths. For functional repair, this tradeoff can be completely characterized as given in the following theorem.

**Theorem 1.** *If the value of $\alpha$ and $\beta_{d_j}$ for all $d_j \in D$ satisfies*

$$\sum_{i=0}^{k-1} \min(\alpha, \min_{d_j \in D}(d_j - i)\beta_{d_j}) \ge \mathsf{M}, \tag{2}$$

*there exist linear network codes that achieve opportunistic distributed storage system with the storage per node given by $\alpha$ and the repair bandwidth from $d_j$ nodes given by $d_j\beta_{d_j}$ for any $d_j \in D$. Further, if the above condition is not satisfied, it is information theoretically impossible to achieve opportunistic distributed storage system with the above properties.*

*Proof.* The proof follows by an extension of the result in [1]. The details can be found in Appendix A. □

An important question of practical interest is whether by taking the opportunistic approach, a loss of storage-repair efficiency is necessarily incurred. By leveraging Theorem 1, we can provide an answer to this question, as given in the next theorem where we let $D = \{d_1, \cdots d_l\}$ for $k \le d_l < \cdots < d_1 < n$.

[1]We do not distinguish between node and disk from here on.

www.manaraa.com

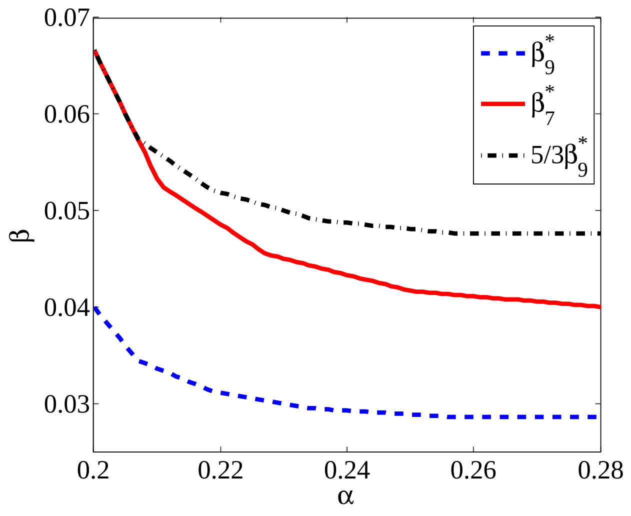

Fig. 3. Loss with Opportunistic Repair for $n = 10$, $k = 5$, and $\mathsf{M} = 1$. We consider the set of possible repair values as $\{7, 9\}$.

**Theorem 2.** *For a given $n$, $k$, $\mathsf{M}$, and $\alpha \geq \frac{\mathsf{M}}{k}$, $(\alpha, \beta^*_{d_1}(\alpha), \cdots, \beta^*_{d_l}(\alpha))$ satisfy (2) for $|D| > 1$ if and only if either $k = 1$ or $\alpha \leq \alpha_o(k, d_1, M) \triangleq \frac{\mathsf{M}(d_1-k+2)}{k(d_1-k+2)-1}$.*

The proof of this theorem is given in the appendix. This theorem essentially states that below the critical threshold $\alpha_o$ of the storage share size, there is no loss of optimality by imposing the opportunistic repair requirement, while above this threshold, such a loss is indeed necessary. A special case of practical relevance is given as a corollary next, which essentially states that there is no loss by requiring the MSR codes to have the opportunistic repair property.

**Corollary 1.** *MSR points for all values of d are simultaneously achievable, i.e., $(\frac{\mathsf{M}}{k}, \beta^*_k(\frac{\mathsf{M}}{k}), \beta^*_{k+1}(\frac{\mathsf{M}}{k}), \cdots, \beta^*_{n-1}(\frac{\mathsf{M}}{k}))$ satisfy* (2).

This particular result has been previously observed in [7]. In fact, even for the more stringent exact-repair case where the failed disk needs to be repaired with exact same copy, the same result holds asymptotically, using the class of asymptotically optimal codes constructed in [7].

The next theorem deals with the case when $(\alpha, \beta_{d_1})$ is operating on the optimal (non-opportunistic-repair) storage-repair-bandwidth tradeoff curve, when $\alpha$ is larger than the given threshold $\alpha_o$. In this case, a loss of repair bandwidth is necessary for all the other values of $d = d_2, d_3, \ldots, d_l$, and the following theorem characterizes this loss precisely.

**Theorem 3.** *For a given $n$, $k$, $\mathsf{M}$, and $\alpha \geq \frac{\mathsf{M}}{k}$, $(\alpha, \beta^*_{d_1}(\alpha), \frac{d_1-k+1}{d_2-k+1}\beta^*_{d_1}(\alpha)\cdots, \frac{d_1-k+1}{d_l-k+1}\beta^*_{d_1}(\alpha))$ satisfies (2). Further, given that $\beta_{d_1} = \beta^*_{d_1}(\alpha)$ in (2), the above point has the smallest value of possible $\beta$'s for the remaining values $d_i$, $i > 1$.*

As an example, we consider $n = 10$ and $k = 5$ in Figure 3. We note that without opportunistic repair, the tradeoffs for $d = 7$ and $d = 9$ could be achieved. However, both these curves are not simultaneously achievable. Till the first linear segment of the curve with largest $d$ ($d = 9$ in this case), the values of $\beta$ on the two tradeoff curves are simultaneously achievable. After that, there is a loss. Assuming that we choose to be on the tradeoff for $d = 9$, the black dash-dotted curve in Figure 3 represents the best possible tradeoff for $d = 7$ and thus shows an increase with respect to the optimal code that works for only $d = 7$.

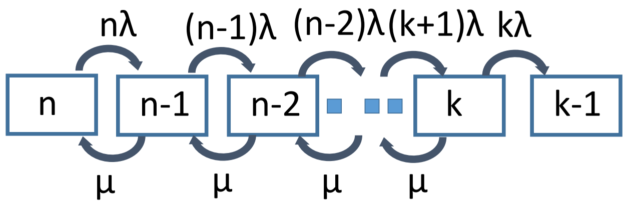

Fig. 4. The state transition diagram for Chen's model.

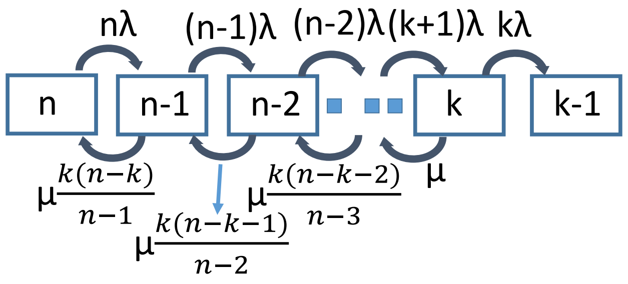

Fig. 5. The state transition diagram for Chen's model with opportunistic repair.

## IV. Mean Time to Data Loss

In this Section, we will consider the improvement in mean time to data loss due to opportunistic repair. We will consider two models which are widely studied in the literature, and consider the impact of opportunistic repair on these systems. The rate at which individual components (hard drives, tapes, etc.) fail is denoted by $\lambda$ while the rate at which those components are repaired is $\mu$. Alternatively, these two rates might instead be expressed as times: Mean- Time-To-Failure (MTTF) and Mean-Time-To-Repair (MTTR) respectively. When $\lambda$ and $\mu$ are constant over time, i.e. exponentially distributed, $MTTF = 1/\lambda$ and $MTTR = 1/\mu$.

The first model is Chen's model [12]. Chen et al. presented models for estimating the MTTDL for various RAID configurations, including RAID 0 (no parity), RAID 5 (single parity) and RAID 6 (dual parity). In this model, failures occur at a rate equal to the number of operational devices times the device failure rate, and repairs occur at the device repair rate regardless of the number of failed devices. Thus, Chen's model assumes that the per-device repair rate is inversely proportional to the number of failed nodes. This happens if there is a bottleneck for bandwidth to repair all the nodes. The repair and failure rates for the Chen's model are shown in Figure 4.

When opportunistic repair at the MSR point is used, the repair time is smaller if there are more than $k$ nodes in the

www.manaraa.com

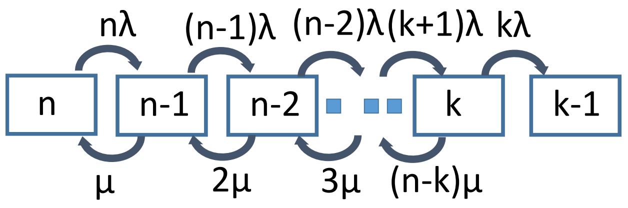

Fig. 6. The state transition diagram for Angus' model.

system. This is because less bandwidth will be needed to repair the failed nodes and hence the mean time to repair is smaller if the number of surviving nodes are greater than $k$. Using the result in the previous section, the parameters for state transition are as in Figure 5.

Next, we will characterize the MTTDL for the Chen's model with opportunistic repair.

**Theorem 4.** *The MTTDL for Chen's model with opportunistic repair is given as*

$$MTDL_{\textit{Chen, Opp}} = \sum_{l=0}^{n-k} \frac{(n-k-l)!}{(n-l)!(n-l-1)!} \sum_{i=0}^{n-k-l} (k\mu)^i \lambda^{-(i+1)} \frac{((n-l-i-1)!)^2}{(n-l-k-i)!} \quad (3)$$

*Proof.* The proof is provided in the Appendix. □

We note by the similar proof steps, we also get that the MTTDL for the original Chen's model is given as follows.

$$MTDL_{\text{Chen, Orig}} = \sum_{l=0}^{n-k} \frac{1}{(n-l)!} \sum_{i=0}^{n-k-l} \mu^i \lambda^{-(i+1)} (n-l-i-1)! \quad (4)$$

We will now consider these expressions in the limit that $\lambda << \mu$. In this regime, the two expressions above are given as follows

$$MTDL_{\text{Chen, Opp}} = \frac{k^{n-k}}{\binom{n-1}{k-1}} \frac{(k-1)!}{n!} \frac{\mu^{n-k}}{\lambda^{n-k+1}} \quad (5)$$

$$MTDL_{\text{Chen, Orig}} = \frac{(k-1)!}{n!} \frac{\mu^{n-k}}{\lambda^{n-k+1}} \quad (6)$$

Thus, we see that MTTDL increases by a factor of $\frac{k^{n-k}}{\binom{n-1}{k-1}}$ with opportunistic repair as compared to that without opportunistic repair.

The second model we consider is Angus' model [13]. Unlike Chen's model, Angus' model assumes that there are unlimited repairmen. This means that whether 1 device or 100 fail simultaneously, each failed device will be repaired at a constant rate. The state transition diagram for Angus' model is described in Figure 6.

We can also use opportunistic repair for Angus' model and hence save bandwidth when there are more surviving nodes. Using opportunistic repair, the modified state transition is described in Figure 7. The MTTDL for Angus' model with opportunistic repair is given as follows.

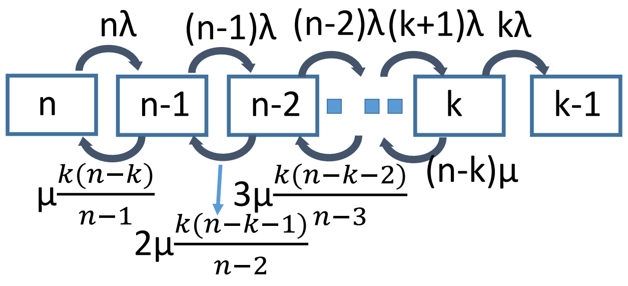

Fig. 7. The state transition diagram for Angus' model with Opportunistic Repair.

**Theorem 5.** *The MTTDL for Angus' model with opportunistic repair is given as*

$$MTDL_{\textit{Angus, Opp}} = \sum_{l=0}^{n-k} \frac{(n-k-l)!}{(n-l)!(n-l-1)!} \sum_{i=0}^{n-k-l} (k\mu)^i \lambda^{-(i+1)} \frac{((n-l-i-1)!)^2}{(n-l-k-i)!} i! \quad (7)$$

Since the proof steps are similar to that in the Chen's model, the proof is omitted. Further, the MTTDL for the original Angus' model is given as follows.

$$MTDL_{\text{Angus, Orig}} = \sum_{l=0}^{n-k} \frac{1}{(n-l)!} \sum_{i=0}^{n-k-l} \mu^i \lambda^{-(i+1)} (n-l-i-1)! i! \quad (8)$$

We will now consider these expressions in the limit that $\lambda << \mu$. In this regime, the two expressions above are given as follows

$$MTDL_{\text{Angus, Opp}} = \frac{k^{n-k}}{\binom{n-1}{k-1}} \frac{(k-1)!}{n!} \frac{\mu^{n-k}}{\lambda^{n-k+1}} (n-k)! \quad (9)$$

$$MTDL_{\text{Angus, Orig}} = \frac{(k-1)!}{n!} \frac{\mu^{n-k}}{\lambda^{n-k+1}} (n-k)! \quad (10)$$

Thus, we see that the MTTDL increases by a factor of $\frac{k^{n-k}}{\binom{n-1}{k-1}}$ with opportunistic repair as compared to that without opportunistic repair. Also, we note that the MTTDL loss for Angus' model is $(n-k)!$ higher than that in Chen's model.

## V. Network Simulation

We consider a network with $n$ distributed nodes, an $(n, k)$ MDS systematic code and consider the case where repair can be performed from any $d$ nodes, $k \le d < n$ nodes. When any node fails, it pings all the other nodes to determine the bandwidths from each of the active node and then determines the helper set, i.e the nodes that will participate in the repair.

We assume that the bandwidth between any two nodes is given by a maximum of a Gaussian random variable with mean 3 and variance 16 and 1/4. The bandwidth between any pair of nodes is independent. If we use a larger $d$ to repair (among the active nodes), we get an advantage in repair time based on Chen's model as $k(d-k+1)/d$ which makes it beneficial to use as many nodes as possible to repair. However, in the

المنارة للاستشارات
www.manaraa.com

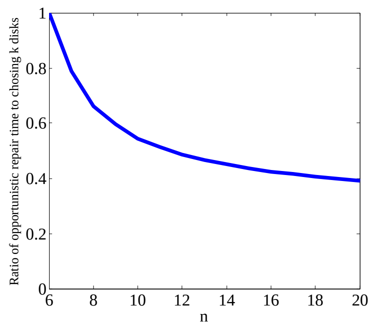

Fig. 8. Average ratio of time taken to repair 1 failed node with opportunistic repair to the the repair time using $k$ nodes.

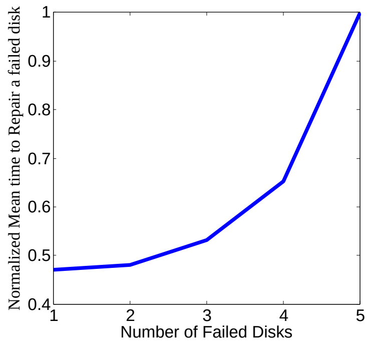

Fig. 9. Average of normalized time taken to repair a failed node with opportunistic repair when multiple nodes have failed. The normalization is taken such that time taken to repair $n-k$ failed nodes is unit.

case of realistic bandwidth, this rate is multiplied by $d^{\text{th}}$ best bandwidth to the node which is to be repaired since the data needs to be downloaded from $d$ servers. This rate decreases with $d$. Overall repair rate as a function of $d$ is the product of the two and is given as $k(d-k+1)/d\mu$ times the $d^{\text{th}}$ largest bandwidth. Thus, there is an optimization needed for $d$ in order to select the value of $d$ to use. Since only a single node is repaired at a time, we choose the node with the maximum rate to repair.

We note that in practice, the value of $\mu$ may not be constant and change with time, and that multiple failed disks can be repaired in parallel. However, in this section, we ignore these factors, and only consider the value of $\mu$ to be constant and that one disk is repaired at a time.

We first see the mean time to repair when 1 node has failed. One option is to choose the best $k$ nodes to repair. This serves as a base-line without opportunistic repair. With opportunistic repair, we can choose optimal $d$ number of nodes. We see the average ratio of time to repair for opportunistic repair with the time to repair from $k$ nodes in Figure 9, where the average is over 1000 runs for the bandwidths between different nodes. For a fixed $k=5$, Figure 9 gives this ratio for different values of $n$. As $n$ increases, the options for using greater than $k$ nodes increase so that the relative time taken to repair with opportunistic repair is smaller.

We next consider $k=5$ and $n=10$. For this system, we consider the repair time when $1,\cdots,4$ nodes fail as compared to the repair time when 5 nodes fail. We see that smaller is the number of nodes that fail, smaller is the time taken to repair a node with opportunistic repair.

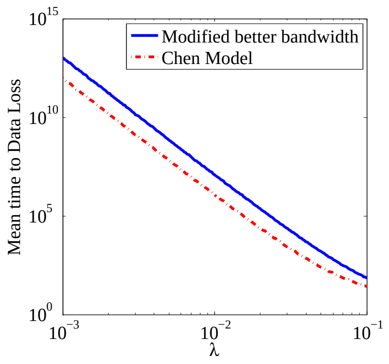

Fig. 10. Mean time to Data Loss of the System with Opportunistic Repair, as compared to Chen's model.

Finally, we consider the mean time to data loss of the system with opportunistic repair, taking the link bandwidths into account. We compare this to Chen's system where only $k$ nodes are used for repair even if more are available. We assume $n=10, k=5$ and $\mu=1$. We find that even taking the link bandwidths into account, the mean time to data loss for the opportunistic system is much higher than in the system that does not exploit opportunistic repair.

## VI. Conclusions

We describe a distributed erasure coded storage system with the capability that a failed node can be repaired from a number of helper nodes $d$ that is not fixed a priori and investigate the repair bandwidth vs. storage tradeoff for such a system. We then demonstrate the usefulness of opportunistic repair in the form of an improvement in the mean time to data loss of the system and show that the improvement is significant even when different nodes have random bandwidth links.

In this paper, we only consider functional repair for opportunistic distributed storage systems. Even though exact repair codes has been shown to exist asymptotically at the MSR point [7], general constructions for exact repair are still open. We have assumed a mesh network to present the benefits of opportunistic repair, the benefits from other network topologies is an open problem. In a general network, finding the constraints on the bandwidth region between different node pairs via a probing technique is needed to be able to decide the number of nodes to access in order to repair the failed node.

## Appendix A
## Proof of Theorem 1

The proof follows a similar line as that in [1]. As in [1], we construct an information flow graph which is a directed acyclic graph, consisting of three kinds of nodes: a single data source $S$, storage nodes $x_{in}^i$, $x_{out}^i$ and data collectors $DC_i$. The single node $S$ corresponds to the source of the original data.

المنارة للاستشارات
www.manaraa.com

Storage node $i$ in the system is represented by a storage input node $x_{in}^i$, and a storage output node $x_{out}^i$; these two nodes are connected by a directed edge $x_{in}^i \to x_{out}^i$ with capacity equal to the amount of data stored at node $i$.

Given the dynamic nature of the storage systems that we consider, the information ow graph also evolves in time. At any given time, each vertex in the graph is either active or inactive, depending on whether it is available in the network. At the initial time, only the source node $S$ is active; it then contacts an initial set of storage nodes, and connects to their inputs ($x_{in}$) with directed edges of innite capacity. From this point onwards, the original source node $S$ becomes and remains inactive. At the next time step, the initially chosen storage nodes become now active; they represent a distributed erasure code, corresponding to the desired state of the system. If a new node $j$ joins the system, it can only be connected with active nodes. If the newcomer $j$ chooses to connect with active storage node $i$, then we add a directed edge from $x_{out}^i$ to $x_{in}^j$, with capacity equal to the amount of information communicated from node $i$ to the newcomer. Finally, a data collector $DC$ is a node that corresponds to a request to reconstruct the data. Data collectors connect to subsets of active nodes through edges with infinite capacity.

An important notion associated with the information flow graph is that of minimum cuts: A (directed) cut in the graph G between the source $S$ and a fixed data collector node $DC$ is a subset $C$ of edges such that, there is no directed path starting from $S$ to $DC$ that does not have one or more edges in $C$. The minimum cut is the cut between $S$ and $DC$ in which the total sum of the edge capacities is smallest.

Following the approach of [1], it is enough to prove the following Lemma.

**Lemma 1.** *Consider any (potentially infinite) information flow graph $G$, formed by having $n$ initial nodes that connect directly to the source and obtain $\alpha$ bits, while additional nodes join the graph by connecting to $d_j \in D$ existing nodes and obtaining $\beta_{d_j}$ bits from each for some $d_j \in D$. Any data collector $t$ that connects to a $k$-subset of "out-nodes" of $G$ must satisfy:*

$$\mathit{mincut}(s,t) \geq \sum_{i=0}^{k-1} \min(\alpha, \min_{d_j \in D}(d_j - i)\beta_{d_j}). \tag{11}$$

*Furthermore, there exists an information flow graph $G^*$ where this bound is matched with equality.*

Let $e_i = \arg\min_{d_j \in D}(d_j - i)\beta_{d_j}$ for $i = 0, 1, \cdots k-1$. For the statement that there exist a flow where the bound is matched with equality, we consider the setup as in Figure 11. In this graph, there are initially $n$ nodes labeled from 1 to $n$. Consider $k$ newcomers labeled as $n+1, \cdots, n+k$. The newcomer node $n+i$ connects to nodes $n+i-e_{i-1}, \cdots, n+i-1$. Consider a data collector $t$ that connects to the last $k$ nodes, i.e., nodes $n+1, \cdots, n+k$, and a cut $(U, \bar{U})$ defined as follows. For each $i \in \{1, \cdots, k\}$, if $\alpha \leq (e_i - 1)\beta_{e_i}$, then we include $x_{out}^{n+i}$ in $\bar{U}$; otherwise, we include $x_{out}^{n+i}$ and $x_{in}^{n+i}$ in $U$. We note that this cut $(U, \bar{U})$ achieves the bound as in the statement of Lemma with equality.

The proof that every cut should satisfy the bound follows very similarly to the proof in [1], using the topological sorting for the graph, and is thus omitted.

## Appendix B
## Proof of Theorem 2

*Proof.* To prove this result, we start with a subset of $D$, say $D' = \{e_1, e_2\}$ for any $\{e_1, e_2\} \subseteq D$ with $e_1 > e_2$. Since $(\alpha, \beta_{e_1}^*(\alpha), \infty)$ satisfy (2), there is a minimum $\beta_{e_2}(\alpha)$ such that $(\alpha, \beta_{e_1}^*(\alpha), \beta_{e_2}(\alpha))$ satisfy (2). We call this minimum $\beta_{e_2}(\alpha)$ as $\widetilde{\beta_{e_2}}(\alpha)$. For $\beta_{e_2}(\alpha) = \widetilde{\beta_{e_2}}(\alpha)$, (2) will be satisfied with equality since if not, the value of $\widetilde{\beta_{e_2}}(\alpha)$ is not optimal.

Thus, we have the following equations

$$\sum_{i=0}^{k-1} \min(\alpha, (e_1 - i)\beta_{e_1}^*(\alpha)) = \mathsf{M}, \tag{12}$$

$$\sum_{i=0}^{k-1} \min(\alpha, (e_1 - i)\beta_{e_1}^*(\alpha), (e_2 - i)\widetilde{\beta_{e_2}}(\alpha)) = \mathsf{M}, \tag{13}$$

Since each term inside the summation in the second expression is at-most that in the first expression, we have

$$(e_2 - i)\widetilde{\beta_{e_2}}(\alpha) \geq \min(\alpha, (e_1 - i)\beta_{e_1}^*(\alpha)), \tag{14}$$

for all $0 \leq i \leq k-1$. Since, $\widetilde{\beta_{e_2}}(\alpha)$ is the minimum possible $\beta_{e_2}(\alpha)$ satisfying the above, we have

$$\widetilde{\beta_{e_2}}(\alpha) = \min_{i=0}^{k-1} \frac{\min(\alpha, (e_1 - i)\beta_{e_1}^*(\alpha))}{e_2 - i} \tag{15}$$

$$= \min_{i=0}^{k-1} \min(\frac{\alpha}{e_2 - i}, \frac{e_1 - i}{e_2 - i}\beta_{e_1}^*(\alpha)) \tag{16}$$

Since both the terms in the minimum increase with $i$, we have that the minimum of these terms is non-decreasing with $i$, and thus

$$\widetilde{\beta_{e_2}}(\alpha) = \min(\frac{\alpha}{e_2 - k + 1}, \frac{e_1 - k + 1}{e_2 - k + 1}\beta_{e_1}^*(\alpha)) \tag{17}$$

Further, we note that $\alpha < (e_1 - k + 1)\beta_{e_1}^*(\alpha)$ is not possible since it violates the optimality of $\beta_{e_1}^*(\alpha)$ and thus, we have

$$\widetilde{\beta_{e_2}}(\alpha) = \frac{e_1 - k + 1}{e_2 - k + 1}\beta_{e_1}^*(\alpha) \tag{18}$$

It now remains to be seen as to when is $\widetilde{\beta_{e_2}}(\alpha) = \beta_{e_2}^*(\alpha)$. If $\widetilde{\beta_{e_2}}(\alpha) = \beta_{e_2}^*(\alpha)$, we have the following

$$\sum_{i=0}^{k-1} \min(\alpha, (e_2 - i)\frac{e_1 - k + 1}{e_2 - k + 1}\beta_{e_1}^*(\alpha)) = \mathsf{M}. \tag{19}$$

Since we know that $\sum_{i=0}^{k-1} \min(\alpha, (e_1 - i)\beta_{e_1}^*(\alpha)) = \mathsf{M}$ and $(e_2 - i)\frac{e_1 - k + 1}{e_2 - k + 1} \geq (e_1 - i)$, we have that

$$\min(\alpha, (e_2 - i)\frac{e_1 - k + 1}{e_2 - k + 1}\beta_{e_1}^*(\alpha)) = \min(\alpha, (e_1 - i)\beta_{e_1}^*(\alpha)), \tag{20}$$

المنارة للاستشارات
www.manaraa.com

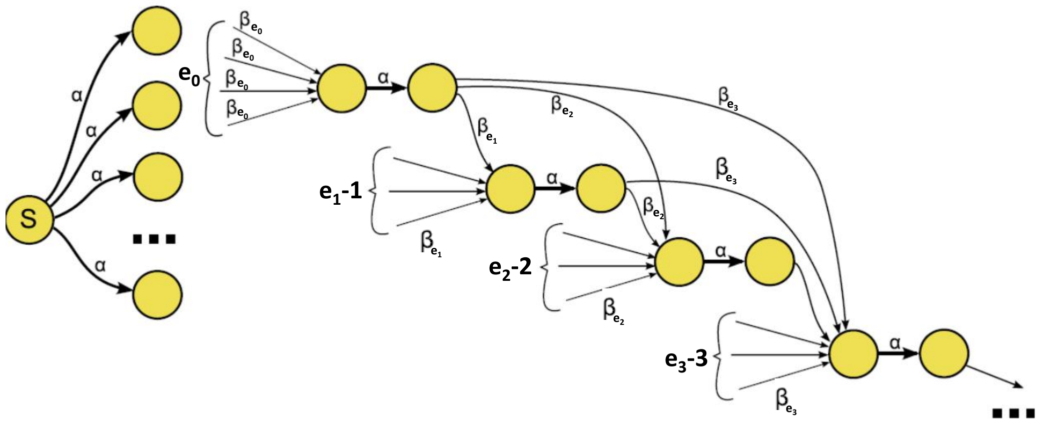

Fig. 11. $G^*$ used in the proof of lemma.

for all $0 \le i \le k-1$. Since for $i = k-1$, the two sides are exactly the same and thus the above holds for $k = 1$. Thus for $k > 1$, we need the two sides to be equal for $0 \le i \le k-2$. Since for $0 \le i < k-1$, $e_1 - i < (e_2 - i)\frac{e_1-k+1}{e_2-k+1}$, we have that the above holds if and only if

$$\alpha \le (e_1 - k + 2)\beta^*_{e_1}(\alpha). \tag{21}$$

From the expression of $\beta^*_{e_1}(\alpha)$, we have that this happens if and only if $\alpha \le \frac{\mathsf{M}(e_1-k+2)}{k(e_1-k+2)-1}$.

Thus, we see that if $\alpha > \frac{\mathsf{M}(d_1-k+2)}{k(d_1-k+2)-1}$, $(\alpha, \beta^*_{d_1}(\alpha), \cdots, \beta^*_{d_l}(\alpha))$ do not satisfy (2). Further, if $\alpha \le \frac{\mathsf{M}(d_1-k+2)}{k(d_1-k+2)-1}$, the same approach shows that $(\alpha, \beta^*_{d_1}(\alpha), \cdots, \beta^*_{d_l}(\alpha))$ satisfies (2). □

## Appendix C
## Proof of Theorem 2

Let $P_c(t)$ denote the probability that $c$ nodes are active at time $t$. The differential equations corresponding to the change of state are given by

$$\frac{dP_n(t)}{dt} = -n\lambda P_n(t) + \frac{k(n-k)}{n-1}\mu P_{n-1}(t) \tag{22}$$

$$\begin{aligned}\frac{dP_{n-c}(t)}{dt} &= (n-c+1)\lambda P_{n-c+1}(t) - ((n-c)\lambda \\ &\quad + \frac{k(n-k-c+1)}{n-c}\mu)P_{n-c}(t) \\ &\quad + \frac{k(n-k-c)}{n-c-1}\mu P_{n-c-1}(t) \\ &\quad (\text{ for } 1 \le c \le n-k+1)\end{aligned} \tag{23}$$

$$\frac{dP_k(t)}{dt} = (k+1)\lambda P_{k+1}(t) - (k\lambda + \mu)P_k(t) \tag{24}$$

$$\frac{dP_{k-1}(t)}{dt} = k\lambda P_k(t). \tag{25}$$

Let $\overline{P}(t) = \begin{bmatrix} P_n(t) \\ \cdots \\ P_{k-1}(t) \end{bmatrix}$.

Solving the differential equations, we have $\overline{P}(t) = \exp(At)\overline{P}(0)$, where $A$ is a tri-diagonal matrix with each column sum as zero, and is given as $A =$

$$\begin{bmatrix} -n\lambda & \frac{k(n-k)}{n-1}\mu & 0 & \cdots & 0 & 0 \\ n\lambda & -(n-1)\lambda - \frac{k(n-k)}{n-1}\mu & \frac{k(n-k)}{n-1}\mu & \cdots & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & -k\lambda - \mu & 0 \\ 0 & 0 & 0 & \cdots & k\lambda & 0 \end{bmatrix}. \tag{26}$$

Since the system starts from the state with all nodes being active, we have $\overline{P}(0)$ being 1 only in the first element, and zero elsewhere. The mean time to data loss is given as

$$MTTDL = \int_{t=0}^{\infty} \sum_{r=k}^{n} P_r(t)dt \tag{27}$$

Note that by Final Value Theorem for Laplace Transform [22], we have

$$MTTDL = \lim_{s \to 0} [111 \cdots 10](sI - A)^{-1}[10 \cdots 0]^T. \tag{28}$$

We let $B \triangleq adj(sI - A)$, where $adj(.)$ represents adjoint of the argument, and let $B_{ij}$ be the element corresponding to $i^{th}$ row and $j^{th}$ column. Then,

$$MTTDL = \lim_{s \to 0} \frac{\sum_{r=1}^{n-k+1} B_{1r}}{\det(sI - A)}. \tag{29}$$

We will now give a result that is a key result in evaluation of the determinant and the adjoints.

**Lemma 2.** *Let $M$ be a $l \times l$ tri-diagonal matrix, such that the $r^{th}$ diagonal element is $c_r\lambda + b_r\mu$, the upper diagonal element $M_{r,r+1} = -b_{r+1}\mu$, and the lower diagonal element $M_{r+1,r} = -c_r\lambda$. Then, the determinant of matrix $M$ is given as*

$$\det(M) = \sum_{i=0}^{l} (b_1 b_2 \cdots b_{l-i})(c_{l-i+1} \cdots c_l)\mu^{l-i}\lambda^i \tag{30}$$

*Proof.* The result can be shown to hold by induction and thus the proof is omitted. □

www.manaraa.com

When we construct matrix $sI-A$, we see that there is no element in the last row or last column except the diagonal element which is $s$. Thus, the determinant of $sI-A$ is given as $s$ times the determinant of the first $n-k+1 \times n-k+1$ matrix. Further, this is equal to $s$ times the determinant of the first $n-k+1 \times n-k+1$ matrix when $s=0 + o(s)$. For $s=0$, the first $n-k+1 \times n-k+1$ is given in the tri-diagonal form in Lemma 2 with $b_1 = 0$, $c_r = (n-r+1)$. Thus, we have

$$\begin{aligned} \det(sI-A) &= sn(n-1)\cdots k\lambda^{n-k+1} + o(s) \\ &= s\frac{n!}{(k-1)!}\lambda^{n-k+1} + o(s) \end{aligned} \tag{31}$$

Using similar approach, we see that

$$\begin{aligned} B_{11} &= s\sum_{i=0}^{n-k}\left(\frac{k(n-k)}{n-1}\frac{k(n-k-1)}{n-1-1}\cdots\right. \\ &\quad \left.\frac{k(n-k-i+1)}{n-i}\right) \\ &\quad ((n-i-1)\cdots k)\,\mu^i\lambda^{n-k-i} + o(s) \qquad (32) \\ &= s\frac{(n-k)!}{(k-1)!(n-1)!}\sum_{i=0}^{n-k}(k\mu)^i\lambda^{n-k-i} \\ &\quad \frac{((n-i-1)!)^2}{(n-k-i)!} + o(s) \qquad (33) \end{aligned}$$

Let $F_n \triangleq \frac{(n-k)!}{(k-1)!(n-1)!}\sum_{i=0}^{n-k}(k\mu)^i\lambda^{n-k-i}\frac{((n-i-1)!)^2}{(n-k-i)!}$. Then, $B_{11} = sF_n + o(s)$. Similarly solving other terms, we have $B_{1(l+1)} = s\frac{n!}{(n-l)!}\lambda^l F_{n-l} + o(s)$ for $l = 0, \cdots n-k$. Thus, the overall mean time to data loss is given as

$$\begin{aligned} MTTDL &= \lim_{s\to 0}\frac{\sum_{r=1}^{n-k+2}B_{1r}}{\det(sI-A)} \qquad (34) \\ &= \lim_{s\to 0}\frac{\sum_{l=0}^{n-k}B_{1(l+1)}}{s\frac{n!}{(k-1)!}\lambda^{n-k+1}+o(s)} \qquad (35) \\ &= \lim_{s\to 0}\frac{s\sum_{l=0}^{n-k}\frac{n!}{(n-l)!}\lambda^l F_{n-l}+o(s)}{s\frac{n!}{(k-1)!}\lambda^{n-k+1}+o(s)} \qquad (36) \\ &= \frac{\sum_{l=0}^{n-k}\frac{n!}{(n-l)!}\lambda^l F_{n-l}}{\frac{n!}{(k-1)!}\lambda^{n-k+1}} \qquad (37) \end{aligned}$$

Solving this expression gives the result as in the statement of the Theorem after some manipulations.

## References

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

[2] R. Ahlswede, Ning Cai, S.-Y.R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Information Theory*, vol. 46, no. 4, pp. 1204-1216, Jul. 2000.

[3] A. G. Dimakis, K. Ramchandran, Y. Wu, C. Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 476-489, Mar. 2011.

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

[5] N. B. Shah, K. V. Rashmi, P. V. Kumar and K. Ramchandran, "Interference alignment in regenerating codes for distributed storage: necessity and code constructions," *IEEE Transactions on Information Theory*, vol. 58, no. 4, pp. 2134-2158, Apr. 2012.

[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," *IEEE Transactions on Information Theory*, vol. 57, no. 8, pp. 5227-5239, Aug. 2011.

[7] 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*, pp. 2974-2987, May 2013.

[8] D. S. Papailiopoulos, A. G. Dimakis, and V. Cadambe, "Repair optimal erasure codes through Hadamard designs," *IEEE Transactions on Information Theory*, pp. 3021-3037, May 2013.

[9] I. Tamo, Z. Wang, and J. Bruck, "MDS array codes with optimal rebuilding," in Proceedings *2011 IEEE International Symposium on Information Theory*, St. Petersberg, Russia, Aug. 2011, pp. 1240-1244.

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

[11] C. Tian, V. Aggarwal, and V. Vaishampayan, "Exact-repair regenerating codes via layered erasure correction and block designs," arXiv:1302.4670.

[12] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz and D. A. Patterson, "RAID: high-performance, reliable secondary storage," Journal of the ACM Volume 26, Issue 2 (1994), p. 145-185.

[13] J. E. Angus, "On computing MTBF for a k-out-of-n: G repairable system," IEEE Transactions on Reliability Volume 37, Number 3 (1988), p. 312-313.

[14] J. Resch and I. Volvovski, "Reliability models for highly fault-tolerant storage systems,"

[15] SpaceMonkey project http://www.spacemonkey.com/

[16] Tahoe: the least-authority file system, https://tahoe-lafs.org/trac/tahoe-lafs

[17] N. Hu, L. Li, Z. M. Mao, P. Steenkiste and J. Wang, "Locating Internet bottlenecks: algorithms, measurements, and implications," in Proc. ACM SIGCOMM, August 2004.

[18] N. Hu and P. Steenkiste, "Exploiting Internet route sharing for large scale available bandwidth estimation," in Proc. 5th ACM SIGCOMM conference on internet measurement, Oct. 2005.

[19] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An efficient scheme for tolerating double disk failures in raid architectures," in IEEE Transactions on Computers, 1995.

[20] A. Kermarrec, G. Straub, and N. Le Scouarnec, "Repairing multiple failures with coordinated and adaptive regenerating codes," arXiv:1102.0204, Feb 2011.

[21] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, "XORing elephants: novel erasure codes for big data," arXiv:1301.3791.

[22] A. D. Poularikas, and S. Seely, "Laplace Transforms." The Transforms and Applications Handbook: Second Edition, Boca Raton: CRC Press LLC, 2000.

المنارة للاستشارات

www.manaraa.com