Irregular Array Codes with Arbitrary Access Sets for Geo-Distributed Storage

Francisco Maturana and K. V. Rashmi
Carnegie Mellon University, Pittsburgh, PA, USA
Email: fmaturan@cs.cmu.edu, rvinayak@cs.cmu.edu

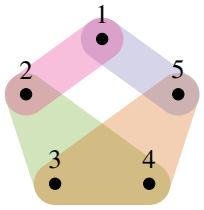
***Abstract*—Distributed storage systems typically use erasure codes to provide tolerance against node failures. An erasure code encodes a message into a codeword made up of several symbols, which are then distributed among nodes in the system. Maximum distance separable (MDS) $[n, k]$ scalar codes are commonly used in practice, which have the property that any subset of k out of n nodes is enough to decode the message. However, in applications such as geo-distributed storage systems, decodability from many of these subsets is unnecessary. In this paper, we study codes where only certain subsets of nodes, named access sets, are required to satisfy decodability.**

Our analysis focuses on two metrics of practical importance: update cost and storage overhead. For minimizing these metrics, we show that it is necessary to employ *irregular array codes*. We derive a lower bound on update cost as a function of the required access sets and show that it is achievable. Existing work provides an achievable lower bound on storage overhead. While both lower bounds are individually achievable, we show that they are not simultaneously achievable in general. Due to the premium in wide-area network bandwidth cost over storage cost, we focus on codes with minimum update cost (termed MUC). Finally, we derive a lower bound on the storage overhead of MUC codes and show the existence of MUC codes meeting this lower bound via a randomized construction. Our results thus show that it is possible to achieve significant savings in update cost and storage overhead by tailoring the design of codes to the required access sets.

I. Introduction

Erasure codes are commonly used in distributed storage systems to provide resiliency against failures. In such applications, an $[n, k]$ block code C is used to encode a message $\mathbf{m}$ consisting of k symbols into a codeword $\mathbf{c}$ consisting of n symbols, where symbols are taken from a finite field $\mathbb{F}_q$ of size q. In a *scalar* code, each codeword symbol is then placed in a different node in the system. *Maximum distance separable* (MDS) codes are widely used in practice, because they require the least amount of storage for a given level of failure tolerance.

Scalar MDS codes have the property that the message $\mathbf{m}$ can be decoded from *any* subset of k out of all n nodes. Some applications, however, require the ability to decode the message from *only a few* of those subsets. In some cases, it may even be desirable to decode the message from certain subsets of size smaller than k. An example of such an application is *geo-distributed storage systems* [1]–[5]. In these systems, data is encoded and distributed across different servers around the globe. Clients in diverse geographical locations then decode and update the data by communicating with these servers. One important constraint is given by the maximum latency facing clients when decoding data (i.e. read latency). Due to the geo-distributed nature of this application, network latency across pairs of clients and servers varies significantly. Thus each client is only able to communicate with a small subset of nearby servers under a given read latency threshold. On top of this, one wants to provide certain failure tolerance guarantees, such as ensuring that each client can decode the data even if one of the nearby servers fails. This leads to the requirement that clients must be able to decode data from specific subsets of nodes.



(a)

node 1	a	b
node 2	c	
node 3	a	
node 4	b	
node 5	c	

(b)

Fig. 1: (a) Example of arbitrary access sets over five nodes and (b) an irregular array code that satisfies them.

We formalize such constraints on decodability through the notion of access sets. An *access set* is a subset of nodes $S \subseteq \{1, \ldots, n\}$ expressing the requirement that the message $\mathbf{m}$ must be decodable using only symbols stored in the nodes in S. We say that a collection of access sets $\mathbb{S}$ is satisfied by a code if the message $\mathbf{m}$ is decodable from each of the access sets in $\mathbb{S}$. Note that there can be other subsets not in $\mathbb{S}$ which are also sufficient for decoding $\mathbf{m}$. Our goal is to design a code satisfying the given access sets while minimizing other cost metrics.

Existing work on codes with arbitrary access sets [6], [7] has focused on minimizing *storage overhead*, i.e. the ratio between the total number of symbols in a codeword and the number of symbols in the message it encodes. In this paper, we focus on an additional metric of practical importance: update cost. The *update cost* of a code is the average number of symbols communicated when a single symbol of the message is updated. In the setting of geo-distributed storage systems, these transmissions would consume the *wide-area network (WAN) bandwidth* which is a scarce and expensive resource in distributed systems [8]. Thus, update cost is an important metric to minimize in geo-distributed storage systems.

When the access sets correspond to all $\binom{n}{k}$ subsets of size k, it can be shown from basic results about MDS codes that the

This work was funded by an NSF CNS grant (CNS-1901410).

2021 IEEE International Symposium on Information Theory (ISIT) | 978-1-5386-8209-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISIT45174.2021.9517809

minimum storage overhead is n/k and the minimum update cost is $(n-k+1)$, both of which are simultaneously achieved by systematic $[n,k]$ Reed-Solomon codes (or other MDS codes).

Towards the goal of optimizing these two metrics in geo-distributed storage systems, we ask the following question: When the required access sets are more relaxed than all subsets of size k, *is it possible to reduce update cost and storage overhead simultaneously by tailoring the code construction to the access sets?* We answer this question in the affirmative.

Consider the following toy example.

Example 1: Consider $n = 5$ nodes and access sets $\mathbb{S} = \{\{1,2\},\{1,5\},\{2,3,4\},\{3,4,5\}\}$ (see Fig. 1a). A code that satisfies $\mathbb{S}$ is a systematic $[5,2]$ Reed-Solomon code, with update cost 4 and storage overhead $5/2 = 2.5$. Another code that satisfies $\mathbb{S}$ is the irregular array code shown on Fig. 1b, which encodes the message $\mathbf{m} = (a,b,c)$. Note that $\mathbf{m}$ can be fully decoded from any of the access sets in $\mathbb{S}$, and that each message symbol is present in only two different nodes. Thus, this code has update cost 2 and storage overhead $6/3 = 2$. ▸

In Example 1, it was possible to reduce storage overhead by placing data unevenly on nodes. This kind of code is called an *irregular array code*, which allows for the storage of a variable number of symbols on each node, in contrast to scalar codes which only store a single symbol per node. Note also that in Example 1 the sparsity of the access sets makes it possible to reduce update cost.

We study the problem of designing codes that satisfy the given access sets while minimizing update cost and storage overhead. We start by presenting the problem formally and exploring its fundamental limits (§III). We do this by first deriving the minimum update cost achievable by an irregular array code satisfying the given access sets (§III-A). We then focus on analyzing update cost and storage overhead in conjunction, and demonstrate that *employing irregular array codes is necessary* for jointly minimizing both of these metrics (§III-B). We also show that, unlike the case where all subsets of size k are access sets, it is not always possible to simultaneously achieve the minimum update cost and minimum storage overhead (§III-C). Since the cost of WAN bandwidth tends to be higher than the cost of storage [8], [9], we focus on codes with minimum update cost (termed MUC) and minimize storage overhead subject to this constraint. We model MUC codes using information flow graphs, and use this model to derive a lower bound on their storage overhead (§IV). Finally, we show the existence of MUC codes meeting this lower bound through a randomized construction (§IV). Overall, the results show that it is possible to obtain significant savings in update cost and storage overhead compared to traditional MDS codes when one adapts the design of a code to the given access sets. This paper also exposes a new trade-off space between update cost and storage overhead under arbitrary access sets, which is of significance in geo-distributed storage systems.

II. Related work and existing results

In this section, we summarize the related work and review existing results which will be used in this paper.

A. Related work

The concept of codes with arbitrary access sets has been previously studied in the information dispersal and secret sharing literatures. *Information dispersal* [6], [7], [10] studies the problem of encoding and distributing a given file f among nodes in a way that satisfies prespecified access sets. While [6], [7] study the minimum storage overhead required by arbitrary access sets, to the best of our knowledge, work in the information dispersal literature has not focused on update cost. *Secret sharing with general access structures* [11] considers the same scenario as information dispersal but adds a security requirement: any subset of nodes that is not an access set leaks no information about f. Security is not among the objectives of this paper. Gonen et al. [12] consider collections of access sets which are restricted to be of the same size k and study the field size requirement of scalar codes satisfying them.

Irregular array codes have also been used by [13]–[15] in a line of work called *irregular MDS array codes*. In this setting, the following parameters are given as input: the number of nodes n, the number of message symbols m_i stored in node $i \in \{1,\ldots,n\}$, and a number k such that *all* message symbols can be decoded from any k nodes. The goal in these works is: 1) to determine the necessary number of parity symbols p_i stored in each node i while minimizing the total number of parity symbols $\sum_{i=1}^{n} p_i$, and 2) to design a code that stores m_i message symbols and p_i parity symbols in node i such that the message can be decoded from any k nodes.

Several works have studied the cost of updates in storage codes via different metrics, such as update complexity, update efficiency, and update bandwidth. *Update complexity* [16]–[30] is defined as the average number of codeword symbols updated when a single message symbol is updated. In linear codes, this is related to the fraction of non-zero entries (i.e. density) of the generator matrix. *Update efficiency* [31]–[33] refers to the asymptotic behavior of update complexity. *Update bandwidth* [15] assumes a systematic irregular array code, and is the average amount of symbols communicated among nodes when *all* message symbols stored in a single node are updated. All these metrics differ from the update cost considered in our work, which is defined as the average number of symbols communicated when a single message symbol is updated. Update complexity and update efficiency focus on the number of symbols *updated*, whereas update cost focuses on the number of symbols *communicated* (when nodes store multiple symbols, a single communicated symbol can be used to update several symbols at a node). We focus on the number of symbols communicated rather than updated in order to capture the usage of WAN bandwidth in geo-distributed storage systems. Update bandwidth also focuses on the number of symbols communicated, but is defined for systematic codes only and is motivated by a setting where data is generated *at* the nodes, which is not a good fit for our target application of geo-distributed storage systems. Several other works have studied handling updates in storage systems in different settings, such as multiversion coding [34] and oblivious updates [35].

The design of erasure codes for distributed storage systems has also been studied by other lines of research with the goal of optimizing other metrics. For example, regenerating codes (e.g. [29], [36]–[42]) minimize the bandwidth cost of node repair, locally repairable codes (e.g. [43]–[47]) reduce the number of nodes that must participate in the repair of a single node, and Piggyback codes [48]–[50] construct codes to reduce the amount of data read and downloaded during repair while having a low number of symbols per node (i.e., substripes).

B. Existing results on storage overhead for arbitrary access sets

In this subsection, we summarize results from previous works that are used in this paper. Naor and Roth [6] proved a lower bound on the minimum storage overhead of a code satisfying given access sets $\mathbb{S}$. Let $[n] = \{1, \ldots, n\}$. Each node $v \in [n]$ is modeled as a variable $w_v \in [0, 1]$ denoting the size of node v as a fraction of the size of the message $\mathbf{m}$. One clear restriction is that the combined size of the nodes in an access set must be at least that of the message. Therefore, a lower bound on storage overhead is given by the solution to the following linear program (LP).

$$\begin{array}{lll} \text{minimize} & \sum_{v\in[n]} w_v & \\ \text{subject to} & \sum_{v\in S} w_v \geq 1, & \forall S \in \mathbb{S} \\ & w_v \in [0,1], & \forall v \in [n] \end{array} \tag{LP1}$$

Theorem 1 [6], [7]: The minimum storage overhead of irregular array codes satisfying access sets $\mathbb{S}$ is given by LP1 and can be achieved by using a sufficiently long MDS code and distributing symbols according to the weights $\{w_v\}_{v\in[n]}$. ∎

III. Fundamental limits on codes with arbitrary access sets

An irregular array code over finite field $\mathbb{F}_q$ with n nodes, message of length k, and node sizes $(\ell_i)_{i\in[n]}$ is defined by a mapping $C : \mathbb{F}_q^k \to \mathbb{F}_q^N$, where $N = \sum_{i\in[n]} \ell_i$ is the length of a codeword. An irregular array code is said to be linear if C is linear. Each codeword $C(\mathbf{m}) = (c_1, \ldots, c_N)$ is interpreted as an n-node array, with node i denoted as $C_i(\mathbf{m}) = (c_{(\ell_i^<+1)}, \ldots, c_{(\ell_i^<+\ell_i)})$ where $\ell_i^< = \sum_{j=1}^{i-1} \ell_j$. Let the symbol $*$ denote an erasure and, for a subset $S \subseteq [n]$, let $C_S(\mathbf{m})$ denote the result of erasing every symbol in $C_i(\mathbf{m})$ for $i \in [n] \setminus S$. We say that an irregular array code satisfies access sets $\mathbb{S} \subseteq 2^{[n]}$ if $\mathbf{m}$ can be decoded from $C_S(\mathbf{m})$, i.e., there exists a decoding function $D : (\mathbb{F}_q \cup \{*\})^N \to \mathbb{F}_q^k$ such that $D(C_S(\mathbf{m})) = \mathbf{m}$ for all $\mathbf{m} \in \mathbb{F}_q^k$ and $S \in \mathbb{S}$. Note that if it is possible to decode $\mathbf{m}$ from an access set S, then it is possible to decode $\mathbf{m}$ from any access set $S' \supseteq S$. Thus, we assume without loss of generality that all subsets in $\mathbb{S}$ are minimal, and thus $|\mathbb{S}| \leq \binom{n}{\lfloor n/2 \rfloor}$ (by Sperner's theorem [51]). The *storage overhead* of an irregular array code is defined as the ratio N/k. We define the *update cost* of an irregular array code as the average number of symbols communicated to nodes when a *single* symbol of the message is updated. In general, update cost is at least the average number of nodes that are updated when a single message symbol is updated, since at least one symbol must be communicated to each updated node. In linear codes, both the number of symbols communicated and the number of nodes updated are equivalent because if message symbol m_i is updated to m_i' it suffices to send symbol $\Delta m_i = (m_i' - m_i)$ to every updated node. Each node then scales Δm_i by the appropriate factor and updates its symbols. Thus, for linear codes update cost is:

$$\text{update-cost}(C) := \frac{\sum_{i=1}^{k} |\{j \in [n] : C_j(\mathbf{e}_i) \neq \mathbf{0}\}|}{k},$$

where $\mathbf{e}_i \in \mathbb{F}_q^k$ is the i-th standard basis vector. When $\ell_i = \ell$ for all $i \in [n]$ we say that the code is a regular array code, and when $\ell = 1$ we say that it is a scalar code.

Our ultimate goal is to construct codes that minimize update cost and storage overhead while satisfying the given access sets $\mathbb{S}$. We begin by studying update cost in isolation (§III-A). Then, we study both update cost and storage overhead in conjunction (§III-C). Along the way, we show that using irregular array codes is necessary for minimizing these two metrics (§III-B).

A. Minimum update cost

Now, we derive a lower bound on the update cost of a code satisfying the given access sets $\mathbb{S}$. To achieve this, we model each node $v \in [n]$ with a binary variable $\dot{w}_v \in \{0, 1\}$ indicating whether the node is updated or not when updating any single arbitrarily chosen message symbol. Observe that if a symbol of the message is updated, then at least one node in each access set $S \in \mathbb{S}$ has to be updated, since otherwise the output of the decoding function on this access set would remain unchanged. Thus, one can compute a lower bound on the number of nodes updated by a single symbol update and, by extension, a lower bound on update cost, through the following integer program (IP).

$$\begin{array}{lll} \text{minimize} & \sum_{v\in[n]} \dot{w}_v & \\ \text{subject to} & \sum_{v\in S} \dot{w}_v \geq 1, & \forall S \in \mathbb{S} \\ & \dot{w}_v \in \{0,1\}, & \forall v \in [n] \end{array} \tag{IP1}$$

Note that this formulation corresponds to computing a set cover of the access sets by nodes, where a node v is said to cover access set S if $v \in S$.

Lemma 2: IP1 gives a lower bound on the update cost of irregular array codes satisfying access sets $\mathbb{S}$. ∎

We show that this bound is achievable via strategic replication.

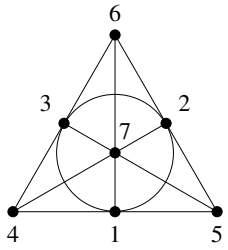
Lemma 3: The bound of Lemma 2 is achievable.

Proof: Let $\{\dot{w}_v^*\}_{v\in[n]}$ be the optimal solution to IP1. Consider a code that places a full copy of the message $\mathbf{m}$ on each node v where $\dot{w}_v^* = 1$. Clearly, this code achieves the minimum update cost and satisfies the access sets $\mathbb{S}$. ∎

The next theorem follows from Lemmas 2 and 3.

Theorem 4: The minimum update cost of an irregular array code satisfying access sets $\mathbb{S}$ is given by IP1 and is achieved by strategic replication. ∎

Minimum update sets or μ-sets: IP1 may have multiple optimal solutions for the given access sets $\mathbb{S}$. Each optimal solution can be interpreted as a subset of nodes U where $v \in U$ iff $\dot{w}_v = 1$. We call such subsets of nodes a *minimum update set* or *μ-set*, and denote the collection of all μ-sets for the given access sets as $\mathcal{U}$. Note that a code can have minimum update cost only if every update to a message symbol updates



(a)

node 1	a	b
node 2	c	d
node 3	e	f
node 4	$a+e$	$d+e$
node 5	$a+f$	$c+f$
node 6	$b+e$	$c+e$
node 7	$b+f$	$d+f$

(b)

Fig. 2: (a) Access sets over seven nodes defined by the Fano plane and (b) an array code that satisfies it.

a number of nodes equal to the minimum update cost. Because of this, μ-sets are important for studying codes with minimum update cost, and each message symbol must be associated to a specific μ-set that is updated when that message symbol is updated. As a consequence, in codes with minimum update cost, a node v depends on a certain message symbol iff it belongs to its corresponding μ-set. For example, in Example 1, one can verify that the minimum update cost is 2, and that each message symbol updates a μ-set: a updates $\{1,3\}$, b updates $\{1,4\}$, and c updates $\{2,5\}$. Note that the size of $\mathcal{U}$ can be exponential in n and, like $\mathbb{S}$, it is upper bounded by $\binom{n}{\lfloor n/2 \rfloor}$.

B. The necessity of irregular array codes

In this subsection, we show that considering irregular array codes (instead of traditional scalar codes) is not only important for the sake of generality, but also a necessity for reducing both update cost and storage overhead.

Lemma 5: Irregular array codes are necessary for achieving the minimum storage overhead of arbitrary access sets.

Proof: The proof proceeds by contradiction using an example. Consider Example 1. Notice that any coding scheme that places the same number of symbols in each node must place at least $k/2$ symbols on each node, due to decoding set $\{1,2\}$ and Theorem 1. This results in an storage overhead of at least 2.5. On the other hand, the code proposed in Example 1 achieves the minimum storage overhead, which is 2 by Theorem 1 (consider the solution $w_1 = 2/3$ and $w_v = 1/3$ for $v \in \{2,\ldots,5\}$). This means that storing a different amount of symbols in each node is necessary for minimizing storage overhead. ■

In general, irregular array codes tend to achieve better storage overhead than scalar codes when the access sets are of different sizes, and when nodes belong to different number of access sets. These two situations arise naturally in geo-distributed storage systems because of the difference in density of servers in distinct regions. Note that existing codes for arbitrary access sets with reduced storage overhead compared to MDS codes (from the literature on information dispersal [6], [7], [10] and secret sharing [11]) are indeed irregular array codes.

Lemma 6: Even for access sets where scalar codes can achieve the minimum storage overhead, array codes are necessary for additionally minimizing the update cost.

We use the next example in the proof to this lemma.

Example 2: Consider $n = 7$ and the access sets $\mathbb{S}$ defined by the Fano plane, where each subset of three nodes is in $\mathbb{S}$ iff they lie on the same line (see Fig. 2a). The minimum storage overhead for $\mathbb{S}$ is $7/3$, by Theorem 1. The minimum update cost is 3, by Theorem 4 (every line is a μ-set). The access sets $\mathbb{S}$ are satisfied by a systematic $[7,3]$ Reed-Solomon code, which has the minimum storage overhead and its update cost is 5 (higher than the minimum update cost). The access sets $\mathbb{S}$ are also satisfied by the irregular array code shown in Fig. 2b, which encodes the message $\mathbf{m} = (a,b,c,d,e,f)$, and has the minimum storage overhead and the minimum update cost. ▸

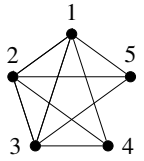


Fig. 3: Example of access sets for which minimum update cost and storage overhead cannot be simultaneously achieved.

Proof of Lemma 6: The proof proceeds by contradiction using an example. Consider Example 2. For these access sets, no code which places a single symbol per node and has minimum storage overhead can achieve the minimum update cost, as explained below. Clearly, the μ-sets associated with the message symbols must cover all nodes, as otherwise uncovered nodes would never be updated. Thus, the code must have $k = 3$, since at least three μ-sets are needed to cover every node, and there are exactly three nodes in each access set. However, for these access sets, any triple of μ-sets that covers all nodes must intersect at exactly one node. No code that places a single symbol in each node can satisfy the access sets in such a triple, since two of the nodes in it would be a function of the same message symbol, and the remaining node would be a function of all three message symbols. ■

C. Tradeoff of update cost vs. storage overhead

So far, we looked at update cost and storage overhead separately, and saw how to construct codes that achieve the minimum cost possible on each metric separately. However, it is easy to see that while the constructions in Theorem 1 and Lemma 3 achieve the minimum cost with respect to one metric, they do not perform well with respect to the other. Therefore, it is a natural question to ask whether it is possible to construct codes that minimize both of these metrics at the same time. For some collections of access sets, it is possible to simultaneously achieve both the minimum update cost and minimum storage overhead. For instance, the collections of access sets discussed in Examples 1 and 2 both have this property. As another example, the access sets consisting of all size k subsets of $[n]$ are satisfied by a systematic $[n,k]$ MDS code, which achieves the minimum update cost $(n-k+1)$ and minimum storage overhead (n/k). However, as the next example shows, this is impossible for some collections of access sets and there is a tradeoff between update cost and storage overhead.

Example 3: Consider $n = 5$ and access sets $\mathbb{S} = \{\{i,j\} : i \neq j \in [5]\} \setminus \{\{4,5\}\}$ (see Fig. 3). From Theorem 1, it follows

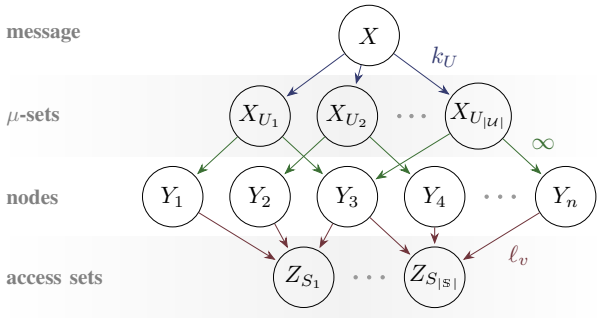


Fig. 4: Information flow graph for given access sets.

that the minimum storage overhead for $\mathbb{S}$ is $5/2$. Here, the only μ-set is $U = \{1, 2, 3\}$, since any other subset of at most three nodes leaves at least one access set uncovered. Thus the minimum update cost is 3.

Any MUC code cannot place any symbols in node 4 or 5, since any update to those nodes would require updating more nodes than the minimum. Since $\{1, 4\}$, $\{2, 4\}$, and $\{3, 4\}$ are access sets and 4 is empty, 1, 2, and 3 each must have at least a full copy of message $\mathbf{m}$. This requires storage overhead of at least 3, which is higher than the minimum. ▸

Since, in general, it is impossible to achieve the minimum cost of both metrics simultaneously, and due to the premium in WAN bandwidth cost over storage cost, we focus on codes with minimum update cost and then minimize the storage overhead subject to that constraint. This approach attains one of the Pareto-optimal points in the tradeoff.

Definition 1 (MUC code): An irregular array code satisfying access sets $\mathbb{S}$ is said to be a minimum update cost (MUC) code if it achieves the minimum update cost (Theorem 4) corresponding to $\mathbb{S}$.

IV. Storage overhead of MUC codes: lower bound and achievability

In this section, we focus on deriving a lower bound on the storage overhead of MUC codes. In order to derive a lower bound on storage overhead, we model the decoding process as a network information flow graph [52].

Recall from §III-A that in MUC codes every message symbol is associated to a μ-set $U \in \mathcal{U}$ that is updated whenever that message symbol is updated. For given access sets $\mathbb{S}$, we build an information flow graph with the following vertices:

- X, for the message $\mathbf{m}$;
- $\{X_U\}_{U\in\mathcal{U}}$, for the fraction of $\mathbf{m}$ encoded in μ-set U;
- $\{Y_v\}_{v\in[n]}$, for the contents of node v;
- $\{Z_S\}_{S\in\mathbb{S}}$, for the decoding of access set S.

The graph also contains the following directed edges:

- $\{(X, X_U)\}_{U\in\mathcal{U}}$, where (X, X_U) has capacity k_U;
- $\{(X_U, Y_v) : v \in U\}_{U\in\mathcal{U}}$, each with unlimited capacity;
- $\{(Y_v, Z_S) : v \in S\}_{S\in\mathbb{S}}$, where (Y_v, Z_S) has capacity ℓ_v.

A necessary condition for a MUC code with parameters n, $k = \sum_{U\in\mathcal{U}} k_U$, and $(\ell_i)_{i\in[n]}$ to exist is that in its information flow graph the maximum flow from the source to each sink is at least k. Therefore, the values of $\{k_U\}_{U\in\mathcal{U}}$ and $\{\ell_v\}_{v\in[n]}$ must be set in order to allow the flows while minimizing the ratio $\sum_{v\in[n]} \ell_v / \sum_{U\in\mathcal{U}} k_U$, which corresponds to the storage overhead.

In order to model the information flow graph, we introduce the following variables:

- $x_U \in [0, 1]$ for $U \in \mathcal{U}$ representing the fraction of message $\mathbf{m}$ associated with μ-set U, i.e., $x_U := k_U/k$;
- $y_v \in [0, 1]$ for $v \in [n]$ representing the size of node v as a fraction of the size of the message $\mathbf{m}$, i.e., $y_v := \ell_v/k$;
- $z_{U,v,S} \in [0, 1]$ representing the flow from μ-set U through node v when access set S is used as the sink.

The following LP captures the information flow graph:

$$
\begin{aligned}
&\text{minimize} \quad \textstyle\sum_{v\in[n]} y_v \\
&\text{subject to:} \\
&\textstyle\sum_{U\in\mathcal{U}} x_U = 1 \\
&z_{U,v,S} \le \mathbb{1}\{v \in (U \cap S)\}, && \forall U \in \mathcal{U},\ v \in [n],\ S \in \mathbb{S} \\
&x_U = \textstyle\sum_{v\in[n]} z_{U,v,S}, && \forall U \in \mathcal{U},\ S \in \mathbb{S} \qquad \text{(LP2)} \\
&\textstyle\sum_{U\in\mathcal{U}} z_{U,v,S} \le y_v, && \forall v \in [n],\ S \in \mathbb{S} \\
&x_U \in [0, 1], && \forall U \in \mathcal{U} \\
&y_v \in [0, 1], && \forall v \in [n] \\
&z_{U,v,S} \in [0, 1], && \forall U \in \mathcal{U},\ v \in [n],\ S \in \mathbb{S}
\end{aligned}
$$

where $\mathbb{1}\{\cdot\}$ is equal to 1 if the condition inside the braces is true, and 0 otherwise.

Theorem 7: For a MUC code satisfying access sets $\mathbb{S}$, LP2 gives a lower bound on the storage overhead.

Proof: Let f_S be the flow of size k from source X to sink Z_S, where $f_S(u, v)$ denotes the flow from vertex u to v. Clearly, because $k = \sum_{U\in\mathcal{U}} k_U$, it must be the case that $f_S(X, X_U) = k_U$ for all $U \in \mathcal{U}$. Similarly, due to the conservation of flow on the X_U vertices, it must hold that $\sum_{v\in S} f_S(X_U, Y_v) = k_U$. Finally, due to the conservation of flow on the Y_v vertices, it must hold that $\sum_{U\in\mathcal{U}} f_S(X_U, Y_v) = f_S(Y_v, Z_S) \le \ell_v$. By defining $z_{U,v,S} := f_S(X_U, Y_v)/k$ and substituting these inequalities with the relevant variables, we obtain the constraints of LP2. A storage overhead lower bound can thus be obtained by specifying storage overhead as the minimization objective. ■

Now, we show that MUC codes achieving the lower bound of Theorem 7 exist over finite fields of size $q > |\mathbb{S}|$. The key idea is to construct a generator matrix with carefully chosen entries set to zero in order to ensure that the code has minimum update cost, and random entries elsewhere. When the field size is large enough, the probability that the constructed code satisfies $\mathbb{S}$ is strictly greater than zero, thus showing that such codes exist.

Theorem 8: MUC codes over $\mathbb{F}_q$ satisfying given access sets $\mathbb{S}$ with storage overhead matching the lower bound of Theorem 7 exist for $q > |\mathbb{S}|$.

Proof sketch: Given a solution to LP2, one can construct a generator matrix $\mathbf{G}$ where the number of rows is given by the x_U variables, the number of columns is given by the y_v variables, and the location of non-zero entries is given by the $z_{U,v,S}$ variables. Let $\mathbf{G}_S$ be the submatrix associated to access set $S \in \mathbb{S}$. The LP constraints ensure that the probability that each $\mathbf{G}_S$ is not full-rank is at most q^{-1}. Thus, by union bound, the probability that any $\mathbf{G}_S$ is not full-rank is at most $|\mathbb{S}|q^{-1}$, which is less than one if $q > |\mathbb{S}|$. ■

REFERENCES

[1] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. C. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford, "Spanner: Google's globally distributed database," *ACM Trans. Comput. Syst.*, 2013.

[2] M. Subramanian, W. Lloyd, S. Roy, C. Hill, E. Lin, W. Liu, S. Pan, S. Shankar, S. Viswanathan, L. Tang, and S. Kumar, "f4: Facebook's warm BLOB storage system," in *11th USENIX Symp. Operating Syst. Design and Implementation, OSDI '14, Broomfield, CO, USA, October 6-8*, 2014.

[3] H. C. H. Chen, Y. Hu, P. P. C. Lee, and Y. Tang, "NCCloud: a network-coding-based storage system in a cloud-of-clouds," *IEEE Trans. Comput.*, 2014.

[4] Y. L. Chen, S. Mu, J. Li, C. Huang, J. Li, A. Ogus, and D. Phillips, "Giza: erasure coding objects across global data centers," in *2017 USENIX Annual Tech. Conf., USENIX ATC 2017, Santa Clara, CA, USA, July 12-14*, 2017.

[5] M. Uluyol, A. Huang, A. Goel, M. Chowdhury, and H. V. Madhyastha, "Near-optimal latency versus cost tradeoffs in geo-distributed storage," in *17th USENIX Symp. Netw. Syst. Des. and Implementation, NSDI 2020, Santa Clara, CA, USA, February 25-27*, 2020.

[6] M. Naor and R. M. Roth, "Optimal file sharing in distributed networks," *SIAM J. Comput. (SICOMP)*, 1995.

[7] P. Béguin and A. Cresti, "General information dispersal algorithms," *Theor. Comput. Sci.*, 1998.

[8] "Bandwidth pricing details," [Online] https://web.archive.org/web/20210503232035/https://azure.microsoft.com/en-us/pricing/details/bandwidth/, accessed: 2021-05-03.

[9] "Azure storage overview pricing," [Online] https://web.archive.org/web/20210503231640/https://azure.microsoft.com/en-us/pricing/details/storage/, accessed: 2021-05-03.

[10] A. D. Santis and B. Masucci, "On information dispersal algorithms," in *IEEE ISIT 2002, Lausanne, Switzerland, July 30-July 5*, 2002.

[11] A. Beimel, "Secret-sharing schemes: A survey," in *Coding and Cryptology - Third Int. Workshop, IWCC 2011, Qingdao, China, May 30-June 3*, ser. Lecture Notes in Computer Science. Springer, 2011.

[12] M. Gonen, I. Haviv, M. Langberg, and A. Sprintson, "Minimizing the alphabet size of erasure codes with restricted decoding sets," in *IEEE ISIT 2020, Los Angeles, CA, USA, June 21-26*, 2020.

[13] C. Chen, S. Lin, and N. Yu, "Irregular MDS array codes with fewer parity symbols," *IEEE Commun. Lett.*, 2019.

[14] F. Tosato and M. Sandell, "Irregular MDS array codes," *IEEE Trans. Inf. Theory*, 2014.

[15] Z. Li and S. Lin, "Update bandwidth for distributed storage," in *IEEE ISIT 2019, Paris, France, July 7-12*, 2019.

[16] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures," *IEEE Trans. Comput.*, 1995.

[17] L. Xu and J. Bruck, "X-code: MDS array codes with optimal encoding," *IEEE Trans. Inf. Theory*, 1999.

[18] M. Blaum, J. Bruck, and A. Vardy, "MDS array codes with independent parity symbols," *IEEE Trans. Inf. Theory*, 1996.

[19] M. Blaum and R. M. Roth, "On lowest density MDS codes," *IEEE Trans. Inf. Theory*, 1999.

[20] J. Hartline, T. Kanungo, and J. Hafner, "R5X0: an efficient high distance parity-based code with optimal update complexity," *IBM Research Report*, vol. RJ 10322, no. A0408-005, 01 2004.

[21] C. Jin, H. Jiang, D. Feng, and L. Tian, "P-Code: a new RAID-6 code with optimal properties," in *Proc. 23rd Int. Conf. Supercomputing, Yorktown Heights, NY, USA, June 8-12, 2009*. ACM, 2009.

[22] Z. Huang, H. Jiang, K. Zhou, C. Wang, and Y. Zhao, "XI-Code: a family of practical lowest density MDS array codes of distance 4," *IEEE Trans. Commun.*, 2016.

[23] S. Lin, G. Wang, D. S. Stones, X. Liu, and J. Liu, "T-Code: 3-erasure longest lowest-density MDS codes," *IEEE J. Sel. Areas Commun.*, 2010.

[24] M. Li and J. Shu, "On cyclic lowest density MDS array codes constructed using starters," in *IEEE ISIT 2010, Austin, Texas, USA, June 13-18*, 2010.

[25] Y. Cassuto and J. Bruck, "Cyclic lowest density MDS array codes," *IEEE Trans. Inf. Theory*, 2009.

[26] E. Louidor and R. M. Roth, "Lowest density MDS codes over extension alphabets," *IEEE Trans. Inf. Theory*, 2006.

[27] Z. Huang, H. Jiang, K. Zhou, Y. Zhao, and C. Wang, "Lowest density MDS array codes of distance 3," *IEEE Commun. Lett.*, 2015.

[28] Z. Huang, H. Jiang, and N. Xiao, "Efficient lowest density MDS array codes of column distance 4," in *IEEE ISIT 2017, Aachen, Germany, June 25-30*, 2017.

[29] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: MDS array codes with optimal rebuilding," *IEEE Trans. Inf. Theory*, 2013.

[30] M. Ye and A. Barg, "Explicit constructions of high-rate MDS array codes with optimal repair bandwidth," *IEEE Trans. Inf. Theory*, 2017.

[31] N. P. Anthapadmanabhan, E. Soljanin, and S. Vishwanath, "Update-efficient codes for erasure correction," in *Allerton*, 2010.

[32] A. Mazumdar, G. W. Wornell, and V. Chandar, "Update efficient codes for error correction," in *IEEE ISIT 2012, Cambridge, MA, USA, July 1-6*, 2012.

[33] A. Mazumdar, V. Chandar, and G. W. Wornell, "Update-efficiency and local repairability limits for capacity approaching codes," *IEEE J. Sel. Areas Commun.*, 2014.

[34] Z. Wang and V. R. Cadambe, "Multi-version coding - an information-theoretic perspective of consistent distributed storage," *IEEE Trans. Inf. Theory*, 2018.

[35] P. Nakkiran, N. B. Shah, and K. Rashmi, "Fundamental limits on communication for oblivious updates in storage networks," in *2014 IEEE Global Commun. Conf.* IEEE, 2014.

[36] A. G. Dimakis, B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, 2010.

[37] 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 Trans. Inf. Theory*, 2011.

[38] 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 Trans. Inf. Theory*, 2012.

[39] C. Suh and K. Ramchandran, "Exact-repair MDS code construction using interference alignment," *IEEE Trans. Inf. Theory*, Mar. 2011.

[40] D. Papailiopoulos, A. Dimakis, and V. Cadambe, "Repair optimal erasure codes through Hadamard designs," *IEEE Trans. Inf. Theory*, May 2013.

[41] S. Goparaju, A. Fazeli, and A. Vardy, "Minimum storage regenerating codes for all parameters," *IEEE Trans. Inf. Theory*, 2017.

[42] M. Ye and A. Barg, "Explicit constructions of optimal-access MDS codes with nearly optimal sub-packetization," *IEEE Trans. Inf. Theory*, 2017.

[43] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Trans. Inf. Theory*, 2012.

[44] N. Prakash, G. M. Kamath, V. Lalitha, and P. V. Kumar, "Optimal linear codes with a local-error-correction property," in *IEEE ISIT 2012, Cambridge, MA, USA, July 1-6*, 2012.

[45] A. S. Rawat, O. O. Koyluoglu, N. Silberstein, and S. Vishwanath, "Optimal locally repairable and secure codes for distributed storage systems," *IEEE Trans. Inf. Theory*, 2013.

[46] I. Tamo and A. Barg, "A family of optimal locally recoverable codes," *IEEE Trans. Inf. Theory*, 2014.

[47] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," *IEEE Trans. Inf. Theory*, 2014.

[48] K. V. Rashmi, N. B. Shah, and K. Ramchandran, "A piggybacking design framework for read-and download-efficient distributed storage codes," *IEEE Trans. Inf. Theory*, 2017.

[49] R. Hulett and M. Wootters, "Limitations of piggybacking codes with low substriping," in *Allerton*. IEEE, 2017, pp. 1131–1138.

[50] A. S. Rawat, I. Tamo, V. Guruswami, and K. Efremenko, "MDS code constructions with small sub-packetization and near-optimal repair bandwidth," *IEEE Trans. Inf. Theory*, 2018.

[51] S. Jukna, *Extremal Combinatorics - With Applications in Computer Science*, ser. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2011.

[52] R. W. Yeung, *A First Course in Information Theory*. Boston, MA: Springer US, 2002.