A ZigZag-Decodable Code with the MDS Property for Distributed Storage Systems

Chi Wan Sung
Department of Electronic Engineering
City University of Hong Kong
Email: albert.sung@cityu.edu.hk

Xueqing Gong
Department of Electronic Engineering
City University of Hong Kong
Email: xgong6@student.cityu.edu.hk

***Abstract*—A code is said to have the MDS property if it maps K source blocks into N coded blocks, while any K out of the N coded blocks allow recovery of the original K source blocks. A new vector code that has the MDS property is designed. It allows the source blocks to be recovered by using a fast algorithm called ZigZag decoding. It can serve as the Fractional Repetition code, a regenerating code for distributed storage systems. The overall design becomes very simple to implement, as node failure can be handled by uncoded and exact repair while data retrieval can be performed by XOR operations.**

I. Introduction

Many applications require a code to map K source blocks into N coded blocks and any K out of the N block can recover the original K source blocks. We term it the Maximum Distance Separable (MDS) property. Take data storage as an example. Suppose there is a data object of B symbols. We want to store it in N nodes, which can be hard disks or data servers. We can first divide the object into K blocks. Each data block can be regarded as a vector of length $L = B/K$. We can then apply an MDS code to map the K blocks into N coded blocks, each also of length L, on a symbol-by-symbol basis. These N coded blocks are then stored in the N nodes. The original object can be recovered from the coded blocks stored in any K nodes.

While Reed-Solomon code is well known to have the MDS property, it sometimes requires a large finite-field size, which makes the encoding and decoding complexities high. For this reason, we design a new vector code that allows fast encoding and decoding, which operate over GF(2). To do this, we relax the traditional constraint that the length of the coded blocks has to be the same as that of the source blocks. We allow the length of a coded block to be slightly larger, say $L+r$, where $r > 0$. This reduces the code rate from K/N to $KL/N(L+r)$. For some applications, like data storage, the block length, L, is typically large. The reduction in code rate becomes small and the slight loss in storage efficiency should be acceptable.

Our code is designed in a way such that ZigZag decoding can be applied. ZigZag decoding was originally designed to combat hidden terminals in wireless networks [1]. When two terminals have their two packets transmitted and collided more than once with different relative time offsets, then ZigZag decoding is able to recover the two packets from the collisions. This method can recover collided packets because collisions in a wireless environment can be represented algebraically as a linear combination of original packets [2]. It was shown in [3] that ZigZag decoding can also be applied to the case where there are three colliding packets. To the best of our knowledge, it is not known under what situations ZigZag decoding can work correctly for more than three collided packets. In this paper, we obtain coded blocks by aligning the source blocks with carefully chosen offsets. In our design, there is no limitation on the number of source blocks, and it is guaranteed that the source blocks can always be recovered by ZigZag decoding.

In addition to traditional data storage systems, our code can also be applied to distributed storage systems, in which the storage nodes are geographically located in different places and connected by a network. One feature of the distributed storage system is that the amount of data required to repair a failed node, called repair bandwidth, should be minimized [4]. An exact Minimum-Bandwidth Regenerating (MBR) code, which consists of an outer MDS code followed by an inner fractional repetition (FR) code, was proposed in [5]. This code has also been used in the design of heterogeneous distributed storage systems [6]. Our new code design can be applied to distributed storage systems by being the inner FR code. The overall design becomes very simple, as only XOR operations are needed for data retrieval, and uncoded and exact repair can be performed when a node fails.

II. Illustrative Examples

To illustrate our idea, we first consider a simple example, which requires a code to map $K = 2$ source blocks of L bits into $N = 5$ coded blocks of $L + 1$ bits. We consider a systematic code so that the first two coded blocks are the same as the two source blocks, except that each coded block has one extra dummy bit appended at the end. The remaining three coded blocks are obtained by applying the XOR operation in a bit-wise manner between the two source blocks with relative offsets of -1, 0, and 1, respectively. The five coded blocks is shown graphically in Fig. 1.

It is easy to see that this code has the MDS property. Given any two coded blocks, the original source blocks can be

This work was partially supported by a grant from the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-02/08).

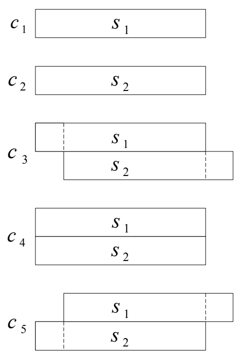

Fig. 1. A code with the MDS property: K = 2, N = 5.

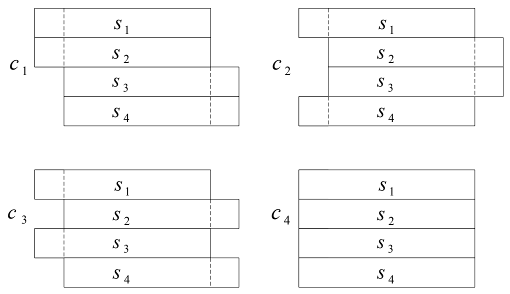

Fig. 2. ZigZag decoding does not always work.

recovered, and the decoding algorithm is simple. For example, suppose the last two coded blocks, $\boldsymbol{c}_4$ and $\boldsymbol{c}_5$, are given. First, we read the first bit of the source block 2 from $\boldsymbol{c}_5$, since it is not “overlapped” with the first source block. Then this bit can be subtracted from the other coded block. Next, we read the first bit of the source block 1 from $\boldsymbol{c}_4$, since it is now exposed, meaning that it is not overlapped with any bit of the other source block. This process can continue until all bits of the two source blocks are determined, and this method is called ZigZag decoding.

From this example, it can be seen that by adding only one redundant bit, i.e., $r = 1$, we can obtain a code, which can be used as an (5,2) MDS code, and has a fast decoding algorithm which uses only the XOR operation.

Next we construct another example. Suppose four coded blocks are given, which are graphically depicted in Fig. 2. We want to recover the original four source blocks. Let $s_{i,1}$ be the first bit of source block i and $c_{i,1}$ be the first bit of coded block i, where $i = 1, 2, 3$ and 4. Then we have the following four linear equations:

$$s_{1,1} + s_{2,1} = c_{1,1} \tag{1}$$

$$s_{1,1} + s_{4,1} = c_{2,1} \tag{2}$$

$$s_{1,1} + s_{3,1} = c_{3,1} \tag{3}$$

$$s_{1,1} + s_{2,1} + s_{3,1} + s_{4,1} = c_{4,1} \tag{4}$$

It can be seen that the coefficient matrix of this linear system has full rank and is invertible. Therefore, the first bits of all source blocks can be uniquely determined, and can be subtracted off from the coded blocks. The same method can be repeatedly applied to find the subsequent bits. However, in this scenario, ZigZag decoding does not work, since the first bit of every source block is overlapped with some bit of other blocks.

The above example is relatively simple, as it just happens that the first bit of each source block can be decoded from the first bits of the coded blocks. In general, decoding the source blocks is equivalent to solving a linear system, which may involve a large number of symbols. This can be done by Gaussian elimination, provided that the associated coefficient matrix is of full rank. ZigZag decoding is more efficient in the sense that it involves only the backward substitution step, since at each iteration, there is at least one bit being exposed and can be directly read from the corresponding coded block.

III. Code Construction

Suppose we have K source blocks, each of which consists of L bits. We represent block i by the polynomial

$$s_i(z) \triangleq s_{i,0} + s_{i,1}z + s_{i,2}z^2 + \cdots s_{i,L-1}z^{L-1}, \tag{5}$$

where $s_{i,j}$ is the $(j+1)$-th bit of block i.

These K source blocks are encoded into N blocks, each of which has $L + r$ bits. In other words, each coded block has r more bits than the source blocks. The i-th coded block is given by

$$c_i(z) = \alpha_{i,1}(z)s_1(z) + \alpha_{i,2}(z)s_2(z) + \cdots \alpha_{i,K}(z)s_K(z), \tag{6}$$

where $i = 1, 2, \ldots, N$. The number of extra bits, r, is equal to the maximum degree of all $\alpha_{i,j}(z)$'s. In matrix form, we have

$$\boldsymbol{c}(z) = \boldsymbol{A}(z)\boldsymbol{s}(z), \tag{7}$$

where $\boldsymbol{c}(z)$ is an N-dimensional vector whose i-th component is $c_i(z)$, $\boldsymbol{s}(z)$ is a K-dimensional vector whose j-th component is $s_j(z)$, and $\boldsymbol{A}(z)$ is an $N \times K$ matrix whose (i, j)-th entry is $\alpha_{i,j}(z)$.

In our construction, $\boldsymbol{A}(z)$ can be partitioned into two blocks:

$$\boldsymbol{A}(z) \triangleq \begin{bmatrix} \boldsymbol{I}_K \\ \boldsymbol{B}(z) \end{bmatrix}, \tag{8}$$

where $\boldsymbol{I}_K$ is the $K \times K$ identity matrix and $\boldsymbol{B}(z)$ is a $(N - K) \times K$ matrix whose (i, j)-th entry is denoted by $\beta_{i,j}(z)$. Furthermore, we define

$$\beta_{i,j}(z) \triangleq z^{i(j-1)}. \tag{9}$$

Then we have $r = (N - K)(K - 1)$.

For example, when $K = 3$ and $N = 6$, we have

$$\boldsymbol{A}(z) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & z & z^2 \\ 1 & z^2 & z^4 \\ 1 & z^3 & z^6 \end{bmatrix}, \tag{10}$$

which is graphically shown in Fig. 3. Clearly, r equals 6.

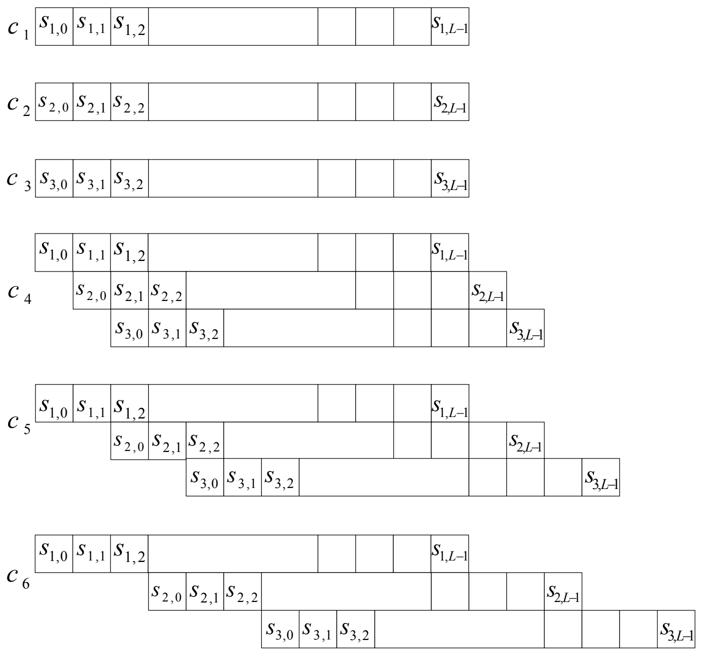
c_1
c_2
c_3
c_4
c_5
c_6
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,L-1}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,L-1}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,L-1}$

Fig. 3. Example: $K = 3$, $N = 6$

IV. ZigZag Decoding

Originally, ZigZag decoding is designed for the case where there are only two "overlapping" blocks. In this section, we generalize it to the case where there is no limitation on the number of blocks.

Consider an $M \times M$ matrix, which takes the following form:

$$\boldsymbol{M}(z) = \begin{bmatrix} z^{t_{1,1}} & z^{t_{1,2}} & \dots & z^{t_{1,M}} \\ z^{t_{2,1}} & z^{t_{2,2}} & \dots & z^{t_{2,M}} \\ \vdots & \vdots & \ddots & \vdots \\ z^{t_{M,1}} & z^{t_{M,2}} & \dots & z^{t_{M,M}} \end{bmatrix}. \tag{11}$$

We further assume that $\boldsymbol{M}(z)$ satisfies the following *increasing-difference* property: If $i < i'$ and $j < j'$, then

$$0 < t_{i,j'} - t_{i,j} < t_{i',j'} - t_{i',j}. \tag{12}$$

Consider the following linear system:

$$\boldsymbol{y}(z) = \boldsymbol{M}(z)\boldsymbol{x}(z), \tag{13}$$

where $\boldsymbol{x}(z)$ and $\boldsymbol{y}(z)$ are M-dimensional vectors, whose i-th components, $x_i(z)$ and $y_i(z)$, are both polynomials in z of degree $L - r + 1$.

Given a polynomial $f(z)$, define $\Omega(f(z))$ as its lowest order term and $\omega(f(z))$ as its lowest order. For example, if $f(z) = az^2 + bz^3 + cz^4$, where a, b and c are variables to be determined, then $\Omega(f(z)) = az^2$ and $\omega(f(z)) = 2$. Besides, define $\mathcal{M} \triangleq \{1, 2, \dots, M\}$.

Now we formally state the ZigZag decoding algorithm as follows:

Step 0: Let $\mathcal{M}' := \mathcal{M}$ and $\hat{\boldsymbol{x}}(z) := \boldsymbol{0}$. For all $j \in \mathcal{M}$, let

$$\eta_j(z) := 1 + z + \dots + z^{L+r-1}.$$

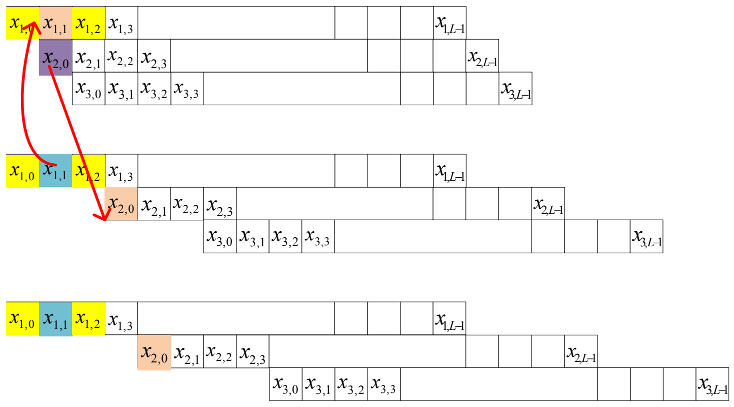
$x_{1,0}$ $x_{1,1}$ $x_{1,2}$ $x_{1,3}$ $x_{1,L-1}$
$x_{2,0}$ $x_{2,1}$ $x_{2,2}$ $x_{2,3}$ $x_{2,L-1}$
$x_{3,0}$ $x_{3,1}$ $x_{3,2}$ $x_{3,3}$ $x_{3,L-1}$

Fig. 4. An illustration of ZigZag decoding

Step 1: Find $i^* \in \mathcal{M}$ and $j^* \in \mathcal{M}'$ such that

$$\omega(z^{t_{i^*,j^*}}\eta_{j^*}(z)) < \omega(z^{t_{i^*,j}}\eta_j(z))$$

for all $j \in \mathcal{M}' \setminus \{j^*\}$. (Any i^* and j^* can be chosen if there are more than one choices.)

Step 2: The variable associated with the lowest order term of $x_{j^*}(z)$ can then be determined from $\Omega(y_{i^*}(z))$. Do the following updating:

- Let $\hat{x}_{j^*}(z) := \hat{x}_{j^*}(z) + \Omega(y_{i^*}(z))$.
- Let $y_i(z) := y_i(z) - z^{t_{i,j^*}}\Omega(x_{j^*}(z))$ for all $i \in \mathcal{M}$.
- Remove from $\eta_{j^*}(z)$ its lowest order term. If $\eta_{j^*}(z)$ has no more terms, then let $\mathcal{M}' := \mathcal{M}' \setminus \{j^*\}$.

Step 3: Go to Step 1 if $\mathcal{M}' \neq \emptyset$.

Step 4: Output $\hat{x}_j(z)$ for all $j \in \mathcal{M}$.

Example: Consider the case where $\boldsymbol{M}(z)$ is a 3×3 matrix given by

$$\boldsymbol{M}(z) = \begin{bmatrix} 1 & z & z^2 \\ 1 & z^3 & z^6 \\ 1 & z^4 & z^8 \end{bmatrix}. \tag{14}$$

The coded blocks, $y_1(z), y_2(z)$ and $y_3(z)$, are graphically shown in Fig. 4. Consider the first iteration when executing the ZigZag decoding algorithm. In Step 1, $j^* = 1$, which means that the lowest order term (i.e. the first bit) of $x_1(z)$, i.e., $x_{1,0}$, can be determined, and i^* can be chosen as either 1, 2, or 3, which means that any of the three coded blocks can be used to determine the lowest order term of $x_1(z)$. (It does not matter which choice was made.) In Step 2, some updating will be performed. Step 3 leads the algorithm to the second iteration. In Step 1, j^* is again equal to 1. But this time the lowest order term of $x_1(z)$ (i.e. the second bit) can only be determined from the second and the third coded block, i.e., i^* can be chosen as either 2 or 3. In Step 2, $y_1(z)$ will be updated by subtracting $x_{1,1}z$ from itself. The algorithm then enters the third iteration. This time, in Step 1, (i^*, j^*) can be chosen as either (1, 2) or (2, 1) or (3,1), meaning that we can either get $x_{2,0}$ (if (i^*, j^*) is chosen as (1,2)) or $x_{1,1}$ (if (i^*, j^*) is chosen as (2,1) or (3,1)). The algorithm then proceeds as before until all source blocks have been determined. ■

Next we want to show that the ZigZag decoding can decode successfully. Before that, we need to define some notions, and prove a lemma. For $i \in \mathcal{M}$, define

$$\mathcal{S}_i \triangleq \arg\min_{j\in\mathcal{M}} \omega(z^{t_{i,j}}\eta_j(z)). \tag{15}$$

We say that $\mathcal{S}_i \succ \mathcal{S}_k$ if the smallest element in $\mathcal{S}_i$ is greater than or equal to the largest element in $\mathcal{S}_k$.

Lemma 1. *$\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_M$ satisfy the following properties:*

1) *$\mathcal{S}_i \succ \mathcal{S}_{i'}$ if $i < i'$.*
2) *$|\mathcal{S}_i \cap \mathcal{S}_{i+1}| \le 1$, for $i = 1, 2, \dots, M-1$.*
3) *$\cup_{i\in\mathcal{M}}\mathcal{S}_i$ can be partitioned into the following M disjoint subsets: $\mathcal{S}_1 \setminus \mathcal{S}_2, \mathcal{S}_2 \setminus \mathcal{S}_3, \dots, \mathcal{S}_{M-1} \setminus \mathcal{S}_M, \mathcal{S}_M$.*

Proof. The first property is equivalent to this: If $j \in \mathcal{S}_i$ and $j' \in \mathcal{S}'_i$, where $i < i'$, then $j \ge j'$. To prove it, first note that $j \in \mathcal{S}_i$ means

$$\omega(z^{t_{i,j}}\eta_j(z)) \le \omega(z^{t_{i,k}}\eta_k(z)), \ \ \forall k \in \mathcal{M} \tag{16}$$

and $j' \in \mathcal{S}'_i$ means

$$\omega(z^{t_{i',j'}}\eta_{j'}(z)) \le \omega(z^{t_{i',k}}\eta_k(z)), \ \ \forall k \in \mathcal{M}. \tag{17}$$

Since $\boldsymbol{M}(z)$ satisfies the increasing-difference property, (16) implies

$$\omega(z^{t_{i',j}}\eta_j(z)) < \omega(z^{t_{i',k}}\eta_k(z)), \ \ \forall k > j. \tag{18}$$

This, together with (17), implies $j' \le j$.

The second property clearly follows from the first one.

We prove the third property by mathematical induction. It is clear that $\mathcal{S}_1 \cup \mathcal{S}_2$ can be partitioned into $\mathcal{S}_1 \setminus \mathcal{S}_2$ and $\mathcal{S}_2$. Assume that $\cup_{i=1}^{k}\mathcal{S}_i$ can be partitioned into

$$\mathcal{S}_1 \setminus \mathcal{S}_2, \mathcal{S}_2 \setminus \mathcal{S}_3, \dots, \mathcal{S}_{k-1} \setminus \mathcal{S}_k, \mathcal{S}_k.$$

Consider $\cup_{i=1}^{k+1}\mathcal{S}_i$. It can be partitioned into

$$\mathcal{S}_1\setminus(\mathcal{S}_2\cup\mathcal{S}_{k+1}), \mathcal{S}_2\setminus(\mathcal{S}_3\cup\mathcal{S}_{k+1}), \dots, \mathcal{S}_{k-1}\setminus(\mathcal{S}_k\cup\mathcal{S}_{k+1}), \mathcal{S}_{k+1}.$$

The first two properties imply that for $i = 1, 2, \dots, k-1$,

$$\mathcal{S}_i \setminus (\mathcal{S}_{i+1} \cup \mathcal{S}_{k+1}) = \mathcal{S}_i \setminus \mathcal{S}_{i+1}. \tag{19}$$

The third property then follows by mathematical induction. □

Theorem 2. *Given $\boldsymbol{y}(z)$ and $\boldsymbol{M}(z)$ which has the increasing-difference property, ZigZag decoding is able to recover $\boldsymbol{x}(z)$.*

Proof. ZigZag decoding can successfully recover $\boldsymbol{x}(z)$ if it can always find an interference-free symbol of a source block from one of the coded blocks, or more formally, if it can always find i^* and j^* in Step 1 during each iteration.

We prove by contradiction. Suppose the algorithm fails in Step 1 during a certain iteration. Then we must have $|\mathcal{S}_i| \ge 2$ for all i. By the third property in Lemma 1, we have

$$\Big|\bigcup_{i\in\mathcal{M}}\mathcal{S}_i\Big| = \sum_{i=1}^{M-1}|\mathcal{S}_i \setminus \mathcal{S}_{i+1}| + |\mathcal{S}_M|. \tag{20}$$

Since $|\mathcal{S}_i| \ge 2$, by the second property of Lemma 1, we have

$$\Big|\bigcup_{i\in\mathcal{M}}\mathcal{S}_i\Big| \ge (M-1) + 2 \ge M+1, \tag{21}$$

which is a contradiction. Therefore, the algorithm can always find i^* and j^* in Step 1, which means that there always exists a source block whose lowest order term is exposed, without overlapping with other bits. The algorithm stops only when all bits of all source blocks are determined. □

In the ZigZag decoding algorithm, totally there will be ML iterations, since there are ML bits to be recovered. In each iteration, a naive implementation of Step 1 would require searching over all i's and j's, which takes $O(M^2)$ operations. The updating in Step 2 requires $O(M)$ operations. Then, the overall computational complexity is $O(M^3L)$. This, however, is not the most efficient way to implement the idea of ZigZag decoding. In fact, Step 1 can be implemented in a more efficient way. For each coded block $i \in \mathcal{M}$, we can first initialize an integer array $Array_i$ of length $L + r$. For $i = 1, 2, \dots, L + r$, let

$$Array_i[n] := \begin{cases} 0 & \text{if } 1 \le n < t_{i,1} \\ j & \text{if } t_{i,j} \le n < t_{i,j+1} \\ M & \text{otherwise} \end{cases} \tag{22}$$

Furthermore, let p_i be the index that specifies the first non-zero entry in $Array_i$. In Step 1, we only need to search for i such that $Array_i[p_i]$ is equal to 1, which takes only $O(M)$ operations. Corresponding updating of all arrays and all p_i's should be done in Step 2, which also takes $O(M)$ operations. The overall complexity can then be reduced to $O(M^2L)$.

V. MDS Property and ZigZag Decodability

Now we show that our code constructed in Section III possesses the MDS property.

Theorem 3. *Given any K out of the N coded blocks, the K source blocks can be recovered.*

Proof. Let $\mathcal{C} \subset \{1, 2, \dots, N\}$ be the index set corresponding to the given K coded blocks. Partition it into $\mathcal{C}_0$ and $\mathcal{C}_1$, where all elements in $\mathcal{C}_0$ are less than or equal to K. Note that $s_i(z)$ is the same as the given coded block $c_i(z)$ for $i \in \mathcal{C}_0$. If $\mathcal{C}_1$ is empty, then all the source blocks are known and we are done. Otherwise, subtract all elements in $\mathcal{C}_1$ by K and put them all into a set $\mathcal{I}$. Define $\mathcal{J}$ as $\{1, 2, \dots, K\} \setminus \mathcal{C}_0$.

Let $\boldsymbol{B}_{\mathcal{I},\mathcal{J}}(z)$ be the submatrix of $\boldsymbol{B}(z)$ obtained by choosing the rows whose indices are in $\mathcal{I}$ and the columns whose indices are in $\mathcal{J}$. Note that we must have $|\mathcal{I}| = |\mathcal{J}|$, and we denote its value by M. Then $\boldsymbol{B}_{\mathcal{I},\mathcal{J}}(z)$ is an $M \times M$ square matrix.

Consider the following linear system:

$$\boldsymbol{c}_{\mathcal{I}}(z) = \boldsymbol{B}_{\mathcal{I},\mathcal{J}}(z)\boldsymbol{s}_{\mathcal{J}}(z), \tag{23}$$

where $\boldsymbol{c}_{\mathcal{I}}(z)$ is the sub-vector obtained from $\boldsymbol{c}(z)$ by retaining only those components in $\mathcal{I}$ and $\boldsymbol{s}_{\mathcal{J}}(z)$ is the sub-vector obtained from $\boldsymbol{s}(z)$ by retaining only those components in $\mathcal{J}$. By construction, $\boldsymbol{B}(z)$ is a Vandermonde matrix [7].

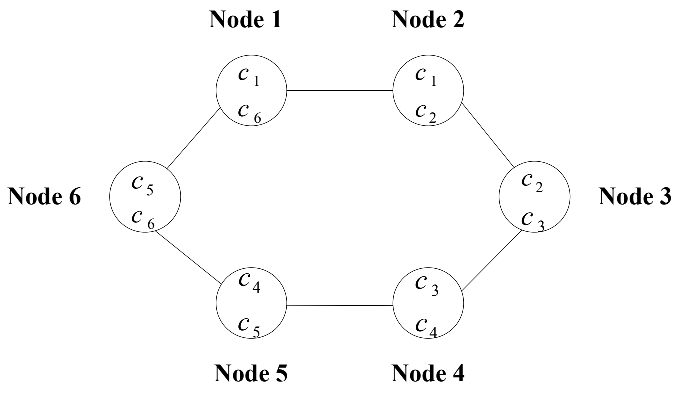

Fig. 5. A distributed storage system with a ring as its repair overlay network.

$\boldsymbol{B}_{\mathcal{I},\mathcal{J}}(z)$, being a square sub-matrix of $\boldsymbol{B}(z)$, is a generalized Vandermonde matrix. Since generalized Vandermonde matrix is invertible [7], $\boldsymbol{s}_{\mathcal{J}}(z)$ can be uniquely determined. □

The next theorem shows that the K source blocks not only can be recovered, but also can be recovered by ZigZag decoding. It also serves as an alternative proof of Theorem 3.

Theorem 4. *Given any K out of the N coded blocks, the K source blocks can be recovered by the ZigZag decoding algorithm.*

Proof. Following the proof of Theorem 3, we have the linear system in (23). By definition, $\boldsymbol{B}_{\mathcal{I},\mathcal{J}}(z)$ has the increasing-difference property. By Theorem 2, $\boldsymbol{s}_{\mathcal{J}}(z)$ can be recovered by zigzag decoding. □

VI. Application to Distributed Storage Systems

Consider storing a data object in a distributed storage system of n nodes. In the original model in [4], it is required that a data collector can recover the data object by downloading data from any k out of the n nodes. If a node fails, a newcomer replaces the failed node by contacting any d out of the $n-1$ survivor nodes. We call this the *conventional* model. A fundamental tradeoff between storage capacity and repair bandwidth is derived in [4].

Under the model in [5], the repair requirement is relaxed. A newcomer has to contact a specific set of d nodes for repair, depending on which node has failed. A code which uses an outer MDS code followed by an inner FR code, is proposed. A useful feature of this code is that it supports uncoded repair, which means that a newcomer can simply download data from d survivor nodes without any encoding or decoding operations. This work was further extended in [6], which further relaxes the data retrieval requirement. It is only required that a data collector can retrieve the data object by contacting k nodes in some specific sets of nodes. For example, consider a system where $n = 6$ and $k = d = 2$. The repair overlay network is a ring, as shown in Fig. 5, which means that a newcomer for a certain failed node can rebuild the lost data by contacting the two neighbors of the failed node. The MDS-FR code is used

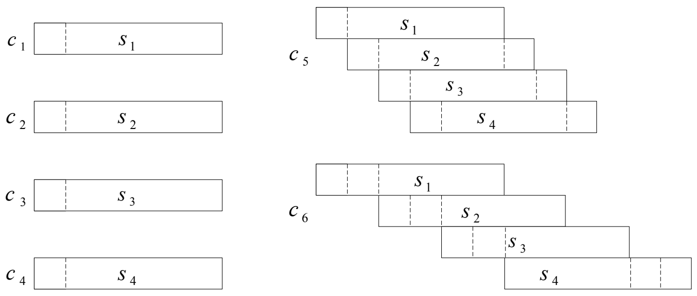

Fig. 6. Our new code with $K = 4$ and $N = 6$.

such that the data object is first divided into four source blocks, and then encoded into six coded blocks. This encoding process can be done by using our new code, which is graphically depicted in Fig. 6. For this example, the overhead per coded block is $(N-K)(K-1) = 2 \times 3 = 6$ bits, and so the relative overhead in the storage system is $6/L$. Each node then stores two coded blocks as shown in Fig. 5. A data collector can retrieve the data object by contacting any two nodes in either $\{1,3,5\}$ or $\{2,4,6\}$ due to the MDS property. It was shown in [6] that such a design, by relaxing some of the over-restrictive requirements, can allow the system operate at a point below the tradeoff curve (Details can be found in [6]). Besides, with our new code, data retrieval can be done by ZigZag decoding, which is fast and simple to implement.

VII. Conclusion

A new vector code which has the MDS property is designed. By slightly increasing the length of each coded block, our code allows the source blocks be recovered by using ZigZag decoding, which is a fast algorithm operating over GF(2). It can be used to replace traditional MDS codes, which often require large finite-field sizes. As a result, the encoding and decoding computation times can then be shortened. When the code is used as the FR code, it can simplify the implementation of distributed storage systems.

References

[1] S. Gollakota and D. Katabi, "Zigzag decoding: combating hidden terminals in wireless networks," in *Proc. SIGCOM*, New York, USA, Oct. 2008, pp. 159–170.

[2] A. ParandehGheibi, J. K. Sundararajan, and M. Medard, "Collision helps: algebraic collision recovery for wireless erasure networks," in *Proc. Wireless Network Coding Conference*, Jun. 2010, pp. 1–6.

[3] S. Gollakota, "Zigzag decoding: combating hidden terminals in wireless networks," *Master thesis, MIT*, Jun. 2008.

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

[5] S. El Rouayheb and K. Ramchandran, "Fractional repetition codes for repair in distributed storage systems," in *Proc. Allerton conference on commun. control and computing*, Monticello, Sep. 2010.

[6] Q. Yu, C. W. Sung, and T. H. Chan, "Repair topology design for distributed storage systems," in *Proc. IEEE ICC*, Ottawa, Canada, Jun. 2012, pp. 7009–7013.

[7] F. R. Gantmacher, *The Theory of Matrices, vol. 2*. New York: Chelsea Pub., 1977.