Fountain Codes Based on Zigzag Decodable Coding

Takayuki Nozaki
Kanagawa University, JAPAN
Email: nozaki@kanagawa-u.ac.jp

***Abstract*—Fountain codes based on non-binary low-density parity-check (LDPC) codes have good decoding performance when the number of source packets is finite. However, the space complexity of the decoding algorithm for fountain codes based on non-binary LDPC codes grows exponentially with the degree of a field extension. Zigzag decodable codes generate the output packets from source packets by using shift and exclusive or. It is known that the zigzag decodable codes are efficiently decoded by the zigzag decoder. In this paper, by applying zigzag decodable coding to fountain codes, we propose a fountain code whose space decoding complexity is nearly equal to that for the Raptor codes. Simulation results show that the proposed fountain coding system outperforms Raptor coding system in terms of the overhead for the received packets.**

I. Introduction

On the Internet, a message is transmitted in a sequence of packets. We consider that the packets which are not correctly received are erased. Hence, the Internet is modeled as the *packet erasure channel* (PEC).

Retransmitting packets is a method to realize reliable communication over the Internet. However, in the case of multicasting, it is difficult to retransmit packets as the number of receiver increases, since the retransmission requests can overwhelm the sender.

Fountain code [1] realizes reliable communication on multicasting. We assume that the transmitted message are divided into k source packets. Fountain code produces infinite output packets from k source packets. The receivers decode the message from *arbitrary* $k(1+\alpha)$ output packets with $\alpha > 0$. Hence, the receivers need not request retransmitting packets. The parameter α is referred to as *overhead for received packets*. For good fountain codes, the value of α is close to zero.

Luby first realized the concepts of the fountain code with LT codes[1] [2]. Each output packet of the LT code is generated as follows. Firstly, the encoder randomly chooses the degree d of the output packet according to the degree distribution $\Omega(x)$. Secondly, the encoder randomly chooses d distinct source packets. Finally, the encoder outputs bit-wise exclusive or (XOR) of the d source packets as an output packet. Decoding of LT codes is similar to that of low-density parity-check (LDPC) codes over the binary erasure channel. More precisely, the decoder constructs the factor graph from the received packets and recover the source packets by using the peeling algorithm [3].

[1]Notice that the Tornado codes [1] are not fountain codes since the Tornado codes produce the *finite* output packets.

Raptor codes [4] are fountain codes which achieves $\alpha \to 0$ as $k \to \infty$ with linear time encoding and decoding. Encoding of Raptor code is divided into two stages. In the first stage, the encoder generates the *precoded packets* from the source packets by using an LDPC code. In the second stage, the encoder generates the output packets from the precoded packets by using an LT code. Decoding of the Raptor codes is similar to that of the LT codes.

When the number k of source packets is finite, the fountain code based on a non-binary LDPC code [5] outperforms Raptor codes. However, the space complexity of the decoding algorithm for the fountain codes based on non-binary LDPC codes grows exponentially with the degree of a field extension, similarly to the decoding algorithm for the non-binary LDPC codes.

Gollakota and Katabi [6] proposed zigzag decoding to combat hidden terminals in wireless networks. Sung and Gong [7] proposed zigzag decodable (ZD) codes which are efficiently decoded by the zigzag decoder, for the distributed storage systems. As a similar study, Qureshi *et al.* [8] proposed triangular codes and back-substitution decoding method for the index decoding problem. Both ZD codes and triangular codes generates output packets from the source packets by using shift and XOR.

Qureshi *et al.* [9] suggested that the triangular codes can be applied to the fountain codes. However, there are no comparison with other fountain codes and there are no analysis of the fountain codes based on triangular coding.

In this paper, we investigate the fountain codes based on ZD coding. The contributions of this paper are the followings: (1) We improve zigzag decoding. (2) We give a factor graph representation of the ZD codes. (3) We propose a fountain code based on ZD coding and its decoding algorithm. (4) We prove that the decoding erasure probability for the proposed fountain coding system is lower than that for the Raptor coding system. As an advantage of the fountain code based on ZD coding, the space complexity of the decoding algorithm grows linearly with the received bits. In other words, the space complexity of the decoding algorithm for the fountain codes based on ZD coding is *slightly* larger than that for the Raptor codes at the same overhead α.

The rest of the paper is organized as follows. Section II briefly explains the ZD codes and zigzag decoding by a toy example. Section III gives factor graph representation of the ZD codes. Section IV proposes the fountain codes based on ZD coding and its decoding algorithm. Section V analyzes the overhead, decoding performance and decoding complexity for the proposed fountain coding system. Moreover, simulation results in Section V give that the proposed fountain coding system outperforms Raptor coding system in terms of the overhead for the received packets.

This work was partially presented in [10].

II. Example of ZD Codes And Zigzag Decoding

This section explains the ZD code and the zigzag decoding algorithm with a toy example. Moreover, we point out a drawback of the zigzag decoding algorithm.

As a toy example, we consider a ZD code which generates two encoded packets from two source packets with length ℓ. The first encoded packet $\boldsymbol{x}_1 = (x_{1,1}, x_{1,2}, \dots, x_{1,\ell})$ is generated from the bit-wise XOR of two source packets $\boldsymbol{s}_1 = (s_{1,1}, s_{1,2}, \dots, s_{1,\ell}), \boldsymbol{s}_2 = (s_{2,1}, s_{2,2}, \dots, s_{2,\ell})$. The second encoded packet $\boldsymbol{x}_2 = (x_{2,1}, x_{2,2}, \dots, x_{2,\ell+1})$ is generated from the bit-wise XOR of $\boldsymbol{s}_2$ with a right shift and $\boldsymbol{s}_1$. Notice that the length of the second packet is $\ell+1$. Figure 1 describes the ZD code.

ZD codes are efficiently decoded by the zigzag decoding algorithm [6], [7]. The zigzag decoding algorithm starts from the *left* of the packets. In a similar way to the peeling decoding algorithm for the LDPC code over the binary erasure channel, the zigzag decoding algorithm proceeds by solving linear equations with one unknown variable.

In the case of the ZD code in Fig.1, the zigzag decoding algorithm proceeds as the following way. The decoder recovers $s_{1,1}$ from $x_{2,1}$ since $s_{1,1} = x_{2,1}$. The decoder recovers $s_{2,1}$ by solving $x_{1,1} = s_{1,1} + s_{2,1} = x_{2,1} + s_{2,1}$. Similarly, the decoder recovers $s_{1,2}, s_{2,2}, \dots, s_{2,\ell}$ and decoding is success.

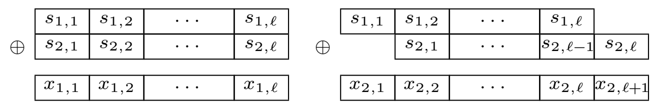
Remark 1: Recall that the original zigzag decoding algorithm [6], [7] starts from the *left* of the encoding packets. Hence, the ZD code described as in Fig.2 is not decoded by the original zigzag decoding algorithm. However, if decoding starts from the *right* of the encoding packets, the ZD code in Fig.2 is decodable. Actually, $s_{i,\ell}$ is recoverable from $x_{i,\ell+1}$ for $i = 1, 2, 3$. Substituting these values, we get $s_{i,\ell-1}$ for $i = 1, 2, 3$ in a similar way. Finally, we get $s_{i,1}$ for $i = 1, 2, 3$ and decoding is success.

Hence, the zigzag decoding algorithm is improved if decoding starts from the *left and right* of the encoding packets. This improved zigzag decoding algorithm is employed in Section IV.

Remark 2: Similar to the ZD codes, the triangular codes [8] generate the encoded packets from the source packets by using shift and XOR. The triangular code choose distinct shift amount of the source packets. Hence, the triangular code are always decodable from the left of the encoded packets. This decoding algorithm is referred as back-substitution algorithm [8]. Since there are no constraints of shift amount of source packets for the ZD codes, the triangular codes are special case of the ZD codes.

III. Factor Graphs for ZD codes

This section explains the matrix representation of the ZD codes [7] and gives factor graph representation of the ZD codes.



$s_{1,1}$ $s_{1,2}$ $\cdots$ $s_{1,\ell}$
$\oplus$
$s_{2,1}$ $s_{2,2}$ $\cdots$ $s_{2,\ell}$
$x_{1,1}$ $x_{1,2}$ $\cdots$ $x_{1,\ell}$
$s_{1,1}$ $s_{1,2}$ $\cdots$ $s_{1,\ell}$
$\oplus$
$s_{2,1}$ $\cdots$ $s_{2,\ell-1}$ $s_{2,\ell}$
$x_{2,1}$ $x_{2,2}$ $\cdots$ $x_{2,\ell}$ $x_{2,\ell+1}$

Fig. 1. A toy example of ZD code

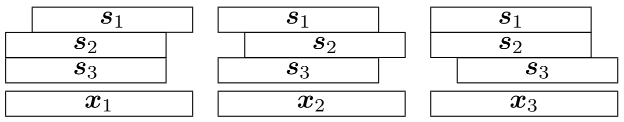


Fig. 2. A ZD code which is not decoded by the original zigzag decoding algorithm

A. Matrix Representation for ZD codes [7]

Let ℓ be the length of source packets. Denote the number of source packets, by k. A polynomial representation of the i-th source packet $(s_{i,1}, s_{i,2}, \dots, s_{i,\ell})$ is defined as

$$s_i(z) = \textstyle\sum_{j=1}^{\ell} s_{i,j} z^j.$$

Then, for the ZD codes, the polynomial representation of the i-th encoded packets is given by

$$x_i(z) = \textstyle\sum_{j=1}^{k} g_{i,j}(z) s_j(z), \tag{1}$$

where $g_{i,j}(z)$ is a monomial of z, i.e., $g_{i,j}(z) \in \{0, 1, z, z^2, \dots\}$. We denote the degree of $g_{i,j}(z)$, by $\deg(g_{i,j})$. Then, the length of the i-th encoded packet is $\ell + \max_j \deg(g_{i,j})$. We get the following matrix presentation with (1):

$$\boldsymbol{x}(z) = \boldsymbol{G}(z)\boldsymbol{s}(z).$$

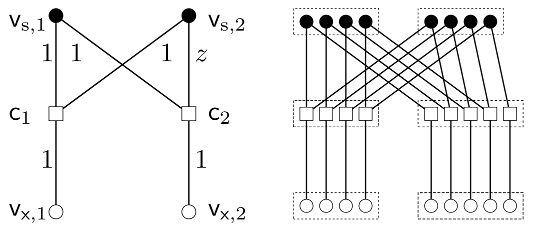
Example 1: The matrix representation of the ZD code in Fig.1 is

$$\boldsymbol{G}(z) = \begin{pmatrix} 1 & 1 \\ 1 & z \end{pmatrix}.$$

B. Factor Graph Representation of ZD codes

In this section, we give packet-wise and bit-wise factor graph representation of the ZD codes.

Firstly, we give packet-wise factor graph representation of the ZD codes. The factor graphs of the ZD codes consist of the sets of nodes $\mathsf{V}_\mathrm{s}, \mathsf{V}_\mathrm{x}, \mathsf{C}$ and *labeled* edges. The nodes in $\mathsf{V}_\mathrm{s}, \mathsf{V}_\mathrm{x}$ represent source packets and encoded packets, and are called source nodes and encoded nodes, respectively. The number of source packets (resp. encoding packets) is equal to $|\mathsf{V}_\mathrm{s}|$ (resp. $|\mathsf{V}_\mathrm{x}|$). The nodes in C represent constraints for the neighbor nodes, and are called factor nodes. The number of nodes in C is equal to $|\mathsf{V}_\mathrm{x}|$. All the encoded nodes are of degree one and the i-th encoded node and the i-th factor node are connected by an edge labeled by 1. If $g_{i,j}(z) \neq 0$, then the j-th source node and the i-th factor node are connected by an edge labeled by $g_{i,j}(z)$. If $g_{i,j}(z) = 0$, then the j-th source node and the i-th factor node are not connected. Note that the i-th factor node represents a constraint such that $\sum_{j \in \mathcal{N}_\mathrm{c}(i)} g_{i,j}(z) s_j(z) = x_i(z)$, where $\mathcal{N}_\mathrm{c}(i)$ gives the set of indexes of the source nodes connecting to the i-th factor node.



(a) Packet-wise representation (b) Bit-wise representation

Fig. 3. Factor graph representation of the ZD code in Fig.1

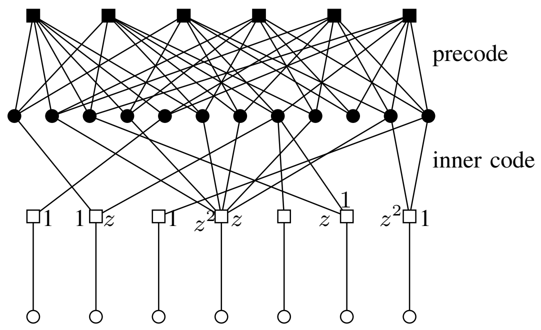


Fig. 4. An example of a factor graph for the proposed fountain code.

Secondly, we give bit-wise factor graph representation of the ZD codes. In this representation, edges are not labeled. Each source node and each encoded node corresponds to a bit of source packets and encoded packets, respectively. Each factor node in this representation gives a constraint such that XOR of the bits corresponding to the neighbor source nodes and the neighbor encoded node is 0. The (j, t)-th source node and $(i, t+t')$-th factor node are connected by an edge if $g_{i,j} = z^{t'}$. The (i, t)-th factor node and (i, t)-th encoded node are also connected by an edge.

Example 2: Figures 3(a) and 3(b) show the packet-wise and bit-wise factor graph representation of the ZD code in Fig.1 with $\ell = 4$. The black circle, white circle and white square represent source nodes, encoded nodes and factor nodes, respectively. Each dashed rectangular in Fig.3(b) corresponds to a node in Fig.3(a).

IV. Fountain Code Based on ZD Coding

In this section, we propose a fountain coding system based on ZD coding. In a similar way to Raptor codes, the proposed fountain code firstly generates precoded packets from source packets by using an LDPC code. Next the proposed fountain code generates the output packets from the precoded packets with inner coding, which is a combination of LT code and ZD code. Moreover, this section gives a decoding algorithm for the proposed fountain codes.

A. Encoding

The system parameters for the proposed fountain coding system are the precode $\mathcal{C}$, the degree distribution for the inner code $\Omega(x) = \sum_i \Omega_i x^i$ and the *shift distribution* $\Delta(x) = \sum_{i=0}^{s_m} \Delta_i x^i$, where Δ_i represents the probability that the shift amount is i. Notice that $\Omega(1) = 1$ and $\Delta(1) = 1$.

Similarly to the Raptor codes, the proposed fountain code generates the precoded packets $(\boldsymbol{a}_1, \dots, \boldsymbol{a}_n)$ from the source packets $(\boldsymbol{s}_1, \dots, \boldsymbol{s}_k)$ by the precode $\mathcal{C}$ in the first stage. In the second stage, the proposed fountain code generates the infinite output packets as the following procedure for $t = 1, 2, \dots$.

1) Choose a degree d of the output packet according to the degree distribution $\Omega(x)$. In other words, choose d with probability Ω_i.
2) Choose d-tuple of shift amount $(\tilde{\delta}_1, \dots, \tilde{\delta}_d) \in [0, s_m]^d$ in independent of each other according to shift distribution $\Delta(x)$. Define $\tilde{\delta}_{\min} := \min_i\{\tilde{\delta}_1, \dots, \tilde{\delta}_d\}$ and calculate $\delta_i := \tilde{\delta}_i - \tilde{\delta}_{\min}$ for $i \in [1, d]$.
3) Choose d distinct precoded packets uniformly. Let $(j_1, j_2, \dots, j_d)$ denote the d-tuple of indexes of the chosen precoded packets. Then the polynomial representation for the t-th output packet is given as
$$\textstyle\sum_{i=1}^{d} z^{\delta_i} a_{j_i}(z).$$
Note that the information of the tuples $(\delta_1, \dots, \delta_d)$, $(j_1, \dots, j_d)$ is in the header of the t-th output packet.

B. Decoding

Let $(\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_{\tilde{k}})$ be a tuple of the received packets, where $\tilde{k} = k(1 + \alpha)$. Firstly, similarly to the Raptor code, the decoder of the proposed fountain coding system constructs a factor graph from the precode $\mathcal{C}$ and headers of the received packets. The generated factor graphs depend on receivers, since the received packets depend on receivers. After constructing a factor graph in bit-wise representation, the decoder recovers source packets from $\tilde{k}$ received packets in a similar way to the peeling decoder [3] for the LDPC code over the BEC.

Figure 4 illustrates an example of factor graph in packet-wise representation. In this example, we employ $(3, 6)$-regular LDPC code as the precode. The black (resp. white) circles represent the precoded (resp. received) packets, and are called variable nodes. The black and white squares represent the check nodes of the precode and factor nodes of the inner code, respectively. Each edge is labeled by a monomial of z. Note that all the edges in the factor graph corresponding to the precode are labeled by 1. Several labels in Fig.4 are omitted for the visibility of the figure.

Remark 3: Qureshi *et al.* [9] suggested a fountain coding system based on triangular coding. This fountain coding system is an improvement of the LT code. Decoding of this fountain coding system starts from the left of the encoded packets. In this fountain coding system, the encoder chooses *distinct* d shift amount $\delta_1, \dots, \delta_d$, namely, $\delta_i \neq \delta_j$ for $i \neq j$.

On the other hand, decoding of the proposed fountain coding system starts both left and right of the received packets. Moreover, the proposed encoding algorithm is a generalization of the fountain code in [9], since the proposed encoding algorithm is an improvement of Raptor code and can choose d shift amount $\delta_1, \dots, \delta_d$ with $\delta_i = \delta_j$ for $i \neq j$.

V. Performance Evaluation

In this section, we evaluate the performance of the proposed fountain coding system.

A. Overhead

For the proposed fountain code, the length of the output packets is slightly longer than that of the source packets. Denote the length of the i-th received packet, as $\ell + \ell_i$, with $\ell_i \geq 0$. Then, the total number of bits in the received packets is $\tilde{k}\ell + \sum_{i=1}^{\tilde{k}} \ell_i$, where $\tilde{k}$ gives the total number of received packets, namely $\tilde{k} = k(1+\alpha)$. Hence, we need to consider not only the number of received packets $\tilde{k}$ but also the total number of the bits in the data section of the received packets[2] $k\ell(1+\beta)$. We refer the value β as the *overhead for the received bits*. The value β is given by

$$\beta = \alpha + (k\ell)^{-1}\textstyle\sum_{i=1}^{\tilde{k}}\ell_i.$$

Notice that for the Raptor codes, $\beta = \alpha$ since $\ell_i = 0$ for all i. From the above equation, to calculate β, we need to evaluate $\ell_1, \ell_2, \dots, \ell_{\tilde{k}}$.

Let L be a random variable which represents a length of a received packet. For a given degree distribution $\Omega(x)$ and a given shift distribution $\Delta(x)$, the expectation of L is given as the following proposition.

Proposition 1: For a given degree distribution $\Omega(x) = \sum_i \Omega_i x^i$ and a given shift distribution $\Delta(x) = \sum_{i=0}^{s_m} \Delta_i x^i$, the following holds:

$$\mathbb{E}[L] = \ell + s_m - \textstyle\sum_{i=0}^{s_m-1}\Omega(\Delta_{[0,i]}) - \sum_{i=1}^{s_m}\Omega(\Delta_{[i,s_m]}),$$

where $\Delta_{[i,j]} := \sum_{t\in[i,j]} \Delta_t$.
Due to the space limitation, we omit the proof.

When the shift distribution $\Delta(x)$ is a uniform distribution, we get the following corollary from Proposition 1.

Corollary 1: When the shift distribution is a uniform distribution, i.e., $\Delta(x) = \sum_{i=0}^{s_m}(s_m+1)^{-1}x^i$, for a given degree distribution $\Omega(x)$, the following holds:

$$\mathbb{E}[L] = \ell + s_m - \textstyle\sum_{i=1}^{s_m}\Omega(i(s_m+1)^{-1}).$$

By using Proposition 1, for a fixed α, the expectation of the overhead for the received bits is

$$\beta = (1+\alpha)\ell^{-1}\mathbb{E}[L] - 1.$$

Since $\mathbb{E}[L] \leq \ell + s_m$, $\beta \to \alpha$ as $\ell \to \infty$.

B. Decoding Erasure Probability

In this section, we compare the decoding erasure probability for the proposed fountain coding system with that for the Raptor coding system.

We denote the proposed fountain code with the precode $\mathcal{C}$, the degree distribution $\Omega(x)$ and the shift distribution $\Delta(x)$, as $\mathcal{F}(\mathcal{C}, \Omega(x), \Delta(x))$. Note that the Raptor code is a special case for the proposed fountain code with $\Delta(x) = 1$. In other words, $\mathcal{F}(\mathcal{C}, \Omega(x), 1)$ represents the Raptor code with the precode $\mathcal{C}$ and the degree distribution $\Omega(x)$. In this section, we will prove that the fountain code $\mathcal{F}(\mathcal{C}, \Omega(x), \Delta(x))$ outperforms the Raptor code $\mathcal{F}(\mathcal{C}, \Omega(x), 1)$ in terms of the decoding erasure probability.

[2]Simply, we refer the total number of the bits in the data section of the received packets as the number of received bits.

To prove the above, we use the following lemma.

Lemma 1: Fix an unlabeled factor graph $\mathtt{G}$ in packet-wise representation. If decoding succeeds for the factor graph $\mathtt{G}$ all the edges of which are labeled by 1, the decoding also succeeds for the factor graph $\mathtt{G}$ with arbitrary labeling.

Outline of the proof: We use the proof by contradiction. We assume that decoding is a failure for the factor graph $\mathtt{G}$ with some labeling. Then, the factor graph $\mathtt{G}$ contains some stopping sets. Thus, the decoding is also a failure for the factor graph $\mathtt{G}$ all the edges of which are labeled by 1. ■

This lemma shows that, for a fixed unlabeled factor graph, the decoding succeeds for the proposed fountain coding system if the decoding succeeds for the Raptor coding system. From this lemma, we obtain the following theorem.

Theorem 1: Let $\mathrm{P}(\alpha, \mathcal{C}, \Omega(x), \Delta(x))$ be the decoding erasure probability for the fountain code $\mathcal{F}(\mathcal{C}, \Omega(x), \Delta(x))$ at the overhead α for the received packets. For arbitrary $\alpha, \mathcal{C}, \Omega(x), \Delta(x)$, the following holds:

$$\mathrm{P}(\alpha, \mathcal{C}, \Omega(x), 1) \geq \mathrm{P}(\alpha, \mathcal{C}, \Omega(x), \Delta(x)).$$

This theorem shows that the fountain code $\mathcal{F}(\mathcal{C}, \Omega(x), \Delta(x))$ outperforms the Raptor code $\mathcal{F}(\mathcal{C}, \Omega(x), 1)$ in terms of the decoding erasure probability. The following corollary derived from Theorem 1.

Corollary 2: If Raptor code $\mathcal{F}(\mathcal{C}, \Omega(x), 1)$ achieves $\alpha = 0$, then the proposed fountain code $\mathcal{F}(\mathcal{C}, \Omega(x), \Delta(x))$ also achieves $\alpha = 0$. In other words, if $\mathrm{P}(0, \mathcal{C}, \Omega(x), 1) = 0$, then $\mathrm{P}(0, \mathcal{C}, \Omega(x), \Delta(x)) = 0$.
From this corollary, the proposed fountain code achieves $\alpha = 0$.

C. Decoding Complexity

Recall that the proposed decoding algorithm works on the factor graph in bit-wise representation in a similar way to the peeling algorithm for the LDPC code over the BEC. Hence the space complexity of the decoding algorithm is equal to the total number of factor nodes in inner code, precoded nodes and factor nodes in precode in bit-wise representation. Then, the space complexity of the decoding algorithm is $k\ell(2+\beta)+n\ell$. Similarly, the decoding complexity of the decoding algorithm for the Raptor code is $k\ell(2+\alpha)+n\ell$. Note that $\beta = \alpha$ in the case of the Raptor code.

The number of iteration of the peeling algorithm is upper bound on the number of check nodes in the factor graph. For the Raptor coding system, since the decoding algorithm works on the factor graph in packet-wise representation, the number of iteration is upper bounded on $\alpha k + n$. On the other hand, for the proposed fountain coding system, the number of iteration is upper bounded on $\ell(\beta k + n)$ since the decoding algorithm works on the factor graph in bit-wise representation. This is the main drawback of the proposed fountain coding system. However, the number of the iteration of the proposed

TABLE I
OVERHEADS α^*, β^* AT WHICH THE DECODING ERASURE PROBABILITIES ACHIEVE 0.1 FOR $s_m = 0, 1, 2, 3$.

(a) $k = 900, n = 1000$

s_m	0	1	2	3
α^*	0.2300	0.0800	0.0556	0.0422
$\mathbb{E}[L]$	100	100.8379	101.6205	102.3864
β^*	0.2300	0.0890	0.0727	0.0648

(b) $k = 1800, n = 2000$

s_m	0	1	2	3
α^*	0.1750	0.0717	0.0483	0.0367
$\mathbb{E}[L]$	100	100.8379	101.6205	102.3864
β^*	0.1750	0.0801	0.0653	0.0614

fountain coding system will be reduced if the scheduling of the decoding algorithm is optimized.

D. Simulation Result

This section shows that the proposed fountain coding system outperforms the Raptor coding system in terms of the overhead for the received packets and the received bits by simulation results.

As a precode, we employ a (3,30)-regular LDPC code ensemble with $(k, n) = (900, 100), (1800, 2000)$. The degree distribution for the inner code is $\Omega(x) = 0.007969x + 0.493570x^2 + 0.166220x^3 + 0.072646x^4 + 0.032558x^5 + 0.056058x^8 + 0.037229x^9 + 0.055590x^{19} + 0.025023x^{65} + 0.003135x^{66}$ [4]. The length of the source packets ℓ is 100 bits. The shift distribution is uniform distribution, i.e., $\Delta(x) = \sum_{i=0}^{s_m} \frac{1}{1+s_m} x^i$.

In this simulation, we examine the necessary number of the received packets to recover the source packets for the Raptor code and the proposed code. Figure 5 displays histograms of the overhead for the received packets. The histogram of the overhead α for the Raptor code is shown in $s_m = 0$. The histograms with $s_m = 1, 2, 3$ give the overhead α for the proposed fountain codes with maximum shift amount $s_m = 1, 2, 3$, respectively. From Fig.5, we see that the necessary number of received packets to recover the source packets for the proposed fountain code is smaller than that for the Raptor code. Moreover, the necessary number of received packets to recover the source packets decreases as the maximum shift amount s_m increases.

Table I shows the overheads α^*, β^* at which the decoding erasure probabilities achieve 0.1. The value $\mathbb{E}[L]$ are derived from Corollary 1. Table I shows that the overheads α^*, β^* for the received packets and received bits decrease as s_m increases. In other words, the proposed fountain coding system outperforms the Raptor coding system in terms of the overhead for the received packets and the received bits. Moreover, Table I shows that the overheads α^*, β^* decrease as the number of source packets increases.

VI. CONCLUSION

In this paper, we have proposed a fountain coding system based on ZD coding. We have shown that the space complexity

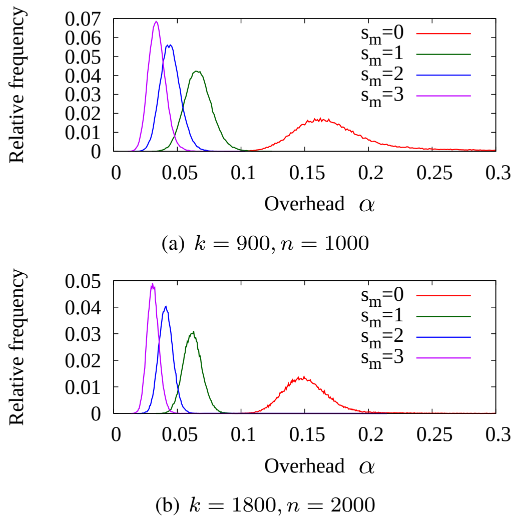


Fig. 5. Histograms of the overhead α to recover the source packets.

of the decoding algorithm for the proposed fountain coding system and the Raptor coding system depends on the received bits. We have proved that the decoding erasure probability for the proposed fountain coding system is lower than that for the Raptor coding system for a fixed precode, degree distribution and overhead α. Moreover, we have shown that the proposed fountain coding system outperforms the Raptor coding system in terms of the overhead for the received packets and the received bits by simulation results.

ACKNOWLEDGMENT

This work was partially supported by JSPS KAKENHI Grant Number 25889061.

REFERENCES

[1] J. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1528–1540, 2002.

[2] M. Luby, "LT codes," in *Proc. The 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'2002)*, 2002, pp. 271–280.

[3] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, "Practical loss-resilient codes," in *Proc. the 29th annual ACM Symposium on Theory of Computing*, 1997, pp. 150–159.

[4] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[5] K. Kasai, D. Declercq, and K. Sakaniwa, "Fountain coding via multiplicatively repeated non-binary LDPC codes," *IEEE Transactions on Communications*, vol. 60, no. 8, pp. 2077–2083, 2012.

[6] S. Gollakota and D. Katabi, "Zigzag decoding: combating hidden terminals in wireless networks," in *Proc. SIGCOMM*, 2008, pp. 159–170.

[7] C. W. Sung and X. Gong, "A zigzag-decodable code with the MDS property for distributed storage systems," in *Proc. 2013 IEEE Int. Symp. Inf. Theory (ISIT)*, 2013, pp. 341–345.

[8] J. Qureshi, C. H. Foh, and J. Cai, "Optimal solution for the index coding problem using network coding over GF(2)," in *Proc. 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2012, pp. 209–217.

[9] ——, "Primer and recent developments on fountain codes," *Recent Advances in Communications and Networking Technology*, vol. 2, pp. 2–11, 2013.

[10] T. Nozaki, "Fountain codes based on triangular coding," in *IEICE Tech. Rep.*, vol. 113, no. 228, IT2013-37, Sep. 2013, pp. 31–36, (in Japanese).