Digital Communications and Networks 9 (2023) 869–878

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/dcan

Triangular code: Near-optimal linear time fountain code

Jalaluddin Qureshi [a,*], Chuan Heng Foh [b]

[a] Openbet North America, Montreal, Canada
[b] 5GIC & 6GIC, Institute for Communication Systems (ICS), University of Surrey, Guildford, United Kingdom

ARTICLE INFO

Keywords:
LT code
Decoder
Computational complexity
Green computing
Network coding

ABSTRACT

In this paper, we propose Triangular Code (TC), a new class of fountain code with near-zero redundancy and linear encoding and decoding computational complexities of $\mathcal{O}(Lk\log k)$, where k is the packet batch size and L is the packet data length. Different from previous works where the optimal performance of codes has been shown under asymptotic assumption, TC enjoys near-zero redundancy even under non-asymptotic settings for small-moderate number of packets. These features make TC suitable for practical implementation in battery-constrained devices in IoT, D2D and M2M network paradigms to achieve scalable reliability, and minimize latency due to its low decoding delay. TC is a non-linear code, which is encoded using the simple shift and XOR addition operations, and decoded using the simple back-substitution algorithm. Although it is nonlinear code at the packet level, it remains linear code when atomized at the bit level. We use this property to show that the back-substitution decoder of TC is equivalent to the Belief Propagation (BP) decoder of LT code. Therefore, TC can benefit from rich prolific literature published on LT code, to design efficient code for various applications. Despite the equivalency between the decoders of TC and LT code, we show that compared to state-of-the-art optimized LT code, TC reduces the redundancy of LT code by 68%–99% for k reaching 1024.

1. Introduction

Wireless transmission is inherently unreliable due to the imperfect wireless Medium Access Control (MAC) protocol which inevitably leads to packet collision, and bit errors caused by channel impairments such as Additive White Gaussian Noise (AWGN) and multipath propagation. Due to deep fading, shadowing, and handoffs, burst errors of long strings of bits in sequence may occur.

It has been shown that by adopting Forward Error Correction (FEC) coding at packet level, known as erasure coding, the total number of transmissions and latency can be reduced compared to retransmission-based Automatic Repeat Request (ARQ) mechanism [1]. In an erasure coding scheme, the receiver can recover k source packets from n coded packets, where $n = (1 + \varepsilon) \cdot k$, and $\varepsilon \geq 0$ is known as the reception redundancy. The idea of erasure coding stemming from communication systems has also been used in distributed data storage systems in resource-constrained wireless networks [2].

Erasure codes such as Raptor code have been incorporated in various transmission standards such as the Multimedia Broadcast Multicast Service (3GPP-MBMS) for multicasting file delivery and streaming applications, and in Digital Video Broadcasting (DVB) for multicasting live videos [3]. Fountain codes have been used as part of deployment by major IPTV providers in Asia and Europe, and satellite system providers in the USA [4].

The Consultative Committee for Space Data Systems (CCSDS), a consortium of government space agencies, has recommended the application of erasure codes in near-earth and deep-space communication [5]. In space communication, fading duration ranging between 2 and 6 ms has been recorded, and adverse weather conditions can last from a fraction of an hour to several hours, leading to very long erasures.

Despite the well-documented usefulness of various erasure codes and several commercial adoptions, the computational complexities of encoding, and decoding in particular have been identified as its major trade-off [5–9]. In an implementation of a cooperative transmission protocol for video streaming using Random Linear (RL) coding over $GF(2^8)$ on Samsung Galaxy Nexus, it is shown that the battery usage is dominated by the encoding-decoding operations, which accounts for up to 50% of the total energy cost in a WiFi network [9].

The encoding-decoding cost of erasure coding will become pressing for a green communication-aware next generation 5G network which

* Corresponding author.
E-mail address: jala0001@e.ntu.edu.sg (J. Qureshi).

https://doi.org/10.1016/j.dcan.2022.12.006
Received 12 September 2020; Received in revised form 27 October 2022; Accepted 7 December 2022
Available online 14 December 2022

needs to achieve an energy reduction of 90% [10]. Next generation wireless networks and smart cities need to support an enormous number of battery-constrained devices with low-end CPUs in the Device to Device (D2D), Machine to Machine (M2M) and the Internet of Things (IoT) network paradigm [10]. Cisco has projected that global mobile data traffic will increase sixfold from 2017 to 2022 [11].

Computationally expensive encoding and decoding processes may also lead to the infeasibility of erasure coding in applications that do not tolerate delay [12]. In a testbed implementation of downloading a 30 MB file, the large decoding delay has been attributed as the primary reason for the longer download time with the BitFountain protocol, which integrates fountain codes with the BitTorrent protocol, when compared to the standard BitTorrent [12].

Erasure codes such as Reed-Solomon (RS) and RL codes are characterized by the high encoding and decoding computational complexities of $\mathcal{O}(k^2L)$ and $\mathcal{O}(k^3+k^2L)$ respectively, where L is the length of the data packet. For the decoding complexity, the term k^3 is the complexity of inverting the coding vector matrix using Gaussian elimination, and k^2L is the complexity of multiplying the coded packets with the inverted matrix. RS codes are optimal codes [8], while RL codes over large field size have near-zero redundancy [7].

To achieve scalable linear order-of-magnitude encoding-decoding computational complexities, a new class of erasure codes known as fountain codes were proposed [1]. In particular Luby-Transform (LT) fountain codes have linear time encoding and decoding complexities of $\mathcal{O}(k\text{Llog } k)$ [5].

However, LT code has high reception redundancy in practical implementation as its optimal performance has been demonstrated in asymptotic settings [1,3]. Even optimized LT code has high redundancy of at least 40% for small packet batch size of $5 \leq k \leq 20$, as reported in Ref. [13], and redundancy of 9% for $k = 1024$, as reported in Ref. [14].

Standardized Raptor codes such as the R10 and RQ code have been the most successful codes so far. However, standardized Raptor codes are patented and owned by Qualcomm Technologies Inc. According to the inventors of fountain codes, this is why fountain codes are not widely used [4].

Nearly linear time complexities of $\mathcal{O}(k\log k)$, $\mathcal{O}(k\log(1/\varepsilon)$ and $\mathcal{O}(k^{1.5})$, which is slightly higher than that of linear time, are called linear time in erasure coding papers [1,3,5,15]. Therefore, in this paper we continue to refer to such complexities as linear time for consistency, following the convention used in the previous research papers. Our contribution is as follows.

The discussion so far shows that the computational complexities and redundancy of erasure coding had been studied in isolation. A departure from linear encoding approach for erasure coding is warranted to solve the unresolved trade-off problem it suffers from.

In this paper we present two important results. First, we propose Triangular Code (TC). As a non-linear code, it is encoded using the simple shift and XOR addition operations, and decoded using the simple back-substitution algorithm. When sparse coding vectors are used, the linear time coding-decoding complexities of TC is $\mathcal{O}(k\text{Llog } k)$, while the redundancy is close to zero even for a small number of packets. Unlike standardized Raptor codes, which is patented, TC can be widely adopted in battery-constrained devices due to its ideal features.

Second, we show that the back-substitution decoder of TC and the Belief Propagation (BP) decoder of LT code [5] are equivalent through a polynomial time mapping scheme. When the packets are atomized at bit-level, we further show that TC achieves a very good approximation of the LT code encoder. By showing that the back-substitution decoder and BP decoder are equivalent, future research work on coding schemes using the shift and XOR operations can converge in one direction.

Besides, as there is rich literature on LT code, TC can benefit from the published important results based on the equivalency of LT code and TC. For instance, the result of adopting LT code for deep-space communication in Ref. [5] can be used to design a TC based encoding scheme with much lower redundancy for deep-space communication. We will show later in Section 6.4, that TC reduces the redundancy of optimized LT code by as much as 68–99% for $k \leq 1024$.

TC has desirable features of being systematic code. Systematic code enjoys lower redundancy and computational complexities than non-systematic code [16]. Therefore, it is adopted in practical applications such as wireless video streaming [9].

Organization: The rest of the paper is organized as follows. We first present preliminaries, and an overview of various coding schemes in Section 2. The design of TC is explained in Section 3. The equivalency between the TC decoder and LT code decoder is shown in Section 4. The numerical result and simulation of the proposed technique are shown in Section 6. We then conclude with the main contribution of our work in Section 7.

2. System model

In this paper we consider the problem of reliably transmitting a set of source packets $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ to m receivers over a lossy wireless channel with the packet erasure probability of p_e using fountain erasure coding. Each data packet P_i is treated as a tuple of L bits and given as $P_i = [b_{i,1}, b_{i,2}, \ldots, b_{i,L}]$. Bit $b_{i,j}$ denotes the j^{th} bit of the i^{th} packet. A v^{th} coded packet is denoted by $C_v = [x_{v,1}, x_{v,2}, \ldots, x_{v,B}]$, and has the length of B bits, $B \geq L$. For encoding using linear code $B = L$, bits denoted by $b_{i,j}$ are known as source bits, whereas bits denoted by $x_{v,l}$ are known as coded bits.

The degree d of a coded packet is the number of source packets XOR added to generate the coded packet. The average sparsity, i.e., the number of source packets used to generate a coded packet is denoted by μ. We call a packet *innovative* if it is linearly independent of the packets a receiver has already received.

The term $E[\delta_p]$ denotes the expected number of redundant packets a receiver needs to receive before successful decoding. The maximum number of bits by which a source packet is shifted is represented by G_s, and $E[\delta_b]$ denotes the average number of redundant zero bits added to a packet due to the shift operation. We will introduce the definition of average redundancy ε for TC later in Section 6.

We study the design of erasure codes with ideal fountain codes properties in this paper. An ideal fountain code is characterized as being rateless and efficient with linear time complexities. As a rateless code, it should be able to generate a virtually unlimited supply of encoded packets on-the-fly analogous to a fountain which can continuously supply a steady stream of water drops. It should be efficient so that the encoded packet can be generated independently, and the decoder should be able to recover the k source packets after receiving *any* n packets, with minimal redundancy. This eliminates the need to frequently collect feedback frames.

In this section we present a brief overview of erasure coding schemes presented in literature to minimize the energy cost of coding. The linear time complexity of LT code is achieved by keeping the coding vector sparse with an average degree of $\log k$. Based on the balls-in-bin problem with large values of k, the average degree of a coded packet should not be lower than $\log k$ if all the k source packets need to be decoded [1]. The design of Raptor codes [3] was motivated to keep the average degree of the coded packet constant. The idea of Raptor codes is to precode a small number of redundant packets using a dense system of equations, and then an outer code is used to generate coded packets through XOR, which adds redundant and input packets using a weakened LT code.

Standardized Raptor codes such as R10 code enjoy negligible redundancy [3], with decoding complexity of approximately $k^{1.5}+4.6 \cdot Lk$, where $k^{1.5}$ is the complexity of inverting a smaller submatrix, and $4.6 \cdot Lk$ is the complexity of performing XOR addition given that the average degree of a R10 coded packet is 4.6 [3,15]. The lower decoding cost in R10 is achieved by designing a code which can be decoded using Gaussian elimination on a small submatrix with columns proportional to $\sqrt{k}$, together with the BP decoding algorithm.

Since standardized Raptor codes are patented, they cannot be ubiquitously adopted. Another limitation of R10 code is that it can only support 8192 source packets at the most. This limitation comes from the design of degree distribution. Once the number of input packets exceeds these limits, the decoding failure probability gradually increases.

The use of systematic RL code has been shown to minimize the reception redundancy and decoding complexity [16]. In a systematic code the transmitter first transmits k source packets, and then in the second phase broadcasts encoded packets, hence the decoding operation needs to be performed on a much smaller number of $k \cdot p$ encoded packets, minimizing the decoding cost.

A new class of rateless RS code based on the Cauchy matrix is proposed in Ref. [8] with zero redundancy and decoding complexity of $5 \cdot k^2+k^2L$ steps. Such a rateless RS code is suitable for implementation for small packet size of $k \leq 200$ in which fountain codes have high redundancy, and the linear time complexity of fountain codes is mitigated by additional time to receive and decode redundant packets.

3. TC

The design and name of TC are inspired by the idea of designing code in which the system of linear equations can be represented by a triangular or row echelon form matrix. Such a matrix can be inverted by using back-substitution with a complexity of $\mathcal{O}(k^2)$. By doing so, TC does not need to perform the triangularization procedure in Gaussian elimination which has a complexity of $\mathcal{O}(k^3)$ [17]. In this way, the receiver can immediately perform back-substitution which significantly reduces the decoder complexity. A proof-of-concept for TC was first proposed in Ref. [18] for the index coding problem, and further developed in [19, Chapter 5]. In this paper, we extend the idea presented in Ref. [18] to design variants of TC, which makes it suitable as a rateless fountain code with linear time complexity.

We first motivate the use of coding over $GF(2)$ for TC. Coding over $GF(2)$ involves the relatively simple XOR addition, whereas coding over $GF(q > 2)$ requires both multiplication and XOR addition operations. In an implementation of RL code on smartphone platform, it was shown that coding over $GF(2)$ consumed 0.06% of battery for every GB of data for the encoding and decoding operations, whereas encoding and decoding over $GF(256)$ consumed 5.06% and 3.6% for every GB of data, respectively [9].

3.1. Motivating example

We show the limitation of $GF(2)$ linear code with simple toy example. Consider a transmitter multicasting two packets P_1 and P_2 to three receivers R_1, R_2 and R_3. Let us assume that P_1 is only received by R_1, and P_2 is only received by R_2. The only coded packet which the transmitter can then linearly generate over $GF(2)$ is $P_1 \oplus P_2$, which we assume is received only by R_3. We illustrate the packets received by each receiver as follows:

$$\begin{cases} R_1 : P_1 \\ R_2 : P_2 \\ R_3 : P_1 \oplus P_2 \end{cases} \tag{1}$$

It is easy to verify that now the transmitter cannot generate a coded packet over $GF(2)$ which will be innovative (i.e., linearly independent) for all the three receivers.

When restricted to linear coding, the transmitter can only generate an innovative packet for all receivers when the source packets are coded over a larger field size. However, such a coded packet will trigger the computationally expensive Gaussian elimination at R_3 for decoding. Our trick around the problem of generating an innovative packet while avoiding Gaussian elimination is to add redundant zero bits at the head-tail of source packets before performing XOR addition.

The process of adding redundant "0" bits at the tail of the packet can be mathematically described by the real field multiplication of the bitstream tuple denoted by P_i with $2^{\ell_{i,v}}$, where $\ell_{i,v}$ denotes the number of redundant "0" bits added at the tail of packet P_i before XOR addition to generate coded packet C_v. However, in practice this process is implemented by using the simple shift operation, instead of performing the multiplication. After the shift, with the rightmost bits (tail) of all the source packets aligned, the bits are XOR added to generate the coded packet.

Returning to our earlier example, the sender can now multicast an innovative packet by encoding it as $C_3 = 2 \times P_1 \oplus P_2$. An illustration of the packet C_3 is shown in Fig. 1, where the coded bit $x_{3,2}$ denotes the second coded bit generated by XOR addition of the second bit $b_{1,2}$ of the source packet P_1, with the first bit $b_{2,1}$ of packet P_2.

Assuming that R_1 already has P_1, as shown in Eq. (1), receives C_3 can recover P_2 by adding a redundant zero bit at the tail of P_1 and then XOR by adding it with C_3. Afterwards, the redundant bit at the head of the packet (i.e., the most significant bit) is discarded.

We illustrate through a set of linear equations shown in Eqs. (2a)-(2c) how the first three source bits of P_2 be decoded, which is in fact equivalent to the back-substitution process of Gaussian elimination.

$$b_{1,2} \oplus x_{3,2} = b_{2,1} \tag{2a}$$

$$b_{1,3} \oplus x_{3,3} = b_{2,2} \tag{2b}$$

$$b_{1,4} \oplus x_{3,4} = b_{2,3} \tag{2c}$$

Similarly, R_2 with P_2 can recover P_1 by adding a redundant zero bit at the head of P_2, and then XOR with C_3. After the addition process, the redundant bit at the tail of the packet (i.e., the least significant bit) can be discarded.

We illustrate how receiver R_3 can decode the first two source bits of P_1 and P_2 from $P_1 \oplus P_2$ and C_3, as shown in Eqs. (3a)-(3d). Coded bit $x_{3,1}$ is equal to $b_{1,1}$. This bit is added in the first coded bit of $P_1 \oplus P_2$ to recover $b_{2,1}$. The newly recovered bit $b_{2,1}$ is then added in the coded bit $x_{3,2}$ to recover $b_{1,2}$. This process then continues iteratively to recover all the source bits of both the packets.

$$x_{3,1} = b_{1,1} \tag{3a}$$

$$(b_{1,1} \oplus b_{2,1}) \oplus b_{1,1} = b_{2,1} \tag{3b}$$

$$x_{3,2} \oplus b_{2,1} = b_{1,2} \tag{3c}$$

$$(b_{1,2} \oplus b_{2,2}) \oplus b_{1,2} = b_{2,2} \tag{3d}$$

So far, in our decoding examples, we have illustrated how a TC generated coded packet along with a source packet, or $P_1 \oplus P_2$ can be used to decode the lost packet. As TC is a rateless code, it can generate additional coded packets on the fly. Fig. 1 illustrates four such possible TC packets generated using P_1 and P_2.

As an example, we now show how a decoder with two TC generated packets C_3 and C_4 can decode all the bits of source packets P_1 and P_2. Coded bits $x_{3,1}$ and $x_{4,1}$ are equal to $b_{1,1}$ and $b_{2,1}$, respectively. The knowledge of $b_{1,1}$ and $b_{2,1}$ can be used to decode $b_{1,2}$ and $b_{2,2}$ from $x_{3,2}$ and $x_{4,2}$ respectively, as shown in Eqs. (4a)-(4d). Knowledge of newly decoded bits $b_{1,2}$ and $b_{2,2}$ can then be used to decode $b_{1,3}$ and $b_{2,3}$. The process continues iteratively until all the $L = 4$ bits of both source packets are decoded.

$$x_{3,1} = b_{1,1} \tag{4a}$$

$$x_{4,1} = b_{2,1} \tag{4b}$$

$$b_{1,1} \oplus x_{3,2} = b_{1,2} \tag{4c}$$

$$b_{2,1} \oplus x_{4,2} = b_{2,2} \tag{4d}$$

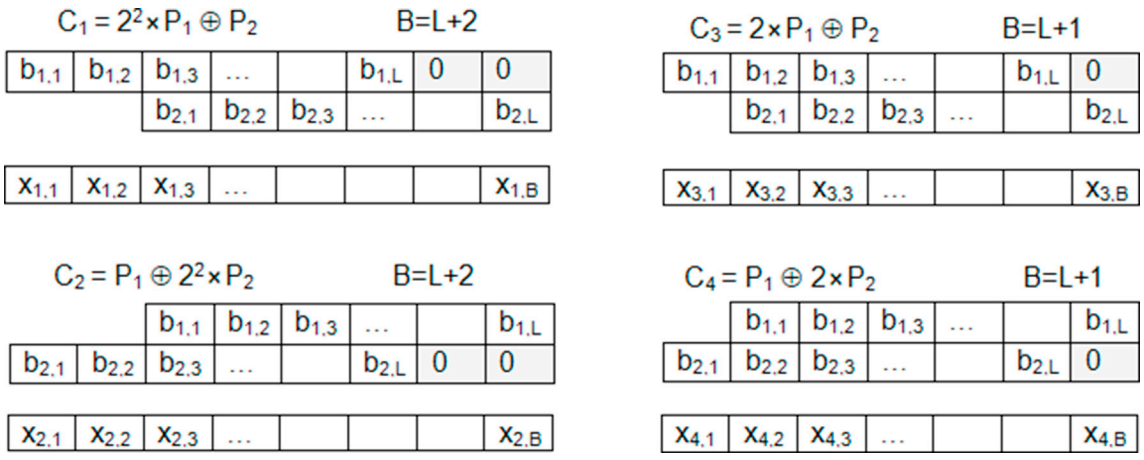

Fig. 1. Four TC generated packets (C_1, C_2, C_3 and C_4). Along with P_1, P_2 and $P_1 \oplus P_2$, it can be verified that any two randomly picked packets from the set of these seven packets can be successfully decoded using back-substitution at bit-level.

For the decoding process using C_3 and C_4, as our proposed decoding scheme involves solving a system of $2B$ linear equations, these can be represented using an augmented matrix as shown in Table 1, assuming that the length L of both packets is four bits. As the matrix is in a row echelon form, these equations can be simply solved with back-substitution [17], requiring a minimum of four XOR addition steps as four of the equations have only one unknown variable.

We will show later in Section 3.3 that there are efficient data structures which can be used to minimize the memory overhead of representing matrix form.

3.2. TC encoder

TC encoding operation consists of three major steps. First, the encoder needs to select the source packets from $\mathcal{P}$; second, for each source packet, the number of bit positions that would be shifted is assigned a value; finally, the packets are XOR added. As we will show in the next two sections, there are various kinds of machinery to perform the first two steps of TC encoding, with each leading to different variants of the TC scheme.

Information about the source packets selected for encoding, and the shift applied to each source packet is included in the header of the coded packet. We will show later in Section 5.3 that this information can be included in the packet header with minimal overhead.

3.3. TC decoder

A pseudo-code of the TC decoder is shown in Table 2. The decoding operation relies on back-substitution, which is a well-known method to solve a system of liner equations in row-echelon form [17]. We have illustrated in Table 1 how this method can be applied when the coded bits are mapped to a matrix of minimum size $kL \times kL$ augmented with a matrix of coded bits. However, practically adopting such a scheme would be prohibitive as it requires a minimum memory overhead of k^2L^2 to store such a matrix.

Table 1
Augmented matrix representing system of linear equations of coded bits for the TC packets C_3 and C_4, $L = 4$, in row echelon form.

$$\left(\begin{array}{cccccccc|c} b_{1,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_{3,1} \\ 0 & b_{1,4} & 0 & 0 & 0 & 0 & 0 & 0 & x_{4,5} \\ 0 & 0 & 0 & b_{1,2} & b_{2,1} & 0 & 0 & 0 & x_{3,2} \\ 0 & 0 & 0 & 0 & b_{2,1} & 0 & 0 & 0 & x_{4,1} \\ 0 & 0 & b_{1,3} & 0 & 0 & b_{2,2} & 0 & 0 & x_{3,3} \\ b_{1,1} & 0 & 0 & 0 & 0 & b_{2,2} & 0 & 0 & x_{4,2} \\ 0 & b_{1,4} & 0 & 0 & 0 & 0 & b_{2,3} & 0 & x_{3,4} \\ 0 & 0 & 0 & b_{1,2} & 0 & 0 & b_{2,3} & 0 & x_{4,3} \\ 0 & 0 & b_{1,3} & 0 & 0 & 0 & 0 & b_{2,4} & x_{4,4} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & b_{2,4} & x_{3,5} \end{array}\right)$$

Table 2
TC back-substitution decoder pseudo-code.

Input - Triangular coded packets, and coding vectors.
Output - Decoded bits.
do
1 Search for an equation with one unknown variable.
2 Decode the source bit, $b_{i,j} = x_{d,c}$.
3 Substitute (XOR add) the decoded bit using for loop.
while {there exist an equation with one unknown variable}

We instead use table-based adoption which captures the information of an augmented matrix in a much compact form. This idea relies on the fact that there are many zeros in each row of the augmented matrix, which unnecessarily increases the memory overhead.

In our proposed implementation, each row of table T_1 stores the non-zero coefficients of the equation for each of the kB coded bits. A count of the unknown variables in each row of T_1 is also maintained in an array A_u. Another table T_2 stores the row id in which each of the kL source bits is stored. An illustration of the information stored in the tables at the start of decoding is shown in Table 1 and the augmented matrix is shown in Fig. 2.

As we had discussed earlier, for large k, the minimum sparsity of a coded packet has to be log k for the decoder to decode all packets. Therefore, for $\mu = \log k$, the number of entries in each row of T_1 will be $\log k$. Hence, the total number of steps required to construct tables T_1 and T_2 is $\mathcal{O}(kL\log k)$.

With tables T_1 and T_2 being constructed, data from T_1 and A_u can be used to complete step 1 of the decoding process. Once the decoder identifies a row (which represents an equation) in which there is one source bit (one unknown variable), then this can be used to decode a source bit from the coded bit. After step 2, with information from T_2, the decoded source bit is XOR added (substituted) in all coded bits in T_1 generated by XOR adding the decoded source bits. After each substitution, the corresponding array element A_u is decremented.

As the sparsity of the coded packet is $\log k$, each decoded bit would be substituted for an average of $\log k$ times. As there are kL source bits in total, the total complexity of step 3 is $\mathcal{O}(kL\log k)$.

Table T_1

Index	A_u	Coded bit	Source bit	
1	1	$x_{3,1}$	$b_{1,1}$	
2	1	$x_{4,5}$	$b_{1,4}$	
3	2	$x_{3,2}$	$b_{1,2}$	$b_{2,1}$
4	1	$x_{4,1}$	$b_{2,1}$	
5	2	$x_{3,3}$	$b_{1,3}$	$b_{2,2}$
6	2	$x_{4,2}$	$b_{1,1}$	$b_{2,2}$
7	2	$x_{3,4}$	$b_{1,4}$	$b_{2,3}$
8	2	$x_{4,3}$	$b_{1,2}$	$b_{2,3}$
9	2	$x_{4,4}$	$b_{1,3}$	$b_{2,4}$
10	1	$x_{3,5}$	$b_{2,4}$	

Table T_2

Index	Source bit	Coded bit	
1	$b_{1,1}$	$x_{3,1}$	$x_{4,2}$
2	$b_{1,2}$	$x_{3,2}$	$x_{4,3}$
3	$b_{1,3}$	$x_{3,3}$	$x_{4,4}$
4	$b_{1,4}$	$x_{4,5}$	$x_{3,4}$
5	$b_{2,1}$	$x_{3,2}$	$x_{4,1}$
6	$b_{2,2}$	$x_{3,3}$	$x_{4,2}$
7	$b_{2,3}$	$x_{3,4}$	$x_{4,3}$
8	$b_{2,4}$	$x_{4,4}$	$x_{3,5}$

Fig. 2. Tables T_1 and T_2 are used to store the system of linear equation information in an augmented matrix in a compact way.

3.4. Memory overhead

The memory overhead of a TC decoder is given by the size of T_1 and T_2 which is approximately $2 \cdot kL \log k$. Once the data from a coded packet is copied in these two tables, the coded packet can be immediately discarded, so the coded packets do not contribute towards the memory overhead. In this way, the order-of-magnitude memory overhead of TC is similar to that of LT code.

In comparison, RL code has memory overhead of k^2+kL, as the RL decoding operation is performed by inverting a $k \times k$ matrix of coding coefficients, and then multiplying it with k packet, each of which is L bits long.

4. Relation to LT decoder

In this section, we show that the back-substitution decoder of TC is equivalent to the BP decoder of LT code. We further show that the TC encoder can achieve a very good approximation of the LT encoder. Toy examples which illustrate the encoding and decoding operations of LT codes can be found in Refs. [1,3].

4.1. LT decoder

In LT encoding, degree d is randomly chosen from a degree distribution $\rho(d)$, which is a probability distribution, where $1 \leq d \leq k$. The degree d is used to decide how many source packets should be selected for encoding. Afterwards, d source packets are then randomly and uniformly selected from $\mathcal{P}$, and XOR added.

LT BP decoding uses a bipartite graph known as a Tanner graph to demonstrate the decoding process. However, in practice, an implementation of the LT BP decoder would require the Tanner graph to be represented with a 2-dimensional array (matrix) or an efficient data structure which represents such a matrix.

Fig. 3 illustrates an example of a Tanner graph. One of the disjoint sets of vertices (represented using round nodes) are known as the source node for the source packet, and the other disjoint vertices (represented as square nodes) as check nodes for the coded packets.

Each check node is connected by an edge to each source node that represents XOR addition. If there is a check node with one neighboring source node, then the value of the neighboring source packet can be decoded. Performing XOR addition of the value of the decoded source packet to all the coded packets which are connected with it by edges, those edges will be removed after the addition.

The process of removing an edge may result in one of the check nodes being left with one neighbour, in which case the corresponding check node can further be used to decode a new source packet. This process continues iteratively until all the source packets are decoded. If the decoder is unable to decode all the packets, then this leads to decoding failure, and the decoder would need additional coded packets to recover from the failure.

Lemma 1. *The back-substitution decoder for TC generated coded packets is equivalent to the BP decoder of LT code, and this mapping can be performed in polynomial time.*

Proof. Both BP decoding and TC decoding are based on the idea of searching for a degree-one coded symbol, which is first decoded and then

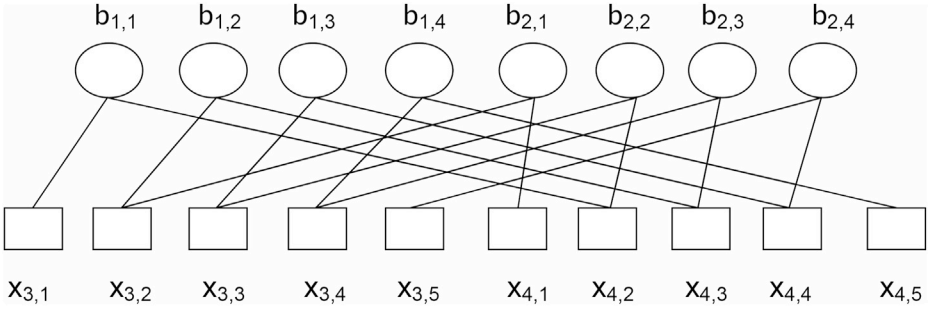

Fig. 3. An instance of Tanner graph representation for LT BP decoding, equivalent to the system of linear equation TC back-substitution decoding shown in Table 1.

back-substituted. The substitution process results in the release of a new degree-one coded symbol (known as ripple in LT code literature), due to which the process continues iteratively until all the symbols are decoded.

First, we atomize each of the source and coded packets used in LT code into L bits. It is intuitive to see that these atomized bits can be represented in a Tanner graph representation, much like how the packets used in LT code were represented with the Tanner graph. Next, we show that the information in such a Tanner graph can be represented using tables T_1 and T_2, as shown earlier in polynomial time.

The information about the neighboring nodes of the coded bit $x_{v,l}$ is stored in table T_1. Similarly, the information about the neighboring nodes of the source bit $b_{i,j}$ is stored in table T_2. As each of the nodes in the Tanner graph has an average of $k \log k$ neighbors, a total of $2\,kL \log k$ steps are required to fill these two tables.

Once the information about the source and coded packets is filled in tables T_1 and T_2, the information can then be used to perform decoding as described in Table 2 for TC generated coded packets. Therefore, it is possible to decode LT code using TC decoder in $\mathcal{O}(kL\log k)$ time complexity. This completes the proof.

The illustration given in Fig. 3 also shows how the augmented matrix shown in Table 1 can be mapped to a Tanner graph, and vice versa.

4.2. Approximation of LT encoder using LT-TC

We now show how a variant of TC, which we call LT-TC, can be designed in such a way that it is a good approximation of LT encoder when the packets (source nodes in Tanner graph) are atomized at bit level. This approximation is achieved for all types of degree distribution proposed in literature on LT code.

Like the LT code encoder, in LT-TC encoder, after d source packets are selected from $\mathcal{P}$, each of the d packets is randomly shifted by a value uniformly and randomly selected from G_s, and then XOR added. As we will show later in Section 6.3, a suitably optimized value of G_s does not exceed few bytes.

The minimum length of an internet packet is 40 bytes [20]. If a relatively small value ($\approx$ 2 bytes) of G_s is used, then at bit level, the degree of most of the coded bits will be d, while the degree of a small number of coded bits will be less than d due to the shift operation. Hence the actual degree distribution $\rho(d)$ of LT-TC at bit level will be a good approximation of the original $\rho(d)$ designed for LT code.

4.3. LT code and LT-TC performance comparison

LT-TC can be decoded at packet level using BP decoding, and when decoding at packet level fails, back-substitution decoding at bit level can be triggered then. A comparison of LT-TC with the optimized LT code is shown in Table 3. For fair comparison, LT-TC uses the same degree distribution as the optimized LT code with which it is compared.

From the table, it can be seen that for $k = 512$, LT-TC ($G_s = 2$) reduces the packet level redundancy from 61.4 to 38.75 (corresponding to a 37% reduction) at the overhead cost of an average of 1.24 redundant bit added to a packet. By using LT-TC ($G_s = 8$), packet redundancy can be further reduced by 55% at the trade-off cost of a higher number of 4.31 redundant bits added to each packet. These results are independent of the packet length L.

For LT code, the packet redundancy ratio $\frac{E[\delta_p]}{k}$ gradually decreases as the number of source nodes (equal to k) in the Tanner graph increases [3, 14]. As the value in the denominator of the ratio increases, packet reception redundancy decreases.

As decoding takes place at bit-level in LT-TC, the number of source nodes (and hence the value in denominator of the ratio) in the Tanner graph of LT-TC is equal to kL. Due to the packet length multiplier L, which has a minimum length of 320 bits for internet data packet [20], this redundancy ratio decreases much faster for LT-TC compared to LT code for the same value of k. Hence LT-TC achieves a significantly lower packet redundancy ratio even for small values of k, at a small trade-off cost of redundant bits added to packets.

Table 3
Comparison of optimized LT code with LT-TC.

Coding Scheme	Redundancy	$k = 8$	$k = 64$	$k = 512$
Optimized LT code	$E[\delta_p]$	3.57 [13]	18.7 [6]	61.4 [14]
LT-TC ($G_s = 2$)	$E[\delta_p]$	0.83	1.1	38.75
	$E[\delta_b]$	0.71	4.78	1.24
LT-TC ($G_s = 8$)	$E[\delta_p]$	0.53	2.75	27.71
	$E[\delta_b]$	2.36	3.81	4.31

5. TC variants

In the previous section, an LT variant of TC was described to show the equivalency of encoder and decoder of LT code and TC. In this section, we shall present two variants of TC, each with different machinery for the first two steps of TC encoding as mentioned in Section 3.2.

We first show that the Vandermonde generator matrix used by the RS code can be used to generate TC packets where the decoder can recover k source packet after receiving any k packets, i.e., the packet redundancy of Vandermonde TC is zero. However, the disadvantage of Vandermonde TC is that the bit redundancy of every successive coded packet due to the shift operation increases by k bit, hence the k^{th} generated coded packet has bit redundancy of approximately k^2 bits. With the above setup, we propose an isomorphic Vandermonde generator matrix for TC which reduces the bit redundancy by a factor of 50%, which we call IV-TC.

While Vandermonde TC code enjoys zero packet redundancy, it has high bit redundancy. To address this issue, we then propose a randomized variant of TC, where the degree d and bit-shifts G_s are randomly selected, which we call Randomized-TC (R-TC). Both variants are described in detail in the following.

5.1. Systematic isomorphic Vandermonde TC

Lemma 2. *Any k TC packets received by a decoder which are generated using the generator matrix $\mathcal{M}_2$ given by the concatenation of an identity matrix $\mathcal{I}$ and the variant of Vandermonde matrix $\mathcal{M}_1$ are nonsingular.*

$$\mathcal{M}_2 = \begin{pmatrix} \mathcal{I} \\ \mathcal{M}_1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 1 & 2 & 2^2 & \dots & 2^{k-1} \\ 1 & 2^2 & 2^4 & \dots & 2^{2(k-1)} \\ \vdots & & \ddots & & \vdots \\ 1 & 2^k & 2^{2k} & \dots & 2^{k(k-1)} \end{pmatrix}$$

Proof. To prove the lemma, we consider three cases, where the k packets are picked from (i) $\mathcal{I}$, (ii) $\mathcal{M}_1$, or (iii) both $\mathcal{I}$ and $\mathcal{M}_1$. For the first case, the nonsingular property of the matrix is because matrix $\mathcal{I}$ is a triangular matrix with non-zero entries in all diagonal entries. For the second case, the submatrix $\mathcal{M}_1$ is non-singular due to the property of the Vandermonde matrix [21].

For the third case, consider that a row from submatrix of $\mathcal{I}$ with the element 1 in the j^{th} column is used to turn the j^{th} column of the submatrix $\mathcal{M}_2$ into a column with $k-1$ zeros and one non-zero entry, with elementary row operations. Afterwards, the j^{th} column cannot be turned into an all-zero column due to the presence of only one non-zero element. Hence an arbitrary submatrix of $\mathcal{I}$ does not lead to reduction in rank of the concatenated submatrix, and the decoder can decode the k source packets after receiving any combination of k source and coded packets. This completes the proof.

We emphasize here that the elements of $\mathcal{M}_2$ are real numbers and not finite field elements, hence the proof which we have provided here applies only to non-linear code. In Ref. [21], it has been shown that the

matrix structure $\mathcal{M}_2$ (when its elements are from a finite field) cannot be used to generate linear code with Maximum Distance Separable (MDS) property. In an MDS code, any k received coded packets are guaranteed to form a k rank matrix.

We now present a generator matrix for isomorphic Vandermonde TC which reduces the redundant bits added to source packet due to the shift operation by 50%.

Proposition 1. *The determinant of the following isomorphic Vandermonde matrix is non-zero.*

$$\mathcal{M}_3 = \begin{pmatrix} 1 & 2 & \dots & 2^{k-2} & 2^{k-1} \\ 2^{k-1} & 2^{k-2} & \dots & 2 & 1 \\ 1 & 2^2 & \dots & 2^{2(k-2)} & 2^{2(k-1)} \\ 2^{2(k-1)} & 2^{2(k-2)} & \dots & 2^2 & 1 \\ \vdots & & \ddots & & \vdots \\ 1 & 2^{\frac{k}{2}} & \dots & 2^{\frac{k}{2}(k-2)} & 2^{\frac{k}{2}(k-1)} \\ 2^{\frac{k}{2}(k-1)} & 2^{\frac{k}{2}(k-2)} & \dots & 2^{\frac{k}{2}} & 1 \end{pmatrix}$$

Proof. Without loss of generality, we rewrite 2^t as a_t. Consider the determinant D of an arbitrary $k \times k$ matrix $A_k^{'}$ after row-interchange as follows,

$$D = |A_k^{'}| = \begin{vmatrix} 1 & a_1 & \dots & a_1^{k-2} & a_1^{k-1} \\ 1 & a_2 & \dots & a_2^{k-2} & a_2^{k-1} \\ \vdots & & \ddots & & \vdots \\ 1 & a_{k/2} & \dots & a_{k/2}^{k-2} & a_{k/2}^{k-1} \\ a_1^{k-1} & a_1^{k-2} & \dots & a_1 & 1 \\ a_2^{k-1} & a_2^{k-2} & \dots & a_2 & 1 \\ \vdots & & \ddots & & \vdots \\ a_{k/2}^{k-1} & a_{k/2}^{k-2} & \dots & a_{k/2} & 1 \end{vmatrix}.$$

We now add -1 times the first row of D to the first $k/2$ rows, and $-a_t^{k-1}$ times the first row to the $(k/2+t)^{th}$ rows. For clarity, we illustrate only four rows of the determinant. The determinant D is now given as

$$\begin{vmatrix} 1 & a_1 & \dots & a_1^{k-2} & a_1^{k-1} \\ 0 & a_2 - a_1 & \dots & a_2^{k-2} - a_1^{k-2} & a_2^{k-1} - a_1^{k-1} \\ \vdots & & \ddots & & \vdots \\ 0 & a_1^{k-2} - a_1^k & \dots & a_1 - a_1^{2k-3} & 1 - a_1^{2k-2} \\ 0 & a_2^{k-2} - a_1 a_2^{k-1} & \dots & a_2 - a_1^{k-2} a_2^{k-1} & 1 - a_1^{k-1} a_2^{k-1} \\ \vdots & & \ddots & & \vdots \end{vmatrix}.$$

Next, we multiply $-a_1$ times the $(k-1)^{th}$ column and then add it to the k^{th} column, multiply $-a_1$ times the $(k-2)^{th}$ column and then add it to the $(k-1)^{th}$ column, and so on until the 3^{rd} column is added. This gives

$$\begin{vmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & a_2 - a_1 & \dots & a_2^{k-3}(a_2 - a_1) & a_2^{k-2}(a_2 - a_1) \\ \vdots & & \ddots & & \vdots \\ 0 & a_1^{k-2}\left(1 - a_1^2\right) & \dots & a_1\left(1 - a_1^2\right) & 1 - a_1^2 \\ 0 & a_2^{k-2}(1 - a_1 a_2) & \dots & a_2(1 - a_1 a_2) & 1 - a_1 a_2 \\ \vdots & & \ddots & & \vdots \end{vmatrix}.$$

Except the first row, the first $k/2$ rows has a common multiple $(a_t - a_1)$. Row $k/2+1$ to row k has common multiple $(1 - a_1 a_t)$. The determinant can now be written as

$$D = \prod_{t=2}^{k/2}(a_t - a_1) \prod_{t=1}^{k/2}(1 - a_1 a_t)|A_{k-1}^{'}|,$$

where $|A_{k-1}^{'}|$ is given by

$$|A_{k-1}^{'}| = \begin{vmatrix} 1 & a_2 & \dots & a_2^{k-3} & a_2^{k-2} \\ 1 & a_3 & \dots & a_3^{k-3} & a_3^{k-2} \\ \vdots & & \ddots & & \vdots \\ a_1^{k-2} & a_1^{k-3} & \dots & a_1 & 1 \\ a_2^{k-2} & a_2^{k-3} & \dots & a_2 & 1 \\ \vdots & & \ddots & & \vdots \end{vmatrix}$$

Continuing this way, another $k/2-1$ 'first row'can be similarly removed using the steps we detailed earlier, with the determinant given as follows,

$$D = \prod_{u=1}^{k/2-1} \prod_{t=u+1}^{k/2} (a_t - a_u) \prod_{u=1}^{k/2} \prod_{t=1}^{k/2} (1 - a_u a_t) |A_{k-\frac{k}{2}}^{'}|,$$

where $|A_{k-\frac{k}{2}}^{'}|$ is now reduced to the determinant of the traditional Vandermonde matrix given as follows,

$$\begin{vmatrix} a_1^{\frac{k}{2}-1} & \dots & a_1 & 1 \\ a_2^{\frac{k}{2}-1} & \dots & a_2 & 1 \\ \vdots & \ddots & & \vdots \\ a_{k/2}^{\frac{k}{2}-1} & \dots & a_{k/2} & 1 \end{vmatrix} = \prod_{u=1}^{k/2-1} \prod_{t=u+1}^{k/2} (a_t - a_u)$$

The determinant D can now be expressed as

$$D = \left(\prod_{u=1}^{k/2-1} \prod_{t=u+1}^{k/2} (a_t - a_u) \right)^2 \prod_{u=1}^{k/2} \prod_{t=1}^{k/2} (1 - a_u a_t)$$

which will be non-zero if each of the numbers a_t is unique. Since $a_t = 2^t$, the matrix $\mathcal{M}_3$ will have a non-zero determinant if t and u are unique, and $t \neq u$. This completes the proof.

Theorem 1. *Any k TC packets received by a decoder which are generated using the generator matrix $\mathcal{M}_4$ given by the concatenation of an identity matrix $\mathcal{I}$ and isomorphic Vandermonde matrix $\mathcal{M}_3$ are nonsingular.*

$$\mathcal{M}_4 = \begin{pmatrix} \mathcal{I} \\ \mathcal{M}_3 \end{pmatrix}$$

Proof. Using the proof technique used in proving Lemma 2 and the result of Proposition 1, it follows that a systematic Isomorphic Vandermonde TC maintains the MDS code property, i.e., the decoder can recover all k source packets using any k coded packets generated by the matrix $\mathcal{M}_4$. This completes the proof.

5.2. Randomized TC

IV-TC has the attractive feature of maintaining MDS code property, which means that the coding vectors of any k received packets form a full-rank invertible matrix, but has a decoding complexity of $\mathcal{O}(k^2 L)$, and bit redundancy $E[\delta_b]$ of $\mathcal{O}(k^3)$, rendering IV-TC impractical for adoption when the value of k is not small.

In this section, we propose the design of Randomized TC (R-TC). In R-TC, the maximum number of redundant zero bits which are added in the source packet is fixed, which means G_s can maintain a fixed number

during encoding, and the average sparsity of the coded packet is kept as $\log k$.

The proposed code R-TC can generate a packet which is linearly dependent on a receiver due to the randomness of encoding. R-TC can also generate a packet which results in decoding failure if the decoder is unable to decode the bits using back-substitution, i.e., the packet may be linearly independent but requires Gaussian elimination to decode the bits. We will show later in Section 6.3 that such redundancy is negligible with respect to k. We now show that the packet level redundancy is a function of G_s with intuition.

Proposition 2. *For R-TC (and LT-TC) as $G_s \to \infty$, the packet level redundancy approaches $E[\delta_p] \to 0$.*

Proof. For large values of G_s (i.e., large shift), the proportion of packet overlap will decrease. As the proportion of such overlap decreases, an increasing number of coded bits will be equal to the XOR addition of a single source bit, which can be decoded immediately. As the proportion of such coded bits approaches kL, the probability of decoding failure will approach zero, resulting in $E[\delta_p] \to 0$. This completes the proof.

While an increasing value of G_s reduces the probability of decoding failure, it comes at the trade-off cost of increasing bit-level redundancy. Therefore, G_s needs to be suitably optimized to keep the overall redundancy minimal. It will be shown later that a redundancy of few packets, irrespective of the value of k, is sufficient to recover all k source packets even when the value of G_s corresponds to a few bytes.

5.3. Communicating coding vector to decoder

Now that the encoding procedure of all the variants of TC have been presented, we shall now show how the encoder communicates coding vector information to the decoder. The coding vector includes information about the source packets used to generate a coded packet, and the number of bits by which each of the source packet has been shifted.

As the decisions about the source packets and shift operation in LT-TC and R-TC are implemented with a randomized approach, the encoder can include the seed value used for the pseudo-random number generator in the packet header similar to fountain coding [1,6]. The decoder can then use the seed value to regenerate the coding vector.

Similarly, for IV-TC, each row of the coding vector has a predetermined fixed pattern due to the use of the Vandermonde matrix. It is sufficient for the encoder to communicate the row id with the decoder. The row id can be used to generate the coding vector for IV-TC. The overhead of including information about the seed value or row id in the packet header is negligible.

6. Numerical results

For linear code, the average redundancy is given as $\frac{n-k}{k}$. However, as TC is a non-linear code, a suitable definition of the transmission rate would need to include bit level redundancy $E[\delta_b]$ in addition to packet level redundancy $E[\delta_p]$.

Definition 1. For TC, the average redundancy ε, is given as follows,

$$\frac{kE[\delta_b] + E[\delta_p](L + E[\delta_b])}{kL}$$

For LT code, with $E[\delta_b] = 0$, the average redundancy ε can be simplified as $\frac{E[\delta_p]}{k}$.

6.1. Analytical model for R-TC and LT-TC

For coded packets generated by R-TC and LT-TC, the number of redundant bits is given by the difference of maximum and minimum shift applied to source packets used to generate the coded packet. This model is adopted since transmitting coded bits completely generated by XOR addition of redundant "0" bits carries no information and can be trimmed from the coded packet. For example, consider the coding vector $[2^2, 2^4]$, which means that four and two redundant "0" bits are added at the tail of P_1 and P_2 respectively, though the last two bits from both the packets can be trimmed, as those will only result in generating a coded packet given by XOR addition of two "0" bits.

A recursion based analytical model for decoding failure probability for finite-length LT code with a given degree distribution and number of coded packets n received by the decoder is given in Ref. [22]. Due to the recursion nature of the model, its computational complexity is $\mathcal{O}(k^2 \log k)$, where k denotes the number of source nodes in the Tanner graph. As decoding for TC takes place at bit level, the corresponding computational complexity of running the analytical model for TC will be $\mathcal{O}(k^2L^2 \log Lk)$. Therefore, an analytical model to determine the decoding failure probability for TC cannot run in reasonable time on a standard computer system even for small values of k and L.

We hence adopt simulation to determine the redundancy of R-TC and LT-TC in this paper. The simulation method is widely accepted in LT and Raptor codes literature to determine decoding failure probabilities [3,5,8,14].

6.2. Analytical model for systematic IV-TC

In a systematic IV-TC, the transmitter first transmits k source packets. The expected number of source packets received is given by $k(1 - p_e)$. The remaining kp_e packets which the decoder receives will be coded packets. The expected number of transmissions β needed before the receiver successfully receives kp_e packets follows geometric distribution and is given as follows,

$$\beta = \frac{kp_e}{1 - p_e}$$

The numbers of redundant bits added in every v^{th} and v+1st coded packets are $(v+1)(k-1)$, assuming d is an odd natural number (without losing generality). The total number of redundant bits T_v added in β coded packets is given by the sum of natural numbers up to $0.5 \cdot \beta$, each of which has a common multiple of $2(k-1)$, which is

$$T_v = 2(k-1) \times \sum_{v=1}^{0.5 \cdot \beta} v = \frac{\beta}{2}(k-1)\left(\frac{\beta}{2} - 1\right)$$

The 0.5 factor is because we only need to sum the redundant bits of v packets and then multiply it by two as v+1st packet has the same bit redundancy as v.

The average bit redundancy of each coded packet can then be given by $\frac{T_v}{\beta}$. As the receiver needs to collect kp_e coded packets, the total redundancy of systematic IV-TC for a given erasure probability p_e is

$$\frac{T_v kp_e}{\beta} = \frac{T_v}{1 - p_e}$$

6.3. Simulation

In this section, we use simulation to determine the average redundancy of the TC variants proposed in the paper. As R-TC introduces bit level redundancy $E[\delta_b]$, which is dwarfed by packets of larger length, results in this section are shown for both large and small packet lengths. We use packet lengths L of 40 and 1500 bytes. These lengths represent the range of packet length in the internet [20]. 40% of internet packets have the length of 40 bytes, while 20% of the internet packets are 1500 bytes long.

We use non-systematic R-TC for simulation as R-TC has the highest redundancy when the packets are transmitted in a non-systematic manner, as this introduces redundant bits in all the transmitted packets. Such an evaluation would provide the minimum reduction in redundancy when comparing TC with other rateless codes. In practical

implementation, the use of systematic R-TC will result in lower redundancy.

As discussed in Section 5.2, we first need to optimize the value of G_s, for which we evaluate the redundancy of R-TC for different representative values of G_s, as shown in Fig. 4. For non-systematic coding, the redundancy is not affected by channel erasure p_e, as all the received packets are coded. The result of the graph shows that as the value of L increases, the value of G_s should increase gradually to enjoy minimal average redundancy.

Using values of $G_s = 8$ for $L = 40$ bytes and $G_s = 32$ for $L = 1500$ bytes, we evaluate the performance of systematic R-TC for a lossy wireless network with $p_e = 0.4$. The same G_s values are also used in Section 6.4. A plot of the result is shown in Fig. 5(a). The result shows that R-TC can enjoy near-zero redundancy for packet length of practical importance.

For a very small packet length of $k \leq 50$, IV-TC should be used based on the channel erasure rate. A plot of IV-TC and R-TC for very small packet length is shown in Fig. 5(b). From the figure, it can be seen that the average redundancy for IV-TC increases very fast as k increases as its bit-level redundancy increases with the increase of $\mathcal{O}(k^3)$. Therefore, the encoding process should be switched to R-TC when the graph for IV-TC and R-TC intersect as shown in the figure. The figure shows that for $p_e \leq 40\%$, the redundancy of less than approximately 3% can be achieved for $L = 40$ bytes, while for $L = 1500$ bytes, the redundancy does not exceed approximately 1%.

6.4. Comparison with optimized LT codes

In this section, we compare the performance of our proposed TC with various state-of-the-art optimized LT codes [6,13,14]. In addition to the fact that unlike standardized Raptor codes which are patent-protected, LT codes are not patent-protected and have the same encoding-decoding complexities of $\mathcal{O}(kL\log k)$ as TC. This makes LT codes a suitable coding scheme for fair comparison with TC.

The reception redundancy ε of the optimized LT codes for various representative values of k is shown in Table 4. The values in the 'Reduction' columns represent the percentage reduction in reception redundancy due to the TC when compared with LT codes. A reduction value of 100% would correspond to an optimal code with $\varepsilon = 0$.

The table shows that for packets of small length of $L = 40$ bytes, compared to optimized LT codes, TC reduces the redundancy by 67.8%–96.4%. Whereas for large packet size ($L = 1500$ bytes), TC reduces the redundancy by 94.7%–99.8%. As the value of k increases, due to the shift operation of the reduction in redundancy gradually decreases. This is consistent with the previous result that the redundancy of LT decreases as k increases [14], hence the contribution of the shift operation in TC to further minimize redundancy gets less prominent as k increases.

For comparison reference with Raptor codes, the expected redundancy ε of standardized Raptor codes has been shown to be 1.96 using an analytical model [3]. This redundancy value has been shown to be independent of the number of packets k.

7. Conclusion

In this paper, we have demonstrated a non-linear erasure code which we call TC. We showed a polynomial time mapping scheme which demonstrates the equivalency of the Belief Propagation (BP) decoder of LT codes and the back-substitution decoder of TC. Despite the equivalency of the decoders, compared to optimized LT codes, depending on the packet length, TC reduces redundancy by 68%–99% for $k \leq 1024$.

Three variants of TC have been shown in this paper (LT-TC, IV-TC and R-TC). IV-TC has decoding complexity of $\mathcal{O}(k^2L)$, which is optimal in the sense that decoding can be successfully completed with k packets. However, it has the bit redundancy of $\mathcal{O}(k^3)$. IV-TC is suitable for applications using small values of k, such as multimedia streaming.

Systematic R-TC has encoding-decoding complexities of $\mathcal{O}(kL\log k)$, with a small redundancy of around 0.2–2%. TC jointly addresses the high decoding cost of existing erasure coding schemes such as RS, RL, R10 and RQ codes, and delivers near-zero redundancy for small packet lengths of practical importance. This makes TC suitable over patented Raptor codes for wide adoption in battery-constrained and low-CPU devices in the 5G network.

As part of future work, various variants of TC can also be designed by proposing efficient encoding machinery for different applications. This can be done by adopting techniques such as optimization methods to

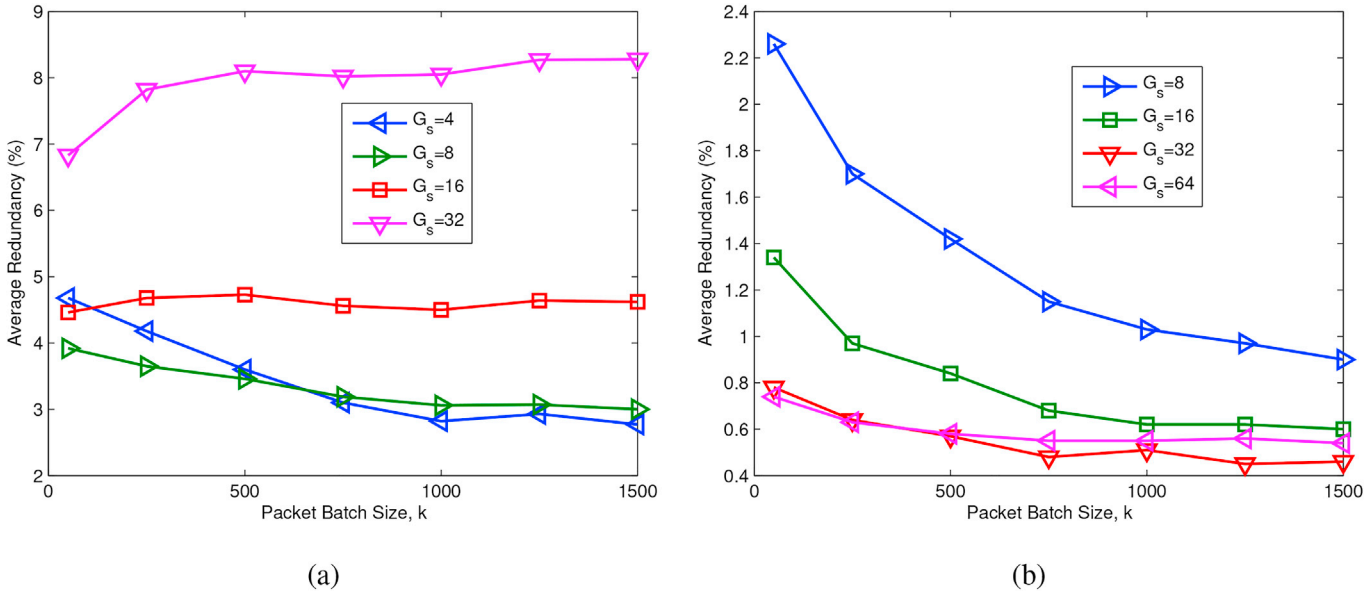

Fig. 4. Redundancy of non-systematic R-TC for different values of G_s, (a) $L = 40$ bytes, and (b) $L = 1500$ bytes.

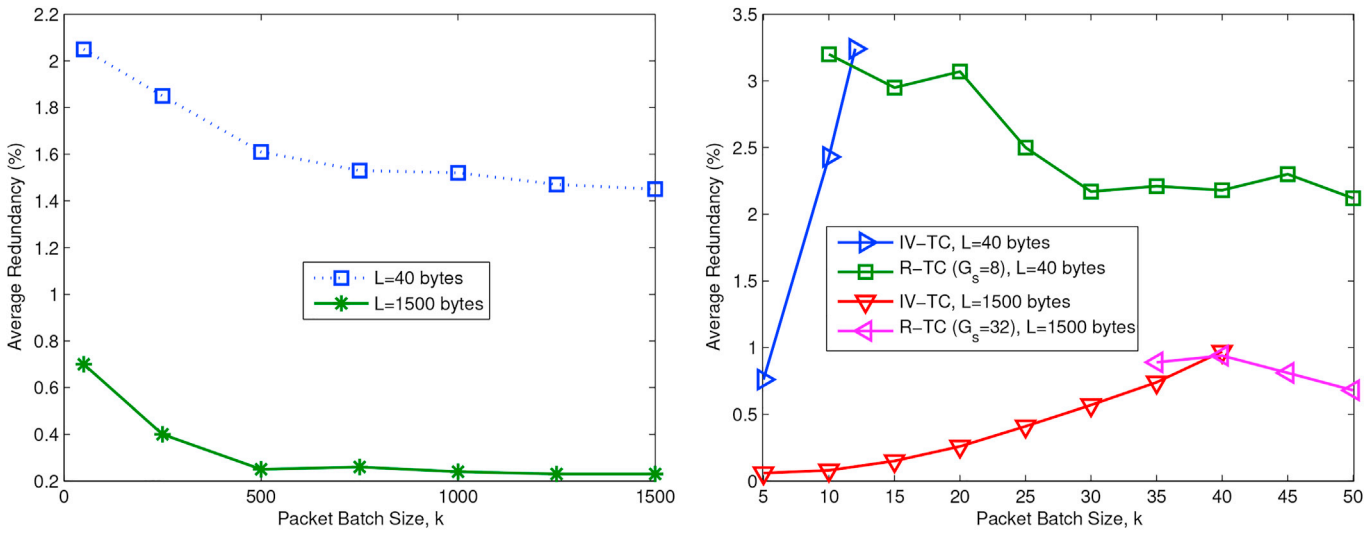

Fig. 5. (a) Performance of systematic R-TC for a relatively high wireless channel erasure p_e of 40%, for $50 \leq k \leq 1500$. (b) Performance of systematic IV-TC and R-TC for small packet batch size of $5 \leq k \leq 50$ for p_e of 40%.

Table 4
Comparison of optimized LT code with R-TC. 'Reduction' values are expressed in percentage.

k	LT code	TC, L = 40 bytes		TC, L = 1500 bytes	
	ε	ε	Reduction	ε	Reduction
8	45 [13]	1.6	96.4	0.07	99.8
20	42 [13]	3.6	91.4	0.25	99.4
64	29.2 [6]	4.5	84.8	0.94	96.8
128	24 [6]	3.7	84.6	0.70	97.1
512	12 [14]	3.4	71.7	0.60	95.0
1024	9 [14]	2.9	67.8	0.48	94.7

improve the encoding process. Due to the randomness of encoding in R-TC and iterative decoding process, a computationally feasible analysis model can also be developed to determine the average redundancy of TC for different values of shift operation G_s, sparsity, and the number of source packets k.

Declaration of competing interest

None.

References

[1] D.J. MacKay, Fountain codes, IEE Proc. Commun. 152 (6) (2005) 1062–1068.
[2] L. Al-Awami, H.S. Hassanein, Robust decentralized data storage and retrieval for wireless networks, Comput. Network. 128 (2017) 41–50.
[3] T. Stockhammer, A. Shokrollahi, M. Watson, M. Luby, T. Gasiba, Application layer forward error correction for mobile multimedia broadcasting, in: Handbook of Mobile Broadcasting DVB-H, DMB, ISDB-T, and MEDIAFLO, Auerbach Publications, 2008.
[4] J.W. Byers, M. Luby, M. Mitzenmacher, A digital fountain retrospective, SIGCOMM Comput. Commun. Rev. 49 (5) (2019) 82–85.
[5] H. Tian, D.-F. Zhao, Y.-F. Yang, R. Xue, Research of LT code based on key information feedback in deep space communication, IEEE Access 8 (2020) 103956–103972.
[6] M. Rossi, N. Bui, G. Zanca, L. Stabellini, R. Crepaldi, M. Zorzi, SYNAPSE++: code dissemination in wireless sensor networks using fountain codes, IEEE Trans. Mobile Comput. 9 (12) (2010) 1749–1765.
[7] H. Sehat, P. Pahlevani, An analytical model for rank distribution in sparse network coding, IEEE Commun. Lett. 23 (4) (2019) 556–559.
[8] R. Rafie Borujeny, M. Ardakani, A new class of rateless codes based on reed–solomon codes, IEEE Trans. Commun. 64 (1) (2016) 49–58.
[9] M.R. Zakerinasab, M. Wang, A cloud-assisted energy-efficient video streaming system for smartphones, in: 2013 IEEE/ACM 21st International Symposium on Quality of Service (IWQoS), 2013, pp. 1–10.
[10] M. Agiwal, A. Roy, N. Saxena, Next generation 5G wireless networks: a comprehensive survey, IEEE Commun. Surveys Tut. 18 (Third Quarter) (2016) 1617–1655.
[11] Cisco, 2020 Global Networking Trends Report, 2019.
[12] A. Cattaneo, A. Sentinelli, A. Vitali, L. Celetto, G. Marfia, M. Roccetti, M. Gerla, Using digital fountains in future IPTV streaming platforms: a future perspective, IEEE Commun. Mag. 50 (5) (2012) 202–207.
[13] E. Hyytia, T. Tirronen, J. Virtamo, Optimal degree distribution for LT codes with small message length, in: IEEE INFOCOM 2007 - 26th IEEE Int. Conf. On Comp. Commun, 2007, pp. 2576–2580.
[14] K.-K. Yen, Y.-C. Liao, H.-C. Chang, Design of LT code degree distribution with profiled output ripple size, in: 2015 IEEE Workshop on Signal Processing Systems (SiPS), IEEE, 2015, pp. 1–6.
[15] QUALCOMM, RaptorQ™Technical Overview, URL, https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/RaptorQ_Technical_Overview.pdf, 2010.
[16] E. Tsimbalo, A. Tassi, R.J. Piechocki, Reliability of multicast under random linear network coding, IEEE Trans. Commun. 66 (6) (2018) 2547–2559.
[17] J.B. Fraleigh, R.A. Beauregard, Linear Algebra, Addison-Wesley Publishing Company, 1995 (Chapter 10).
[18] J. Qureshi, C.H. Foh, J. Cai, Optimal solution for the index coding problem using network coding over GF(2), 9th Annu, in: IEEE Commun. Soc. Conf. On Sensor, Mesh and Ad Hoc Commun. and Networks (SECON), 2012, pp. 209–217.
[19] J. Qureshi, Achieving Reliability for Wireless Multicast Transmission Using Network Coding, Doctoral thesis, School of Computer Engineering, Nanyang Technological University, Singapore, 2014. http://dr.ntu.edu.sg/handle/10356/60866.
[20] P. Torres Compta, F.H.P. Fitzek, D.E. Lucani, Network coding is the 5G key enabling technology: effects and strategies to manage heterogeneous packet lengths, Trans. Emerg. Telecommun. Technol. 26 (1) (2015) 46–55.
[21] J.S. Plank, Y. Ding, Note: correction to the 1997 tutorial on reed solomon coding, Software Pract. Ex. 35 (2) (2005) 189–194.
[22] E. Maneva, A. Shokrollahi, New model for rigorous analysis of LT-codes, in: 2006 IEEE International Symposium on Information Theory, 2006, pp. 2677–2679.