entropy

MDPI

Article

Design and Analysis of Systematic Batched Network Codes

Licheng Mao [1], Shenghao Yang [1,*], Xuan Huang [2] and Yanyan Dong [3]

[1] School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, Shenzhen 518172, China; lichengmao@link.cuhk.edu.cn
[2] Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, China; 1155136647@link.cuhk.edu.hk
[3] Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117597, Singapore
* Correspondence: shyang@cuhk.edu.cn

Abstract: Systematic codes are of important practical interest for communications. Network coding, however, seems to conflict with systematic codes: although the source node can transmit message packets, network coding at the intermediate network nodes may significantly reduce the number of message packets received by the destination node. Is it possible to obtain the benefit of network coding while preserving some properties of the systematic codes? In this paper, we study the systematic design of batched network coding, which is a general network coding framework that includes random linear network coding as a special case. A batched network code has an outer code and an inner code, where the latter is formed by linear network coding. A systematic batched network code must take both the outer code and the inner code into consideration. Based on the outer code of a BATS code, which is a matrix-generalized fountain code, we propose a general systematic outer code construction that achieves a low encoding/decoding computation cost. To further reduce the number of random trials required to search a code with a close-to-optimal coding overhead, a triangular embedding approach is proposed for the construction of the systematic batches. We introduce new inner codes that provide protection for the systematic batches during transmission and show that it is possible to significantly increase the expected number of message packets in a received batch at the destination node, without harm to the expected rank of the batch transfer matrix generated by network coding.

Keywords: network coding; systematic code; random linear network coding; batched network coding; BATS code

Citation: Mao, L.; Yang, S.; Huang, X.; Dong, Y. Design and Analysis of Systematic Batched Network Code. *Entropy* **2023**, *25*, 1055. https://doi.org/10.3390/e25071055

Academic Editor: Boris Ryabko

Received: 31 May 2023
Revised: 25 June 2023
Accepted: 28 June 2023
Published: 13 July 2023

1. Introduction

Network coding has great advantages compared with the traditional store-and-forward in network communications [1–3]. Random linear network coding (RLNC) provides a decentralized approach to network coding and achieves the multicast capacity of networks with packet loss in a broad setting [4–10]. In the past twenty years, extensive studies have been performed towards resolving the implementation issues of RLNC, such as the computational complexity and the coefficient overhead [11–14]. Batched network coding extends RLNC by introducing an inner code–outer code structure [15–21]. In particular, the outer code of a batched network code encodes the message packets into a sequence of batches, each of which is a number of coded packets, and the inner code is formed by linear network coding applied on the coded packets belonging to the same batch. The design of the outer code and the inner code can be separated, where the outer code achieves end-to-end reliability and the inner code maximizes the network efficiency [22]. The number of packets in a batch (called the *batch size*) affects the coefficient overhead and the computational complexity. To achieve the benefits of network coding and constrain the overhead/complexity, the batch size is usually a small integer larger than 1, e.g., 8 or 16 [23].

Batched network coding allows joint batch encoding/decoding, while the original RLNC schemes can be regarded as special batched network codes where the outer code has a single batch or multiple batches encoded/decoded separately.

In coding theory, a code is said to be systematic if all message symbols form a subset of the coded symbols [24]. Many practical codes can be designed to be systematic—for example, Reed–Solomon codes [25], fountain codes [26] and polar codes [27]. Standardized LDPC codes in both 802.11 and 5 G NR are systematic. For network communications, the retransmission-based end-to-end reliability scheme can be regarded as a systematic code. The benefits of the systematic codes are also attractive for practical applications of batched network codes, especially for latency-sensitive applications [28–30].

Different from systematic channel coding, systematic batched network coding needs to take both the outer code and the inner code into consideration. Though not optimal in general, an overlapping outer code with batches formed by subsets of the message packets proposed in [15–17,20] is already systematic in the sense that the union of some batches includes all the message packets. However, even with a systematic outer code, the benefits of systematic codes cannot be obtained due to network coding: using random linear coding at the intermediate nodes prevents the reception of the message packets at the destination nodes. The problem cannot be solved by simply excluding the message packets from network coding, which reduces the benefits of network coding. Most existing works on systematic RLNC focus on encoding and decoding at the source node and destination nodes, respectively, without considering network coding at the intermediate nodes [31–36].

In this paper, we study systematic batched network codes that have a systematic outer code and an inner code that can preserve the benefits of the systematic outer code. Our contributions are summarized as follows.

1.1. Contributions Regarding Systematic Outer Codes

The outer code of a batched network code can be designed by extending fountain codes or LDPC codes [19,21], which achieve higher rates than the overlapping outer codes for the same inner code. The existing outer codes obtained by coding are not designed to be systematic. In principle, any linear code can be systematic by transforming the generator matrix to the reduced echelon form. As the existing outer codes obtained by coding are linear, they can also be systematic. The main issue, however, is how to preserve the low encoding/decoding computation cost: a general transformation of the generator matrix by Gaussian elimination affects the structure of the codes and hence may increase the computation cost.

In this paper, we design a systematic outer code based on the BATched Sparse (BATS) outer code, which is a matrix-generalized fountain code [19]. When the batch size is 1, the BATS outer code becomes a fountain code. The BATS outer code preserves the rateless feature of fountain codes, i.e., the number of batches that can be generated is unlimited (i.e., the rateless property) and can achieve a nearly optimal outer code rate with low encoding/decoding complexity. To preserve the salient features of the BATS outer code, the systematic outer code is also expected to be rateless, where the first n_s batches (called *systematic batches*) consist of a partition of the message packets. In addition to the systematic batches, the outer code can further generate more batches, called *non-systematic batches*. The fountain code has a low-complexity systematic design that benefits from the universal degree distribution [37]. When the batch size is larger than 1, the degree distribution of the BATS outer code depends on the rank distribution of the batch transfer matrices and hence is not universal. For this reason, the systematic design of BATS codes has to consider some new issues that do not appear in the systematic fountain code design.

In this paper, we generalize the fountain code approach to design a systematic outer code, which uses a (non-systematic) BATS outer code that satisfies the *consistency* requirement. In particular, a consistent outer code generates the first n_s batches deterministically, which can recover all the message packets. To ensure a small coding overhead, n_s is expected to be as small as possible. For a fountain code, the minimum value of n_s is the same

as the number of message packets, and a consistent code with the minimum value of n_s can be found using a number of trials of the random encoding procedure of the fountain code. As fountain codes are universal, for each number of message packets, a consistent code can be designed once and used forever. However, BATS outer codes are not universal, and, even for the same number of message packets, the consistent code is different for different rank distributions. Our experiments show that when the number of message packets is larger, many more random trials are required to find a consistent outer code with a small coding overhead.

To design a systematic outer code with a small value of n_s more efficiently, we propose a structured encoding approach for the first n_s batches, called *triangular embedding*. Using triangular embedding, zero-coding-overhead outer codes can be designed with one or two random trials for a large range of the number of message packets. Triangular embedding does not increase the computation costs of both encoding and decoding. Moreover, we also verify in experiments that the batches generated by triangular embedding can be used with the batches generated by the BATS outer code and demonstrate superior decoding performance compared to the BATS outer code.

We also analyze the encoding and decoding computation costs of the proposed systematic outer code. For encoding, the systematic outer code has a lower computation cost than the corresponding BATS outer code. The decoding computation cost of the systematic outer code depends on the number of message packets received at the destination node. When all the message packets are received, no computation is required for decoding. When some of the message packets are not received, the decoding computation cost of the systematic outer code increases with the number of message packets that are not received and is at most 2 times the computation cost of the BATS outer code decoding.

1.2. Contributions Regarding Inner Codes

We further study the inner code that can protect the message packets in the systematic batches. For line networks, *systematic inner coding* has been discussed for batched network coding [23], where an intermediate node transmits both the received packets and the recoded packets generated by linear combinations of the received packets. For a line network without packet loss, the destination node can receive all the message packets generated by the systematic outer code when using systematic recoding. However, if the packet loss rate for each communication link is bounded below by a positive number, the number of message packets that can be received by the destination node decreases exponentially rapidly as the network length increases. For systematic RLNC, a decode–recode network coding approach has been proposed to protect the message packets [38], where an intermediate node first tries to decode the message packets and then transmits the decoded message packets together with some recoded packets. Systematic RLNC is a special systematic batched network code with only the systematic batches, and the decode–recode approach is mainly discussed for extended window recoding.

In this paper, we extend and refine the decode–recode approach for the inner code of batched network coding. For a general batched network code, it is not necessary that the received packets of a batch at an intermediate node can decode all the original packets. In other words, the batch transfer matrix formed by the coefficient vectors of all the received packets of a batch at an intermediate node may have a rank lower than the batch size. We instead study how to decode some of the message packets uniquely at an intermediate node. We say that a message packet in a systematic batch is *recoverable* at an intermediate node if it can be uniquely solved by the received packets of the batch at the intermediate node. We give a necessary and sufficient condition such that a message packet in a batch is recoverable, and we show that using Gauss–Jordan elimination, we can find all the recoverable message packets in a batch. We also analyze the recovery of the message packets at the next hop subject to packet loss and side information. Our analysis shows that generating all recoded packets using random linear coding is not preferable, and knowing

more information about recoding than the coefficient vectors does not aid in the recovery of message packets.

Based on our analysis, we improve systematic inner coding to protect the message packets in a batch, where the level of protection can be tuned by a parameter. Our inner codes can achieve the same network coding gain as the existing inner codes, while significantly improving the number of received message packets. By tuning the parameter, the number of received message packets can be further increased with the cost of lower coding rates. Both the recovery of the message packets and the message protection recoding are linear operations on a batch, and hence our inner code does not increase the coefficient overhead for decoding at the destination node.

1.3. Paper Organization

The remainder of this paper is organized as follows. Section 2 is a self-contained introduction of batched network coding with the BATS outer code. In Section 3, we propose a general approach to systematic outer codes based on the BATS outer code. In Section 4, we introduce the triangular embedding approach to improve the design efficiency of the systematic outer code. In Section 5, we discuss the inner coding schemes that can protect the message packets in systematic batches. Section 6 presents the concluding remarks.

2. Ordinary Batched Network Coding

We briefly introduce ordinary (non-systematic) batched coding to assist the further discussion of the systematic design. A batched network code is formed by an outer code and an inner code. Here, we focus on a specific outer code called the BATS outer code, which was originally introduced by the BATS code. Readers are referred to [23] for more information about the BATS code.

2.1. BATS Outer Code

The outer code introduced here is also called the *ordinary* outer code, in contrast to the systematic outer code, to be discussed in the next section.

A finite field of size q, denoted as $\mathbb{F}_q$, is called the base field. A packet of length T is a column vector in $\mathbb{F}_q^T$, and a set of packets of the same length is equated to the matrix formed by juxtaposing the packets in the set. We consider the transmission of K message packets, which form the $T \times K$ matrix $\mathbf{B}$ from the source node to the destination node in a network.

The (ordinary) outer code encodes the K message packets in two steps. The first step uses a systematic precode to generate a number of redundant packets, which are also called *parity check packets*. Let $K' \geq K$ be the total number of packets containing the message packets and the parity check packets. Denote by $\mathbf{B}_p$ the $K' - K$ parity check packets. Let $\mathbf{P}$ the $K' \times (K' - K)$ parity check matrix of the precode, i.e.,

$$[\mathbf{B} \ \ \mathbf{B}_p]\mathbf{P} = \mathbf{0}. \tag{1}$$

The parity check packets can include both low-density parity check (LDPC) and high-density parity check (HDPC) packets to balance the computation cost and the decoding performance. Refer to [37] for such a design of $\mathbf{P}$.

Let $\mathbf{B}' = [\mathbf{B} \ \ \mathbf{B}_p]$, which are called the precoded packets. The second encoding step of the outer code generates batches of coded packets. Let M be a positive integer called the *batch size*, which is usually less than a hundred. For $i = 1, 2, \ldots$, the ith batch $\mathbf{X}_i$ includes M packets generated from a subset $\mathbf{B}_i \subset \mathbf{B}'$ as follows:

$$\mathbf{X}_i = \mathbf{B}_i \mathbf{G}_i,$$

where $\mathbf{G}_i$ is a matrix of M columns called the *batch generator matrix*. The number of packets in $\mathbf{B}_i$, which is also the number of rows of $\mathbf{G}_i$, will be specified later. When $M = 1$, the outer code becomes a fountain code. The design of $\mathbf{B}_i$ is discussed as follows.

Here, we discuss general batch encoding that can be used for various decoding approaches, including inactivation decoding. The precoded packets are further separated into two parts:

- *active packets* that include a subset of the message packets and all the LDPC packets, and
- *inactive packets* that include all the other message packets and all the HDPC packets.

Denote by A the number of active packets. Then, the number of inactive packets is $K' - A$. We require $A \geq K$. As a special case, when there are no HDPC packets or inactive packets during encoding, we have $A = K'$. The encoding of a batch uses both active and inactive packets.

The number of active packets used in a batch is determined using a degree distribution $\mathbf{\Psi} = (\Psi_1, \ldots, \Psi_{D_{\max}})$, and it affects the decoding performance of both belief propagation decoding and inactivation decoding. The degree distribution $\mathbf{\Psi}$ is designed based on the *batched transfer matrix rank distribution* induced by the inner code. The maximum number $D_{\max}$ for the active packets is sufficient to be a couple of multiples of M, as proven in [19]. For the encoding of each batch $\mathbf{X}_i$,

1. Independently sample $\mathbf{\Psi}$ and obtain an integer d_i^A, which is called the active degree of the batch;
2. Uniformly, at random, choose d_i^A active packets to be included in $\mathbf{B}_i$.

The inactive packets can help to further improve the inactivation decoding performance. When $M = 1$, on average, each batch may involve 2 or 3 inactive packets [26]. When $M > 1$, the number of inactive packets in a batch can be $3(K' - A)/n$, where n is the number of batches expected to be used for decoding. Denote by d_i^B the number of inactive packets used in the ith batch.

Considering both active and inactive packets, $\mathbf{B}_i$ has $d_i = d_i^A + d_i^B$ packets, where d_i is called the total degree of the batch. $\mathbf{G}_i$ is a $d_i \times M$ uniformly random matrix with entries from the base field. In practice, random encoding can be implemented by a pseudorandom number generator. The random values in the encoding process can be used for decoding if they share the same pseudorandom number generator at the source node and the destination node.

Denote by ENC the encoder that implements the above encoding process of the BATS outer code. The pseudocodes of ENC are given in Appendix C for reference.

2.2. General Inner Code Formulation

We use a line network as an example to introduce the inner code, and the inner code can be extended to other network typologies as discussed in [23]. A line network of length L is formed by a sequence of network nodes labeled by $0, 1, \ldots, L$, where the first node 0 is the source node and the last node L is the destination node. All the other nodes are called intermediate nodes. Network links exist only between two consecutive network nodes, modeled by packet erasure channels, i.e., a packet transmitted on a network link is either correctly received or erased. Figure 1 illustrates the line network.

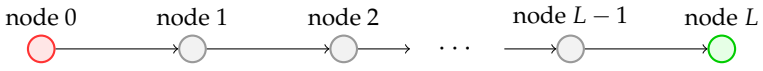


Figure 1. A line network of length L. Node 0 is the source node, and node L is the destination node. The direct edge from node i to node $i + 1$ ($i = 0, 1, \ldots, L - 1$) illustrates the network link.

The inner code is the composition of the *recoding* operations performed on each batch separately. The recoding at the source node takes the batches generated by the outer code as the input, and the recoding at an intermediate node takes the received packets of a batch as the input. For each batch, recoding generates a number of linear combinations of the packets belonging to the batch, and the packets generated by recoding are supposed to belong to the same batch. There are various approaches to the recoding operation, which

is determined by the linear combination coefficients. The original RLNC schemes use coefficients chosen uniformly at random from the base field [4,6,7], and extensive research has been carried out towards recoding with lower complexity and latency [39–44]. In this paper, we study the recoding schemes that can fulfil the systematic coding requirement.

Without specifying a recoding scheme, we give a general formulation of recoding. Fix a certain network node u. Let $\mathbf{Y}_i^{(u)}$ be the received packets of the ith batch at the node u. At the source node, $\mathbf{Y}_i^{(0)} = \mathbf{X}_i$. As recoding is linear, for $v = 1, \ldots, L$,

$$\mathbf{Y}_i^{(u)} = \mathbf{X}_i \mathbf{H}_i^{(u)} = \mathbf{B}_i \mathbf{G}_i \mathbf{H}_i^{(u)}, \tag{2}$$

where $\mathbf{H}_i^{(u)}$ is called the *(batch) transfer matrix* of the ith batch at the node u. The number of rows of $\mathbf{H}_i^{(u)}$ is M. The number of columns of $\mathbf{H}_i^{(u)}$ corresponds to the number of packets received for the ith batch at the node u, which may vary for different batches and is finite. If no packets are received for a batch, $\mathbf{Y}_i^{(u)}$ ($\mathbf{H}_i^{(u)}$) is the empty matrix of 0 columns.

Note that the transfer matrices are determined not only by the recoding scheme, but also by the network packet loss pattern. Due to the randomness in both recoding and packet loss, the transfer matrices cannot be derived from the recoding design. To obtain the transfer matrices, RLNC introduces *coefficient vectors* embedded in the packet header immediately after $\mathbf{X}_i$ is generated. The matrix formed by the coefficient vectors is the identity matrix. The same linear operations performed on a batch are performed on the coefficient vectors as well, so that $\mathbf{H}_i^{(u)}$ can be known at each node u that receives batch i from the header of the batch.

We say that a set of packets of a batch are linearly independent/dependent if their corresponding coefficient vectors in the packet header are linearly independent/dependent. We call $\text{rank}(\mathbf{H}_i^{(u)})$ the *rank of the ith batch* at node u.

2.3. Decoding Algorithms

Suppose that n batches $\mathbf{Y}_i^{(L)}$, $i = 1, \ldots, n$ are received at the destination node L. A decoder is expected to recover $\mathbf{B}$ using $\mathbf{Y}_i^{(L)}$, $i = 1, \ldots, n$, which are related by a linear system. From this perspective, we obtain an upper bound on the decoding performance [23]:

$$K \leq \sum_{i=1}^{n} \text{rank}(\mathbf{H}_i^{(L)}).$$

When used as a block code with a fixed number n of batches, the *(outer) coding rate* defined as K/n, together with the decoding success probability, is used to measure the outer code performance. When used as a rateless code, decoding allows more batches to be used until all the message packets are decoded, and the *(outer) coding overhead* defined as $\sum_{i=1}^{n} \text{rank}(\mathbf{H}_i^{(L)}) - K$ is used to measure the decoding performance.

As $\mathbf{B}$ and $\mathbf{Y}_i^{(L)}$, $i = 1, \ldots, n$ are related by a linear system, Gaussian elimination is the optimal algorithm to solve $\mathbf{B}$. However, Gaussian elimination incurs a computational complexity linear in K when decoding one message packet on average, which is not tolerable when K is slightly large. In the remainder of this section, we introduce several approaches that can achieve $O(1)$ complexity in decoding one message packet. In the following, we first discuss two decoding algorithms without inactive packets and then discuss inactivation decoding.

2.3.1. Two-Step Decoding

Suppose that the number of inactive packets during encoding is 0, so that $d_i^B = 0$ for all batches. We first discuss the two-step decoding approach. The first step recovers a fraction $\eta \geq K/K'$ of precoded packets using a belief propagation (BP) algorithm, which repeats the following operations:

1. A batch i is said to be decodable if $d_i^A = \text{rank}(\mathbf{G}_i\mathbf{H}_i^{(L)})$; solve a decodable batch by Gaussian elimination;
2. Substitute the decoded (precoded) packets into other undecoded batches and update the corresponding batch degree and generator matrix.

The BP decoding algorithm has a low computation cost that does not depend on the total number of message packets K. The second step decodes the precoded packets to recover the message packets, which is expected to be successful if the first step recovers at least η fractions of all the precoded packets.

Assume that the ranks of batch transfer matrices at the destination node $\text{rank}(\mathbf{H}_i^{(L)})$ are i.i.d and follow the distribution $\mathbf{h} = (h_0, h_1, \ldots, h_M)$. We call $\mathbf{E}[\mathbf{h}] = \sum_{i=1}^{M} ih_i$ the expected rank. According to the theory of BATS codes [23], it is possible to design a degree distribution $\mathbf{\Psi}$ for a given rank distribution $\mathbf{h}$ such that when K is large, the BP decoding can recover a given η fraction of the precoded packets with a high probability when the coding rate K/n is larger, but very close to $\mathbf{E}[\mathbf{h}]$. In other words, we only need slightly more than $K/\mathbf{E}[\mathbf{h}]$ batches to recover the K message packets.

2.3.2. Joint Decoding

The above two-step decoding algorithm can be improved by combining the two steps when the precoding includes LDPC. For LDPC precoding, each parity check constraint can be regarded as a batch with batch size 1 and only one all-zero received packet. Then, the BP decoding of the batches in the first step of the two-step approach can also include the parity checks.

In practice, the decoding of the LDPC precode and the decoding of the batches in the two-step decoding algorithm can be combined together to improve the performance. The joint decoding algorithm can improve the decoding success rate and reduce the coding overhead of the two-step decoding algorithm, but does not increase the computation cost of the two-step decoding.

2.3.3. Inactivation Decoding

When K is relatively small or the coding overhead is small, BP decoding tends to stop before decoding all the message packets. Although we can continue decoding by Gaussian elimination, the computational complexity is high.

A better approach is to use *inactivation decoding*: when BP decoding stops, an undecoded message packet is marked as inactive and substituted into the batches as a decoded packet to resume the BP decoding procedure. The decoding of batches with inactive packets also induces linear constraints on the inactive packets. Eventually, all the message packets are either decoded or inactive. The inactive packets are then solved by the linear constraints induced by decoding batches and the precodes. Inactivation decoding has the same decoding performance as Gaussian elimination, but can have a much lower computation cost if the number of inactive packets is small.

Moreover, when using inactivation decoding, we can use the inactive packets during encoding. Inactive packets during encoding are treated as inactive from the beginning of inactivation decoding and hence are also called *pre-inactive packets*. The extra inactive packets added during decoding are called the *dynamic inactive packets*. See [23] for a detailed discussion of inactivation decoding for BATS codes.

Denote by DEC the decoder that implements one of the above decoding processes of the BATS outer code. The pseudocodes of DEC for two-step decoding are given in Appendix C for reference.

3. Systematic Outer Codes

In this section, we design a systematic outer code that can preserve the silent features of the ordinary BATS outer code. We call those batches that are designed to include all the message packets the *systematic batches*.

3.1. Naive Approaches

Before introducing our approach, we first discuss some naive approaches and their limitations. For a fixed number n of batches, the outer code is a linear block code and hence the encoding process can be described as

$$\begin{bmatrix}\mathbf{X}_1 & \cdots & \mathbf{X}_n\end{bmatrix} = \mathbf{B}\tilde{\mathbf{G}}$$

where $\tilde{\mathbf{G}}$ is the $K \times nM$ generator matrix of the first n batches. Suppose that $nM \geq K$. If $\tilde{\mathbf{G}}$ has K columns forming the identity matrix, the outer code is systematic.

First, we show that the random encoding of the ordinary BATS outer code is not a systematic code with high probability. For a batch of total degree d, the probability that a coded packet is equal to a precoded packet is dq^{-d}. As not all precoded packets are message packets, the probability that a coded packet is equal to a message packet is no greater than dq^{-d}. Typically, $d \geq M \geq 2$ and $q = 256$. Thus, it is unlikely that a message packet appears in a batch using the ordinary outer code.

When n is slightly larger than K/M, the matrix $\tilde{\mathbf{G}}$ obtained from the ordinary BATS outer code has rank K with a high probability. The general procedure to make a linear code systematic is to transform $\tilde{\mathbf{G}}$ by elementary row operations into the reduced row echelon form. Although a systematic code can be obtained, the drawback of this approach is that the low encoding/decoding computation cost of the BATS outer code cannot be preserved.

Now, we discuss another naive approach that seems solve the computation cost issue. To simplify the discussion, suppose that the number of message packets K is a multiple of the batch size M. In this naive approach, the first K/M batches form a partition of all the message packets, and more (non-systematic) batches are generated according to the encoding of batches as an ordinary outer code discussed in Section 2.1. However, to guarantee good decoding performance using the naive approach, a high degree must be applied to all the non-systematic batches.

We show two cases wherein a high degree of the non-systematic batches is necessary. In the first case, one systematic batch is completely erased during the communication and all the other systematic batches are received by the destination nodes, together with a non-systematic batch. Suppose that the erased batch is randomly chosen. For all the received batches, the batch transfer matrix is the $M \times M$ identity matrix so that the decoding problem becomes one of traditional erasure coding. The total number of received packets is K. To guarantee the decoding of all the message packets, it is necessary that the degree of the received non-systematic batch is K.

In the second case, we consider that for M systematic batches, only one packet is erased during communication and all the other packets are received correctly. In other words, the batch transfer matrix of these M systematic batches is the $M \times M$ identity matrix with one column removed, chosen uniformly at random. The destination node also receives all the other systematic batches, together with a non-systematic batch, all with the identity batch transfer matrix. The total number of received packets is K. To guarantee the decoding of all the message packets, it is necessary that the degree of the received non-systematic batch is K.

From these cases, we see that to achieve a high coding rate using the naive approach, the degree of the non-systematic code must be high and hence the encoding/decoding complexity is high. In the remainder of this section, we derive an approach to obtain a systematic outer code that has similar encoding/decoding complexity to the ordinary BATS outer code.

3.2. General Approach to Systematic Outer Codes

We give a general approach tp systematic outer codes, which extends the idea of systematic fountain codes [37]. Suppose that we have K message packets $\mathbf{B}$ for encoding using a systematic outer code with batch size M, where K is not necessarily a multiple of M. Let n_s be an integer larger than or equal to K/M, to be decided later. We wish to

design an outer code such that the first n_s batches are systematic batches that include all the message packets.

Our approach to a systematic outer code uses an ordinary outer code $(\mathrm{ENC}, \mathrm{DEC})$, where ENC is the encoder and DEC is the decoder, as described in Sections 2.1 and 2.3, respectively. The encoder ENC has two parts ENC_{n_s} and $\mathrm{ENC}_{n_s^+}$, where ENC_{n_s} generates the first n_s batches and $\mathrm{ENC}_{n_s^+}$ generates all the further batches. The decoder DEC in general applies to all the batches subject to any batch transfer matrices. We denote by DEC_{n_s} the case of DEC when applying to the first n_s batches with the rank-M batch transfer matrices.

To construct the systematic outer code, we require $(\mathrm{ENC}, \mathrm{DEC})$ satisfying some additional requirements. The pair $(\mathrm{ENC}, \mathrm{DEC})$ is said to be *consistent* if the following conditions are satisfied:

1. ENC_{n_s} and DEC_{n_s} are deterministic; and
2. for any K packets $\mathbf{B}$,
$$\mathbf{B} = \mathrm{DEC}_{n_s}\big(\mathrm{ENC}_{n_s}(\mathbf{B})\big). \tag{3}$$

For the consistency requirement 2, it is possible to verify (3) without any specific value $\mathbf{B}$ of K packets, i.e., it is not necessary to check all choices of K packets. The reason is that both ENC_{n_s} and DEC_{n_s} are linear operations and, if the decoding is successful, their joint effect is to multiply the $K \times K$ identity matrix. We discuss how to design a consistent outer code later. Here, we focus on how to use it to construct a systematic outer code.

For a consistent $(\mathrm{ENC}, \mathrm{DEC})$, the decoder DEC_{n_s} solves K message packets from the $n_s M$ coded packets generated by ENC_{n_s}. Among the $n_s M$ coded packets, $n_s M - K$ coded packets are redundant and can be removed without affecting the decoding performance (The decoding of a BATS code requires us to solve a system of linear equations by elementary equation operations. Each coded packet corresponds to an equation of the system. Each equation can solve at most one message packet. Therefore, exactly K equations are eventually transformed into the solutions of the message packets. The other equations are redundant). All the redundant packets can be identified by a trial of DEC_{n_s}. For $i = 1, \ldots, n_s$, let M_i be the number of non-redundant coded packets in the ith batch. We know that $\sum_{i=1}^{n_s} M_i = K$. Denote by $\mathrm{DEC}^*_{n_s}$ the same decoder as DEC_{n_s} except that the redundant coded packets are removed from the decoder input.

Now, we can construct the systematic outer code. For the systematic outer code, the encoding at the source node works as follows:

1. Partition the message packets $\mathbf{B}$ into n_s subsets $\tilde{\mathbf{X}}_i$, $i = 1, \ldots, n_s$, where the number of packets in the ith subset $\tilde{\mathbf{X}}_i$ is M_i;
2. Calculate $\tilde{\mathbf{B}} = \mathrm{DEC}^*_{n_s}(\tilde{\mathbf{X}}_1, \ldots, \tilde{\mathbf{X}}_{n_s}) = \mathrm{DEC}^*_{n_s}(\mathbf{B})$;
3. Generate the first n_s batches $\mathrm{ENC}_{n_s}(\tilde{\mathbf{B}})$;
4. Generate more batches by performing $\mathrm{ENC}_{n_s^+}$ on $\tilde{\mathbf{B}}$.

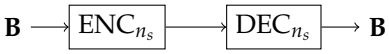
See Figure 2b for an illustration of the above encoding process.

We justify that the above encoding process is systematic by showing that the first n_s batches include all the message packets. Denote by $\mathrm{ENC}^*_{n_s}$ the encoder that generates only the M_i non-redundant coded packets in the ith batch, where $i = 1, \ldots, n_s$. For any K packets $\mathbf{B}$, $\mathrm{DEC}^*_{n_s}\big(\mathrm{ENC}^*_{n_s}(\mathbf{B})\big) = \mathrm{DEC}_{n_s}\big(\mathrm{ENC}_{n_s}(\mathbf{B})\big) = \mathbf{B}$. Note that ENC_K and DEC_K can be expressed as square matrices that are inverse to each other, and hence their order can be interchanged without changing the output, i.e., $\mathrm{ENC}^*_{n_s}\big(\mathrm{DEC}^*_{n_s}(\mathbf{B})\big) = \mathrm{ENC}^*_{n_s}\big(\tilde{\mathbf{B}}\big) = \mathbf{B}$.

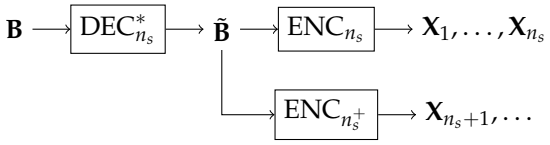
The computation cost of the third step of encoding can be simplified as not all the packets in the systematic batches need to be regenerated. Let $(\mathbf{X}_1, \ldots, \mathbf{X}_{n_s}) = \mathrm{ENC}_{n_s}(\tilde{\mathbf{B}})$. We have $\tilde{\mathbf{X}}_i \subset \mathbf{X}_i$ and $\mathbf{X}_i \setminus \tilde{\mathbf{X}}_i$ includes only the redundant packets for DEC_{n_s} in the ith batch. As $\tilde{\mathbf{X}}_i$ is a subset of the message packets, it is not necessary to generate it again. Denote by ENC^-_n the encoder of $\tilde{\mathbf{B}}$ that generates only $\mathbf{X}_i \setminus \tilde{\mathbf{X}}_i$ for $i = 1, \ldots, n$. Let $(\bar{\mathbf{X}}_1, \ldots, \bar{\mathbf{X}}_n) = \mathrm{ENC}^-_n(\tilde{\mathbf{B}})$. Then, the n systematic batches are $\mathbf{X}_i = \tilde{\mathbf{X}}_i \cup \bar{\mathbf{X}}_i$.

The batches generated by the above systematic encoding process will be further transmitted through a network and processed by the inner code. Let $\mathbf{Y}'$ be the coded

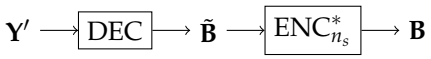
packets received by the destination node. To decode, first, DEC is applied on $\mathbf{Y}'$ to output $\tilde{\mathbf{B}}$. Then, we apply ENC_{n_s} on $\tilde{\mathbf{B}}$ to recover $\mathbf{B}$. See Figure 2c for an illustration of the decoding process.



(**a**) normal encoding and decoding



(**b**) encoding of the systematic code



(**c**) decoding of the systematic code

Figure 2. Illustration of the approach to systematic outer codes. (**a**) shows the normal use of a consistent pair of the outer code encoder ENC_{n_s} and decoder DEC_{n_s}. (**b**) shows the encoding of the systematic code, where $\mathrm{ENC}_{n_s^+}$ is the outer code encoder that generates the coded packets beyond the first n_s batches. (**c**) shows the decoding of the systematic code, where $\mathbf{Y}'$ is the received coded packets generated by inner coding.

3.3. Computation Cost

At first, it seems that the systematic outer code increases the encoding and decoding computation cost because an additional decoding step is employed in the systematic encoding, and an additional ordinary encoding step is employed in the systematic decoding. However, after careful evaluation, we see that the encoding computation cost of the systematic outer code is lower than that of the ordinary outer code. The decoding computation cost of the systematic outer code depends on the number of message packets received at the destination node. In the worst case, where no message packets are received, the decoding computation cost is doubled.

To assist our discussion, we denote by b the average computation cost of encoding a packet using the ordinary outer code, and we denote by c the computation cost of decoding the ordinary outer code using K coded packets. Here, we assume that the decoding is successful with zero coding overhead. Suppose that the packet length T is much larger than M, which means that the coefficient vector length is much less than T. According to the analysis in [23], $b = O(M)$ and $c = O(KM)$ linear combination operations (LCOs). (A linear combination operation (LCO) refers to the computation of a linear combination $\mathbf{x} + \alpha\mathbf{y}$, where $\mathbf{x}$ and $\mathbf{y}$ are two packets of T field elements and α is an element from the base field.) Moreover, for the two-step decoding and the joint decoding, $Kb \approx c$. For the inactivation decoding, if the number of inactive packets is bounded by a constant, $Kb \approx c$.

3.3.1. Encoding Computation Cost

The encoding computation cost depends on the number of coded packets generated. For the ordinary outer encoding, the computation cost of encoding k packets is kb, where $k = 1, 2, \ldots$. For the systematic outer code, we assume $n_s M = K$ (we will discuss how to design such a code). As the first K packets are the message packets, the encoding of the first K packets requires no computation. To encode more packets, the systematic outer code needs to execute $\mathrm{DEC}^*_{n_s}$, which has a computation cost c, and $\mathrm{ENC}_{n_s^+}$, which takes computation cost b on average to generate a packet. Therefore, when $k > K$, the computation cost of generating the first k coded packets using the systematic outer code is $(k - K)b + c \approx kb$. See the illustration in Figure 3a regarding the computation cost of generating the first k packets.

To further understand how the encoding computation cost affects the operation at the source node, we consider two models of message packet arrival at the source node. In the first model, the message packets arrive one-by-one with a unit time interval between two consecutive packets. The ordinary outer code encoding can only start to generate the first coded packets from the time K when a precode with HDPC is employed. Let Δ be the time taken by the ordinary encoder to generate K coded packets, where $\Delta \propto Kb \approx c$. The systematic outer code can generate a coded packet upon the arrival of each message packet. At the time K, the systematic outer code executes $\text{DEC}^*_{n_s}$, which also takes Δ time. In the second model, all the K message packets arrive together at the same time, e.g., time K. For this model, the ordinary outer code behaves in the same way as for the previous model, and the systematic outer code can generate the first K coded packets at time K.

We see that for both message packet arrival models, the systematic outer code generates the first K packets earlier than the ordinary outer code. When $k > K$, both encoders generate the kth packet at the same time. See an illustration of this in Figure 3b.

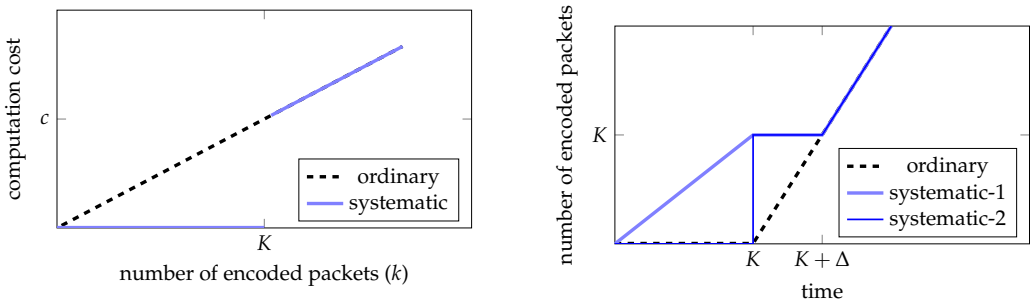


Figure 3. Illustration of the encoding computation cost for the ordinary outer code and the systematic outer code. (**a**) shows the encoding computation cost of generating the first k coded packets. For the ordinary outer code, the computation cost increases linearly with k. For the systematic outer code, the computation cost is 0 when $k \leq K$. The jump in the computation cost after time K is used to execute $\text{DEC}^*_{n_s}$. (**b**) illustrates the number of encoded packets generated over time. The curve "systematic-1" is for the systematic outer code encoder when the message packets arrive one-by-one in each unit time. The curve "systematic-2" is for the systematic outer code encoder when the message packets arrive all at time K. From time K, these two curves overlap. The ordinary outer code behaves in the same way for both message packet arrival models.

3.3.2. Decoding Computation Cost

For the systematic outer code, the decoding computation cost depends on the number K_m of message packets received by the destination node. When $K_m = K$, i.e., all the message packets are received, no computation is required for decoding. When $K_m < K$, the systematic code decoder needs to execute DEC, which has a computation cost c, and $\text{ENC}^*_{n_s}$, which takes computation cost b on average to generate a packet. As K_m message packets have been received, we only need to use $\text{ENC}^*_{n_s}$ to generate the remaining $K - K_m$ message packets. Therefore, the overall decoding computation cost is $(K - K_m)b + c \approx 2c - K_m b$. When K_m is close to K, the systematic outer code decoding computation cost is close to the ordinary outer code decoding. In the worst case, i.e., $K_m = 0$, the systematic outer code decoding computation cost is doubled compared with the ordinary outer code decoding. See an illustration in Figure 4a.

To illustrate how the decoding computation cost affects the operation at the destination node, we consider that coded packets are received one-by-one with a unit time interval between two consecutive packets. We assume that the ordinary outer code decoder starts decoding at time K and takes additional Δ time to decode all the message packets. When $K_m = K$, all the message packets are decoded at time K. When $K_m < K$, the systematic outer code decoder executes DEC at time K and starts to use $\text{ENC}^*_{n_s}$ from time $K + \Delta$ to generate the $K - K_m$ message packets that have not been received. In the worst case, where

$K_m = 0$, the systematic outer code decodes all the message packets at time $K + 2\Delta$. See an illustration in Figure 4b.

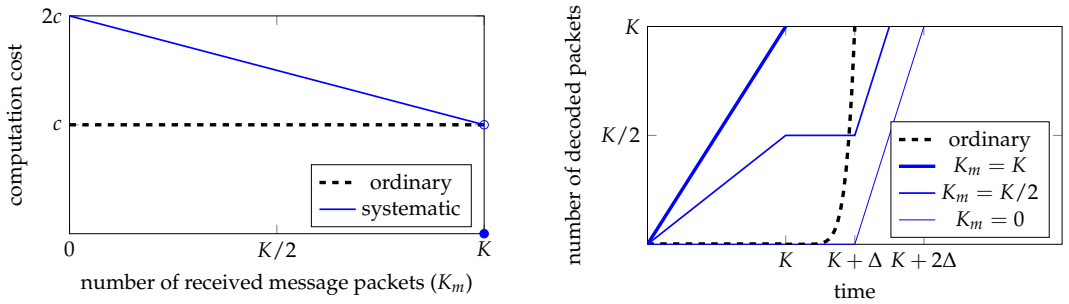


Figure 4. Illustration of the decoding computation costs of the ordinary outer code and the systematic outer code. (**a**) illustrates the decoding computation cost for different numbers K_m of message packets received. For the ordinary outer code, the decoding computation cost is c. For the systematic outer code, the decoding computation cost is approximately $2c - K_m b$ when $K_m < K$ and 0 when $K_m = K$. (**b**) shows the number of decoded packets over time. The three curves labeled $K_m = K, K/2, 0$ are for the systematic outer code decoder with K_m message packets received.

3.4. Random Design

To implement the general approach to the systematic outer code, we only need to design a consistent pair $(\mathrm{ENC}, \mathrm{DEC})$. In the following part of this section, we discuss the traditional random approach to designing a consistent $(\mathrm{ENC}, \mathrm{DEC})$. In the next section, we discuss a new approach that can design a consistent $(\mathrm{ENC}, \mathrm{DEC})$ more effectively.

Denote by $\mathbf{h}$ the rank distribution of the batches and let $\mathbf{\Psi}^A$ be the degree distribution optimized for $\mathbf{h}$ as in ([23], Chapter 6), which achieves the near-to-optimal rate of the ordinary outer code as in Section 2.1 asymptotically. We can use the ordinary encoder and decoder as introduced in Section 2.1 with the degree distribution $\mathbf{\Psi}^A$ to design $\mathrm{ENC}_{n_s^+}$ and DEC for a consistent outer code $(\mathrm{ENC}, \mathrm{DEC})$.

The ordinary outer code is random, but we need a deterministic encoder–decoder pair $(\mathrm{ENC}_{n_s}, \mathrm{DEC}_{n_s})$ to satisfy the consistent properties. For a given $n_s \geq K/\,\mathbb{E}[\mathbf{h}]$, we can perform random trials of the ordinary outer code using the degree distribution $\mathbf{\Psi}^A$ until an instance $(\mathrm{ENC}_{n_s}, \mathrm{DEC}_{n_s})$ is found such that (3) is satisfied. Note that it is sufficient for us to find only one such instance. As both ENC_{n_s} and $\mathrm{ENC}_{n_s^+}$ generate batch instances following the random outer code encoder with the degree distribution $\mathbf{\Psi}^A$, which is optimized for $\mathbf{h}$, DEC can guarantee a high decoding success probability for a sufficiently large number of received batches [23].

If such an instance cannot be found for a certain value n_s, we can increase the value of n_s by 1 and try again. The ordinary outer code is expected to decode correctly with a high probability when the number of batches is sufficiently large, and we expect to design a systematic code with the *expected coding overhead* $n_s\,\mathbb{E}[\mathbf{h}] - K$ as small as possible.

When $M = 1$ and $\mathbb{E}[\mathbf{h}] = 1$, i.e., the case of fountain codes, a consistent outer code exists for a range of the values of K when $n_s = K$ using this approach [26]. For fountain codes, the random design works well as fountain codes have a universal design that can handle all packet loss patterns. The random design is only performed once for each value of the number of message packets K. Therefore, the efficiency of the random design is not an issue. In other words, a large number of random trials can be performed to find a consistent outer code with a small or zero coding overhead.

Although the random design is suitable for fountain codes, it can be less efficient when $M > 1$. BATS codes are not universal in the sense that the optimal degree distribution depends on the rank distribution $\mathbf{h}$. Therefore, even for the same value of K, the random design needs to be repeated for each $\mathbf{h}$, and this may need to be carried out for $\mathbf{h}$ obtained online. Hence, the efficiency of the random design becomes an issue for BATS codes

with batch size $M \geq 2$. For $M = 16$, we perform some experiments using the BATS code implementation in [45] with the parameters in Appendix B. Inactivation decoding is applied to achieve a lower coding overhead. To limit the computation cost of inactivation decoding, the number of inactive packets is limited to 150. In the experiments, we use the rank distribution $\mathbf{h}$ with $\mathbb{E}[\mathbf{h}] = M$, which is also called the rank-M distribution. The experimental results are summarized in Table 1. We observe that when K is up to $400M$, a consistent instance with $n_s = K/M$ can be found. However, the larger the value of K, the lower the probability of a code with zero coding overhead. For example, when $K = 10M$, most instances have zero overhead. Meanwhile, when $K = 400M$, only four instances have zero coding overhead. However, when K is $600M$, no instance is found with zero coding overhead.

Table 1. Experiments of random design. Here, $M = 16$ and $\mathbf{h}$ has rank M. For each value of $K = 10M, 100M, 1000M$, 5000 instances of the ordinary outer code are sampled. As a BATS code is rateless, for each instance, we can try a range of values of n_s. The table gives the number of consistent instances when $n_s = K/M, K/M+1, K/M+2$, and $n_s \geq K/M+3$.

$n_s - K/M$	$K = 10M$	$K = 100M$	$K = 200M$	$K = 400M$	$K = 600M$
0	4784	3552	836	4	0
1	173	175	50	0	0
2	34	113	53	0	0
≥ 3	9	1160	4061	4996	5000

4. Triangular Embedding: A Structured Systematic Outer Code Design

We propose a structured design of consistent outer codes with a general batch size $M \geq 1$. Our approach is based on the following observation. For a consistent instance found by the random design, DEC_{n_s} gives an order of the batches such that the ith batch is solvable if all the previous batches are solved. Our approach, called *triangular embedding*, tries to design ENC_{n_s} so that the order of the batches for solving is predefined. When $M = 1$, our approach also gives a new design of systematic fountain codes.

4.1. Triangular Embedding Design

Consider the encoding of K message packets with respect to a general rank distribution $\mathbf{h}$. We discuss how to generate the first n_s batches, where $n_s \geq K/\mathbb{E}[\mathbf{h}]$. The precode is the same as the ordinary outer code. Let K and K' be the number of message packets and the number of precoded packets, respectively. The precoded packets are also separated into active and inactive packets. Let A be the number of active packets, where $A \geq K$.

For the degree distribution $\mathbf{\Psi}$ optimized for $\mathbf{h}$, we assume that the degree probability is zero for degrees from 1 to $M-1$ (This assumption does not affect the generality of our design as it is asymptotically optimal to use such a degree distribution when the rank M probability of the batch transfer matrix is positive. When the probability of transfer matrix rank M is zero, we should reduce the batch size to improve the network's communication efficiency). Generate the active degree values $d_1^{\mathrm{A}}, \ldots, d_{n_s}^{\mathrm{A}}$ for the first n_s batches by sampling $\mathbf{\Psi}$. To simplify the discussion, we assume that the degree values are ordered so that $M \leq d_1^{\mathrm{A}} \leq d_2^{\mathrm{A}} \leq \cdots \leq d_{n_s}^{\mathrm{A}}$. The inactive degree d_i^B of the ith batch is obtained in the same way as the ordinary outer code.

Fix positive integers $M_1, M_2, \ldots, M_{n_s}$ such that $\sum_{i=1}^{n_s} M_i = K$ and $M_i \leq M \leq d_i^{\mathrm{A}}$. For example, when n_s divides K, we may choose $M_i = K/n_s$. When n_s does not divide K, there exist unique non-negative integers a and $b < n_s$ such that $K = an_s + b$. We may let $M_i = a+1$ for $i = 1, \ldots, b$ and let $M_i = a$ for $i = b+1, \ldots, n_s$.

Let N_{inac} be the maximum number of dynamic inactive packets allowed during inactivation decoding. We should determine the total number of inactive packets $N_{\text{inac}} + K' - A$ according to the decoding computation cost constraint. For example, $N_{\text{inac}} + K' - A = 2\lceil\sqrt{K}\rceil$. We further assume that for $i = 1, \ldots, n_s$,

$$d_i^{\text{A}} \leq \min\{A - K, N_{\text{inac}}\} + \sum_{j=1}^{i} M_j. \tag{4}$$

This assumption is usually satisfied by the degrees sampled as 1) $A - K$ is linear in K and 2) the average degree of a BATS code degree distribution is only around two times the batch size M and even the maximum degree is $O(M)$. If d_i^{A} does not satisfy (4), which should occur rarely, we can modify d_i^{A} to this upper bound or re-sample the active degrees.

Let $m_1 = 0$, and, for $i \geq 2$, let $m_i = m_{i-1} + M_{i-1}$. For $i \geq 1$, the d_i^{A} active packets in $\mathbf{B}_i$ include

- the $(m_i + 1)$th to the $(m_i + M_i)$th active packets, and
- a set of $d_i^{\text{A}} - M_i$ packets chosen from the first m_i active packets and the last $\min\{A - K, N_{\text{inac}}\}$ active packets.

The inactive packets in $\mathbf{B}_i$ are obtained in the same way as the ordinary outer code. The ith batch is generated as $\mathbf{B}_i\mathbf{G}_i$, where $\mathbf{G}_i$ is a $d_i \times M$ matrix different from the ordinary outer code. The rows of $\mathbf{G}_i$ corresponding to the $(m_i + 1)$th to the $(m_i + M_i)$th active packets have the form $\begin{bmatrix}\mathbf{I}_{M_i} & \mathbf{U}\end{bmatrix}$, where $\mathbf{I}_{M_i}$ is the $M_i \times M_i$ identity matrix and $\mathbf{U}$ is the $M_i \times (M - M_i)$ uniformly random matrix. The other rows of $\mathbf{G}_i$ are uniformly random. The batches generated can be transmitted following an arbitrary order.

Define $\tilde{\mathbf{G}}_i$ as the $K' \times M$ matrix by inserting zero rows into $\mathbf{G}_i$ so that $[\mathbf{B}\ \mathbf{B}_p]\tilde{\mathbf{G}}_i = \mathbf{B}_i\mathbf{G}_i$. The *overall generator matrix* of ENC_{n_s} can be written as $\tilde{\mathbf{G}} = \begin{bmatrix}\tilde{\mathbf{G}}_1 & \cdots & \tilde{\mathbf{G}}_{n_s}\end{bmatrix}$. According to the design of ENC_{n_s}, $\tilde{\mathbf{G}}$ is of the form in Figure 5.

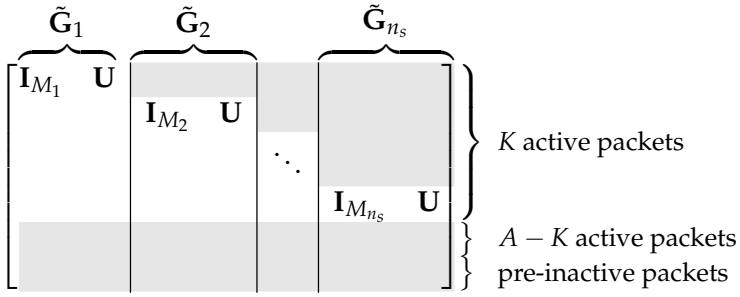


Figure 5. Illustration of encoding using triangular embedding. The gray part contains non-zero entries and the white part contains only zero.

4.2. Decoder Design

It is possible to use the decoders discussed in Section 2.3 to decode the batches generated by triangular embedding. However, due to the structure of the triangular embedding encoding, the decoder can be simplified.

We design a decoder DEC_{n_s} using only the first M_i packets of the ith batch, $i = 1, \ldots, n_s$. The overall generator matrix $\tilde{\mathbf{G}}^*$ of $\text{ENC}_{n_s}^*$ is of the form in Figure 6.

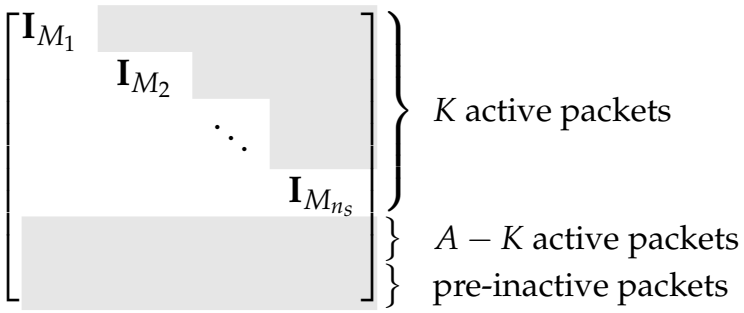


Figure 6. Illustration of decoding using triangular embedding. The gray part contains non-zero entries and the white part contains only zero.

An inactivation decoder can be applied to decode the message packets:

- First, inactivate all the packets used by the first n_s batches among the last $A-K$ active packets;
- Second, apply belief propagation decoding to solve all the batches;
- Last, solve the inactive packets.

Note that as, at most, $\min\{A-K, N_{\text{inac}}\}$ packets are used among the last $A-K$ active packets during encoding, the total number of inactive packets is no more than $N_{\text{inac}}+K'-A$.

4.3. Design Verification

We verify the triangular embedding design from two aspects. First, it can help to generate zero-coding-overhead consistent outer codes using a small number of random trials. Second, when jointly decoded with batches generated by the ordinary outer code, the decoding performance is similar to the case of decoding only the batches generated by the ordinary outer code.

We perform the experiments using the batch size $M=16$ and the rank-M rank distribution $\mathbf{h}$. As with the experiments in Section 3.4, we use the BATS code implementation in [45] with the parameters in Appendix B. The experimental results of the triangular embedding outer code are shown in Table 2. We see that, using triangular embedding, for K up to $1000M$, more than 99.5% instances are of zero coding overhead. In fact, for the remaining instances that are not of zero coding overhead, the coding overhead is only 1 packet (generated using the ordinary outer code). The last row in Table 2 gives the maximum number of inactive packets for all the instances tested for each value of K. We see that the number of inactivations is lower than 150, the number of inactivations in the random design. Therefore, diagonal embedding also reduces the decoding computation cost.

Table 2. Experiments using triangular embedding for consistent outer codes. Here, $M=16$ and $\mathbf{h}$ has rank M with probability 1. For each value of $K=10M, 100M, 1000M$, in total, 5000 instances of the triangular embedded outer code are tested. The table gives the number of consistent instances.

	$K=10M$	$K=100M$	$K=1000M$
zero overhead	4978	4983	4977
max total inact.	27	91	149

As ENC_{n_s} uses a different encoding approach to the ordinary outer code ENC_{n_s+}, we consider whether the batches generated by diagonal embedding and the batches generated by the ordinary outer code together form a good outer code. We perform some numerical experiments to verify the joint decoding performance of these two types of batches. For each batch generated by triangular embedding, we discard the batch with probability $\epsilon=0.1, 0.3, 0.5$ and send the remaining batches to the decoder. After the first n_s batches,

the ordinary outer code is applied to generate more batches for the decoder. We adopt the same degree distribution $\mathbf{\Psi}$ optimized for the rank-M distribution. The results are shown in Table 3. We see that for all the cases of ϵ and for $K = 10M, 100M, 200M$, the number of zero-coding-overhead instances is higher than that in Table 1 and the number of instances with a coding overhead larger than $2M$ is lower than that in Table 1. For $K = 400M, 600M$, the decoding performance is similar to that in Table 1 in terms of both the ratio of zero coding overhead and the ratio of coding overhead larger than $2M$.

Table 3. Joint decoding of batches generated by triangular embedding and the ordinary outer code. Here, $M = 16$ and $\mathbf{h}$ has rank M with probability 1. In our experiments, each batch generated by triangular embedding has a probability ϵ of being discarded, and the remaining batches are sent to the decoder. Following the batches generated by triangular embedding, batches generated by the ordinary outer code are also sent to the decoder. For each value of $K = 10M, 100M, 200M, 400M, 600M$ and $\epsilon = 0.1, 0.3, 0.5$, in total, 5000 instances are tested.

Coding Overhead	$K = 10M$	$K = 100M$	$K = 200M$	$K = 400M$	$K = 600M$
(a) $\epsilon = 0.1$					
0	4982	4955	3446	3	0
$1 \sim M$	18	26	69	0	0
$M+1 \sim 2M$	0	3	51	0	0
$> 2M$	0	16	1434	4997	5000
(b) $\epsilon = 0.3$					
0	4983	4511	1037	17	0
$1 \sim M$	17	73	31	0	0
$M+1 \sim 2M$	0	39	25	1	0
$> 2M$	0	377	3907	4982	5000
(c) $\epsilon = 0.5$					
0	4987	4155	1823	5	0
$1 \sim M$	13	107	101	1	0
$M+1 \sim 2M$	0	61	72	1	0
$> 2M$	0	677	3004	4993	5000

5. Inner Code for Systematic Batches

In this section, we study the design of the inner code for systematic batches. Based on the discussion in Section 3.3, the decoding complexity at the destination node depends on the number of message packets received. However, using the existing inner coding schemes, the number of message packets in a systematic batch reduces significantly during communication. In the worst case, when no message packets are received at the destination node, the decoding computation cost at the destination node is doubled when compared with the ordinary BATS outer code. To resolve this issue, we discuss how to design the inner code to preserve the message packets in systematic batches.

5.1. Detailed Inner Code Formulation

We first formulate in detail how each network node performs the inner code. We also discuss the existing inner coding schemes for systematic batches.

We consider the inner code on a line network as described in Section 2. As the inner code is performed on each batch individually, we consider a generic systematic batch $\mathbf{X}$ without the subscripts. We assume that the packets in $\mathbf{X}$ are all message packets. By (2), the received packets $\mathbf{Y}^{(u)}$ of the batch $\mathbf{X}$ at node u satisfy

$$\mathbf{Y}^{(u)} = \mathbf{X}\mathbf{H}^{(u)}, \tag{5}$$

where $\mathbf{H}^{(u)}$ is the transfer matrix of the batch at the node u.

Let N_u be the number of columns of $\mathbf{Y}^{(u)}$ (or $\mathbf{H}^{(u)}$), i.e., the number of received packets of the batch at node u. For a non-destination node u, we use $u+$ to denote the receiver of the outgoing link of u in the line network. Suppose that the node u needs to transmit N'_u packets of the batch $\mathbf{X}$ to the node $u+$. The transmitted packets, called *recoded packets*, are generated by linear combinations as $\mathbf{Y}^{(u)}\Phi^{(u)} = \mathbf{X}\mathbf{H}^{(u)}\Phi^{(u)}$, where $\Phi^{(u)}$ is an $N_u \times N'_u$ matrix over the base field $\mathbb{F}_q$. Due to packet loss, the set of received packets at $u+$ is a subset of $\mathbf{Y}^{(u)}\Phi^{(u)}$. Let $\mathbf{E}^{(u)}$ be an $N'_u \times N_{u+}$ matrix obtained by removing the columns of an identity matrix specifying the packet erasures. We can write

$$\mathbf{Y}^{(u+)} = \mathbf{X}\mathbf{H}^{(u)}\Phi^{(u)}\mathbf{E}^{(u)} = \mathbf{X}\mathbf{H}^{(u+)}, \tag{6}$$

where $\mathbf{H}^{(u+)} = \mathbf{H}^{(u)}\Phi^{(u)}\mathbf{E}^{(u)}$.

There are many solutions to design the recoding matrix $\Phi^{(u)}$ in the literature. One common method for RLNC is a uniformly random matrix over the base field, which is also called the *random linear inner code* (RLIC). For multicast communications, it has been shown that RLIC achieves the multicast capacity for networks with packet loss [5,8–10]. For the line network discussed here, the *systematic inner code* (SIC) has been proposed [23], where all the linearly independent received packets are directly used as recoded packets. We first discuss the performance of these two existing inner code schemes for systematic batches.

- When using RLIC for systematic batches, the probability that a recoded packet (a column of $\mathbf{X}\Phi^{(u)}$) is a message packet is q^{-M}.
- When using SIC for systematic batches, if the network links have no packet loss, the destination node receives all the message packets without decoding. If each link has an erasure probability $\epsilon > 0$ independently, the number of received message packets at the destination node drops exponentially rapidly with L increasing.

In other words, for both RLIC and SIC, the destination node cannot benefit from the systematic outer code.

We are motivated to study the recoding $\Phi^{(u)}$ such that a non-source node u can receive more message packets from a systematic batch even when there are packet losses.

5.2. Recovery of Individual Message Packets

Although $\mathbf{Y}^{(u)}$ does not include any message packets, it may be possible to decode some message packets from (5). When $\mathrm{rank}(\mathbf{H}^{(u)}) = M$, all the message packets of a batch can be decoded at node u. Note that for batched network coding, $\mathbf{H}^{(u)}$ does not necessarily need to be of rank M. We say that a message packet, i.e., a column of $\mathbf{X}$, can be recovered at node u if it can be uniquely solved from the system (5). When $\mathrm{rank}(\mathbf{H}^{(u)}) < M$, some of the message packets can be recovered by operations within a systematic batch.

Denote by $\mathrm{Col}(\mathbf{H}^{(u)})$ the column space of the matrix $\mathbf{H}^{(u)}$. Let $\mathbf{e}_i$ be the length-M column vector with its ith entry 1 and all the other entries 0. A necessary and sufficient condition such that a message packet can be recovered from $\mathbf{Y}^{(u)}$ is as follows.

Lemma 1. *Under the condition that* $\mathbf{Y}^{(u)} = \mathbf{X}\mathbf{H}^{(u)}$ *is consistent, the ith packet in* $\mathbf{X}$ *has a unique solution if and only if* $\mathbf{e}_i \in \mathrm{Col}(\mathbf{H}^{(u)})$.

Proof. The lemma can be proven by the equivalence of the following statements:

1. The ith packet in $\mathbf{X}$ has a unique solution;
2. All the vectors $\mathbf{x} \in \mathbb{F}_q^{N_u}$ such that $\mathbf{x}\mathbf{H}^{(u)} = \mathbf{0}$ (called the left nullspace collectively) have the ith entry 0;
3. $\mathbf{e}_i$ is orthogonal to the left nullspace of $\mathbf{H}^{(u)}$;
4. $\mathbf{e}_i$ is in the column space of $\mathbf{H}^{(u)}$. □

The following proposition shows that we can test the recoverability of all the message packets in a systematic batch from the reduced column echelon form of $\mathbf{H}^{(u)}$, which can be obtained by (column-wise) Gauss–Jordan elimination.

Proposition 1. *Let* $\mathbf{L}$ *be the reduced column echelon form for a matrix* $\mathbf{H}^{(u)}$*. Then,* $\mathbf{e}_i \in \mathrm{Col}(\mathbf{H}^{(u)})$ *if and only if* $\mathbf{e}_i$ *is a column of* $\mathbf{L}$*.*

Proof. See Appendix A. □

The next proposition shows that if a message packet cannot be recovered at a node, it cannot be recovered at any of the following nodes. Equivalently, if a message packet can be recovered at a node, it can be recovered at all the previous nodes.

Proposition 2. *If a message packet cannot be recovered at the node* u*, then it cannot be recovered at the node* $u+$ *on the next hop.*

Proof. If the ith message packet cannot be recovered at the node u, by Lemma 1, we have $\mathbf{e}_i \notin \mathrm{Col}(\mathbf{H}^{(u)})$. Due to $\mathrm{Col}(\mathbf{H}^{(u+)}) = \mathrm{Col}(\mathbf{H}^{(u)}\Phi^{(u)}\mathbf{E}^{(u)}) \subseteq \mathrm{Col}(\mathbf{H}^{(u)})$, $\mathbf{e}_i \notin \mathrm{Col}(\mathbf{H}^{(u+)})$ and hence the ith message packet cannot be recovered at the node u^+. □

In general, performing an elementary operation as used in Gauss–Jordan elimination on the received packets of a batch does not affect the rank of the batch, and hence does not affect the decoding performance. However, recovering message packets at the intermediate nodes helps to improve the number of message packets to be received/recovered in the next hop. We use an example to illustrate this fact.

Example 1. *Consider a line network with* $L = 2$*,* $M = 3$ *and* $N_1 = M$ *at node* 1*. Suppose that*

$$\mathbf{H}^{(1)} = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & c \end{bmatrix},$$

where $a, b, c \neq 0$ *are elements from the base field. Using systematic recoding on* $\mathbf{H}^{(1)}$*, no additional packets should be generated and* $\mathbf{Y}^{(1)}$ *is transmitted by node* 1*. When the second packet is lost from node* 1 *to* 2*, we obtain*

$$\mathbf{H}^{(2)} = \begin{bmatrix} 1 & a \\ 0 & b \\ 0 & c \end{bmatrix},$$

At destination node 2, we can only recover one message packet. On the other hand, suppose that we apply the Gaussian elimination step at node 1 *and the result should be* $\mathbf{H}^{(1)}\mathbf{D} = \mathbf{I}$*. Then, node* 1 *transmits* $\mathbf{Y}^{(1)}\mathbf{D}$ *instead of* $\mathbf{Y}^{(1)}$*. In this case, if we still erase the second packet, the following node can recover* 2 *message packets. Moreover, since the Gaussian elimination step preserves the column space of the batch transfer matrix, (*$\mathrm{Col}(\mathbf{H}^{(u)}) = \mathrm{Col}(\mathbf{H}^{(u)}\mathbf{D})$*), the rank and number of recoverable message packets at the destination node should be at least as good as in the recoding schemes without this step.*

Note that the recovery of the message packets at an intermediate node is a linear operation on a batch and hence can be regarded as a part of the inner code. The effect of the recovery of the message packets can be captured by the coefficient vectors: the same operation applied on the received packets of a batch is applied on the coefficient vectors as well. The destination node does not need to know the exact operations at each intermediate, but only the coefficient vectors of the received packets.

5.3. Side Information for Message Packet Recovery

We discuss some general properties involved in the recovery of message packets at the node $u+$, which provide guidance for the design of new inner codes. The recoverability of a message packet depends on the knowledge of $\mathbf{H}^{(u)}$, which is delivered by the coefficient vectors. Note that the original purpose of the coefficient vectors is for the destination node

to decode the batches. A natural question to consider is the following: if more information is delivered from node u to $u+$, could more message packets be recovered at node $u+$?

Proposition 3. *Suppose that* $\mathbf{X}$, $\mathbf{H}^{(u)}$, $\Phi^{(u)}$ *and* $\mathbf{E}^{(u)}$ *in* (6) *are mutually independent.* $\Phi^{(u)}$ *and* $\mathbf{X}$ *are conditionally independent given* $\mathbf{H}^{(u+)}$ *and* $\mathbf{Y}^{(u+)}$.

Proof. See Appendix A. □

The above proposition states that $\Phi^{(u)}$ and $\mathbf{X}$ are conditionally independent at the node $u+$. The next proposition further shows that knowing $\Phi^{(u)}$ at the node $u+$ does not help to recover more message packets at node $u+$. It actually shows a stronger result that knowing any variable that is independent with $\mathbf{X}$ given $\mathbf{H}^{(u+)}$ and $\mathbf{Y}^{(u+)}$ at the node $u+$ does not help in recovering more message packets at the node $u+$.

Proposition 4. *Suppose that* $\mathbf{X}$, $\mathbf{H}^{(u)}$, $\Phi^{(u)}$ *and* $\mathbf{E}^{(u)}$ *in* (6) *are mutually independent. Let* S *be any random variable that is conditionally independent with* $\mathbf{X}$ *given* $\mathbf{H}^{(u+)}$ *and* $\mathbf{Y}^{(u+)}$. *Given the instance of* $\mathbf{H}^{(u+)}$ *and* $\mathbf{Y}^{(u+)}$ *at the node* $u+$, *further knowing the instance of* S *at the node* $u+$ *does not help to recover more message packets at* $u+$.

Proof. See Appendix A. □

Based on the above analysis, we know that the existing coefficient vectors are sufficient for the recovery of message packets at the intermediate nodes. In other words, it is not necessary for a network node to add further information to assist the recovery of the message packets in the following nodes.

5.4. Recoding with Message Packet Protection

Let $r = \mathrm{rank}(\mathbf{H}^{(u)})$ and $\mathbf{V}$ be an $N_u \times N_u$ matrix such that $\mathbf{H}^{(u)}\mathbf{V}$ is in reduced column echelon form. To recover message packets, we perform the same column operations on $\mathbf{Y}^{(u)}$ and obtain $\mathbf{Y}^{(u)}\mathbf{V} = \mathbf{X}\mathbf{H}^{(u)}\mathbf{V}$. If $\mathbf{e}_i$ is the jth column of $\mathbf{H}^{(u)}\mathbf{V}$, then the jth column of $\mathbf{Y}^{(u)}\mathbf{V}$ is equal to the ith message packet.

Let s be the number of message packets that can be recovered from $\mathbf{Y}^{(u)}$. By Proposition 1, there are exactly s distinct columns in $\mathbf{H}^{(u)}\mathbf{V}$ with only 1 non-zero entry being one. Therefore, by proper row and column permutations, $\mathbf{H}^{(u)}\mathbf{V}$ is of the form

$$\begin{bmatrix} \mathbf{I}_s & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{r-s} & \mathbf{0} \\ \mathbf{0} & \mathbf{T} & \mathbf{0} \end{bmatrix}, \tag{7}$$

where $\mathbf{I}_k$ is the $k \times k$ identity matrix, $\mathbf{0}$ is an all-zero matrix of proper size, and $\mathbf{T}$ is an $(N_u - r) \times r$ matrix where each column is not zero.

Denoting the first r columns of the corresponding column permutation matrix as the $N_u \times r$ matrix $\mathbf{P}$, each of the first s columns of $\mathbf{H}^{(u)}\mathbf{V}\mathbf{P}$ has only 1 non-zero entry.

We have discussed the decoding step, which is represented by $\mathbf{V}$. However, to generate a recoded batch, some redundant packets are to be generated. The following proposition states that using the random linear inner code at node u, the node $u+$ can recover almost no message packets when the number of received packets at $u+$ is fewer than $\mathrm{rank}(\mathbf{H}^{(u)})$. Denote by $\zeta_k^{m,n}$ the probability that an $m \times n$ uniformly random matrix over $\mathbb{F}_q$ has rank k. See, e.g., ([23], Section 3.3.2) for the formula of $\zeta_k^{m,n}$.

Proposition 5. *Suppose that the random linear inner code over* $\mathbb{F}_q$ *is used at the node* u *and* $N_{u+} < r = \mathrm{rank}(\mathbf{H}^{(u)})$. *Under the condition that* $\mathbf{e}_i \in \mathrm{Col}(\mathbf{H}^{(u)})$, *the probability that* $\mathbf{e}_i \in \mathrm{Col}(\mathbf{H}^{(u+)})$ *is* $1 - \sum_{k=0}^{N_{u+}} \zeta_k^{r-1,N_{u+}} q^{k-N_{u+}}$ *and it converges to zero as* $q \to \infty$.

Proof. See Appendix A. □

It is unavoidable that the number of received packets at $u+$ is smaller than $\text{rank}(\mathbf{H}^{(u)})$ due to packet loss. Together with Proposition 2, Proposition 5 implies that as long as the event $N_{u+} < \text{rank}(\mathbf{H}^{(u)})$ occurs once at some node u, the destination node receives almost no message packets from a systematic batch. Therefore, random linear recoding is not preferred for the recovery of message packets. Thus, we are motivated to extend systematic inner codes for the recovery of message packets.

We propose two designs of recoding that can protect the message packets during recoding. We first define two recoding matrices. Suppose that s message packets are recoverable at the node u and the rank of the batch is r.

Message Protection Recoding

For an integer w with $0 \leq w \leq N'_u - s$, let $\mathbf{R}$ be an $r \times N'_u$ matrix of the form

$$\mathbf{R} = \begin{bmatrix} \mathbf{I}_s & \mathbf{U}_{s,w} & \mathbf{U}_{r,N'_u-s-w} \\ \mathbf{0} & \mathbf{0} & \end{bmatrix},$$

where $\mathbf{U}_{m,n}$ is the $m \times n$ uniformly random matrix.

Systematic Message Protection Recoding

For an integer w with $0 \leq w \leq N'_u - s$, let $\mathbf{R}^{\text{sys}}$ be an $r \times N'_u$ matrix of the form: when $w < N'_u - r$,

$$\mathbf{R}^{\text{sys}} = \begin{bmatrix} \mathbf{I}_s & \mathbf{U}_{s,w} & \mathbf{0} & \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{r-s} & \mathbf{U}_{r,N'_u-r-w} \end{bmatrix};$$

when $w \geq N'_u - r$,

$$\mathbf{R}^{\text{sys}} = \begin{bmatrix} \mathbf{I}_s & \mathbf{U}_{s,w} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{J} \end{bmatrix},$$

where $\mathbf{J}$ is the first $N'_u - w - s$ columns of $\mathbf{I}_{r-s}$.

The inner code operations at node u consist of (i) the Gauss–Jordan elimination represented by the matrix $\mathbf{V}$, (ii) the column permutation and removal of the all-zero columns represented by the matrix $\mathbf{P}$, and (iii) (systematic) message protection recoding $\mathbf{R}$ ($\mathbf{R}^{\text{sys}}$). When the overall recoding matrix at node u is $\mathbf{VPR}$, the inner code is called the *message protection inner code (MPIC)*. When the overall recoding matrix at node u is $\mathbf{VPR}^{\text{sys}}$, the inner code is called the *systematic message protection inner code (SMPIC)*.

The value of w controls the level of protection of message packets. When $w = 0$, no additional protection is provided for message packets, and we can check that SMPIC has the same rank performance as the systematic inner code. When $w = N'_u - s$, all recoded packets generated by linear combinations are used to protect the message packets.

The computation cost of the proposed message protection recoding at a network node is mainly determined by (1) the Gauss–Jordan elimination for the recovery of the message packets, and (2) the generation of the recoded packets. To simplify the discussion, we only consider the case with $w = 0$. At node u, Gauss–Jordan elimination is applied on the N_u received packets. As the packet length T is much larger than the batch size M, the computation cost of processing the transfer matrix $\mathbf{H}^{(u)}$ is ignored. Hence, when the rank of $\mathbf{H}^{(u)}$ is r, the computation cost of recovering the message packets is about $r(N_u - 1)$ LCOs. If the previous node also uses message protection recoding, the cost at node u can be reduced, as the message packets received directly can help to simplify the Gauss–Jordan elimination. Let s_0 be the message packet received at node u, and we have $s_0 \leq s \leq r$. In this case, the computation cost for Gauss–Jordan elimination is about $(r - s_0)(N_u - 1)$ LCOs. For a batch of rank r, the cost of generating recoded packets using $\mathbf{R}$ or $\mathbf{R}^{\text{sys}}$ is linear with the number of entries in uniformly random sub-matrices. Therefore, the overall recoding computation cost for SMPIC with $w = 0$ is about $((r - s_0)(N_u - 1) + r(N'_u - r))$ LCOs. In contrast, for RLIC, the computation cost is $N'_u N_u$ LCOs, and for SIC, the computation cost is $(N'_u - N_u)N_u$ LCOs.

5.5. Numerical Evaluations

We perform numerical evaluations to verify the performance of the new inner codes in terms of both the average rank and the average number of recoverable message packets and compare it with that of the random linear inner code (RLIC) and the systematic inner code (SIC). We use line networks of length up to 50 hops, where each link has the same independent packet erasure probability 0.2. The batch size $M = 16$ and the number of packets to transmit $N'_u = 20$ for all nodes u. Since the performance of SMPIC and MPIC shows negligible differences in simulation, we only show the results for SMPIC, where we evaluate $w = 0$ and $w = N'_u - s$ as representatives.

Our numerical evaluation results are shown in Figure 7. We plot the average number of recoverable message packets and the average rank at node 0 to 50 for SIC, RLIC, SMPIC with $w = 0$ (denoted by SMPIC0) and SMPIC with $w = N'_u - s$ (denoted by SMPIC1), each with 500 trials. We have the following observations.

- SIC and RLIC have almost the same rank performance. SIC has a larger number of recoverable message packets than RLIC. However, for both SIC and RLIC, the number of recoverable message packets drops quickly.
- SMPIC0 has similar rank performance to SIC and RLIC and has a much higher average number of recoverable message packets than that of SIC and RLIC.
- SMPIC1 has the highest average number of recoverable message packets among the four inner codes, at the cost of a reduced average rank.

The recoding computation costs at each network node are also determined in the experiments and are illustrated in Figure 8. For RLIC, as $N'_u = 20$ and the expectation of $N_u = 16$, the recoding computation cost is about 320 LCOs. For SIC, the recoding computation cost is about 64 LCOs. The recoding computation cost of SMPIC0 also matches the formula that we have derived, where the expectation of s_0 is $1 - \epsilon = 0.8$ multiplied by the number of recovered message packets in the previous hop. In Figure 8, we also show the computation cost of SMPIC1, which is close to that of SMPIC0.

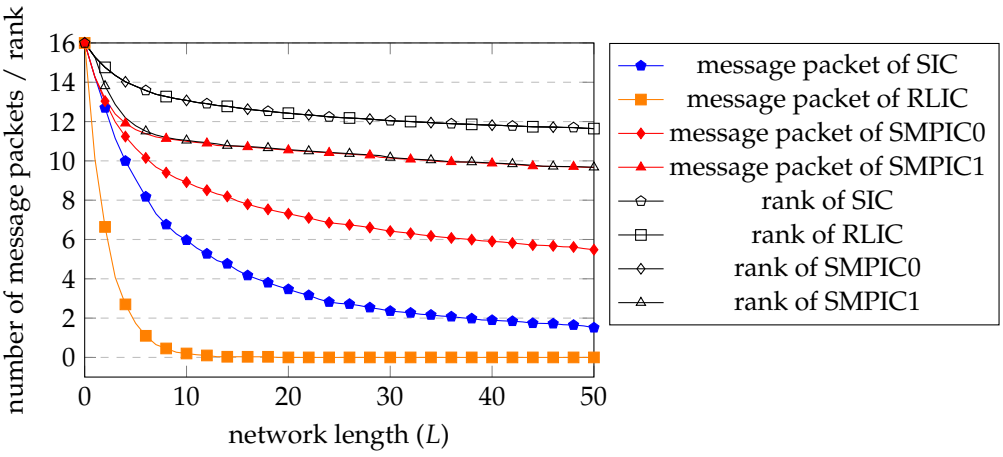


Figure 7. The average number of recovered message packets and the average rank at node 0 to 50 for SIC, RLIC, SMPIC with $w = 0$ (denoted by SMPIC0) and SMPIC with $w = N'_u - s$ (denoted by SMPIC1), each with 500 trials.

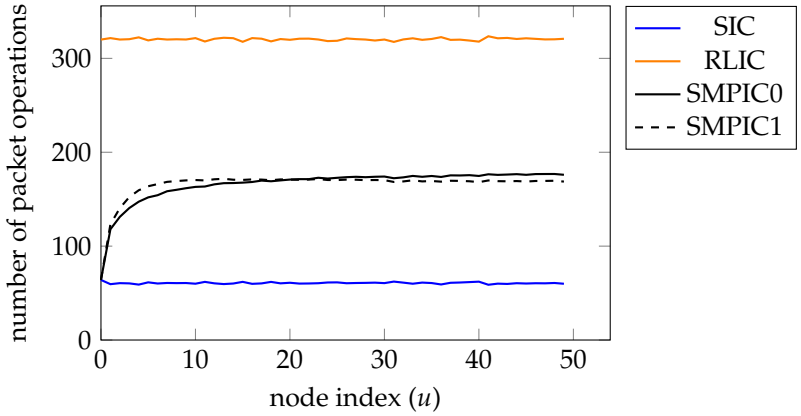


Figure 8. The average number of linear combination operations at node 0 to node 49 for SIC, RLIC, SMPIC with $w = 0$ (denoted by SMPIC0) and SMPIC with $w = N'_u - s$ (denoted by SMPIC1), each with 500 trials.

6. Concluding Remarks

In this paper, we propose a design for systematic batched network codes, where the outer code is systematic and the inner code can protect the systematic property during network coding. Our design of the systematic code preserves the most salient features of the BATS code. The diagonal embedding approach is proposed to improve the design efficiency of the systematic outer code, and it can also be used for non-systematic outer coding to reduce the coding overhead and computation cost.

The discussion in this paper can help to evaluate when and how to adopt systematic batched network codes. When the computation cost and the encoding latency are the major concerns, the use of systematic outer codes is preferred due to the lower computation cost compared to the ordinary outer code. The decision regarding whether to use message protection recoding depends on both the computation constraints and the application scenario. When the decoding computation is sensitive and the intermediate nodes have an additional computation capability, it is beneficial to use message protection recoding. Message protection recoding is also preferred for some application scenarios, e.g., for communications where part of the content can be consumed when ready, a systematic code is better. Another useful scenario for systematic codes is a network with dynamic network link qualities: the communication is reliable most of the time and serious packet loss occurs only in a small fraction of the time.

There are still many refinements to be applied for the systematic batched network codes. This paper focused on the inner code design for unicast communications. The current inner codes designed to protect the message packets may not be suitable to achieve the multicast gain of network coding. Further study of the inner code design for multicast communication is desired.

7. Patents

Patents resulting from this work are listed in the following:

CN115811381A The design framework of the systematic BATS code (including the outer code and inner code), invented by the authors of this paper, published on 17 March 2023.

CN2023105394085 The design of the triangular embedded outer code, invented by L.M. and S.Y., filed on 15 May 2023.

Author Contributions: Conceptualization, L.M. and S.Y.; methodology, L.M. and S.Y.; implementation, L.M.; validation, L.M.; formal analysis, L.M., Y.D., X.H. and S.Y.; writing—original draft preparation, L.M. and S.Y.; writing—review and editing, L.M., Y.D., X.H. and S.Y.; visualization, L.M. and S.Y.; supervision, S.Y.; project administration, S.Y.; funding acquisition, S.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by NSFC under Grants 62171399 and 12141108.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors wish to acknowledge that the research conducted by Dong in this paper was performed while she was affiliated with CUHK-Shenzhen.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

BATS code	BATched Sparse code
HDPC	High-Density Parity Check
LDPC	Low-Density Parity Check
LCO	Linear Combination Operation
MPIC	Message Protection Inner Code
RLIC	Random Linear Inner Code
RLNC	Random Linear Network Codes/Random Linear Network Coding
SIC	Systematic Inner Code
SMPIC	Systematic Message Protection Inner Code

Appendix A. Proofs Regarding Recoverability of Message Packets

Proof of Proposition 1. For the sufficiency, if $\mathbf{e}_i$ is a column of $\mathbf{L}$, then $\mathbf{e}_i \in \mathrm{Col}(\mathbf{L}) = \mathrm{Col}(\mathbf{H}^u)$.

Now, we prove the necessity. If $\mathbf{e}_i \in \mathrm{Col}(\mathbf{H}^u)$, then $\mathbf{e}_i \in \mathrm{Col}(\mathbf{L})$. Let $r = \mathrm{rank}(\mathbf{H}^u)$. By the property of reduced column echelon form, all the zero columns are on the right of the non-zero columns in $\mathbf{L}$, and hence the first r columns are all the non-zero columns of $\mathbf{L}$. Then, we can write $\mathbf{e}_i = \sum_{j=1}^{r} c_j \mathbf{l}_j$, where c_j is a constant and $\mathbf{l}_j$ is the jth non-zero column of $\mathbf{L}$.

For $j = 1, \ldots, r$, denote by i_j the row index of the leading 1 of $\mathbf{l}_j$, which must exist due to the property of reduced column echelon form. Further, as the (i_j, j) entry is the only non-zero entry on the i_jth row of $\mathbf{L}$, $c_j = 0$ for j such that $i_j \neq i$. If there exists no j such that $i_j = i$, then $\mathbf{e}_i = \mathbf{0}$, a contradiction. Therefore, there must a unique j^* such that $i_{j^*} = i$, and hence $\mathbf{e}_i = \mathbf{l}_{j^*}$. □

Proof of Proposition 3. Denote by $\mathbf{x}$, $\mathbf{y}^+$, $\mathbf{h}^+$ and ϕ the instances of $\mathbf{X}$, $\mathbf{Y}^{u+}$, $\mathbf{H}^{u+}$ and Φ^u, respectively. As $P(\mathbf{x}, \phi | \mathbf{y}^+, \mathbf{h}^+) = P(\phi | \mathbf{y}^+, \mathbf{h}^+) P(\mathbf{x} | \mathbf{y}^+, \mathbf{h}^+, \phi)$; to prove this claim, it is sufficient to show that

$$P(\mathbf{x} | \mathbf{y}^+, \mathbf{h}^+, \phi) = P(\mathbf{x} | \mathbf{y}^+, \mathbf{h}^+) \tag{A1}$$

for all instances $\mathbf{x}$, $\mathbf{y}^+$, $\mathbf{h}^+$ and ϕ. If $\mathbf{y}^+ \neq \mathbf{x}\mathbf{h}^+$, (A1) holds as both sides are 0.

Suppose that $\mathbf{y}^+ = \mathbf{x}\mathbf{h}^+$. As $\mathbf{X}, \mathbf{H}^u, \Phi^u$ and $\mathbf{E}^u$ are independent and $\mathbf{H}^{u+} = \mathbf{H}^u\Phi^u\mathbf{E}^u$, we obtain

$$\begin{aligned} P(\mathbf{x}|\mathbf{y}^+, \mathbf{h}^+, \phi) &= \frac{P(\mathbf{x})P(\mathbf{h}^+, \phi)}{P(\mathbf{y}^+, \mathbf{h}^+, \phi)} \\ &= \frac{P(\mathbf{x})P(\mathbf{h}^+, \phi)}{\sum_{\mathbf{x}':\mathbf{x}'\mathbf{h}^+=\mathbf{y}^+} p(\mathbf{x}')P(\mathbf{h}^+, \phi)} \\ &= \frac{P(\mathbf{x})}{\sum_{\mathbf{x}':\mathbf{x}'\mathbf{h}^+=\mathbf{y}^+} p(\mathbf{x}')}. \end{aligned}$$

Similarly,

$$\begin{aligned} P(\mathbf{x}|\mathbf{y}^+, \mathbf{h}^+) &= \frac{P(\mathbf{x})P(\mathbf{h}^+)}{P(\mathbf{y}^+, \mathbf{h}^+)} \\ &= \frac{P(\mathbf{x})P(\mathbf{h}^+)}{\sum_{\mathbf{x}':\mathbf{x}'\mathbf{h}^+=\mathbf{y}^+} p(\mathbf{x}')P(\mathbf{h}^+)} \\ &= \frac{P(\mathbf{x})}{\sum_{\mathbf{x}':\mathbf{x}'\mathbf{h}^+=\mathbf{y}^+} p(\mathbf{x}')}. \end{aligned}$$

Therefore, (A1) holds when $\mathbf{y}^+ = \mathbf{x}\mathbf{h}^+$. The proof is completed. □

Proof of Proposition 4. Denote by $\mathbf{x}$, $\mathbf{y}^+$, $\mathbf{h}^+$ and s the instances of $\mathbf{X}$, $\mathbf{Y}^{u+}$, $\mathbf{H}^{u+}$ and S, respectively. It is sufficient to show that $P(\mathbf{x}|\mathbf{y}^+, \mathbf{h}^+, s) = P(\mathbf{x}|\mathbf{y}^+, \mathbf{h}^+)$ for all instances $\mathbf{x}, s, \mathbf{y}^+, \mathbf{h}^+$. If $\mathbf{y}^+ \neq \mathbf{x}\mathbf{h}^+$, the equality holds as both sides are 0. Suppose that $\mathbf{y}^+ = \mathbf{x}\mathbf{h}^+$. As $\mathbf{X}$ and S are independent given $\mathbf{H}^{u+}$ and $\mathbf{Y}^{u+}$, we obtain

$$\begin{aligned} P(\mathbf{x}|\mathbf{y}^+, \mathbf{h}^+, s) &= \frac{P(\mathbf{x}, s|\mathbf{y}^+, \mathbf{h}^+)}{P(s|\mathbf{y}^+, \mathbf{h}^+)} \\ &= \frac{P(s|\mathbf{y}^+, \mathbf{h}^+)P(\mathbf{x}|\mathbf{y}^+, \mathbf{h}^+)}{P(s|\mathbf{y}^+, \mathbf{h}^+)} \\ &= P(\mathbf{x}|\mathbf{y}^+, \mathbf{h}^+). \quad \square \end{aligned}$$

Proof of Proposition 5. For convenience, we omit the subscripts of $\mathbf{H}^u$, Φ^u and $\mathbf{E}^u$.

Assume that $\mathbf{H} \in \mathbb{F}_q^{M\times N_u}$ is fixed with $\text{rank}(\mathbf{H}) = r$ and $\mathbf{e}_i \in \text{Col}(\mathbf{H})$. Since $\text{rank}(\mathbf{H}) = r$, and $\mathbf{e}_i \in \text{Col}(\mathbf{H})$, we can extend $\{\mathbf{e}_i\}$ to a basis of $\text{Col}(\mathbf{H})$, denoted by $\mathbf{W}$. Then, there exists a full row rank matrix $\mathbf{C} \in \mathbb{F}_q^{r\times N_u}$ such that $\mathbf{H} = \mathbf{W}\mathbf{C}$ and $\mathbf{H}^{u+} = \mathbf{W}\mathbf{C}\Phi\mathbf{E}$. Let $\Phi^* = \Phi\mathbf{E}$; then, Φ^* is an $N_u \times N_{u+}$ uniformly random matrix.

Notice that $\mathbf{C}$ is full row rank, and $\mathbf{C}$ can be written as $\mathbf{C} = \mathbf{K}\mathbf{C}'$, where $\mathbf{C}'$ is an invertible matrix with the first r rows being $\mathbf{C}$ and $\mathbf{K}$ is made up of the first r rows of an identity matrix. Since $\mathbf{C}'\Phi^*$ is still an $N_u \times N_{u+}$ uniformly random matrix, we have that $\mathbf{C}\Phi^*$ is an $r \times N_{u+}$ uniformly random matrix. In the following, we let $\mathbf{M} = \mathbf{C}\Phi^*$ and we have $\mathbf{e}_i \in \text{Col}(\mathbf{H}^{u+})$ if and only if $\mathbf{e}_1 \in \text{Col}(\mathbf{M})$. Let $\boldsymbol{m}^T$ be the first row of $\mathbf{M}$ and $\tilde{\mathbf{M}}$ be the submatrix of $\mathbf{M}$ with the first row removed. Then, $\mathbf{e}_1 \in \text{Col}(\mathbf{M})$ is equivalent to $\exists \mathbf{x}$ s.t. $\tilde{\mathbf{M}}\mathbf{x} = \mathbf{0}, \boldsymbol{m}^T\mathbf{x} \neq 0$; in other words, $\boldsymbol{m} \not\perp \text{Null}(\tilde{\mathbf{M}})$.

When $\tilde{\mathbf{M}}$ has rank k, the null space of $\tilde{\mathbf{M}}$ has dimension $N_{u+} - k$. The probability $\boldsymbol{m} \not\perp \text{Null}(\tilde{\mathbf{M}})$ is $1 - \frac{q^k}{q^{N_{u+}}}$.

Therefore, the probability $\mathbf{e}_1 \in \mathrm{Col}(\mathbf{M})$ is:

$$\begin{aligned}\Pr(\mathbf{e}_1 \in \mathrm{Col}(\mathbf{M})) &= \sum_{k=0}^{N_{u+}} \zeta_k^{r-1,N_{u+}} \left(1 - \frac{q^k}{q^{N_{u+}}}\right) \\ &= 1 - \sum_{k=0}^{N_{u+}} \zeta_k^{r-1,N_{u+}} q^{k-N_{u+}}.\end{aligned}$$

Observe that

$$\begin{aligned}\Pr(\mathbf{e}_1 \in \mathrm{Col}(\mathbf{M})) &= \sum_{k=0}^{N_{u+}-1} \zeta_k^{r-1,N_{u+}} \left(1 - \frac{q^k}{q^{N_{u+}}}\right) \\ &\leq \sum_{k=0}^{N_{u+}-1} \zeta_k^{r-1,N_{u+}} \\ &= 1 - \zeta_{N_{u+}}^{r-1,N_{u+}} \\ &= 1 - \sum_{i=0}^{N_{u+}-1} (1 - q^{-r+1+i}).\end{aligned}$$

Since $\sum_{i=0}^{N_{u+}-1}(1 - q^{-r+1+i}) \to 1$ as $q \to \infty$, $\Pr(\mathbf{e}_1 \in \mathrm{Col}(\mathbf{M})) \to 0$, as $q \to \infty$. □

Appendix B. BATS Code Parameters Used in Numerical Experiments

For the numerical experiments of the BATS outer code in Sections 3.4 and 4.3, we use the BATS code with the following parameters.

- The batch size M is 16.
- We use the degree distribution $\mathbf{\Upsilon}$ asymptotically optimized for the rank-M rank distribution. The non-zero entries of $\mathbf{\Upsilon}$ are listed in Table A1.
- The following formula determines the number of LDPC packets:

$$\begin{cases} 0.0101K + \sqrt{3K}, & K < 20000 \\ 0.0101K + \sqrt{4K}, & \text{otherwise.} \end{cases}$$

- The number of HDPC packets is $\max(\ln(K), 5)$.
- The decoder has a limit on the number of inactivated packets and the limit is 150.

Table A1. The non-zero entries of the degree distribution used in the numerical experiments.

$\mathbf{\Upsilon}_{17}$	$\mathbf{\Upsilon}_{18}$	$\mathbf{\Upsilon}_{19}$	$\mathbf{\Upsilon}_{20}$	$\mathbf{\Upsilon}_{23}$	$\mathbf{\Upsilon}_{27}$	$\mathbf{\Upsilon}_{31}$	$\mathbf{\Upsilon}_{35}$
0.0588	0.0571	0.0245	0.0899	0.1170	0.0921	0.0678	0.0679
$\mathbf{\Upsilon}_{43}$	$\mathbf{\Upsilon}_{45}$	$\mathbf{\Upsilon}_{63}$	$\mathbf{\Upsilon}_{73}$	$\mathbf{\Upsilon}_{123}$	$\mathbf{\Upsilon}_{126}$	$\mathbf{\Upsilon}_{239}$	
0.0608	0.0604	0.0671	0.0671	0.0599	0.0222	0.0457	

Appendix C. Pseudocodes for BATS Outer Encoding and Decoding

Algorithm A1 is the pseudocode for the encoding of the BATS outer code, and Algorithm A2 is the pseudocode for the two-step decoding of the BATS outer code.

Algorithm A1 The encoding process of the BATS outer code.

Input:

- **B**: all the input packets.
- range: a range of index of batches.

Output:

- **X**: an array of the generated batches.

1: **procedure** ENC(**B**, range)
2: $\mathbf{B}_p \leftarrow$ Solve the precode constraint $\begin{bmatrix}\mathbf{B} & \mathbf{B}_p\end{bmatrix}\mathbf{P} = \mathbf{0}$.
3: $\mathbf{B}' \leftarrow \begin{bmatrix}\mathbf{B} & \mathbf{B}_p\end{bmatrix}$
4: $\mathbf{B}_A, \mathbf{B}_I \leftarrow$ Split $\mathbf{B}'$ into active packets and inactive packets.
5: $\mathbf{X} \leftarrow []$ ▷ Initialize an array for the generated batches.
6: **for** i in range **do** ▷ Line 7 to line 13 generate batches with key i.
7: Use i as the seed for the pseudo random number generator.
8: $d_i^A \leftarrow$ Sample a degree from the degree distribution $\mathbf{\Psi}$.
9: $d_i^B \leftarrow$ Randomly choose an inactive degree.
10: $\mathbf{B}_i \leftarrow$ Randomly choose d_i^A packets from $\mathbf{B}_A$ and d_i^B packets from $\mathbf{B}_I$.
11: $\mathbf{G}_i \leftarrow$ Create a $(d_i^A + d_i^B) \times M$ matrix with independently uniform entries.
12: $\mathbf{X}_i \leftarrow \mathbf{B}_i\mathbf{G}_i$
13: Append $\mathbf{X}_i$ onto the end of $\mathbf{X}$.
14: **return X**

Algorithm A2 The two-step decoding process of the BATS outer code.

Input:

- $[\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_n]$: an array of n batches.
- $[\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_n]$: an array of batch transfer matrices of the n batches.

Output:

- **B**: Recovered input packets

1: **procedure** DEC($[\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_n]$, $[\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_n]$)
2: $\Omega \leftarrow \{1, 2, \ldots, n\}$ ▷ The index set of unsolved batches
3: **for** i in $1 \ldots n$ **do**
4: Use i as the seed for the pseudo number generator.
5: $d_i^A \leftarrow$ Sample a degree from the degree distribution $\mathbf{\Psi}$.
6: $\mathbf{G}_i \leftarrow$ Create a $(d_i^A) \times M$ matrix with independently uniform entries.
7: **while** $\exists i \in \Omega$, such that $d_i^A = \text{rank}(\mathbf{G}_i\mathbf{H}_i)$ **do**
8: $\mathbf{B}_i \leftarrow$ Solve the system $\mathbf{B}_i\mathbf{G}_i\mathbf{H}_i = \mathbf{Y}_i$.
9: **for b** in $\mathbf{B}_i$ **do**
10: Mark **b** as decoded.
11: **for** j in indices of other batch such that $\mathbf{b} \in \mathbf{B}_j$ **do**
12: Update the batch equation $\mathbf{B}_j\mathbf{G}_j\mathbf{H}_j = \mathbf{Y}_j$ by canceling **b** from that equation.
13: Remove i from Ω.
14: $[\mathbf{B}, \mathbf{B}_p] \leftarrow$ Decode the precode using the decoded packets.
15: **return B**

References

1. Ahlswede, R.; Cai, N.; Li, S.Y.R.; Yeung, R.W. Network information flow. *IEEE Trans. Inform. Theory* **2000**, *46*, 1204–1216. [CrossRef]
2. Li, S.Y.R.; Yeung, R.W.; Cai, N. Linear network coding. *IEEE Trans. Inform. Theory* **2003**, *49*, 371–381. [CrossRef]
3. Koetter, R.; Medard, M. An Algebraic Approach to Network Coding. *IEEE/ACM Trans. Netw.* **2003**, *11*, 782–795. [CrossRef]
4. Ho, T.; Leong, B.; Medard, M.; Koetter, R.; Chang, Y.; Effros, M. The benefits of coding over routing in a randomized setting. In Proceedings of the IEEE International Symposium on Information Theory (ISIT '03), Yokohama, Japan, 29 June–4 July 2003. [CrossRef]

5. Ho, T.; Médard, M.; Koetter, R.; Karger, D.R.; Effros, M.; Shi, J.; Leong, B. A Random Linear Network Coding Approach to Multicast. *IEEE Trans. Inform. Theory* **2006**, *52*, 4413–4430. [CrossRef]
6. Jaggi, S.; Chou, P.A.; Jain, K. Low complexity optimal algebraic multicast codes. In Proceedings of the International Symposium on Information Theory (ISIT '03), Yokohama, Japan, 29 June–4 July 2003.
7. Sanders, P.; Egner, S.; Tolhuizen, L. Polynomial time algorithms for network information flow. In Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '03), San Diego, CA, USA, 7–9 June 2003; pp. 286–294.
8. Wu, Y. A Trellis Connectivity Analysis of Random Linear Network Coding with Buffering. In Proceedings of the 2006 IEEE International Symposium on Information Theory, Seattle, WA, USA, 9–14 July 2006; pp. 768–772. [CrossRef]
9. Dana, A.F.; Gowaikar, R.; Palanki, R.; Hassibi, B.; Effros, M. Capacity of wireless erasure networks. *IEEE Trans. Inform. Theory* **2006**, *52*, 789–804. [CrossRef]
10. Yeung, R.W. Avalanche: A network coding analysis. *Commun. Inf. Syst.* **2007**, *7*, 353–358. [CrossRef]
11. Chou, P.A.; Wu, Y.; Jain, K. Practical Network Coding. In Proceedings of the Allerton Conference on Communication, Control, and Computing, Monticello, IL, USA, 29 September–1 October 2004; Invited paper.
12. de Alwis, C.; Kodikara Arachchi, H.; Fernando, A.; Kondoz, A. Towards minimising the coefficient vector overhead in random linear Network Coding. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013; pp. 5127–5131. [CrossRef]
13. Silva, D. Minimum-overhead network coding in the short packet regime. In Proceedings of the 2012 International Symposium on Network Coding (NetCod), Cambridge, MA, USA, 29–30 June 2012; pp. 173–178. . [CrossRef]
14. Gligoroski, D.; Kralevska, K.; Øverby, H. Minimal header overhead for random linear network coding. In Proceedings of the 2015 IEEE International Conference on Communication Workshop (ICCW), London, UK, 8–12 June 2015; pp. 680–685. [CrossRef]
15. Silva, D.; Zeng, W.; Kschischang, F.R. Sparse Network Coding with Overlapping Classes. In Proceedings of the 2009 Workshop on Network Coding, Theory, and Applications (NetCod '09), Lausanne, Switzerland, 15–16 June 2009; pp. 74–79. [CrossRef]
16. Heidarzadeh, A.; Banihashemi, A.H. Overlapped Chunked network coding. In Proceedings of the 2010 IEEE Information Theory Workshop on Information Theory (ITW '10), Cairo, Egypt, 6–8 January 2010, pp. 1–5.
17. Li, Y.; Soljanin, E.; Spasojevic, P. Effects of the Generation Size and Overlap on Throughput and Complexity in Randomized Linear Network Coding. *IEEE Trans. Inform. Theory* **2011**, *57*, 1111–1123. [CrossRef]
18. Mahdaviani, K.; Ardakani, M.; Bagheri, H.; Tellambura, C. Gamma Codes: A low-overhead linear-complexity network coding solution. In Proceedings of the 2012 International Symposium on Network Coding (NetCod), Cambridge, MA, USA, 29–30 June 2012; pp. 125–130. [CrossRef]
19. Yang, S.; Yeung, R.W. Batched Sparse Codes. *IEEE Trans. Inform. Theory* **2014**, *60*, 5322–5346. [CrossRef]
20. Tang, B.; Yang, S.; Yin, Y.; Ye, B.; Lu, S. Expander Chunked Codes. *EURASIP J. Adv. Signal Process.* **2015**, *2015*, 106. [CrossRef]
21. Tang, B.; Yang, S. An LDPC Approach for Chunked Network Codes. *IEEE/ACM Trans. Netw.* **2018**, *26*, 605–617. [CrossRef]
22. Yang, S.; Yeung, R.W. Network Communication Protocol Design from the Perspective of Batched Network Coding. *IEEE Commun. Mag.* **2022**, *60*, 89–93. [CrossRef]
23. Yang, S.; Yeung, R.W. *BATS Codes: Theory and Practice*; Synthesis Lectures on Communication Networks; Morgan & Claypool Publishers: Williston, VT, USA, 2017. [CrossRef]
24. MacWilliams, F.; Sloane, N. *The Theory of Error-Correcting Codes*; North-Holland Publishing: Amsterdam, The Netherlands, 1978.
25. Versfeld, D.J.; Ridley, J.N.; Ferreira, H.C.; Helberg, A.S. On systematic generator matrices for Reed–Solomon codes. *IEEE Trans. Inform. Theory* **2010**, *56*, 2549–2550. [CrossRef]
26. Luby, M.; Shokrollahi, A.; Watson, M.; Stockhammer, T.; Minder, L. *RaptorQ Forward Error Correction Scheme for Object Delivery–RFC 6330*; Internet Engineering Task Force: Fremont, CA, USA, 2011.
27. Arikan, E. Systematic polar coding. *IEEE Commun. Lett.* **2011**, *15*, 860–862. [CrossRef]
28. Badr, A.; Khisti, A.; Tan, W.T.; Apostolopoulos, J. Perfecting Protection for Interactive Multimedia: A survey of forward error correction for low-delay interactive applications. *IEEE Signal Process. Mag.* **2017**, *34*, 95–113. [CrossRef]
29. Garcia-Saavedra, A.; Karzand, M.; Leith, D.J. Low Delay Random Linear Coding and Scheduling Over Multiple Interfaces. *IEEE Trans. Mob. Comput.* **2017**, *16*, 3100–3114. [CrossRef]
30. Li, Y.; Zhang, F.; Wang, J.; Quek, T.Q.S.; Wang, J. On Streaming Coding for Low-Latency Packet Transmissions Over Highly Lossy Links. *IEEE Commun. Lett.* **2020**, *24*, 1885–1889. [CrossRef]
31. Prior, R.; Rodrigues, A. Systematic network coding for packet loss concealment in broadcast distribution. In Proceedings of the The International Conference on Information Networking 2011 (ICOIN2011), Kuala Lumpur, Malaysia, 26–28 January 2011; pp. 245–250. [CrossRef]
32. Lucani, D.E.; Medard, M.; Stojanovic, M. On Coding for Delay—Network Coding for Time-Division Duplexing. *IEEE Trans. Inform. Theory* **2012**, *58*, 2330–2348. [CrossRef]
33. Yu, M.; Aboutorab, N.; Sadeghi, P. From Instantly Decodable to Random Linear Network Coded Broadcast. *IEEE Trans. Commun.* **2014**, *62*, 3943–3955. [CrossRef]
34. Gabriel, F.; Wunderlich, S.; Pandi, S.; Fitzek, F.H.P.; Reisslein, M. Caterpillar RLNC With Feedback (CRLNC-FB): Reducing Delay in Selective Repeat ARQ Through Coding. *IEEE Access* **2018**, *6*, 44787–44802. [CrossRef]

35. Phung, C.V.; Engelmann, A.; Jukan, A. Error Correction with Systematic RLNC in Multi-Channel THz Communication Systems. In Proceedings of the 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), Opatija, Croatia, 28 September–2 October 2020; pp. 512–517. [CrossRef]
36. Karetsi, F.; Papapetrou, E. A Low Complexity Network-Coded ARQ protocol for Ultra-Reliable Low Latency Communication. In Proceedings of the 2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), Pisa, Italy, 7–11 June 2021; pp. 11–20. [CrossRef]
37. Shokrollahi, A.; Luby, M. Raptor Codes. *Found. Trends Commun. Inf. Theory* **2011**, *6*, 213–322. [CrossRef]
38. Tasdemir, E.; Tömösközi, M.; Cabrera, J.A.; Gabriel, F.; You, D.; Fitzek, F.H.P.; Reisslein, M. SpaRec: Sparse Systematic RLNC Recoding in Multi-Hop Networks. *IEEE Access* **2021**, *9*, 168567–168586. [CrossRef]
39. Pandi, S.; Gabriel, F.; Cabrera, J.A.; Wunderlich, S.; Reisslein, M.; Fitzek, F.H.P. PACE: Redundancy Engineering in RLNC for Low-Latency Communication. *IEEE Access* **2017**, *5*, 20477–20493. [CrossRef]
40. Wunderlich, S.; Gabriel, F.; Pandi, S.; Fitzek, F.H.P.; Reisslein, M. Caterpillar RLNC (CRLNC): A Practical Finite Sliding Window RLNC Approach. *IEEE Access* **2017**, *5*, 20183–20197. [CrossRef]
41. Lucani, D.E.; Pedersen, M.V.; Ruano, D.; Sørensen, C.W.; Fitzek, F.H.P.; Heide, J.; Geil, O.; Nguyen, V.; Reisslein, M. Fulcrum: Flexible Network Coding for Heterogeneous Devices. *IEEE Access* **2018**, *6*, 77890–77910. [CrossRef]
42. Nguyen, V.; Tasdemir, E.; Nguyen, G.T.; Lucani, D.E.; Fitzek, F.H.P.; Reisslein, M. DSEP Fulcrum: Dynamic Sparsity and Expansion Packets for Fulcrum Network Coding. *IEEE Access* **2020**, *8*, 78293–78314. [CrossRef]
43. Tasdemir, E.; Nguyen, V.; Nguyen, G.T.; Fitzek, F.H.P.; Reisslein, M. FSW: Fulcrum sliding window coding for low-latency communication. *IEEE Access* **2022**, *10*, 54276–54290. [CrossRef]
44. Compta, P.T.; Fitzek, F.H.P.; Lucani, D.E. Network Coding is the 5G Key Enabling Technology: Effects and Strategies to Manage Heterogeneous Packet Lengths. *Trans. Emerg. Telecommun. Technol.* **2015**, *6*, 46–55. [CrossRef]
45. Yang, S. Simbats. 2015. Available online: https://github.com/shhyang/simbats (accessed on 10 June 2022).