Fifty-Fifth Annual Allerton Conference
Allerton House, UIUC, Illinois, USA
October 3-6, 2017

Limitations of Piggybacking Codes with Low Substriping*

Reyna Hulett[1] and Mary Wootters[2]

***Abstract*— The *piggybacking* framework for designing erasure codes for distributed storage has empirically proven to be very useful, and has been used to design codes with desirable properties, such as low repair bandwidth and complexity. However, the theoretical properties of this framework remain largely unexplored. We address this by adapting a general characterization of repair schemes (previously used for Reed-Solomon codes) to analyze piggybacking codes with low substriping. With this characterization, we establish a separation between piggybacking and general erasure codes, and several impossibility results for subcategories of piggybacking codes; for certain parameters, we also present explicit, optimal constructions of piggybacking codes.**

I. Introduction

The modern world is practically overwhelmed with data, much of which is kept in large-scale distributed storage systems. These systems store large files across a number of servers, or *nodes*. Due to the scale of such systems, node failure is an everyday occurrence, and the system must be robust to such failures. One way to achieve robustness is by replicating the data. However, this clearly has a high storage overhead; erasure coding can achieve the same reliability as replication with far less overhead.

The rapidly developing field of coding for distributed storage seeks to design erasure codes with desirable properties in this setting. Two important properties include an optimal reliability-redundancy trade-off, and bandwidth-efficient repair of failed nodes. The work of [2] introduced a new *piggybacking* design framework which modifies a base code to improve its repair properties. This framework has been employed several times to design new codes [2]–[5], including one code that has been implemented in the Hadoop Distributed File System [2].

Although the piggybacking framework has clearly been productive in practice, there has not been much theoretical analysis of its possibilities and limitations. That is the subject of this paper.

A. Our Contributions

We build on a framework introduced by [6] for characterizing and analyzing repair schemes in the context of scalar MDS codes, and obtain a characterization of piggybacking code repair schemes. This allows us to prove impossibility results for piggybacking schemes, and to design schemes with optimal repair bandwidth for certain parameters. Specifically, our contributions are the following.

1) **Extension of the framework of [6].** We adapt the characterization of repair schemes by [6], originally introduced for Reed-Solomon codes, to our setting. More precisely, their scheme works for scalar MDS codes over finite fields, while piggybacking codes are not scalar. We modify their approach to obtain a characterization of linear repair schemes for MDS array codes. We specialize this to piggybacking codes for our main results, but the general framework may be of broader interest.
2) **Separation between piggybacking and general erasure codes.** Using this framework, we demonstrate that for certain parameter regimes, piggybacking cannot achieve the optimal repair bandwidth achievable by general erasure codes. Thus piggybacking codes are strictly less powerful than general erasure codes.
3) **Other bounds.** We additionally utilize this framework to give some limited lower bounds for piggybacking in other settings, as well as upper bounds and explicit code constructions for some specific parameters. Some of these bounds suggest approaches to using the piggybacking design framework which may improve the attainable repair bandwidth, compared to existing constructions.

II. Setup and Preliminaries

We begin with some notation. In general, we will use (parenthetical) superscripts to denote different matrices and subscripts to index within a matrix. For indexing into a matrix $M^{(\ell)}$, the entry in row i, column j will be denoted $m^{(\ell)}_{i,j}$, the i^{th} row will be denoted $m^{(\ell)}_{i,\bullet}$, and the j^{th} column will be denoted $m^{(\ell)}_{\bullet,j}$. Vectors generated by indexing into a matrix will be rows or columns corresponding to their orientation in the matrix (e.g., $m^{(\ell)}_{i,\bullet}$ is a row vector but $m^{(\ell)}_{\bullet,j}$ is a column vector). Other vectors will be considered column vectors by default. They will be typeset in bold as $\boldsymbol{v^{(\ell)}} = [v_0^{(\ell)}\ v_1^{(\ell)}\ \cdots]^T$ or $\boldsymbol{v} = [\text{—}\boldsymbol{v_0}^T\text{—}\ \text{—}\boldsymbol{v_1}^T\text{—}\ \cdots]^T$ if we wish to refer to vector "chunks" within $\boldsymbol{v}$.

A. Erasure Coding and the Exact Repair Problem

In this paper, we restrict our focus to linear, maximum distance separable codes with linear repair schemes. We first briefly recall some definitions. A *code* $\mathcal{C}$ over an alphabet A is a subset of A^n, so each *codeword* consists of n *symbols* of

*Many details have been omitted from this extended abstract, and can be found in the full version of the paper [1]

[1]RH is with the Computer Science Department, Stanford University. rmhulett@stanford.edu. RH's research supported in part by an NSF Graduate Research Fellowship under grant DGE-1656518.

[2]MW is with the Computer Science and Electrical Engineering Departments, Stanford University. marykw@stanford.edu. RH and MW's research supported in part by NSF grant CCF-1657049.

978-1-5386-3266-6/17/$31.00 ©2017 IEEE

A. If the code has size $|\mathcal{C}| = |A|^k$, we say that the *dimension* of $\mathcal{C}$ is k. We say that such a code has the Maximum Distance Separable property (MDS property) if any k symbols of a codeword $\boldsymbol{c} \in \mathcal{C}$ can determine $\boldsymbol{c}$.

If the alphabet A is a field, $A = \mathbb{F}_q$, and if $\mathcal{C} \subset \mathbb{F}_q^n$ is a linear subspace of $\mathbb{F}_q^n$, then we say $\mathcal{C}$ is *linear.* A linear code $\mathcal{C}$ can always be written as the image of a *generator matrix* $F \in \mathbb{F}_q^{n \times k}$; given a *message* $\boldsymbol{a} \in \mathbb{F}_q^k$, the corresponding *codeword* is $F\boldsymbol{a}$. If $\mathcal{C}$ additionally has the MDS property—equivalently, if its generator matrix F has the property that any k rows are linearly independent—we say $\mathcal{C}$ is an MDS code. We say that a code (along with an encoding map from messages to codewords) is *systematic* if the k symbols of the message appear as symbols of the codeword; it is not hard to see that any linear code has a systematic encoding map. We study *array codes,* where the alphabet A is in fact a vector space $A = \mathbb{F}_q^t$. These codes are not linear (indeed, it does not immediately make sense for a code to be linear over $\mathbb{F}_q^t$), but we study codes that are $\mathbb{F}_q$-linear.

Definition 1. An *array code with t substripes* over an alphabet $\mathbb{F}_q$ is a code $\mathcal{C} \subset (\mathbb{F}_q^t)^n$ over $\mathbb{F}_q^t$. We say that $\mathcal{C}$ has *linear substripes* if $\mathcal{C}$ is closed under $\mathbb{F}_q$-linear operations. That is, for any $\boldsymbol{c}, \boldsymbol{c'} \in \mathcal{C}$ and for any $\lambda \in \mathbb{F}_q$, $\boldsymbol{c} + \lambda \boldsymbol{c'} \in \mathcal{C}$. If $\mathcal{C}$ has the MDS property, we say that it is an *MDS array code.*

We will often think of codewords of an array code as matrices $C \in \mathbb{F}_q^{n \times t}$, rather than vectors $\boldsymbol{c} \in (\mathbb{F}_q^t)^n$, and we will write $\mathcal{C} \subset \mathbb{F}_q^{n \times t}$.

In coding for distributed storage, the message $\boldsymbol{a}$ corresponds to a file to be stored, and the corresponding codeword $\boldsymbol{c} \in \mathcal{C}$ captures how the data should be stored on the n nodes: node i holds the symbol c_i. In this setting, we would always like to tolerate as many node failures as possible, which means that we demand that the code $\mathcal{C}$ have the MDS property. Moreover, there are certain operations we would like to be efficient. First, we would like to be able to recover the original message (the stored file) efficiently. This can always be done directly if the code is systematic. Second, while we would like to be able to recover from $n - k$ failures in the worst case, a much more common scenario is a single failure [7]. Thus, we would like to be able to repair a single failed node as efficiently as possible. In this work, the measure of efficiency we consider is the *repair bandwidth,* which measures how much data must be downloaded to repair a single failure.

Formally, let $\mathcal{C}$ be an MDS array code over $\mathbb{F}_q$ with t substripes. If node i^* fails, then a *repair scheme* to repair i^* using a *repair set* $S \subset \{0, \dots, n-1\} \setminus \{i^*\}$ is a collection of functions[1] $g_i : \mathbb{F}_q^t \to \mathbb{F}_q^{b_i}$ so that for all $\boldsymbol{c} \in \mathcal{C}$, c_{i^*} can be determined from $\{g_i(c_i) \mid i \in S\}$. If $\mathcal{C}$ is a linear MDS array code, and if the functions g_i and the method of determining c_{i^*} is linear, we say that the repair scheme is *linear.*

The above defines a repair scheme for a particular node i^* and a particular repair set S. A (linear) repair scheme with *locality* d for an MDS array code $\mathcal{C}$ over $\mathbb{F}_q$ consists of (linear) repair schemes with repair sets S of size d for every possible failed node i^*. There are two important regimes. In the "any d" regime, there must be a valid repair scheme for any repair set S of size d. On the other hand, in the "some d" regime, we require only one valid repair set of size d per possible failed node.

The *bandwidth* of a repair scheme for an MDS array code $\mathcal{C}$ over $\mathbb{F}_q$ is the maximum number of symbols of $\mathbb{F}_q$ needed to repair any node i^*. In the language above, it is the maximum, over all i^* and all repair sets S in the scheme, of $\sum_{i \in S} b_i$. The *exact repair problem* is the problem of minimizing the repair bandwidth. There have been several solutions proposed in the literature. In this work, we focus on the *piggybacking framework,* which we discuss below.

B. Piggybacking

In this paper, we study the piggybacking framework introduced by [2], (with a few assumptions, discussed below). A *piggybacking code* $\mathcal{C}$ over $\mathbb{F}_q$ with t substripes is constructed from a "base code" $\mathcal{C}_0 \subset \mathbb{F}_q^n$ and $\binom{t}{2}$ "piggybacking functions" $p^{(i,j)} : \mathbb{F}_q^k \to \mathbb{F}_q^n$. For this work, we assume that the base code $\mathcal{C}_0$ is a (scalar) MDS code over $\mathbb{F}_q$; in particular, it is linear, with a generator matrix $F \in \mathbb{F}_q^{n \times k}$. We also assume that the piggybacking functions $p^{(i,j)}$ are linear; in particular, they can be represented by matrices $P^{(i,j)} \in \mathbb{F}_q^{n \times k}$. With these assumptions, we define a piggybacking code (with a scalar MDS base code) over $\mathbb{F}_q$ as follows.

Definition 2. Let $F \in \mathbb{F}_q^{n \times k}$ be the generator matrix of an MDS code $\mathcal{C}_0$, and fix a collection of piggybacking matrices $\{P^{(i,j)} \mid i \in [0, t-2], j \in [i+1, t-1]\} \subset \mathbb{F}_q^{n \times k}$. Consider the MDS array code $\mathcal{C}$ over $\mathbb{F}_q$ with t linear substripes, defined as follows. Given a message $\boldsymbol{a} \in \mathbb{F}_q^{kt}$ given by $\boldsymbol{a} = [\text{—}\boldsymbol{a_0}^T\text{—} \ \cdots \ \text{—}\boldsymbol{a_{t-1}}^T\text{—}]^T$ (where each $\boldsymbol{a_i} \in \mathbb{F}_q^k$), we form a codeword $C \in \mathbb{F}_q^{n \times t}$ so that the i^{th} substripe (column) is

$$c_{\bullet,i} = P^{(0,i)}\boldsymbol{a_0} + \cdots + P^{(i-1,i)}\boldsymbol{a_{i-1}} + F\boldsymbol{a_i}.$$

We say that $\mathcal{C}$ is a (n, k) *piggybacking code with a scalar MDS base code* (henceforth a *piggybacking code*) with t substripes over $\mathbb{F}_q$.

We illustrate a piggybacking code formed from F and $\{P^{(i,j)}\}$ in Fig. 1.

As with all MDS array codes, we will represent codewords as matrices in $\mathbb{F}_q^{n \times t}$, so they will have the same layout as the table in Fig. 1: rows correspond to nodes and columns to substripes. As noted in [2], piggybacking codes using an MDS base code remain MDS, but may have improved repair properties; in particular, they may have reduced repair bandwidth.

In addition to general piggybacking codes, we will also consider a subcategory of codes which only piggyback in the last substripe of each node, inspired by the approach of [3]. We dub these *linebacking codes*.

[1] In this work, we will only consider repair bandwidth, rather than disk access, so we allow the nodes to do arbitrary local computation.

	substripe 0	substripe 1	$\cdots$	substripe t-1
node 0				
$\vdots$	$F\boldsymbol{a_0}$	$P^{(0,1)}\boldsymbol{a_0} + F\boldsymbol{a_1}$	$\cdots$	$P^{(0,t-1)}\boldsymbol{a_0} + \cdots + P^{(t-2,t-1)}\boldsymbol{a_{t-2}} + F\boldsymbol{a_{t-1}}$
node n-1				

Fig. 1. A piggybacking codeword formed from generator matrix F, piggybacking matrices $\{P^{(i,j)}\}$, and message $\boldsymbol{a}$ stored on a set of n nodes.

Definition 3. An (n, k) *linebacking code with a scalar MDS base code* (henceforth a *linebacking code*) $\mathcal{C}$ over the finite field $\mathbb{F}_q$ with t substripes is a (n, k) piggybacking code, with the additional property that all piggybacking matrices $P^{(i,j)}$ such that $j \neq t-1$ are zero. Thus we drop the index j indicating which substripe the piggyback is added to and denote $P^{(i,t-1)}$ by $P^{(i)}$.

III. Related Work and Our Results

The piggybacking framework for designing error correcting codes for distributed storage was introduced by [2]. It is as described in Definition 2, except that we have made the following assumptions. First, that the piggybacking functions are linear—in general this is not required—and second, that the base code is a scalar MDS code—in general, the base code may itself be an MDS array code. (However, we note that all piggybacking codes in the literature do use linear piggybacking functions [2]–[5]. Almost all use scalar MDS base codes as well, except [3] and one of four constructions in [2], which are specifically designed to improve the repair properties of parity nodes for existing array codes.) Furthermore, in [2], an invertible linear tranformation may be applied to the data stored on each node in order to reduce the data-read. However, since in this work we are only concerned with repair bandwidth, this does not matter for us and we omit it from Definition 2.

The piggybacking design framework has been used to produce codes with low data-read and bandwidth for repairing individual failed nodes. Initially, [2] introduced and used the framework to design explicit codes with the lowest data-read and bandwidth among known solutions for a few specific settings, including (high-rate) MDS codes with low substriping, the domain of interest in this paper. Extending their ideas, [3] showed how to modify codes with optimal repair bandwidth for systematic nodes to use piggybacking to obtain asymptotically optimal bandwidth for parity nodes as well. Interestingly, [3] obtained these results for linebacking codes (Definition 3), which is more restricted than general piggybacking. The piggybacking framework was also employed in [4] to design codes with low repair complexity, and in [5] as part of a compound design using both piggybacking and simple parity checks.

However, little is understood about the theoretical possibilities and limitations of codes designed using the piggybacking framework. Nor is much known about how to choose piggybacking functions to achieve desirable repair properties. Although [3] takes a more principled approach to choosing the piggybacking functions, some of the choices—including piggybacking only in the last substripe and always using all the systematic nodes in repairing a failed parity node—do not have a rigorous theoretical backing.

Here we explore the theoretical limitations on achievable repair bandwidth for piggybacking codes with scalar MDS base codes and with a small number of substripes $t \leq n-k$. As in Definition 1, we do not allow for *symbol extension;* that is, we treat the elements of $\mathbb{F}_q$ as indivisible and measure bandwidth in units of symbols of $\mathbb{F}_q$. We study both the "any d" and "some d" regimes, though the latter regime is less restrictive, and the achievable repair bandwidth is less well characterized. While both regimes have been studied in the literature [8], the piggybacking design framework has primarily been employed in the latter, less restrictive regime [2]–[5].

A. Known Lower Bounds

In any setting, for MDS codes, we have the *cut-set bound* on the repair bandwidth, which states that we must download $b \geq \frac{td}{d-k+1}$ symbols [10]. Since this is decreasing in d, we can also set d to the maximum/optimal value $n-1$ to get the bound $b \geq t\frac{n-1}{n-k}$. However, if $t < n-k$—the setting we consider here—this is not achievable, since it would require downloading less than a full symbol from each node. Most existing MSR codes seek to achieve this bound on bandwidth and thus focus on the setting where $t \geq n-k$. While some codes, such as that of [11], can also operate with substriping $t < n-k$, they often do not achieve the lowest possible bandwidth. In particular, codes with $d = n-1$, i.e., all remaining nodes participating in repair, must have bandwidth at least $n-1$. But for $t < n-k$, we can derive and in some cases achieve a lower bound $< n-1$ as shown below.

Starting from the cut-set bound $b \geq \frac{td}{d-k+1}$, we can derive a different lower bound applicable in the setting considered here. Since any repair scheme must download at least one full symbol from every participating node, we can say $b \geq d$ which gives

$$b \geq \frac{td}{d-k+1} \geq \frac{tb}{b-k+1},$$

which implies

$$b \geq k+t-1,$$

matching the trivial lower bound for any MDS code [6]. We will call $b = k+t-1$ *perfect bandwidth*.[2]

[2]Throughout we will assume $2 \leq t \leq n-k$ and $2 \leq k$. Otherwise, if $t = 1$, there is no piggybacking and any MDS code achieves perfect bandwidth; if $t > n-k$, achieving perfect bandwidth is impossible; and if $k = 1$, the straightforward lower bound on bandwidth $k+t-1$ and the trivially achievable bandwidth for MDS codes kt are equal.

TABLE I

EXISTENCE OF PERFECT BANDWIDTH MDS ARRAY CODES WHEN $t \leq n-k$ IN THE "ANY d" REGIME.

	General	Piggybacking	Linebacking
$k \geq 3$	$t > k-3$: exist for $q \geq n-k+t$ $t \leq k-3$: do not exist [8]	do not exist (Thm. 6)	do not exist (Thm. 6)
$k = 2$	exist for $q \geq n-k+t$ [8]	exist for some q (Thm. 7)	exist for some q (Thm. 7)

TABLE II

EXISTENCE OF PERFECT BANDWIDTH MDS ARRAY CODES WHEN $t \leq n-k$ IN THE "SOME d" REGIME.

	General	Piggybacking	Linebacking
$k \geq 3$	$t \geq k$: exist for $k \leq \max\{\frac{n}{2}, 3\}$, $q \geq 2(n-k)$ [9]	$t = n-k$: do not exist (Thm. 6) $t = 2$: $(6,3)$ construction for $q = 7$ (Thm. 8)	$t = n-k$: do not exist (Thm. 6) $t > \frac{n-k+1}{2}$: do not exist (Thm. 11) $t = 2$: $(6,3)$ construction for $q = 7$ (Thm. 8)
$k = 2$	exist for $q \geq 2(n-k)$ [9]	exist for some q (Thm. 7) $t = 2$: construction for $q \geq k+1$ (Thm. 9)	exist for some q (Thm. 7) $t = 2$: construction for $q \geq k+1$ (Thm. 9)

Note that this bound cannot be tight for large $t > n-k$. However, for $t \leq n-k$ in the "any d" regime, [8] demonstrated that perfect bandwidth is achievable for $t > k-3$, but cannot be achieved by a linear code without symbol extension for $t \leq k-3$. In the less restrictive "some d" regime, [9] showed that the cut-set bound (and thus perfect bandwidth) is achievable provided $k \leq \max\{\frac{n}{2}, 3\}$ and $d \geq 2k-1$ which in our setting translates to substriping $t \geq k$, with field size at most $2(n-k)$. Although the cut-set bound has been shown to be achievable for large t, q and general n, k [12], [13], to our knowledge the question of achieving perfect bandwidth when $t \leq n-k$ in general remains open.

B. Our Results

We study the ability of piggybacking codes (with scalar MDS base codes) to achieve perfect bandwidth when $t \leq n-k$, using linear repair schemes. By adapting the framework of [6], we are able to completely characterize when perfect bandwidth is possible in the "any d" regime, and we are able to give partial results in the "some d" regime. These results are summarized in Tables I and II.

Our results imply piggybacking codes are strictly weaker than general erasure codes when $k \geq 3$ in both regimes—for "any d" there is separation with $t > k-3$ and for "some d" with $t = n-k$. While the "any d" regime is well characterized, the partial "some d" results include an explicit construction for $k = 2, t = 2$ with small field size, an example construction for $k = 3$ showing that piggybacking codes are more powerful in that regime, and a few impossibility results for the restricted case of linebacking codes. We do not know whether piggybacking is strictly more powerful than linebacking, but these impossibility results present a possible method of establishing this separation.

IV. CHARACTERIZATION OF REPAIR SCHEMES

The work of [6] provides a characterization of linear repair schemes for scalar MDS codes. Their framework relies on the fact that a scalar MDS code $\mathcal{C}$ is linear over its alphabet. In our case, piggybacking codes—and more generally MDS array codes with linear substripes—are linear over $\mathbb{F}_q$, but not over $\mathbb{F}_q^t$. (Indeed, linearity over $\mathbb{F}_q^t$ does not immediately make sense, as $\mathbb{F}_q^t$ does not have a natural notion of multiplication.) However, the approach of [6] still makes sense in this context. The main reason linearity was important to the approach of [6] was because their characterization involved the *dual code,* $\mathcal{C}^\perp$. We may introduce a similar notion for MDS array codes.

Definition 4. Let $\mathcal{C} \subset \mathbb{F}_q^{n \times t}$ be an MDS array code with t linear substripes over $\mathbb{F}_q$. The *dual code* of $\mathcal{C}$ is $\mathcal{C}^\perp := \{X \in \mathbb{F}_q^{n \times t} \mid \langle X, C \rangle = 0 \ \forall C \in \mathcal{C}\}$, where $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product.

Theorem 1. *Let $\mathcal{C} \subset \mathbb{F}_q^{n \times t}$ be a (n,k) MDS array code with t linear substripes over $\mathbb{F}_q$. For a fixed node i^* and set of nodes $S \not\ni i^*$, the following are equivalent.*

1) *There is a linear repair scheme for node i^* from S with bandwidth b.*
2) *There exists a set of t dual codewords, the* repair matrices*, $\{W^{(0)}, W^{(1)}, \ldots, W^{(t-1)}\} \subset \mathcal{C}^\perp$ such that the only non-zero rows of each $W^{(j)}$ are i^* and S, and furthermore that*

$$\dim(\{w^{(j)}_{i^*,\bullet} \mid j \in [0, t-1]\}) = t,$$

and

$$\sum_{i \neq i^*} \dim(\{w^{(j)}_{i,\bullet} \mid j \in [0, t-1]\}) \leq b.$$

See Fig. 2 for an illustration of a set of repair matrices. The proof of Theorem 1 follows very similar to the approach in [6]. The full proof (and those of all following results) may be found in the full version of this paper, but we sketch one direction in Fig. 3, showing how a set of repair matrices yields a repair scheme.

Henceforth we will refer interchangeably to a linear repair scheme for i^* from a set S, and a set of t repair matrices for i^*, S as defined above.

A. Piggybacking Code Repair Schemes

Now that we have characterized repair schemes as sets of dual codewords, we can analyze the repair schemes of piggybacking codes, and those that achieve perfect bandwidth

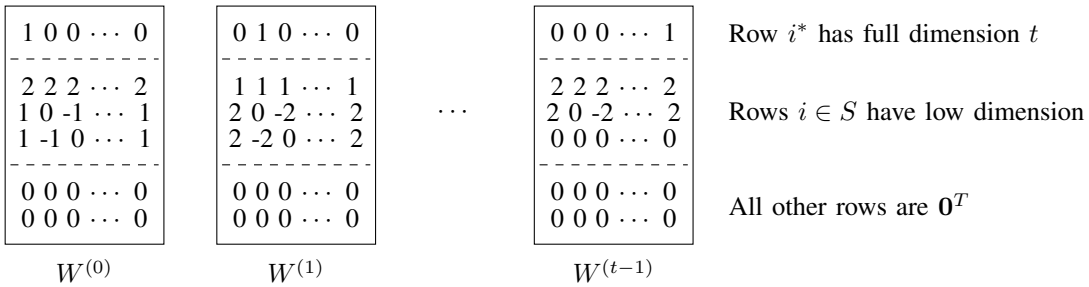

Fig. 2. An example illustrating the structure of a linear repair scheme with $q = 3$.

Linear repair scheme, given repair matrices. Suppose that $\{W^{(0)}, W^{(1)}, \dots W^{(t-1)}\}$ is a set of repair matrices for a node i^* with repair set S, as in Theorem 1. Let $C \in \mathcal{C}$ be a codeword of the MDS array code. Then we can define a linear repair scheme as follows.

1) For every node $i \neq i^*$, let $Q_i \subset \mathbb{F}_q^t$ be a basis of span$(\{w_{i,\bullet}^{(j)} \mid j \in [0, t-1]\})$. We say that Q_i is the *query set* for node i. Observe that $Q_i = \emptyset \ \forall i \notin S$ so only nodes in the repair set will be queried.
2) For every query vector $\boldsymbol{q} \in Q_i$, node i sends $\boldsymbol{q} \cdot c_{i,\bullet}$ to the replacement node. Since $\sum_{i \neq i^*} |Q_i| \leq b$, at most b symbols of $\mathbb{F}_q$ are downloaded.
3) The replacement node can now recover $w_{i^*,\bullet}^{(j)} \cdot c_{i^*,\bullet} = \langle W^{(j)}, C \rangle - \sum_{i \neq i^*} w_{i,\bullet}^{(j)} \cdot c_{i,\bullet} i$ for all j, since $W^{(j)} \in \mathcal{C}^\perp$ implies $\langle W^{(j)}, C \rangle = 0$ and $w_{i,\bullet}^{(j)} \cdot c_{i,\bullet}$ for $i \neq i^*$ can be recovered from the responses to the query set Q_i. Since $\dim(\{w_{i^*,\bullet}^{(j)} \mid j \in [0, t-1]\}) = t$, this gives t linearly independent equations, and we can solve for $c_{i^*,\bullet}$, thus repairing the failed node.

Fig. 3. Turning a set of repair matrices into a linear repair scheme (proves one direction of Theorem 1).

in particular. In the remainder of Section IV, we present intermediate results using this characterization which will lead to the main theorems of Sections V and VI. Proofs of these results, as well as additional lemmas establishing notions of equivalence and a standard form for repair matrices, may be found in the full version of this paper. The next lemma specializes the definition of a dual code to piggybacking codes.

Lemma 2. *Let $\mathcal{C}$ be an (n, k) piggybacking code over $\mathbb{F}_q$ with t substripes, base code $\mathcal{C}_0$ with generator matrix F, and piggybacking matrices $\{P^{(i,j)} \mid i \in [0, t-2], j \in [i+1, t-1]\}$. A matrix $X \in \mathbb{F}_q^{n \times t}$ is in $\mathcal{C}^\perp$ if and only if*

$$x_{\bullet,i}^T F + x_{\bullet,i+1}^T P^{(i,i+1)} + \cdots + x_{\bullet,t-1}^T P^{(i,t-1)} = \mathbf{0}^T \ \forall i.$$

We can strengthen this characterization of repair schemes by adding the requirement of perfect bandwidth, that is, that $b = k + t - 1$.

Observation 3. Any perfect bandwidth (n, k) MDS code with t linear substripes must have locality $d = k+t-1$, and download a single symbol from each node of the repair set. This follows from the cut-set bound [10].

In the notation of Theorem 1, the above implies $|S| = k + t - 1$ and $\dim(\{w_{i,\bullet}^{(j)} \mid j \in [0, t-1]\}) = 1 \ \forall i \in S$, for any perfect bandwidth repair scheme. Lemma 2 and Observation 3 together give a fairly restricted form for perfect bandwidth piggybacking code repair schemes, which leads to the remaining results of this paper, such as the following lemma.

Lemma 4. *Let $\mathcal{C}$ be a perfect bandwidth (n, k) piggybacking code over $\mathbb{F}_q$ with $t = 2$ substripes. Then $q \geq k + 1$.*

Proof idea. From Observation 3, we know any repair scheme for $\mathcal{C}$ uses repair sets S of size $d = k+t-1 = k+1$, and each row of S has dimension 1, i.e., the rows of one repair matrix are scalar multiples of those of the other. We use properties of the dual code $\mathcal{C}^\perp$ from Lemma 2 to show that these $k+1$ scalars in $\mathbb{F}_q$ must all be distinct, and thus $q \geq k + 1$. □

V. "Any d" Regime

As noted in Observation 3, perfect bandwidth piggybacking codes with low substriping $t \leq n - k$ must have locality $d = k+t-1$. In this section, we consider the regime where, when a node fails, *any* set of d other nodes must be able to repair it with bandwidth b (as opposed to the less strict "some d" regime, which is treated in Section VI). For general linear erasure codes in this regime, [14] showed that for $t \leq k - 3$, the cut-set bound is not achievable when only a single symbol is downloaded from each node of the repair set (and thus perfect bandwidth is not achievable). However, they also constructed a code which does achieve the cut-set bound for any $t > k-3$, at least for repairing the systematic nodes. In this section, we will show that piggybacking codes cannot achieve perfect bandwidth for $k \geq 3$, and thus are strictly weaker than general linear codes in this regime.

Our main impossibility result, Theorem 6 below, is a consequence of the following more general lemma.

Lemma 5. *No (n,k) piggybacking code with $k \geq 3$ and t substripes can have two bandwidth $b = k+t-1$ repair schemes for two different failed nodes with the same set of $k+t$ non-zero rows.*

Proof idea. Consider two such repair schemes (sets of repair matrices), $\{W^{(0)}, \ldots, W^{(t-1)}\}$ which repairs node i^*_W, and $\{V^{(0)}, \ldots, V^{(t-1)}\}$ which repairs node i^*_V. Using the characterization of Section IV, we show that these repair schemes can be modified so that the last columns of $W^{(j)}, V^{(j)}$ are equal for all j, and the next-to-last columns are equal up to addition of a multiple $\kappa^{(j)} \in \mathbb{F}_q$ of the last column. However, using the fact that $k \geq 3$, we also establish that the W's and V's share a non-zero row (in addition to i^*_W, i^*_V) which must have dimension 1 in both by Observation 3. This forces $\kappa^{(j)} = \kappa \ \forall j$, but in turn this makes it impossible for row i^*_W to have full dimension in the W's but dimension 1 in the V's, as would be required for valid, perfect bandwidth repair schemes. This gives the desired contradiction. □

Theorem 6. *No (n,k) piggybacking code with $k \geq 3$ and t substripes can achieve perfect bandwidth in the "any d" regime.*

Proof. Such a code must have two repair schemes meeting the conditions of Lemma 5, which is impossible. □

Therefore no piggybacking code for $k \geq 3$ can achieve perfect bandwidth, even for systematic nodes (in fact, even for only two nodes), and thus piggybacking codes are strictly less powerful than general linear codes in the "any d" regime. However, we are also able to use our characterization to prove an achievability result for $k = 2$, provided the field size is sufficiently large. Moreover, this holds even if we only permit linebacking (as in [3]).

Theorem 7. *Let $\mathcal{C}_0 \subset \mathbb{F}_q^n$ be an MDS code with dimension $k = 2$ and generator matrix $F \in \mathbb{F}_q^{n \times k}$, let $2 \leq t \leq n-k$, and suppose that q is sufficiently large so that $n\binom{n-1}{t+1}\left(1 - \Pi_{i=1}^{t-1}(1 - \frac{1}{q^i})\right) < 1$. Then there exists a linebacking code with base code $\mathcal{C}_0$ and t substripes, which achieves perfect bandwidth ($b = k+t-1 = t+1$).*

Proof idea. The proof of this theorem is non-constructive. We proceed by choosing the set of piggybacking matrices $\{P^{(i)} \mid i \in [0, t-2]\} \subset \mathbb{F}_q^{n \times k}$ uniformly at random. Using the characterization of Section IV, we fix a particular structure for the repair matrices such that given piggybacking matrices $\{P^{(i)}\}$, a failed node i^*, and the repair set S, there is exactly one candidate set of repair matrices which are guaranteed to be dual codewords and to have perfect bandwidth. Furthermore, for uniformly random $\{P^{(i)}\}$, we show that row i^* of the repair matrices is also uniformly random—except for the last entry of each row i^* which is a fixed non-zero in $\mathbb{F}_q$. Thus the probability that row i^* has full rank, or equivalently that we have a valid repair scheme for i^* from S, is $\Pi_{i=1}^{t-1}(1 - \frac{1}{q^i})$. Union bounding over all choices of i^*, S gives the desired result. □

VI. "Some d" Regime

In this section, we consider the regime where, when a node fails, there must exist only some single set of $d = k+t-1$ nodes which can repair it with bandwidth b. As we shall see, this is less strict than the "any d" regime addressed in Section V. It is also a popular regime for instantiating the piggybacking design framework, e.g., in [2]–[5].

Some results immediately transfer over from the "any d" regime to the "some d" regime. For example, since perfect bandwidth linebacking codes for $k = 2$ exist in the "any d" regime, then they exist in the "some d" regime as well. Additionally, when $t = n-k$ and $d = k+t-1 = n-1$, the two regimes coincide, and so all of the results of Section V still hold if $t = n-k$. This implies perfect bandwidth piggybacking codes for $t = n-k$ are impossible in the "some d" regime, and since the constructions of [8], [9] give perfect bandwidth MDS codes for $t = n-k$, this implies piggybacking codes are strictly weaker than general MDS codes in this regime as well.

However, the two regimes are not equivalent; in Theorem 8 below, we note the existence an example piggybacking code achieving perfect bandwidth for $k = 3$, which is impossible in the "any d" regime. Additionally, although piggybacking codes exist for $k = 2$ even in the "any d" regime, we can strengthen Theorem 7 in the "some d" regime, and even give an explicit construction of a perfect bandwidth piggybacking code for $k = 2, t = 2$ in Theorem 9 below.

Finally, while we are so far unable to prove impossibility results for piggybacking codes in general in the "some d" regime, we are able to prove impossibility results for linebacking codes. In Theorem 11 we show that perfect bandwidth linebacking codes do not exist for $k \geq 3$ and $t > \frac{n-k+1}{2}$.

We begin with our positive results. In the full version of the paper, we give an example (the parameters of which are given below in Theorem 8) of a perfect bandwidth piggybacking code for $k = 3$ in the "some d" regime. This establishes a separation between the "some d" and "any d" regimes, since Theorem 6 shows that there is no such scheme in the "any d" regime.

Theorem 8. *There is a piggybacking code with $n = 6, k = 3, t = 2, q = 7$ which achieves perfect bandwidth in the "some d" regime.*

We can also improve Theorem 7 in the "some d" regime. Trivially, since only *some* set of d nodes must be able to repair a given failed node, we need only union bound over choices of i^* (not sets of d nodes), and thus q must only satisfy $n\left(1 - \Pi_{i=1}^{t-1}\left(1 - \frac{1}{q^i}\right)\right) < 1$. However, we can additionally improve on this non-constructive result by giving an explicit construction for optimal piggybacking codes with $k = 2, t = 2$.

Theorem 9. *Let $\mathcal{C}_0 \subset \mathbb{F}_q^n$ be an MDS code with dimension $k = 2$ and generator matrix $F \in \mathbb{F}_q^{n \times k}$, let $t = 2$, and suppose that $n \geq 4$, $q \geq 3$. Then there is an explicit construction of a piggybacking code $\mathcal{C}$ with base code $\mathcal{C}_0$*

and with t substripes, which achieves perfect bandwidth $b = k + t - 1 = 3$, in the "some d" regime.

Proof idea. For $t = 2$, there is only one piggybacking matrix, $P^{(0,1)} = P$. We choose P to be all zero, except for one position $p_{i,0} = 1$. This allows us to design sparse repair matrices where the entries in row i are carefully chosen so that the repair matrices are in $\mathcal{C}^\perp$ and the zero pattern ensures low bandwidth. The design for repairing node $i^* = i$ is slightly more complicated; we set up a similar structure and then argue that one of $q-1$ non-zero choices for a certain entry must yield a valid repair scheme. □

While general impossibility results for piggybacking codes in the "some d" regime remain elusive, we can prove impossibility results for *linebacking* codes in this regime. We note that multiple constructions of piggybacking codes in the literature are in fact linebacking codes—including the design of [3] and any piggybacking codes with $t = 2$ substripes, such as two constructions of [2]—and so such lower bounds provide useful design insights. Our main theorems on linebacking codes follow from a more general lemma.

Lemma 10. *No (n, k) linebacking code with t substripes and $k \geq 3$ can have two bandwidth $b = k+t-1$ repair schemes for two different failed nodes, where each node participates in the other's repair* and *the respective repair sets overlap by at least k nodes.*

Proof idea. We proceed similarly to the proof of Lemma 5, but considering only two of the repair matrices. Let the repair schemes be $\{W^{(0)}, \ldots, W^{(t-1)}\}$ which repairs node i^*_W, and $\{V^{(0)}, \ldots, V^{(t-1)}\}$ which repairs node i^*_V. Using the characterization of Section IV and the restricted structure of linebacking codes, we show that these repair schemes can be modified so that not just the last columns are equal, but in fact $W^{(0)} = V^{(0)}$ and $W^{(1)} = V^{(1)}$. However, this makes it impossible for row i^*_W to have full dimension in the W's but dimension 1 in the V's, as would be required for valid, perfect bandwidth repair schemes. This gives the desired contradiction. □

Theorem 11. *No (n, k) linebacking code with $k \geq 3$ and $t > \frac{n-k+1}{2}$ substripes can achieve perfect bandwidth.*

Proof. Assume to obtain a contradiction that a perfect bandwidth (n, k) linebacking code $\mathcal{C}$ with $t < \frac{n-k+1}{2}$ substripes does exist for some $k \geq 3$. Observe that there must exist a pair of nodes each of which participates in repairing the other. Consider a directed graph where edges go from each node to the nodes it repairs. Each node has $k+t-1$ in-edges so there are $n(k+t-1)$ edges. However, $n(k+t-1) > n(k + \frac{n-k+1}{2} - 1) = \frac{n(n+k-1)}{2} > \binom{n}{2}$. But $\binom{n}{2}$ is the maximum number of edges a directed graph (with no self-loops) can have without having a 2-cycle; thus this graph has a 2-cycle meaning two nodes participate in each others' repair. Furthermore, these same two nodes must have an overlap in their repair sets of size at least k: In addition to repairing each other, they each have $k + t - 2 > \frac{n+k-3}{2}$ helper nodes drawn from the remaining $n-2$ nodes, so the overlap is at least $2(\frac{n+k-2}{2}) - (n-2) = k$. Thus if such a linebacking code existed, it would necessarily have two repair schemes meeting the assumptions of Lemma 10, which is impossible. □

We remark that the constraint on t is tight in the sense that the example code of Theorem 8 is a perfect bandwidth linebacking code with $t = \frac{n-k+1}{2}$. In fact, for $t \leq \frac{n-k+1}{2}$, it is trivial to construct the repair sets (disregarding whether they admit valid repair schemes) to avoid any pair satisfying the assumptions of Lemma 10; for instance, each node can be repaired by the $k+t-1$ nodes immediately following it (mod n). However, most existing piggybacking codes do not choose their repair sets this way. Most, including those of [2], [3] (who use linebacking codes), always use all remaining systematic nodes in the repair. For linebacking codes, this further restricts the parameters for which perfect bandwidth may be achievable.

Theorem 12. *No (n, k) linebacking code with $k \geq 3$, $t > \frac{n-k-1}{\sqrt{k}}$ substripes, and which uses all remaining systematic nodes to repair a failed node can achieve perfect bandwidth.*

Proof. Assume to obtain a contradiction that a perfect bandwidth (n, k) linebacking code $\mathcal{C}$ with $k \geq 3$ and t substripes where $t(t-1) > \frac{(n-k)(n-k-1)}{k}$ always uses all remaining systematic nodes to repair a failed node. Consider only the repair of the systematic nodes. By assumption, for any pair of systematic nodes, each participates in the other's repair. Their repair sets overlap by at least k if and only if there are at least 2 parity nodes which repair both, and each systematic node has t parity nodes repairing it. Per [15], the maximum number of sets of t parity nodes such that no two sets have 2 parity nodes in common is $\frac{\binom{n-k}{2}}{\binom{t}{2}}$. Thus if $k > \frac{(n-k)(n-k-1)}{t(t-1)}$, two systematic nodes must share 2 helper parity nodes, and thus have an overlap of size at least k in their repair sets as well as each participating in the other's repair. However, this meets the assumptions of Lemma 10, which is impossible. □

Theorem 12 suggests that linebacking codes may achieve better bandwidth if they do *not* follow the standard practice of using all remaining systematic nodes in every repair, since the bound $t \leq \frac{n-k-1}{\sqrt{k}}$ is more restrictive than $t \leq \frac{n-k+1}{2}$ from Theorem 11 as k grows.

Finally, we end our discussion of the "some d" regime with a theorem showing that decreasing the number of substripes in this regime does not make constructing perfect bandwidth codes any harder. More precisely, if a perfect bandwidth piggybacking code with t substripes exists, then such codes also exist with $t-1$ substripes.

Theorem 13. *Let $\mathcal{C}$ be a perfect bandwidth (n, k) piggybacking code over $\mathbb{F}_q$ with t substripes. Then there exist perfect bandwidth piggybacking codes for the same n, k, q and any number of substripes up to t.*

Proof idea. We begin with a perfect bandwidth code $\mathcal{C}$, with base code generator matrix F and piggybacking matrices $\{P^{(i,j)} \mid i \in [0, t-2], j \in [i+1, t-1]\}$. We construct a new code $\mathcal{C}'$ with $t-1$ stripes by removing the 0^{th} substripe of the message and each node, along with the corresponding piggybacking functions $\{P^{(0,j)}\}$. We then establish that any repair scheme for $\mathcal{C}$ can be transformed into a repair scheme for $\mathcal{C}'$ with strictly decreased bandwidth. This is done by modifying the original repair scheme so that one repair matrix repairs only the 0^{th} substripe and is guaranteed to contribute at least one unique query to the total bandwidth. We then delete this matrix, and the 0^{th} columns of the remaining repair matrices, and prove that this results in a valid repair scheme for $\mathcal{C}'$. This process can be repeated to achieve any number of substripes less than t. □

Thus, a negative result for any fixed t_0 would imply a negative result for all $t \geq t_0$, and a positive result for t_0 would imply a positive result for all $t \leq t_0$.

Additionally, Theorem 13, along with Lemma 4 about the required alphabet size for $t = 2$, imply that perfect bandwidth piggybacking codes must have large alphabets.

Corollary 14. *Let $\mathcal{C}$ be a perfect bandwidth (n, k) piggybacking code over $\mathbb{F}_q$. Then $q \geq k + 1$.*

VII. Conclusion

We adapted the framework of [6] in order to analyze the achievable bandwidth of piggybacking codes introduced by [2] with scalar MDS base codes for low substriping $t \leq n - k$. In the regime where any d nodes must be able to repair a failed node, we showed that for $k \geq 3$ piggybacking codes cannot achieve the lower bound on bandwidth, and thus are less powerful than general linear codes. We established by counterexample that this result does not in general extend to the regime where only some d nodes repair a failed node, and partially addressed the question of whether piggybacking codes can achieve the lower bound on bandwidth in this regime. We additionally gave impossibility results for linebacking, a subcategory of piggybacking in the style of [3]. Some questions about the theoretical capabilities and limitations of piggybacking codes remain to be addressed, and we conclude with these.

1) When do there exist perfect bandwidth piggybacking codes for the "some d" regime and $k \geq 3$? When they do not exist, how close can piggybacking codes get to the lower bound on bandwidth?
2) Is linebacking less powerful than piggybacking? We conjecture that this is so, and note that an example of a perfect bandwidth piggybacking code in the regime where Theorem 11 holds would establish this.
3) Adding the (commonly used) assumption that all remaining systematic nodes assist in the repair of a failed node gave us a stronger impossibility result for linebacking in Theorem 12. Does this assumption actually worsen the achievable bandwidth for piggybacking (or general) codes?
4) Our analysis of piggybacking codes was limited compared to the proposal of [2] in a few ways. How does the analysis change if we permit non-linear piggybacking functions, or allow an MDS array base code? How well can piggybacking codes perform on other metrics such as data-read and computation as well as bandwidth?

Acknowledgment

We thank Rashmi Vinayak for introducing us to the problem, and for very helpful correspondences.

References

[1] R. Hulett and M. Wootters, "Limitations on the achievable repair bandwidth of piggybacking codes with low substriping," 2017, [arXiv preprint https://arxiv.org/abs/1707.02337].

[2] K. V. Rashmi, N. B. Shah, and K. Ramchandran, "A piggybacking design framework for read- and download-efficient distributed storage codes." IEEE, 7 2013, pp. 331–335.

[3] B. Yang, X. Tang, and J. Li, "A systematic piggybacking design for minimum storage regenerating codes," *IEEE Transactions on Information Theory*, vol. 61, no. 11, pp. 5779–5786, 2015.

[4] C. Shangguan and G. Ge, "New piggybacking design for systematic MDS storage codes," 2016.

[5] S. Kumar, A. Graell i Amat, I. Andriyanova, and F. Brännström, "A family of erasure correcting codes with low repair bandwidth and low repair complexity." IEEE, 12 2015, pp. 1–6.

[6] V. Guruswami and M. Wootters, "Repairing Reed-Solomon codes." ACM, 6 2016, pp. 216–226.

[7] K. V. Rashmi, N. B. Shaw, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure coded storage systems: A study on the Facebook warehouse cluster," *UNISEX HotStorage*, 2013.

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

[9] C. Suh and K. Ramchandran, "Exact-repair MDS code construction using interference alignment," *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1425–1442, 2011.

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

[11] K. Kralevska, D. Gligoroski, and H. Øverby, "General sub-packetized access-optimal regenerating codes," *IEEE Communications Letters*, vol. 20, no. 7, pp. 1281–1284, 7 2016.

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

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

[14] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Explicit codes minimizing repair bandwidth for distributed storage." IEEE, 1 2010, pp. 1–11.

[15] P. Erdös and H. Hanani, "On a limit theorem in combinatorial analysis," *Publicationes Mathematicae Debrecen*, vol. 10, pp. 10–13, 1963.