Cooperative Conversion of MDS Codes in the Merge Regime

Myna Vajha
IIT Hyderabad, India
mynaramana@gmail.com

Xiangliang Kong
Bilkent University, Turkey
rongxlkong@gmail.com

M. Nikhil Krishnan
IIT Palakkad, India
nikhilkrishnan.m@gmail.com

Vinayak Ramkumar
TU Munich, Germany
vinram93@gmail.com

***Abstract*—Recent work by Kadekodi et al. has demonstrated the advantages of code conversions in distributed storage systems. A key performance metric in this context is the bandwidth cost, defined as the total number of symbols transmitted during conversion. In this work, we focus on merge conversions, where multiple codewords of an initial vector MDS code are combined into a single codeword of a final vector MDS code. The optimal bandwidth cost for such merge conversions was established by Maturana and Rashmi for the centralized setting utilizing a central coordinator. In contrast, we propose a cooperative conversion framework in which nodes exchange data directly, without a central coordinator. Although the centralized scheme of Maturana and Rashmi can be adapted to the cooperative conversion setting to yield some bandwidth reduction, we present a new convertible code construction that exploits a specific strong access property to achieve further reduced bandwidth costs for a large range of parameters.**

I. Introduction

Erasure codes are widely used in distributed storage systems to provide resilience against disk failures. An $[n, k]$ code over a finite field $\mathbb{F}_q$ encodes k data symbols into n coded symbols, which are distributed across distinct storage nodes. Typically, the parameters $\{n, k\}$ are static, fixed at deployment, based on an anticipated disk failure rate. However, failure rates in large-scale clusters vary significantly over time [1]–[4]. While adapting code parameters to these changing conditions is desirable for maintaining storage efficiency, performing such adaptations via naive re-encoding incurs prohibitive I/O and bandwidth overheads [5], [6].

To address this challenge, Maturana and Rashmi introduced the framework of *convertible codes* [7]. Specifically, an $(n^I = k^I + r^I, k^I; n^F = k^F + r^F, k^F)$ convertible code enables the efficient transition from an initial $[n^I, k^I]$ code $\mathcal{C}^I$ to a final $[n^F, k^F]$ code $\mathcal{C}^F$. The efficiency of the conversion is evaluated in terms of two metrics: (a) *access cost*: the number of disks accessed and (b) *bandwidth cost*: the total number of symbols transferred across the network. Recently, there has been significant research interest in convertible codes [8]–[17].

In this paper, we focus on the *merge* conversion regime, where $\lambda \geq 2$ codewords from an initial Maximum Distance Separable (MDS) code are consolidated into a single codeword of a final MDS code. For this setting, Maturana and Rashmi [7] derived a tight upper bound on the access cost. Convertible codes that achieve this bound are referred to as *access-optimal*.

While access cost is minimized by reading the smallest sufficient subset of symbols, minimizing bandwidth cost often necessitates the use of *vector codes*. An $[n, k, \alpha]$ vector code over a base field $\mathbb{F}_q$ encodes k message symbols from $\mathbb{F}_q^\alpha$ into a codeword of n symbols from $\mathbb{F}_q^\alpha$. The parameter $\alpha > 1$ is referred to as the *sub-packetization level* and elements of $\mathbb{F}_q$ are referred to as *sub-symbols*. Such a code is a vector MDS code if any k code symbols suffice to recover the entire codeword. In a vector $(n^I, k^I; n^F, k^F; \alpha)$ convertible code, the initial and final codes are vector codes over the alphabet $\mathbb{F}_q^\alpha$. We denote the parameters of the initial and final vector codes as $[n^I, k^I, \alpha]$ and $[n^F, k^F, \alpha]$, respectively. A sub-packetization $\alpha > 1$ allows the system to read and transmit only a fraction of a code symbol from each accessed node, a technique analogous to that employed in regenerating codes [18]–[20]. The fundamental limits of bandwidth cost for vector MDS codes in the merge regime were characterized in [8]. The optimal bandwidth cost (normalized with respect to α) is given by:

$$\gamma = \begin{cases} \lambda \min\{k^I, r^F\} + r^F, & \text{if } r^I \geq r^F \text{or } k^I \leq r^F, \\ \lambda(r^I + k^I(1 - r^I/r^F)) + r^F, & \text{otherwise.} \end{cases} \quad (1)$$

However, the analysis in [8] is restricted to a *centralized* setting, where a designated coordinator node aggregates data from surviving nodes to generate the final code symbols.

Our Contributions: In this work, we study the bandwidth cost of vector MDS convertible codes under the merge regime. In contrast to [8], we consider a more general cooperative setting where nodes can exchange data directly without a central coordinator. Specifically, our contributions are twofold:

1) *Cooperative Conversion Framework:* We propose a novel *cooperative conversion* framework for vector MDS codes, enabling direct data exchange between nodes.
2) *Convertible Codes Under the Cooperative Framework*: We propose a new convertible coding scheme for the cooperative setting. While the bandwidth-optimal centralized schemes of [8] can be adapted to this setting, we show that our new proposed scheme achieves a strictly lower bandwidth cost for a large set of parameters.

Notation: We use $\mathbb{Z}$ to denote the set of all integers. Let $[a : b] \triangleq \{i \in \mathbb{Z} \mid a \leq i \leq b\}$. When $a = 1$, we use the shorthand $[b] \triangleq [1 : b]$. Let A be an $m \times n$ matrix with rows indexed by $[0 : m-1]$ and columns indexed by $[0 : n-1]$. Its (i, j)-entry is denoted by $A(i, j)$. For $R \subseteq [0 : m-1]$ and $C \subseteq [0 : n-1]$, the submatrices obtained by restricting A to

rows in R, columns in C, or both, are denoted by $A(R,:)$, $A(:,C)$, and $A(R,C)$, respectively.

II. Cooperative Conversion Framework

In this section, we introduce our cooperative code conversion framework. To describe this framework, we classify nodes into four categories:

1) *unchanged nodes:* nodes storing symbols that are common to both the initial and final codewords,
2) *changed nodes:* nodes present before and after conversion, whose stored code symbols are updated during the conversion,
3) *retired nodes:* nodes that store initial code symbols but not final code symbols,
4) *new nodes:* nodes that were not present before conversion and are newly added to store some of the final code symbols.

We denote the sets of unchanged, changed, retired and new nodes by $\mathcal{U}$, $\mathcal{V}$, $\mathcal{R}$ and $\mathcal{N}$, respectively. While prior work [8] considered only unchanged, retired and new nodes, we introduce an additional category in this work: changed nodes. In [8], the optimal bandwidth cost for the merge regime was determined for a centralized conversion setting, where a coordinator node (not storing code symbols) downloads data from unchanged and retired nodes, generates new code symbols and transfers them to new nodes. In this work, we consider a framework where nodes exchange data directly without a central coordinator node. We refer to this as cooperative conversion, motivated by similar frameworks in the regenerating codes literature [21].

In this work, we focus on the merge conversion of vector MDS codes. Consider $\lambda \geq 2$ codewords of an $[n^I = k^I + r^I, k^I, \alpha]$ initial vector MDS code $\mathcal{C}^I$. Our goal is to merge these codewords into a single codeword of an $[n^F = k^F + r^F, k^F = \lambda k^I, \alpha]$ final vector MDS code $\mathcal{C}^F$. As in [8], we assume that the (vector) code symbols of the initial codewords are stored on distinct disks, and similarly for the final codeword. Before the conversion, there are λn^I nodes storing α sub-symbols, i.e., one vector symbol of an initial codeword. These λn^I initial nodes can be divided into three categories: unchanged nodes, changed nodes and retired nodes. We index the initial nodes by $[0 : \lambda n^I - 1]$ and hence $\mathcal{U} \cup \mathcal{V} \cup \mathcal{R} = [0 : \lambda n^I - 1]$.After the conversion, there are n^F nodes storing α sub-symbols each, i.e., one vector symbol of the final codeword. Among the final n^F nodes, we have unchanged nodes, changed nodes and new nodes. We index the final nodes by $[0 : n^F - 1]$ and hence $\mathcal{U} \cup \mathcal{V} \cup \mathcal{N} = [0 : n^F - 1]$. Unchanged nodes and changed nodes are part of both initial and final codewords. Unchanged nodes store the same code symbols before and after conversion, and thus no communication is needed for them. On the other hand, code symbols stored in the changed nodes differ before and after conversion. If $n^F \geq \lambda n^I$, there are no retired nodes; conversely, if $n^F \leq \lambda n^I$, there are no new nodes.

We propose a conversion framework that consists of two communication phases. In the first phase, the changed nodes and the new nodes (i.e., nodes in $\mathcal{V} \cup \mathcal{N}$) download sub-symbols from the initial nodes (i.e., nodes in $\mathcal{U} \cup \mathcal{V} \cup \mathcal{R}$). See Fig. 1 for an illustration. After the first phase, each changed node generates new sub-symbols using the sub-symbols downloaded in the first phase together with the initial code symbols it stores. Note that no communication is required for a changed node to access the initial code symbols it stores. Each new node generates new sub-symbols using the data it received in the first phase. In the second phase, the changed nodes and the new nodes (i.e., nodes in $\mathcal{V} \cup \mathcal{N}$) exchange newly generated sub-symbols to assemble the final code symbols to be stored in the changed nodes and the new nodes. See Fig. 2 for an illustration. The bandwidth cost of this cooperative conversion procedure is the total amount of data that is exchanged in the two phases.

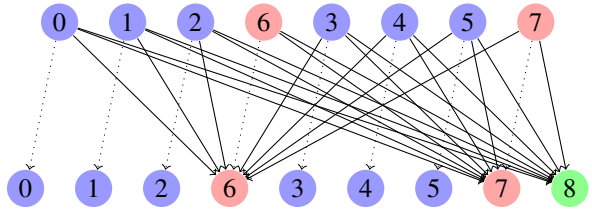

Fig. 1: An illustration of the first phase of cooperative conversion. Here $n^I = 4, \lambda = 2, n^F = 9, k^I = 3$. The eight initial nodes are shown at the top. Among the initial nodes, nodes $0, 1, 2$, and 6 store one initial codeword, while nodes $3, 4, 5$, and 7 store another. The nine final nodes are shown at the bottom. The six unchanged nodes (blue) and the two changed nodes (red) are part of both initial and final nodes. There is one new node (green). The solid lines indicate data transfer that contributes towards the first phase bandwidth. Dotted lines connect the same physical node and hence do not contribute to bandwidth.

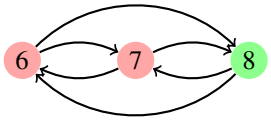

Fig. 2: An illustration of the second phase of cooperative conversion. This is a continuation of Fig. 1, in which there are two changed nodes (red) and one new node (green). The solid lines indicate data transfer that contributes towards the second phase bandwidth.

To define bandwidth cost more formally, let $\beta^{(1)}_{i\to j}$ denote the number of symbols over $\mathbb{F}_q$ (i.e., sub-symbols) transferred from node i to node j during the first phase and let $\beta^{(2)}_{i\to j}$ be the corresponding quantity for the second phase. By definition, for all possible i, $\beta^{(1)}_{i\to i} = \beta^{(2)}_{i\to i} = 0$. The *normalized cooperative conversion bandwidth cost* is defined as $\gamma := \gamma_1 + \gamma_2$ where

$$\gamma_1 := \frac{1}{\alpha} \sum_{i \in \mathcal{U}\cup\mathcal{V}\cup\mathcal{R}} \sum_{j \in \mathcal{V}\cup\mathcal{N}} \beta^{(1)}_{i\to j} \quad \text{and}$$

$$\gamma_2 := \frac{1}{\alpha} \sum_{i \in \mathcal{V}\cup\mathcal{N}} \sum_{j \in \mathcal{V}\cup\mathcal{N}} \beta^{(2)}_{i\to j}.$$

Here, γ_1 and γ_2 denote the normalized bandwidth costs of the first and second phases, respectively. We use normalized conversion bandwidth cost as the performance metric so that schemes with different α can be compared.

A. Modified Maturana-Rashmi Scheme

In [8], Maturana and Rashmi presented a construction of vector MDS convertible codes in the merge regime that builds on an access-optimal MDS convertible code and incorporates the piggybacking framework. The piggybacking framework introduced in [22] is a technique for obtaining vector codes using a scalar code as a *base code*. In the following, we modify the centralized conversion procedure in [8] to suit the cooperative conversion setting, while keeping the code construction unchanged. As in [8], the sub-packetization level is set to r^F. Both the initial and final codes in [8] are linear and systematic. For convenience, in a systematic code we refer to nodes storing information symbols as *systematic nodes* and those storing parity symbols as *parity nodes*.

During the conversion process, the λk^I systematic nodes of the λ initial codewords remain unchanged in the final codeword and are classified as unchanged nodes. We next specify the categories of the r^F parity nodes of the final codeword. If $\lambda r^I \geq r^F$, r^F final parity nodes are selected arbitrarily from the λr^I initial parity nodes and are classified as changed nodes. If $\lambda r^I < r^F$, all λr^I initial parity nodes are retained in the final codeword with updated contents and are classified as changed nodes, and $r^F - \lambda r^I$ new nodes are introduced.

One of the selected changed nodes acts like the coordinator node in the centralized scheme from [8]. In the first phase of conversion, this coordinator node downloads data as follows. When $r^F \geq k^I$, it downloads all λk^I message symbols from the unchanged nodes. When $r^F \leq \min\{r^I, k^I\}$, it downloads r^F parity symbols from each of the initial codewords. When $r^I < r^F < k^I$, it downloads the last $r^F - r^I$ sub-symbols of the λk^I information symbols from the unchanged nodes, as well as all parity symbols from the λr^I initial parity nodes. Note that for the last two cases above, the coordinator node itself stores one of the symbols it needs to download. Thus $\gamma_1 = \lambda k^I$ when $r^F \geq k^I$, $\gamma_1 = \lambda r^F - 1$ when $r^F \leq \min\{r^I, k^I\}$, and $\gamma_1 = \lambda k^I(1 - r^I/r^F) + \lambda r^I - 1$ when $r^I < r^F < k^I$. Using the downloaded sub-symbols, the coordinator node can compute all r^F final parity symbols as described in Section V-A of [8]. In the second phase, the coordinator node (which is one of the final parity nodes) distributes the corresponding contents to the remaining $r^F - 1$ final parity nodes. This results in $\gamma_2 = r^F - 1$. In summary, the cooperative version of the conversion procedure in [8] yields a vector MDS convertible code with normalized cooperative conversion bandwidth:

$$\gamma_{\text{MR}} = \begin{cases} \lambda(r^I + k^I(1 - \frac{r^I}{r^F})) + r^F - 2, & \text{for } r^I < r^F < k^I, \\ \lambda k^I + r^F - 1, & \text{for } r^F \geq k^I, \\ \lambda r^F + r^F - 2, & \text{for } r^F \leq \min(r^I, k^I). \end{cases} \tag{2}$$

Thus, adapting the Maturana-Rashmi scheme to the cooperative conversion framework achieves a strictly lower bandwidth cost than the centralized bound (1). This straightforward adaptation is mentioned in Remark 1 of [8]. In the next section, we demonstrate that leveraging a stronger notion of access-optimality in code design yields further non-trivial bandwidth reductions in the cooperative setting.

III. A New Cooperative Conversion Scheme

In this section, we present a vector MDS convertible code construction tailored for the cooperative setting that achieves a smaller normalized conversion bandwidth than γ_{MR}. To achieve this, the construction exploits a property of scalar base codes we term *strong per-symbol access-optimality*.

A. Strong Per-Symbol Access-Optimality

In this subsection, we introduce the specific property of scalar base codes used in our construction. Let $G^I \triangleq [I_{k^I}\ P^I]$ and $G^F \triangleq [I_{k^F}\ P^F]$ denote the generator matrices of the initial $[n^I = k^I + r, k^I]$ and final $[n^F = k^F + r, k^F = \lambda k^I]$ MDS base codes, respectively. In both codes, the first k^I (resp. k^F) symbols of a codeword are message symbols and the remaining r symbols are parity symbols. We focus on scalar MDS base codes that possess the following property.

Definition 1 (Strongly Parallel Block Reconstructibility)**.** Let P be a $k^F \times r$ superregular matrix, with $k^F = \lambda k^I$. For each $l \in [\lambda]$, let $P^{(l)}$ be the submatrix formed by the k^I consecutive rows of P from index $(l-1)k^I$ to $lk^I - 1$, i.e., $P^{(l)} \triangleq P((l-1)k^I : lk^I - 1, :)$. Let $P^{(l)}$ be expressed in terms of its columns as $P^{(l)} = [\mathbf{p}_0^{(l)}\ \mathbf{p}_1^{(l)}\ \dots\ \mathbf{p}_{r-1}^{(l)}]$. The matrix P is said to be (k^I, k^F, r)-*strongly-parallel-block-reconstructible (SPBR)* if for every $l \in [\lambda]$ and every column index $i \in [0 : r-1]$, $\mathbf{p}_i^{(l)}$ is a scalar multiple of $\mathbf{p}_i^{(1)}$.

We note that the SPBR property is a special case of the block reconstructibility and the parallel block reconstructibility defined in [7] and [13], respectively. The MDS base convertible code is defined to be $(k^I + r, k^I; \lambda k^I + r, \lambda k^I)$ *strong per-symbol access-optimal* if P^F is (k^I, k^F, r)-SPBR and $P^I = P^F(0 : k^I - 1, :)$. For such a code, the i-th parity symbol of the final codeword is simply a linear combination of the i-th parity symbols from each of the λ initial codewords. Consequently, constructing each final parity symbol requires accessing only λ symbols (one from each initial codeword), while the message symbols remain unchanged.

If P is a superregular Vandermonde matrix (see, e.g., [7], [11]), then its j-th row, $j \in [0 : k^F - 1]$, is given by $P(j, :) = [\alpha_1^{j-1}\ \alpha_2^{j-1}\ \cdots \alpha_r^{j-1}]$. It can be verified that such a matrix P satisfies $\mathbf{p}_i^{(l)} = \alpha_i^{(l-1)k^I}\mathbf{p}_i^{(1)}$ for all $i \in [0 : r-1], l \in [\lambda]$ and therefore satisfies the (k^I, k^F, r)-SPBR property. To the best of our knowledge, aside from the constructions in [7] and [11], no other access-optimal convertible code satisfies this stronger property. The existing superregular Vandermonde constructions, however, possess an $O((k^F)^r)$ field-size requirement, and it remains an open question to construct SPBR matrices over small finite fields.

B. Convertible Code Construction

We present our code construction and corresponding conversion scheme for the case where $r^I \leq r^F$. For the case $r^F < r^I$, the cooperative conversion scheme based on [8] from Section II-A can be employed.

In our construction, we set $\alpha = r^F$. Hence, the initial and final codes are $[n^I = k^I + r^I, k^I, \alpha = r^F]$ and

$[n^F = k^F + r^F, k^F = \lambda k^I, \alpha = r^F]$ vector MDS codes, respectively. Both our initial and final codes are constructed using the piggybacking framework, unlike the approach in [8], which applies the piggybacking framework only to the initial code. Furthermore, the method of adding piggybacked symbols in our initial code differs from that in [8]. To differentiate between the two constructions, we refer to ours as the Modified Piggybacked Convertible Code (MPCC).

1) Base code: As the base code for piggybacking, we use a $(k^I + r^F, k^I; \lambda k^I + r^F, \lambda k^I)$ strong per-symbol access-optimal convertible code. Let $\mathcal{C}^{I'}$ denote the $[k^I + r^F, k^I]$ initial MDS code and $\mathcal{C}^{F'}$ the $[n^F, k^F]$ final MDS code of this strong per-symbol access-optimal convertible code. Note that the construction based on [8] relies on access-optimality. The strong per-symbol access-optimality enables the bandwidth savings achieved in the cooperative framework compared to γ_{MR} (given by (2)).

Let $[I_{k^F} \ P^F]$ and $[I_{k^I} \ P^I]$ be the generator matrices of $\mathcal{C}^{F'}$ and $\mathcal{C}^{I'}$, respectively, where $P^I = P^F([0 : k^I - 1], :)$. We write $P^F = [\mathbf{p}_0^F \cdots \mathbf{p}_{r^F-1}^F]$ and $P^I = [\mathbf{p}_0 \cdots \mathbf{p}_{r^F-1}]$. By definition of strong per-symbol access-optimality, P^F is (k^I, k^F, r^F)-SPBR and $\mathbf{p}_i^{(l)} = a_{l,i}\mathbf{p}_i$ for some $a_{l,i} \in \mathbb{F}_q \setminus \{0\}$ for all $i \in [0 : r^F - 1]$ and $l \in [\lambda]$.

2) Initial and final codes: We now describe the initial and final codes of MPCC. We assume that $r^F \geq r^I$ and let $\alpha = r^F = cr^I + d$ for integers $c \geq 1$ and $d \in [0 : r^I - 1]$. The initial code $\mathcal{C}^I$ is an $[n^I, k^I, \alpha = r^F]$ vector MDS code constructed by applying the piggybacking framework to the initial base code $\mathcal{C}^{I'}$. Note that although the initial base code has r^F parity symbols, the initial code of the MPCC has only r^I parities. Consider λ messages $\mathbf{m}^l \in \mathbb{F}_q^{k^I\alpha}$, $l \in [\lambda]$. Let $\mathbf{m}^l = (\mathbf{m}_0^l, \ldots, \mathbf{m}_{\alpha-1}^l)$, where $\mathbf{m}_j^l = (m_{0,j}^l, \ldots, m_{k^I-1,j}^l) \in \mathbb{F}_q^{k^I}$. Each symbol of a codeword of $\mathcal{C}^I$ is an element in $\mathbb{F}_q^\alpha$. Let $(\mathbf{c}_0^l, \mathbf{c}_1^l, \ldots, \mathbf{c}_{n^I-1}^l)$ denote the l-th initial codeword of $\mathcal{C}^I$, where $\mathbf{c}_i^l = (c_{i,0}^l, c_{i,1}^l, \ldots, c_{i,\alpha-1}^l) \in \mathbb{F}_q^\alpha$.

To define the initial code, we now describe the l-th initial codeword as a function of message $\mathbf{m}^l$. The code is systematic and we have $c_{i,j}^l = m_{i,j}^l$ for $i \in [0 : k^I - 1], j \in [0 : \alpha - 1]$. The r^I parity symbols of the l-th codeword of $\mathcal{C}^I$ are defined as follows. For $j \in [0 : \alpha - 1]$ and $i \in [0 : r^I - 1]$, let $j = j_1 r^I + j_2$ for non-negative integers j_1, j_2 such that $j_2 < r^I$.

$$c_{k^I+i,j}^l = \begin{cases} \mathbf{m}_j^l \mathbf{p}_i & \text{for } j_1 = 0, \\ \mathbf{m}_j^l \mathbf{p}_i + \mathbf{m}_i^l \mathbf{p}_j & \text{for } j_1 = c, i \geq d, \\ \mathbf{m}_j^l \mathbf{p}_i + \mathbf{m}_{j_2}^l \mathbf{p}_{j_1 r^I + i} & \text{otherwise.} \end{cases} \tag{3}$$

An example of an initial codeword is shown in Table I.

The final code $\mathcal{C}^F$ is an $[n^F = k^F + r^F, k^F = \lambda k^I, \alpha = r^F]$ vector MDS code. Let $(\hat{\mathbf{c}}_0, \hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_{n^F-1})$ be the merged final codeword of $\mathcal{C}^F$, where $\hat{\mathbf{c}}_i = (\hat{c}_{i,0}, \hat{c}_{i,1}, \ldots, \hat{c}_{i,\alpha-1}) \in \mathbb{F}_q^\alpha$. This code is also systematic, with $\hat{c}_{(l-1)k^I+i,j} = m_{i,j}^l$ for $i \in [0 : k^I - 1], l \in [\lambda], j \in [0 : \alpha - 1]$. We now describe the r^F parity symbols of the final codeword as a function of messages $\mathbf{m}^1, \ldots, \mathbf{m}^\lambda$. For $i \in [0 : r^F - 1]$ and $j \in [0 : \alpha - 1]$, let $i = i_1 r^I + i_2$ with $0 \leq i_2 < r^I$. Then:

$$\hat{c}_{k^F+i,j} = \begin{cases} \mathbf{m}_j \mathbf{p}_i^F + f_{i,j}(\mathbf{m}_{i_1 r^I + j}) & \text{for } j < r^I \leq i, i_1 r^I + j < r^F, \\ \mathbf{m}_j \mathbf{p}_i^F + f_{i,j}(\mathbf{m}_i) & \text{for } j < r^I \leq i, i_1 = c, j \geq d, \\ \mathbf{m}_j \mathbf{p}_i^F & \text{otherwise,} \end{cases} \tag{4}$$

where $\mathbf{m}_j = (\mathbf{m}_j^1, \cdots, \mathbf{m}_j^\lambda) \in \mathbb{F}_q^{\lambda k^I}$. The piggyback interference terms $f_{i,j}(\cdot)$ are defined as: $f_{i,j}(m_{i_1 r^I + j}) = \sum_{l \in [\lambda]} a_{l,i} \mathbf{m}_{i_1 r^I + j} p_{i_2}$ and $f_{i,j}(m_i) = \sum_{l \in [\lambda]} a_{l,i} \mathbf{m}_i p_j$, corresponding to the first and second cases of (4), respectively.

It can be seen that both $\mathcal{C}_I$ and $\mathcal{C}_F$ are MDS codes; see Appendix A. We now present an example to illustrate the construction described above.

Example 1. Let $r^I = 2$ and $r^F = 5$. For $l \in [\lambda]$, the l-th initial codeword is shown in Table I and the final codeword is shown in Table II. The number of columns corresponds to the sub-packetization level $\alpha = r^F = 5$. For the initial codeword, $k^I\alpha$ message symbols are shown in the first row across α columns. The next 2 rows correspond to the r^I parity symbols. The initial codeword construction method described in (3) can be viewed as taking $\alpha = 5$ codewords of a punctured code of the initial base code $\mathcal{C}^{I'}$ and then adding piggybacks as shown in red below.

$\mathbf{m}_0^l$	$\mathbf{m}_1^l$	$\mathbf{m}_2^l$	$\mathbf{m}_3^l$	$\mathbf{m}_4^l$
$\mathbf{m}_0^l\mathbf{p}_0$	$\mathbf{m}_1^l\mathbf{p}_0$	$\mathbf{m}_2^l\mathbf{p}_0 + \mathbf{m}_0^l\mathbf{p}_2$	$\mathbf{m}_3^l\mathbf{p}_0 + \mathbf{m}_1^l\mathbf{p}_2$	$\mathbf{m}_4^l\mathbf{p}_0 + \mathbf{m}_0^l\mathbf{p}_4$
$\mathbf{m}_0^l\mathbf{p}_1$	$\mathbf{m}_1^l\mathbf{p}_1$	$\mathbf{m}_2^l\mathbf{p}_1 + \mathbf{m}_0^l\mathbf{p}_3$	$\mathbf{m}_3^l\mathbf{p}_1 + \mathbf{m}_1^l\mathbf{p}_3$	$\mathbf{m}_4^l\mathbf{p}_1 + \mathbf{m}_1^l\mathbf{p}_4$

TABLE I: The l-th initial codeword for $r^I = 2$, $r^F = 5$.

For the final codeword, the first row corresponds to $\lambda k^I \alpha$ message symbols spread across $\alpha = 5$ columns. The final codeword is formed by placing codewords of the final base code $\mathcal{C}^{F'}$ in the columns and adding the piggybacks shown in red below.

$\mathbf{m}_0$	$\mathbf{m}_1$	$\mathbf{m}_2$	$\mathbf{m}_3$	$\mathbf{m}_4$
$\mathbf{m}_0\mathbf{p}_0^F$	$\mathbf{m}_1\mathbf{p}_0^F$	$\mathbf{m}_2\mathbf{p}_0^F$	$\mathbf{m}_3\mathbf{p}_0^F$	$\mathbf{m}_4\mathbf{p}_0^F$
$\mathbf{m}_0\mathbf{p}_1^F$	$\mathbf{m}_1\mathbf{p}_1^F$	$\mathbf{m}_2\mathbf{p}_1^F$	$\mathbf{m}_3\mathbf{p}_1^F$	$\mathbf{m}_4\mathbf{p}_1^F$
$\mathbf{m}_0\mathbf{p}_2^F + f_{2,0}(\mathbf{m}_2)$	$\mathbf{m}_1\mathbf{p}_2^F + f_{2,1}(\mathbf{m}_3)$	$\mathbf{m}_2\mathbf{p}_2^F$	$\mathbf{m}_3\mathbf{p}_2^F$	$\mathbf{m}_4\mathbf{p}_2^F$
$\mathbf{m}_0\mathbf{p}_3^F + f_{3,0}(\mathbf{m}_2)$	$\mathbf{m}_1\mathbf{p}_3^F + f_{3,1}(\mathbf{m}_3)$	$\mathbf{m}_2\mathbf{p}_3^F$	$\mathbf{m}_3\mathbf{p}_3^F$	$\mathbf{m}_4\mathbf{p}_3^F$
$\mathbf{m}_0\mathbf{p}_4^F + f_{4,0}(\mathbf{m}_4)$	$\mathbf{m}_1\mathbf{p}_4^F + f_{4,1}(\mathbf{m}_4)$	$\mathbf{m}_2\mathbf{p}_4^F$	$\mathbf{m}_3\mathbf{p}_4^F$	$\mathbf{m}_4\mathbf{p}_4^F$

TABLE II: The final codeword for $r^I = 2$, $r^F = 5$.

C. Cooperative Conversion Scheme

To describe the conversion procedure, we first assign each node to one of the categories defined in Section II. There are λn^I initial nodes, among which λk^I are systematic nodes. The same holds for the n^F final nodes. The unchanged nodes are the λk^I systematic nodes and $\mathcal{U} = [0 : \lambda k^I - 1]$. The assignment of remaining nodes depends on whether $r^F \geq \lambda r^I$ or not. The j-th parity node of l-th initial codeword has index $\lambda k^I + (l-1)r^I + j$. See Fig. 1 for an illustration.

- $r^F \geq \lambda r^I$: In this case, the λr^I initial parity nodes become changed nodes and $r^F - \lambda r^I$ new nodes are

employed. Thus, $\mathcal{R}=\phi$, $\mathcal{N}=[\lambda k^I+\lambda r^I : n^F-1], \mathcal{V}=[\lambda k^I : \lambda k^I+\lambda r^I-1]$.

- $r^F<\lambda r^I$: There will be no new nodes as we can reuse r^F initial parity nodes among the available λr^I initial parity nodes. Thus, $\mathcal{N}=\phi, \mathcal{R}=[\lambda k^I+r^F : \lambda n^I-1], \mathcal{V}=[\lambda k^I : n^F-1]$.

Now we describe two examples of the conversion procedure (a) for the case when there are no new nodes, i.e., $r^F<\lambda r^I$ and (b) for the case when there are no retired nodes, i.e., $r^F \geq \lambda r^I$.

Example 2. Consider Example 1, where $r^I=2$ and $r^F=5$. Let $\lambda=2$. Then, $\lambda r^I=4<r^F=5$, and hence there is one new node. The node assignment is given by $\mathcal{R}=\emptyset, \mathcal{N}=\{2k^I+4\}, \mathcal{V}=\{2k^I, 2k^I+1, 2k^I+2, 2k^I+3\}$. Tables I and II describe the initial and final codewords. The data received by node $2k^I+j$ in both phases are listed below.

j	0	1	2	3	4
Phase 1	$\mathbf{m}_0^2\mathbf{p}_0$ $\mathbf{m}_1^2\mathbf{p}_0$	$\mathbf{m}_0^2\mathbf{p}_1$ $\mathbf{m}_1^2\mathbf{p}_1$	$\mathbf{m}_2$ $\mathbf{m}_2^1\mathbf{p}_0+\mathbf{m}_0^1\mathbf{p}_2$ $\mathbf{m}_3^1\mathbf{p}_0+\mathbf{m}_1^1\mathbf{p}_2$	$\mathbf{m}_3$ $\mathbf{m}_2^1\mathbf{p}_1+\mathbf{m}_0^1\mathbf{p}_3$ $\mathbf{m}_3^1\mathbf{p}_1+\mathbf{m}_1^1\mathbf{p}_3$	$\mathbf{m}_4$ $\mathbf{m}_4^1\mathbf{p}_0+\mathbf{m}_0^1\mathbf{p}_4$ $\mathbf{m}_4^2\mathbf{p}_0+\mathbf{m}_0^2\mathbf{p}_4$ $\mathbf{m}_4^1\mathbf{p}_1+\mathbf{m}_1^1\mathbf{p}_4$ $\mathbf{m}_4^2\mathbf{p}_1+\mathbf{m}_1^2\mathbf{p}_4$
Phase 2	$\mathbf{m}_2\mathbf{p}_0^F$ $\mathbf{m}_3\mathbf{p}_0^F$ $\mathbf{m}_4\mathbf{p}_0^F$	$\mathbf{m}_2\mathbf{p}_1^F$ $\mathbf{m}_3\mathbf{p}_1^F$ $\mathbf{m}_4\mathbf{p}_1^F$	 $\mathbf{m}_3\mathbf{p}_2^F$ $\mathbf{m}_4\mathbf{p}_2^F$	$\mathbf{m}_2\mathbf{p}_3^F$ $\mathbf{m}_4\mathbf{p}_3^F$	$\mathbf{m}_2\mathbf{p}_4^F$ $\mathbf{m}_3\mathbf{p}_4^F$

The node $2k^I+j$ stores the j-th final parity symbol after conversion, and hence is referred to as final parity node j. In Phase 1, for $j\in\{2,3,4\}$, node $2k^I+j$ downloads λk^I message sub-symbols $\mathbf{m}_j$ that allow it to compute the $r^F=5$ final parities shown in the j-th column of the final codeword in Table II. However, only $\mathbf{m}_j\mathbf{p}_j^F$ is the sub-symbol required by this node; the remaining four sub-symbols are communicated in Phase 2 as shown in the table above. For example, when $j=2$, sub-symbol $\mathbf{m}_2\mathbf{p}_{j'}^F$, $j'\in\{0,1,3,4\}$, is communicated to node $2k^I+j'$. Final parity node 0 is obtained by the 0-th initial parity node of the first initial codeword (i.e., node $2k^I$). Hence, it has access to $\mathbf{m}_i^1\mathbf{p}_0$ for $i=0,1$. Using the two sub-symbols downloaded in Phase 1 and the three sub-symbols received in Phase 2, it can recover the final parity sub-symbols $\mathbf{m}_i\mathbf{p}_0^F$ for $i=0,1$. Similarly, final parity node 1 can recover $\mathbf{m}_i p_1^F$ for $i=0,1$. It remains to show the recovery of the final parity sub-symbols in the bottom left 3×2 sub-matrix of Table II. Final parity node 2 is obtained from the 0-th initial parity of the 2-nd initial codeword (i.e. node $2k^I+2$), which has access to $\mathbf{m}_2^2\mathbf{p}_0+\mathbf{m}_0^2\mathbf{p}_2, \mathbf{m}_3^2\mathbf{p}_0+\mathbf{m}_1^2\mathbf{p}_2$. Using these sub-symbols along with parity sub-symbols $\mathbf{m}_2^1\mathbf{p}_0+\mathbf{m}_0^1\mathbf{p}_2, \mathbf{m}_3^1\mathbf{p}_0+\mathbf{m}_1^1\mathbf{p}_2$ downloaded in Phase 1, it can recover

$$\begin{aligned}\hat{c}_{2k^I+2,0} &= a_{1,2}(\mathbf{m}_2^1\mathbf{p}_0+\mathbf{m}_0^1\mathbf{p}_2)+a_{2,2}(\mathbf{m}_2^2\mathbf{p}_0+\mathbf{m}_0^2\mathbf{p}_2)\\ &= \mathbf{m}_0\mathbf{p}_2^F+f_{2,0}(\mathbf{m}_2)\end{aligned}$$

where the second equality follows from the SPBR property, and $\hat{c}_{2k^I+2,1}$ can be recovered in a similar manner. For final parity nodes $j\in\{3,4\}$, the recovery of $\hat{c}_{2k^I+j,i}$, $i\in\{0,1\}$, follows the same procedure. The only difference is that final parity node 4 is a new node and therefore requires $r^I=2$ additional sub-symbols, since it does not store any initial symbols. In this example, a total of $6k^I+24$ symbols are downloaded, yielding a normalized bandwidth cost of $\frac{6k^I}{5}+\frac{24}{5}$. This is smaller than γ_{MR}, which equals $\frac{6k^I}{5}+7$ for $k^I>5$ and $2k^I+4$ for $k^I\leq 5$.

We now consider an example in which there are no new nodes, but some retired nodes.

Example 3. Again, consider Example 1, where $r^I=2$ and $r^F=5$. Let $\lambda=3$. Then $r^F=5<\lambda r^I=6$. Thus, there is a retired node, given by $\mathcal{R}=\{3k^I+5\}$, and $\mathcal{N}=\emptyset$, $\mathcal{V}=\{3k^I, 3k^I+1, 3k^I+2, 3k^I+3, 3k^I+4\}$. Tables I and II illustrate the initial and final codewords, respectively. The data downloaded by node $3k^I+j$ in both phases are listed below.

j	0	1	2	3	4
Phase 1	$\mathbf{m}_0^2\mathbf{p}_0$ $\mathbf{m}_0^3\mathbf{p}_0$ $\mathbf{m}_1^2\mathbf{p}_0$ $\mathbf{m}_1^3\mathbf{p}_0$	$\mathbf{m}_0^2\mathbf{p}_1$ $\mathbf{m}_0^3\mathbf{p}_1$ $\mathbf{m}_1^2\mathbf{p}_1$ $\mathbf{m}_1^3\mathbf{p}_2$	$\mathbf{m}_2$ $\mathbf{m}_2^1\mathbf{p}_0+\mathbf{m}_0^1\mathbf{p}_2$ $\mathbf{m}_2^3\mathbf{p}_0+\mathbf{m}_0^3\mathbf{p}_2$ $\mathbf{m}_3^1\mathbf{p}_0+\mathbf{m}_1^1\mathbf{p}_2$ $\mathbf{m}_3^3\mathbf{p}_0+\mathbf{m}_1^3\mathbf{p}_2$	$\mathbf{m}_3$ $\mathbf{m}_2^1\mathbf{p}_1+\mathbf{m}_0^1\mathbf{p}_3$ $\mathbf{m}_2^3\mathbf{p}_1+\mathbf{m}_0^3\mathbf{p}_3$ $\mathbf{m}_3^1\mathbf{p}_1+\mathbf{m}_1^1\mathbf{p}_3$ $\mathbf{m}_3^3\mathbf{p}_1+\mathbf{m}_1^3\mathbf{p}_3$	$\mathbf{m}_4$ $\mathbf{m}_4^1\mathbf{p}_0+\mathbf{m}_0^1\mathbf{p}_4$ $\mathbf{m}_4^2\mathbf{p}_0+\mathbf{m}_0^2\mathbf{p}_4$ $\mathbf{m}_4^1\mathbf{p}_1+\mathbf{m}_1^1\mathbf{p}_4$ $\mathbf{m}_4^2\mathbf{p}_1+\mathbf{m}_1^2\mathbf{p}_4$ $\mathbf{m}_4^3\mathbf{p}_1+\mathbf{m}_1^3\mathbf{p}_4$
Phase 2	$\mathbf{m}_2\mathbf{p}_0^F$ $\mathbf{m}_3\mathbf{p}_0^F$ $\mathbf{m}_4\mathbf{p}_0^F$	$\mathbf{m}_2\mathbf{p}_1^F$ $\mathbf{m}_3\mathbf{p}_1^F$ $\mathbf{m}_4\mathbf{p}_1^F$	 $\mathbf{m}_3\mathbf{p}_2^F$ $\mathbf{m}_4\mathbf{p}_2^F$	$\mathbf{m}_2\mathbf{p}_3^F$ $\mathbf{m}_4\mathbf{p}_3^F$	$\mathbf{m}_2\mathbf{p}_4^F$ $\mathbf{m}_3\mathbf{p}_4^F$

The recovery procedure is similar to the earlier example and it can be checked that the symbols downloaded are enough to assemble the final parity symbols. The total number of downloaded symbols here is $9k^I+33$, yielding a normalized bandwidth cost of $\frac{9k^I}{5}+\frac{33}{5}$. In this case, γ_{MR} equals $\frac{9k^I}{5}+9$ for $k^I>5$ and $3k^I+4$ for $k^I\leq 5$, both of which are larger than the normalized bandwidth cost of this example when $k^I>2$.

We will now describe the general conversion scheme. The conversion procedure comprises two phases as described in Section II. We start by describing the Phase 1 first. Consider final node λk^I+j or the j-th final parity node.

1) For $j\in[0:r^I-1]$: j-th final parity node downloads following parity symbols:

$$\{c_{k+j,i}^l \mid i\in[0:r^I-1], l\in[2:\lambda]\}$$

Therefore, it downloads $(\lambda-1)r^I$ symbols in total. It has access to the symbols $\{c_{k+j,i}^1$ as it is a changed node and j-th parity in the first initial codeword. It can therefore recover the final sub-parity symbols $\hat{c}_{\lambda k+j,i}=\mathbf{m}_i\mathbf{p}_j^F$ from $\{c_{k+j,i}^l=\mathbf{m}_i^l\mathbf{p}_j \mid l\in[\lambda]\}$ for $i\in[0:r^I-1]$ due to the SPBR property.

2) For $j\in[r^I:r^F-1]$. This node could either be changed or a new node, depending on whether $r^F\geq\lambda r^I$ or not. Irrespective of that, it downloads the message symbols $\{\mathbf{m}_j\}$. These are same as the message symbols that appear in the j-th column of all the λ initial codeword tables. Node j therefore downloads a total of λk^I symbols. With these message symbols, node $\lambda k+j$ can recover the final parity symbols: $\hat{c}_{\lambda k^I+i,j}=\mathbf{m}_j\mathbf{p}_i^F$ for all $i\in[0:r^F-1]$. On top of these message symbols, j-th parity node downloads the following symbols.
   a) $j\geq\lambda r^I$: This case is possible only if $r^F>\lambda r^I$ and node λk^I+j is a new node. Suppose $j=j_1r^I+j_2$ for $j_2\in[0:r^I-1]$. We will show that in phase 1 by

downloading λr^I symbols node $\lambda k^I + j$ can recover the symbols $\{\hat{c}_{k^I\lambda+j,i} \mid i \in [0 : r^I - 1]\}$. If $j_1 < c$, node $\lambda k^I + j$ downloads the following λr^I symbols:

$$\{c^l_{k+j_2,i} \mid i \in [j_1 r^I : (j_1+1)r^I - 1], l \in [\lambda]\}. \quad (5)$$

Since $c^l_{k+j_2,i} = \mathbf{m}^l_i \mathbf{p}_{j_2} + \mathbf{m}^l_{j_1} \mathbf{p}_{i_2}$ where $i_2 = i \mod r^I$ and $\mathbf{p}^l_i = a_{l,i}\mathbf{p}_i$ from the SPBR property, the following computation results in the final parity symbols.

$$\begin{aligned}\sum_{l=1}^{\lambda} a_{l,i} c^l_{k^I+j_2,i} &= \sum_{l=1}^{\lambda} a_{l,i}(\mathbf{m}^l_i \mathbf{p}_{j_2} + \mathbf{m}^l_{i_2}\mathbf{p}_j) \\ &= \mathbf{m}_{i_2}\mathbf{p}^F_j + f_{j_2,i}(\mathbf{m}_i) \\ &= \hat{c}_{\lambda k^I+j,i_2}. \end{aligned} \quad (6)$$

where $i_2 = i \mod r^I$. When $j_1 = c$ it follows that $j_2 \in [0 : d-1]$. In this case the node $\lambda k^I + j$ downloads:

$$\begin{aligned}&\{c^l_{k^I+j_2,i} \mid l \in [\lambda], i \in [cr^I : cr^I + d - 1]\} \\ &\quad \cup \{c^l_{k^I+\hat{i},j} \mid l \in [\lambda],\ \hat{i} \in [d : r-1]\} \end{aligned} \quad (7)$$

From the initial code definition shown in equation (3) it follows that for $i \in [cr^I : cr^I + d - 1]$:

$$\sum_{l=1}^{\lambda} a_{l,j} c^l_{k^I+j_2,i} = \mathbf{m}_{i_2}\mathbf{p}^F_j + f_{j,i_2}(\mathbf{m}_i) = \hat{c}_{\lambda k^I+j,i_2},$$

$$\sum_{l=1}^{\lambda} a_{l,j} c^l_{k^I+\hat{i},j} = \mathbf{m}_{\hat{i}}\mathbf{p}^F_j + f_{j,\hat{i}}(\mathbf{m}_j) = \hat{c}_{\lambda k^I+j,\hat{i}}.$$

and for $\hat{i} \in [d, r-1]$. Total number of initial parity symbols downloaded by node $\lambda k^I + j$ is λr^I.

b) $j < \lambda r^I$: Let $j = j_1 r^I + j_2$ for $j_2 \in [0 : r^I - 1]$. The node $\lambda k^I + j$ in this case is a changed node and it already has access to the initial parity symbols $\{c^{j_1+1}_{k^I+j_2,i} \mid i \in [0 : r^F - 1]\}$. If $j_1 < c$ then the node j downloads same symbols as shown in equation (5) except for the r^I symbols $\{c^{j_1+1}_{k^I+j_2,i} \mid i \in [0 : r^I - 1]\}$ that it already has access to. The recovery of $\hat{c}_{\lambda k^I+j,i} \mid j \in [0 : r^I - 1]$ follows in the same lines as shown in equation (6). The number of symbols downloaded in this case is equal to $r^I(\lambda - 1)$.

For the case when $j_1 = c$ it follows that $j_2 \in [0 : d-1]$, $\lambda k^I + j$ node downloads the symbols shown in equation (7) except for symbols $\{c^{j_1}_{k^I+j_2,i} \mid i \in [cr^I : cr^I + d - 1]\}$. Total symbols downloaded in this case is equal to $\lambda r^I - d$.

In the phase two, node $\lambda k^I + j$ downloads the following symbols.

- $j \in [0 : r^I - 1]$: In this case the node $\lambda k^I + j$ downloads the final parity symbols $\{\hat{c}_{\lambda k^I+j,i} \mid i \in [r^I : r^F - 1]\}$ where $\hat{c}_{\lambda k^I+j,i}$ is downloaded from node $\lambda k^I + i$ as it has been recovered by it in the Phase 1. Number of symbols downloaded by this node is $(r^F - r^I)$.
- $j \in [r^I : r^F - 1]$: The node $\lambda k^I + j$ already has access to $\hat{c}_{\lambda k^I+j,j}$ from Phase 1. It downloads the symbols $\{\hat{c}_{\lambda k^I+j,i} \mid i \in [r^I : r^F - 1] \setminus \{j\}\}$. Number of symbols downloaded in this case is $(r^F - r^I - 1)$.

The total number of symbols downloaded in Phase 2 is $r^I(r^F - r^I) + (r^F - r^I)(r^F - r^I - 1) = (r^F - 1)(r^F - r^I)$.

IV. Bandwidth calculation

We will do the bandwidth calculation for the two cases (a) $r^F \geq \lambda r^I$ and (b) $r^F < \lambda r^I$ when there can be retired nodes.

1) $r^F \geq \lambda r^I$: There are no retired nodes in this case. The total bandwidth incurred in phase 1 is $(r^F - r^I)\lambda n^I$.
2) $r^F < \lambda r^I$: There are no retired nodes in this case. The total bandwidth incurred in phase 1 is equal to $(r^F - r^I)\lambda k^I + r^F(\lambda - 1)r^I + (r^I - d)d$.

By adding the phase 2 bandwidth and normalizing with the sub-packetization level $\alpha = r^F$, the total normalized bandwidth corresponding to the cooperative scheme, corresponding to the MPCC, is given by:

$$\gamma_{\text{mpcc}} = \begin{cases} (1 - \frac{r^I}{r^F})(n^F + \lambda r^I - 1) & \text{for } r^F \geq \lambda r^I, \\ (1 - \frac{r^I}{r^F})(n^F - 1) + (\lambda - 1)r^I + \frac{d(r^I - d)}{r^F} & \text{for } r^F < \lambda r^I. \end{cases}$$

For $r^I < r^F < k^I$, the savings $\gamma_{\text{MR}} - \gamma_{\text{mpcc}}$ of this scheme in comparison to γ_{MR} is given by:

$$\begin{cases} \frac{r^I}{r^F}(\lambda r^I + r^F - 1) - 1 & \text{for } r^F \geq \lambda r^I, \\ \frac{1}{r^F}((cr^I - 1)r^I + d^2) + r^I - 1 & \text{otherwise.} \end{cases}$$

Clearly, $\gamma_{\text{MR}} - \gamma_{\text{mpcc}}$ is always positive and hence in this case there are always savings. For the other case where $\max(k^I, r^I) \leq r^F$, it is possible that γ_{MR} is smaller in some cases, especially when k^I is small. The bandwidth difference for this case is given by:

$$\gamma_{\text{MR}} - \gamma_{\text{mpcc}} = \begin{cases} \frac{1}{r^F}(\lambda r^I(k^I + r^I - r^F) + r^I(r^F - 1)) & r^F \geq \lambda r^I \\ \frac{1}{r^F}(\lambda(k^I r^I - r^F) + r^F + (r^F - 1)r^I - d(r^I - d)) & \text{otherwise.} \end{cases}$$

V. Conclusion

The cooperative conversion framework shows promise with respect to bandwidth cost savings in comparison to the centralized framework. The fundamental limits of bandwidth cost incurred in the cooperative framework are yet to be characterized and remains an open problem. It also remains open to see if cooperative schemes can be constructed over smaller finite fields.

References

[1] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, and J. Schindler, "An analysis of latent sector errors in disk drives," in *Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 2007, pp. 289–300.

[2] E. Pinheiro, W.-D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population," 2007.

[3] B. Schroeder, S. Damouras, and P. Gill, "Understanding latent sector errors and how to protect against them," *ACM Transactions on storage (TOS)*, vol. 6, no. 3, pp. 1–23, 2010.

[4] A. Ma, R. Traylor, F. Douglis, M. Chamness, G. Lu, D. Sawyer, S. Chandra, and W. Hsu, "Raidshield: characterizing, monitoring, and proactively protecting against disk failures," *ACM Transactions on Storage (TOS)*, vol. 11, no. 4, pp. 1–28, 2015.

[5] 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. USENIX Conference on File and Storage Technologies, FAST*, 2019, pp. 345–358.
[6] S. Kadekodi, F. Maturana, S. J. Subramanya, J. Yang, K. V. Rashmi, and G. R. Ganger, "PACEMAKER: Avoiding HeART attacks in storage clusters with disk-adaptive redundancy," in *Proc. USENIX Symposium on Operating Systems Design and Implementation, OSDI*, 2020, pp. 369–385.
[7] 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, 2022.
[8] ——, "Bandwidth Cost of Code Conversions in Distributed Storage: Fundamental Limits and Optimal Constructions," *IEEE Trans. Inf. Theory*, vol. 69, no. 8, pp. 4993–5008, 2023.
[9] ——, "Bandwidth Cost of Code Conversions in the Split Regime," in *Proc. IEEE International Symposium on Information Theory, ISIT*, 2022, pp. 3262–3267.
[10] F. Maturana, V. S. C. Mukka, and K. V. Rashmi, "Access-optimal Linear MDS Convertible Codes for All Parameters," in *Proc. IEEE International Symposium on Information Theory, ISIT*, 2020, pp. 577–582.
[11] S. Chopra, F. Maturana, and K. V. Rashmi, "On Low Field Size Constructions of Access-Optimal Convertible Codes," in *Proc. IEEE International Symposium on Information Theory, ISIT*, 2024, pp. 1456–1461.
[12] S. Ge, H. Cai, and X. Tang, "MDS generalized convertible code," *arXiv preprint arXiv:2407.14304*, 2024.
[13] M. N. Krishnan, M. Vajha, V. Ramkumar, G. Y. Sai, and X. Kong, "On linear field size access-optimal mds convertible codes," in *2025 IEEE International Symposium on Information Theory (ISIT)*, 2025, pp. 1–6.
[14] H. Shi, W. Fang, and Y. Gao, "Bounds and optimal constructions of generalized merge-convertible codes for code conversion into lrcs," *CoRR*, vol. abs/2504.09580, 2025.
[15] J. Zhang and K. V. Rashmi, "Secure convertible codes for passive eavesdroppers," in *IEEE International Symposium on Information Theory, ISIT 2025, Ann Arbor, MI, USA, June 22-27, 2025*. IEEE, 2025, pp. 1–6.
[16] L. Wang and S. Hu, "Lower bounds on conversion bandwidth for MDS convertible codes in split regime," *CoRR*, vol. abs/2511.00953, 2025.
[17] S. Singhvi, S. Chopra, and K. V. Rashmi, "Tight lower bounds on the bandwidth cost of MDS convertible codes in the split regime," *CoRR*, vol. abs/2511.12279, 2025.
[18] A. G. Dimakis, 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, 2010.
[19] 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, 2011.
[20] V. Ramkumar, S. B. Balaji, B. Sasidharan, M. Vajha, M. N. Krishnan, and P. V. Kumar, "Codes for distributed storage," *Foundations and Trends® in Communications and Information Theory*, vol. 19, no. 4, pp. 547–813, 2022.
[21] K. W. Shum and Y. Hu, "Cooperative regenerating codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 11, pp. 7229–7258, 2013.
[22] K. V. Rashmi, N. B. Shah, and K. Ramchandran, "A piggybacking design framework for read-and download-efficient distributed storage codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 9, pp. 5802–5820, 2017.

APPENDIX A
MDS CODE PROOF

For the initial code, the piggybacked sub-symbols appear in columns $j \in [r^I : r^F - 1]$ (see Table I). In the presence of any r^I erasures, the sub-symbols in the initial r^I columns can be recovered first by the MDS property of the initial base code $\mathcal{C}^{I'}$. Since all the piggybacked sub-symbols are functions of message sub-symbols in the initial r^I columns, their effect can be removed to recover the rest of the message sub-symbols in the last $r^F - r^I$ columns.

Similarly, for the final code, the piggybacked sub-symbols appear in columns $j \in [0 : r^I - 1]$ (see Table II). MDS property follows as we can decode the message symbols from any r^F erasures by first decoding $\lambda k^I(r^F - r^I)$ sub-symbols $\mathbf{m}_j$ for $j \in [r^I : r^F - 1]$ using the fact that the base code of the final code $\mathcal{C}^{F'}$ is an MDS code. The effect of piggybacked sub-symbols can then be removed to recover the rest of the message sub-symbols $\mathbf{m}_j$ for $j \in [0 : r^I - 1]$.