Zigzag-Decodable Reconstruction Codes with Asymptotically Optimal Repair for All Nodes

Hanxu Hou, *Member, IEEE*, Patrick P. C. Lee, *Senior Member, IEEE* and Yunghsiang S. Han, *Fellow, IEEE*

***Abstract*—Zigzag-decodable codes have been proposed for distributed storage systems to achieve fast decoding of uncoded data packets through the iterative decoding of data bits from coded packets. To maintain high data availability, it is critical to minimize the repair bandwidth by downloading the least amount of bits for repairing any lost packet. In this work, we propose zigzag-decodable reconstruction (ZDR) codes which achieve asymptotically minimum repair bandwidth for repairing a single node, while preserving the high computational efficiency due to zigzag decoding. We present two explicit constructions of ZDR codes such that any node of ZDR codes can be repaired with asymptotically minimum repair bandwidth. The first construction is based on the well-designed encoding matrix and a generic transformation, while the second construction is designed by recursively employing the proposed generic transformation for any existing zigzag-decodable code. Moreover, we show that the proposed two classes of ZDR codes can be decoded by the zigzag decoding algorithm and have less computational complexity than the existing codes with asymptotically or exactly minimum repair bandwidth.**

***Index Terms*—Zigzag-decodable codes, minimum repair bandwidth, zigzag-decodable reconstruction codes, computational complexity.**

I. Introduction

Distributed storage systems achieve high fault tolerance by striping data redundancy across multiple storage nodes. Reed-Solomon (RS) codes [1] are a well-known family of erasure codes that have been widely implemented in production [2], [3]. An (n, k) RS code divides file data into k uncoded data packets of L bits, which are encoded into additional $r = n-k$ coded packets of the same size. The n packets are stored in n nodes (one packet per node). The *maximum distance separable (MDS) property*, that any k out of the n nodes can recover all the original data packets, should be maintained to tolerate any $n-k$ node failures. RS codes are *storage-optimal*, as they incur the minimum amount of redundancy to achieve the MDS property. However, RS codes have two drawbacks. First, RS codes perform all coding operations over a large finite field, thereby incurring high encoding and decoding complexities. Second, RS codes download k packets from surviving nodes to repair any lost packet, thereby incurring high repair bandwidth (i.e., the amount of bits transferred during a repair operation).

H. Hou and Y. S. Han are with the School of Electrical Engineering & Intelligentization, Dongguan University of Technology (E-mail: hhx@dgut.edu.cn). P. P. C. Lee is with Department of Computer Science and Engineering, The Chinese University of Hong Kong (E-mail: pclee@cse.cuhk.edu.hk). This work was partially supported by the National Natural Science Foundation of China (No. 61701115, 61671007), Start Fund of Dongguan University of Technology (No. GB200902-19, KCYXM2017025), and Research Grants Council of Hong Kong (GRF 14216316 and CRF C7036-15G).

Zigzag decoding [4] is a fast decoding technique that allows data bits to be iteratively decoded from coded bits. It is originally designed for wireless communications [4]. Follow-up studies propose zigzag-decodable codes for distributed storage systems [5], [6] to achieve fast decoding of data packets using only XOR operations with the overall decoding complexity $O(k^2L)$ [5]. However, existing zigzag-decodable codes still incur high repair bandwidth as in RS codes.

As failures are prevalent in practice [2], it is critical to minimize the repair bandwidth in the face of failures to maintain high data availability. Assume that a file data with $k\alpha$ data packets is encoded with additional $r\alpha$ coded packets. The data packets and coded packets are stored in $n = k+r$ nodes, where each node stores α packets with the MDS property being satisfied. The number of bits stored in each node, αL, is called *sub-packetization level*. The first k nodes that store data packets are called *data nodes* and the last r nodes that store coded packets are called *coded nodes*. Let d (where $k \leq d \leq n-1$) be the number of surviving nodes that are accessed to repair a single failed node. It is shown in [7] that repairing the α packets in a single failed node needs to download at least $\frac{L\alpha}{d-k+1}$ bits from each of the d surviving nodes, and hence the minimum repair bandwidth is

$$\frac{dL\alpha}{d-k+1} \text{ (bits)}, \tag{1}$$

where L is the size of data packets. Many constructions [7]–[11] have been proposed to achieve the minimum repair bandwidth. However, such constructions operate on a sufficiently large finite field and incur high computational complexity. Thus, we pose the following question: *Can we minimize the repair bandwidth of zigzag-decodable codes, while preserving the decoding efficiency?* This question will be answered in this work.

A. Related Work

Several MDS codes [12]–[19] have been proposed to achieve asymptotically or exactly minimum repair bandwidth, while incurring lower computational complexity. One way to reduce the computational complexity is to reduce the underlying field size. Some constructions over small nonbinary field with optimal repair bandwidth are given in [11], [20], [21]. Computational complexity would be especially lower if a code is designed over the binary field. For example, MDR codes [12] and ButterFly codes [13] achieve the minimum repair bandwidth with two coded nodes (i.e., $r = 2$). Some constructions over the binary field support more than two

coded nodes with the asymptotically minimum repair bandwidth [15]–[18]; however, they do not provide any efficient decoding algorithm when some nodes are erased. A generic transformation for EVENODD codes [22], [23] is proposed in [24] to enable the minimum repair bandwidth such that the existing efficient decoding method of EVENODD codes [23], [25] can be employed in the transformed EVENODD codes. However, the efficient decoding methods in [23], [25] are only applicable for some special cases corresponding to solving the linear systems with Vandermonde matrix. The decoding complexity of the general case is with $O(k^2L^2)$, where L is the packet length.

Another way to reduce the computational complexity, especially the decoding complexity, is to design codes that can avoid the step of transforming the system of linear equations to row echelon form or computing the inverse matrix of the encoding matrix, where only backward substitution is required. Zigzag-decodable codes [5] are such codes that only require backward substitution (call *zigzag decoding*) in the decoding process. Some constructions of zigzag-decodable codes have been proposed in [5], [6], [26], however, they do not consider the repair problem of one failed node and the repair bandwidth is high. Similar idea is used in index coding [27] and secret sharing scheme [28]. Product-matrix regenerating codes [14] achieve the minimum repair bandwidth and can be augmented with zigzag decoding; however, when they build on the product-matrix coding framework [8], they only have a low *code rate* (i.e., the ratio of the file size to the total storage space) up to 0.5.

B. Contribution

This paper proposes a new class of erasure codes called *zigzag-decodable reconstruction* (ZDR) codes which achieve five practical properties that are critical for real-world deployment: (i) MDS property (i.e., accessing any k out of n nodes can recover the information stored in the k data nodes); (ii) efficient decoding (i.e., any erased data nodes can be decoded through zigzag decoding); (iii) XOR-only operations (i.e., all encoding, decoding, and repair operations involve only bitwise XORs); (iv) high code rate (i.e., the code rate is larger than 0.5); and (v) asymptotically minimum repair bandwidth for a single node (i.e., repairing any single node achieves the minimum repair bandwidth asymptotically in n or L by connecting to $d = n-1$ surviving nodes). Note that in ZDR codes, the size of data packet is L and the size of coded packet is no less than L, but the additional storage overhead of coded packet becomes negligible if L is sufficiently large.

We first give a construction of zigzag-decodable codes by choosing a well-designed encoding matrix that have asymptotically minimum repair bandwidth for a single data node, and then present a generic transformation that can transform any zigzag-decodable codes into a new zigzag-decodable code with asymptotically minimum repair bandwidth for each of any r nodes. Next, we present two explicit constructions of ZDR codes that have the above five practical properties. The first construction is based on the proposed zigzag-decodable codes with asymptotically minimum repair bandwidth for any one of data nodes and the generic transformation for zigzag-decodable codes. The second construction is designed by recursively applying the generic transformation for a zigzag-decodable codes.

Although both the proposed ZDR codes and the zigzag-decodable codes in [5] can be decoded by zigzag decoding, the two codes are fundamental different. First, our ZDR codes have asymptotically or exactly optimal repair bandwidth for all nodes, while not for the codes in [5]. Second, the technique employed in the construction is different. In this paper, we present two classes of ZDR codes. The first class of ZDR codes is constructed by first choosing a well-design encoding matrix and then apply the transformation designed for ZDR codes. The second class of ZDR codes is constructed by recursively applying the proposed transformation for any zigzag-decodable codes, such as the codes in [5]. The zigzag-decodable codes in [5] only consider the zigzag decoding, but not to reduce the repair bandwidth.

Similar transformation in optimizing repair bandwidth of MDS codes can be found in [10], [24], [29]. Li *et al.* [10] propose a transformation for non-binary MDS codes to enable optimal repair bandwidth. Two different transformations for binary MDS array codes are considered in [24], [29] to enable optimal repair bandwidth. A more general transformation for MDS codes to enable optimal repair bandwidth with more parameters is given in [30]. The main differences between our transformation and the transformations in [10], [24], [29], [30] are summarized as follows. First, the structures of our transformation and the transformations in [10], [24], [29], [30] are different. In our transformation, we first make some cyclic-shifts for the $r \times r$ square matrix corresponding to the chosen r nodes that are enabled to have optimal repair bandwidth, and then replace some entries in the $r \times r$ square matrix by linear combinations. In contrast, cyclic-shifts for the $r \times r$ square matrix are not made in the transformations [10], [24], [29], [30]. Because of the above difference, we can apply the transformation for the designed zigzag-decodable codes with asymptotically optimal repair bandwidth for any data node to obtain ZDR codes that have asymptotically or exactly optimal repair bandwidth for any node. Second, the existing transformations [10], [24], [29], [30] do not work for zigzag-decodable codes, while our transformation not only works for zigzag-decodable codes but also for MDS array codes.

This paper is organized as follows. In Section II, we first present a simple motivating example that illustrates the proposed approach. In Section III, we give a new construction of zigzag-decodable codes that have asymptotically minimum repair bandwidth for every data node. Section IV presents a generic transformation for any one zigzag-decodable codes to enable asymptotically minimum repair bandwidth for any coded node. In Section V, we propose ZDR codes that have asymptotically minimum repair bandwidth for all nodes, and give two code constructions of ZDR codes. We also give the repair algorithm and decoding method for the proposed two classes of ZDR codes in Section V. Section VII concludes the paper.

TABLE I: Example of ZDR codes with $(n, k, \alpha) = (4, 2, 2)$ and $L = 8$.

Node 0	Node 1	Node 2	Node 3
$s^0_{0,0}$ $s^1_{0,0}$	$s^0_{0,1}$ $s^1_{0,1}$	$s^0_{0,2} = s^0_{0,0} + s^0_{0,1}$ $+s^1_{0,0} + s^1_{0,1}$ $s^1_{0,2} = s^1_{0,0}$	$s^0_{0,3} = s^0_{0,0}$ $s^1_{0,3} = s^0_{0,0} + s^0_{0,1}$ $+s^1_{4,0} + s^1_{4,1}$
$s^0_{1,0}$ $s^1_{1,0}$	$s^0_{1,1}$ $s^1_{1,1}$	$s^0_{1,2} = s^0_{1,0} + s^0_{1,1}$ $+s^1_{1,0} + s^1_{1,1}$ $s^1_{1,2} = s^1_{1,0} + s^1_{0,1}$	$s^0_{1,3} = s^0_{1,0} + s^0_{0,1}$ $s^1_{1,3} = s^0_{1,0} + s^0_{1,1}$ $+s^1_{5,0} + s^1_{5,1}$
$s^0_{2,0}$ $s^1_{2,0}$	$s^0_{2,1}$ $s^1_{2,1}$	$s^0_{2,2} = s^0_{2,0} + s^0_{2,1}$ $+s^1_{2,0} + s^1_{2,1}$ $s^1_{2,2} = s^1_{2,0} + s^1_{1,1}$	$s^0_{2,3} = s^0_{2,0} + s^0_{1,1}$ $s^1_{2,3} = s^0_{2,0} + s^0_{2,1}$ $+s^1_{6,0} + s^1_{6,1}$
$s^0_{3,0}$ $s^1_{3,0}$	$s^0_{3,1}$ $s^1_{3,1}$	$s^0_{3,2} = s^0_{3,0} + s^0_{3,1}$ $+s^1_{3,0} + s^1_{3,1}$ $s^1_{3,2} = s^1_{3,0} + s^1_{2,1}$	$s^0_{3,3} = s^0_{3,0} + s^0_{2,1}$ $s^1_{3,3} = s^0_{3,0} + s^0_{3,1}$ $+s^1_{7,0} + s^1_{7,1}$
$s^0_{4,0}$ $s^1_{4,0}$	$s^0_{4,1}$ $s^1_{4,1}$	$s^0_{4,2} = s^0_{4,0} + s^0_{4,1}$ $+s^1_{4,0} + s^1_{4,1}$ $s^1_{4,2} = s^1_{4,0} + s^1_{3,1}$	$s^0_{4,3} = s^0_{4,0} + s^0_{3,1}$ $s^1_{4,3} = s^0_{4,0} + s^0_{4,1} + s^1_{4,0}$ $+s^1_{4,1} + s^1_{0,0} + s^1_{0,1}$
$s^0_{5,0}$ $s^1_{5,0}$	$s^0_{5,1}$ $s^1_{5,1}$	$s^0_{5,2} = s^0_{5,0} + s^0_{5,1}$ $+s^1_{5,0} + s^1_{5,1}$ $s^1_{5,2} = s^1_{5,0} + s^1_{4,1}$	$s^0_{5,3} = s^0_{5,0} + s^0_{4,1}$ $s^1_{5,3} = s^0_{5,0} + s^0_{5,1} +$ $s^1_{5,0} + s^1_{5,1} + s^1_{1,0} + s^1_{1,1}$
$s^0_{6,0}$ $s^1_{6,0}$	$s^0_{6,1}$ $s^1_{6,1}$	$s^0_{6,2} = s^0_{6,0} + s^0_{6,1}$ $+s^1_{6,0} + s^1_{6,1}$ $s^1_{6,2} = s^1_{6,0} + s^1_{5,1}$	$s^0_{6,3} = s^0_{6,0} + s^0_{5,1}$ $s^1_{6,3} = s^0_{6,0} + s^0_{6,1} + s^1_{6,0}$ $+s^1_{6,1} + s^1_{2,0} + s^1_{2,1}$
$s^0_{7,0}$ $s^1_{7,0}$	$s^0_{7,1}$ $s^1_{7,1}$	$s^0_{7,2} = s^0_{7,0} + s^0_{7,1}$ $+s^1_{7,0} + s^1_{7,1}$ $s^1_{7,2} = s^1_{7,0} + s^1_{6,1}$	$s^0_{7,3} = s^0_{7,0} + s^0_{6,1}$ $s^1_{7,3} = s^0_{7,0} + s^0_{7,1} + s^1_{7,0}$ $+s^1_{7,1} + s^1_{3,0} + s^1_{3,1}$
		$s^1_{8,2} = s^1_{7,1}$	$s^0_{8,3} = s^0_{7,1}$

II. A Motivating Example

We present a motivating example with $(n, k, \alpha) = (4, 2, 2)$ and $L = 8$ to show the construction of ZDR codes. Table I presents the code given in the example. Note that each data node stores $\alpha L = 16$ bits and each coded node stores 17 bits, i.e., the additional storage overhead of each coded node is one bit. For nodes 0 and 1, each of them stores $2L = 16$ information bits $s^0_{0,0}, s^0_{1,0}, \ldots, s^0_{7,0}, s^1_{0,0}, s^1_{1,0}, \ldots, s^1_{7,0}$ and $s^0_{0,1}, s^0_{1,1}, \ldots, s^0_{7,1}, s^1_{0,1}, s^1_{1,1}, \ldots, s^1_{7,1}$, respectively. Let $s^0_{i,0} = s^1_{i,0} = s^0_{i,1} = s^1_{i,1} = 0$ for $i < 0$ and $i > 7$. Node 2 stores 17 parity bits $s^0_{0,2}, s^0_{1,2}, \ldots, s^0_{7,2}$ and $s^1_{0,2}, s^1_{1,2}, \ldots, s^1_{8,2}$ that are computed as

$$s^0_{i,2} = s^0_{i,0} + s^0_{i,1} + s^1_{i,0} + s^1_{i,1} \text{ for } i = 0, 1, \ldots, 7,$$
$$s^1_{i,2} = s^1_{i,0} + s^1_{i-1,1} \text{ for } i = 0, 1, \ldots, 8.$$

Node 3 stores 17 parity bits $s^0_{0,3}, s^0_{1,3}, \ldots, s^0_{8,3}$ and $s^1_{0,3}, s^1_{1,3}, \ldots, s^1_{7,3}$ that are computed as

$$s^0_{i,3} = s^0_{i,0} + s^0_{i-1,1} \text{ for } i = 0, 1, \ldots, 8,$$
$$s^1_{i,3} = s^0_{i,0} + s^0_{i,1} + s^1_{4+i,0} + s^1_{4+i,1} \text{ for } i = 0, 1, 2, 3,$$
$$s^1_{i,3} = s^0_{i,0} + s^0_{i,1} + s^1_{i,0} + s^1_{i,1} + s^1_{i-4,0} + s^1_{i-4,1} \text{ for } i = 4, 5, 6, 7.$$

We argue that we can recover the information bits from any two nodes (i.e., the MDS property holds). From node 0 and node 1, we can directly obtain the information bits. Also, we can verify that the information bits can be obtained from node 0 or node 1 plus any of node 2 and node 3. For example, if we want to decode the information bits from node 0 and node 2, we can subtract $s^1_{i,0}$ from $s^1_{i,0} + s^1_{i-1,1}$ to get the value of $s^1_{i-1,1}$ for $i = 1, 2, \ldots, 8$, and then decode $s^0_{i,1}$ by subtracting $s^0_{i,0}, s^1_{i,0}, s^1_{i,1}$ from $s^0_{i,0} + s^0_{i,1} + s^1_{i,0} + s^1_{i,1}$ for $i = 0, 1, \ldots, 7$.

Finally, we can decode the information bits from node 2 and node 3. First, we can compute $s^0_{i,0} + s^0_{i,1}$ and $s^1_{i,0} + s^1_{i,1}$ for $i = 0, 1, \ldots, 7$ from $s^0_{0,2}, s^0_{1,2}, \ldots, s^0_{7,2}, s^1_{0,3}, s^1_{1,3}, \ldots, s^1_{7,3}$. Specifically, we can obtain $s^1_{i,0} + s^1_{i,1}$ for $i = 0, 1, 2, 3$ by $s^0_{i+4,2} + s^1_{i+4,3}$; obtain $s^0_{i,0} + s^0_{i,1}$ for $i = 0, 1, 2, 3$ by $s^0_{i,2} + (s^1_{i,0} + s^1_{i,1})$; obtain $s^1_{i,0} + s^1_{i,1}$ for $i = 4, 5, 6, 7$ by $s^1_{i,3} + (s^0_{i-4,0} + s^0_{i-4,1})$; and then obtain $s^0_{i,0} + s^0_{i,1}$ for $i = 4, 5, 6, 7$ by $s^0_{i,2} + (s^0_{i,0} + s^0_{i,1})$. We can decode $s^1_{i,0}, s^1_{i,1}$ from $s^1_{i,0} + s^1_{i,1}$ and $s^1_{i,0} + s^1_{i-1,1}$ for $i = 0, 1, \ldots, 7$ via zigzag decoding as follows. We can directly obtain $s^1_{0,0}$, and then subtract $s^1_{0,0}$ from $s^1_{0,0} + s^1_{0,1}$ to obtain $s^1_{0,1}$, which is subtracted from $s^1_{1,0} + s^1_{0,1}$ to obtain $s^1_{1,0}$, until all 16 information bits $s^1_{i,0}, s^1_{i,1}$ with $i = 0, 1, \ldots, 7$ are decoded. We can also decode $s^0_{i,0}, s^0_{i,1}$ from $s^0_{i,0} + s^0_{i,1}$ and $s^0_{i,0} + s^0_{i-1,1}$ for $i = 0, 1, \ldots, 7$ with the same method.

Our code construction also permits efficient repair upon the failure of any single node. We consider the case that node 0 fails as an example. Instead of decoding the information bits of node 0 by retrieving the bits from $k = 2$ nodes, we can recover node 0 with smaller repair bandwidth by retrieving the bits from the remaining three surviving nodes. Specifically, we recover the bits $s^1_{2\ell,0}$ for $\ell = 0, 1, 2, 3$ by

$$\begin{aligned} s^1_{0,0} &= s^1_{0,1} + (s^0_{4,2} + s^1_{4,3}), \\ s^1_{2,0} &= s^1_{2,1} + (s^0_{6,2} + s^1_{6,3}), \\ s^1_{4,0} &= s^1_{4,1} + (s^0_{0,2} + s^1_{0,3}), \\ s^1_{6,0} &= s^1_{6,1} + (s^0_{2,2} + s^1_{2,3}), \end{aligned}$$

and $s^0_{2\ell,0}$ for $\ell = 0, 1, 2, 3$ by

$$s^0_{2\ell,0} = s^0_{2\ell,1} + s^1_{2\ell,0} + s^1_{2\ell,1} + s^0_{2\ell,2},$$

and then repair the bits $s^1_{2\ell+1,0}$ and $s^0_{2\ell+1,0}$ for $\ell = 0, 1, 2, 3$ by

$$\begin{aligned} s^1_{2\ell+1,0} &= s^1_{2\ell,1} + s^1_{2\ell+1,2}, \\ s^0_{2\ell+1,0} &= s^0_{2\ell,1} + s^0_{2\ell+1,3}. \end{aligned}$$

The repair bandwidth is 24 bits, and is equal to the optimal value in Eq. (1) with $(k, d, \alpha, L) = (2, 3, 2, 8)$. On the other hand, we can repair the bits in node 1 by

$$\begin{aligned} s^1_{0,1} &= s^1_{0,0} + (s^0_{4,2} + s^1_{4,3}), \\ s^1_{2,1} &= s^1_{2,0} + (s^0_{6,2} + s^1_{6,3}), \\ s^1_{4,1} &= s^1_{4,0} + s^1_{0,0} + s^1_{0,1} + (s^0_{0,2} + s^1_{0,3}), \\ s^1_{6,1} &= s^1_{6,0} + s^1_{2,0} + s^1_{2,1} + (s^0_{2,2} + s^1_{2,3}), \\ s^0_{2\ell,1} &= s^0_{2\ell,0} + s^1_{2\ell,0} + s^1_{2\ell,1} + s^0_{2\ell,2}, \text{ for } \ell = 0, 1, 2, 3, \end{aligned}$$

and

$$\begin{aligned} s^1_{2\ell+1,1} &= s^1_{2\ell+2,0} + s^1_{2\ell+2,2} \text{ for } \ell = 0, 1, 2, 3, \\ s^0_{2\ell+1,1} &= s^0_{2\ell+2,0} + s^0_{2\ell+2,3} \text{ for } \ell = 0, 1, 2, 3. \end{aligned}$$

The repair bandwidth is also 24 bits. Furthermore, we can repair the bits in node 2 by downloading 24 bits $s^1_{i,0}, s^1_{i,1}, s^1_{i,3}$ for $i = 0, 1, \ldots, 7$. Specifically, we can repair $s^1_{i,2}$ by

$$s^1_{i,2} = s^1_{i,0} + s^1_{i-1,1} \text{ for } i = 0, 1, \ldots, 8,$$

and repair $s^0_{0,2}, s^0_{1,2}, \ldots, s^0_{7,2}$ by

$$\begin{aligned} s^0_{i,2} =& s^1_{i,3} + s^1_{i,0} + s^1_{i,1} + s^1_{i+4,0} + s^1_{i+4,1} \text{ for } i = 0,1,2,3,\\ s^0_{i,2} =& s^1_{i,3} + s^1_{i-4,0} + s^1_{i-4,1} \text{ for } i = 4,5,6,7. \end{aligned}$$

Similarly, the repair bandwidth of node 3 is also 24 bits, as we can repair $s^0_{i,3}$ by

$$s^0_{i,3} = s^0_{i,0} + s^0_{i-1,1} \text{ for } i = 0,1,\ldots,8,$$

and repair $s^1_{i,3}$ by

$$\begin{aligned} s^1_{i,3} =& s^0_{i+4,2} + s^0_{i,0} + s^0_{i,1} + s^0_{i+4,0} + s^0_{i+4,1} \text{ for } i = 0,1,2,3,\\ s^1_{i,3} =& s^0_{i,2} + s^0_{i-4,2} + s^0_{i-4,0} + s^0_{i-4,1} \text{ for } i = 4,5,6,7. \end{aligned}$$

In the following, we first present a construction of zigzag-decodable codes that have asymptotically minimum repair bandwidth for k data nodes. Then, we propose a generic transformation that can transform any zigzag-decodable codes into a new zigzag-decodable codes with asymptotically optimal repair bandwidth for any arbitrary r nodes.

III. New Zigzag-Decodable Codes with Asymptotically Minimum Repair Bandwidth for Data Nodes

In the section, we propose the explicit construction of zigzag-decodable codes with each node storing one packet ($\alpha = 1$), which achieve the asymptotically minimum repair bandwidth for repairing any data node, while preserving the zigzag decoding property.

Assume that zigzag-decodable codes have $k \geq 2$ data nodes and $r \geq 2$ coded nodes. Each data node stores one data packet and each coded node stores one coded packet. We identify the bits in a packet using the coefficients of a polynomial. We can represent a packet by a polynomial in the binary polynomial ring $\mathbb{F}_2[z]$. Let $s_0(z), s_1(z), \ldots, s_{k-1}(z) \in \mathbb{F}_2[z]$ be the k data packets with size L bits, where $s_j(z) = \sum_{i=0}^{L-1} s_{i,j} z^i$ for $j = 0,1,\ldots,k-1$. Let $s_{k+h}(z)$ be a coded packet, for $h = 0,1,\ldots,r-1$, such that it can be expressed as a linear combination of the k data packets:

$$s_{k+h}(z) = c_{h,0}(z)s_0(z) + \cdots + c_{h,k-1}(z)s_{k-1}(z),$$

with encoding coefficients $c_{h,j}(z)$ being drawn from $\mathbb{F}_2[z]$. If $c_{h,j}(z)$ is a power of z, the exponent of $c_{h,j}(z)$ denotes the number of right-shifts of the packet $s_j(z)$ in computing the coded packet $s_{k+h}(z)$. For example, the coefficients $(0, s_{0,1}, s_{1,1}, \ldots, s_{L-1,1})$ of the polynomial $zs_1(z)$ are viewed as a right-shift of the coefficients $(s_{0,1}, s_{1,1}, \ldots, s_{L-1,1})$ of polynomial $s_1(z)$.

The encoding coefficients $c_{h,0}(z), c_{h,1}(z), \ldots, c_{h,k-1}(z)$ form a vector, which we call the *encoding vector* of the associated coded packet. Due to bitwise shifting, the length of the coded packet is L plus the maximum degree of the encoding coefficients. The maximum degree of the coefficients thus provide a measure of the storage overhead.

Given the k data packets, the r coded packets are computed as

$$\begin{bmatrix} s_k(z) \\ s_{k+1}(z) \\ \vdots \\ s_{k+r-1}(z) \end{bmatrix} = \mathbf{E}_{r\times k} \cdot \begin{bmatrix} s_0(z) \\ s_1(z) \\ \vdots \\ s_{k-1}(z) \end{bmatrix},$$

where the encoding matrix is

$$\mathbf{E}_{r\times k} = [c_{h,j}(z)]^{0\leq j\leq k-1}_{0\leq h\leq r-1} = [z^{h\cdot r^j - h}]^{0\leq j\leq k-1}_{0\leq h\leq r-1}. \quad (2)$$

The codes with encoding matrix given in Eq. (2) are the proposed zigzag-decodable codes. Due the bitwise right-shift, the length of coded packet is L plus the maximum degree of the encoding coefficients. According to the encoding matrix in Eq. (2), the length of coded packet is upper bounded by $L + (r-1)r^{k-1} - r + 1$. The storage overhead is at most $(r-1)r^{k-1} - r + 1$ which is negligible if $L \gg (r-1)r^{k-1} - r + 1$.

A. Zigzag Decoding

In the remaining of the section, let ℓ be the number of coded packets involved in the decoding procedure. Consider the $\ell \times \ell$ linear system

$$[c_0(z), \ldots, c_{\ell-1}(z)]^T = \mathbf{E}(z)[s_0(z), \ldots, s_{\ell-1}(z)]^T,$$

where $s_0(z), \ldots, s_{\ell-1}(z)$ are packets with L bits, and $\mathbf{E}(z) = [z^{e_{h,j}}]^{0\leq j\leq \ell-1}_{0\leq h\leq \ell-1}$ is an $\ell \times \ell$ matrix. It is shown in [5] that we can decode $s_0(z), \ldots, s_{\ell-1}(z)$ from $c_0(z), \ldots, c_{\ell-1}(z)$ by zigzag decoding, if

$$0 < e_{h,j'} - e_{h,j} < e_{h',j'} - e_{h',j}$$

holds for all $h < h'$ and $j < j'$. The above condition is called *increasing-difference property* in [5]. For $h = 0,1,\ldots,\ell-1$, we have

$$c_h(z) = z^{e_{h,0}} s_0(z) + z^{e_{h,1}} s_1(z) + \cdots + z^{e_{h,k-1}} s_{k-1}(z).$$

Denote the *exponent matrix* of $\mathbf{E}(z)$ by $E_{\ell\times\ell} = [e_{h,j}]^{0\leq j\leq \ell-1}_{0\leq h\leq \ell-1}$. Define $w(f(z))$ as the lowest degree of terms in $f(z)$ and $\Omega(f(z))$ as the term with lowest degree in $f(z)$. The zigzag decoding algorithm is briefly given in Algorithm 1. For details, please refer to [5].

In the next theorem, we show the correctness of zigzag decoding.

Theorem 1. *If one can always find h^* and j^* in Step 2 of Algorithm 1 when $\mathcal{M}' \neq \emptyset$, then zigzag decoding can successfully recover the unknown data packets.*

Proof. Recall that $e_{h,j}$ denotes the number of right-shifts of the data packet $s_j(z)$ for the coded packet $c_h(z)$. We use $w(\eta_j(z))$ to represent the number of coefficients of $s_j(z)$ decoded in the algorithm and $\bar{s}_j(z)$ be the decoded portion of $s_j(z)$ in each iteration. When $\eta_j(z) = 0$, we set $w(\eta_j(z)) = L$.

In the first iteration, we have $w(z^{e_{h,j}}\eta_j(z)) = e_{h,j}$ and there exists at least one $h^* \in \{0,1,\ldots,\ell-1\}$ such that $w(z^{e_{h^*,j^*}}\eta_{j^*}(z)) - w(z^{e_{h^*,j}}\eta_j(z)) < 0$ for all $j \in \{0,1,\ldots,\ell-1\} \setminus \{j^*\}$ by the assumption. For each of such h^*, we have $\min\{w(z^{e_{h^*,j}}\eta_j(z)) -$

Algorithm 1 Zigzag Decoding Algorithm.

Inputs:

Encoding matrix $\mathbf{E}_{r\times k}$ in (2), $\mathcal{M}' = \{0,1,\ldots,\ell-1\}$, $\bar{s}_h(z) = 0$ for all $h \in \{0,1,\ldots,\ell-1\}$ and $\eta_j(z) = 1+z+\ldots+z^{L-1}$ for all $j \in \{0,1,\ldots,\ell-1\}$.

Outputs:

Output $\bar{s}_j(z)$ for all $j \in \{0,1,\ldots,\ell-1\}$.

1: **if** $\mathcal{M}' \neq \emptyset$ **then**
2: Find $h^* \in \{0,1,\ldots,\ell-1\}$ and $j^* \in \{0,1,\ldots,\ell-1\}$ such that $w(z^{e_{h^*,j^*}}\eta_{j^*}(z)) - w(z^{e_{h^*,j}}\eta_j(z)) < 0$ for all $j \in \{0,1,\ldots,\ell-1\} \setminus \{j^*\}$. (We can select any pair of h^* and j^* if there are more than one pair have been chosen). If no such h^* and j^* exists, exit and report decoding failure.
3: Let $\bar{s}_{j^*}(z) = \bar{s}_{j^*}(z) + \Omega(c_{h^*}(z))$.
4: **for** $h \in \{0,1,\ldots,\ell-1\}$ **do**
5: Let $c_h(z) = c_h(z) + \Omega(c_{h^*}(z))$.
6: Remove from $\eta_{j^*}(z)$ its lowest degree term. If $\eta_{j^*}(z)$ has no more terms, then let $\mathcal{M}' = \mathcal{M}' \setminus \{j^*\}$.

$w(z^{e_{h^*,j^*}}\eta_{j^*}(z))\}_{j\neq j^*, j\in\{0,1,\ldots,\ell-1\}} > 0$ and the lowest degree term of the coded packet $c_{h^*}(z)$ is equal to the lowest degree term of the data packet $s_{j^*}(z)$. Therefore, we can update $\bar{s}_{j^*}(z)$ by $\bar{s}_{j^*}(z) = \bar{s}_{j^*}(z) + \Omega(c_{h^*}(z))$, and further subtract the decoded term of $s_{j^*}(z)$ from the other $\ell-1$ coded packets, i.e., update $c_h(z)$ by $c_h(z) = c_h(z) + \Omega(c_{h^*}(z))$ for all $h \in \{0,1,\ldots,\ell-1\}$. In the iteration, we have decoded the lowest term of $s_{j^*}(z)$ and we can remove from $\eta_{j^*}(z)$ its lowest degree term.

By assumption, we can always find at least one $h^* \in \{0,1,\ldots,\ell-1\}$ in each iteration such that $w(z^{e_{h^*,j^*}}\eta_{j^*}(z)) - w(z^{e_{h^*,j}}\eta_j(z)) < 0$ for all $j \in \{0,1,\ldots,\ell-1\} \setminus \{j^*\}$. Therefore, we can decode the lowest degree term among all the unsolved terms of the data packet $s_{j^*}(z)$ from the coded packet $c_{h^*}(z)$. Specifically, we can update $\bar{s}_{j^*}(z)$ by $\bar{s}_{j^*}(z) = \bar{s}_{j^*}(z) + \Omega(c_{h^*}(z))$, and further subtract the decoded term of $s_{j^*}(z)$ from the other $\ell-1$ coded packets, i.e., update $c_h(z)$ by $c_h(z) = c_h(z) + \Omega(c_{h^*}(z))$ for all $h \in \{0,1,\ldots,\ell-1\}$. In the iteration, we have decoded the lowest term among all unsolved terms of $s_{j^*}(z)$ and we can remove from $\eta_{j^*}(z)$ its lowest degree term.

When $\eta_{j^*}(z)$ has no more terms, it means that all the coefficients of the data packet $s_{j^*}(z)$ are decoded and we remove the entry j^* from $\mathcal{M}'$. When $\mathcal{M}' = \emptyset$, it means that all ℓ data packets are decoded and the outputs $\bar{s}_j(z)$ are the decoded data packets for all $j \in \{0,1,\ldots,\ell-1\}$. Therefore, all the data bits are successfully decoded and the unknown data packets are successfully decoded by the zigzag decoding algorithm. □

Note that when the code is not zigzag decodable, Algorithm 1 exits in Step 2. If the $\ell\times\ell$ matrix satisfies the increasing-difference property, it has been proved that we can always find h^* and j^* in Step 2 [5] and it is zigzag decodable.

Consider that $k-\ell$ data packets and ℓ coded packets of the proposed new zigzag-decodable codes are used to decode the k data packets, where $0 \leq \ell \leq r$. We can first subtract $k-\ell$ data packets from each of the ℓ coded packets to obtain ℓ packets that are linear combinations of the unknown ℓ data packets with encoding matrix corresponding to $\ell\times\ell$ sub-matrix of $\mathbf{E}_{r\times k}$ in Eq. (2). By Theorem 1, we can recover ℓ unknown data packets, if we can always find h^* and j^* in Algorithm 1. The next theorem shows that the zigzag decoding condition in Theorem 1 is always satisfied.

Theorem 2. *Given $k-\ell$ data packets and ℓ coded packets of the proposed new zigzag-decodable codes with encoding matrix in Eq. (2). One can always recover the ℓ unknown data packets by zigzag decoding for $\ell = 1,2,\ldots,r$.*

Proof. We first check that the matrix consisting of the last $r-1$ rows of $\mathbf{E}_{r\times k}$ in Eq. (2) satisfies the increasing-difference property. Recall that the exponent matrix $E_{r\times k}$ of the encoding matrix in Eq. (2) is

$$E_{r\times k} = [e_{h,j}]_{0\leq h\leq r-1}^{0\leq j\leq k-1} = [hr^j - h]_{0\leq h\leq r-1}^{0\leq j\leq k-1}.$$

Let $1 \leq x < x' \leq r-1$ and $0 \leq y < y' \leq k-1$, we have that

$$\begin{aligned} e_{x',y'} - e_{x',y} =& x'r^{y'} - x' - (x'r^y - x') \\ =& x'(r^{y'} - r^y) \\ >& x(r^{y'} - r^y) \\ =& e_{x,y'} - e_{x,y} > 0. \end{aligned}$$

Thus, the matrix consisting of the last $r-1$ rows of $\mathbf{E}_{r\times k}$ satisfies the increasing-difference property and we can always recover the ℓ unknown data packets if the $\ell\times\ell$ matrix satisfies the increasing-difference property [5].

We only need to consider the $\ell\times\ell$ sub-matrix of Eq. (2) consisting of the first row and any other $\ell-1$ rows. The matrix $E_{\ell\times\ell}$ can be written as

$$E_{\ell\times\ell} = [e_{h_x,j_y}]_{0\leq x\leq \ell-1}^{0\leq y\leq \ell-1} = [h_x r^{j_y} - h_x]_{0\leq x\leq \ell-1}^{0\leq y\leq \ell-1}, \quad (3)$$

where $0 = h_0 < \ldots < h_{\ell-1} \leq r-1$, $0 \leq j_0 < \ldots < j_{\ell-1} \leq k-1$ and $1 \leq \ell \leq r$. Since $e_{h_0,j_y} = 0$, Eq. (3) does not satisfy the increasing-difference property. However, we will prove that we still can apply the zigzag decoding on Eq. (3).

After determining $\alpha_0, \alpha_1, \ldots, \alpha_{\ell-1}$ information bits of the ℓ unknown data packets by zigzag decoding, respectively, the updated lowest degree $w(z^{e_{h_x,j_y}}\eta_{j_y}(z))$ becomes

$$w(z^{e_{h_x,j_y}}\eta_{j_y}(z)) = h_x r^{j_y} - h_x + \alpha_y,$$

where $0 \leq \alpha_0, \alpha_1, \ldots, \alpha_{\ell-1}$ and $0 \leq x, y \leq \ell-1$. Let $\mathcal{S}_x$ be the set

$$\mathcal{S}_x = \arg\min_{y\in\{0,1,\ldots,\ell-1\}} \{h_x r^{j_y} - h_x + \alpha_y\}.$$

If the number of elements in $\mathcal{S}_x$, $|\mathcal{S}_x|$, is greater than or equal to 2 for all x, then the zigzag decoding algorithm fails due to the failure of Step 1. We next prove that the above failure does not occur by contradiction. The proof is similar to that given in the proofs of Lemma 1 and Theorem 2 in [5].

Assume that $|\mathcal{S}_x| \geq 2$ for all x. First, we prove that $\mathcal{S}_x$ has the following properties:

1) $\mathcal{S}_{x'} \succ \mathcal{S}_{x^*}$ for $x' < x^*$. That is, the smallest element in $\mathcal{S}_{x'}$ is larger than or equal to the largest element in $\mathcal{S}_{x^*}$.

2) $|\mathcal{S}_x \cap \mathcal{S}_{x+1}| \leq 1$, for $x = 0, 1, \ldots, \ell - 2$.
3) $\cup_{x=0,1,\ldots,\ell-1}\mathcal{S}_x$ can be partitioned into ℓ disjoint subsets: $\mathcal{S}_0 \setminus \mathcal{S}_1, \mathcal{S}_1 \setminus \mathcal{S}_2, \ldots, \mathcal{S}_{\ell-2} \setminus \mathcal{S}_{\ell-1}, \mathcal{S}_{\ell-1}$.

Consider the first property. Let $x' < x^*$ and $y' \in \mathcal{S}_{x'}, y^* \in \mathcal{S}_{x^*}$, we need to prove that $y' \geq y^*$. As $y' \in \mathcal{S}_{x'}, y^* \in \mathcal{S}_{x^*}$, for all $y \in \{0, \ldots, \ell - 1\}$, we have

$$h_{x'} r^{j_{y'}} - h_{x'} + \alpha_{y'} \leq h_{x'} r^{j_y} - h_{x'} + \alpha_y, \tag{4}$$

$$h_{x^*} r^{j_{y^*}} - h_{x^*} + \alpha_{y^*} \leq h_{x^*} r^{j_y} - h_{x^*} + \alpha_y. \tag{5}$$

Recall that the matrix consisting of the last $r - 1$ rows of $\mathbf{E}_{r\times k}$ in Eq. (2) satisfies the increasing-difference property. For $0 \neq x' < x^*$, we have

$$h_{x'}(r^{j_y} - r^{j_{y'}}) < h_{x^*}(r^{j_y} - r^{j_{y'}}), \forall y > y'.$$

When $x' = 0$, the above inequality is still hold since $0 < h_{x^*}(r^{j_y} - r^{j_{y'}}), \forall y > y'$. By Eq. (4), we have

$$\alpha_{y'} - \alpha_y \leq h_{x'}(r^{j_y} - r^{j_{y'}}), \forall y \in \{0, \ldots, \ell - 1\}.$$

We thus obtain that

$$\alpha_{y'} - \alpha_y < h_{x^*}(r^{j_y} - r^{j_{y'}}), \forall y > y',$$

which can be further written as

$$h_{x^*} r^{j_{y'}} - h_{x^*} + \alpha_{y'} < h_{x^*} r^{j_y} - h_{x^*} + \alpha_y, \forall y > y'. \tag{6}$$

Eq. (6) means that $f(y) = h_{x^*} r^{j_y} - h_{x^*} + \alpha_y$ is larger than $f(y')$ for $y > y'$. By Eq. (5), we have $f(y)$ is larger than or equal to $f(y^*)$ for all y, which means that $f(y)$ achieves the minimum value when $y = y^*$. Clearly, y^* is not larger than y', as $f(y) > f(y')$ for $y > y'$. Therefore, we obtain that $y^* \leq y'$.

The second property follows from the first one. The third property can be proved by mathematical induction. We can partition $\mathcal{S}_0 \cup \mathcal{S}_1$ into $\mathcal{S}_0 \setminus \mathcal{S}_1$ and $\mathcal{S}_1$. Assume that $\cup_{x=0,1,\ldots,m-1}\mathcal{S}_x$ can be partitioned into $\mathcal{S}_0 \setminus \mathcal{S}_1, \mathcal{S}_1 \setminus \mathcal{S}_2, \ldots, \mathcal{S}_{m-2} \setminus \mathcal{S}_{m-1}, \mathcal{S}_{m-1}$. We can partition $\cup_{x=0,1,\ldots,m}\mathcal{S}_x$ into

$$\mathcal{S}_0 \setminus (\mathcal{S}_1 \cup \mathcal{S}_m), \mathcal{S}_1 \setminus (\mathcal{S}_2 \cup \mathcal{S}_m), \ldots, \mathcal{S}_{m-2} \setminus (\mathcal{S}_{m-1} \cup \mathcal{S}_m), \mathcal{S}_m.$$

The first two properties imply that for $x = 0, 1, \ldots, m - 2$,

$$\mathcal{S}_x \setminus (\mathcal{S}_{x+1} \cup \mathcal{S}_m) = \mathcal{S}_x \setminus \mathcal{S}_{x+1}.$$

We thus have proved the third property by induction. Therefore, we have

$$\begin{aligned} |\cup_{x=0,1,\ldots,\ell-1} \mathcal{S}_x| &= \sum_{x=0}^{\ell-2} |\mathcal{S}_x \setminus \mathcal{S}_{x+1}| + |\mathcal{S}_{\ell-1}| \\ &\geq \ell - 1 + 2 = \ell + 1, \end{aligned}$$

which is a contradiction. Therefore, we can always iteratively decode the information bits by the zigzag decoding method for all selected ℓ coded packets. □

From Theorem 2, we can decode all the patterns of failures for the proposed zigzag-decodable codes by zigzag decoding method.

B. Repair Process

We present the repair algorithm for any data packet. Assume that packet f fails, where $0 \leq f \leq k-1$ and all $n-1$ surviving packets participate in repairing packet f. By the h-th row of the encoding matrix in Eq. (2), where $h = 0, 1, \ldots, r - 1$, we have

$$s_{i,k+h} = s_{i,0} + s_{i-hr+h,1} + \cdots + s_{i-hr^{k-1}+h,k-1},$$

for $i = 0, 1, \ldots, L + hr^{k-1} - h - 1$. Thus, we can repair a bit $s_{i,f}$ by

$$\begin{cases} s_{i,k+h} + \sum_{j=1}^{k-1} s_{i-hr^j+h,j} & f = 0; \\ s_{i+hr^f-h,k+h} + \sum_{j=0,j\neq f}^{k-1} s_{i+hr^f-hr^j,j} & 1 \leq f \leq k-1. \end{cases} \tag{7}$$

When we say a bit $s_{i,f}$ is repaired by packet $k + h$, it means that we access all the bits in Eq. (7) to recover $s_{i,f}$. Given integers a and b, let $(a)_b$ be the remainder of a divided by b. The repair algorithm is stated in Algorithm 2. There are some common bits between the bits downloaded by different packets. Hence, we can carefully choose the packets to repair the failed bits in order to make the number of common bits as large as possible in Algorithm 2. This is the essential reason for achieving asymptotically optimal repair bandwidth of any data packet. The next theorem shows the repair bandwidth of one data packet with Algorithm 2.

Algorithm 2 Repair procedure of one data packet failure.

1: Suppose that the packet f has failed.
2: **for** $(i)_{r^{f+1}} \in \{0, 1, \ldots, r^f - 1\}$. **do**
3: Repair $s_{i,f}$ by packet k, i.e., by Eq. (7) with $h = 0$.
4: **for** $t = 1, 2, \ldots, r - 1$ **do**
5: **for** $(i)_{r^{f+1}} \in \{tr^f, tr^f + 1, \ldots, (t + 1)r^f - 1\}$. **do**
6: Repair $s_{i,f}$ by packet $k + r - t$, i.e., by Eq. (7) with $h = r - t$.

Theorem 3. *Assume that L is a multiple of r^k. We can repair data packet f by Algorithm 2 and the repair bandwidth is*

$$nL/r - L/r^{f+1}. \tag{8}$$

Proof. We first show that packet f can be repaired by Algorithm 2. By Steps 2 and 3 in Algorithm 2, the bits $s_{i,f}$ are recovered by Eq. (7) with $h = 0$ when

$$(i)_{r^{f+1}} \in \{0, 1, \ldots, r^f - 1\} \tag{9}$$

and $i \in \{0, 1, \ldots, L - 1\}$. As i ranges from 0 to $L - 1$ and L is a multiple of r^{f+1}, $(i)_{r^{f+1}}$ is uniform distributed over $\{0, 1, \ldots, r^{f+1} - 1\}$. Thus, the total number of bits $s_{i,f}$ that are recovered by Eq. (7) with $h = 0$ is $\frac{L}{r^{f+1}} \cdot r^f = L/r$.

By Steps 4 to 6 in Algorithm 2, for $t = 1, 2, \ldots, r - 1$, the bits $s_{i,f}$ are recovered by Eq. (7) with $h = r - t$ when

$$(i)_{r^{f+1}} \in \{tr^f, tr^f + 1, \ldots, (t + 1)r^f - 1\} \tag{10}$$

and $i \in \{0, 1, \ldots, L - 1\}$. As $(i)_{r^{f+1}}$ is uniform distribution, the number of bits $s_{i,f}$ that are recovered by Eq. (7) with $h = r - t$ for $t = 1, 2, \ldots, r - 1$ is $(r - 1)L/r$. Since

$$\{tr^f, \ldots, (t + 1)r^f - 1\} \cap \{t'r^f, \ldots, (t' + 1)r^f - 1\} = \emptyset$$

for $0 \leq t \neq t' \leq r-1$, the indices of all the bits in packet f are distinct and all L bits are recovered by Algorithm 2.

Next, we calculate the repair bandwidth of packet f by Algorithm 2. First, consider the case of $1 \leq f \leq k-1$. We first download L/r information bits $s_{i,j}$ for $j = 0, 1, \ldots, f-1, f+1, \ldots, k-1$ and L/r parity bits $s_{i,k}$ with indices in Eq. (9) in Steps 2 and 3. Then, we only need to download the bits in Steps 4 to 6 which have not downloaded in Steps 2 and 3. Consider the needed information bits $s_{i+(r-t)r^f-(r-t)r^j,j}$ with $j = 0, 1, \ldots, f-1, f+1, \ldots, k-1$ in Steps 4 to 6, where i are in Eq. (10). Given i, the index of the needed bit $s_{i+(r-t)r^f-(r-t)r^j,j}$ is $i' = i + (r-t)r^f - (r-t)r^i$. If $(i')_{r^{f+1}} \in \{0, 1, \ldots, r^f - 1\}$, then we do not need to download the bit $s_{i',j}$, as it is downloaded in Steps 2 and 3. Otherwise, it should be downloaded.

We first consider the bits $s_{i',j}$ for $j = 0, 1, \ldots, f-1$. If $(i)_{r^{f+1}} = tr^f$, then there exists an integer m such that $i = mr^{f+1} + tr^f$. Thus, we have

$$\begin{aligned}(i')_{r^{f+1}} =& (i + (r-t)r^f - (r-t)r^j)_{r^{f+1}} \\ =& r^{f+1} - (r-t)r^j \text{ as } f > j-1.\end{aligned}$$

By repeating the above procedure for $(i)_{r^{f+1}} = tr^f + 1, \ldots, (t+1)r^f - 1$, we can obtain that

$$\begin{aligned}(i')_{r^{f+1}} = r^{f+1} - (r-t)r^j, \ldots, r^{f+1} - 1, \\ 0, 1, \ldots, r^f - (r-t)r^j - 1\end{aligned} \tag{11}$$

when $(i)_{r^{f+1}}$ runs from tr^f to $(t+1)r^f - 1$. Thus, the bits $s_{i',j}$ with $j = 0, 1, \ldots, f-1$ and i' in the union set of all the values in Eq. (11) are needed, and the union set of all the values in Eq. (11) is

$$\{r^{f+1} - (r-1)r^j, \ldots, r^{f+1} - 1, 0, 1, \ldots, r^f - r^j - 1\},$$

which can be rearranged as

$$\{0, 1, \ldots, r^f - r^j - 1, r^{f+1} - (r-1)r^j, \ldots, r^{f+1} - 1\}. \tag{12}$$

Since $r^f - r^i - 1 \leq r^f - 1 < r^{f+1} - (r-1)r^j$,

$$\begin{aligned}&\{0, 1, \ldots, r^f - r^j - 1, r^{f+1} - (r-1)r^i, \ldots, r^{f+1} - 1\} \\ &\backslash \{0, 1, \ldots, r^f - 1\} = \{r^{f+1} - (r-1)r^j, \ldots, r^{f+1} - 1\}.\end{aligned}$$

We only need to download $(r-1)Lr^{j-f}$ information bits $s_{i',j}$ from packets j for $j = 0, 1, \ldots, f-1$ with $(i')_{r^{f+1}} \in \{r^{f+1} - (r-1)r^j, \ldots, r^{f+1} - 1\}$ in Steps 4 to 6.

By applying the same procedure to the information bits $s_{i',j}$ with $j = f+1, f+2, \ldots, k-1$, we can prove that

$$(i')_{r^{f+1}} = 0, 1, \ldots, r^f - 1$$

when $(i)_{r^{f+1}}$ runs from tr^f to $(t+1)r^f - 1$. In this case, all needed bits have already been downloaded in Steps 2 and 3 and we thus do not need to download bits from packets j for $j = f+1, f+2, \ldots, k-1$ in Steps 4 to 6.

We can count that the total number of bits downloaded from $k+r-1$ packets to repair the data packet f is

$$kL/r + (r-1)L/r + \sum_{j=0}^{f-1}(r-1)Lr^{j-f} = nL/r - L/r^{f+1},$$

which is equal to Eq. (8). When $f = 0$, we can show that the repair bandwidth is $(n-1)L/r$ with the same argument. □

By Theorem 3, the repair bandwidth increases when f increases. When $f = 0$, the repair bandwidth is $(n-1)L/r$, which achieves the lower bound in Eq. (1). When $f = k-1$, the repair bandwidth is

$$nL/r - L/r^k < nL/r,$$

which is strictly less than $\frac{n}{n-1}$ times of the value in Eq. (1). Therefore, the repair bandwidth of any one information failure can achieve the optimal repair in Eq. (1) asymptotically when n is large enough.

IV. A Generic Transformation for Zigzag-Decodable Codes

A zigzag-decodable code contains k data packets $s_0(z), s_1(z), \ldots, s_{k-1}(z)$ and r coded packets $s_k(z), s_{k+1}(z), \ldots, s_{k+r-1}(z)$, where each node stores $\alpha = 1$ packet. We call $k+r$ packets $s_0(z), s_1(z), \ldots, s_{k+r-1}(z)$ as a codeword of the zigzag-decodable code. Note that the repair bandwidth of zigzag-decodable code is sub-optimal. In order to minimize the repair bandwidth of this code, in this section, we present a generic transformation that can convert a zigzag-decodable code into a transformed code with minimum repair bandwidth for a set of any r nodes. For ease of presentation, we assume that the chosen r nodes are the last r nodes in the following.

We first provide a brief overview of the proposed generic transformation. In the transformed code, each node contains r packets. We first generate r codewords $s_0^\ell(z), s_1^\ell(z), \ldots, s_{k+r-1}^\ell(z)$ of a zigzag-decodable code according to the $r \times k$ data packages we want to encoded in the transformed code, where ℓ is the index of the r codewords of a zigzag-decodable code and $\ell = 0, 1, \ldots, r-1$. Note that the $r \times k$ data packages are in $s_0^\ell(z), s_1^\ell(z), \ldots, s_{k-1}^\ell(z)$, where $\ell = 0, 1, \ldots, r-1$. The packets $s_k^\ell(z), s_{k+1}^\ell(z), \ldots, s_{k+r-1}^\ell(z)$ are called *intermediate packets* as they are used to compute the coded packets stored in coded nodes of the transformed code. Specifically, the r intermediate packets $s_k^\ell(z), s_{k+1}^\ell(z), \ldots, s_{k+r-1}^\ell(z)$ are computed by the multiplication of k data packets $[s_0^\ell(z), s_1^\ell(z), \ldots, s_{k-1}^\ell(z)]$ and a specific encoding matrix such as the matrix in Eq. (2). We can represent the $r(k+r)$ packets of the r codewords by an $r \times (k+r)$ array, where the entry in row ℓ and column j is $s_j^\ell(z)$ with $\ell = 0, 1, \ldots, r-1$ and $j = 0, 1, \ldots, k+r-1$. Data node j stores r data packets in column j of the array, where $j = 0, 1, \ldots, k-1$. In order to enable minimum repair bandwidth of each coded node, we should store the r coded packets in coded node which are linear combinations of the chosen intermediate packets. By Eq. (1), the minimum repair bandwidth of repairing the failed r coded packets in a coded node is $k+r-1$ packets when $d = k+r-1$. In the repair procedure, we first download k data packets in a row of the $r \times (k+r)$ array (within a codeword) from k data nodes to retrieve the r intermediate packets in the same row of the $r \times (k+r)$ array, and then download $r-1$ coded packets from other surviving $r-1$ coded

nodes to recover the failed r coded packets, which achieves the minimum repair bandwidth. Therefore, we should carefully design the linear combinations to satisfy the following two requirements: (i) we can recover the failed r coded packets from the downloaded $r-1$ coded packets and k data packets within a codeword; (ii) we can retrieve all data packets from any k nodes, i.e., the transformed codes satisfy MDS property. Note that the size of coded packet is no less than L.

Let m_j be the length of packet $s_j^\ell(z)$, where $j = 0,1,\ldots,k+r-1$. When $j=0,1,\ldots,k-1$, we have $m_j = L$ and when $j = k, k+1, \ldots, k+r-1$, we have $m_j \geq L$. Denote the packet $s_j^\ell(z)$ with length m_j by column vector $\mathbf{s}_j^\ell = [s_{0,j}^\ell, s_{1,j}^\ell, \ldots, s_{m_j-1,j}^\ell]^T$. Suppose that m_j is an even number in the following, otherwise, we add a zero at the head of $\mathbf{s}_j$, i.e.,

$$\mathbf{s}_j = [0, s_{0,j}, s_{1,j}, \ldots, s_{m_j-1,j}].$$

We also assume that L is an even integer. In order to enable optimal repair bandwidth for the last r nodes, we need to replace some entries in the $r \times r$ square matrix by a linear transformation. On the other hand, in order to make sure that the repair bandwidths of the first k nodes of the obtained transformed codes are the same as those of the underlying codes, we need to carefully design the linear transformation. The following definitions are used in the transformation.

Given a column vector $\mathbf{s}_j = [s_{0,j}, s_{1,j}, \ldots, s_{m_j-1,j}]$ with length m_j being an even number, we define $\hat{\mathbf{s}}_j$ as

$$\begin{aligned}\hat{\mathbf{s}}_j =&[s_{\frac{m_j-L}{2},j}, s_{\frac{m_j-L}{2}+1,j}, \ldots, s_{m_j-L-1,j}, s_{0,j}, s_{1,j}, \ldots,\\ & s_{\frac{m_j-L}{2}-1,j}, s_{m_j-\frac{L}{2},j}, s_{m_j-\frac{L}{2}+1,j}, \ldots, s_{m_j-1,j}, s_{m_j-L,j},\\ & s_{m_j-L+1,j}, \ldots, s_{m_j-\frac{L}{2}-1,j}]^T, \text{ when } m_j > L,\\ \hat{\mathbf{s}}_j =&[s_{m_j-\frac{L}{2},j}, s_{m_j-\frac{L}{2}+1,j}, \ldots, s_{m_j-1,j}, s_{m_j-L,j}, s_{m_j-L+1,j},\\ & \ldots, s_{m_j-\frac{L}{2}-1,j}]^T, \text{ when } m_j = L,\end{aligned}$$

and define $\bar{\mathbf{s}}_j$ as

$$\begin{aligned}\bar{\mathbf{s}}_j =&[s_{0,j}, s_{1,j}, \ldots, s_{\frac{m_j-L}{2}-1,j}, \underbrace{0,\ldots,0}_{\frac{m_j-L}{2}}, s_{m_j-L,j},\\ & s_{m_j-L+1,j}, \ldots, s_{m_j-\frac{L}{2}-1,j}, \underbrace{0,\ldots,0}_{\frac{L}{2}}]^T, \text{ when } m_j > L,\\ \bar{\mathbf{s}}_j =&[s_{m_j-L,j}, s_{m_j-L+1,j}, \ldots, s_{m_j-\frac{L}{2}-1,j}, \underbrace{0,\ldots,0}_{\frac{L}{2}}]^T, \text{when } m_j = L.\end{aligned}$$

For example, when $m_j = 6$ and $L = 4$, we have

$$\begin{aligned}\hat{\mathbf{s}}_j^1 =&[s_{1,1}^1, s_{0,1}^1, s_{4,1}^1, s_{5,1}^1, s_{2,1}^1, s_{3,1}^1]^T,\\ \bar{\mathbf{s}}_j^1 =&[s_{0,1}^1, 0, s_{2,1}^1, s_{3,1}^1, 0, 0]^T.\end{aligned}$$

The definitions of $\hat{\mathbf{s}}_j$ and $\bar{\mathbf{s}}_j$ are used in designing the linear transformation that is used to replace entries of the $r\times r$ matrix to enable optimal repair bandwidth of the last r nodes. The summation of two column vectors $\mathbf{s}_j^1, \mathbf{s}_j^2$ both with length m_j is defined by

$$\mathbf{s}_j^1 \oplus \mathbf{s}_j^2 = [s_{0,j}^1 + s_{0,j}^2, s_{1,j}^1 + s_{1,j}^2, \ldots, s_{m_j,j}^1 + s_{m_j,j}^2]^T.$$

For $j = 0,1,\ldots,k-1$, data node j stores r vectors $\mathbf{s}_j^0, \mathbf{s}_j^1, \ldots, \mathbf{s}_j^{r-1}$, where each vector has L bits and there are in total rL information bits. The rm_j parity bits stored in each coded node can be created by the following three steps.

1) Cyclic-right-shift i positions of the i-row ($i = 0,1,\ldots,r-1$) of the following $r\times r$ matrix
$$\mathbf{P}_{r\times r}^1 = \begin{bmatrix} \mathbf{s}_k^0 & \mathbf{s}_{k+1}^0 & \cdots & \mathbf{s}_{k+r-1}^0 \\ \mathbf{s}_k^1 & \mathbf{s}_{k+1}^1 & \cdots & \mathbf{s}_{k+r-1}^1 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{s}_k^{r-1} & \mathbf{s}_{k+1}^{r-1} & \cdots & \mathbf{s}_{k+r-1}^{r-1} \end{bmatrix}$$
to obtain the matrix
$$\mathbf{P}_{r\times r}^2 = \begin{bmatrix} \mathbf{s}_k^0 & \mathbf{s}_{k+1}^0 & \mathbf{s}_{k+2}^0 & \cdots & \mathbf{s}_{k+r-2}^0 & \mathbf{s}_{k+r-1}^0 \\ \mathbf{s}_{k+r-1}^1 & \mathbf{s}_k^1 & \mathbf{s}_{k+1}^1 & \cdots & \mathbf{s}_{k+r-3}^1 & \mathbf{s}_{k+r-2}^1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{s}_{k+1}^{r-1} & \mathbf{s}_{k+2}^{r-1} & \mathbf{s}_{k+3}^{r-1} & \cdots & \mathbf{s}_{k+r-1}^{r-1} & \mathbf{s}_k^{r-1} \end{bmatrix}.$$
2) Recall that $(a)_b$ is the remainder of a divided by b. For $i,j \in \{0,1,\ldots,r-1\}$, the entry in i row and j column of the matrix $\mathbf{P}_{r\times r}^2$ is $\mathbf{s}_{k+(r-i+j)_r}^i$. When $i+j < r-1$, replace the entry in i row and j column of the matrix $\mathbf{P}_{r\times r}^2$ by $\mathbf{s}_{k+(r-i+j)_r}^i \oplus \mathbf{s}_{k+(r-i+j)_r}^{r-1-j}$. When $i+j > r-1$, replace the entry in i row and j column of the matrix $\mathbf{P}_{r\times r}^2$ by $\mathbf{s}_{k+(r-i+j)_r}^i \oplus \hat{\mathbf{s}}_{k+(r-i+j)_r}^{r-1-j} \oplus \bar{\mathbf{s}}_{k+(r-i+j)_r}^{r-1-j}$. The resulting matrix $\mathbf{P}_{r\times r}^3$ is given in Eq. (13).
3) The rm_j parity bits $\mathbf{p}_j^0, \mathbf{p}_j^1, \ldots, \mathbf{p}_j^{r-1}$ stored in coded node j are the r entries in column j of the matrix $\mathbf{P}_{r\times r}^3$ for $j = 0,1,\ldots,r-1$.

Note that the storage overhead of the transformed codes is the same as that of the original zigzag-decodable codes.

Given columns j_1 and j_2 ($j_2 > j_1 \geq 0$) of the matrix $\mathbf{P}_{r\times r}^3$ in Eq. (13), the entry in row $r-1-j_1$ and column j_2 is $\mathbf{s}_{k+(1+j_1+j_2)_r}^{r-(1+j_2)} \oplus \hat{\mathbf{s}}_{k+(1+j_1+j_2)_r}^{r-(1+j_1)} \oplus \bar{\mathbf{s}}_{k+(1+j_1+j_2)_r}^{r-(1+j_1)}$, and the entry in row $r-1-j_2$ and column j_1 is $\mathbf{s}_{k+(1+j_1+j_2)_r}^{r-(1+j_2)} \oplus \mathbf{s}_{k+(1+j_1+j_2)_r}^{r-(1+j_1)}$. The next lemma shows that we can obtain $\mathbf{s}_{k+(1+j_1+j_2)_r}^{r-(1+j_2)}$ and $\mathbf{s}_{k+(1+j_1+j_2)_r}^{r-(1+j_1)}$ from the two entries.

Lemma 4. *The following statements are valid.*
1) $\mathbf{s}_{k+\ell}^i$ *and* $\mathbf{s}_{k+\ell}^j$ *from* $\mathbf{s}_{k+\ell}^i \oplus \mathbf{s}_{k+\ell}^j$ *and* $\mathbf{s}_{k+\ell}^i \oplus \hat{\mathbf{s}}_{k+\ell}^j \oplus \bar{\mathbf{s}}_{k+\ell}^j$;
2) $\mathbf{s}_{k+\ell}^i \oplus \mathbf{s}_{k+\ell}^j$ *from* $\mathbf{s}_{k+\ell}^j$ *and* $\mathbf{s}_{k+\ell}^i \oplus \hat{\mathbf{s}}_{k+\ell}^j \oplus \bar{\mathbf{s}}_{k+\ell}^j$;
3) $\mathbf{s}_{k+\ell}^i \oplus \hat{\mathbf{s}}_{k+\ell}^j \oplus \bar{\mathbf{s}}_{k+\ell}^j$ *from* $\mathbf{s}_{k+\ell}^j$ *and* $\mathbf{s}_{k+\ell}^i \oplus \mathbf{s}_{k+\ell}^j$;
4) $\mathbf{s}_{k+\ell}^j$ *from* $\hat{\mathbf{s}}_{k+\ell}^j \oplus \bar{\mathbf{s}}_{k+\ell}^j$.

Proof. Consider the first statement. From $\mathbf{s}_{k+\ell}^i \oplus \mathbf{s}_{k+\ell}^j$ and $\mathbf{s}_{k+\ell}^i \oplus \hat{\mathbf{s}}_{k+\ell}^j \oplus \bar{\mathbf{s}}_{k+\ell}^j$, we can obtain

$$\begin{aligned}& s_{t,k+\ell}^i + s_{t,k+\ell}^j \text{ for } 0 \leq t \leq m_{k+\ell}-1,\\ & s_{t,k+\ell}^i + s_{\frac{m_{k+\ell}-L}{2}-t,k+\ell}^j + s_{t,k+\ell}^j \text{ for } 0 \leq t \leq \frac{m_{k+\ell}-L}{2}-1,\\ & s_{t,k+\ell}^i + s_{t-\frac{m_{k+\ell}-L}{2},k+\ell}^j \text{ for } \frac{m_{k+\ell}-L}{2} \leq t \leq m_{k+\ell}-L-1,\\ & s_{t,k+\ell}^i + s_{t+\frac{m_{k+\ell}-L}{2},k+\ell}^j + s_{t,k+\ell}^j \text{ for } m_{k+\ell}-L \leq t \leq m_{k+\ell}-\frac{L}{2}-1,\\ & s_{t,k+\ell}^i + s_{t+\frac{L}{2},k+\ell}^j \text{ for } m_{k+\ell}-\frac{L}{2} \leq t \leq m_{k+\ell}-1.\end{aligned}$$

We can first compute $s_{\frac{m_{k+\ell}-L}{2}-t,k+\ell}^j$ for $0 \leq t \leq \frac{m_{k+\ell}-L}{2}-1$ by

$$(s_{t,k+\ell}^i + s_{t,k+\ell}^j) + (s_{t,k+\ell}^i + s_{\frac{m_{k+\ell}-L}{2}-t,k+\ell}^j + s_{t,k+\ell}^j),$$

$$\begin{bmatrix} \mathbf{s}_k^0 \oplus \mathbf{s}_k^{r-1} & \mathbf{s}_{k+1}^0 \oplus \mathbf{s}_{k+1}^{r-2} & \cdots & \mathbf{s}_{k+r-2}^0 \oplus \mathbf{s}_{k+r-2}^1 & \mathbf{s}_{k+r-1}^0 \\ \mathbf{s}_{k+r-1}^1 \oplus \mathbf{s}_{k+r-1}^{r-1} & \mathbf{s}_k^1 \oplus \mathbf{s}_k^{r-2} & \cdots & \mathbf{s}_{k+r-3}^1 & \mathbf{s}_{k+r-2}^0 \oplus \hat{\mathbf{s}}_{k+r-2}^1 \oplus \bar{\mathbf{s}}_{k+r-2}^1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{s}_{k+2}^{r-2} \oplus \mathbf{s}_{k+2}^{r-1} & \mathbf{s}_{k+3}^{r-2} & \cdots & \mathbf{s}_k^1 \oplus \hat{\mathbf{s}}_k^{r-2} \oplus \bar{\mathbf{s}}_k^{r-2} & \mathbf{s}_{k+1}^0 \oplus \hat{\mathbf{s}}_{k+1}^{r-2} \oplus \bar{\mathbf{s}}_{k+1}^{r-2} \\ \mathbf{s}_{k+1}^{r-1} & \mathbf{s}_{k+2}^{r-2} \oplus \hat{\mathbf{s}}_{k+2}^{r-1} \oplus \bar{\mathbf{s}}_{k+2}^{r-1} & \cdots & \mathbf{s}_{k+r-1}^1 \oplus \hat{\mathbf{s}}_{k+r-1}^{r-1} \oplus \bar{\mathbf{s}}_{k+r-1}^{r-1} & \mathbf{s}_k^0 \oplus \hat{\mathbf{s}}_k^{r-1} \oplus \bar{\mathbf{s}}_k^{r-1} \end{bmatrix}. \quad (13)$$

and compute $s^i_{\frac{m_{k+\ell}-L}{2}-t,k+\ell}$ for $0 \le t \le \frac{m_{k+\ell}-L}{2}-1$ by

$$s^j_{\frac{m_{k+\ell}-L}{2}-t,k+\ell} + (s^i_{\frac{m_{k+\ell}-L}{2}-t,k+\ell} + s^j_{\frac{m_{k+\ell}-L}{2}-t,k+\ell}),$$

Let $t' = \frac{m_{k+\ell}-L}{2} - t$, we can obtain that

$$t' = 1, 2, \ldots, \frac{m_{k+\ell}-L}{2},$$

when $t = 0, 1, \ldots, \frac{m_{k+\ell}-L}{2}-1$. That is to say, we have already computed the bits $s^i_{t,k+\ell}$ and $s^j_{t,k+\ell}$ for $t = 1, 2, \ldots, \frac{m_{k+\ell}-L}{2}$. We can compute $s^j_{0,k+\ell}$ by $s^i_{\frac{m_{k+\ell}-L}{2},k+\ell} + (s^i_{\frac{m_{k+\ell}-L}{2},k+\ell} + s^j_{0,k+\ell})$, and compute $s^i_{0,k+\ell}$ by $s^j_{0,k+\ell} + (s^i_{0,k+\ell} + s^j_{0,k+\ell})$. Then, we compute $s^i_{t,k+\ell}$ for $\frac{m_{k+\ell}-L}{2} \le t \le m_{k+\ell}-L-1$ by

$$s^j_{t-\frac{m_{k+\ell}-L}{2},k+\ell} + (s^i_{t,k+\ell} + s^j_{t-\frac{m_{k+\ell}-L}{2},k+\ell}),$$

and $s^j_{t,k+\ell}$ for $\frac{m_{k+\ell}-L}{2} \le t \le m_{k+\ell}-L-1$ by

$$s^i_{t,k+\ell} + (s^i_{t,k+\ell} + s^j_{t,k+\ell}).$$

The other bits of $\mathbf{s}^i_{k+\ell}$ and $\mathbf{s}^j_{k+\ell}$ can be computed by repeating the above procedure for $m_{k+\ell}-L \le t \le m_{k+\ell}$. We can compute $\mathbf{s}^i_{k+\ell}$ and $\mathbf{s}^j_{k+\ell}$ with $\frac{5}{2}m_{k+\ell}$ XORs involved.

The other three statements can be proved similarly. □

In the following, we show that the transformed codes also satisfy the MDS property, if the original zigzag-decodable codes satisfy the MDS property.

Theorem 5. *The transformed codes satisfy the MDS property if the zigzag-decodable codes satisfy the MDS property.*

Proof. The transformed codes satisfy the MDS property if any k out of n nodes can reconstruct all the information bits. It is equivalent to show that the k data nodes can be reconstructed from any t data nodes and any $k-t$ coded nodes, where $\max\{0, k-r\} \le t \le k$. When $t = k$, we can obtain the k data nodes directly.

In the following, we consider the case of $t < k$. Suppose that data node $i_1, i_2, \ldots, i_t$ and coded nodes $j_1, j_2, \ldots, j_{k-t}$ are selected with $0 \le i_1 < \ldots < i_t \le k-1$ and $0 \le j_1 < \ldots < j_{k-t} \le r-1$. The entry in row $r-1-j_1$ and column j_2 of the matrix $\mathbf{P}^3_{r\times r}$ is $\mathbf{s}^{r-(1+j_2)}_{k+(1+j_1+j_2)_r} \oplus \hat{\mathbf{s}}^{r-(1+j_1)}_{k+(1+j_1+j_2)_r} \oplus \bar{\mathbf{s}}^{(r-(1+j_1))*}_{k+(1+j_1+j_2)_r}$; while the entry in row $r-1-j_2$ and column j_1 is $\mathbf{s}^{r-(1+j_2)}_{k+(1+j_1+j_2)_r} \oplus \mathbf{s}^{r-(1+j_1)}_{k+(1+j_1+j_2)_r}$. By Lemma 4, we can obtain $\mathbf{s}^{r-(1+j_2)}_{k+(1+j_1+j_2)_r}$ and $\mathbf{s}^{r-(1+j_1)}_{k+(1+j_1+j_2)_r}$ from the above two entries. Similarly, we can obtain $\mathbf{s}^{r-(1+j_\ell)}_{k+(1+j_1+j_\ell)_r}$ and $\mathbf{s}^{r-(1+j_1)}_{k+(1+j_1+j_\ell)_r}$ from the entries in row $r-1-j_1$ column j_ℓ and row $r-1-j_\ell$ column j_1, for $\ell = 2, 3, \ldots, k-t$. Recall that the entry in row $r-1-j_1$ and column j_1 is $\mathbf{s}^{r-(1+j_1)}_{k+(1+2j_1)_r}$. We obtain

$$\{\mathbf{s}^{r-(1+j_1)}_{k+(1+2j_1)_r}, \mathbf{s}^{r-(1+j_1)}_{k+(1+j_1+j_2)_r}, \ldots, \mathbf{s}^{r-(1+j_1)}_{k+(1+j_1+j_{k-t})_r}\},$$

together with the data vectors in row $r-1-j_1$ and data nodes $i_1, i_2, \ldots, i_t$, we can compute all other data vectors in row $r-1-j_1$ according to the MDS property of the zigzag-decodable codes.

By the same argument, we can recover all data vectors in rows $r-1-j_2, \ldots, r-1-j_{k-t}$. Once the vectors in rows $r-1-j_2, \ldots, r-1-j_{k-t}$ are known, we can recover all other vectors similarly, followed by solving the unknown packets according to the MDS property of the $(k+r, k)$ zigzag-decodable codes. □

The idea behind the proof of Theorem 5 is similar to the proof of Theorem 1 in [10] and Theorem 1 in [24]. The technical difference is that our linear transformation is designed for zigzag-decodable code but the transformations in [10] and [24] are designed for non-binary MDS codes and binary MDS codes, respectively. The next theorem shows that the r coded nodes of the transformed code are with the optimal repair bandwidth.

Theorem 6. *The r coded nodes of the transformed codes are with the optimal repair bandwidth.*

Proof. For $j = 0, 1, \ldots, r-1$, we show that we can recover coded node j by accessing k vectors $\mathbf{s}_0^{r-1-j}, \mathbf{s}_1^{r-1-j}, \ldots, \mathbf{s}_{k-1}^{r-1-j}$ and $r-1$ entries in row $r-1-j$ of the matrix $\mathbf{P}^3_{r\times r}$ except the entry in the failed column j.

By accessing $\mathbf{s}_0^{r-1-j}, \mathbf{s}_1^{r-1-j}, \ldots, \mathbf{s}_{k-1}^{r-1-j}$, we can compute $\mathbf{s}_k^{r-1-j}, \mathbf{s}_{k+1}^{r-1-j}, \ldots, \mathbf{s}_{k+r-1}^{r-1-j}$. With the obtained $\mathbf{s}^{r-1-j}_{k+(1+2j)_r}$ and the accessed $r-1$ entries in row j of the matrix $\mathbf{P}^3_{r\times r}$ except the entry in the failed column j, we can compute all r vectors in parity column j by Lemma 4. □

We show in the next theorem that the repair bandwidth of both the original zigzag-decoded codes and the transformed codes is the same if the repair process of a data node satisfies some condition.

Theorem 7. *In the $(k+r, k)$ zigzag-decodable codes, suppose that we can repair node f by downloading the bits $s_{i,j}$ for all $i \in S_j$ and $j = 0, 1, \ldots, f-1, f+1, \ldots, k+r-1$, where $0 \le f \le k-1$ and S_j denotes the set of indices of the downloaded bits from node j. Then, we can repair node f of the transformed codes by downloading $s^\ell_{i,j}$ for all $i \in S_j$,*

$\ell = 0, 1, \ldots, r-1$, $j = 0, 1, \ldots, f-1, f+1, \ldots, k-1$, *and the following bits*

$$\begin{bmatrix} p_{i,0}^0, \forall i \in S_k & p_{i,1}^0, \forall i \in S_{k+1} & \cdots & p_{i,r-1}^0, \forall i \in S_{k+r-1} \\ p_{i,0}^1, \forall i \in S_{k+r-1} & p_{i,1}^1, \forall i \in S_k & \cdots & p_{i,r-1}^1, \forall i \in S_{k+r-2} \\ \vdots & \vdots & \ddots & \vdots \\ p_{i,0}^{r-2}, \forall i \in S_{k+2} & p_{i,1}^{r-2}, \forall i \in S_{k+3} & \cdots & p_{i,r-1}^{r-2}, \forall i \in S_{k+1} \\ p_{i,0}^{r-1}, \forall i \in S_{k+1} & p_{i,1}^{r-1}, \forall i \in S_{k+2} & \cdots & p_{i,r-1}^{r-1}, \forall i \in S_k \end{bmatrix}$$

from nodes $k, k+1, \ldots, k+r-1$, *if*

$$i + \frac{L}{2} \in S_{k+j}, \forall i \in S_{k+j} \text{ and } m_j - L \leq i \leq m_j - \frac{L}{2} - 1,$$

where $j = 0, 1, \ldots, r-1$.

Proof. Given $j = 0, 1, \ldots, r-1$, for $i \in S_{k+j}$ and $m_j - L \leq i \leq m_j - \frac{L}{2} - 1$, the bits downloaded from $p_{i,0}^0, \forall i \in S_k$ and $p_{i,r-1}^{r-1}, \forall i \in S_k$ are

$$\begin{aligned} p_{i,0}^0 =& s_{i,k}^0 + s_{i,k}^{r-1}, \\ p_{i+\frac{L}{2},0}^0 =& s_{i+\frac{L}{2},k}^0 + s_{i+\frac{L}{2},k}^{r-1}, \end{aligned}$$

and

$$\begin{aligned} p_{i,r-1}^{r-1} =& s_{i,k}^0 + s_{i+\frac{L}{2},k}^{r-1}, \\ p_{i+\frac{L}{2},r-1}^{r-1} =& s_{i+\frac{L}{2},k}^0 + s_{i+\frac{L}{2},k}^{r-1} + s_{i,k}^{r-1}, \end{aligned}$$

respectively. We can first compute $s_{i,k}^{r-1}$ by

$$s_{i,k}^{r-1} = p_{i+\frac{L}{2},0}^0 + p_{i+\frac{L}{2},r-1}^{r-1},$$

and then compute $s_{i,k}^0$ by $s_{i,k}^0 = p_{i,0}^0 + s_{i,k}^{r-1}$. Finally, we can compute $s_{i+\frac{L}{2},k}^{r-1}$ and $s_{i+\frac{L}{2},k}^0$ by

$$s_{i+\frac{L}{2},k}^{r-1} = s_{i,k}^0 + p_{i,r-1}^{r-1},$$

and

$$s_{i+\frac{L}{2},k}^0 = s_{i+\frac{L}{2},k}^{r-1} + p_{i+\frac{L}{2},0}^0,$$

respectively. Similarly, we can compute

$$\begin{aligned} & s_{i,k+(1+j_1+j_2)_r}^{r-1-j_2}, s_{i,k+(1+j_1+j_2)_r}^{r-1-j_1}, \\ & s_{i+\frac{L}{2},k+(1+j_1+j_2)_r}^{r-1-j_2}, s_{i+\frac{L}{2},k+(1+j_1+j_2)_r}^{r-1-j_1} \end{aligned}$$

from

$$\begin{aligned} p_{i,j_2}^{r-1-j_1} =& s_{i,k+(1+j_1+j_2)_r}^{r-1-j_2} + s_{i+\frac{L}{2},k+(1+j_1+j_2)_r}^{r-1-j_1}, \\ p_{i+\frac{L}{2},j_2}^{r-1-j_1} =& s_{i+\frac{L}{2},k+(1+j_1+j_2)_r}^{r-1-j_2} + s_{i+\frac{L}{2},k+(1+j_1+j_2)_r}^{r-1-j_1} + \\ & s_{i,k+(1+j_1+j_2)_r}^{r-1-j_1}, \\ p_{i,j_1}^{r-1-j_2} =& s_{i,k+(1+j_1+j_2)_r}^{r-1-j_2} + s_{i,k+(1+j_1+j_2)_r}^{r-1-j_1}, \\ p_{i+\frac{L}{2},j_1}^{r-1-j_2} =& s_{i+\frac{L}{2},k+(1+j_1+j_2)_r}^{r-1-j_2} + s_{i+\frac{L}{2},k+(1+j_1+j_2)_r}^{r-1-j_1}, \end{aligned}$$

for $j_2 > j_1 \geq 0$, $i \in S_{k+(1+j_1+j_2)_r}$ and $m_{(1+j_1+j_2)_r} - L \leq i \leq m_{(1+j_1+j_2)_r} - \frac{L}{2} - 1$. We thus obtain the bits $s_{i,j}^\ell$ for all $i \in S_j$, $j = k, k+1, \ldots, k+r-1$ and $\ell = 0, 1, \ldots, r-1$. Recall that we can recover $s_{0,f}, s_{1,f}, \ldots, s_{L-1,f}$ by downloading the bits $s_{i,j}$ for all $i \in S_j$ and $j = 0, 1, \ldots, f-1, f+1, \ldots, k+r-1$. Therefore, we obtain the bits $s_{i,j}^\ell$ for all $i \in S_j$, $j = 0, 1, \ldots, f-1, f+1, \ldots, k+r-1$ and $\ell = 0, 1, \ldots, r-1$, and all the bits in node f can be recovered. □

In the repair process of the transformed codes, the bits downloaded from nodes $k, k+1, \ldots, k+r-1$ are linear combinations of the needed bits in repairing a data node. We need to solve the needed bits from the downloaded bits of nodes $k, k+1, \ldots, k+r-1$ in order to achieve the sam repair bandwidth of the original zigzag-decodable codes. In Theorem 7, we give the condition under which we can solve the needed bits from the downloaded bits of nodes $k, k+1, \ldots, k+r-1$. Although our transformation is designed for zigzag-decodable codes, it is also applicable for MDS array codes. We only need to make the length of both data packets and coded packets be the same when apply the transformation for MDS array codes.

V. Constructions of Zigzag-Decodable Reconstruction Codes

In Section III, we present a construction of zigzag-decodable codes that can achieve asymptotically optimal repair bandwidth for repairing any data node. In this section, we propose two explicit constructions of zigzag-decodable reconstruction (ZDR) codes that can achieve asymptotically optimal repair bandwidth for not only any data node but also coded node. The first constructed ZDR code is obtained by directly applying the transformation in Section IV for the code in Section III. The second constructed ZDR code is obtained by recursively applying the transformation in Section IV for any zigzag-decodable code.

A. The First Construction

The proposed ZDR codes contain $k \geq 2$ data nodes and $r \geq 2$ coded nodes. Each data node stores r data packets and each coded node stores r coded packets. For easier understanding, we divide the encoding process into two steps. First, we generate r instances of zigzag-decodable codes with the encoding matrix in Eq. (2). Second, convert the obtained zigzag-decodable codes with the encoding matrix in Eq. (2) into the transformed codes by the transformation given in Section IV.

According to Eq. (2) and the transformation, the length of the vector $\mathbf{p}_{k+j}^\ell$ is $L + jr^{k-1} - j$. The proposed ZDR codes is denoted by $\mathcal{ZDR}_1(k, r)$. To differential vectors $\mathbf{p}_{k+j}^\ell$ and $\mathbf{s}_{k+j}^\ell$, we call $\mathbf{s}_{k+j}^\ell$ as *intermediate vector* and $\mathbf{p}_{k+j}^\ell$ as *coded vector*. By the transformation in Section IV, $\mathcal{ZDR}_1(k, r)$ has the same storage overhead as that of the zigzag-decodable codes, the storage overhead of $\mathcal{ZDR}_1(k, r)$ is at most $(r-1)r^{k-1} - r + 1$ that is negligible when $L \gg (r-1)r^{k-1} - r + 1$.

For example, when $(k, r) = (2, 2)$, we have four data packets $s_0^0(z), s_0^1(z), s_1^0(z), s_1^1(z)$ and the encoding matrix is

$$\begin{bmatrix} 1 & 1 \\ z & z^2 \end{bmatrix}.$$

We can first compute four packets as

$$\begin{bmatrix} s_2^\ell(z) \\ s_3^\ell(z) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ z & z^2 \end{bmatrix} \cdot \begin{bmatrix} s_0^\ell(z) \\ s_1^\ell(z) \end{bmatrix},$$

where $\ell = 0, 1$. Then, we can compute four coded vectors stored in two coded nodes as

$$\begin{aligned}\mathbf{p}_2^0 =& \mathbf{s}_2^0 \oplus \mathbf{s}_2^1,\\ \mathbf{p}_3^0 =& \mathbf{s}_3^0,\\ \mathbf{p}_2^1 =& \mathbf{s}_3^1,\\ \mathbf{p}_3^1 =& \mathbf{s}_2^0 \oplus \hat{\mathbf{s}}_2^1 \oplus \bar{\mathbf{s}}_2^1.\end{aligned}$$

The example given in Section II is exactly the example of the proposed codes with $(k, r) = (2, 2)$.

According to Theorem 5, we can decode all data packets of $\mathcal{ZDR}_1(k, r)$ by zigzag decoding from any k nodes. Furthermore, according to Theorem 6, we can repair a coded node f by downloading all $k + r - 1$ packets in row f and the repair bandwidth is optimal. Recall that the repair algorithm of data node f of the original zigzag-decodable codes is given in Algorithm 2, and the repair bandwidth of each data node is asymptotically optimal by Theorem 3. In Algorithm 2, if a bit $s_{i,k+j}$ in node $k + j$ with $j = 0, 1, \ldots, r - 1$ is downloaded to recover node f, then $s_{i+L/2,k+j}$ is also downloaded in the repair process, as L is a multiple of k^r. Therefore, the condition given in Theorem 7 is satisfied, and we can recover data node f of $\mathcal{ZDR}_1(k, r)$ with asymptotically optimal repair bandwidth.

B. The Second Construction

The second construction of ZDR codes is designed by recursively applying the transformation for a zigzag-decodable code. In the following, we take an example of zigzag-decodable codes with the encoding matrix given as

$$\mathbf{E}_{r\times k} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & z & \cdots & z^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & z^{r-1} & \cdots & z^{(r-1)(k-1)} \end{bmatrix} \tag{14}$$

to show the construction.

By applying the transformation for the r coded nodes of the zigzag-decodable code with the encoding matrix given in Eq. (14), according to Theorem 5, we can obtain a transformed code that can be decoded by zigzag decoding from any k nodes and, according to Theorem 6, has minimum repair bandwidth for r coded nodes. We can also apply the transformation for the chosen r data nodes of the above resulting codes such that, by Theorem 5, the newly transformed code has the zigzag decoding property. Furthermore, by Theorem 6, it has minimum repair bandwidth for both the chosen r data nodes and r coded nodes. By applying the transformation recursively for $\lceil \frac{k+r}{r} \rceil$ times, we can obtain the transformed ZDR code, denoted by $\mathcal{ZDR}_2(k, r)$, that has minimum repair bandwidth for all nodes. Note that in $\mathcal{ZDR}_2(k, r)$, $\alpha = r^{\lceil \frac{k+r}{r} \rceil}$. The storage overhead of the original zigzag-decodable codes is $(r-1)(k-1)$. As $\mathcal{ZDR}_2(k, r)$ has the same storage overhead as that of the original zigzag-decodable codes, the storage overhead of $\mathcal{ZDR}_2(k, r)$ is also $(r-1)(k-1)$ that is negligible when $L \gg (r - 1)(k - 1)$.

We present an example of $(k, r) = (2, 2)$ to illustrate the construction process. First, we choose a zigzag-decodable code with $(k, r) = (2, 2)$. Given two data vectors $\mathbf{s}_0, \mathbf{s}_1$, we compute two coded vectors $\mathbf{s}_2, \mathbf{s}_3$ of the zigzag-decodable code such that any two vectors can retrieve the two data vectors. Second, we apply the transformation for the zigzag-decodable code to obtain the transformed code in Table II. Finally, we obtain the $\mathcal{ZDR}_2(2, 2)$ by applying the transformation for the code in Table II, and $\mathcal{ZDR}_2(2, 2)$ is shown in Table III.

TABLE II: Codes with $(k, r) = (2, 2)$ by applying the transformation for the two coded nodes.

Node 0	Node 1	Node 2	Node 3
$\mathbf{s}_0^0$	$\mathbf{s}_1^0$	$\mathbf{s}_2^0 \oplus \mathbf{s}_2^1$	$\mathbf{s}_3^0$
$\mathbf{s}_0^1$	$\mathbf{s}_1^1$	$\mathbf{s}_3^1$	$\mathbf{s}_2^0 \oplus \hat{\mathbf{s}}_2^1 \oplus \bar{\mathbf{s}}_2^1$

TABLE III: Example of $\mathcal{ZDR}_2(2, 2)$.

Node 0	Node 1	Node 2	Node 3
$\mathbf{s}_0^0 \oplus \mathbf{s}_0^2$	$\mathbf{s}_1^0$	$\mathbf{s}_2^0 \oplus \mathbf{s}_2^1$	$\mathbf{s}_3^0$
$\mathbf{s}_0^1 \oplus \mathbf{s}_0^3$	$\mathbf{s}_1^1$	$\mathbf{s}_3^1$	$\mathbf{s}_2^0 \oplus \hat{\mathbf{s}}_2^1 \oplus \bar{\mathbf{s}}_2^1$
$\mathbf{s}_1^2$	$\mathbf{s}_0^0 \oplus \hat{\mathbf{s}}_0^2 \oplus \bar{\mathbf{s}}_0^2$	$\mathbf{s}_2^2 \oplus \mathbf{s}_2^3$	$\mathbf{s}_3^2$
$\mathbf{s}_1^3$	$\mathbf{s}_0^1 \oplus \hat{\mathbf{s}}_0^3 \oplus \bar{\mathbf{s}}_0^3$	$\mathbf{s}_3^3$	$\mathbf{s}_2^2 \oplus \hat{\mathbf{s}}_2^3 \oplus \bar{\mathbf{s}}_2^3$

We can recover all the data vectors from any two nodes in this example. From the first two nodes, by Lemma 4, we can obtain $\mathbf{s}_0^0$, $\mathbf{s}_0^1$, $\mathbf{s}_0^2$ and $\mathbf{s}_0^3$ from

$$\mathbf{s}_0^0 \oplus \mathbf{s}_0^2, \mathbf{s}_0^1 \oplus \mathbf{s}_0^3, \mathbf{s}_0^0 \oplus \hat{\mathbf{s}}_0^2 \oplus \bar{\mathbf{s}}_0^2, \mathbf{s}_0^1 \oplus \hat{\mathbf{s}}_0^3 \oplus \bar{\mathbf{s}}_0^3.$$

Similarly, by Lemma 4, we can compute all the data vectors from nodes 2 and 3. Now we consider the decoding from nodes 1 and 2. We can first compute $\mathbf{s}_0^1$ and $\mathbf{s}_2^1$ from two vectors $\mathbf{s}_1^1$ and $\mathbf{s}_3^1$, as the original zigzag-decodable code has the MDS property. Then, we can compute $\mathbf{s}_2^0$ by $\mathbf{s}_2^1 \oplus (\mathbf{s}_2^0 \oplus \mathbf{s}_2^1)$, together with $\mathbf{s}_1^0$, we can compute $\mathbf{s}_0^0$ and $\mathbf{s}_3^0$. After computing $\hat{\mathbf{s}}_0^3 \oplus \bar{\mathbf{s}}_0^3$ by subtracting $\mathbf{s}_0^1$ from $\mathbf{s}_0^1 \oplus \hat{\mathbf{s}}_0^3 \oplus \bar{\mathbf{s}}_0^3$, by Lemma 4, we can decode $\mathbf{s}_0^3$ from $\hat{\mathbf{s}}_0^3 \oplus \bar{\mathbf{s}}_0^3$. We thus can compute $\mathbf{s}_1^3$ and $\mathbf{s}_2^3$ from $\mathbf{s}_0^3$ and $\mathbf{s}_3^3$. The two vectors $\mathbf{s}_0^2$ and $\mathbf{s}_1^2$ can be computed similarly. With the same argument, we can decode all the data vectors for other cases.

Next, we demonstrate that we can repair any one node by downloading two vectors from each of the other three nodes, where the repair bandwidth of any one node is optimal. Assume node 0 fails, we can recover the four vectors in node 0 by downloading the following six vectors

$$\mathbf{s}_0^0 \oplus \hat{\mathbf{s}}_0^2 \oplus \bar{\mathbf{s}}_0^2, \mathbf{s}_0^1 \oplus \hat{\mathbf{s}}_0^3 \oplus \bar{\mathbf{s}}_0^3, \mathbf{s}_2^2 \oplus \mathbf{s}_2^3, \mathbf{s}_3^3, \mathbf{s}_3^2, \mathbf{s}_2^2 \oplus \hat{\mathbf{s}}_2^3 \oplus \bar{\mathbf{s}}_2^3.$$

First, by Lemma 4, we compute $\mathbf{s}_2^2$ and $\mathbf{s}_2^3$ from $\mathbf{s}_2^2 \oplus \mathbf{s}_2^3$ and $\mathbf{s}_2^2 \oplus \hat{\mathbf{s}}_2^3 \oplus \bar{\mathbf{s}}_2^3$. Then, we compute $\mathbf{s}_0^2, \mathbf{s}_1^2$ and $\mathbf{s}_0^3, \mathbf{s}_1^3$ from $\mathbf{s}_2^2, \mathbf{s}_3^2$ and $\mathbf{s}_2^3, \mathbf{s}_3^3$, respectively. Finally, we recover $\mathbf{s}_0^0 \oplus \mathbf{s}_0^2$ and $\mathbf{s}_0^1 \oplus \mathbf{s}_0^3$ from $\mathbf{s}_0^2, \mathbf{s}_0^0 \oplus \hat{\mathbf{s}}_0^2 \oplus \bar{\mathbf{s}}_0^2$ and $\mathbf{s}_0^3, \mathbf{s}_0^1 \oplus \hat{\mathbf{s}}_0^3 \oplus \bar{\mathbf{s}}_0^3$, respectively.

Similarly, we can repair node 1 by downloading six vectors

$$\mathbf{s}_0^0 \oplus \mathbf{s}_0^2, \mathbf{s}_0^1 \oplus \mathbf{s}_0^3, \mathbf{s}_2^0 \oplus \mathbf{s}_2^1, \mathbf{s}_3^1, \mathbf{s}_3^0, \mathbf{s}_2^0 \oplus \hat{\mathbf{s}}_2^1 \oplus \bar{\mathbf{s}}_2^1,$$

repair node 2 by downloading

$$\mathbf{s}_0^1 \oplus \mathbf{s}_0^3, \mathbf{s}_1^3, \mathbf{s}_3^1, \mathbf{s}_3^3, \mathbf{s}_2^0 \oplus \hat{\mathbf{s}}_2^1 \oplus \bar{\mathbf{s}}_2^1, \mathbf{s}_2^2 \oplus \hat{\mathbf{s}}_2^3 \oplus \bar{\mathbf{s}}_2^3,$$

and repair node 3 by downloading

$$\mathbf{s}_0^0 \oplus \mathbf{s}_0^2, \mathbf{s}_1^2, \mathbf{s}_1^0, \mathbf{s}_0^0 \oplus \hat{\mathbf{s}}_0^2 \oplus \bar{\mathbf{s}}_0^2, \mathbf{s}_2^0 \oplus \mathbf{s}_2^1, \mathbf{s}_2^2 \oplus \bar{\mathbf{s}}_2^3.$$

C. Other Constructions

There exist other constructions by employing the transformation given in Section IV for a zigzag-decodable code with asymptotically minimum repair bandwidth for some data nodes. Specifically, we can first construct a new zigzag-decodable codes that have asymptotically minimum repair bandwidth for the first γ data nodes by designing an encoding matrix, where $k > \gamma > 0$. Then, we can employ the transformation with for the zigzag-decodable codes with $\lceil \frac{k+r-\gamma}{r} \rceil$ times. We can show that the transformed codes also have asymptotically minimum repair bandwidth for all nodes, as in the transformed $\mathcal{ZDR}_1(k,r)$ codes in Section V-A.

VI. Comparisons

In this section, we compare in the decoding complexity and storage overhead of the proposed ZDR codes with other related codes, such as MDS codes with exactly or asymptotically minimum repair bandwidth in [10], [11], [18], [24].

Table IV shows the comparison of existing high code rate MDS codes with asymptotically optimal repair bandwidth or exactly optimal repair bandwidth with the proposed ZDR codes. Suppose that r data nodes are erased and we define the decoding complexity as the ratio of the number of XOR operations involved in recovering the erased r data nodes to the number of packets stored in a node. In $\mathcal{ZDR}_1(k,r)$, each node contains r packets. We need to recover the erased r^2 data packets from the surviving $k-r$ data nodes and r coded nodes. First, by Lemma 4, we can compute $r(r-1)$ coded packets from $\frac{r(r-1)}{2}$ pairs of linear combinations and it takes $\frac{5(r-1)rL}{4}$ XORs.[1] Then, we subtract $r(k-r)$ data packets in $k-r$ data nodes from the obtained r^2 coded packets and it takes $(k-r)r^2L$ XORs. Finally, we can recover the erased r^2 data packets in r data nodes by solving r linear systems each of size $r \times r$ by zigzag decoding and it takes r^3L XORs. Therefore, the decoding complexity of $\mathcal{ZDR}_1(k,r)$ is $O((k-r)rL+r^2L)$. Similarly, we can show that the decoding complexity of $\mathcal{ZDR}_2(k,r)$ is also $O((k-r)rL+r^2L)$. Since the decoding complexity of the existing MDS codes in [10], [11], [18], [24] is $O((k-r)rL^2+r^2L^2)$, the proposed ZDR codes have less decoding complexity than the other existing MDS codes listed in the table. Note that the lower decoding complexity of ZDR codes is obtained at a cost of the storage overhead. The storage overhead of $\mathcal{ZDR}_1(k,r)$ and $\mathcal{ZDR}_2(k,r)$ is $(r-1)r^{k-1}-r+1$ and $(k-1)(r-1)$, respectively. When the packet length L is sufficiently large, then the storage overhead is negligible.

VII. Conclusion

We propose ZDR codes with any $k \geq 2$ data nodes and any $r \geq 2$ coded nodes. We present two explicit constructions of ZDR codes that have asymptotically minimum repair bandwidth for all $k+r$ nodes. Compared with other related codes with exactly or asymptotically minimum repair bandwidth, the proposed ZDR codes have the zigzag decoding property and thus have less encoding complexity and decoding complexity. However, the proposed ZDR codes only have efficient repair bandwidth for $d = k+r-1$, i.e., all the surviving nodes are connected to repair the failed node. How to design ZDR codes with asymptotically minimum repair bandwidth for more flexible parameters ($k+1 \leq d \leq k+r-1$) is our future work.

[1]When L is large enough, the additional storage overhead is negligible and the size of a coded packet is roughly L.

References

[1] I. S. Reed and G. Solomon, "Polynomial Codes over Certain Finite Fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[2] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in Globally Distributed Storage Systems," in *Proc. of USENIX OSDI*, 2010.

[3] S. Muralidhar, W. Lloyd, S. Roy, C. Hill, E. Lin, W. Liu, S. Pan, S. Shankar, V. Sivakumar, L. Tang *et al.*, "F4: Facebook's Warm Blob Storage System," in *Proc. of USENIX OSDI*, 2014.

[4] S. Gollakota and D. Katabi, "Zigzag Decoding: Combating Hidden Terminals in Wireless Networks," in *Proc. of SIGCOMM*, vol. 38, no. 4, 2008.

[5] C. W. Sung and X. Gong, "A ZigZag-Decodable Code with the MDS Property for Distributed Storage Systems," in *Proc. IEEE Int. Symp. Inf. Theory*, Istanbul, Jul. 2013, pp. 341–345.

[6] M. Dai, W. S. Chi, H. Wang, X. Gong, and Z. Lu, "A New Zigzag-Decodable Code with Efficient Repair in Wireless Distributed Storage," *IEEE Trans. on Mobile Computing*, vol. 16, no. 5, pp. 1218–1230, 2017.

[7] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network Coding for Distributed Storage Systems," *IEEE Trans. on Information Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[8] 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. on Information Theory*, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.

[9] C. Suh and K. Ramchandran, "Exact-Repair MDS Code Construction Using Interference Alignment," *IEEE Trans. on Information Theory*, vol. 57, no. 3, pp. 1425–1442, Mar. 2011.

[10] J. Li, X. Tang, and C. Tian, "A Generic Transformation to Enable Optimal Repair in MDS Codes for Distributed Storage Systems," *IEEE Trans. on Information Theory*, vol. 64, no. 9, pp. 6257–6267, 2018.

[11] M. Ye and A. Barg, "Explicit Constructions of High-Rate MDS Array Codes with Optimal Repair Bandwidth," *IEEE Trans. on Information Theory*, vol. 63, no. 4, pp. 2001–2014, 2017.

[12] Y. Wang, X. Yin, and X. Wang, "MDR Codes: A New Class of RAID-6 Codes with Optimal Rebuilding and Encoding," *IEEE J. on Selected Areas in Commun.*, vol. 32, no. 5, pp. 1008–1018, 2013.

[13] E. E. Gad, R. Mateescu, F. Blagojevic, C. Guyot, and Z. Bandic, "Repair-Optimal MDS Array Codes over GF(2)," in *Proc. IEEE Int. Symp. Inf. Theory*, 2013, pp. 887–891.

[14] H. Hou, K. W. Shum, M. Chen, and H. Li, "BASIC Regenerating Code: Binary Addition and Shift for Exact Repair," in *Proc. IEEE Int. Symp. Inf. Theory*, Istanbul, Jul. 2013, pp. 1621–1625.

[15] ——, "BASIC Codes: Low-Complexity Regenerating Codes for Distributed Storage Systems," *IEEE Trans. on Information Theory*, vol. 62, no. 6, pp. 3053–3069, 2016.

[16] H. Hou, P. P. C. Lee, Y. S. Han, and Y. Hu, "Triple-Fault-Tolerant Binary MDS Array Codes with Asymptotically Optimal Repair," in *Proc. IEEE Int. Symp. Inf. Theory*, Aachen, Jun. 2017.

[17] H. Hou and Y. S. Han, "A Class of Binary MDS Array Codes with Asymptotically Weak-Optimal Repair," *SCIENCE CHINA Information Sciences http://engine.scichina.com/doi/10.1007/s11432-018-9485-7*, vol. 61, no. 10, pp. 1–12, 2018.

[18] H. Hou, Y. S. Han, P. P. Lee, Y. Hu, and H. Li, "A New Design of Binary MDS Array Codes with Asymptotically Weak-Optimal Repair," *IEEE Trans. on Information Theory*, vol. 65, no. 11, pp. 7095–7113, 2019.

[19] H. Hou, Y. S. Han, and P. P. Lee, "Binary MDS Array Codes with Asymptotically Optimal Repair for All Columns," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, 2019, pp. 1–9.

[20] I. Tamo, Z. Wang, and J. Bruck, "Zigzag Codes: MDS Array Codes with Optimal Rebuilding," *IEEE Trans. on Information Theory*, vol. 59, no. 3, pp. 1597–1616, 2013.

TABLE IV: Comparison of high code rate erasure codes with efficient repair procedure.

Codes	Storage overhead	Repair band.	Sub-packetization	Packet length L	Decoding com.
Codes in [10]	0	optimal	$L \cdot r^{\lceil \frac{n}{r} \rceil}$	$\log_2 n$	$O((k-r)rL^2 + r^2L^2)$
Codes 1 in [11]	0	optimal	$L \cdot r^n$	$\log_2 n \operatorname{lcm}(1, \ldots, r)$	$O((k-r)rL^2 + r^2L^2)$
Codes 2 in [11]	0	optimal	$L \cdot r^{n-1}$	$\log_2 n$	$O((k-r)rL^2 + r^2L^2)$
Codes in [18]	0	asym. optimal	L	$(\frac{r}{2})^{d-k+1}$	$O((k-r)rL^2 + r^2L^2)$
Codes in [24]	0	optimal	$L(d-k+1)^{\lceil \frac{n}{d-k+1} \rceil}$	n	$O((k-r)rL^2 + r^2L^2)$
$\mathcal{ZDR}_1(k,r)$	$(r-1)r^{k-1} - r + 1$	asym. optimal	$L \cdot r$	r^k	$O((k-r)rL + r^2L)$
$\mathcal{ZDR}_2(k,r)$	$(k-1)(r-1)$	optimal	$L \cdot r^{\lceil \frac{n}{r} \rceil}$	kr	$O((k-r)rL + r^2L)$

[21] N. Raviv, N. Silberstein, and T. Etzion, "Constructions of High-Rate Minimum Storage Regenerating Codes over Small Fields," *IEEE Trans. on Information Theory*, vol. 63, no. 4, pp. 2015–2038, 2017.
[22] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures," *IEEE Trans. on Computers*, vol. 44, no. 2, pp. 192–202, 1995.
[23] M. Blaum, J. Bruck, and A. Vardy, "MDS Array Codes with Independent Parity Symbols," *IEEE Trans. on Information Theory*, vol. 42, no. 2, pp. 529–542, 1996.
[24] H. Hou and P. P. C. Lee, "Binary MDS Array Codes with Optimal Repair," *IEEE Trans. on Information Theory*, vol. 66, no. 3, pp. 1405–1422, 2020.
[25] H. Hou, Y. S. Han, K. W. Shum, and H. Li, "A Unified Form of EVENODD and RDP Codes and Their Efficient Decoding," *IEEE Trans. on Communications*, vol. 66, no. 11, pp. 5053–5066, 2018.
[26] J. Chen, H. Li, H. Hou, B. Zhu, T. Zhou, L. Lu, and Y. Zhang, "A New Zigzag MDS Code with Optimal Encoding and Efficient Decoding," in *IEEE International Conference on Big Data*, 2014, pp. 1–6.
[27] J. Qureshi, C. H. Foh, and J. Cai, "Optimal Solution for the Index Coding Problem Using Network Coding over GF(2)," in *Proc. 9th Annu. IEEE Commun. Soc. Conf. Sensor Mesh Ad Hoc Commun. Netw.*, 2012, pp. 209–217.
[28] X. Gong, P. Hu, S. K. W., and W. S. Chi, "A Zigzag-Decodable Ramp Secret Sharing Scheme," *IEEE Trans. on Information Forensics and Security*, vol. 13, no. 8, pp. 1906–1916, 2018.
[29] J. Li and X. Tang, "A Note on the Transformation to Enable Optimal Repair in MDS Codes for Distributed Storage Systems," *arXiv preprint: arXiv:1901.06067v1*, 2019.
[30] H. Hou, P. P. C. Lee, and Y. S. Han, "Multi-Layer Transformed MDS Codes with Optimal Repair Access and Low Sub-Packetization," *arXiv preprint arXiv:1907.08938*, 2019.

Hanxu Hou received the B.Eng. degree in Information Security from Xidian University, Xian, China, in 2010, and Ph.D. degrees in the Dept. of Information Engineering from The Chinese University of Hong Kong in 2015 and in the School of Electronic and Computer Engineering, Peking University. He is now an Associate Professor with the School of Electrical Engineering & Intelligentization, Dongguan University of Technology. His research interests include erasure coding and coding for distributed storage systems.

Patrick P. C. Lee received the B.Eng. degree (first class honors) in Information Engineering from the Chinese University of Hong Kong in 2001, the M.Phil. degree in Computer Science and Engineering from the Chinese University of Hong Kong in 2003, and the Ph.D. degree in Computer Science from Columbia University in 2008. He is now an Associate Professor of the Department of Computer Science and Engineering at the Chinese University of Hong Kong. His research interests are in various applied/systems topics including storage systems, distributed systems and networks, operating systems, dependability, and security

Yunghsiang S. Han received B.Sc. and M.Sc. degrees in electrical engineering from the National Tsing Hua University, Hsinchu, Taiwan, in 1984 and 1986, respectively, and a Ph.D. degree from the School of Computer and Information Science, Syracuse University, Syracuse, NY, in 1993. He was from 1986 to 1988 a lecturer at Ming-Hsin Engineering College, Hsinchu, Taiwan. He was a teaching assistant from 1989 to 1992, and a research associate in the School of Computer and Information Science, Syracuse University from 1992 to 1993. He was, from 1993 to 1997, an Associate Professor in the Department of Electronic Engineering at Hua Fan College of Humanities and Technology, Taipei Hsien, Taiwan. He was with the Department of Computer Science and Information Engineering at National Chi Nan University, Nantou, Taiwan from 1997 to 2004. He was promoted to Professor in 1998. He was a visiting scholar in the Department of Electrical Engineering at University of Hawaii at Manoa, HI from June to October 2001, the SUPRIA visiting research scholar in the Department of Electrical Engineering and Computer Science and CASE center at Syracuse University, NY from September 2002 to January 2004 and July 2012 to June 2013, and the visiting scholar in the Department of Electrical and Computer Engineering at University of Texas at Austin, TX from August 2008 to June 2009. He was with the Graduate Institute of Communication Engineering at National Taipei University, Taipei, Taiwan from August 2004 to July 2010. From August 2010 to January 2017, he was with the Department of Electrical Engineering at National Taiwan University of Science and Technology as Chair Professor. Now he is with School of Electrical Engineering & Intelligentization at Dongguan University of Technology, China. He is also a Chair Professor at National Taipei University from February 2015. His research interests are in error-control coding, wireless networks, and security.

Dr. Han was a winner of the 1994 Syracuse University Doctoral Prize and a Fellow of IEEE. One of his papers won the prestigious 2013 ACM CCS Test-of-Time Award in cybersecurity.