Bandwidth Cost of Code Conversions in Distributed Storage: Fundamental Limits and Optimal Constructions

Francisco Maturana, *Student Member, IEEE*, and K. V. Rashmi, *Member, IEEE*

***Abstract*— Erasure codes have become an integral part of distributed storage systems as a tool for providing data reliability and durability under the constant threat of device failures. In such systems, an $[n, k]$ code over a finite field $\mathbb{F}_q$ encodes k message symbols from $\mathbb{F}_q$ into n codeword symbols from $\mathbb{F}_q$ which are then stored on n different nodes in the system. Recent work has shown that significant savings in storage space can be obtained by tuning n and k to variations in device failure rates. Such a tuning necessitates *code conversion*: the process of converting already encoded data under an initial $[n^I, k^I]$ code to its equivalent under a final $[n^F, k^F]$ code. The default approach to conversion is to re-encode the data under the new code, which places significant burden on system resources. *Convertible codes* are a recently proposed class of codes for enabling resource-efficient conversions. Existing work on convertible codes has focused on minimizing the access cost, i.e., the number of code symbols accessed during conversion. Bandwidth, which corresponds to the amount of data read and transferred, is another important resource to optimize during conversions. In this paper, we study the fundamental limits on bandwidth used during code conversion and present constructions for bandwidth-optimal convertible codes. First, we model the code conversion problem using network information flow graphs with variable capacity edges. Second, focusing on MDS codes and an important parameter regime called the merge regime, we derive tight lower bounds on conversion bandwidth. The derived bounds show that conversion bandwidth can be significantly reduced as compared to the default approach even in regions where it has been shown that access cost cannot be reduced. Third, we present a new construction for MDS convertible codes which matches the proposed lower bound and is thus bandwidth-optimal during conversion.**

***Index Terms*— Convertible codes, distributed storage systems, erasure codes.**

I. Introduction

ERASURE codes are an essential tool in distributed storage systems used to add redundancy to data in order to avoid data loss when device failures occur [2], [3], [4], [5]. In particular, Maximum Distance Separable (MDS) codes are widely used for this purpose in practice because they require the minimum amount of storage overhead for a given level of failure tolerance. In this setting, an $[n, k]$ MDS code over a finite field $\mathbb{F}_q$ is used to encode a message consisting of k symbols of $\mathbb{F}_q$ into a codeword consisting of n symbols of $\mathbb{F}_q$.[1] Each of these n codeword symbols are then stored on n distinct nodes of the distributed storage system (typically, nodes correspond to storage devices residing on different servers). Large-scale distributed storage systems usually comprise hundreds to thousands of nodes, while n is much smaller in comparison, meaning that these systems store many such codewords distributed across different subsets of nodes. The MDS property ensures that any subset of k symbols out of the n symbols in the codeword is enough to decode the original data. This provides tolerance for up to $(n - k)$ node failures.

The parameters n and k are typically set based on the reliability of storage devices and additional requirements on system performance and storage overhead. Recent work by Kadekodi et al. [6] has shown that the failure rate of disks can vary drastically over time, and that significant savings in storage space (and hence operating costs) can be achieved by tuning the code rate to the observed failure rates. Such tuning typically needs to change both n and k of the code, due to other practical system constraints on these parameters [6]. Other reasons for tuning parameters include changing k in response to changes in data popularity, and adapting the code rate to limit the total amount of storage space used. Such tuning of parameters requires converting the already encoded data from one set of parameters to the newly chosen set of parameters. The *default approach* to achieving this is to re-encode, that is, read the encoded data, decode if necessary, re-encode it under the new code, and then write it back into the relevant nodes. However, such an approach necessitates significantly high overhead in terms of network bandwidth, I/O, and CPU resources in the cluster. This disrupts the normal operation of the storage system.

These applications have led to the study of the *code conversion* problem [7], [8]. Code conversion (Figure 1) is the process of transforming a collection of codewords encoding data under an initial code $\mathcal{C}^I$ into a collection of codewords

Manuscript received 29 September 2022; revised 8 February 2023; accepted 18 March 2023. Date of publication 7 April 2023; date of current version 14 July 2023. This work was supported in part by the NSF CAREER under Award 19434090, in part by the NSF Computer and Network Systems (CNS) under Award 1956271, in part by the Google Faculty Research Award, and in part by the Facebook Distributed Systems Research Award. An earlier version of this paper was presented in part at the 2021 IEEE International Symposium on Information Theory (ISIT) [DOI: 10.1109/ISIT45174.2021.9518121]. *(Corresponding author: Francisco Maturana.)*

The authors are with the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: fmaturan@cs.cmu.edu; rvinayak@cs.cmu.edu).

Communicated by G. Ge, Associate Editor for Coding and Decoding.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2023.3265512.

Digital Object Identifier 10.1109/TIT.2023.3265512

[1] In the literature, this set of n symbols is sometimes called a *stripe* instead of a codeword. In this work, we make no distinctions between these two terms.

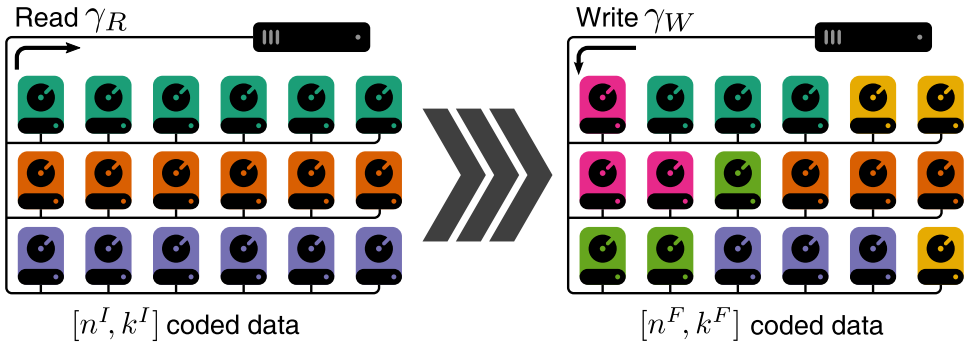

Fig. 1. Conversion process of codewords of an $[n^I, k^I]$ initial code into codewords of an $[n^F, k^F]$ final code. In this figure, each color represents a different codeword. Code conversion is performed by downloading data from storage nodes to a central location, processing the data, and writing back the processed data to the nodes. The total amount of data read is denoted by γ_R, and the total amount of data written is denoted by γ_W.

encoding the same data under a final code $\mathcal{C}^F$.[2] Given certain parameters for $\mathcal{C}^I$ and $\mathcal{C}^F$, the goal is to design the codes $\mathcal{C}^I$ and $\mathcal{C}^F$ along with a conversion procedure from $\mathcal{C}^I$ to $\mathcal{C}^F$ that is efficient in conversion (according to some notion of conversion cost as will be discussed subsequently). The design is subject to additional decodability constraints on the codes $\mathcal{C}^I$ and $\mathcal{C}^F$, such as both satisfying the MDS property, since both these codes encode data in the storage system at different snapshots in time. A pair of codes designed to efficiently convert encoded data from an $[n^I, k^I]$ code to an $[n^F, k^F]$ code is called an $(n^I, k^I; n^F, k^F)$ *convertible code*, and the initial $[n^I, k^I]$ code is said to be (n^F, k^F)-*convertible*. In practice, the exact value of the final parameters n^F and k^F might not be known at the time of code construction, as it might depend on future failure rates. Instead, one might have some finite set of possible values for the pair (n^F, k^F) that will be chosen from at the time of conversion. For this reason, we will also seek to construct initial codes which are simultaneously (n^F, k^F)-convertible for all (n^F, k^F) in a given finite set of final parameter values. This allows the flexibility to choose the parameters n^F and k^F at the time conversion is performed.

Existing works on convertible codes have studied efficiency in terms of the *access cost* of conversion, which corresponds to the number of codeword symbols accessed during conversion. In particular, previous works [7], [8] have derived tight lower bounds on the access cost of conversion for linear MDS convertible codes, and presented explicit constructions of MDS convertible codes that meet those lower bounds (i.e. access-optimal MDS convertible codes). Another important resource overhead incurred during conversion is that on the network bandwidth, which we call *conversion bandwidth*. In the system, this corresponds to the total amount of data transferred between nodes during conversion. Access-optimal convertible codes, by virtue of reducing the number of code symbols accessed, also reduce conversion bandwidth as compared to the default approach. However, *it is not known if these codes are also optimal with respect to conversion bandwidth*.

In this paper, we study the conversion bandwidth of code conversions. We specifically focus on MDS convertible codes and a parameter regime known as the *merge regime*, which has been shown to play the most critical role in the analysis and construction of convertible codes [7]. The merge regime corresponds to conversions where multiple initial codewords are merged into a single final codeword (i.e. $k^F = \lambda^I k^I$ for some integer $\lambda^I \geq 2$).

For the access cost of conversion in the merge regime, it is known [8] that one cannot do better than the default approach for a wide range of parameters (specifically, when $(n^I - k^I) < (n^F - k^F)$, which we term the *increasing-redundancy region*). For the remaining set of parameters (which we term the *decreasing-redundancy region*), access-optimal convertible codes lead to considerable reduction in access cost compared to the default approach. Yet, it is viable that there is room for a significant reduction in conversion bandwidth in both of these regimes. This is possible by considering codes over finite extensions of finite fields $\mathbb{F}_{q^\alpha}$, where each codeword symbol can be interpreted as an α-length vector of sub-symbols from the base field $\mathbb{F}_q$. Such codes are called *vector codes*. Vector codes allow conversion procedures to download elements of the base field from nodes, allowing them to download only a fraction of the codeword symbols. This is inspired by the work on regenerating codes by Dimakis et al. [9] who used vector codes to reduce bandwidth cost of reconstructing a subset of the codeword symbols.

In this paper, first, to analyze the conversion bandwidth, we model the code conversion problem via a *network information flow graph*. This is a directed acyclic graph with capacities, where vertices represent nodes and edges represent the communication between nodes. The approach of information flow graphs was used by Dimakis et al. [9] in the study on regenerating codes. Unlike in the case of regenerating codes, the proposed model involves *variable capacities on edges* representing data download during conversion. This feature turns out to be critical; we show that conversion procedures which download a uniform amount of data from each node are necessarily sub-optimal.

Second, by using the information flow model, we derive a tight lower bound on the conversion bandwidth for MDS convertible codes in the merge regime. Specifically, we use the information flow graph to derive constraints on edge capacities that we then feed into an optimization

[2]The superscripts I and F stand for initial and final, respectively.

TABLE I

COMPARISON OF THE READ CONVERSION BANDWIDTH (γ_R) OF DIFFERENT APPROACHES FOR MERGE CONVERSION. HERE $r^I := n^I - k^I$, $r^F := n^F - k^F$, AND WE ASSUME THAT $r^F \leq k^I$. FOR THE THREE APPROACHES, THE WRITE CONVERSION BANDWIDTH IS CONSTANT ($\gamma_W = r^F\alpha$)

Approach	Read bandwidth ($r^I < r^F$)	Read bandwidth ($r^I \geq r^F$)
Default	$\lambda^I k^I \alpha$	$\lambda^I k^I \alpha$
Access optimal [7]	$\lambda^I k^I \alpha$	$\lambda^I r^F \alpha$
This paper	$\lambda^I k^I \alpha - \lambda^I r^I \alpha \left(\frac{k^I}{r^F} - 1\right)$	$\lambda^I r^F \alpha$

problem whose objective is to minimize the bandwidth of conversion. With this we derive a tight lower bound on the total conversion bandwidth for given code parameters $(n^I, k^I; n^F, k^F)$.

Third, using the above derived (tight) lower bound, we show that (1) in the increasing-redundancy region, where no reduction in access cost as compared to the default approach is possible, a substantial reduction in bandwidth cost can be achieved, and (2) in the decreasing-redundancy region, the access-optimal convertible codes are indeed bandwidth-optimal.

Fourth, we present an explicit construction of MDS convertible codes in the merge regime which achieves this lower bound and is therefore optimal in terms of conversion bandwidth. Table I shows a comparison of the conversion bandwidth required by different approaches to conversion in the merge regime. This construction exploits the *Piggybacking framework* [10], which is a general framework for constructing vector codes, and uses access-optimal MDS convertible codes [8] as a building block.

Above, only a single value of final parameters n^F and k^F was considered. In general, the ideal value of n^F and k^F might be uncertain at the time of encoding, because it depends on future observations. In such cases, having the ability to choose n^F and k^F at the time of conversion is essential. So finally, we propose a technique to transform our construction so as to be *simultaneously* bandwidth-optimal in conversion for any given set of potential final parameter values. The proposed transformation exploits the piggybacking technique [10] in a recursive fashion.

Organization: We review the necessary background and discuss related work in Section II. In Section III, we describe our model for the code conversion process as an information flow graph. In Section IV, we derive a lower bound on the conversion bandwidth of MDS convertible codes in the merge regime. In Section V, we propose an explicit construction for bandwidth-optimal MDS convertible codes in the merge regime, including the transformation to make the construction simultaneously bandwidth-optimal in conversion for multiple final parameter values. In Section VI, we analyze the savings enabled by bandwidth-optimal convertible codes. We conclude the paper in Section VII.

II. BACKGROUND AND RELATED WORK

In this section we start by introducing concepts from the existing literature that are used in this paper. We then do an overview of other related work.

A. Vector Codes and Puncturing

In this section we introduce the basic notation for vector codes. Let $[i]$ denote the subset $\{1, 2, \ldots, i\}$, for a natural number i. An $[n, k, \alpha]$ vector code $\mathcal{C}$ over a finite field $\mathbb{F}_q$ is an injective mapping $\mathcal{C} : \mathbb{F}_q^{\alpha k} \to \mathbb{F}_q^{\alpha n}$. For a given codeword $\mathbf{c} = \mathcal{C}(\mathbf{m})$ and $i \in [n]$, define $\mathbf{c}_i = \mathcal{C}_i(\mathbf{m}) = (c_{\alpha(i-1)+1}, \ldots, c_{\alpha i})$ as the i-th *symbol* of $\mathbf{c}$, which is a vector of length α over $\mathbb{F}_q$. We refer to elements from the base field $\mathbb{F}_q$ as *subsymbols*. A code is said to be *systematic* if it always maps $\mathbf{m}$ to a codeword that contains all the subsymbols of $\mathbf{m}$ uncoded. In a *linear* $[n, k, \alpha]$ vector code $\mathcal{C}$, the encoding of message $\mathbf{m} \in \mathbb{F}_q^{k\alpha}$ is given by the mapping $\mathbf{m} \mapsto \mathbf{mG}$ where $\mathbf{G} \in \mathbb{F}_q^{k\alpha \times n\alpha}$ is called the *generator matrix* of $\mathcal{C}$, and the columns of $\mathbf{G}$ are called *encoding vectors*. The *minimum distance* of a vector code is defined as:

$$\mathrm{dist}(\mathcal{C}) := \min_{\mathbf{m} \neq \mathbf{m}'} |\{i \in [n] : \mathcal{C}_i(\mathbf{m}) \neq \mathcal{C}_i(\mathbf{m}')\}| .$$

An $[n, k, \alpha]$ vector code $\mathcal{C}$ is said to be *maximum-distance-separable* (MDS) if $\mathrm{dist}(\mathcal{C}) = n - k + 1$ (i.e., it achieves the Singleton bound [11]). MDS codes are commonly used in practice because they achieve the optimal tradeoff between storage overhead and failure tolerance.

A *scalar code* is a vector code with $\alpha = 1$. We will omit the parameter α when it is clear from context or when $\alpha = 1$. A *puncturing* of a vector code $\mathcal{C}$ is the resulting vector code after removing a fixed subset of symbols from every codeword.

B. Convertible Codes [7], [8]

Convertible codes are designed to enable encoded data to undergo efficient conversion. Let $\mathcal{C}^I$ be an $[n^I, k^I]$ code over $\mathbb{F}_q$, and $\mathcal{C}^F$ be an $[n^F, k^F]$ code over $\mathbb{F}_q$. In the initial configuration, data will be encoded under the *initial code* $\mathcal{C}^I$, and in the final configuration data will be encoded under the *final code* $\mathcal{C}^F$. Let $r^I = (n^I - k^I)$ and $r^F = (n^F - k^F)$. In order to allow for a change in code dimension from k^I to k^F, multiple codewords of codes $\mathcal{C}^I$ and $\mathcal{C}^F$ are converted at the same time. The reason behind this is that in the initial and final configurations, the system must encode the same total number of message symbols (though encoded differently). Thus, even the simplest non-trivial instance of the problem involves multiple codewords in the initial and final configuration. Let $\mathbf{m}$ be a message of length $M = \mathrm{lcm}(k^I, k^F)$ which in the initial configuration is encoded as $\lambda^I = (M/k^I)$ codewords of $\mathcal{C}^I$ and in the final configuration is encoded as $\lambda^F = (M/k^F)$ codewords of $\mathcal{C}^F$.

For a subset $\mathcal{I} \subseteq [M]$, we denote the restriction of $\mathbf{m}$ to the coordinates in $\mathcal{I}$ as $\mathbf{m}|_{\mathcal{I}} \in \mathbb{F}_q^{|\mathcal{I}|}$, and use the term submessage to refer to such vectors. The mapping of message symbols from $\mathbf{m}$ to different codewords is specified by two partitions of the message symbol indices $[M]$: an initial partition $\mathcal{P}_I$ and a final partition $\mathcal{P}_F$. Each subset $P_i^I \in \mathcal{P}_I$ must be of size $|P_i^I| = k^I$, and indicates that the submessage $\mathbf{m}|_{P_i^I}$ is encoded by initial codeword i, for $i \in [\lambda^I]$. Similarly, each subset $P_j^F \in \mathcal{P}_I$ must be of size $|P_j^F| = k^F$, and indicates that the submessage $\mathbf{m}|_{P_j^F}$ is encoded by final codeword j, for $j \in [\lambda^F]$. A conversion from initial code $\mathcal{C}^I$ to final $\mathcal{C}^F$ is a procedure that takes the initial codewords $\{\mathcal{C}^I(\mathbf{m}|_{P_i^I}) : i \in [\lambda^I]\}$ and outputs the final codewords $\{\mathcal{C}^F(\mathbf{m}|_{P_i^F}) : i \in [\lambda^F]\}$. Putting all these elements together, a convertible code is formally defined as follows.

Definition 1 (Convertible Code [7]): An $(n^I, k^I; n^F, k^F)$ convertible code over $\mathbb{F}_q$ is defined by: (1) a pair of initial and final codes $(\mathcal{C}^I, \mathcal{C}^F)$ over $\mathbb{F}_q$, where $\mathcal{C}^I$ is an $[n^I, k^I]$ code and $\mathcal{C}^F$ is an $[n^F, k^F]$ code, (2) initial and final partitions $(\mathcal{P}_I, \mathcal{P}_F)$ of M such that $|P_i^I| = k^I$ $(\forall P_i^I \in \mathcal{P}_I)$ and $|P_j^F| = k^F (\forall P_j^F \in \mathcal{P}_F)$, (3) a conversion procedure from $\mathcal{C}^I$ to $\mathcal{C}^F$. ▶

The access cost of a conversion procedure is the sum of the *read access cost*, i.e. the total number of code symbols read, and the *write access cost*, i.e. the total number of code symbols written. An *access-optimal convertible code* is a convertible code whose conversion procedure has the minimum access cost over all convertible codes with given parameters $(n^I, k^I; n^F, k^F)$. Similarly, an $[n^I, k^I]$ code is said to be (n^F, k^F)-access-optimally convertible if it is the initial code of an access-optimal $(n^I, k^I; n^F, k^F)$ convertible code.

Definition 1 considers single fixed values for parameters n^F and k^F. In practice, the values of n^F and k^F for the conversion might be unknown. Thus, constructing convertible codes which are simultaneously (n^F, k^F)-access-optimally convertible for several possible values of n^F and k^F is also important (as will be discussed in Section V-B).

Though the definition of convertible codes allows for any kind of initial and final codes, this work focuses on MDS codes. A convertible code is said to be MDS when both $\mathcal{C}^I$ and $\mathcal{C}^F$ are MDS. The access cost lower bound for linear MDS convertible codes is known.

Theorem 1 [8]: Let d_1 be the read access cost of a linear MDS $(n^I, k^I; n^F, k^F)$ convertible code, and d_2 its write access cost. When $k^I \neq k^F$, for every access-optimal code:

$$d_1 = \begin{cases} \lambda^I r^F + [\lambda^I \bmod \lambda^F](k^I - \max\{[k^F \bmod k^I], r^F\}), \\ \qquad\qquad \text{if } r^I \geq r^F \text{ and } r^F < \min\{k^I, k^F\}, \\ M, \qquad\qquad \text{otherwise.} \end{cases}$$

$$d_2 = \lambda^F r^F.$$

There are explicit constructions of access-optimal convertible codes for all valid parameters $(n^I, k^I; n^F, k^F)$: [12] gives a construction when k^F is a multiple of k^I, and [8] gives a construction for the general case. Notice that for the increasing-redundancy region $(r^I < r^F)$, read access cost is always M, which is the same as the default approach. In the decreasing-redundancy region $(r^I \geq r^F)$, on the other hand, one can achieve lower access cost than the default approach when $r^F < \min\{k^I, k^F\}$.

During conversion, code symbols from the initial codewords can play multiple roles: they can become part of different final codewords, their contents might be read or written, additional code symbols may be added and existing code symbols may be removed. Based on their role, code symbols can be divided into three groups: (1) *unchanged symbols*, which are present both in the initial and final codewords without any modifications; (2) *retired symbols*, which are only present in the initial codewords but not in the final codewords; and (3) *new symbols*, which are present only in the final codewords but not in the initial codewords. Both unchanged and retired symbols may be read during conversion, and then linear combinations of data read are written into the new symbols.

The *merge regime* is a fundamental regime of convertible codes which corresponds to conversions which merge multiple initial codewords into a single final codeword. Thus, convertible codes in the merge regime are such that $k^F = \lambda^I k^I$ for some integer $\lambda^I \geq 2$, and $\lambda^F = 1$. Notice that in this regime, d_1 in Theorem 1 reduces to $\lambda^I \min\{r^F, k^I\}$ if $r^I \geq r^F$. We recall two lemmas from previous work which are useful for analyzing the merge regime.

Proposition 1 [7]: For every $(n^I, k^I; n^F, \lambda^I k^I)$ convertible code, all possible pairs of initial and final partitions $(\mathcal{P}_I, \mathcal{P}_F)$ are equivalent up to relabeling.

In the merge regime, all data gets mapped to the same final stripe. Thus, the initial and final partition do not play an important role in this case.

Proposition 2 [7]: In an MDS $(n^I, k^I; n^F, \lambda^I k^I)$ convertible code, there can be at most k^I unchanged symbols from each initial codeword.

This is because having more than k^I unchanged symbols in an initial codeword would contradict the MDS property.

Definition 2 (Stability): A convertible code is said to be *stable* if its conversion procedure has the maximum number of unchanged symbols (M when $k^I \neq k^F$).

1) Access-Optimal Convertible Code for Merge Regime : When $r^I < r^F$, Theorem 1 implies that the default approach has optimal access cost, and so constructing an access-optimal code for this case is trivial. When $r^I \geq r^F$ and the code is in the merge regime, the bound from Theorem 1 in the case where $r^I \geq r^F$ and $r^F < k^I$ reduces to $d_1 \geq \lambda^I r^F$ and $d_2 \geq r^F$. Thus in access-optimal conversion in the merge regime, only r^F code symbols from each initial codeword need to be read. These symbols are then used to compute r^F new code symbols.

In [7], several constructions for access-optimal convertible codes in the merge regime are presented. Codes built using these constructions are (1) systematic, (2) linear, (3) during conversion only access r^F parities from each initial stripe, and (4) when constructed with a given value of $\lambda^I = \lambda$ and $r^F = r$, the initial $[n^I, k^I]$ code is (n^F, k^F)-access-optimally convertible for all $k^F = \lambda' k^I$ and $n^F = k^F + r'$ such that $1 \leq \lambda' \leq \lambda$ and $1 \leq r' \leq r$. In Section V we use an access-optimal convertible code in the merge regime as part of our

Symbol 1	$f_1(\mathbf{m}_1)$	$f_1(\mathbf{m}_2)$	$\cdots$	$f_1(\mathbf{m}_\alpha)$
Symbol 2	$f_2(\mathbf{m}_1)$	$f_2(\mathbf{m}_2)$	$\cdots$	$f_2(\mathbf{m}_\alpha)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Symbol n	$f_n(\mathbf{m}_1)$	$f_n(\mathbf{m}_2)$	$\cdots$	$f_n(\mathbf{m}_\alpha)$

(a) α instances of the base code

$f_1(\mathbf{m}_1)$	$f_1(\mathbf{m}_1) + g_{2,1}(\mathbf{m}_2)$	$\cdots$	$f_1(\mathbf{m}_\alpha) + g_{\alpha,1}(\mathbf{m}_1, \ldots, \mathbf{m}_\alpha)$
$f_2(\mathbf{m}_1)$	$f_2(\mathbf{m}_1) + g_{2,2}(\mathbf{m}_2)$	$\cdots$	$f_2(\mathbf{m}_\alpha) + g_{\alpha,2}(\mathbf{m}_1, \ldots, \mathbf{m}_\alpha)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$f_n(\mathbf{m}_1)$	$f_n(\mathbf{m}_1) + g_{2,n}(\mathbf{m}_2)$	$\cdots$	$f_1(\mathbf{m}_\alpha) + g_{\alpha,n}(\mathbf{m}_1, \ldots, \mathbf{m}_\alpha)$

(b) Piggybacked code

Fig. 2. Piggybacking framework [10] for constructing vector codes.

construction of bandwidth-optimal convertible codes for the merge regime. Next, we give a brief description of the code construction that we use from [7] (referred to as the "general construction").

Consider the case where $r^I \geq r^F$ and $r^F < k^I$ (otherwise, the construction is trivial). The codes $\mathcal{C}^I$ and $\mathcal{C}^F$ over finite field $\mathbb{F}_q$ are defined via the matrices $\mathbf{G}^I = \left[\mathbf{I}_{k^I} \mid \mathbf{P}^I\right]$ and $\mathbf{G}^F = \left[\mathbf{I}_{k^F} \mid \mathbf{P}^F\right]$ where:

- $\mathbf{I}_k$ is the $k \times k$ identity matrix,
- $\alpha_1, \alpha_2, \ldots, \alpha_{r^I}$ are distinct elements from $\mathbb{F}_q$,
- $\mathbf{P}^I$ is the $k^I \times r^I$ Vandermonde matrix with evaluation points $(\alpha_1, \ldots, \alpha_{r^I})$,
- $\mathbf{P}^F$ is the $k^F \times r^F$ Vandermonde matrix with evaluation points $(\alpha_1, \ldots, \alpha_{r^F})$.

(In the original construction [7], α_i is chosen as θ^{i-1} for some primitive element $\theta \in \mathbb{F}_q$.) One important aspect of this construction is that, due to the nature of Vandermonde matrices, the i-th column of $\mathbf{P}^F$ is equal to the vertical concatenation of the respective i-th columns of $\mathbf{P}^I, \alpha_i^{k^I}\mathbf{P}^I, \ldots, \alpha_i^{(\lambda^I-1)k^I}\mathbf{P}^I$. This property ensures that each final parity can be constructed during conversion as a linear combination of one initial parity from each initial codeword. As shown in [7], this construction satisfies the properties (1–4) described above, and is MDS for appropriately chosen points α_i ($i \in [r^I]$) and sufficiently large $\mathbb{F}_q$.

Example 1 (Access-Optimal Code): Consider the parameters $(n^I = 7, k^I = 4; n^F = 11, k^F = 8)$ over $\mathbb{F}_{17}$: the evaluation points $(\alpha_1 = 1, \alpha_2 = 2, \alpha_3 = 6)$ yield an MDS access-optimal code. It is easy to check that the codes defined by the following matrices are MDS:

$$\mathbf{P}^I = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 6 \\ 1 & 4 & 2 \\ 1 & 8 & 12 \end{bmatrix} \qquad \mathbf{P}^F = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 6 \\ 1 & 4 & 2 \\ 1 & 8 & 12 \\ 1 & 16 & 4 \\ 1 & 15 & 7 \\ 1 & 13 & 8 \\ 1 & 9 & 14 \end{bmatrix}$$

Now, suppose the data $(a_1, \ldots, a_4)$ and $(a_5, \ldots, a_8)$ are encoded with the initial code. It is easy to check that the following holds:

$$(a_1, \ldots, a_4)\mathbf{P}^I + (a_5, \ldots, a_8)\mathbf{P}^I \begin{bmatrix} 1 & 0 & 0 \\ 0 & 16 & 0 \\ 0 & 0 & 4 \end{bmatrix} = (a_1, \ldots, a_8)\mathbf{P}^F.$$

►

C. Network Information Flow

Network information flow [13] is a class of problems that model the transmission of information from sources to sinks in a point-to-point communication network. *Network coding* [14], [15], [16], [17], [18] is a generalization of store-and-forward routing, where each node in the network is allowed to combine its inputs using a code before communicating messages to other nodes. For the purposes of this paper, an *information flow graph* is a directed acyclic graph $G = (V, E)$, where V is the set of nodes, $E \subseteq V \times V \times \mathbb{R}_{\geq 0}$ is the set of edges with non-negative capacities, and $(i, j, c) \in E$ represents that information can be sent noiselessly from node i to node j at rate c. Let $X_1, X_2, \ldots, X_m$ be mutually independent information sources with rates $x_1, x_2, \ldots, x_m$ respectively. Each information source X_i is associated with a source $s_i \in V$, where it is generated, and a sink $t_i \in V$, where it is required. In this paper we mainly make use of the *information max-flow bound* [19] which indicates that it is impossible to transmit X_i at a higher rate than the maximum flow from s_i to t_i. In other words, $x_i \leq \text{max-flow}(s_i, t_i)$ for all $i \in [m]$ is a necessary condition for a network coding scheme satisfying all constraints to exist. In our analysis, we will consider s_i-t_i-cuts of the information flow graph, which give an upper bound on $\text{max-flow}(s_i, t_i)$ and thus an upper bound on x_i as well. We will also utilize the fact that two independent information sources with the same source and sink can be considered as a single information source with rate equal to the sum of their rates.

In [9], information flow and network coding is applied to the *repair problem* in distributed storage systems. The repair problem is the problem of reconstructing a small number of failed code symbols in an erasure code (without having to decode the full codeword). Dimakis et al. [9] use information flow to establish bounds on the storage size and repair network-bandwidth of erasure codes. In this work we use information flow to model the process of code conversion and establish lower bounds on the total amount of network bandwidth used during conversion.

D. Piggybacking framework for Constructing Vector Codes

The *Piggybacking framework* [10], [20] is a framework for constructing new vector codes building on top of existing codes. The main technique behind the Piggybacking framework is to take an existing code as a *base code*, create a new vector code consisting of multiple instances of the base code (as described below), and then add carefully designed functions of the data (called *piggybacks*) from one instance to the others. These piggybacks are added in a way such that

it retains the decodability properties of the base code (such as the MDS property). The piggyback functions are chosen to confer additional desired properties to the resulting code. In [10], the authors showcase the Piggybacking framework by constructing codes that are efficient in reducing bandwidth consumed in repairing codeword symbols.

More specifically, the Piggybacking framework works as follows. Consider a length n code defined by the function $f(\mathbf{m}) = (f_1(\mathbf{m}), f_2(\mathbf{m}), \ldots, f_n(\mathbf{m}))$. Now, consider α instances of this base code, each corresponding to a coordinate of the α-length vector of each symbol in the new vector code. Let $(\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_\alpha)$ denote the independent messages encoded under these α instances, as shown in Figure 2a. For every i such that $2 \leq i \leq \alpha$, one can add to the data encoded in instance i an arbitrary function of the data encoded by instances $\{1, \ldots, (i-1)\}$. Such functions are called *piggyback functions*, and the piggyback function corresponding to code symbol $j \in [n]$ of instance $i \in \{2, \ldots, \alpha\}$ is denoted as $g_{i,j}$.

The decoding of the piggybacked code proceeds as follows. Observe that instance 1 does not have any piggybacks. First, instance 1 of the base code is decoded using the base code's decoding procedure in order to obtain $\mathbf{m}_1$. Then, $\mathbf{m}_1$ is used to compute and subtract any of the piggybacks $\{g_{2,i}(\mathbf{m}_1)\}_{i=1}^{n}$ from instance 2 and the base code's decoding can then be used to recover $\mathbf{m}_2$. Decoding proceeds like this, using the data decoded from previous instances in order to remove the piggybacks until all instances have been decoded. It is clear that if an $[n, k, \alpha]$ vector code is constructed from an $[n, k]$ MDS code as the base code using the Piggybacking framework, then the resulting vector code is also MDS. This is because any set of k symbols from the vector code contains a set of k subsymbols from each of the α instances.

In this paper, we use the Piggybacking framework to design a code where piggybacks store data which helps in making the conversion process efficient.

E. Other Related Work

Apart from [7], which presented a general formulation for the code conversion problem, special cases of code conversion have been studied in the literature. In [21], [22], the authors study the problem of minimizing bandwidth usage when adding extra parities, which corresponds to the case where $k^I = k^F$ and $n^I < n^F$. In [23], the authors propose two specific pairs of non-MDS codes for a distributed storage system which support conversion with lower access cost than the default approach. In [24], the authors study two kinds of conversion in the context of distributed matrix multiplication. In [25], the authors propose an approach to conversion similar to that of [7] and [8], but allow each individual codeword to belong to a different code. These works focus on reducing the access cost of conversion, whereas the focus of the current paper is on conversion bandwidth. Furthermore, the approaches proposed in these works [23], [24] do not come with any theoretical guarantees on optimality, whereas the current paper also presents tight lower bounds on the conversion bandwidth along with bandwidth-optimal constructions.

A related line of research is that of regenerating codes. Regenerating codes are erasure codes which are designed to solve the repair problem (described in Section II-C above) by downloading the least amount of data from the surviving nodes. Regenerating codes were first proposed by Dimakis et al. [9]. Several subsequent works have provided constructions of codes with bandwidth-efficient repair [10], [21], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], lower bounds on the subpacketization of such codes [46], [47], [48], constructions and bounds for scalar codes [49], [50], [51], [52], [53], [54] and other generalizations of regenerating codes [55], [56], [57], [58]. The regenerating codes framework measures the cost of repair in a similar way to how we measure the cost of conversion in this work: in terms of the total amount of network bandwidth used, i.e. the total amount of data transferred during repair. Thus, some of the techniques used in this paper are inspired by the existing regenerating codes literature, as further explained in Section II-C. Furthermore, specific instances of code conversion can be viewed as instances of the repair problem, for example, increasing n while keeping k fixed as studied in [10], [21] and [22]. In such a scenario, one can view adding additional nodes as "repairing" them as proposed in [21]. Note that this setting imposes a relaxed requirement of repairing only a specific subset of nodes as compared to regenerating codes which require optimal repair of all nodes. Yet, the lower bound from regenerating codes still applies for MDS codes, since as shown in [31], the regenerating codes lower bound for MDS codes applies even for repair of only a single specific node.

There have been several works studying the scaling problem [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70]. This problem considers upgrading an erasure-coded storage system with s new empty data nodes. The general goal is to efficiently and evenly redistribute data across all nodes, while updating parities to reflect the new placement of the data. This is a fundamentally different problem from the code conversion problem we study in this paper, due to the scaling problem's need to redistribute data across nodes.

III. Modeling Conversion for Optimizing Conversion Bandwidth

In this section, we model the conversion process as an information flow problem. We utilize this model primarily for deriving lower bounds on the total amount of information that needs to be transferred during conversion. Since our focus is on modeling the conversion process, we consider a single value for each of the final parameters n^F and k^F. This model continues to be valid for each individual conversion, even when the final parameters might take multiple values.

In Section II-B, we reviewed the definition of convertible codes from literature [7], [8]. Existing works on convertible codes [7], [8] have considered only *scalar codes*, where each code symbol corresponds to a scalar from a finite field $\mathbb{F}_q$. Considering scalar codes is sufficient when optimizing for access cost, which was the focus in these prior works, since the access cost is measured at the granularity of code

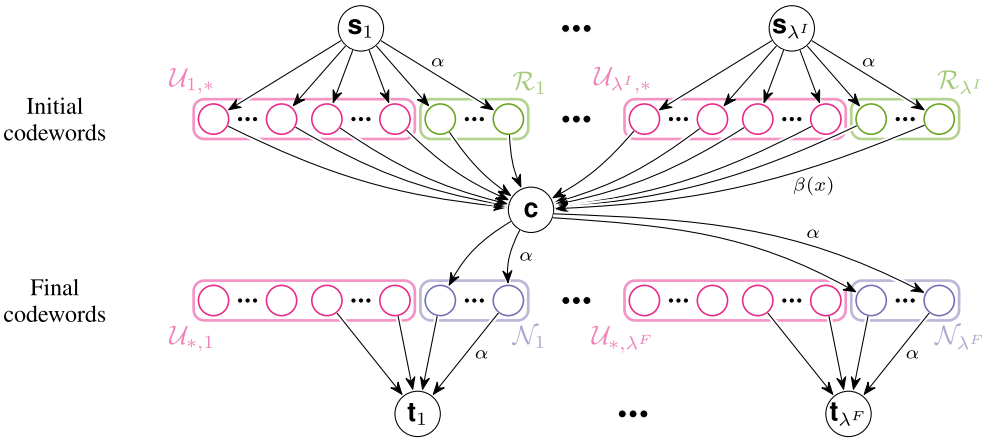

Fig. 3. Information flow graph of conversion in the general case. Unchanged, retired, and new nodes are shown in different colors. Notice that each unchanged node in this figure is drawn twice: once in the initial codewords and once in the final codewords. These correspond to exactly the same node, but are drawn twice for clarity. Some representative edges are labeled with their capacities.

symbols. However, when optimizing conversion bandwidth, vector codes can perform better than scalar codes since they allow partial download from a node. This allows conversion procedures to only download a fraction of a code symbol and thus only incur the conversion bandwidth associated with the size of that fraction. This can potentially lead to significant reduction in the total conversion bandwidth. For this reason, we consider the initial code $\mathcal{C}^I$ as an $[n^I, k^I, \alpha]$ MDS code and the final code $\mathcal{C}^F$ as an $[n^F, k^F, \alpha]$ MDS code, where $\alpha \geq 1$ is considered as a free parameter chosen to minimize conversion bandwidth. This move to vector codes is inspired by the work of Dimakis et al. [9] on regenerating codes, who showed the benefit of vector codes in reducing network bandwidth in the context of the repair problem. For MDS convertible codes, message size will be $B = M\alpha = \text{lcm}(k^I, k^F)\alpha$, which we interpret as a vector $\mathbf{m} \in \mathbb{F}_q^{M\alpha}$ composed of M symbols made up of α subsymbols each. We will denote the number of subsymbols downloaded from node s during conversion as $\beta(s) \leq \alpha$ and extend this notation to sets of nodes as $\beta(\mathcal{S}) = \sum_{s\in\mathcal{S}} \beta(s)$.

Consider an $(n^I, k^I; n^F, k^F)$ MDS convertible code with initial partition $\mathcal{P}_I = \{P_1^I, \ldots, P_{\lambda^I}^I\}$ and final partition $\mathcal{P}_F = \{P_1^F, \ldots, P_{\lambda^F}^F\}$. We model conversion using an information flow graph as the one shown in Figure 3 where message symbols are generated at source nodes, and sinks represent the decoding constraints of the final code. Symbols of message $\mathbf{m}$ are modeled as information sources $X_1, X_2, \ldots, X_M$ of rate α (over $\mathbb{F}_q$) each. For each initial codeword $i \in [\lambda^I]$, we include one source node s_i, where the information sources corresponding to the message symbols in P_i^I are generated. Each code symbol of initial codeword i is modeled as a node with an incoming edge from s_i. A *coordinator node* c models the central location where the contents of new symbols are computed, and it has incoming edges from all nodes in the initial codewords. During conversion, some of the initial code symbols will remain unchanged, some will be retired, and some new code symbols will be added. Thus, we also include the nodes corresponding to unchanged symbols in the final codewords (that is, every unchanged node is shown twice in Figure 3). Note that the unchanged nodes in the initial codewords and the unchanged nodes in the final codewords are identical, and thus do not add any conversion bandwidth. For each new symbol we add a node that connects to the coordinator node. From this point, we will refer to code symbols and their corresponding nodes interchangeably. For each final codeword $j \in [\lambda^F]$, we add a sink t_j which connects to some subset of nodes from final codeword j, and recovers the information sources corresponding to the message symbols in P_j^F.

Thus, the information flow graph for a convertible code comprises the following nodes:

- unchanged nodes $\mathcal{U}_{i,j} = \{u_{i,j,1}, \ldots, u_{i,j,|\mathcal{U}_{i,j}|}\}$ for all $i \in [\lambda^I]$, $j \in [\lambda^F]$, which are present both in the initial and final codewords;
- retired nodes $\mathcal{R}_i = \{v_{i,1}, \ldots, v_{i,|\mathcal{R}_i|}\}$ for $i \in [\lambda^I]$, which are only present in the initial codewords;
- new nodes $\mathcal{N}_j = \{w_{j,1}, \ldots, w_{j,|\mathcal{N}_j|}\}$ for $j \in [\lambda^F]$, which are only present in the final codewords;
- source nodes s_i for $i \in [\lambda^I]$, representing the data to be encoded;
- sink nodes t_j for $j \in [\lambda^F]$, representing the data decoded; and
- a coordinator node c.

In the information flow graph, information source X_l is generated at node s_i if and only if $l \in P_i^I$, and recovered at node t_l if and only if $l \in P_j^F$.

Throughout this paper, we use the disjoint union symbol $\sqcup$ when appropriate to emphasize that the two sets in the union are disjoint. To simplify the notation, when $*$ is used as an index, it denotes the disjoint union of the indexed set over the range of that index, e.g. $\mathcal{U}_{*,j} = \bigsqcup_{i=1}^{\lambda^I} \mathcal{U}_{i,j}$.

The information flow graph must be such that the following conditions hold: (1) the number of nodes per initial codeword is n^I, i.e., $|\mathcal{U}_{i,*}| + |\mathcal{R}_i| = n^I$ for all $i \in [\lambda^I]$; and (2) the number of nodes per final codeword is n^F, i.e., $|\mathcal{U}_{*,j}| + |\mathcal{N}_j| = n^F$ for all $j \in [\lambda^F]$. Additionally, the information flow graph contains the following set of edges E, where a directed edge

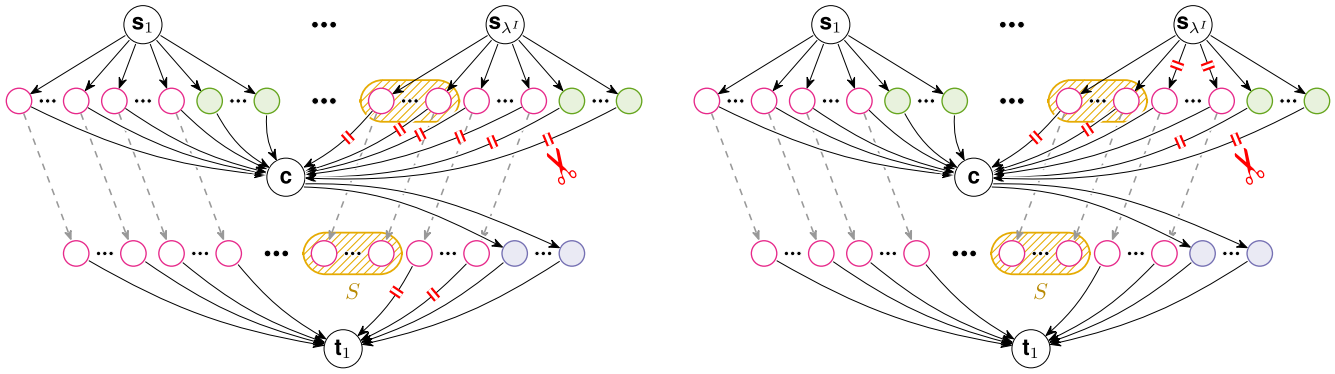

Fig. 4. Information flow graph of conversion in the merge regime with two different cuts (used in proofs). For clarity, each unchanged node is drawn twice: once in the initial codewords and once in the final codeword. These two instances are connected by a dashed arrow. Marked edges denote a graph cut.

from node u to v with capacity δ is represented with the triple (u, v, δ):

- $\{(s_i, x, \alpha) : x \in \mathcal{U}_{i,*} \sqcup \mathcal{R}_i\} \subset E$ for each $i \in [\lambda^I]$, where the capacity corresponds to the size of the data stored on each node;
- $\{(x, c, \beta(x)) : x \in \mathcal{U}_{i,*} \sqcup \mathcal{R}_i\} \subset E$ for each $i \in [\lambda^I]$, where the capacity corresponds to the amount of data downloaded from node x;
- $\{(c, y, \alpha) : y \in \mathcal{N}_j\} \subset E$ for each $j \in [\lambda^F]$, where the capacity corresponds to the size of the data stored on each new node;
- $\{(y, t_j, \alpha) : y \in V_j\} \subset E$ for $V_j \subseteq \mathcal{U}_{*,j} \sqcup \mathcal{N}_j$ such that $|V_j| = k^F$, for all $j \in [\lambda^F]$, where the capacity corresponds to the size of the data stored on each node.

The sinks t_j represent the decoding constraints of the final code, and each choice of set V_j will represent a different choice of k^F code symbols for decoding the final codeword. A necessary condition for a conversion procedure is to satisfy all sinks t_j for all possible $V_1, \ldots, V_{\lambda^F}$. The sets $\mathcal{U}_{i,j}, \mathcal{R}_i, \mathcal{N}_j$ and the capacities $\beta(x)$ are determined by the conversion procedure of the convertible code. Figure 3 shows the information flow graph of an arbitrary convertible code.

Definition 3 (Conversion Bandwidth): The *read conversion bandwidth* γ_R is the total amount of data transferred from the initial nodes to the coordinator node c. The *write conversion bandwidth* γ_W is the total amount of data transferred from the coordinator node c to the new nodes. The (total) *conversion bandwidth* γ is the sum of the read conversion bandwidth and the write conversion bandwidth. Formally:

$$\gamma_R := \beta(\mathcal{U}_{*,*} \sqcup \mathcal{R}_*), \quad \gamma_W := |\mathcal{N}_*|\alpha, \quad \gamma := \gamma_R + \gamma_W. \tag{1}$$

▶

Once the structure of the graph is set and fixed, information flow analysis gives lower bounds on the capacities $\beta(x)$. Therefore, a part of our objective in designing convertible codes is to set $\mathcal{U}_{i,j}, \mathcal{R}_i, \mathcal{N}_j$ so as to minimize the lower bound on γ.

Notice that the conversion process, as defined above, is *not* a single-source multi-cast problem; therefore, the information max-flow bound is not guaranteed to be achievable. Nonetheless, information flow can be applied to obtain a lower bound (Section IV), which we show is achievable by providing a construction (Section V).

Remark 1: In practice, conversion bandwidth can sometimes be further reduced by placing the coordinator node along with a new node and/or a retired node in the same server. One can even first split the coordinator node into several coordinator nodes, each processing data which is not used in conjunction with data processed by other coordinator nodes, and then place them in the same server as a new node and/or a retired node. Such "optimizations" do not fundamentally alter our result, and hence are left out in order to make the exposition clear.

IV. Optimizing Conversion Bandwidth in the Merge Regime

In this section, we use the information flow model presented in Section III to derive a lower bound on the conversion bandwidth for MDS codes in the merge regime. Recall from Section II-B, that convertible codes in the merge regime are those where $k^F = \lambda^I k^I$ for some integer $\lambda^I \geq 2$, i.e., this regime corresponds to conversions where multiple initial codewords are merged into a single final codeword. As in the previous section, our analysis focuses on a single conversion, and thus a single value for the final parameters n^F and k^F. However, our analysis only depends on the conversion process itself; therefore, the bound on the bandwidth still applies even if we consider multiple conversions.

Consider an $(n^I, k^I; n^F, \lambda^I k^I)$ convertible code in the merge regime, for some integer $\lambda^I \geq 2$. Note that for all convertible codes in the merge regime, it holds that the number of final codewords is $\lambda^F = 1$. Since all initial and final partitions $(\mathcal{P}_I, \mathcal{P}_F)$ are equivalent up to relabeling in this regime (by Proposition 1 [7]), we can omit them from our analysis. Note also that all information sources are recovered at the same sink node, t_1. Thus, we may treat each source node s_i as having a single information source X_i of rate αk^I ($i \in [\lambda^I]$). For each source node and each sink node pair, we can invoke the information max-flow bound (Section II-C) to derive an inequality. For conversion to be possible, the variable-capacity edges must take on values such that all these

inequalities are simultaneously satisfied. Figure 4a shows the information flow graph for a convertible code in the merge regime.

First, we derive a general lower bound on conversion bandwidth in the merge regime by considering a simple cut in the information flow graph. Intuitively, this lower bound emerges from the fact that new nodes need to have a certain amount of information from each initial codeword in order to fulfill the MDS property of the final code. This lower bound depends on the number of unchanged nodes and achieves its minimum when the number of unchanged nodes is maximized. Recall from Section II-B that convertible codes with maximum number of unchanged nodes are called *stable* convertible codes. Thus, the derived lower bound is minimized for stable convertible codes.

Lemma 2: Consider an MDS $(n^I, k^I; n^F, \lambda^I k^I)$ convertible code. Then $\gamma_R \geq \lambda^I \alpha \min\{r^F, k^I\}$ and $\gamma_W \geq r^F \alpha$, where equality is only possible for stable codes.

Proof: We prove this inequality via an information flow argument. Let $i \in [\lambda^I]$ and consider the information source generated at source s_i. Let $S \subseteq \mathcal{U}_{i,1}$ be a subset of unchanged nodes from initial codeword i of size $\tilde{r}_i = \min\{r^F, |\mathcal{U}_{i,1}|\}$. Consider a sink t_1 that connects to nodes $\mathcal{U}_{*,1} \setminus S$. We choose the graph cut defined by nodes $\{s_i\} \sqcup \mathcal{U}_{i,1} \sqcup \mathcal{R}_i$ (see Figure 4a, which depicts the cut for $i = \lambda^I$). This cut yields the following inequality:

$$k^I \alpha \leq \max\{|\mathcal{U}_{i,1}| - r^F, 0\}\alpha + \beta(\mathcal{U}_{i,1} \sqcup \mathcal{R}_i)$$
$$\iff \beta(\mathcal{U}_{i,1} \sqcup \mathcal{R}_i) \geq (k^I + r^F - \max\{|\mathcal{U}_{i,1}|, r^F\})\alpha$$

This inequality must hold for every $i \in [\lambda^I]$ simultaneously; otherwise, it would be impossible for the sink to recover the full data. By summing this inequality over all sources $i \in [\lambda^I]$ and using the definition of γ (Equation (1)), we obtain:

$$\gamma \geq \sum_{i=1}^{\lambda^I} (k^I + r^F - \max\{|\mathcal{U}_{i,1}|, r^F\})\alpha + |\mathcal{N}_1|\alpha$$

By Proposition 2 [7], $|\mathcal{U}_{i,1}| \leq k^I$. Therefore, it is clear that the right hand side achieves its minimum if and only if $|\mathcal{U}_{i,1}| = k^I$ for all $i \in [\lambda^I]$ (i.e. the code is stable). Proposition 2 also implies that $\gamma_W \geq r^F \alpha$, proving the lemma. ∎

Remark 2: Note that the conversion bandwidth lower bound described in Lemma 2 coincides with the access-cost lower bound described in Theorem 1 when $r^I \geq r^F$. This follows by recalling that each node corresponds to an α-length vector, and for scalar codes $\alpha = 1$. ▶

In particular, this implies that convertible codes in the merge regime which are access-optimal and have $r^I \geq r^F$ are also bandwidth-optimal (i.e. those in the decreasing-redundancy region). However, as we will show next, this property fails to hold when $r^I < r^F$ (that is, increasing-redundancy region).

We next derive a lower bound on conversion bandwidth which is tighter than Lemma 2 when $r^I < r^F$. Nevertheless, it allows for less conversion bandwidth usage than the access-optimal codes.

Intuitively, the data downloaded from retired nodes during conversion will be "more useful" than the data downloaded from unchanged nodes, since unchanged nodes already form part of the final codeword. At the same time, it is better to have the maximum amount of unchanged nodes per initial codeword (k^I) because this minimizes the number of new nodes that need to be constructed. However, this leads to fewer retired nodes per initial codeword (r^I). If the number of retired nodes per initial codeword is less than the number of new nodes ($r^I < r^F$), then conversion procedures are forced to download data from unchanged nodes. This is because one needs to download at least $r^F \alpha$ from each initial codeword (by Lemma 2). Since data from unchanged nodes is "less useful", more data needs to be downloaded in order to construct the new nodes.

As in the case of Lemma 2, this lower bound depends on the number of unchanged nodes in each initial codeword, and achieves its minimum in the case of stable convertible codes.

Lemma 3: Consider an MDS $(n^I, k^I; n^F, \lambda^I k^I)$ convertible code, with parameters such that $r^I < r^F \leq k^I$. Then $\gamma_R \geq \lambda^I \alpha \left(r^I + k^I \left(1 - \frac{r^I}{r^F}\right)\right)$ and $\gamma_W \geq r^F \alpha$, where equality is only possible for stable codes.

Proof: We prove this via an information flow argument. Let $i \in [\lambda^I]$ and consider the information source generated at source s_i. Let $S \subseteq \mathcal{U}_{i,1}$ be a subset of size $\tilde{r}_i = \min\{r^F, |\mathcal{U}_{i,1}|\}$. Consider a sink t_1 that connects to the nodes in $\mathcal{U}_{*,1} \setminus S$. Now, we choose a different cut from the one considered in Lemma 2, which allows to derive a tighter bound when $r^I < r^F$. We choose the graph cut defined by nodes $\{s_i\} \sqcup S \sqcup \mathcal{R}_i$ (see Figure 4b, which depicts the cut when $i = \lambda^I$). This yields the following inequality:

$$k^I \alpha \leq (|\mathcal{U}_{i,1}| - \tilde{r}_i)\alpha + \beta(S) + \beta(\mathcal{R}_i)\,.$$

This inequality must hold for all possible $S \subseteq \mathcal{U}_{i,1}$ simultaneously; otherwise, there would exist at least one sink incapable of recovering the full data, which violates the MDS property. By rearranging this inequality and summing over all possible choices of subset S, we obtain the following inequality:

$$\begin{aligned} &\binom{|\mathcal{U}_{i,1}|}{\tilde{r}_i} \left(k^I + \tilde{r}_i - |\mathcal{U}_{i,1}|\right) \alpha \\ &\qquad \leq \binom{|\mathcal{U}_{i,1}|-1}{\tilde{r}_i - 1} \beta\left(\mathcal{U}_{i,1}\right) + \binom{|\mathcal{U}_{i,1}|}{\tilde{r}_i} \beta\left(\mathcal{R}_i\right) \\ &\Leftrightarrow |\mathcal{U}_{i,1}| \left(k^I + \tilde{r}_i - |\mathcal{U}_{i,1}|\right) \alpha \leq \tilde{r}_i \beta\left(\mathcal{U}_{i,1}\right) + |\mathcal{U}_{i,1}| \beta\left(\mathcal{R}_i\right). \end{aligned} \tag{2}$$

Then, our strategy to obtain a lower bound is to find the minimum value for conversion bandwidth γ which satisfies Equation (2) for all $i \in [\lambda^I]$, which can be formulated as the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \gamma = \textstyle\sum_{i \in \lambda^I} [\beta(\mathcal{U}_{i,1}) + \beta(\mathcal{R}_i)] + |\mathcal{N}_1|\alpha \\ \text{subject to} \quad & \text{inequality}(2), \text{for all} i \in [\lambda^I] \\ & 0 \leq \beta(x) \leq \alpha, \text{ for all } x \in \mathcal{U}_{*,1} \sqcup \mathcal{R}_*. \end{aligned} \tag{3}$$

Intuitively, this linear program shows that it is preferable to download more data from retired nodes ($\beta(\mathcal{R}_i)$) than unchanged nodes ($\beta(\mathcal{U}_{i,1})$), since both have the same impact on γ but the contribution of $\beta(\mathcal{R}_i)$ towards satisfying Equation (2) is greater than or equal to that of $\beta(\mathcal{U}_{i,1})$, because $\tilde{r}_i \leq |\mathcal{U}_{i,1}|$ by definition. Thus to obtain an optimal solution we

first set $\beta(\mathcal{R}_i) = \min\{k^I + \tilde{r}_i - |\mathcal{U}_{i,1}|, |\mathcal{R}_i|\}\alpha$ to the maximum needed for all $i \in [\lambda^I]$, and then set:

$$\sum_{x \in \mathcal{U}_{i,1}} \beta(x) = \frac{\max\{\tilde{r}_i - r^I, 0\}|\mathcal{U}_{i,1}|\alpha}{\tilde{r}_i}, \qquad \text{for all } i \in [\lambda^I]$$

to satisfy the constraints. It is straightforward to check that this solution satisfies the KKT (Karush-Kuhn-Tucker) conditions, and thus is an optimal solution to linear program 3. By replacing these terms back into γ and simplifying we obtain the optimal objective value:

$$\gamma^* = \sum_{i=1}^{\lambda^I} \left[k^I - \min\{r^I, \tilde{r}_i\} \left(\frac{|\mathcal{U}_{i,1}|}{\tilde{r}_i} - 1 \right) \right] \alpha + |\mathcal{N}_1|\alpha$$

It is easy to show that the right hand side achieves its minimum if and only if $|\mathcal{U}_{i,1}| = k^I$ for all $i \in [\lambda^I]$ (i.e., the code is stable). This gives the following lower bound for conversion bandwidth:

$$\gamma \geq \lambda^I \alpha \left(r^I + k^I \left(1 - \frac{r^I}{r^F} \right) \right) + r^F \alpha.$$

By Theorem 1, $\gamma_W \geq r^F \alpha$, which proves the lemma. ∎

By combining Lemmas 2 and 3 we obtain the following general lower bound on conversion bandwidth of MDS convertible codes in the merge regime.

Theorem 4: For any MDS $(n^I, k^I; n^F, \lambda^I k^I)$ convertible code:

$$\gamma_R \geq \begin{cases} \lambda^I \alpha \min\{k^I, r^F\}, & \text{if } r^I \geq r^F \text{ or } k^I \leq r^F, \\ \lambda^I \alpha \left(r^I + k^I \left(1 - \frac{r^I}{r^F} \right) \right), & \text{otherwise.} \end{cases}$$

$$\gamma_W \geq r^F \alpha.$$

where equality can only be achieved by stable convertible codes.

Proof: Follows from Lemmas 2 and 3. ∎

In Section V, we show that the lower bound of Theorem 4 is indeed achievable for all parameter values in the merge regime, and thus it is tight. We will refer to convertible codes that meet this bound with equality as *bandwidth-optimal*.

Remark 3: Observe that the model above allows for nonuniform data download during conversion, that is, it allows the amount of data downloaded from each node during conversion to be different. If instead one were to assume uniform download, i.e. $\beta(x) = \beta(y)$ for all $x, y \in \mathcal{U}_{*,*} \sqcup \mathcal{R}_*$, then a higher lower bound for conversion bandwidth γ is obtained (mainly due to Equation (2) in the proof of Lemma 3). Since the lower bound of Theorem 4 is achievable, this implies that assuming uniform download necessarily leads to a suboptimal solution.

Remark 4: The case where $k^I = k^F$ can be analyzed using the same techniques used in this section. In this case, $\lambda^I = 1$. There are some differences compared to the case of the merge regime: for example, in this case the number of unchanged nodes can be at most $\min\{n^I, n^F\}$ (in contrast to the $\lambda^I k^I$ maximum of the merge regime). So, conversion bandwidth in the case where $n^I \geq n^F$ is zero, since we can simply keep n^F nodes unchanged. In the case where $n^I < n^F$, the same analysis from Lemma 3 is followed, but the larger number of unchanged nodes will lead to a slightly different inequality. Thus, in the case of $k^I = k^F$ the lower bound on conversion bandwidth is:

$$\gamma \geq \begin{cases} 0, & \text{if } n^I \geq n^F \\ \alpha \left(k^I + r^I \right) \left(1 - \frac{r^I}{r^F} \right) + (r^F - r^I)\alpha, & \text{otherwise.} \end{cases}$$

Readers familiar with regenerating codes might notice that the above lower bound is equivalent to the lower bound on the repair bandwidth [9], [42] when $(r^F - r^I)$ symbols of an $[k^I + r^F, k^I]$ MDS code are to be repaired with the help of the remaining $(k^I + r^I)$ symbols. Note that this setting imposes a relaxed requirement of repairing only a specific subset of symbols as compared to regenerating codes which require optimal repair of all nodes. Yet, the lower bound remains the same. This is not surprising though, since it has been shown [31] that the regenerating codes lower bound for MDS codes applies even for repair of only a single specific symbol. ▶

V. Explicit Construction of Bandwidth-Optimal MDS Convertible Codes in the Merge Regime

In this section, we present an explicit construction for bandwidth-optimal convertible codes in the merge regime. Our construction employs the Piggybacking framework [10]. Recall from Section II-D that the Piggybacking framework is a framework for constructing vector codes using an existing code as a base code and adding specially designed functions called piggybacks which impart additional properties to the resulting code. We use an access-optimal convertible code to construct the base code and design the piggybacks to help achieve minimum conversion bandwidth. First, in Section V-A, we describe our construction of bandwidth-optimal convertible codes in the case where we only consider fixed unique values for the final parameters n^F and $k^F = \lambda^I k^I$. Then, in Section V-B, we show that initial codes built with this construction are not only (n^F, k^F)-bandwidth-optimally convertible, but also simultaneously bandwidth-optimally convertible for multiple other values of the pair (n^F, k^F). Additionally, we present a construction which given any finite set of possible final parameter values (n^F, k^F), constructs an initial $[n^I, k^I]$ code which is simultaneously (n^F, k^F)-bandwidth-optimally convertible for every (n^F, k^F) in that set.

A. Bandwidth-Optimal MDS Convertible Codes for Fixed Final Parameters

The case where $r^F \geq k^I$ is trivial, since the default approach to conversion is bandwidth-optimal in this case. Therefore, in the rest of this section, we only consider $r^F < k^I$. Moreover, in the case where $r^I \geq r^F$ (decreasing-redundancy region), access-optimal convertible codes (for which explicit constructions are known) are also bandwidth-optimal. Therefore, we focus on the case $r^I < r^F$ (increasing-redundancy region).

We start by describing the base code used in our construction, followed by the design of piggybacks, and then describe the conversion procedure along with the role of piggybacks during conversion. To help illustrate the construction, we keep a running example showing each step.

1) Base Code for Piggybacking: As the base code for our construction, we use a *punctured* initial code of an access-optimal $(k^I + r^F, k^I; n^F, k^F)$ convertible code. Any access-optimal convertible code can be used. However, as mentioned in Section II-B, we assume that this convertible code is: (1) systematic, (2) linear, and (3) only requires accessing the first r^F parities from each initial codeword during access-optimal conversion. We refer to the $[k^I + r^F, k^I]$ initial code of this access-optimal convertible code as $\mathcal{C}^{I'}$, to its $[n^F, k^F]$ final code as $\mathcal{C}^{F'}$. Let $\mathcal{C}^{I''}$ be the punctured version of $\mathcal{C}^{I'}$ where the last $(r^F - r^I)$ parity symbols are punctured.

Example 2: Suppose we want to construct a bandwidth-optimal $(5, 4; 10, 8)$ convertible code over a finite field $\mathbb{F}_q$ (assume that q is sufficiently large). As a base code, we use a punctured access-optimal $(6, 4; 10, 8)$ convertible code. For this example, we use the code presented in Example 1 and puncture the last parity. Thus, $\mathcal{C}^{I'}$ is a $[6, 4]$ code, $\mathcal{C}^{F'}$ is a $[10, 8]$ code, and $\mathcal{C}^{I''}$ is a $[5, 4]$ code. ▶

2) Piggyback Design: Now, we describe how to construct the $[n^I, k^I, \alpha]$ initial vector code $\mathcal{C}^I$ and the $[n^F, k^F, \alpha]$ final vector code $\mathcal{C}^F$ that make up the bandwidth-optimal $(n^I, k^I; n^F, \lambda^I k^I)$ convertible code.

The first step is to choose the value of α. Let us reexamine the lower bound derived in Theorem 4 for $r^I < r^F < k^I$, which is rewritten below in a different form.

$$\gamma \geq \lambda^I \left(r^I \alpha + k^I \left(1 - \frac{r^I}{r^F} \right) \alpha \right) + r^F \alpha.$$

We can see that one way to achieve this lower bound would be to download exactly $\beta_1 = \alpha$ subsymbols from each of the r^I retired nodes in the λ^I initial codewords, and to download $\beta_2 = (1 - r^I/r^F)\,\alpha$ subsymbols from each of the k^I unchanged nodes in the λ^I initial stripes. Thus, we choose $\alpha = r^F$, which is the smallest value that makes β_1 and β_2 integers, thus making:

$$\beta_1 = r^F \qquad \text{and} \qquad \beta_2 = (r^F - r^I).$$

The next step is to design the piggybacks. We first provide the intuition behind the design. Recall from above that we can download $\beta_2 = (r^F - r^I)$ subsymbols from each unchanged node and all the α subsymbols from each retired node. Hence, we can utilize up to $\beta_2 = (r^F - r^I)$ coordinates from each of the r^I parity nodes for piggybacking. Given that there are precisely $(r^F - r^I)$ punctured symbols and α instances of $\mathcal{C}^{I''}$, we can store piggybacks corresponding to r^I instances of each of these punctured symbols. During conversion, these punctured symbols can be reconstructed and used for constructing the new nodes.

Consider a message $\mathbf{m} \in \mathbb{F}_q^{\lambda^I k^I \alpha}$ split into $\lambda^I \alpha$ submessages $\mathbf{m}_j^{(s)} \in \mathbb{F}_q^{k^I}$, representing the data encoded by instance $j \in [\alpha]$ of the base code in initial codeword $s \in [\lambda^I]$. Recall that $\mathcal{C}^{I''}$ is systematic by construction. Therefore, the submessage $\mathbf{m}_j^{(s)}$ will correspond to the contents of the j-th coordinate of the k^I systematic nodes in initial codeword s. Let $c_{i,j}^I(s)$ denote the contents of the j-th coordinate of parity symbol i in initial codeword s under code $\mathcal{C}^I$, and $c_{i,j}^F$ let denote the same for the single final codeword encoded under $\mathcal{C}^F$. These are constructed as follows:

$$c_{i,j}^I(s) = \begin{cases} \mathbf{m}_j^{(s)} \mathbf{p}_i^I, & \text{for } \begin{array}{r} s \in [\lambda^I], \\ i \in [r^I], \\ 1 \leq j \leq r^I \end{array} \\ \mathbf{m}_j^{(s)} \mathbf{p}_i^I + \mathbf{m}_i^{(s)} \mathbf{p}_j^I, & \text{for } \begin{array}{r} s \in [\lambda^I], \\ i \in [r^I], \\ r^I < j \leq r^F \end{array} \end{cases}$$

$$c_{i,j}^F = [\mathbf{m}_j^{(1)} \cdots \mathbf{m}_j^{(\lambda^I)}] \mathbf{p}_i^F, \qquad \text{for } i \in [r^F],\ j \in [r^F],$$

where $\mathbf{p}_i^I$ corresponds to the encoding vector of the i-th parity of $\mathcal{C}^{I'}$ and $\mathbf{p}_i^F$ corresponds to the encoding vector of the i-th parity of $\mathcal{C}^{F'}$. By using the access-optimal conversion procedure from the base code, we can compute $c_{i,j}^F = [\mathbf{m}_j^{(1)} \cdots \mathbf{m}_j^{(\lambda^I)}] \mathbf{p}_i^F$ from $\{\mathbf{m}_j^{(s)} \mathbf{p}_i^I : s \in [\lambda^I]\}$ for all $i \in [r^F]$ and $j \in [r^F]$. Notice that each initial codeword is independent and encoded in the same way (as required).

This piggybacking design, that of using parity code subsymbols of the base code as piggybacks, is inspired by one of the piggybacking designs proposed in [10], where it is used for efficiently reconstructing failed (parity) code symbols.

Example 2 (Continued): Let $\mathbf{p}_1^I, \mathbf{p}_2^I \in \mathbb{F}_q^{4\times 1}$ be the encoding vectors for the parities of $\mathcal{C}^{I'}$, and $\mathbf{p}_1^F, \mathbf{p}_2^F \in \mathbb{F}_q^{8\times 1}$ be the encoding vector for the parities of $\mathcal{C}^{F'}$. Since $\alpha = r^F = 2$, we construct a $[5, 4, 2]$ initial vector code $\mathcal{C}^I$ and a $[10, 8, 2]$ final vector code $\mathcal{C}^F$. Let $\mathbf{a} = (a_1, \ldots, a_8)$ and $\mathbf{b} = (b_1, \ldots, b_8)$. Figure 5 shows the resulting piggybacked codes encoding submessages $\mathbf{a}^{(1)} = (a_1, \ldots, a_4), \mathbf{a}^{(2)} = (a_5, \ldots, a_8), \mathbf{b}^{(1)} = (b_1, \ldots, b_4), \mathbf{b}^{(2)} = (b_5, \ldots, b_8) \in \mathbb{F}_q^{1\times 4}$. Recall from Example 1, that $\mathbf{a}^{(1)} \mathbf{p}_i^I + \alpha_i^4 \mathbf{a}^{(2)} \mathbf{p}_i^I = \mathbf{a} \mathbf{p}_i^F$ for $i \in \{1, 2\}$ (and equivalently for $\mathbf{b}$). ▶

3) Conversion Procedure: Conversion proceeds as follows:

1) Download $D = \{\mathbf{m}_j^{(s)} : s \in [\lambda^I]$ and $r^I < j \leq r^F\}$, $C_1 = \{c_{i,j}^I(s) : s \in [\lambda^I],\ i \in [r^I],$ and $1 \leq j \leq r^I\}$, and $C_2 = \{c_{i,j}^I(s) : s \in [\lambda^I],\ i \in [r^I],$ and $r^I < j \leq r^F\}$.
2) Recover the piggybacks $C_3 = \{\mathbf{m}_j^{(s)} \mathbf{p}_i^I : s \in [\lambda^I],\ r^I < i \leq r^F,$ and $1 \leq j \leq r^I\}$ by computing $\mathbf{m}_i^{(s)} \mathbf{p}_j^I$ from D and obtaining $\mathbf{m}_j^{(s)} \mathbf{p}_i^I = c_{j,i}^I(s) - \mathbf{m}_i^{(s)} \mathbf{p}_j^I$ using C_2.
3) Compute the remaining base code symbols from the punctured symbols $C_4 = \{\mathbf{m}_i^{(s)} \mathbf{p}_j^I : s \in [\lambda^I],\ r^I < i \leq r^F,$ and $r^I < j \leq r^F\}$ using D.
4) Compute the parity nodes of the final codeword specified by the subsymbols $C_5 = \{c_{i,j}^F : i \in [r^F],\ j \in [r^F]\}$. This is done by using the conversion procedure from the access-optimal convertible code used as base code to compute C_5 from C_1, C_2, C_3, and C_4.

This procedure requires downloading β_1 subsymbols from each retired node and β_2 subsymbols from each unchanged node. Thus, the read conversion bandwidth is:

$$\begin{aligned} \gamma_R &= \lambda^I \left(r^I \beta_1 + k^I \beta_2 \right) \\ &= \lambda^I \left(r^I \alpha + k^I \left(1 - \frac{r^I}{r^F} \right) \alpha \right). \end{aligned}$$

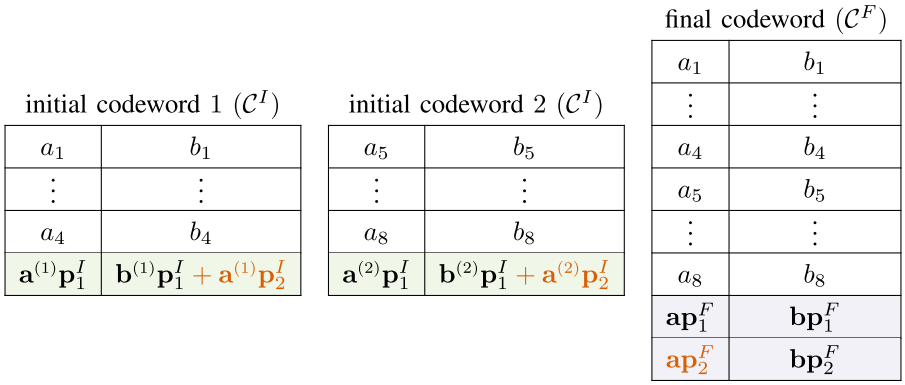

Fig. 5. Example of a bandwidth-optimal $(5, 4; 10, 8)$ convertible code. Each block in this diagram represents a codeword, where each column corresponds to a distinct coordinate of the α-length vector ($\alpha = 2$ in this case), and each row corresponds to a node. The shaded rows correspond to retired nodes for the first two blocks (initial codewords), and new nodes for the third block (final codeword). For the initial codewords, text color is used emphasize the piggybacks. In the final codeword, text color is used to denote the base code subsymbol that is constructed from the piggybacks.

Additionally, $r^F\alpha$ write conversion bandwidth is required for the new nodes.

$$\gamma_W = r^F\alpha$$

Since $\gamma = \gamma_R + \gamma_W$, this matches Theorem 4.

Example 2 (Continued): During conversion, only 12 subsymbols need to be downloaded: $\mathbf{b}^{(1)}, \mathbf{b}^{(2)}$ and all the parity symbols from both codewords. From these subsymbols, we can recover the piggyback terms $\mathbf{a}^{(1)}\mathbf{p}_2^I$ and $\mathbf{a}^{(2)}\mathbf{p}_2^I$, and then compute $\mathbf{b}^{(1)}\mathbf{p}_2^I$ and $\mathbf{b}^{(2)}\mathbf{p}_2^I$ in order to reconstruct the second parity symbol of $\mathcal{C}^{I'}$. Finally, we use $\mathbf{a}^{(i)}\mathbf{p}_1^I, \mathbf{b}^{(i)}\mathbf{p}_1^I, \mathbf{a}^{(i)}\mathbf{p}_2^I, \mathbf{b}^{(i)}\mathbf{p}_2^I$ for $i \in \{1, 2\}$ with the conversion procedure from the access-optimal convertible code to compute the base code symbols $\mathbf{a}\ \mathbf{p}_1^F, \mathbf{a}\ \mathbf{p}_2^F, \mathbf{b}\ \mathbf{p}_1^F$ and $\mathbf{b}\ \mathbf{p}_2^F$ of the new nodes.

The default approach would require one to download 16 subsymbols in total from the initial nodes. Both approaches require downloading 4 subsymbols in total from the coordinator node to the new nodes. Thus, the proposed construction leads to 20% reduction in conversion bandwidth as compared to the default approach of reencoding. ▶

B. Convertible Codes With Bandwidth-Optimal Conversion for Multiple Final Parameters

In practice, the final parameters n^F, k^F might depend on observations made after the initial encoding of the data and hence they may be unknown at code construction time. In particular, for a $(n^I, k^I; n^F, \lambda^I k^I)$ convertible code in the merge regime this means that the values of λ^I and $r^F = (n^F - k^F)$ are unknown.

To ameliorate this problem, we now present convertible codes which support bandwidth-optimal conversion *simultaneously* for multiple possible values of the final parameters. Recall property (4) of the access-optimal base code which we reviewed in Section II-B: when constructed with a given value of $\lambda^I = \lambda$ and $r^F = r$, the initial $[n^I, k^I]$ code is (n^F, k^F)-access-optimally convertible for all $k^F = \lambda' k^I$ and $n^F = k^F + r'$ such that $1 \leq \lambda' \leq \lambda$ and $1 \leq r' \leq r$.

1) Supporting Multiple Values of λ^I: The construction from Section V with a particular value of $\lambda^I = \lambda$, intrinsically supports bandwidth-optimal conversion for any $\lambda^I = \lambda' < \lambda$. This is a consequence of property (4) above, and can be done easily by considering one or multiple of the initial codewords as consisting of zeroes only, and ignoring them during conversion. From Theorem 4, it is easy to see that this modified conversion procedure achieves the optimal conversion bandwidth for the new parameter $\lambda^I = \lambda'$.

2) Supporting Multiple Values of r^F: We break this scenario into two cases:

Case 1 (supporting $r^F \leq r^I$): due to property (4) above, the base code used in the construction from Section V supports access-optimal conversion for any value of $r^F = r$ such that $r \leq r^I$. Using this property, one can achieve bandwidth optimality for any $r \leq r^I$ by simply using the access-optimal conversion on each of the α instances of the base code independently. The only difference is that some of the instances might have piggybacks, which can be simply ignored. The final code might still have these piggybacks, however they will still satisfy the property that the piggybacks in instance i ($2 \leq i \leq \alpha$) only depend on data from instances $\{1, \ldots, (i-1)\}$. Thus, the final code will have the MDS property and the desired parameters.

Case 2 (supporting $r^F > r^I$): for supporting multiple values of $r^F \in \{r_1, r_2, \ldots, r_s\}$ such that $r_i > r^I$ ($i \in [s]$), we start with an access-optimal convertible code having $r^F = \max_i r_i$. Then we repeat the piggybacking step of the construction (see Section V-A) for each r_i, using the resulting code from step i (with the punctured symbols from $\mathcal{C}^{I'}$ added back) as a base code for step $(i+1)$. Therefore, the resulting code will have $\alpha = \prod_{i=1}^{s} r_i$. Since the piggybacking step will preserve the MDS property of its base code, and the initial code used in the first piggyback step is MDS, it is clear that the initial code resulting from the last piggybacking step will also be MDS. Conversion for one of the supported $r^F = r_i$ is performed as described in Section V-A on each of the additional instances created by steps $(i+1), \ldots, s$ (i.e. $\prod_{i'=(i+1)}^{s} r_{i'}$ in total). As before, some of these instances after conversion will have piggybacks, which can be simply ignored, as the resulting code will continue to have the property that piggybacks from a given instance only depend on data from earlier instances.

Example 3 (Bandwidth-Optim Conversion for Multiple Final Parameters): In this example, we will extend the $(5, 4; 10, 8)$ convertible code from Example 2 ($r^F = 2$) to

initial codeword i ($\mathcal{C}^I$)

a_{4i-3}	b_{4i-3}	c_{4i-3}	d_{4i-3}	e_{4i-3}	f_{4i-3}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
a_{4i}	b_{4i}	c_{4i}	d_{4i}	e_{4i}	f_{4i}
$\mathbf{a}^{(i)}\mathbf{p}_1^I$	$\mathbf{b}^{(i)}\mathbf{p}_1^I + \mathbf{a}^{(i)}\mathbf{p}_2^I$	$\mathbf{c}^{(i)}\mathbf{p}_1^I + \mathbf{a}^{(i)}\mathbf{p}_2^I$	$\mathbf{d}^{(i)}\mathbf{p}_1^I + \mathbf{c}^{(i)}\mathbf{p}_2^I + \mathbf{b}^{(i)}\mathbf{p}_2^I$	$\mathbf{e}^{(i)}\mathbf{p}_1^I + \mathbf{a}^{(i)}\mathbf{p}_3^I$	$\mathbf{f}^{(i)}\mathbf{p}_1^I + \mathbf{e}^{(i)}\mathbf{p}_2^I + \mathbf{b}^{(i)}\mathbf{p}_3^I$

final codeword ($r^F = 2$)

a_1	b_1	c_1	d_1	e_1	f_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
a_8	b_8	c_8	d_8	e_8	f_8
$\mathbf{a}\mathbf{p}_1^F$	$\mathbf{b}\mathbf{p}_1^F$	$\mathbf{c}\mathbf{p}_1^F + \mathbf{a}^{(1)}\mathbf{p}_2^I + \mathbf{a}^{(2)}\mathbf{p}_2^I$	$\mathbf{d}\mathbf{p}_1^F$	$\mathbf{e}\mathbf{p}_1^F + \mathbf{a}^{(1)}\mathbf{p}_3^I + \mathbf{a}^{(2)}\mathbf{p}_3^I$	$\mathbf{f}\mathbf{p}_1^F$
$\mathbf{a}\mathbf{p}_2^F$	$\mathbf{b}\mathbf{p}_2^F$	$\mathbf{c}\mathbf{p}_2^F$	$\mathbf{d}\mathbf{p}_2^F$	$\mathbf{e}\mathbf{p}_2^F$	$\mathbf{f}\mathbf{p}_2^F$

final codeword ($r^F = 3$)

a_1	b_1	c_1	d_1	e_1	f_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
a_8	b_8	c_8	d_8	e_8	f_8
$\mathbf{a}\mathbf{p}_1^F$	$\mathbf{b}\mathbf{p}_1^F + \mathbf{a}^{(1)}\mathbf{p}_2^I + \mathbf{a}^{(2)}\mathbf{p}_2^I$	$\mathbf{c}\mathbf{p}_1^F$	$\mathbf{d}\mathbf{p}_1^F$	$\mathbf{e}\mathbf{p}_1^F$	$\mathbf{f}\mathbf{p}_1^F$
$\mathbf{a}\mathbf{p}_2^F$	$\mathbf{b}\mathbf{p}_2^F$	$\mathbf{c}\mathbf{p}_2^F$	$\mathbf{d}\mathbf{p}_2^F$	$\mathbf{e}\mathbf{p}_2^F$	$\mathbf{f}\mathbf{p}_2^F$
$\mathbf{a}\mathbf{p}_3^F$	$\mathbf{b}\mathbf{p}_3^F$	$\mathbf{c}\mathbf{p}_3^F$	$\mathbf{d}\mathbf{p}_3^F$	$\mathbf{e}\mathbf{p}_3^F$	$\mathbf{f}\mathbf{p}_3^F$

Fig. 6. Example of a $[5, 4]$ MDS code that supports bandwidth-optimal conversion to multiple final codes. This code supports bandwidth-optimal conversion to a $[8 + r, 8]$ MDS code for $r = 1, 2, 3$. Piggybacks from the first round ($r = 2$) are colored orange and piggybacks from the second round ($r = 3$) are colored magenta. In the possible final codewords, text color is used to show base code symbols which are directly computed from the corresponding piggybacks, or to denote leftover piggybacks that were not used during conversion.

construct a code which additionally supports bandwidth-optimal conversion to an $[11, 8]$ MDS code ($r^F = 3$). Figure 6 shows initial codeword $i \in \{1, 2\}$ of the new initial vector code, which has $\alpha = 2 \cdot 3 = 6$. Here $\mathbf{a}^{(1)} = (a_1, \ldots, a_4)$, $\mathbf{a}^{(2)} = (a_5, \ldots, a_8) \in \mathbb{F}_q^{1\times 4}$, $\mathbf{a} = (a_1, \ldots, a_8) \in \mathbb{F}_q^{1\times 8}$, and similarly for $\mathbf{b}, \ldots, \mathbf{f}$. The vectors $\mathbf{p}_i^I \in \mathbb{F}_q^{4\times 1}$ are the encoding vectors of the initial code $\mathcal{C}^{I'}$ and $\mathbf{p}_i^F \in \mathbb{F}_q^{8\times 1}$ are encoding vectors of the final code $\mathcal{C}^{F'}$ ($i \in \{1, 2, 3\}$). Since the maximum supported r^F is 3, we start with an access-optimal $(7, 4; 11, 8)$ convertible code. Thus, $\mathcal{C}^{I'}$ is a $[7, 4]$ code, $\mathcal{C}^{F'}$ is a $[11, 8]$ code, and $\mathcal{C}^{I''}$ is a $[5, 4]$ code. In the first round of piggybacking we consider $r^F = 2$, which yields the code shown in Example 2. In the second round of piggybacking we consider $r^F = 3$ and piggyback the code resulting from the first round, which yields the code shown in Figure 6. Conversion for $r^F = 1$ proceeds by simply downloading the contents of the single parity node and using the access-optimal conversion procedure. Conversion for $r^F = 2$ proceeds by treating this code as three instances of the code from Example 2 and performing conversion for each one independently. Conversion for $r^F = 3$ proceeds by treating this code as a vector code with $\alpha = 3$ and base field $\mathbb{F}_{q^2}$ (i.e. each element is a vector over $\mathbb{F}_q$ of length 2).

Remark 5 (Field Size Requirement): The field size requirement for $\mathbb{F}_q$ of the constructions presented in this section is given by the field size requirement of the base code used. The currently lowest known field size requirement for an explicit construction of systematic linear access-optimal convertible codes in the merge regime is given by [7]. For typical parameters, this requirement is roughly $q \geq \Omega(2^{\lambda^I (n^I)^3})$. When $r^F \leq r^I - \lambda^I + 1$, this can be significantly reduced to $q \geq k^I r^I$. And when $r^F \leq \lfloor r^I/\lambda^I \rfloor$, this can be further reduced to $q \geq \max\{n^I, n^F\}$. ▶

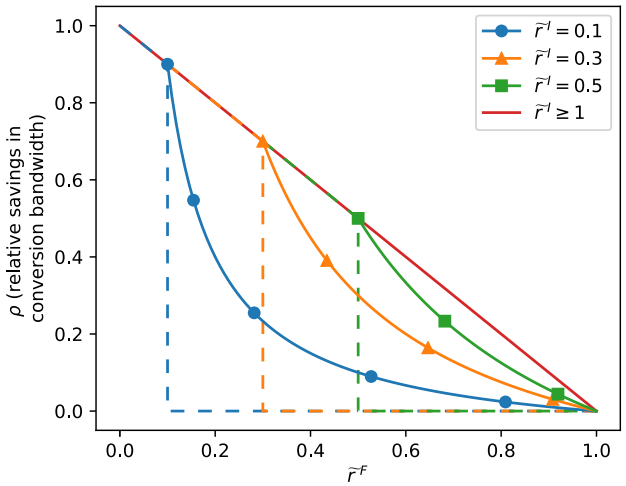

Fig. 7. Achievable savings in conversion bandwidth by bandwidth-optimal convertible codes in comparison to the default approach to conversion. Here $\tilde{r}^I = r^I/k^I$ and $\tilde{r}^F = r^F/k^I$ are the initial and final redundancies, divided by the initial code dimension. Each curve shows the relative savings for a fixed value of $\tilde{r}^I$, as $\tilde{r}^F$ varies. Solid lines indicate bandwidth-optimal convertible codes, and dashed lines indicate access-optimal convertible codes. Notice that each curve overlaps with the red curve ($\tilde{r}^I \geq 1$) in the range $\tilde{r}^F \in (0, \tilde{r}^I]$.

VI. Bandwidth Savings of Bandwidth-Optimal Convertible Codes

In this section, we show the amount of savings in bandwidth that can be obtained by using bandwidth-optimal convertible codes in the merge regime, relative to the default approach to conversion. We present the amount of savings in terms of two ratios:

$$\tilde{r}^I = (r^I/k^I) \qquad \text{and} \qquad \tilde{r}^F = (r^F/k^I),$$

i.e. the initial and final amount of "redundancy" relative to the initial dimension of the code. For simplicity, we only consider

the read conversion bandwidth (data sent from initial nodes to the coordinator node), since the write conversion bandwidth (data sent from the coordinator node to the new nodes) is fixed for stable convertible codes (specifically, it is equal to αr^F). Thus, the conversion bandwidth of the default approach is always $\lambda^I k^I \alpha$. Figure 7 shows the relative savings, i.e. the ratio between the conversion bandwidth of optimal conversion and the conversion bandwidth of the default approach, for fixed values of $\widetilde{r}^I \in (0, \infty)$ and varying $\widetilde{r}^F \in (0, \infty)$.

Each curve shown in Figure 7 can be divided into three regions, depending on the value of $\widetilde{r}^F$:

- **Region $0 < \widetilde{r}^F \leq \widetilde{r}^I$ and $\widetilde{r}^F < 1$:** these conditions imply that $r^F \leq r^I$, so by Lemma 2 the conversion bandwidth is $\lambda^I r^F \alpha$, and the relative savings are:

$$\rho = 1 - \frac{\lambda^I r^F \alpha}{\lambda^I k^I \alpha} = 1 - \widetilde{r}^F.$$

  This region corresponds to the decreasing-redundancy region, and in this region access-optimal convertible codes are also bandwidth-optimal. This region of the curve is linear, and the amount of savings is not affected by $\widetilde{r}^I$.
- **Region $\widetilde{r}^I < \widetilde{r}^F < 1$:** this implies that $r^I \leq r^F \leq k^I$, and by Lemma 2 the conversion bandwidth is $\lambda^I \alpha (r^I + k^I(1 - r^I/r^F))$, and the relative savings are:

$$\rho = 1 - \frac{\lambda^I \alpha \left(r^I + k^I \left(1 - \frac{r^I}{r^F} \right) \right)}{\lambda^I k^I \alpha} = \widetilde{r}^I \left(\frac{1}{\widetilde{r}^F} - 1 \right).$$

  This corresponds to the increasing-redundancy region, where access-optimal convertible codes provide no conversion bandwidth savings. Thus bandwidth-optimal convertible codes provide substantial savings in conversion bandwidth in this regime, compared to access-optimal convertible codes.
- **Region $\widetilde{r}^F \geq 1$:** this implies that $r^F \geq k^I$ and by Lemma 2 a conversion bandwidth of $\lambda^I k^I \alpha$ is required. Thus no savings in conversion bandwidth are possible in this region.

Thus, bandwidth-optimal convertible codes allow for savings in conversion bandwidth on a much broader region relative to access-optimal convertible codes.

VII. Conclusion and Future Directions

In this paper, we studied the conversion bandwidth of convertible codes. We showed that the conversion problem can be effectively modeled using network information flow to obtain lower bounds on conversion bandwidth. Using the bounds derived, we showed that for the merge regime access-optimal convertible codes are also bandwidth optimal when $r^I \geq r^F$ (increasing-redundancy region) and that there is room for reducing conversion bandwidth when $r^I < r^F$ (decreasing-redundancy region). We proposed an explicit construction which achieves the optimal conversion bandwidth for all parameters in the merge regime. Finally, we showed that bandwidth-optimal convertible codes can achieve substantial savings in conversion bandwidth over the default approach and access-optimal convertible codes.

This work leads to several open problems. One of the main open problems is to extend the conversion bandwidth lower bounds and bandwidth-optimal constructions to encompass all possible parameter regimes (i.e. the general regime). Another important open problem is characterizing the optimal value of α, especially in the case of multiple possible final parameter values, where α can become very large when using the construction proposed in this paper. Yet another open problem is lowering the field size requirement of bandwidth-optimal convertible code constructions, as well as deriving lower bounds for their field size requirements.

References

[1] F. Maturana and K. V. Rashmi, "Bandwidth cost of code conversions in distributed storage: Fundamental limits and optimal constructions," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 2334–2339.

[2] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proc. 19th ACM Symp. Operating Syst. Princ.*, M. L. Scott and L. L. Peterson, Eds. Bolton Landing, NY, USA, Oct. 2003, pp. 29–43.

[3] D. Borthakur, R. Schmidt, R. Vadali, S. Chen, and P. Kling. *HDFS RAID—Facebook*. Accessed: Jul. 23, 2019. [Online]. Available: http://www.slideshare.net/ydn/hdfs-raid-facebook

[4] C. Huang et al., "Erasure coding in Windows azure storage," in *Proc. USENIX Annu. Tech. Conf.*, G. Heiser and W. C. Hsieh, Eds. Boston, MA, USA: USENIX Association, Jun. 2012, pp. 15–26.

[5] Apache Software Foundation. *Apache Hadoop: HDFS Erasure Coding*. Accessed: Jul. 23, 2019. [Online]. Available: https://hadoop.apache.org/docs/r3.0.0/hadoop-project-dist/hadoop-hdfs/HDFSErasureCoding.html

[6] S. Kadekodi, K. V. Rashmi, and G. R. Ganger, "Cluster storage systems gotta have heart: Improving storage efficiency by exploiting disk-reliability heterogeneity," in *Proc. 17th USENIX Conf. File Storage Technol. (FAST)*, A. Merchant and H. Weatherspoon, Eds. Boston, MA, USA: USENIX Association, Feb. 2019, pp. 345–358.

[7] F. Maturana and K. V. Rashmi, "Convertible codes: New class of codes for efficient conversion of coded data in distributed storage," in *Proc. 11th Innov. Theor. Comput. Sci. Conf. (ITCS)*, vol. 151, T. Vidick, Ed. Seattle, Washington, DC, USA: Schloss Dagstuhl-Leibniz-Zentrum für Informatik, Jan. 2020, p. 66.

[8] F. Maturana, V. S. C. Mukka, and K. V. Rashmi, "Access-optimal linear MDS convertible codes for all parameters," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angeles, CA, USA, Jun. 2020, pp. 577–582.

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

[10] K. Rashmi, N. B. Shah, and K. Ramchandran, "A piggybacking design framework for read-and download-efficient distributed storage codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 9, pp. 5802–5820, Sep. 2017.

[11] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, vol. 16. Amsterdam, The Netherlands: Elsevier, 1977.

[12] F. Maturana and K. V. Rashmi, "Convertible codes: Enabling efficient conversion of coded data in distributed storage," *IEEE Trans. Inf. Theory*, vol. 68, no. 7, pp. 4392–4407, Jul. 2022.

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

[14] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.

[15] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[16] T. Ho et al., "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[17] P. Sanders, S. Egner, and L. Tolhuizen, "Polynomial time algorithms for network information flow," in *Proc. 15th Annu. ACM Symp. Parallel Algorithms Archit.*, A. L. Rosenberg and F. M. auf der Heide, Eds. San Diego, CA, USA, Jun. 2003, pp. 286–294.

[18] S. Jaggi et al., "Polynomial time algorithms for multicast network code construction," *IEEE Trans. Inf. Theory*, vol. 51, no. 6, pp. 1973–1982, Jun. 2005.

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

[20] K. V. Rashmi, N. B. Shah, and K. Ramchandran, "A piggybacking design framework for read-and download-efficient distributed storage codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Istanbul, Turkey, Jul. 2013, pp. 331–335.

[21] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Enabling node repair in any erasure code for distributed storage," in *Proc. IEEE Int. Symp. Inf. Theory*, A. Kuleshov, V. M. Blinovsky, and A. Ephremides, Eds. St. Petersburg, Russia, Jul. 2011, pp. 1235–1239.

[22] S. Mousavi, T. Zhou, and C. Tian, "Delayed parity generation in MDS storage codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 1889–1893.

[23] M. Xia, M. Saxena, M. Blaum, and D. Pease, "A tale of two erasure codes in HDFS," in *Proc. 13th USENIX Conf. File Storage Technol. (FAST)*, J. Schindler and E. Zadok, Eds. Santa Clara, CA, USA: USENIX Association, Feb. 2015, pp. 213–226.

[24] X. Su, X. Zhong, X. Fan, and J. Li, "Local re-encoding for coded matrix multiplication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angeles, CA, USA, Jun. 2020, pp. 221–226.

[25] S. Wu, Z. Shen, and P. P. C. Lee, "Enabling I/O-efficient redundancy transitioning in erasure-coded KV stores via elastic Reed–Solomon codes," in *Proc. Int. Symp. Reliable Distrib. Syst. (SRDS)*, Shanghai, China, Sep. 2020, pp. 246–255.

[26] 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*, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.

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

[28] C. Suh and K. Ramchandran, "Exact-repair MDS code construction using interference alignment," *IEEE Trans. Inf. Theory*, vol. 57, no. 3, pp. 1425–1442, Mar. 2011.

[29] V. R. Cadambe, C. Huang, J. Li, and S. Mehrotra, "Polynomial length MDS codes with optimal repair in distributed storage," in *Proc. Conf. Rec. 45th Asilomar Conf. Signals, Syst. Comput. (ASILOMAR)*, M. B. Matthews, Eds. Pacific Grove, CA, USA, Nov. 2011, pp. 1850–1854.

[30] Z. Wan, I. Tamo, and J. Bruck, "On codes for optimal rebuilding access," in *Proc. 49th Annu. Allerton Conf. Commun., Control, Comput.*, Monticello, IL, USA, Sep. 2011, pp. 1374–1381.

[31] 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*, vol. 58, no. 4, pp. 2134–2158, Apr. 2012.

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

[33] D. Papailiopoulos, A. G. Dimakis, and V. Cadambe, "Repair optimal erasure codes through Hadamard designs," *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 3021–3037, May 2013.

[34] A. Chowdhury and A. Vardy, "New constructions of MDS codes with asymptotically optimal repair," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 1944–1948.

[35] K. Mahdaviani, S. Mohajer, and A. Khisti, "Product matrix MSR codes with bandwidth adaptive exact repair," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 3121–3135, Apr. 2018.

[36] B. Sasidharan, M. Vajha, and P. V. Kumar, "An explicit, coupled-layer construction of a high-rate MSR code with low sub-packetization level, small field size and d $<$ (n - 1)," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Germany, Jun. 2017, pp. 2048–2052.

[37] M. Ye and A. Barg, "Explicit constructions of high-rate MDS array codes with optimal repair bandwidth," *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 2001–2014, Apr. 2017.

[38] 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*, vol. 64, no. 10, pp. 6506–6525, Oct. 2018.

[39] S. Goparaju, A. Fazeli, and A. Vardy, "Minimum storage regenerating codes for all parameters," *IEEE Trans. Inf. Theory*, vol. 63, no. 10, pp. 6318–6328, Oct. 2017.

[40] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A 'hitchhiker's' guide to fast and efficient data reconstruction in erasure-coded data centers," in *Proc. ACM Conf. SIGCOMM*, F. E. Bustamante, Y. C. Hu, A. Krishnamurthy, and S. Ratnasamy, Eds. Chicago, IL, USA, Aug. 2014, pp. 331–342.

[41] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the Facebook warehouse cluster," in *Proc. 5th USENIX Workshop Hot Topics Storage File Syst.*, A. Gulati, Ed. San Jose, CA, USA: USENIX Association, Jun. 2013, pp. 1–67.

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

[43] Z. Wang, I. Tamo, and J. Bruck, "Long MDS codes for optimal repair bandwidth," in *Proc. IEEE Int. Symp. Inf. Theory*, Cambridge, MA, USA, Jul. 2012, pp. 1182–1186.

[44] N. B. Sha, K. V. Rashmi, and P. V. Kumar, "A flexible class of regenerating codes for distributed storage," in *Proc. IEEE Int. Symp. Inf. Theory*, Austin, TX, USA, Jun. 2010, pp. 1943–1947.

[45] M. Ye and A. Barg, "Explicit constructions of MDS array codes and RS codes with optimal repair bandwidth," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Barcelona, Spain, Jul. 2016, pp. 1202–1206.

[46] O. Alrabiah and V. Guruswami, "An exponential lower bound on the sub-packetization of MSR codes," in *Proc. 51st Annu. ACM SIGACT Symp. Theory Comput.*, M. Charikar and E. Cohen, Eds. Phoenix, AZ, USA, Jun. 2019, pp. 979–985.

[47] S. B. Balaji and P. V. Kumar, "A tight lower bound on the sub-packetization level of optimal-access MSR and MDS codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 2381–2385.

[48] I. Tamo, Z. Wang, and J. Bruck, "Access versus bandwidth in codes for storage," *IEEE Trans. Inf. Theory*, vol. 60, no. 4, pp. 2028–2037, Apr. 2014.

[49] H. Dau, I. Duursma, H. M. Kiah, and O. Milenkovic, "Repairing Reed–Solomon codes with multiple erasures," *IEEE Trans. Inf. Theory*, vol. 64, no. 10, pp. 6567–6582, Oct. 2018.

[50] V. Guruswami and M. Wootters, "Repairing Reed–Solomon codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 9, pp. 5684–5698, Sep. 2017.

[51] J. Li, X. Tang, and C. Tian, "A generic transformation to enable optimal repair in MDS codes for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 64, no. 9, pp. 6257–6267, Sep. 2018.

[52] J. Mardia, B. Bartan, and M. Wootters, "Repairing multiple failures for scalar MDS codes," *IEEE Trans. Inf. Theory*, vol. 65, no. 5, pp. 2661–2672, May 2019.

[53] K. Shanmugam, D. S. Papailiopoulos, A. G. Dimakis, and G. Caire, "A repair framework for scalar MDS codes," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 998–1007, May 2014.

[54] I. Tamo, M. Ye, and A. Barg, "Optimal repair of Reed–Solomon codes: Achieving the cut-set bound," in *Proc. IEEE 58th Annu. Symp. Found. Comput. Sci. (FOCS)*, Berkeley, CA, USA, C. Umans, Ed. Oct. 2017, pp. 216–227.

[55] K. W. Shum, "Cooperative regenerating codes for distributed storage systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kyoto, Japan, Jun. 2011, pp. 1–5.

[56] G. M. Kamath, N. Prakash, V. Lalitha, and P. V. Kumar, "Codes with local regeneration and erasure correction," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 4637–4660, Aug. 2014.

[57] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 10, pp. 5843–5855, Oct. 2014.

[58] M. Ye, "New constructions of cooperative MSR codes: Reducing node size to exp(O(n))," *IEEE Trans. Inf. Theory*, vol. 66, no. 12, pp. 7457–7464, Dec. 2020.

[59] G. Zhang, W. Zheng, and J. Shu, "ALV: A new data redistribution approach to RAID-5 scaling," *IEEE Trans. Comput.*, vol. 59, no. 3, pp. 345–357, Mar. 2010.

[60] W. Zheng and G. Zhang, "Fastscale: Accelerate RAID scaling by minimizing data migration," in *Proc. 9th USENIX Conf. File Storage Technol.*, G. R. Ganger and J. Wilkes, Eds. San Jose, CA, USA, Feb.2011, pp. 149–161.

[61] C. Wu and X. He, "GSR: A global stripe-based redistribution approach to accelerate RAID-5 scaling," in *Proc. 41st Int. Conf. Parallel Process.*, Pittsburgh, PA, USA, Sep. 2012, pp. 460–469.

[62] G. Zhang, W. Zheng, and K. Li, "Rethinking RAID-5 data layout for better scalability," *IEEE Trans. Comput.*, vol. 63, no. 11, pp. 2816–2828, Nov. 2014.

[63] J. Huang, X. Liang, X. Qin, P. Xie, and C. Xie, "Scale-RS: An efficient scaling scheme for RS-coded storage clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 6, pp. 1704–1717, Jun. 2015.

[64] S. Wu, Y. Xu, Y. Li, and Z. Yang, "I/O-efficient scaling schemes for distributed storage systems with CRS codes," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2639–2652, Sep. 2016.

[65] X. Zhang, Y. Hu, P. P. C. Lee, and P. Zhou, "Toward optimal storage scaling via network coding: From theory to practice," in *Proc. IEEE Conf. Comput. Commun.*, Honolulu, HI, USA, Apr. 2018, pp. 1808–1816.

[66] Y. Hu, X. Zhang, P. P. C. Lee, and P. Zhou, "Generalized optimal storage scaling via network coding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 956–960.

[67] X. Y. Zhang and Y. C. Hu, "Efficient storage scaling for MBR and MSR codes," *IEEE Access*, vol. 8, pp. 78992–79002, 2020.

[68] B. K. Rai, V. Dhoorjati, L. Saini, and A. K. Jha, "On adaptive distributed storage systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Hong Kong, Jun. 2015, pp. 1482–1486.

[69] B. K. Rai, "On adaptive (functional MSR code based) distributed storage systems," in *Proc. Int. Symp. Netw. Coding (NetCod)*, Sydney, NSW, Australia, Jun. 2015, pp. 46–50.

[70] S. Wu, Z. Shen, and P. P. C. Lee, "On the optimal repair-scaling trade-off in locally repairable codes," in *Proc. IEEE Conf. Comput. Commun.*, Jul. 2020, pp. 2155–2164.

Francisco Maturana (Student Member, IEEE) received the B.S. and M.S. degrees in computer science from the Pontificia Universidad de Chile, Santiago, Chile, in 2017. He is currently pursuing the Ph.D. degree with the Computer Science Department, Carnegie Mellon University, USA. His research interests include the intersection of theoretical computer science and computer systems.

K. V. Rashmi (Member, IEEE) received the Ph.D. degree from UC Berkeley in 2016. She was a Post-Doctoral Scholar with UC Berkeley from 2016 to 2017. She is currently an Assistant Professor with the Computer Science Department, Carnegie Mellon University. Her research interests include information/coding theory and computer/networked systems. Her work has received the USENIX NSDI 2021 Community (Best Paper) Award and the IEEE Data Storage Best Paper and Best Student Paper Awards for the years 2011/2012. During her Ph.D. studies, she was a recipient of the Facebook Fellowship from 2012 to 2013, the Microsoft Research Ph.D. Fellowship from 2013 to 2015, and the Google Anita Borg Memorial Scholarship from 2015 to 2016. Her Ph.D. thesis was awarded the UC Berkeley Eli Jury Dissertation Award in 2016. She was a recipient of the Facebook Communications and Networking Research Award in 2017, the Google Faculty Research Award in 2018, the Facebook Distributed Systems Research Award in 2019, the VMWare Systems Research Award in 2021, the Tata Institute of Fundamental Research Memorial Lecture Award in 2020, and the NSF CAREER Award from 2020 to 2025.