entropy

Special Issue Reprint

Information Theory and Network Coding II

Edited by
Shenghao Yang and Kenneth Shum

mdpi.com/journal/entropy

MDPI

Information Theory and Network Coding II

Information Theory and Network Coding II

Guest Editors

Shenghao Yang
Kenneth Shum

MDPI

Basel • Beijing • Wuhan • Barcelona • Belgrade • Novi Sad • Cluj • Manchester

Guest Editors
Shenghao Yang
School of Science and Engineering
The Chinese University of Hong Kong, Shenzhen
Shenzhen
China

Kenneth Shum
School of Science and Engineering
The Chinese University of Hong Kong, Shenzhen
Shenzhen
China

Editorial Office
MDPI AG
Grosspeteranlage 5
4052 Basel, Switzerland

This is a reprint of the Special Issue, published open access by the journal *Entropy* (ISSN 1099-4300), freely accessible at: https://www.mdpi.com/journal/entropy/special_issues/inf_cod.

For citation purposes, cite each article independently as indicated on the article page online and as indicated below:

Lastname, A.A.; Lastname, B.B. Article Title. *Journal Name* **Year**, *Volume Number*, Page Range.

ISBN 978-3-7258-3877-6 (Hbk)
ISBN 978-3-7258-3878-3 (PDF)
https://doi.org/10.3390/books978-3-7258-3878-3

Contents

entropy

MDPI

Editorial

Updates on Information Theory and Network Coding

Shenghao Yang * and Kenneth W. Shum

School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen 518172, China; wkshum@cuhk.edu.cn

* Correspondence: shyang@cuhk.edu.cn

Around the year 2000, network coding introduced the concept that coding can replace the basic packet forwarding operation used in traditional network communication systems [1–3]. This innovation simplified the achievement of maximum communication rates for unicast and improved the maximum communication rates for multicast. Random linear network coding (RLNC) bridged the gap between theory and practical applications by demonstrating that randomly selected linear combination coefficients are highly likely to be effective for linear network coding under some conditions [4,5]. Over the past two decades, significant research has focused on efficient RLNC. One notable approach is batched network coding (BNC), which integrates erasure coding with network coding in an outer-code inner-code manner (e.g., [6,7]).

Recently, network coding techniques have been discussed in 3GPP for 5G and within the Internet Research Task Force (IRTF), the research counterpart of the Internet Engineering Task Force (IETF). The Coding for Efficient Network Communication Research Group (NWCRG) of IRTF published six RFCs from 2018 to 2023. Network coding applications have been explored for satellite systems [8] and content-centric networking [9]. Two network coding protocols are described in [10,11], while the relationship between coding and congestion control is examined in [12].

As Guest Editors of this Special Issue of *Entropy*, titled *"Information Theory and Network Coding II"*, we are delighted to present ten cutting-edge research papers that explore advancements in coding theory and network coding. These papers cover a diverse array of subjects, such as finite-blocklength analysis (contribution 10), erasure coding (contribution 1), coded caching (contribution 8), vector-linear network coding (contribution 9), and secure network coding (contribution 6). Contribution 2 proposes a scalable RLNC scheme that adapts to the computational power of devices, while contribution 7 focuses on the application of RLNC in Vehicle-to-Vehicle (V2V) networks. Contributions 5 and 4 discuss the systematic design and adaptive recoding of BNC, respectively. Contribution 3 applies RLNC and BNC to distributed computation over lossy wireless networks.

We have witnessed rapid advancement in Large Language Models (LLMs) in recent years. This trend has provided numerous application scenarios for network coding and highlighted several promising directions for future research. Notably, training LLMs necessitates innovative GPU cluster networking technologies and distributed computation techniques. We eagerly anticipate the emergence of more groundbreaking research in network coding in the coming years.

We would like to express our sincere gratitude to all the authors who have contributed their high-quality work to this Special Issue. Their dedication and effort have made this collection of papers a reality. We also thank the reviewers for their time and expertise in evaluating the submitted manuscripts. Their constructive feedback has significantly enhanced the quality of the accepted papers. Furthermore, we are grateful to the Editorial

Received: 20 December 2024
Accepted: 26 December 2024
Published: 30 December 2024

Citation: Yang, S.; Shum, K.W. Updates on Information Theory and Network Coding. *Entropy* **2025**, *27*, 17. https://doi.org/10.3390/e27010017

Office of *Entropy* for their invaluable support and assistance in bringing this Special Issue to fruition.

Author Contributions: Writing—original draft preparation, S.Y.; writing—review and editing, K.W.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grants 12141108, 62171399 and 62171400.

Conflicts of Interest: The authors declare no conflicts of interest.

List of Contributions

1. Zhou, W.; Hou, H. Three Efficient All-Erasure Decoding Methods for Blaum–Roth Codes. *Entropy* **2022**, *24*, 1499. https://doi.org/10.3390/e24101499.
2. Tang, H.; Zheng, R.; Li, Z.; Long, K.; Sun, Q. Scalable Network Coding for Heterogeneous Devices over Embedded Fields. *Entropy* **2022**, *24*, 1510. https://doi.org/10.3390/e24111510.
3. Fan, B.; Tang, B.; Qu, Z.; Ye, B. Network Coding Approaches for Distributed Computation over Lossy Wireless Networks. *Entropy* **2023**, *25*, 428. https://doi.org/10.3390/e25030428.
4. Yin, H.H.F.; Yang, S.; Zhou, Q.; Yung, L.M.L.; Ng, K.H. BAR: Blockwise Adaptive Recoding for Batched Network Coding. *Entropy* **2023**, *25*, 1054. https://doi.org/10.3390/e25071054.
5. Mao, L.; Yang, S.; Huang, X.; Dong, Y. Design and Analysis of Systematic Batched Network Codes. *Entropy* **2023**, *25*, 1055. https://doi.org/10.3390/e25071055.
6. Bai, Y.; Guang, X.; Yeung, R.W. Multiple Linear-Combination Security Network Coding. *Entropy* **2023**, *25*, 1135. https://doi.org/10.3390/e25081135.
7. Zhang, Y.; Zhu, T.; Li, C. Efficient Communications in V2V Networks with Two-Way Lanes Based on Random Linear Network Coding. *Entropy* **2023**, *25*, 1454. https://doi.org/10.3390/e25101454.
8. Wang, W.; Tao, Z.; Liu, N.; Kang, W. Fundamental Limits of Coded Caching in Request-Robust D2D Communication Networks. *Entropy* **2024**, *26*, 250. https://doi.org/10.3390/e26030250.
9. Tang, H.; Liu, H.; Jin, S.; Liu, W.; Sun, Q. On Matrix Representation of Extension Field GF(pL) and Its Application in Vector Linear Network Coding. *Entropy* **2024**, *26*, 822. https://doi.org/10.3390/e26100822.
10. Jiang, M.; Wang, Y.; Ding, F.; Xu, Q. Finite-Blocklength Analysis of Coded Modulation with Retransmission. *Entropy* **2024**, *26*, 863. https://doi.org/10.3390/e26100863.

References

1. Koetter, R.; Medard, M. An Algebraic Approach to Network Coding. *IEEE/ACM Trans. Netw.* **2003**, *11*, 782–795. [CrossRef]
2. Li, S.Y.R.; Yeung, R.W.; Cai, N. Linear network coding. *IEEE Trans. Inform. Theory* **2003**, *49*, 371–381. [CrossRef]
3. Ahlswede, R.; Cai, N.; Li, S.Y.R.; Yeung, R.W. Network information flow. *IEEE Trans. Inform. Theory* **2000**, *46*, 1204–1216. [CrossRef]
4. 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]
5. Lun, D.S.; Médard, M.; Koetter, R.; Effros, M. On coding for reliable communication over packet networks. *Phys. Commun.* **2008**, *1*, 3–20. [CrossRef]
6. 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]
7. Yang, S.; Yeung, R.W. Batched Sparse Codes. *IEEE Trans. Inform. Theory* **2014**, *60*, 5322–5346. [CrossRef]
8. Kuhn, N.; Lochin, E. Network Coding for Satellite Systems. RFC 8975, 2021. Available online: https://doi.org/10.17487/RFC8975 (accessed on 19 December 2024).
9. Matsuzono, K.; Asaeda, H.; Westphal, C. Network Coding for Content-Centric Networking/Named Data Networking: Considerations and Challenges. RFC 9273, 2022. Available online: https://doi.org/10.17487/RFC9273 (accessed on 19 December 2024).
10. Yang, S.; Huang, X.; Yeung, R.W.; Zao, D.J.K. BATched Sparse (BATS) Coding Scheme for Multi-hop Data Transport. RFC 9426, 2023. Available online: https://doi.org/10.17487/RFC9426 (accessed on 19 December 2024).

11. Detchart, J.; Lochin, E.; Lacan, J.; Roca, V. Tetrys: An On-the-Fly Network Coding Protocol. RFC 9407, 2023. Available online: https://doi.org/10.17487/RFC9407 (accessed on 19 December 2024).
12. Kuhn, N.; Lochin, E.; Michel, F.; Welzl, M. Forward Erasure Correction (FEC) Coding and Congestion Control in Transport. RFC 9265, 2022. Available online: https://doi.org/10.17487/RFC9265 (accessed on 19 December 2024).

entropy

MDPI

Article

Three Efficient All-Erasure Decoding Methods for Blaum–Roth Codes

Weijie Zhou [1,†] and Hanxu Hou [2,*,†]

[1] School of Computer Science and Technology, Dongguan University of Technology, Dongguan 523820, China
[2] School of Electrical Engineering and Intelligentization, Dongguan University of Technology, Dongguan 523820, China
* Correspondence: houhanxu@163.com
† These authors contributed equally to this work.

Abstract: Blaum–Roth Codes are binary maximum distance separable (MDS) array codes over the binary quotient ring $\mathbb{F}_2[x]/(M_p(x))$, where $M_p(x) = 1 + x + \cdots + x^{p-1}$, and p is a prime number. Two existing all-erasure decoding methods for Blaum–Roth codes are the syndrome-based decoding method and the interpolation-based decoding method. In this paper, we propose a modified syndrome-based decoding method and a modified interpolation-based decoding method that have lower decoding complexity than the syndrome-based decoding method and the interpolation-based decoding method, respectively. Moreover, we present a fast decoding method for Blaum–Roth codes based on the LU decomposition of the Vandermonde matrix that has a lower decoding complexity than the two modified decoding methods for most of the parameters.

Keywords: distributed storage; Blaum–Roth codes; all-erasure decoding; decoding complexity

Citation: Zhou, W.; Hou, H. Three Efficient All-Erasure Decoding Methods for Blaum–Roth Codes. *Entropy* **2022**, *24*, 1499. https://doi.org/10.3390/e24101499

Academic Editors: Shenghao Yang and Kenneth Shum

Received: 5 September 2022
Accepted: 17 October 2022
Published: 20 October 2022

1. Introduction

Redundancy is necessary in storage systems in order to provide high data reliability in case of disk failures [1]. Replication and erasure codes are two main ways of including redundancy. The idea of replication is that the data in one disk are copied to multiple disks. The storage system replaces damaged disks with their copies when some disks are erased. It is fast to repair the erased disks but requires a lot of storage space. In contrast, erasure codes provide higher data reliability with a small storage cost.

Maximum distance separable (MDS) codes [2] are typical erasure codes that have optimal tradeoff between storage cost and data reliability, i.e., they can achieve the minimum storage cost given a level of data reliability. Binary MDS codes are special MDS codes that have lower computational complexity in the encoding/decoding procedures, since only XORs and cyclic-shift operations are involved. Some existing constructions of binary MDS codes are EVENODD codes [3,4], RDP codes [5], and X-codes [6,7], which can correct any two-column (we use "column" and "disk" interchangeably in this paper) erasures. RTP codes [8], Star codes [9,10], and extended EVENODD codes [11–14] can correct any three-column erasures. With the rapid increase in the data scale in storage systems [15], we need to design binary MDS codes that can correct any number of erasures as well as efficient encoding/decoding methods. Graftage codes [16] can achieve various tradeoffs between storage and repair bandwidth, while we focus on efficient decoding methods of binary MDS codes. Blaum–Roth codes [17] are this type of code, which are designed over the ring $\mathcal{R}_p = \mathbb{F}_2[x]/(M_p(x))$, where $M_p(x) = 1 + x + \cdots + x^{p-1}$, and p is a prime number.

When some columns are erased, the syndrome-based decoding method [17] and the interpolation-based decoding method [18] have been proposed to recover the erased columns. In the decoding methods [17,18], there are three basic operations over the ring $\mathcal{R}_p$: (i) addition, (ii) multiplication of a power of x and a polynomial, and (iii) division of factor $1 + x^b$ with $1 \leq b \leq p - 1$. It is shown in the decoding methods [17,18] that we can first take

the operations (i) and (ii) modulo $1+x^p$ and then take the results of modulo $M_p(x)$, while operation (iii) in the decoding methods [17,18] is directly taken as modulo $M_p(x)$.

In this paper, we show that we can also compute operation (iii) as modulo $1+x^p$, which has lower computational complexity than modulo $M_p(x)$. We propose modified decoding methods for the two existing decoding methods [17,18] that have a lower decoding complexity than the original decoding methods by computing operation (iii) as modulo $1+x^p$ instead of modulo $M_p(x)$. The reason our modified decoding methods have much lower decoding complexity than the decoding methods [17,18] is twofold. First, all the operations in our decoding methods are taken as modulo $1+x^p$, while the existing decoding methods execute the divisions as modulo $M_p(x)$. Second, we propose new algorithms in the decoding procedure to reduce the number of operations. Please refer to Section 3 for our two modified decoding methods. Moreover, the efficient LU decoding method [19] proposed for extended EVENODD codes decoding can also be employed to recover the erased columns of Blaum–Roth codes. We show that the LU decoding method has lower decoding complexity than the two modified decoding methods for most of the parameters. We define the decoding complexity as the total number of XORs required to recover the erased columns.

2. Blaum–Roth Codes

In this section, we first review the construction of Blaum–Roth codes [17] and then show the efficient operations over the ring $\mathbb{F}_2[x]/(1+x^p)$. Finally, we present an algorithm to compute multiple multiplications, which have two nonzero terms over $\mathbb{F}_2[x]/(1+x^p)$ with lower complexity.

2.1. Construction of Blaum–Roth Codes [17]

The codeword of Blaum–Roth codes [17] is a $(p-1)\times n$ array $[c_{i,j}]_{i=0,j=0}^{p-2,n-1}$ that is encoded from the $(p-1)k$ information bits, where $c_{i,j}\in\mathbb{F}_2$ and $n\le p$. We can view any k columns of the $(p-1)\times n$ array as information columns that store the $(p-1)k$ information bits and the other $r=n-k$ columns as parity columns that store the $(p-1)r$ parity bits. For $j=0,1,\ldots,n-1$, we represent the $p-1$ bits in column j by a polynomial $c_j(x)=\sum_{i=0}^{p-2}c_{i,j}x^i$. The $(p-1)\times n$ array of Blaum–Roth codes is defined as

$$\begin{pmatrix} c_0(x) & c_1(x) & \cdots & c_{n-1}(x)\end{pmatrix}\cdot \mathbf{H}_{r\times n}^T \equiv \mathbf{0} \pmod{M_p(x)},$$

where $\mathbf{H}_{r\times n}$ is the $r\times n$ parity-check matrix

$$\mathbf{H}_{r\times n}=\begin{bmatrix} 1 & 1 & 1 & \cdots & 1\\ 1 & x & x^2 & \cdots & x^{n-1}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & x^{(r-1)} & x^{(r-1)2} & \cdots & x^{(r-1)(n-1)}\end{bmatrix},$$

and $\mathbf{0}$ is the all-zero row of length r. We denote the Blaum–Roth codes defined above as $\mathcal{C}(p,n,r)$. When $p\ge n$ and p is a prime number, we can always retrieve all the information bits from any k out of the n polynomials [17], i.e., $\mathcal{C}(p,n,r)$ are MDS codes.

If we let $c_{p-1,j}=0$ for all $j=0,1,\ldots,n-1$, then $\mathcal{C}(p,n,r)$ can be equivalently defined as the following $p\cdot r$ linear constraints. (The subscripts are taken as modulo p unless otherwise specified.)

$$\sum_{j=0}^{n-1} c_{\langle m-\ell\cdot j\rangle_p,j}=0,$$

where $0\le m\le p-1$ and $0\le \ell\le r-1$.

Suppose that the λ columns $\{e_i\}_{i=0}^{\lambda-1}$ are erased, where $\lambda\ge 2$ and $0\le e_0<\cdots<e_{\lambda-1}<n$. Let the $\delta=n-\lambda$ surviving columns be $\{h_j\}_{j=0}^{\delta-1}$, where $0\le h_0<\cdots<h_{\delta-1}<n$ and $\{e_i\}_{i=0}^{\lambda-1}\cup\{h_j\}_{j=0}^{\delta-1}=\{0,1,\ldots,n-1\}$. We have

$$\begin{pmatrix} c_{e_0}(x) & c_{e_1}(x) & \cdots & c_{e_{\lambda-1}}(x) \end{pmatrix} \cdot \mathbf{V}_{\lambda\times\lambda}^T = \mathbf{S}, \tag{1}$$

over the ring $\mathcal{R}_p$, where $\mathbf{V}_{\lambda\times\lambda}$ is the $\lambda \times \lambda$ square

$$\mathbf{V}_{\lambda\times\lambda} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ x^{e_0} & x^{e_1} & x^{e_2} & \cdots & x^{e_{\lambda-1}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x^{(\lambda-1)e_0} & x^{(\lambda-1)e_1} & x^{(\lambda-1)e_2} & \cdots & x^{(\lambda-1)e_{\lambda-1}} \end{bmatrix},$$

and $\mathbf{S} = \begin{pmatrix} S_0(x) & S_1(x) & \cdots & S_{\lambda-1}(x) \end{pmatrix}$, where the λ syndrome polynomials are

$$S_\ell(x) = \sum_{j=0}^{\delta-1} x^{\ell\cdot h_j} c_{h_j}(x) \text{ for } 0 \le \ell \le \lambda - 1. \tag{2}$$

In this paper, we present three efficient decoding methods to solve the linear systems in Equation (1) over the ring $\mathbb{F}_2[x]/(1+x^p)$.

2.2. Efficient Operations over $\mathbb{F}_2[x]/(1+x^p)$

It is more efficient to compute the multiplication of a power of x and division of the factor $1+x^b$ over the ring $\mathbb{F}_2[x]/(1+x^p)$ rather than over the ring $\mathcal{R}_p$: (i) Let $a(x) \in \mathcal{R}_p$, and the multiplication $x^i \cdot a(x)$ over the ring $\mathcal{R}_p$ in [17] (Equation (19)) takes $p-1$ XORs, while the multiplication $x^i \cdot a(x)$ over the ring $\mathbb{F}_2[x]/(1+x^p)$ takes no XORs [20]. (ii) Let $g(x), f(x) \in \mathbb{F}_2[x]/(1+x^p)$, where d is a positive integer, which is coprime with p. Consider the equation

$$(1+x^d)g(x) \equiv f(x) \pmod{1+x^p}, \tag{3}$$

where $f(x)$ has an even number of nonzero terms. Given such $f(x)$ and d, we can compute $g(x)$ by Lemma 1.

Lemma 1. *[Lemma 8] in [21] The coefficients of* $g(x)$ *in Equation* (3) *are given by*

$$g_{p-1} = 0,\ g_{p-d-1} = f_{p-1},\ g_{d-1} = f_{d-1},$$
$$g_{p-(\ell+1)d-1} = g_{p-\ell d-1} + f_{p-\ell d-1} \text{ for } \ell = 1, 2, \ldots, p-3.$$

By Lemma 1, computing the division $\frac{f(x)}{1+x^d}$ takes $p-3$ XORs, but we are not sure whether $g(x)$ has an even number of nonzero terms or not. If we want to guarantee that $g(x)$ has an even number of nonzero terms, we should use Lemma 2 to compute the division $\frac{f(x)}{1+x^d}$.

Lemma 2. *[Lemma 13] in [20] The coefficients of* $g(x)$ *in Equation* (3) *are given by*

$$g_0 = f_{2d} + f_{4d} + \cdots + f_{(p-1)d},$$
$$g_{\ell d} = g_{(\ell-1)d} + f_{\ell d} \text{ for } \ell = 1, 2, \ldots, p-1.$$

By Lemma 2, the division $\frac{f(x)}{1+x^d}$ takes $\frac{3p-5}{2}$ XORs, and $g(x)$ has an even number of nonzero terms. However, computing the division $\frac{f(x)}{1+x^d}$ in [Corollary 2] in [17] takes $2(p-1)$ XORs over the ring $\mathcal{R}_p$, which is strictly larger than the decoding methods in Lemmas 1 and 2. It is shown in [Theorem 5] in [19] that we can always solve the equations in Equation (1) over the ring $\mathbb{F}_2[x]/(1+x^p)$ of which all the solutions are congruent to each other after modulo $M_p(x)$. Therefore, we can first solve the equations in Equation (1) over the ring $\mathbb{F}_2[x]/(1+x^p)$ and then obtain the unique solution by taking modulo $M_p(x)$ to reduce the computational complexity.

2.3. Multiple Multiplications over $\mathbb{F}_2[x]/(1+x^p)$

Note that in our modified syndrome-based decoding method and the modified interpolation-based decoding method, we need to compute multiple polynomial multiplications, where each polynomial has two nonzero terms. Suppose that we want to compute the following m multiplications

$$\mathcal{L}(x^\tau) = \prod_{i=0}^{m-1}(x^\tau - x^{\xi_i}) \pmod{1+x^p}, \tag{4}$$

where m is a positive integer, $0 \le \tau \le p-1$ such that $\tau \notin \{\xi_0, \xi_1, \ldots, \xi_{m-1}\}$, and $0 \le \xi_0 < \cdots < \xi_{m-1} < n$.

We can derive from Equation (4) that

$$\mathcal{L}(x^\tau) = x^\pi \cdot \prod_{i=0}^{m-1}(1+x^{d_i}) \pmod{1+x^p}, \tag{5}$$

where $\pi = \sum_{i=0}^{m-1}\min(\tau, \xi_i)$ modulo p and $d_i = |\tau - \xi_i|$ for $i = 0, 1, \ldots, m-1$.

Algorithm 1 presents a method to simplify the multiplications in Equation (4). In Algorithm 1, we use Γ_ℓ to denote the number of the polynomial $1+x^\ell$ in the multiplication $\mathcal{L}(x^\tau)$. Note that we only need to count the number of $1+x^\ell$ for $1 \le \ell \le \frac{p-1}{2}$, because the equation $1+x^\ell \equiv x^\ell \cdot (1+x^{p-\ell})$ modulo $1+x^p$ holds for $\frac{p-1}{2} < \ell < n$. If $\Gamma_\ell > 1$, then we have $(1+x^\ell)^{\Gamma_\ell} = (1+x^\ell)^{\Gamma_\ell - 2\lfloor\frac{\Gamma_\ell}{2}\rfloor} \cdot (1+x^{2\ell})^{\lfloor\frac{\Gamma_\ell}{2}\rfloor}$. Therefore, we can always merge Γ_ℓ multiplications $(1+x^\ell)^{\Gamma_\ell}$ into $\Gamma_\ell - \lfloor\frac{\Gamma_\ell}{2}\rfloor$ multiplications and the computational complexity can be reduced with Algorithm 1. When Algorithm 1 is executed, all elements of count-array Γ should be zero or one, and the length η of the final $\mathcal{L}(x^\tau)$ is between 1 and m.

Algorithm 1: Simplify the multiple multiplications.

Data: $\mathcal{L}(x^\tau) = \prod_{i=0}^{m-1}(x^\tau - x^{\xi_i})$
$\pi \leftarrow \sum_{i=0}^{m-1}\min(\tau, \xi_i) \pmod{p}$;
count-array $\Gamma[\frac{p-1}{2}] = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_{\frac{p-1}{2}}\} \leftarrow \{\mathbf{0}\}$;
for $i \leftarrow 0$ **to** $m-1$ **do** // `Hash`.
  $d_i \leftarrow |\tau - \xi_i| \bmod p$;
  if $d_i \le \frac{p-1}{2}$ **then** $\Gamma_{d_i} \leftarrow \Gamma_{d_i} + 1$;
  else // `Use` $1 + x^{d_i} \equiv x^{d_i} \cdot (1 + x^{p-d_i})$
    $\pi \leftarrow (\pi + d_i) \bmod p$;
    $\Gamma_{p-d_i} \leftarrow \Gamma_{p-d_i} + 1$;
$\omega \leftarrow 0$;
while $\omega \ne \frac{p-1}{2}$ **do**
  for $\ell \leftarrow 1$ **to** $\frac{p-1}{2}$ **do** // `Merge Multiplications` $(1+x^\ell)^{\Gamma_\ell}$
    if $\Gamma_\ell \le 1$ **then Continue**;
    if $2\ell \le \frac{p-1}{2}$ **then** $\Gamma_{2\ell} \leftarrow \Gamma_{2\ell} + \lfloor\frac{\Gamma_\ell}{2}\rfloor$;
    else // `Use` $1 + x^{2\ell} \equiv x^{2\ell} \cdot (1 + x^{p-2\ell})$
      $\pi \leftarrow (\pi + 2\lfloor\frac{\Gamma_\ell}{2}\rfloor \ell) \bmod p$;
      $\Gamma_{p-2\ell} \leftarrow \Gamma_{p-2\ell} + \lfloor\frac{\Gamma_\ell}{2}\rfloor$;
    $\Gamma_\ell \leftarrow \Gamma_\ell - 2\lfloor\frac{\Gamma_\ell}{2}\rfloor$;
  $\omega \leftarrow$ the amount of elements no greater than one in count-array Γ;
$\{\xi_i\}_{i=0}^{\eta-1} \leftarrow$ the subscript of one in count-array Γ, i.e., $\Gamma_{\xi_i} = 1$;
return $x^\pi \cdot \prod_{i=0}^{\eta-1}(1+x^{\xi_i})$;

3. Decoding Algorithm

In this section, we present two decoding methods over the ring $\mathbb{F}_2[x]/(1+x^p)$ by modifying two existing decoding methods [17,18] that can reduce the decoding complexity.

Recall that the λ erased columns are λ columns $\{e_i\}_{i=0}^{\lambda-1}$, and the $\delta = n - \lambda$ surviving columns are δ columns $\{h_j\}_{j=0}^{\delta-1}$.

3.1. Modified Syndrome-Based Method

We define the function of the indeterminate z

$$G_i(z) = \prod_{s=0,\neq i}^{\lambda-1} (1 - x^{e_s} z) = \sum_{\ell=0}^{\lambda-1} G_{i,\ell}(x) z^\ell,$$

and the syndrome function $S(z) = \sum_{\ell=0}^{r-1} S_\ell(x) z^\ell$, where $0 \le i \le \lambda - 1$ and $S_\ell(x)$ is given in Equation (2). We can obtain in [Equation (18)] in [17] that

$$\begin{aligned} \prod_{s=0,\neq i}^{\lambda-1} (x^{e_i} - x^{e_s}) c_{e_i}(x) &\equiv \sum_{\ell=0}^{\lambda-1} G_{i,\lambda-1-\ell}(x) S_\ell(x) \\ &\equiv \sigma_i(x) \pmod{M_p(x)}. \end{aligned}$$

Therefore, the $\sigma_i(x)$ can be regarded as the coefficient of $z^{\lambda-1}$ of the polynomial $G_i(z)S(z)$. Then, the erased column $c_{e_i}(x)$ is given by $\frac{\sigma_i(x)}{\prod_{s=0,\neq i}^{\lambda-1}(x^{e_i} - x^{e_s})}$, where $0 \le i \le \lambda - 1$.

Note that the terms of set $\{S_\ell(x) z^\ell\}_{\ell=\lambda}^{r-1}$ are not involved in computing the coefficient of $z^{\lambda-1}$ of the polynomial $G_i(z)S(z)$. Thus, we can just consider the first λ terms (the λ coefficients of degrees less than λ) of $S(z)$ when computing these coefficients, but all the r terms of $S(z)$ are calculated in [Step 1] in [17]. This is one essential way our modified syndrome-based decoding method obtains a lower decoding complexity than the original method in [17].

Moreover, the syndrome polynomials $S_\ell(x)$ satisfy

$$S_0(1) = S_1(1) = \cdots = S_{\lambda-1}(1), \tag{6}$$

i.e., the λ syndrome polynomials $S_\ell(x)$ either all have an even number of nonzero terms, or they all have an odd number of nonzero terms, from the definition of Equation (2).

Let $G(z) = (1 - x^{e_i} z) G_i(z)$ and $Q(z) = G(z)S(z)$. Then, we have

$$\begin{aligned} Q(z) &= (1 - x^{e_i} z) \prod_{s=0,\neq i}^{\lambda-1} (1 - x^{e_s} z) S(z) \\ &= \prod_{s=0}^{\lambda-1} (1 - x^{e_s} z) S(z) = \sum_{\ell=0}^{r+\lambda-1} Q_\ell(x) z^\ell. \end{aligned} \tag{7}$$

Thus, $Q(z)$ is independent of the erasure index i, and we only need to compute $Q(z)$ once in the decoding procedure. Recall that $\sigma_i(x)$ is the coefficient of $z^{\lambda-1}$ of the polynomial $G_i(z)S(z)$; then, the $\sigma_i(x)$ is also the coefficient of $z^{\lambda-1}$ of the polynomial $\frac{Q(z)}{(1-x^{e_i}z)} = \frac{(1-x^{e_i}z)G_i(z)S(z)}{(1-x^{e_i}z)}$ for all $0 \le i \le \lambda - 1$. Suppose that

$$\frac{Q(z)}{(1 - x^{e_i} z)} = f_0^i(x) + f_1^i(x) z + \cdots + f_{\lambda-1}^i(x) z^{\lambda-1} + \cdots,$$

we can derive the recurrence formula

$$f_\ell^i(x) = \begin{cases} Q_0(x), & \ell = 0; \\ x^{e_i} \cdot f_{\ell-1}^i(x) + Q_\ell(x), & \ell > 0; \end{cases} \tag{8}$$

where $0 \le i \le \lambda - 1$. Notice that $\sigma_i(x) = f^i_{\lambda-1}(x)$ holds. Similar to $S(z)$, we only compute the first λ terms (the λ coefficients of degrees less than λ) of $Q(z)$, since the other coefficients of $Q(z)$ are not needed, but all the $r + \lambda$ terms of $Q(z)$ are calculated in [Step 2] in [17]. This is another way our modified syndrome-based decoding method obtains a lower decoding complexity than the original method in [17]. Algorithm 2 shows our modified syndrome-based decoding method over the ring $\mathbb{F}_2[x]/(1 + x^p)$.

The following Lemma shows that we can always compute the divisions in steps 11–12 of Algorithm 2 by Lemmas 1 and 2 when $\lambda \ge 2$.

Lemma 3. *In steps 11–12 of Algorithm 2, the $\sigma_i(x)$ has an even number of nonzero terms for all $0 \le i \le \lambda - 1$, and we can employ Lemmas 1 and 2 to compute the divisions.*

Proof. From Equation (8) and steps 7–10 of Algorithm 2, we obtain

$$\sigma_i(x) = x^{(\lambda-1)e_i} Q_0(x) + x^{(\lambda-2)e_i} Q_1(x) + \cdots + Q_{\lambda-1}(x),$$

where $0 \le i \le \lambda - 1$. If the number of polynomials in the set $\{Q_j(x)\}_{j=0}^{\lambda-1}$, which has an odd number of nonzero terms, is an even number, then the $\sigma_i(x)$ has an even number of nonzero terms for $0 \le i \le \lambda - 1$. In the following, we will show this is true. According to Equation (6) and step 3 of Algorithm 2, $Q_0(1) = \cdots = Q_{\lambda-1}(1)$ holds.

Firstly, we consider $Q_0(1) = \cdots = Q_{\lambda-1}(1) = 1$. We denote the λ polynomials $\{Q_j(x)\}_{j=0}^{\lambda-1}$ with $\varepsilon = 0, 1, \ldots, \lambda$ as $\{Q^\varepsilon_j(x)\}_{j=0}^{\lambda-1}$. Let $Q^0_j(x)$ be the initial $Q_j(x)$ for $0 \le j \le \lambda - 1$.

To prove that the number of polynomials with an odd number of nonzero terms in the set $\{Q^\varepsilon_j(x)\}_{j=0}^{\lambda-1}$ is even, it is equivalent to prove that $\sum_{j=0}^{\lambda-1} Q^\varepsilon_j(1) = 0$.

Algorithm 2: Modified syndrome-based decoding method.

Input: The λ erased columns $\{e_i\}_{i=0}^{\lambda-1}$ and the $\delta = n - \lambda$ surviving columns $\{h_j\}_{j=0}^{\delta-1}$.

1 **for** $\ell \leftarrow 0$ **to** $\lambda - 1$ **do** `// Use Equation (2) and subscript ℓ means slope.`
2 $\quad S_\ell(x) \leftarrow \sum_{j=0}^{\delta-1} x^{\ell \cdot h_j} c_{h_j}(x)$;
3 $Q(z) = \sum_{\ell=0}^{\lambda-1} Q_\ell(x) z^\ell \leftarrow S(z) = \sum_{\ell=0}^{\lambda-1} S_\ell(x) z^\ell$;
4 **for** $s \leftarrow 0$ **to** $\lambda - 1$ **do** `// Use Equation (7).`
5 $\quad$ **for** $\ell \leftarrow \lambda - 1$ **down to** 1 **do** `// Calculate` $Q_\ell(x)$ `by backward additions.`
6 $\quad\quad Q_\ell(x) \leftarrow x^{e_s} \cdot Q_{\ell-1}(x) + Q_\ell(x)$;
7 **for** $i \leftarrow 0$ **to** $\lambda - 1$ **do** `// Use Equation (8).`
8 $\quad \sigma_i(x) \leftarrow Q_0(x)$;
9 $\quad$ **for** $\ell \leftarrow 1$ **to** $\lambda - 1$ **do**
10 $\quad\quad \sigma_i(x) \leftarrow x^{e_i} \cdot \sigma_i(x) + Q_\ell(x)$;
11 **for** $i \leftarrow 0$ **to** $\lambda - 1$ **do** `// Apply Algorithm 1.`
12 $\quad c_{e_i}(x) \leftarrow \frac{\sigma_i(x)}{\prod_{s=0,\neq i}^{\lambda-1}(x^{e_i} - x^{e_s})}$;

Output: The erased columns $\{c_{e_i}(x)\}_{i=0}^{\lambda-1}$.

According to Equation (7) and steps 4–6 of Algorithm 2, we have

$$Q^\varepsilon_j(1) = \begin{cases} Q^{\varepsilon-1}_j(1), & j = 0; \\ Q^{\varepsilon-1}_{j-1}(1) + Q^{\varepsilon-1}_j(1), & 1 \le j \le \lambda - 1; \end{cases} \tag{9}$$

where $\varepsilon = 1, 2, \ldots, \lambda$. The $Q_j^1(1) = 0$ holds for all $j \geq 1$. We can obtain by induction

$$Q_j^\varepsilon(1) = Q_{j-1}^{\varepsilon-1}(1) + Q_j^{\varepsilon-1}(1) = 0 \text{ for all } j \geq \varepsilon \geq 1. \tag{10}$$

Note that $\sum_{j=0}^{\lambda-1} Q_j^2(1) = 0$; we can suppose that there are an even number of polynomials in the set $\{Q_j^\varepsilon(x)\}_{j=0}^{\lambda-1}$, which has an odd number of nonzero terms, when $\varepsilon = y \geq 2$, i.e., $\sum_{j=0}^{\lambda-1} Q_j^y(1) = 0$ first. We have $\sum_{j=0}^{\lambda-1} Q_j^{y+1}(1)$; so,

$$\begin{aligned}\sum_{j=0}^{\lambda-1} Q_j^{y+1}(1) &= Q_0^y(1) + \sum_{j=1}^{\lambda-1} \left(Q_{j-1}^y(1) + Q_j^y(1)\right) \\ &= \sum_{j=0}^{\lambda-1} Q_j^y(1) + \sum_{j=0}^{\lambda-2} Q_j^y(1) \\ &= Q_{\lambda-1}^y(1) = 0.\end{aligned} \tag{11}$$

Equation (11) comes from Equation (10) with $j = \lambda - 1$. Therefore, there are an even number of polynomials in the set $\{Q_j^{y+1}(x)\}_{j=0}^{\lambda-1}$, which has an odd number of nonzero terms.

Secondly, when $Q_0(1) = \cdots = Q_{\lambda-1}(1) = 0$, the argument is similar. This completes the proof. □

According to Lemma 3, we can use Lemmas 1 and 2 to compute the divisions in step 12. The number of divisions required in step 12 is recorded as L_i, which ranges from 1 to $\lambda - 1$ for $i = 0, 1, \ldots, \lambda - 1$. So, we can obtain $c_{e_i}(x)$ in step 12 by recursively computing the division L_i times, while the number of nonzero terms of the polynomial resulting from the first $L_i - 1$ divisions is even. Therefore, we can execute these divisions by Lemma 2 and execute the last division by Lemma 1. The computational complexity T_D in steps 11–12 of Algorithm 2 is

$$T_D = \sum_{i=0}^{\lambda-1} \left((L_i - 1)\frac{3p-5}{2} + p - 3\right), \tag{12}$$

where $\lambda(p-3) \leq T_D \leq \lambda(\lambda-2)\frac{3p-5}{2} + \lambda(p-3)$.

In steps 11-12 of Algorithm 2, we take the $\lambda(\lambda-1)$ division without Algorithm 1, in which λ divisions are executed by Lemma 1 and $\lambda(\lambda-2)$ divisions are executed by Lemma 2; however, the number of the divisions can be reduced with Algorithm 1. In Table 1, we show the average number of divisions in steps 11–12 of Algorithm 2 executed by Lemma 1 and Lemma 2 with Algorithm 1 for $(p, n) \in \{(5, 5), (7, 7)\}$.

Table 1. The average number of XORs involved in steps 11–12 of Algorithm 2.

p, n	λ	Without Algorithm 1			Apply Algorithm 1			Improvement(%)
		Lemma 2	Lemma 1	XORs	Lemma 2	Lemma 1	XORs	
(5,5)	2	0	2	4	0	2	4	0%
	3	3	3	21	2	3	16	23.81%
	4	8	4	48	0	4	8	83.33%
(7,7)	2	0	2	8	0	2	8	0%
	3	3	3	36	2.4	3	31.2	13.33%
	4	8	4	80	4.4	4	51.2	36%
	5	15	5	140	1	5	28	80%
	6	24	6	216	6	6	72	66.67%

We specify the computational complexity of Algorithm 2 as follows:

- Steps 1–2 take $\lambda(\delta-1)p = \lambda(n-\lambda-1)p$ XORs.

- Steps 3–6 take $\lambda(\lambda-1)p$ XORs.
- Steps 7–10 take $\lambda(\lambda-1)p$ XORs.
- Steps 11–12 take T_D XORs by Equation (12).

Then, the computational complexity $T_{Alg\ 2}$ of Algorithm 2 is

$$T_{Alg\ 2} = \lambda(n+\lambda-3)p + T_D, \tag{13}$$

where

$$p\lambda^2+\big((n-2)p-3\big)\lambda \le T_{Alg\ 2} \le \frac{5(p-1)}{2}\lambda^2+\big((n-5)p+2\big)\lambda.$$

Recall that the computational complexity of the decoding method in [17] is

$$\frac{7p-4}{2}\lambda^2 - \frac{7p-2}{2}\lambda + r(n-1)p.$$

which is strictly larger than $T_{Alg\ 2}$.

Table 2 evaluates the computational complexity of the decoding method in [17] and Algorithm 2 for some parameters. The results in Table 2 demonstrate that Algorithm 2 has much lower decoding complexity, compared with the original decoding method in [17]. For example, Algorithm 2 has 40.60% less decoding complexity than the decoding method in [17] when $(p,n,r)=(7,7,4), \lambda=3$.

Table 2. Decoding complexity of method in [17] and Algorithm 2.

p, n, r	λ	**XORs in [17]**	**XORs of $T_{Alg\ 2}$**	**Improvement(%)**
$(5,5,3)$	2	89	44	50.56%
	3	150	91	39.33%
$(7,7,4)$	2	211	92	56.40%
	3	300	178.2	40.60%
	4	434	275.2	36.59%

The reason why Algorithm 2 has lower decoding complexity than the decoding method in [17] can be summarized as the following three points.

Firstly, we only consider the first λ terms (the λ coefficients of degrees less than λ) for both $S(z)$ and $Q(z)$ in computing the coefficients of $z^{\lambda-1}$, while all r terms of $S(z)$ and all $r+\lambda$ terms of $Q(z)$ are calculated in the decoding method in [17], where $r \ge \lambda$.

Secondly, all the divisions in Algorithm 2 are executed over the ring $\mathbb{F}_2[x]/(1+x^p)$ by Lemmas 1 and 2, which takes $p-3$ XORs and $\frac{3p-5}{2}$ XORs for each division, respectively. In addition, the division in [17] is executed over the ring $\mathcal{R}_p$, which takes $2(p-1)$ XORs [17] (Corollary 2).

Thirdly, we apply Algorithm 1 to steps 11–12 of Algorithm 2, which can significantly reduce the number of divisions, thus reducing the number of XORs required.

3.2. Modified Interpolation-Based Decoding Method

According to the decoding method in [18], we can recover the erased column $c_{e_i}(x)$ with $0 \le i \le \lambda-1$ by

$$c_{e_i}(x) = \sum_{j=0}^{\delta-1} c_{h_j}(x)\frac{f_i(x^{h_j})}{f_i(x^{e_i})} \pmod{M_p(x)}, \tag{14}$$

where $f_i(y) = \prod_{s=0,\neq i}^{\lambda-1}(y-x^{e_s})$ and $f(y) = \prod_{s=0}^{\lambda-1}(y-x^{e_s})$. Let

$$a_j(x) = c_{h_j}(x)\cdot f(x^{h_j}) = \prod_{s=0}^{\lambda-1}(x^{h_j}-x^{e_s})\cdot c_{h_j}(x) \pmod{M_p(x)}, \tag{15}$$

where $0 \le j \le \delta - 1$. Then, $a_j(x)$ has an even number of nonzero terms, and we only need to compute once for $a_j(x)$ in the decoding procedure, since $a_j(x)$ is independent of the erasure index i. Let

$$b_i(x) = \sum_{j=0}^{\delta-1} \frac{a_j(x)}{x^{h_j} - x^{e_i}} \pmod{M_p(x)}, \tag{16}$$

$$c_{e_i}(x) = \frac{b_i(x)}{f_i(x^{e_i})} = \frac{b_i(x)}{\prod_{s=0,\neq i}^{\lambda-1}(x^{e_i} - x^{e_s})} \pmod{M_p(x)}, \tag{17}$$

where $0 \le i \le \lambda - 1$, and $M_p(x) = 1 + x + \cdots + x^{p-1}$. Algorithm 3 shows our modified interpolation-based method over the ring $\mathbb{F}_2[x]/(1 + x^p)$.

After using Algorithm 1, the number of polynomial multiplications in step 2 ranges from 1 to λ. Thus, the computational complexity T_M in steps 1–2 of Algorithm 3 is

$$(n - \lambda)p \le T_M \le (n - \lambda)\lambda p. \tag{18}$$

Algorithm 3: Modified interpolation-based method.

Input: The λ erased columns $\{e_i\}_{i=0}^{\lambda-1}$ and the $\delta = n - \lambda$ surviving columns $\{h_j\}_{j=0}^{\delta-1}$.

1 **for** $j \leftarrow 0$ **to** $\delta - 1$ **do** `// Use Equation (15) and apply Algorithm 1`

2 $\quad a_j(x) \leftarrow f(x^{h_j}) \cdot c_{h_j}(x) = \prod_{s=0}^{\lambda-1}(x^{h_j} - x^{e_s}) \cdot c_{h_j}(x)$;

3 **for** $i \leftarrow 0$ **to** $\lambda - 1$ **do** `// Use Equation (16)`

4 $\quad b_i(x) \leftarrow \sum_{j=0}^{\delta-1} \frac{a_j(x)}{x^{h_j} - x^{e_i}}$;

5 **for** $i \leftarrow 0$ **to** $\lambda - 1$ **do** `// Use Equation (17) and apply Algorithm 1`

6 $\quad c_{e_i}(x) \leftarrow \frac{b_i(x)}{f_i(x^{e_i})} = \frac{b_i(x)}{\prod_{s=0,\neq i}^{\lambda-1}(x^{e_i} - x^{e_s})}$;

Output: The erased columns $\{c_{e_i}(x)\}_{i=0}^{\lambda-1}$.

In steps 1–2, we need to take λ multiplications without Algorithm 1, which takes $(n - \lambda)\lambda p$ XORs; however, with Algorithm 1, the number of multiplications involved in steps 1–2 can be reduced. In Table 3, we show the average number of XORs involved in steps 1–2 of Algorithm 3 with Algorithm 1 for $(p, n) \in \{(5, 5), (7, 7)\}$. The results in Table 3 show that we can reduce the number of XORs with Algorithm 1, especially for a large value of λ.

Table 3. The average number of XORs involved in steps 1–2 of Algorithm 3.

p, n	λ	Without Algorithm 1		Apply Algorithm 1		Improvement(%)
		Multiplication	XORs	Multiplication	XORs	
(5,5)	2	6	30	5	25	16.67%
	3	6	30	2	10	66.67%
	4	4	20	2	10	50%
(7,7)	2	10	70	9	63	10%
	3	12	84	8.4	58.8	30%
	4	12	84	3.6	25.2	70%
	5	10	70	4	28	60%
	6	6	42	3	21	50%

Only steps 4 and 6 of Algorithm 3 are needed to compute the division. We should employ Lemma 2 to execute the divisions in steps 3–4 in Algorithm 3, since $b_i(x)$ in step 6 of Algorithm 3 should have an even number of nonzero terms. Notice that steps 5–6 of Algorithm 3 are exactly the same as steps 11–12 of Algorithm 2.

We specify the computational complexity of Algorithm 3 as follows:

- Steps 1–2 require T_M XORs by Equation (18).
- Steps 3–4 need $\lambda(\delta-1)$ additions and $\lambda\delta$ divisions by Lemma 2, which require $\lambda(n-\lambda-1)p+\lambda(n-\lambda)\frac{3p-5}{2}$ XORs in total.
- Steps 5–6 require T_D XORs by Equation (12).

Then, the computational complexity of Algorithm 3 is

$$T_{Alg\ 3} = T_M + \lambda(n-\lambda-1)p + \lambda(n-\lambda)\frac{3p-5}{2} + T_D, \tag{19}$$

where

$$-\frac{5(p-1)}{2}\lambda^2 + (\frac{5n-2}{2}p - \frac{5}{2}n - 3)\lambda + np \le T_{Alg\ 3} \le -2p\lambda^2 + (\frac{7n-6}{2}p - \frac{5}{2}n + 2)\lambda.$$

Recall that the computational complexity of the decoding method in [18] is

$$(-2p+1)\lambda^2 + \big(4(n-1)p - 3n + 4\big)\lambda + n(p-1),$$

which is larger than that of our Algorithm 3.

Table 4 evaluates the computational complexity of the decoding method in [18] and Algorithm 3 for some parameters. The results in Table 4 demonstrate that our Algorithm 3 had much lower decoding complexity, compared with the original decoding method in [18]. For example, Algorithm 3 had a 34.13% lower decoding complexity than the decoding method in [18], when $(p,n,r) = (7,7,4), \lambda = 3$.

Table 4. Decoding complexities of the decoding method in [18] and our Algorithm 3.

$\mathbf{p,n,r}$	λ	XORs in [18]	XORs of $T_{Alg\ 3}$	Improvement(%)
$(5,5,3)$	2	122	79	35.25%
	3	146	71	51.37%
$(7,7,4)$	2	292	207	29.11%
	3	378	249	34.13%
	4	438	228.4	47.85%

The reason why Algorithm 3 has a lower decoding complexity than that of the decoding method in [18] is summarized as follows.

Firstly, all the divisions in Algorithm 3 were executed over the ring $\mathbb{F}_2[x]/(1+x^p)$ by Lemmas 1 and 2, which used $p-3$ XORs and $(3p-5)/2$ XORs for each division, respectively. The division in the decoding method in [18] was executed over the ring $\mathcal{R}_p$, which used $2(p-1)$ XORs.

Secondly, we applied our Algorithm 1 to steps 1–2 and steps 5–6, which significantly reduced the number of multiplications, thus reducing the number of XORs required.

4. LU Decomposition-Based Method

The LU factorization of a matrix [22] is to express the matrix as a product of a lower triangular matrix **L** and an upper triangular matrix **U**. According to the LU factorization of the Vandermonde matrix [23], we can express a Vandermonde matrix as a product of several lower triangular matrices and several upper triangular matrices. Therefore, we can solve the Vandermonde linear equations by first solving the linear equations with the encoding matrices that are the upper triangular matrices and then solving the linear equations with the encoding matrices that are the lower triangular matrices.

Suppose that the λ erased columns are λ columns $\{e_i\}_{i=0}^{\lambda-1}$ and the $\delta = n-\lambda$ surviving columns are $\{h_j\}_{j=0}^{\delta-1}$. Algorithm 4 shows our LU decomposition-based method over the ring $\mathbb{F}_2[x]/(1+x^p)$.

According to [Theorem 8] in [19], Equation (1) can be factorized into

$$\begin{pmatrix} c_{e_0}(x) & c_{e_1}(x) & \cdots & c_{e_{\lambda-1}}(x) \end{pmatrix} \cdot (\mathbf{L}_\lambda^{(1)} \mathbf{L}_\lambda^{(2)} \cdots \mathbf{L}_\lambda^{(\lambda-1)}) \cdot (\mathbf{U}_\lambda^{(\lambda-1)} \mathbf{U}_\lambda^{(\lambda-2)} \cdots \mathbf{U}_\lambda^{(1)}) = \mathbf{S}, \quad (20)$$

over the ring $\mathcal{R}_p$, where $\mathbf{U}_\lambda^{(\theta)}$ is the upper triangle matrix

$$\mathbf{U}_\lambda^{(\theta)} = \left[\begin{array}{c|cccccc} \mathbf{I}_{\lambda-\theta-1} & \multicolumn{6}{c}{\mathbf{0}} \\ \hline & 1 & x^{e_0} & 0 & \cdots & 0 & 0 \\ & 0 & 1 & x^{e_1} & \cdots & 0 & 0 \\ ine\mathbf{0} & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ & 0 & 0 & 0 & \cdots & 1 & x^{e_{\theta-1}} \\ & 0 & 0 & 0 & \cdots & 0 & 1 \end{array}\right], \quad (21)$$

and $\mathbf{L}_\lambda^{(\theta)}$ is the lower triangle matrix

$$\mathbf{L}_\lambda^{(\theta)} = \left[\begin{array}{c|cccc} \mathbf{I}_{\lambda-\theta-1} & \multicolumn{4}{c}{\mathbf{0}} \\ \hline 1 & 0 & \cdots & 0 & 0 \\ 1 & x^{e_{\lambda-\theta}} + x^{e_{\lambda-\theta-1}} & \cdots & 0 & 0 \\ ine\mathbf{0} & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & x^{e_{\lambda-2}} + x^{e_{\lambda-\theta-1}} & 0 \\ 0 & 0 & \cdots & 1 & x^{e_{\lambda-1}} + x^{e_{\lambda-\theta-1}} \end{array}\right], \quad (22)$$

for $\theta = 1, 2, \ldots, \lambda - 1$.

Algorithm 4: LU decomposition-based method.

Input: The λ erased columns $\{e_i\}_{i=0}^{\lambda-1}$ and the $\delta = n - \lambda$ surviving columns $\{h_j\}_{j=0}^{\delta-1}$.

1 **for** $\ell \leftarrow 0$ **to** $\lambda - 1$ **do** // use Equation (2) and the subscript ℓ means slope.
2 $\quad S_\ell(x) \leftarrow \sum_{j=0}^{\delta-1} x^{\ell \cdot h_j} c_{h_j}(x)$;
3 $\begin{pmatrix} c_{e_0}(x) & c_{e_1}(x) & \cdots & c_{e_{\lambda-1}}(x) \end{pmatrix} \leftarrow \begin{pmatrix} S_0(x) & S_1(x) & \cdots & S_{\lambda-1}(x) \end{pmatrix}$;
// Eliminate $\lambda - 1$ upper triangular matrices $\mathbf{U}_\lambda^{(1)}, \mathbf{U}_\lambda^{(2)}, \ldots, \mathbf{U}_\lambda^{(\lambda-1)}$.
4 **for** $\theta \leftarrow 1$ **to** $\lambda - 1$ **do**
$\quad$ // Eliminate upper triangular matrix $\mathbf{U}_\lambda^{(\theta)}$ by forward additions.
5 $\quad$ **for** $i \leftarrow \lambda - \theta$ **to** $\lambda - 1$ **do**
6 $\quad\quad c_{e_i}(x) \leftarrow c_{e_i}(x) + x^{e_{\theta+i-\lambda}} \cdot c_{e_{i-1}}(x)$;
// Eliminate $\lambda - 1$ lower triangular matrix $\mathbf{L}_\lambda^{(\lambda-1)}, \mathbf{L}_\lambda^{(\lambda-2)}, \ldots, \mathbf{L}_\lambda^{(1)}$.
7 **for** $\theta \leftarrow \lambda - 1$ **down to** 1 **do**
$\quad$ // Eliminate lower triangular matrix $\mathbf{L}_\lambda^{(\theta)}$ by backward additions.
8 $\quad$ Solve $c_{e_{\lambda-1}}(x)$ from $c_{e_{\lambda-1}}(x) = \frac{c_{e_{\lambda-1}}(x)}{x^{e_{\lambda-1}} + x^{e_{\lambda-\theta-1}}}$ by Lemma 2 and Lemma 1(only when $\theta = 1$);
9 $\quad$ **for** $i \leftarrow \lambda - 2$ **down to** $\lambda - \theta$ **do**
10 $\quad\quad$ Solve $c_{e_i}(x)$ from $c_{e_i}(x) = \frac{c_{e_i}(x) - c_{e_{i+1}}(x)}{x^{e_i} + x^{e_{\lambda-\theta-1}}}$ by Lemma 2 and Lemma 1(only when $i = \lambda - \theta$);
11 $\quad c_{e_{\lambda-\theta-1}}(x) \leftarrow c_{e_{\lambda-\theta-1}}(x) - c_{e_{\lambda-\theta}}(x)$;

Output: The erased columns $\{c_{e_i}(x)\}_{i=0}^{\lambda-1}$.

We specify the computational complexity of Algorithm 4 as follows:

- Steps 1–2 require $\lambda(\delta - 1)p = \lambda(n - \lambda - 1)p$ XORs.

- Steps 3–11 require $\lambda(\lambda-1)p+(\lambda-1)(p-3)+(\lambda-1)(\lambda-2)(3p-5)/4$ XORs at most, according to [Theorem 10] in [19].

Then, the computational complexity of Algorithm 4 is

$$T_{Alg\ 4}=\frac{3p-5}{4}\lambda^2+\frac{(4n-13)p+3}{4}\lambda+\frac{p+1}{2}. \tag{23}$$

5. Comparison and Conclusions

Table 5 evaluates the decoding complexity of Algorithm 2–4 for some parameters. The results of Table 5 demonstrate that Algorithm 2 performs better than Algorithm 3 if $\lambda \leq \frac{n}{2}$; otherwise, if $\lambda > \frac{n}{2}$, then Algorithm 3 has less decoding complexity. Algorithm 4 has less decoding complexity than both Algorithms 2 and 3, when λ is small. However, when λ is large, Algorithm 3 is more efficient than Algorithm 4. For example, compared with Algorithm 2–4 have 21.98% and 40.66% less decoding complexity, respectively, when $(p,n,r)=(5,5,4),\lambda=3$.

Table 5. Decoding complexities of the proposed three decoding methods.

$\mathbf{p,n,r}$	λ	total XORs			$\frac{T_{Alg\ 2}-T_{Alg\ 3}}{T_{Alg\ 2}}$	$\frac{T_{Alg\ 2}-T_{Alg\ 4}}{T_{Alg\ 2}}$
		$T_{Alg\ 2}$	$T_{Alg\ 3}$	$T_{Alg\ 4}$		
$(5,5,4)$	2	44	79	32	−79.55%	27.27%
	3	91	71	54	21.98%	40.66%
	4	128	38	81	70.31%	36.72%
$(7,7,6)$	2	92	207	74	−125%	19.57%
	3	178.2	249	121	−39.73%	32.10%
	4	275.2	228.4	176	17.01%	36.05%
	5	343	171	239	50.15%	30.32%
	6	492	141	310	71.34%	36.99%

In this paper, we presented three efficient decoding methods for the erasures of Blaum–Roth codes that all have lower decoding complexity than the existing decoding methods. The efficient implementation of the proposed decoding methods in practical storage systems is one of our future works.

Author Contributions: Funding acquisition, H.H. methodology, H.H.; writing—original draft preparation, W.Z.; writing—review and editing, H.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China grant number 62071121 and National Key R&D Program of China grant number 2020YFA0712300.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: All data generated or analysed during this study are included in this published article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Peng, P. Redundancy Allocation in Distributed Systems. Ph.D. Thesis, Rutgers The State University of New Jersey, School of Graduate Studies, New Brunswick, NJ, USA, 2022.
2. MacWilliams, F.J.; Sloane, N.J.A. *The Theory of Error Correcting Codes*; Elsevier: Amsterdam, The Netherlands, 1977; Volume 16.
3. Blaum, M.; Brady, J.; Bruck, J.; Menon, J. EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures. *IEEE Trans. Comput.* **1995**, *44*, 192–202. [CrossRef]
4. Hou, H.; Lee, P.P.C. A New Construction of EVENODD Codes With Lower Computational Complexity. *IEEE Commun. Lett.* **2018**, *22*, 1120–1123. [CrossRef]

5. Corbett, P.; English, B.; Goel, A.; Grcanac, T.; Kleiman, S.; Leong, J.; Sankar, S. Row-diagonal Parity for Double Disk Failure Correction. In Proceedings of the 3rd USENIX Conference on File and Storage Technologies, San Francisco, CA, USA, 31 March–4 April 2004; pp. 1–14.
6. Xu, L.; Bruck, J. X-code: MDS Array Codes with Optimal Encoding. *IEEE Trans. Inf. Theory* **1999**, *45*, 272–276.
7. Tsunoda, Y.; Fujiwara, Y.; Ando, H.; Vandendriessche, P. Bounds on separating redundancy of linear codes and rates of X-codes. *IEEE Trans. Inf. Theory* **2018**, *64*, 7577–7593. [CrossRef]
8. Goel, A.; Corbett, P. RAID Triple Parity. *ACM SIGOPS Oper. Syst. Rev.* **2012**, *46*, 41–49. [CrossRef]
9. Huang, C.; Xu, L. STAR: An Efficient Coding Scheme for Correcting Triple Storage Node Failures. *IEEE Trans. Comput.* **2008**, *57*, 889–901. [CrossRef]
10. Hou, H.; Lee, P.P.C. STAR+ Codes: Triple-Fault-Tolerant Codes with Asymptotically Optimal Updates and Efficient Encoding/Decoding. In Proceedings of the 2021 IEEE Information Theory Workshop (ITW 2021), Kanazawa, Japan, 17–21 October 2021.
11. Blaum, M.; Brady, J.; Bruck, J.; Jai Menon, J.; Vardy, A. The EVENODD Code and its Generalization: An Effcient Scheme for Tolerating Multiple Disk Failures in RAID Architectures. In *High Performance Mass Storage and Parallel I/O*; Wiley-IEEE Press: Hoboken, NJ, USA, 2002; Chapter 8, pp. 187–208.
12. Blaum, M.; Bruck, J.; Vardy, A. MDS Array Codes With Independent Parity Symbols. *IEEE Trans. Inf. Theory* **1996**, *42*, 529–542. [CrossRef]
13. Hou, H.; Shum, K.W.; Chen, M.; Li, H. New MDS Array Code Correcting Multiple Disk Failures. In Proceedings of the 2014 IEEE Global Communications Conference, Austin, TX, USA, 8–12 December 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 2369–2374.
14. Fu, H.; Hou, H.; Zhang, L. Extended EVENODD+ Codes with Asymptotically Optimal Updates and Efficient Encoding/Decoding. In Proceedings of the 2021 XVII International Symposium "Problems of Redundancy in Information and Control Systems" (REDUNDANCY), Moscow, Russia, 25–29 October 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6.
15. Chiniah, A.; Mungur, A. On the Adoption of Erasure Code for Cloud Storage by Major Distributed Storage Systems. *EAI Endorsed Trans. Cloud Syst.* **2022**, *7*, e1. [CrossRef]
16. Rui, J.; Huang, Q.; Wang, Z. Graftage Coding for Distributed Storage Systems. *IEEE Trans. Inf. Theory* **2021**, *67*, 2192–2205. [CrossRef]
17. Blaum, M.; Roth, R.M. New Array Codes for Multiple Phased Burst Correction. *IEEE Trans. Inf. Theory* **1993**, *39*, 66–77. [CrossRef]
18. Guo, Q.; Kan, H. On Systematic Encoding for Blaum-Roth Codes. In Proceedings of the 2011 IEEE International Symposium on Information Theory Proceedings, St. Petersburg, Russia, 31 July–5 August 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 2353–2357.
19. Hou, H.; Han, Y.S.; Shum, K.W.; Li, H. A Unified Form of EVENODD and RDP Codes and Their Efficient Decoding. *IEEE Trans. Commun.* **2018**, *66*, 5053–5066. [CrossRef]
20. Hou, H.; Shum, K.W.; Chen, M.; Li, H. BASIC Codes: Low-complexity Regenerating Codes for Distributed Storage Systems. *IEEE Trans. Inf. Theory* **2016**, *62*, 3053–3069. [CrossRef]
21. Hou, H.; Han, Y.S. A New Construction and An Efficient Decoding Method for Rabin-like Codes. *IEEE Trans. Commun.* **2017**, *66*, 521–533. [CrossRef]
22. Strang, G.; Strang, G.; Strang, G.; Strang, G. *Introduction to Linear Algebra*; Wellesley-Cambridge Press: Wellesley, MA, USA, 1993; Volume 3.
23. Yang, S.l. On the LU factorization of the Vandermonde matrix. *Discret. Appl. Math.* **2005**, *146*, 102–105. [CrossRef]

entropy

MDPI

Article

Scalable Network Coding for Heterogeneous Devices over Embedded Fields

Hanqi Tang [1], Ruobin Zheng [2], Zongpeng Li [3], Keping Long [1] and Qifu Sun [1,*]

[1] Department of Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China
[2] Network Technology Lab, Huawei Technologies Co., Ltd., Shenzhen 518000, China
[3] Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China
* Correspondence: qfsun@ustb.edu.cn

Abstract: In complex network environments, there always exist heterogeneous devices with different computational powers. In this work, we propose a novel scalable random linear network coding (RLNC) framework based on embedded fields, so as to endow heterogeneous receivers with different decoding capabilities. In this framework, the source linearly combines the original packets over embedded fields based on a precoding matrix and then encodes the precoded packets over GF(2) before transmission to the network. After justifying the arithmetic compatibility over different finite fields in the encoding process, we derive a sufficient and necessary condition for decodability over different fields. Moreover, we theoretically study the construction of an optimal precoding matrix in terms of decodability. The numerical analysis in classical wireless broadcast networks illustrates that the proposed scalable RLNC not only guarantees a better decoding compatibility over different fields compared with classical RLNC over a single field, but also outperforms Fulcrum RLNC in terms of a better decoding performance over GF(2). Moreover, we take the sparsity of the received binary coding vector into consideration, and demonstrate that for a large enough batch size, this sparsity does not affect the completion delay performance much in a wireless broadcast network.

Keywords: random linear network coding (RLNC); wireless broadcast network; scalable network coding

Citation: Tang, H.; Zheng, R.; Li, Z.; Long, K.; Sun, Q. Scalable Network Coding for Heterogeneous Devices over Embedded Fields. *Entropy* **2022**, *24*, 1510. https://doi.org/10.3390/e24111510

Academic Editors: Shenghao Yang and Kenneth Shum

Received: 29 September 2022
Accepted: 17 October 2022
Published: 22 October 2022

1. Introduction

In a communication network, linear network coding (LNC) advocates intermediate nodes to linearly combine received messages before transmission, so as to improve various network performances, such as increasing network throughput, reliability, and reducing transmission delay. Random linear network coding (RLNC) provides a distributed and asymptotically optimal approach for linear coding with coefficients randomly selected from a base field [1]. It shows the potential to improve the performance of unreliable or topologically unknown networks such as D2D networks [2], ad hoc networks [3], and wireless broadcast networks [4–7].

One of the reasons that hinder the large-scale practical applications of RLNC is the compatibility issue of different computational overheads. In complex network environments, there exist heterogeneous devices with different computational powers. Specifically, sources and certain receivers usually have ample computational powers while a large number of intermediate nodes and other receivers are computationally constrained such as the data collectors in ad hoc networks or low-cost devices in the Internet of Things paradigm [8]. It turns out that the coding compatibility among heterogeneous devices with different computational powers has to be considered in RLNC design.

This paper proposes a novel framework for scalable RLNC design based on embedded fields. The adjective *scalable* means that the finite fields chosen in the encoding process are not limited to a single base field but a set of *embedded fields* which consists of a large finite

field and all its subfields. The encoding process at the source consists of two stages. In stage 1, based on a precoding matrix, all original packets are linearly combined over different finite fields to form precoded packets. In stage 2, the final packets to be transmitted are formed by randomly combining the precoded packets over GF(2). The heterogeneous receivers can recover the original packets over different fields under different computational constraints.

It is worthwhile to remark that prior to this work, there have been studies [9–14] that have taken different fields into account in the course of RLNC design. On one hand, the so-called Telescopic codes [9–11] and Revolving codes [12] considered different fields aiming at reducing the decoding complexity. However, they assume that all receivers have the same decoding capability, that is, they need support the arithmetic over the largest defined finite field. On the other hand, a flexible RLNC scheme called Fulcrum [13,14] makes use of GF(2) and its extension field GF(2^8) for code design and it supports receivers to decode over both fields. Actually, Fulcrum can be regarded as a special instance in our proposed framework, while the decoding rule over GF(2) considered therein is weaker than the one proposed in this paper. In addition, there is limited discussion on the construction of an optimal encoding matrix in Fulcrum.

The main contributions of this paper are summarized as follows.

- We mathematically justify how to make the arithmetic over different finite fields compatible.
- We derive a necessary and sufficient condition for decodability at a receiver over different finite fields. In particular, the proposed decoding rule over GF(2) is stronger than the one proposed in Fulcrum.
- We theoretically study the construction of an optimal precoding matrix in terms of the decodability performance.
- By numerical analysis in classical wireless broadcast networks, we demonstrate that the proposed scalable RLNC not only guarantees a better decoding compatibility over different fields compared with classical RLNC over a single field, but also provides a better decoding performance over GF(2) in terms of smaller average completion delay compared with Fulcrum.
- In numerical analysis, we also take the sparsity of the received binary coding vector into consideration, and demonstrate that for a large enough batch size, this sparsity does not affect the completion delay performance much in a wireless broadcast network.

This paper is structured as follows. Section 2 reviews the mathematical fundamentals of embedded fields. Section 3 first presents the general principles of the proposed scalable RLNC framework and then formulates the encoding and decoding process. Section 4 investigates the design of an optimal precoding matrix. Section 5 numerically analyzes the proposed scalable RLNC and compares its performance with classical RLNC over a single finite field as well as Fulcrum. Moreover, we take the sparsity into consideration and illustrate the influence on its performance. Conclusion is given in Section 6.

2. Mathematical Fundamentals

In our proposed scalable RLNC framework, different receivers will be able to recover the original packets over different finite fields, upon their different computational powers. In order to make the arithmetic over different finite fields compatible, we need the concept of embedded fields, which will be briefly reviewed in this section. One may refer to [15] for a detailed introduction on finite fields.

Recall that a finite field GF(2^{d_1}) is a subfield of GF(2^{d_2}) if and only if $d_1|d_2$. Thus, GF(2^{d_1}), GF(2^{d_2}), ..., GF(2^{d_D}) are said to form embedded fields $\mathcal{F}$ if $d_1 < d_2 < \ldots < d_D$ and $d_1|d_2|\ldots|d_D$. For arbitrary GF(2^{d_i}) and GF(2^{d_j}) in $\mathcal{F}$ with $i < j$, as GF(2^{d_j}) can be regarded as GF$\left((2^{d_i})^{d_j/d_i}\right)$, it can be expressed not only as a d_{d_j}-dimensional vector space over GF(2), but also as a d_j/d_i-dimensional vector space over GF(2^{d_i}) at the same time.

Example 1. *Assume that* $d_1 = 1$, $d_2 = 2$, $d_3 = 4$. *The field GF*(2^4) *can be expressed as a four-dimensional vector space over GF*(2) *as well as a two-dimensional vector space over GF*(2^2). *Let* α *be a root of the irreducible polynomial* $x^2 + x + 1$ *over GF*(2) *so that GF*(2^2) $= \{0, 1, \alpha, \alpha^2\}$. *The polynomial* $g(x) = x^4 + x + 1$ *is irreducible over GF*(2) *but reducible over GF*(2^2) *and can be factorized as* $g(x) = (x^2 + x + \alpha)(x^2 + x + \alpha^2)$. *Let* β *be a root of the irreducible polynomial* $f(x) = x^2 + x + \alpha$ *over GF*(2^2) *and* β *a root of* $f(x)$, *so that* $g(\beta) = \beta^4 + \beta + 1 = 0$ *as well. Then, every element in* GF(2^4) *can be expressed as* $a_0 + a_1\beta + a_2\beta^2 + a_3\beta^3$ *with* $a_j \in \{0, 1\}$. *Moreover,* $\alpha = \beta^2 + \beta = \beta^5$, *so that* GF($2^2$) $= \{0, \beta^0, \beta^5, \beta^{10}\}$. *Based on this, every element in GF*(2^4) *can also be uniquely expressed as* $b_0 + b_1\beta$, $b_0, b_1 \in$ GF(2^2), *which is summarized in Figure 1. In Figure 1, the integers* 0 *to* 15 *are the decimal representation of the binary* 4*-tuple* (a_3, a_2, a_1, a_0), *e.g.,* 13 *refers to* $1 + \beta^2 + \beta^3$, *which can be expressed* $1 + \alpha\beta$.

b_1 \ b_0	0	1	α	α^2
0	0	1	6	7
1	2	3	4	5
α	12	13	10	11
α^2	14	15	8	9

Figure 1. Every element $a_0 + a_1\beta + a_2\beta^2 + a_3\beta^3, a_j \in \{0, 1\}$ in GF(2^4) has a unique expression in the form of $b_0 + b_1\beta$, $b_0, b_1 \in \{0, 1, \alpha, \alpha^2\} =$ GF(4), where $\alpha^2 + \alpha + 1 = \beta^2 + \beta + \alpha = \beta^4 + \beta + 1 = 0$. The integers 0 to 15 represent the decimal expression of the binary 4-tuple (a_3, a_2, a_1, a_0).

3. Framework Description

3.1. General Principles

In this paper, we focus on the construction of a general scalable RLNC framework over embedded fields, so we attempt to alleviate the influence of specific models of networks. In the course of framework description, we merely classify the nodes in a network into three types: a unique source node, intermediate nodes and receiver nodes. Assume that the source has the highest computational power, so that it can generate coded packets over embedded fields. The intermediate nodes in the network just recode the received data packets over GF(2), so as to fully reduce the overall computational complexities in the network. The heterogeneous receivers have different decoding capabilities. Under its own computational constraint, every receiver can judge whether sufficient coded packets have been received for decoding. More importantly, even though a receiver may not have sufficient computational power to deal with the arithmetic in a larger field over which some received packets are coded, it can still fully utilize these packets instead of directly throwing away in the process of decoding. For instance, assume that two received packets $\mathbf{w}_1$ and $\mathbf{w}_2$ are respectively equal to $\mathbf{p}_1 + \mathbf{p}_2 + \alpha\mathbf{p}_3$ and $\mathbf{p}_2 + \alpha\mathbf{p}_3$, where $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ are original packets generated by the source node and α is an element not equal to 0 and 1 in the field GF(2^{d_D}). For the receiver under the strongest field constraint GF(2), the original packet $\mathbf{p}_1$ can be recovered by $\mathbf{w}_1 + \mathbf{w}_2$ instead of directly throwing $\mathbf{w}_1$, $\mathbf{w}_2$ away. Consequently, the proposed scalable RLNC framework not only ensures the decoding capabilities of heterogeneous network devices but also fully reduces the required number of received packets for decoding.

3.2. Encoding and Recoding

In every batch, the source s has n original packets $\mathbf{p}_i, 1 \leq i \leq n$, each of which is an M-dimensional column vector over GF(2), to be transmitted to receivers. Without loss of generality, assume M is divisible by 2^{2^D}, which can be achieved by padding dummy bits into every packet. With increasing D, the double exponentially increasing packet length M may cause the practical issue of an excessive padding overhead. Such an issue can be effectively solved based on the methods proposed in [16,17].

The encoding process at s has two stages. First, based on $\mathbf{p}_i, 1 \leq i \leq n$, for each $1 \leq d \leq D$, extra r_d precoded packets are generated based on coding coefficients selected

from GF(2^{2^d}). In this process, every original packet $\mathbf{p}_i$ is regarded as a vector of $m_d = M/2^d$ symbols, each of which consists of 2^d bits and represents an element in GF(2^{2^d}). The multiplication of $\mathbf{p}_i$ by a coefficient in GF(2^{2^d}) is thus realized by symbol-wise multiplication. Note that when $d_1 < d_2$, the coefficients in GF($2^{2^{d_1}}$) also appear in GF($2^{2^{d_2}}$), but the coding arithmetic changes. The mathematical fundamentals in the previous section guarantee the coding compatibility which will be illustrated in the next example.

Example 2. *Assume $M = 4$, $n = 2$, $d_1 = 1$ and $d_2 = 2$. Based on two original packets $\mathbf{p}_1 = [1\,0\,0\,0]^{\mathrm{T}}$ and $\mathbf{p}_2 = [1\,1\,0\,1]^{\mathrm{T}}$, a precoded packet is to be generated over* GF(4) $= \{0, 1, \alpha, \alpha^2\}$ *by the linear combination $\alpha\mathbf{p}_1 + \alpha^2\mathbf{p}_2$. First regard $\mathbf{p}_1$ and $\mathbf{p}_2$ as vectors of* 2 *symbols over GF(2^2), that is, $\mathbf{p}_1 = \begin{bmatrix}\alpha\\0\end{bmatrix}$ and $\mathbf{p}_2 = \begin{bmatrix}\alpha+1\\1\end{bmatrix} = \begin{bmatrix}\alpha^2\\1\end{bmatrix}$. Then,*

$$\alpha\mathbf{p}_1 + \alpha^2\mathbf{p}_2 = \begin{bmatrix}\alpha^2\\0\end{bmatrix} + \begin{bmatrix}\alpha\\\alpha^2\end{bmatrix} = \begin{bmatrix}1\\\alpha+1\end{bmatrix} = [0\,1\,1\,1]^{\mathrm{T}} \tag{1}$$

According to Figure 1, in GF(2^4), $\alpha = \beta^2 + \beta = \beta^5$ and $\alpha^2 = \beta^2 + \beta + 1 = \beta^{10}$. As every element in GF(2^4) = GF(4^2) can be uniquely expressed as $b_0 + b_1\beta$, $b_0, b_1 \in$ GF(4)*, every four-dimensional vector $[a_3\ a_2\ a_1\ a_0]^{\mathrm{T}}$ over GF(*2*) as the following element in GF(*16*)*

$$[a_3\ a_2\ a_1\ a_0]^{\mathrm{T}} = a_3\beta^6 + a_2\beta + a_1\beta^5 + a_0.$$

*Based on this rule, $\mathbf{p}_1 = \beta^6$ and $\mathbf{p}_2 = \beta^6 + \beta + 1$. Consequently, $\beta^5\mathbf{p}_1 + \beta^{10}\mathbf{p}_2 = \beta + \beta^{10} = \beta + \beta^5 + 1$, which is $[1\ \alpha + 1]^{\mathrm{T}}$ over GF(*4*) and $[0\ 1\ 1\ 1]^{\mathrm{T}}$ over GF(*2*), same as* (1) *obtained by the GF(*4*) arithmetic.*

After stage 1, there are a total of $N = n + r_1 + r_2 + \ldots + r_D$ precoded packets, the first n of which are just the original packets. Let $\mathbf{G} = \begin{bmatrix}\mathbf{I}_n\ \mathbf{A}_1\ \ldots\ \mathbf{A}_D\end{bmatrix}$ denote the $n \times N$ precoding matrix for the N precoded packets, where $\mathbf{I}_n$ refers to the $n \times n$ identity matrix and $\mathbf{A}_d$ is a coefficient matrix defined over GF(2^{2^d}).

In stage 2, every coded packet $\mathbf{c}$ the source finally sends out is a random GF(2)-linear combination of the N precoded packets, that is,

$$\mathbf{c} = [\mathbf{p}_1\ \mathbf{p}_2\ \cdots\ \mathbf{p}_n]\mathbf{G}\mathbf{h},$$

for some randomly generated N-dimensional column vector $\mathbf{h}$ over GF(2), which is referred to as the *coding vector* for packet $\mathbf{c}$. For a *systematic* scheme, the first n coded packets $\mathbf{c}_1, \ldots, \mathbf{c}_n$ transmitted by the source are just n original packets, that is, the coding vector for $\mathbf{c}_j$ is just an N-dimensional unit vector with the j^{th} position equal to 1. Every coded packet will affix its coding vector to its header. In contrast, the information of precoding matrix $\mathbf{G}$ can either be affixed to the header of every packet or presettled to be known at every receiver.

At an intermediate node, the coded packets it transmits are GF(2)-linear combinations of its received packets. Specifically, if an intermediate node receives coded packets $\mathbf{c}_1, \ldots, \mathbf{c}_l$ with respective coding vectors $\mathbf{h}_1, \ldots, \mathbf{h}_l$, then it will recode them to generate a new coded packet $\mathbf{c}'$ to be transmitted as

$$\mathbf{c}' = a_1\mathbf{c}_1 + \ldots + a_l\mathbf{c}_l,$$

where $a_1, \ldots, a_l$ are random binary coefficients. The concomitant coding vector for $\mathbf{c}'$ is $a_1\mathbf{h}_1 + \ldots + a_l\mathbf{h}_l$.

It is worthwhile to note that prior to this work, a flexible RLNC scheme called Fulcrum has been investigated in [13,14]. Fulcrum can be regarded as a special instance in our proposed framework with the setting $D = 3$ and $r_1 = r_2 = 0$.

3.3. Decoding

Define a linear map $\varphi : \mathrm{GF}(2)^N \to \mathrm{GF}(2^{2^D})^n$ by

$$\varphi(\mathbf{v}) = \mathbf{G}\mathbf{v}.$$

for every column vector $\mathbf{v} \in \mathrm{GF}(2)^N$. The notation φ also applies to a set $\mathcal{V}$ of vectors: $\varphi(\mathcal{V}) = \{\varphi(\mathbf{v}) : \mathbf{v} \in \mathcal{V}\}$.

Moreover, let $\mathcal{U}_d$, $0 \le d \le D$, denote the vector subspace of $\mathrm{GF}(2)^N$ spanned by unit vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{\sum_{d'=0}^{d} r_{d'}}$ where a unit vector $\mathbf{u}_j$ refers to an N-dimensional vector with the only nonzero entry at position j.

For a receiver t, assume d_t is the largest field for computation, and m packets have been received. Let $\mathbf{H}$ denote the $N \times m$ matrix over GF(2) obtained by columnwise juxtaposition of the coding vectors of the m received packets, and $\mathcal{H}$ the column space (over GF(2)) of $\mathbf{H}$.

In order to recover original packets under the field constraint $\mathrm{GF}(2^{2^{d_t}})$, we need make use of coding packets with coding vectors in $\mathcal{U}_{d_t} \cap \mathcal{H}$ rather than in $\mathcal{H}$. This is because the lower $\sum_{d'>d_t} r_{d'}$ entries in every coding vector corresponds to the original precoded packets generated by the source over a larger field than $\mathrm{GF}(2^{2^{d_t}})$. We next characterize the following necessary and sufficient condition for decodability at t up to field constraint $\mathrm{GF}(2^{2^{d_t}})$.

Theorem 1. *Based on the m received packets, the original n source packets can be recovered at t if and only if*

$$\dim(\varphi(\mathcal{U}_{d_t} \cap \mathcal{H})) = n. \tag{2}$$

Proof. First assume (2) holds. Then, there must exist n vectors, denoted by $\mathbf{v}_1, \ldots, \mathbf{v}_n$ in $\mathcal{U}_{d_t} \cap \mathcal{H}$ such that

$$\dim(\varphi(\{\mathbf{v}_1, \ldots, \mathbf{v}_n\})) = n. \tag{3}$$

Consequently, there exists an $m \times n$ matrix $\mathbf{K}$ over GF(2) such that $[\mathbf{v}_1 \ \ldots \ \mathbf{v}_n] = \mathbf{H}\mathbf{K}$, and (3) implies the full rank n of $\mathbf{G}\mathbf{H}\mathbf{K}$. As the last $\sum_{d'>d_t} r_{d'}$ rows in $\mathbf{H}\mathbf{K}$ are all zero, the elements in $\mathbf{G}\mathbf{H}\mathbf{K}$ belong to $\mathrm{GF}(2^{2^{d_t}})$, and hence there exists an $n \times n$ matrix $\mathbf{D}$ over $\mathrm{GF}(2^{2^{d_t}})$ subject to $\mathbf{G}\mathbf{H}\mathbf{K}\mathbf{D} = \mathbf{I}_n$, that is, the original packets can be recovered at t.

Next assume that the original n packets can be recovered at t. Then, there exists an $m \times n$ matrix $\mathbf{D}$ over $\mathrm{GF}(2^{2^{d_t}})$ such that $\mathbf{G}\mathbf{H}\mathbf{D} = \mathbf{I}_n$. Further, $\mathbf{D}$ can be written as $\mathbf{D}_1\mathbf{D}_2$, where $\mathbf{D}_1$, $\mathbf{D}_2$ are over $\mathrm{GF}(2^{2^{d_t}})$ and of respective size $m \times n$ and $n \times n$. Thus, $\mathbf{G}\mathbf{H}\mathbf{D}_1$ is a matrix over $\mathrm{GF}(2^{2^{d_t}})$ of full rank n. Recall that none of the elements in the last $\sum_{d'>d_t} r_{d'}$ columns in $\mathbf{G}$ is in $\mathrm{GF}(2^{2^{d_t}})$. Thus, every element in $\mathbf{G}\mathbf{H}\mathbf{D}_1$ belonging to $\mathrm{GF}(2^{2^{d_t}})$ implies that the last $\sum_{d'>d_t} r_{d'}$ rows in $\mathbf{H}\mathbf{D}_1$ are all zero. Moreover, as $\mathbf{H}$ is defined over GF(2), we can further deduce that $\mathbf{D}_1$ can be written as $\mathbf{D}_1'\mathbf{D}_1''$ for an $m \times n$ matrix $\mathbf{D}_1'$ over GF(2) and an $n \times n$ matrix $\mathbf{D}_1''$ over $\mathrm{GF}(2^{2^{d_t}})$, such that the last $\sum_{d'>d_t} r_{d'}$ rows in $\mathbf{H}\mathbf{D}_1'$ are all zero too, that is, the columns in $\mathbf{H}\mathbf{D}_1'$ belong to $\mathcal{U}_{d_t} \cap \mathcal{H}$. In addition, the full rank of $\mathbf{G}\mathbf{H}\mathbf{D}_1$ implies the full rank of $\mathbf{G}\mathbf{H}\mathbf{D}_1'$. Equation (2) is thus proved to hold. □

Based on the above theorem, we can further characterize the following equivalent condition for decodability at a receiver from the perspective of matrix rank. For $0 \le d \le D$, denote by $\mathbf{H}_{d_t}$ the $\sum_{d'>d_t} r_{d'} \times m$ submatrix of $\mathbf{H}$ obtained by restricting $\mathbf{H}$ to the last $\sum_{d'>d_t} r_{d'}$ rows.

Corollary 1. *Based on the m received packets, the original n source packets can be recovered at t if and only if*

$$\mathrm{rank}(\mathbf{G}(\mathbf{H}\mathbf{K}_{d_t})) = n, \tag{4}$$

where $\mathbf{K}_{d_t}$ *is an* $m \times (m - \mathrm{rank}(\mathbf{H}_{d_t}))$ *matrix whose columns constitute a basis for the kernel of the column space of* $\mathbf{H}_{d_t}$ *such that* $\mathbf{H}_{d_t}\mathbf{K}_{d_t} = \mathbf{0}$.

Note that the column space of $\mathbf{HK}_{d_t}$ are exactly the subspace $\mathcal{U}_{d_t} \cap \mathcal{H}$ in (2), and all entries in the last $\sum_{d'>d_t} r_{d'}$ rows of $\mathbf{HK}_{d_t}$ are zero, so the computation of (4) only involve arithmetic over GF($2^{2^{d_t}}$). Moreover, in order to check (4), it suffices to select rank($\mathbf{HK}_{d_t}$) linearly independent column vectors in $\mathbf{HK}_{d_t}$, juxtapose them into a matrix $\mathbf{H}'$, and check whether rank($\mathbf{GH}'$) $= n$. With the number m of received packets at t increasing, the matrix $\mathbf{K}_{d_t}$ and $\mathbf{H}'$ can be established in the following iterative way.

Algorithm 1. *Denote by $\mathbf{h}^m$ the N-dimensional coding vector over GF(2) for the m^{th} received packet at receiver t. Without loss of generality, assume that there is at least one non-zero entry in $\mathbf{h}^m$. Let $\mathbf{h}^m_{d_t}$ denote the vector restricted from $\mathbf{h}^m$ to the last $\sum_{d'>d_t} r_{d'}$ entries. The next procedure efficiently produces desired $\mathbf{K}_{d_t}$ and $\mathbf{H}'$.*

Initialization. Let $\mathbf{K}_{d_t}$, $\mathbf{H}'$, $\mathbf{B}$ and $\mathbf{B}_{d_t}$ be empty matrices. They are to consist of a m rows, N rows, N rows and $\sum_{d'>d_t} r_{d'}$ rows respectively.

Iteration. Consider the case that the m^{th} packet with coding vector $\mathbf{h}^m$ is just received, and assume receiver t has dealt with the former $m-1$ coding vectors $\mathbf{h}^j, 1 \le j < m$. Perform either of the following two depending on $\mathbf{h}^m_{d_t}$.

- *If $\mathbf{h}^m_{d_t}$ is a zero vector, then update $\mathbf{K}_{d_t}$ as*

$$\mathbf{K}_{d_t} = \begin{bmatrix} \mathbf{K}_{d_t} & \mathbf{0} \\ 0 \dots 0 & 1 \end{bmatrix}, \tag{5}$$

  and respectively append a zero column vector to $\mathbf{B}$ and to $\mathbf{B}_{d_t}$ on the right. Further check whether $\mathbf{h}^m$ is a GF(2)-linear combination of columns in $\mathbf{H}'$. If so, keep $\mathbf{H}'$ unchanged. Otherwise, update $\mathbf{H}'$ as $[\mathbf{H}'\ \mathbf{h}^m]$. The iteration for the current value of m completes.
- *If $\mathbf{h}^m_{d_t}$ is not a zero vector, check whether it is a GF(2)-linear combination of columns in $\mathbf{B}_{d_t}$. If no, respectively update $\mathbf{B}$, $\mathbf{B}_{d_t}$ and $\mathbf{K}_{d_t}$ as*

$$\mathbf{B} = [\mathbf{B}\ \mathbf{h}^m], \mathbf{B}_{d_t} = [\mathbf{B}_{d_t}\ \mathbf{h}^m_{d_t}], \mathbf{K}_{d_t} = \begin{bmatrix} \mathbf{K}_{d_t} \\ 0 \dots 0 \end{bmatrix}, \tag{6}$$

  and the iteration for the current value of m completes. Otherwise, perform the following steps. First compute an $(m-1)$-dimensional vector $\mathbf{k}$ subject to $\mathbf{B}_{d_t}\mathbf{k} = \mathbf{h}^m_{d_t}$, and then update $\mathbf{K}_{d_t}$ as

$$\mathbf{K}_{d_t} = \begin{bmatrix} \mathbf{K}_{d_t} & \mathbf{k} \\ 0 \dots 0 & 1 \end{bmatrix}. \tag{7}$$

  Further compute a new vector $\mathbf{v} = \mathbf{Bk} + \mathbf{h}^m$, and respectively append a zero column vector to $\mathbf{B}$ and to $\mathbf{B}_{d_t}$ on the right. Check whether $\mathbf{v}$ is a GF(2)-linear combination of columns in $\mathbf{H}'$. If so, keep $\mathbf{H}'$ unchanged. Otherwise, update $\mathbf{H}'$ as $[\mathbf{H}'\ \mathbf{v}]$. The iteration for the current value of m completes.

Note that after the above procedure, the sum of the number of nonzero columns in $\mathbf{B}_{d_t}$ and the number of columns in $\mathbf{K}_{d_t}$ is m. The nonzero columns of $\mathbf{B}_{d_t}$ keep to form a basis of the column space of $\mathbf{H}_{d_t} = [\mathbf{h}^1_{d_t} \dots \mathbf{h}^m_{d_t}]$. The columns of $\mathbf{K}_{d_t}$ keep to form a basis of the null space spanned by columns of $\mathbf{H}_{d_t}$. The columns in $\mathbf{H}'$ keep to be a basis of the column space of $\mathbf{HK}_{d_t}$, where $\mathbf{H} = [\mathbf{h}^1 \dots \mathbf{h}^m]$.

Example 3. *Assume that $D = 2$, $n = r_0 = 3$, and $r_1 = r_2 = 1$. The 3×5 precoding matrix $\mathbf{G}$ is designed as*

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & \alpha & \beta \\ 0 & 1 & 0 & \alpha^2 & \beta \\ 0 & 0 & 1 & 1 & \beta \end{bmatrix}$$

where β is a primitive element in GF(2^4) and $\alpha = \beta^5$, which can be regarded as a primitive element of $\mathrm{GF}(2^2) \subset \mathrm{GF}(2^4)$.

Assume that at a receiver t, GF(2^2) is the largest field for computation, and 4 packets have been received with the columnwise juxtaposition of the respective coding vectors prescribed by

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

As $\mathbf{H}_{d_t} = [1\ 1\ 1\ 1]$ herein, the aforementioned iterative approach can yield the following $\mathbf{K}_{d_t}$ and concomitant $\mathbf{H}'$:

$$\mathbf{K}_{d_t} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{H}' = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix},$$

where the columns of $\mathbf{H}'$ form a basis for the subspace $\mathcal{U}_{d_t} \cap \mathcal{H}$. Consequently, $\mathbf{GH}' = \begin{bmatrix} 1 & 1 & \alpha \\ 1 & 0 & 1+\alpha^2 \\ 0 & 1 & 0 \end{bmatrix}$. Since $1 + \alpha + \alpha^2 = 0$ in GF(2^2), $\mathrm{rank}(\mathbf{GH}') = 2$, that is, (4) does not hold. Therefore, the receiver requires to receive more packets before decoding all original packets.

Assume $\mathbf{h}^5 = [1\ 0\ 0\ 1\ 1]^{\mathrm{T}}$ is the coding vector for the 5^{th} received packet. Then, the matrix $\mathbf{K}_{d_t}$ is dynamically updated to

$$\mathbf{K}_{d_t} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

but there is no change for $\mathbf{H}'$, because $\mathbf{H} \cdot [1\ 0\ 0\ 0\ 1]^{\mathrm{T}}$ belongs to the column space of $\mathbf{H}'$.

Assume $\mathbf{h}^6 = [0\ 0\ 1\ 0\ 0]^{\mathrm{T}}$ is the coding vector for the 6^{th} received packet. First, dynamically update $\mathbf{K}_{d_t}$ to

$$\mathbf{K}_{d_t} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

Then, as $\mathbf{h}^6 = [0\ 0\ 1\ 0\ 0]^{\mathrm{T}}$ does not belong to the column space of $\mathbf{H}'$, update $\mathbf{H}'$ as $[\mathbf{H}'\ \mathbf{h}^6]$:

$$\mathbf{H}' = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Consequently, $\mathbf{GH}' = \begin{bmatrix} 1 & 1 & \alpha & 0 \\ 1 & 0 & 1+\alpha^2 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$, and it has full rank 3, so the receiver can recover the source packets. Actually, in this case, the source packets can be recovered by merely GF(2)-based operations.

In two special cases that $d_t = D$ and $d_t = 0$, *i.e.*, receiver t has the highest and the lowest computational power respectively, (4) degenerates to a more concise form.

Corollary 2. *When $d_t = D$, (4) is equivalent to*

$$\mathrm{rank}(\mathbf{GH}) = n. \tag{8}$$

When $d_t = 0$, (4) *is equivalent to*

$$\text{rank}(\mathbf{H}) - \text{rank}(\mathbf{H}_{d_t}) = n. \tag{9}$$

Recall that Fulcrum [13,14] can be regarded as a special RLNC scheme of our framework. One may notice that in Fulcrum, the decoding rule over GF(2) at a receiver is

$$\text{rank}(\mathbf{H}) = N, \tag{10}$$

which is sufficient but not necessary. In contrast, (9) is both necessary and sufficient. As to be seen in Section 5, there is an observable performance gain when (9) is adopted as the decoding rule instead of (10). Moreover, our proposed scalable RLNC is more flexible than Fulcrum, because the receivers with intermediate computational power can fully utilize its decoding capability to decode over intermediate fields (rather than only over GF(2)), so that the number of required coded packets can be reduced.

3.4. Decoding Complexity Analysis

In this subsection, we briefly analyze the computational complexity of the proposed scalable RLNC scheme at receiver t with the field constraint GF(2^{d_t}). We assume that after a sufficiently large recoding process over GF(2), the last r positions in every received binary column vector $\mathbf{h}$, which corresponds to the r precoded packets generated over the larger fields than GF(2), are nonzero. According to Corollary 1, when enough coded packets have been received such that the condition

$$\text{rank}(\mathbf{G}(\mathbf{H}\mathbf{K}_{d_t})) = n$$

is satisfied, receiver t can recover all original packets by linear combining n coded packets over GF(2^{d_t}). Accordingly, it requires at most n^2M/d_t multiplications and $n(n-1)M/d_t$ additions over GF(2^{d_t}) in the decoding process. Following the same consideration in [4,18,19], we assume that it respectively takes d_t and $2d_t^2$ binary operations to realize addition and multiplication between two elements in GF(2^{d_t}). Consequently, the total number of required binary operations can be characterized as $\mathcal{O}(Mnd_t)$ to recover every M-bit original packet.

Herein, we did not consider the complexity to compute the inverse matrix of $\mathbf{G}\mathbf{H}\mathbf{K}_{d_t}$ because in practice the packet length M is much larger than n, and this convention has also been adopted in [4,19] for computational complexity analysis.

4. Optimal Construction of Precoding Matrix G

Based on the analysis in the previous section, we are motivated to carefully design such a precoding matrix $\mathbf{G}$ that the full rank of $\mathbf{H}$ is equivalent to the full rank of $\mathbf{GH}$, which can optimize the decodability performance for fixed parameters n and N. To achieve this goal, for the precoding matrix $\mathbf{G}$, we first introduce the following condition that is stronger than the conventional maximal distance separable (MDS) property.

Definition 1. *An* $n \times N$ *matrix* $\mathbf{G}$ *over GF*(2^{2^D}) *is said to be* MDS under GF(2)-mapping *if for any full-rank* $N \times n$ *matrix* $\mathbf{H}$ *over GF*(2), $\text{rank}(\mathbf{GH}) = n$.

Recall that if $\mathbf{G}$ satisfies the conventional MDS property, all n columns in it are linearly independent. Obviously, the conventional MDS property is a prerequisite for the proposed MDS property under GF(2)-mapping. However, Example 3 demonstrates an MDS matrix $\mathbf{G}$ that is not MDS under GF(2)-mapping. To the best of our knowledge, except for a brief attempt in [13], there is no prior literature involving the construction of a matrix satisfying the MDS property under GF(2)-mapping. We next characterize an equivalent condition on

the MDS property under GF(2)-mapping, so as to facilitate the explicit construction. Given an $n \times N$ matrix $\mathbf{G}$, let $\mathcal{C}$ denote the set of row vectors generated by $\mathbf{G}$:

$$\mathcal{C} = \{\mathbf{mG} : \mathbf{m} \in \mathrm{GF}(2^{2^D})^n\}. \tag{11}$$

For every $\mathbf{c} \in \mathcal{C}$, let $\mathcal{N}_\mathbf{c}$ denote its null space in $\mathrm{GF}(2^{2^D})^N$.

Theorem 2. *An $n \times N$ matrix $\mathbf{G}$ is MDS under GF(2)-mapping if and only if*

$$\dim(\mathcal{N}_\mathbf{c} \cap \mathrm{GF}(2)^N) < n, \ \forall \ \mathbf{c} \in \mathcal{C} \backslash \{\mathbf{0}\} \tag{12}$$

Proof. We prove the theorem in a contrapositive argument. Assume that there exists a nonzero $\mathbf{c} \in \mathcal{C}$ such that $\dim(\mathcal{N}_\mathbf{c} \cap \mathrm{GF}(2)^N) \geq n$, and let $\mathbf{m}$ be a row vector over $\mathrm{GF}(2^{2^D})$ satisfying $\mathbf{c} = \mathbf{mG}$. Then, we can select n linearly independent column vectors $\mathbf{h}_1, \ldots, \mathbf{h}_n$ over GF(2) from $\mathcal{N}_\mathbf{c}$. Write $\mathbf{H} = [\mathbf{h}_1 \ldots \mathbf{h}_n]$. Thus, $\mathbf{mGH} = \mathbf{cH} = \mathbf{0}$, so that $\mathbf{GH}$ is not full rank n, *i.e.*, $\mathbf{G}$ is not MDS under GF(2)-mapping.

Assume that $\mathbf{G}$ is not MDS under GF(2)-mapping, and let $\mathbf{H}$ be a full rank $N \times n$ matrix over GF(2) subject to $\mathrm{rank}(\mathbf{GH}) < n$. Then, there exists an n-dimensional row vector $\mathbf{m}$ such that $\mathbf{mGH} = \mathbf{0}$. Write $\mathbf{c} = \mathbf{mG}$, so that $\mathbf{cH} = \mathbf{0}$. Since $\mathbf{H}$ is full rank n, there are at least n linearly independent vectors (which are the columns of $\mathbf{H}$) belonging to $\mathcal{N}_\mathbf{c}$, *i.e.*, $\dim(\mathcal{N}_\mathbf{c} \cap \mathrm{GF}(2)^N) \geq n$. □

For $\mathbf{c} \in \mathcal{C}$, let $\eta(\mathbf{c})$ denote the number of elements in $\mathbf{c}$ belonging to $\mathrm{GF}(2^{2^D}) \backslash \{0, 1\}$, and define an indicator δ which is set to 1 if $\mathbf{c}$ consists of an element equal to 1 and set to 0 otherwise. The following is a useful corollary of Theorem 2.

Corollary 3. *If an $n \times N$ matrix $\mathbf{G}$ is MDS under GF(2)-mapping, then the followings hold*

$$\eta(\mathbf{c}) + \delta > N - n, \ \forall \ \mathbf{c} \in \mathcal{C} \backslash \{\mathbf{0}\}. \tag{13}$$

$$\mathcal{C} \cap \mathrm{GF}(2)^N = \{\mathbf{0}\}. \tag{14}$$

Proof. Assume there is a nonzero $\mathbf{c} \in \mathcal{C}$ with $\eta(\mathbf{c}) + \delta \leq N - n$, *i.e.*, $N - \eta(\mathbf{c}) \geq n + \delta$. Define a new vector $\mathbf{c}'$ by restricting to its components belonging to GF(2), so that the dimension of $\mathbf{c}'$ is $N - \eta(\mathbf{c})$. Thus, the dimension of the null space of $\mathbf{c}'$ in $\mathrm{GF}(2)^{N-\eta(\mathbf{c})}$ is $N - \eta(\mathbf{c}) - \delta$, which is no smaller than n. Correspondingly, $\dim(\mathcal{N}_\mathbf{c} \cap \mathrm{GF}(2)^N) \geq n$, a contradiction to the MDS property under GF(2)-mapping for $\mathbf{G}$ according to (12).

If there is a nonzero $\mathbf{c} \in \mathcal{C}$ belonging to $\mathrm{GF}(2)^N$, then $\eta(\mathbf{c}) = 0$ so that (13) cannot hold as $N > n$, and thus $\mathbf{G}$ cannot be MDS under GF(2)-mapping. □

Conditions (13) and (14) are insufficient for the MDS property under GF(2)-mapping. The key reason is the possibility of the following

$$\sum\nolimits_{<j>} \alpha_j \in \mathrm{GF}(2), \ \alpha_j \in \mathrm{GF}(2^{2^D}) \backslash \{0, 1\}. \tag{15}$$

For this reason, we should pay more attention in the matrix design to avoid the involvement of those elements in (15). The special case $N = n + 1$ is easier to manipulate.

Proposition 1. *When $N = n + 1$, an $n \times N$ matrix $\mathbf{G}$ is MDS under GF(2)-mapping if and only if* (14) *holds.*

Proof. The necessity has been shown in Corollary 3. To prove sufficiency, assume (14) holds for $\mathcal{C}$ defined in (11) based on $\mathbf{G}$. Let $\mathbf{c}$ be an arbitrary vector in $\mathcal{C}$. As (14) holds, $\eta(\mathbf{c}) > 0$. In the case $\eta(\mathbf{c}) = 1$, there must be at least one element in $\mathbf{c}$ equal to 1, because otherwise we can find another vector in $\mathcal{C}$ with all elements in GF(2), a contradiction to (14). Thus, $\dim(\mathcal{N}_\mathbf{c} \cap \mathrm{GF}(2)^N) < n$ for this case. Consider the case $\eta(\mathbf{c}) \geq 2$. Without loss of

generality, write $\mathbf{c} = [c_1 \ \dots \ c_{\eta(\mathbf{c})} \ 0 \ \dots \ 0]$ with $c_j \neq 0$. We can assume c_j not all identical, because otherwise we can again find another vector in $\mathcal{C}$ with all elements in GF(2), a contradiction to (14). Moreover, for arbitrary two elements $a, b \in \mathrm{GF}(2^{2^D})$, $a + b = 0$ if and only if $a = b$. Hence, there are at most $\eta(\mathbf{c}) - 2$ linearly independent vectors in $\mathrm{GF}(2)^{\eta(\mathbf{c})}$ that are in the null space of $\mathbf{c}$, which further implies $\dim(\mathcal{N}_{\mathbf{c}} \cap \mathrm{GF}(2)^N) < n$. We have proved (12) and thus the considered $\mathbf{G}$ is MDS under GF(2)-mapping. □

Corollary 4. *When $N = n + 1$, there exists a systematic $n \times N$ matrix $\mathbf{G} = [\mathbf{I}_n \ \mathbf{A}_D]$ over GF(2^{2^D}) that is MDS under GF(2)-mapping if and only if $n < 2^D$.*

Proof. Assume $n < 2^D$. Define an n-dimensional column vector $\mathbf{a} = [\alpha, \alpha^2, \dots, \alpha^n]^{\mathrm{T}}$, where α is a primitive element of GF(2^{2^D}). In this way, all elements in $\mathbf{a}$ are distinct and every GF(2)-combination $\sum_{1 \leq j \leq n} a_j \alpha^j$ among them does not belong to GF(2). By Proposition 1, $[\mathbf{I}_n \ \mathbf{a}]$ is an MDS matrix under GF(2)-mapping. When $n \geq 2^D$, let $\mathbf{a} = [\alpha_1, \dots, \alpha_n]^{\mathrm{T}}$ be an arbitrary n-dimensional vector in GF(2^{2^D}). In order to make $[\mathbf{I}_n \ \mathbf{a}]$ MDS under GF(2)-mapping, according to (14) in Corollary 3, there is not any element α_j belonging to GF(2). If there is a basis, say $\{\alpha_1, \dots, \alpha_{2^D}\}$ of GF(2^{2^D}) in $\mathbf{a}$, then 1 can be written as a GF(2)-linear combination of the basis, so that (14) does not hold. If there is not a basis of GF(2^{2^D}) in $\mathbf{a}$, then there exists an n-dimensional nonzero row vector $\mathbf{v}$ over GF(2) subject to $\mathbf{va} = 0$, so that (14) does not hold either. Thus, it is impossible for $[\mathbf{I}_n \ \mathbf{a}]$ to be MDS under GF(2)-mapping. □

Based on the above corollary, the required field size is exponentially larger than N in the construction of an $n \times N$ systematic MDS matrix under GF(2)-mapping. This implies that it is infeasible to construct such a practical precoding matrix $\mathbf{G}$ for large N. For this reason, it is alternative to choose to randomly generate $\mathbf{G}$, which may cause a near-optimal decodability behavior as illustrated in the next example.

Example 4. *Define the following vectors $\mathbf{a}_1 = [\alpha \ \alpha^2 \ \alpha^3 \ \dots \ \alpha^7]^{\mathrm{T}}$ and $\mathbf{a}_2 = [\alpha^2 \ \alpha^4 \ \alpha^6 \ \dots \ \alpha^{14}]^{\mathrm{T}}$ over GF(2^8) in which α is a primitive element. It can be checked that both matrices $[\mathbf{I}_7 \ \mathbf{a}_1]$ and $[\mathbf{I}_7 \ \mathbf{a}_2]$ are MDS under GF(2)-mapping. Although the 7×9 matrix $\mathbf{G} = [\mathbf{I}_7 \ \mathbf{a}_1 \ \mathbf{a}_2]$ is not MDS under GF(2)-mapping, among 42435 7-dimensional subspaces of $\mathrm{GF}(2)^9$, there are only 127 instances to break the desired MDS property, that is, every basis for each of the instances forms a 9×7 matrix $\mathbf{H}$ with $\mathrm{rank}(\mathbf{GH}) < 7$.*

5. Numerical Analysis

In this section, we numerically analyze the performance of applying the proposed systematic scalable RLNC scheme to a wireless broadcast network, which is a classical model to demonstrate the advantage of RLNC [4–7]. The number n of original packets in a batch is varied from $n = 6$ to 24. In every timeslot, the source broadcasts one packet to all receivers. The memoryless and independent packet loss probability for every receiver is $p_e = 0.2$, that is, in every timeslot, every receiver can successfully receive a packet with probability $1 - p_e$. We consider the scheme with parameters $D = 2$, $r = 2$ where $r_1 = r_2 = 1$. In the $n \times N$ precoding matrix $\mathbf{G} = [\mathbf{I}_n \ \mathbf{A}_1 \ \mathbf{A}_2]$, the entries in $\mathbf{A}_1$ and $\mathbf{A}_2$ are randomly selected from GF(2^2) and GF(2^4), respectively. In the numerical analysis of scalable RLNC, the single source s has n original packets to be broadcast to a total of 30 receivers with different decoding capabilities. Specifically, the 30 receivers fall into 3 different groups and the 10 receivers in every group has the same decoding capability, and can decode based on the decoding rule (4) over GF(2), GF(2^2) and GF(2^4), respectively. In the first n timeslots, the source broadcasts n original packets, whose coding vectors are $(n + r)$-dimensional unit vectors, to all receivers. Starting from timeslot $n + 1$, the source broadcasts coded packets, each of which is generated based on a random N-dimensional column vector $\mathbf{h}$ over GF(2), till all the receivers can recover the n original packets. Herein, for every parameter setting and every considered RLNC scheme, we conduct 1200 independent rounds of simulation which result in 95% confidence intervals.

Figure 2 depicts the average group completion delay per packet for the 3 groups of receivers, respectively labeled as "Scalable-GF(2^x)", $x \in \{1, 2, 4\}$ of the considered scalable RLNC scheme. The group completion delay means the number of extra coded packets the source broadcasts till all the 10 receivers in the group can recover n original packets. For a better comparison, the figure also depicts the average group completion delay per packet, labeled as "RLNC-GF(2^x)", for a group of 10 receivers of three *different* classical systematic RLNC schemes over different fields GF(2^x), $x \in \{1, 2, 4\}$. Recall that in the classical systematic RLNC scheme over GF(2^x), the source first broadcasts n original packets and then randomly coded packets with n-dimensional coding vectors over GF(2^x). One may observe from Figure 2 that for the case of GF(2^4), the average completion delay of scalable RLNC is almost same as the classical RLNC. Over other smaller fields, even though scalable RLNC yields higher average completion delay than classical RLNC, it simultaneously guarantees the decoding compatibility at heterogeneous receivers, which cannot be endowed by classical RLNC schemes. For instance, assume that the source adopts classical RLNC over GF(2^2) to generate coded packets. On one hand, the group of receivers with decoding capability constrained to GF(2) will fail to recover the original packets. On the other hand, the groups of receivers with decoding capability over GF(2^4) cannot fully utilize their higher computational power so that the average completion delay cannot be further reduced compared with decoding over GF(2^2). As a result, the performance loss for the cases of smaller fields in our proposed scalable RLNC compared to classical RLNC is the cost of decoding compatibility over different fields.

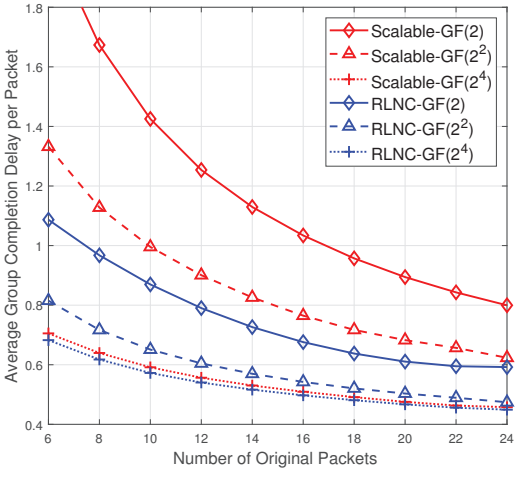


Figure 2. The average group completion delay per packet for the receivers of different systematic RLNC schemes in a wireless broadcast network with $r_1 = r_2 = 1$ and packet loss probability $p_e = 0.2$.

For the considered systematic scalable RLNC scheme, recall that for decoding over GF(2) in the proposed scalable RLNC, Equation (9) obtained in Sec. III is a necessary and sufficient rule while Equation (10), originally adopted in [13,14] for Fulcrum decoding, is a non-necessary rule. Figure 3 compares the average group completion delay per packet for 10 receivers as well as the average completion delay per packet at a single receiver when the receivers adopt different decoding rules (9) and (10) over GF(2). For the average completion delay at a single receiver, a noticeable performance gain can be observed. In particular, when the number of original packets is less than 10, the average completion delay at a single receiver is reduced by more than 20% based on the decoding rule (9) instead of (10). For the average group completion delay, the performance gain by adopting (9) instead of (10) becomes less obvious because it is offset by the increasing number of receivers in a group. Compared with Fulcrum, which only supports decoding over the smallest field GF(2) or

the largest field GF(2^{2^D}), in addition to the performance gain illustrated in Figure 3, our proposed scalable RLNC is more flexible. This is because the receivers with intermediate computational power can fully utilize its decoding capability to decode over intermediate fields (rather than only over GF(2)), so that the average completion delay can be reduced.

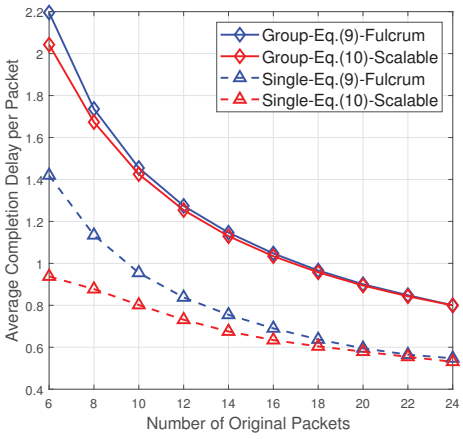


Figure 3. The average group completion delay per packet for 10 receivers as well as the average completion delay per packet at a single receiver when the receivers adopt different decoding rules (9) and (10) over GF(2).

In the remaining part of this section, we shall analyze the performance of our scalable RLNC scheme by adjusting the *sparsity* $0 < P_h < 1$ of $\mathbf{h}$, which controls to the probability for every component in $\mathbf{h}$ to be one. Specifically, for every packet to be transmitted by the source, the expected number of precoded packets to form it is $P_h(n+r)$. In previous analysis of this section, P_h is set to $1/2$. We next consider a more *sparse* $\mathbf{h}$ with $P_h \leq 1/2$.

According to the work in [20], given that there are $(i-1)$ $(n+r)$-dimensional linearly independent binary vectors with sparsity P_h, the probability that a new randomly generated $(n+r)$-dimensional binary coding vector $\mathbf{h}^i$ with sparsity P_h is linearly independent with them is lower bounded by

$$1-(1-P_h)^{n+r-i}. \tag{16}$$

This bound indicates that except for the case i close to $(n+r)$, the lower bound keeps very close to 1. Further, at the end of Sec. IV, we have illustrated that a random $\mathbf{G}$ will bring a near-optimal decodability behavior, that is, the full rank of $\mathbf{H}$ will lead to the full rank of $\mathbf{GH}$ with high probability. As a result, although our proposed scalable RLNC scheme with two-stage encoding process is different from the conventional sparse RLNC described in [20], we are motivated to bring the *sparsity* into our proposed scheme and attempt to meet a balance between completion delay and decoding complexity. The work in [14] has taken the sparsity into consideration in their performance analysis of Fulcrum, which is a special instance of our proposed scalable RLNC scheme.

In simulation, besides the consideration of sparsity P_h, we also extend the value range of n from $[6,24]$ to $[8,64]$ and set $r_1 = r_2 = 2$. All other parameter settings are same as those in Figure 2. The 3 solid curves in Figure 4 illustrate the average group completion delay per packet for the 3 groups of 10 receivers under different field constraints GF(2), GF(2^2) and GF(2^4) for scalable RLNC with sparsity $P_h = 1/2$. The 3 dotted curves in Figure 4 illustrate the completion delay performance under different field constraints GF(2), GF(2^2) and GF(2^4) for scalable RLNC with $P_h = 1/4$. It is interesting to observe that with the batch size n increasing, under the same decoding constraint (*i.e.*, two curves in the same color), the completion delay performance for the case $P_h = 1/4$ will converge to the case $P_h = 1/2$. This result indicates that the lower bound in (16) is rather loose when i is close to $n+r$,

and moreover, for a large enough batch size n, a more sparse vector $\mathbf{h}$ will not affect the completion delay performance much in a wireless broadcast network.

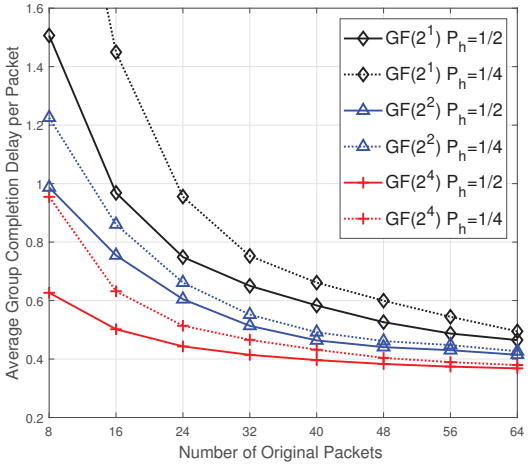


Figure 4. The average group completion delay per packet for scalable RLNC with different sparsity P_h.

6. Conclusions

In this work, the proposed scalable RLNC framework based on embedded fields aims at endowing heterogeneous receivers with different decoding capabilities in complex network environments. In this framework, we derive a general decodability condition by the arithmetic compatibility of embedded fields. Moreover, we theoretically study the specific construction of an optimal precoding matrix $\mathbf{G}$ and illustrate the rationality of the near-optimal behavior of a randomly generated $\mathbf{G}$.

In numerical analysis, we demonstrate that the proposed scalable RLNC not only guarantees a better decoding compatibility compared with classical RLNC, but also provides a better decoding performance over GF(2) in terms of smaller average completion delay compared with Fulcrum. In addition, the numerical analysis also demonstrates that for a large enough batch size, the sparsity of the vector $\mathbf{h}$ does not affect the completion delay performance much. As a potential future work, the theoretical insight behind this observation deserves a further investigation so as to facilitate the design of a scalable RLNC scheme with a better tradeoff between decoding complexity and completion delay.

Last, the present scalable RLNC framework assumes block-based coding. It would also be interesting to make use of the embedded fields structure to generalize the design of sliding window-based random linear coding schemes such as the ones studied in [21–23].

Author Contributions: R.Z. and Q.S. conceived and designed the mathematical model. H.T. designed the whole coding framework and wrote the paper with the help of Q.S., K.L. and Z.L. All authors were involved in problem formulation, data analysis and editing of this paper. All authors have agreed to the published version of the manuscript.

Funding: This work was partially supported by the National Natural Science Foundation of China under Grant 62101028 and 62271044, and by China Postdoctoral Science Foundation under Grant 2021TQ0031, and by Huawei TC20211126644 and by China Telecom 20222910016.

Acknowledgments: This paper was partly presented in [24] at the IEEE/CIC International Conference on Communications in China (ICCC) 2021.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ho, T.; Médard, M.; Koetter, R.; Karger, D.R.; Effros, M.; Shi, J.; Leong, B. A random linear network coding approach to multicast network. *IEEE Trans. Inf. Theory* **2006**, *52*, 4413–4430. [CrossRef]
2. Huang, J.; Gharavi, H.; Yan, H.; Xing, C.C. Network coding in relay-based device-to-device communications. *IEEE Netw.* **2017**, *31*, 102–107. [CrossRef] [PubMed]
3. Asterjadhi, A.; Fasolo, E.; Rossi, M.; Widmer, J.; Zorzi, M. Toward network coding-based protocols for data broadcasting in wireless ad hoc networks. *IEEE Trans. Wirel. Commun.* **2010**, *9*, 662–673. [CrossRef]
4. Su, R.; Sun Q.; Zhang Z. Delay-complexity trade-off of random linear network coding in wireless broadcast. *IEEE Trans. Commun.* **2020**, *68*, 5606–5618. [CrossRef]
5. Eryilmaz, A.; Ozdaglar, A.; Médard, M.; Ahmed, E. On the delay and throughput gains of coding in unreliable networks. *IEEE Trans. Inf. Theory* **2008**, *54*, 5511–5524. [CrossRef]
6. Swapna, B.T.; Eryilmaz, A.; Shroff, N.B. Throughput-delay analysis of random linear network coding for wireless broadcasting. *IEEE Trans. Inf. Theory* **2013**, *59*, 6328–6341. [CrossRef]
7. Zhu, H.; Ouahada, K. Investigating random linear coding from a pricing perspective. *Entropy* **2018**, *20*, 548. [CrossRef] [PubMed]
8. Wunderlich, S.; Cabrera, J.A.; Fitzek, F.H.; Reisslein, M. Network coding in heterogeneous multicore IoT nodes with DAG scheduling of parallel matrix block operations. *IEEE Internet Things J.* **2017**, *4*, 917–933. [CrossRef]
9. Heide, J.; Lucani, D.E. Composite extension finite fields for low overhead Network Coding: Telescopic codes. In Proceedings of the 2015 IEEE International Conference on Communications (ICC), London, UK, 8–12 June 2015.
10. Marcano, N.J.H.; Heide, J.; Lucani, D.E.; Fitzek, F.H. On the overhead of telescopic codes in network coded cooperation. In Proceedings of the 2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall), Boston, MA, USA, 6–9 September 2015.
11. Heide, J. Composite extension finite fields for distributed storage erasure coding. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 22–27 May 2016.
12. Yazdani, V.; Lucani, D. Revolving codes: Overhead and computational complexity analysis. *IEEE Commu. Lett.* **2021**, *25*, 374–378. [CrossRef]
13. Lucani, D.E.; Pedersen, M.V.; Ruano, D.; Sørensen, C.W.; Fitzek, F.H.; Heide, J.; Geil, O.; Nguyen, V.; Reisslein, M. Fulcrum: Flexible network coding for heterogeneous devices. *IEEE Access* **2018**, *6*, 77890–77910. [CrossRef]
14. Nguyen, V.; Tasdemir, E.; Nguyen, G.T.; Lucani, D.E.; Fitzek, F.H.; Reisslein, M. DSEP Fulcrum: Dynamic sparsity and expansion packets for fulcrum network coding. *IEEE Access* **2020**, *8*, 78239–78314. [CrossRef]
15. Lidl, R.; Niederreiter, H. *Finite Fields*, 3rd ed.; Cambridge University Press: Cambridge, UK, 1997.
16. Schutz, B.; Aschenbruck, N. Packet-preserving network coding schemes for padding overhead reduction. In Proceedings of the 2019 IEEE 44th Conference on Local Computer Networks (LCN), Osnabrueck, Germany, 14–17 October 2019.
17. Taghouti, M.; Lucani, D.E.; Cabrera, J.A.; Reisslein, M.; Pedersen, M.V.; Fitzek, F.H. Reduction of padding overhead for RLNC media distribution with variable size packets. *IEEE Trans. Broadcast.* **2019**, *65*, 558–576. [CrossRef]
18. Tang, H.; Sun, Q.T.; Li, Z.; Yang, X.; Long, K. Circular-shift linear network coding. *IEEE Trans. Inf. Theory* **2019**, *65*, 65–80. [CrossRef]
19. Hou, H.; Shum, K.W.; Chen, M.; Li, H. BASIC codes: Low-complexity regenerating codes for distributed storage systems. *IEEE Trans. Inf. Theory* **2016**, *62*, 3053–3069. [CrossRef]
20. Feizi, S.; Lucani, D.E.; Sørensen, C.W.; Makhdoumi, A.; Médard, M. Tunable Sparse Network Coding for Multicast Networks. In Proceedings of the 2014 IEEE International Symposium on Network Coding (NetCod), Aalborg Oest, Denmark, 27–28 June 2014; pp. 1–6.
21. Karetsi, F.; Papapetrou, E. Lightweight network-coded ARQ: An approach for ultra-reliable low latency communication. *Comput. Commun.* **2022**, *185*, 118–129. [CrossRef]
22. Ma, S.; Liu, X.; Yan, Y.; Zhang, B.; Zheng, J. Sliding-window based batch forwarding using intra-flow random linear network coding. In Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 15–19 June 2020.
23. Tasdemir, E.; Nguyen, V.; Nguyen, G.T.; Fitzek, F.H.; Reisslein, M. FSW: Fulcrum sliding window coding for low-latency communication. *IEEE Access* **2022**, *10*, 54276–54290. [CrossRef]
24. Tang, H.; Zheng, R.; Li, Z.; Sun, Q.T. Scalable Network Coding over Embedded Fields. In Proceedings of the 2021 IEEE/CIC International Conference on Communications in China (ICCC), Xiamen, China, 28–30 July 2021; pp. 641–646.

entropy

MDPI

Article

Network Coding Approaches for Distributed Computation over Lossy Wireless Networks

Bin Fan [1,2], Bin Tang [1,2,*], Zhihao Qu [1,2] and Baoliu Ye [1,2]

[1] Key Laboratory of Water Big Data Technology of Ministry of Water Resources, Hohai University, Nanjing 211100, China
[2] School of Computer and Information, Hohai University, Nanjing 211100, China
* Correspondence: cstb@hhu.edu.cn

Abstract: In wireless distributed computing systems, worker nodes connect to a master node wirelessly and perform large-scale computational tasks that are parallelized across them. However, the common phenomenon of straggling (i.e., worker nodes often experience unpredictable slowdown during computation and communication) and packet losses due to severe channel fading can significantly increase the latency of computational tasks. In this paper, we consider a heterogeneous, wireless, distributed computing system performing large-scale matrix multiplications which form the core of many machine learning applications. To address the aforementioned challenges, we first propose a random linear network coding (RLNC) approach that leverages the linearity of matrix multiplication, which has many salient properties, including ratelessness, maximum straggler tolerance and near-ideal load balancing. We then theoretically demonstrate that its latency converges to the optimum in probability when the matrix size grows to infinity. To combat the high encoding and decoding overheads of the RLNC approach, we further propose a practical variation based on batched sparse (BATS) code. The effectiveness of our proposed approaches is demonstrated by numerical simulations.

Keywords: distributed computing; coded computation; network coding; lossy wireless network; BATS codes

Citation: Fan, B.; Tang, B.; Qu, Z.; Ye, B. Network Coding Approaches for Distributed Computation over Lossy Wireless Networks. *Entropy* **2023**, *25*, 428. https://doi.org/10.3390/e25030428

Academic Editor: Syed A. Jafar

Received: 10 January 2023
Revised: 20 February 2023
Accepted: 23 February 2023
Published: 27 February 2023

1. Introduction

In recent years, due to the proliferation of computationally intensive applications at the wireless edge, such as federated learning [1] and image recognition [2], wireless distributed computing has drawn great interest [3,4], where large-scale computational tasks are carried out by a cluster of wireless devices collaboratively. Meanwhile, due to the inherent randomness of wireless environment, wireless distributed computing systems are facing multiple challenges. One main challenge is called the straggler issue, where computing devices often experience unpredictable slowdown or even dropout during computation and communication, which can lead the computational task to much larger latency or even failure [5]. Another challenge is the packet-loss issue, where the packets can be lost during transmission due to severe channel fading of wireless networks.

In this paper, we consider a typical wireless distributed computing system consisting of multiple worker nodes and a master node. We focus on distributed matrix multiplication $\mathbf{y} = \mathbf{A}\mathbf{x}$, which forms the core of many computation-intensive machine learning applications, such as linear regression, and aims at tackling the two above challenges. One common approach to mitigate the effect of stragglers is providing redundancy through replication [6–8], which has been widely used in large distributed systems such as MapReduce [9] and Spark [10]. However, this kind of r-replication strategy can only tolerate r stragglers, and using a larger r increases the computation redundancy, which can lead to poor performance.

Recently, Lee et al. [11] firstly introduced coding-based computation framework, and then proposed an (n, k) maximum-distance-separable (MDS) code approach, such that the master node can recover the desired result from the local computation results of any k out of n worker nodes. Based on this, Das et al. further proposed a fine-grained model such that the partial results of stragglers can be leveraged. However, MDS codes fail to make full use of the partial work done by stragglers. Ferdinand et al. [12] and Kiani et al. [13] proposed approaches to make use of stragglers by allocating more fine-grained computing tasks to each worker. Very recently, Mallick et al. [14] proposed the use of rateless codes such as LT codes [15] and Raptor codes [16] and demonstrated that a rateless coding approach can achieve an asymptotically optimal latency. However, all these approaches assumed that the communication between each worker node and the master node is reliable and can only lead to inferior performance in wireless distributed computing.

In fact, the packet-loss issue has been widely investigated in communication networks, and the existing approaches roughly belong to two categories. The first is automatic repeat-request (ARQ) based, which employs feedback-based retransmissions to combat packet loss. It has been adopted by Han et al. [17] in a MDS-code-based wireless distributed computing system. However, the feedbacks from the master node can increase the computation latency significantly due to the inherent delays of feedback, especially when the communication traffic between worker nodes and the master node is large. The other is forward error correction (FEC)-based, employing error-correcting code to combat packet losses. Traditional FEC approaches mainly focus on achieving reliable transmission over each communication link, but in the context of distributed matrix multiplication, the objective is to recover the desired computation result. How to tackle both the straggler issue and the packet-loss issue for distributed matrix multiplication in wireless distributed computing system remains an open problem.

In this paper, by leveraging the linearity of matrix multiplication, we show how network coding [18] can be applied to solve the two issues efficiently in a joint manner. The main contributions of this paper are summarized as follows:

- We first propose a random linear network coding (RLNC) [19] based approach. In this approach, the matrix $\mathbf{A}$ to be multiplied is first split into multiple submatrices $\mathbf{A}_1, \ldots, \mathbf{A}_k$, and each worker node is assigned multiple submatrices, each of which is a random linear combination of the $\mathbf{A}_1, \ldots, \mathbf{A}_k$. Each worker node multiplies each assigned submatrix with the input $\mathbf{x}$, and it generates random linear combinations of submatrix-vector products that have been created for transmission. Once receiving enough packets with independent global encoding vectors, the master node can recover the desired result $\mathbf{Ax}$ by Gaussian elimination. We model the computation and communication process as a continuous-time trellis, and by conducting a probabilistic analysis of the connectivity of the trellis, we theoretically show that the latency of RLNC approach converges to the optimum in probability when the matrix size grows to infinity.
- Since RLNC approach has high encoding and decoding costs, we further propose a practical variation of RLNC approach based on batched sparse (BATS) code [20] and show how to optimize the performance of the BATS approach.
- We conducted numerical simulations to evaluate the proposed RLNC and BATS approaches. The simulation results show that both approaches can overcome the straggler issue and the packet-loss issue effectively and achieve near-optimal performance.

The reminder of the paper is organized as follows. Section 2 introduces the system model. Sections 3 and 4 introduce the RLNC approach and the BATS approach, respectively. Section 5 presents the numerical evaluation results. Finally, Section 6 concludes.

2. System Model

2.1. Coding-Based Wireless Distributed Computation

As shown in Figure 1, we consider a heterogeneous, wireless distributed computing system consisting of a master node and n heterogeneous worker nodes. These worker

nodes, denoted by $w_1, w_2, \ldots, w_n$, are connected wirelessly to the master node. We focus on the matrix-vector multiplication problem, whose goal is to compute the result $\mathbf{y} = \mathbf{A}\mathbf{x}$ for a given matrix $\mathbf{A} \in \mathbb{R}^{m \times d}$ and an arbitrary vector $\mathbf{x} \in \mathbb{R}^{d \times 1}$, where $\mathbb{R}$ is a set of real numbers. Our results can be directly extended to matrix-matrix multiplication, where $\mathbf{x}$ is a small matrix.

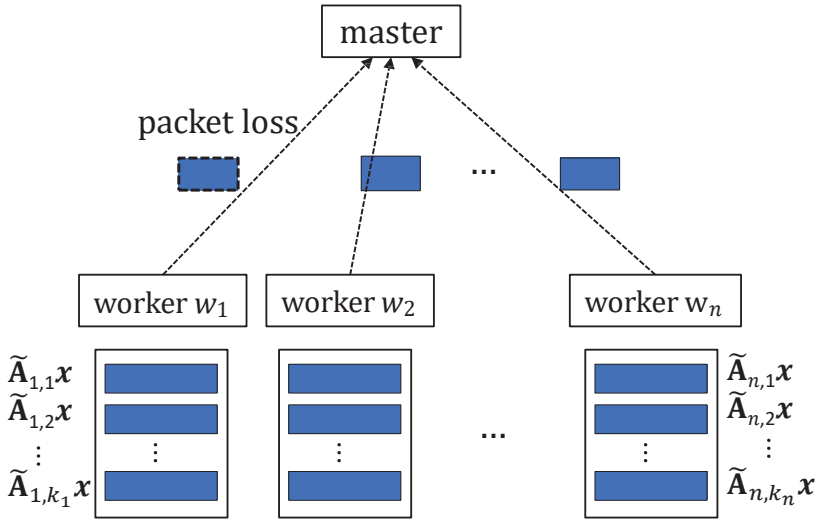


Figure 1. Illustration of the wireless distributed computing system for matrix multiplication.

In order to mitigate the effect of unpredictable node slowdown during computation and communication, we consider an error-correcting code based computing framework which consists of four components:

- **Encoding before computation**: The matrix $\mathbf{A}$ is first split along its rows equally into k submatrices $\mathbf{A}_1, \ldots, \mathbf{A}_k$, i.e., $\mathbf{A}^\mathsf{T} = [\mathbf{A}_1^\mathsf{T} \quad \mathbf{A}_2^\mathsf{T} \quad \cdots \quad \mathbf{A}_k^\mathsf{T}]$. Without loss of generality, here we assume that m/k is an integer. These submatrices are encoded into more submatrices using an error-correcting code, which are further placed on worker nodes. The submatrices assigned to worker node w_i are denoted as $\tilde{\mathbf{A}}_{i,1}, \tilde{\mathbf{A}}_{i,2}, \ldots, \tilde{\mathbf{A}}_{i,k_i}$, where k_i is the number of submatrices assigned to w_i. Here, we emphasize that, in many applications, such as linear regression, this encoding will be used for multiple computations with different inputs $\mathbf{x}$ [11], so that the encoding is often required to be executed before the arrival of any $\mathbf{x}$.
- **Computation at each worker node**: When an input $\mathbf{x}$ is arrived at the master node, the master node will broadcast $\mathbf{x}$ to all these worker nodes. Once worker node w_i receives $\mathbf{x}$, it will compute $\tilde{\mathbf{A}}_{i,1}\mathbf{x}, \tilde{\mathbf{A}}_{i,2}\mathbf{x}, \ldots, \tilde{\mathbf{A}}_{i,k_i}\mathbf{x}$ in a sequential manner.
- **Communication from each worker node**: During the computation, each worker node also keeps on sending its local computation results to the master node in some manner. For this, each submatrix-vector product which is a vector of length m/k is encapsulated into a packet. We assume that the communication link between worker i and the master node can be modeled as a packet erasure channel, where each packet is erased independently with probability ε_i. In order to combat these packet losses, each worker node can transmit its local computation results using a coding based approach.
- **Decoding at the master node**: Once the master node receives enough information, it will recover the desired result $\mathbf{y} = \mathbf{A}\mathbf{x}$ and notify all the worker nodes to stop the computation.

2.2. Delay Model

In this paper, we mainly focus on minimizing the latency, which is the time required by the wireless computing system so that the result $\mathbf{y} = \mathbf{A}\mathbf{x}$ can be successfully decoded at the master node by aggregating the results sent from the worker nodes. For the characterization of the latency, we consider the following two models, one for computation delay and the other for communication delay.

As in [14], we consider a computation delay model as follows. The computation delay at each worker node w_i consists of two parts. The first is an initial setup time before w_i starts to perform any submatrix-vector multiplication, denoted by X_i, which is assumed to follow an exponential distribution with rate λ_i. The second is a constant time for calculating each submatrix-vector product, which is denoted by τ_i. Hence, the delay for computing r submatrix-vector products by w_i is $X_i + \tau_i r$.

In order to characterize the straggling effect during the communication, we model the communication time of a packet from worker node i to the master node as a shifted-exponential distribution with rate μ_i and shift parameter θ_i. Additionally, the communication times of all packets are mutually independent. The model has also been adopted by [17,21].

3. A Network Coding Approach

In order to combat the straggling effects during both computation and communication and the packet losses during communication, in this section, we propose a random linear network coding (RLNC)-based approach and show that it can achieve optimal latency performance in the asymptotic sense, i.e., when the number of rows of $\mathbf{A}$ goes to infinity, when the overheads incurred are ignored. A practical version of this approach is given in the next section.

3.1. Description

We describe the RLNC based approach based on the computing framework given in Section 2.1:

Encoding before computation: In the RLNC-based approach, each submatrix $\tilde{\mathbf{A}}_{i,j}$ assigned to worker node w_i is a random linear combination of $\mathbf{A}_1, \ldots, \mathbf{A}_k$; i.e.,

$$\tilde{\mathbf{A}}_{i,j} = \sum_{e=1}^{k} c_{i,j,e}\mathbf{A}_e, \quad j = 1, 2, \ldots, k_i \tag{1}$$

where $c_{i,j,e}$ is chosen randomly and independently according to a standard normal distribution. Since this encoding approach is rateless, k_i can be arbitrarily large.

Computation at each worker node: When the worker node w_i receives an input $\mathbf{x}$, it starts to compute the local results $\tilde{\mathbf{y}}_{i,1} = \tilde{\mathbf{A}}_{i,1}\mathbf{x}, \tilde{\mathbf{y}}_{i,2} = \tilde{\mathbf{A}}_{i,2}\mathbf{x}, \ldots, \tilde{\mathbf{y}}_{i,k_i} = \tilde{\mathbf{A}}_{i,k_i}\mathbf{x}$, in a sequential manner.

Communication from each worker node: For each packet transmission starting at time t, the worker node w_i will generate a linear combination of all the local computation results in hand as

$$\hat{\mathbf{y}}_{i,t} = \sum_{j=1}^{d_i(t)} c'_j \tilde{\mathbf{y}}_{i,j}, \tag{2}$$

where $d_i(t)$ is the number of local results that have been computed before time t by w_i. Here, $(c'_1, \ldots, c'_{d_i(t)})$ is referred to as the local encoding vector of $\hat{\mathbf{y}}_{i,t}$.

Decoding at the master node: Due to the linearity of matrix-vector multiplication, we can see that

$$\begin{aligned}\hat{\mathbf{y}}_{i,t} &= \sum_{j=1}^{d_i(t)} c'_j \tilde{\mathbf{y}}_{i,j} = \sum_{j=1}^{d_i(t)} c'_j \tilde{\mathbf{A}}_{i,j}\mathbf{x} \\ &= \sum_{j=1}^{d_i(t)} c'_j \sum_{e=1}^{k} c_{i,j,e}\mathbf{A}_e\mathbf{x} \\ &= \sum_{e=1}^{k} \left(\sum_{j=1}^{d_i(t)} c'_j c_{i,j,e} \right) \mathbf{A}_e\mathbf{x}\end{aligned} \tag{3}$$

i.e., each packet received by the master node is a linear combination of $\mathbf{A}_1\mathbf{x}, \mathbf{A}_2\mathbf{x}, \ldots, \mathbf{A}_k\mathbf{x}$. Here,

$$\left(\sum_{j=1}^{d_i(t)} c'_j c_{i,j,1}, \sum_{j=1}^{d_i(t)} c'_j c_{i,j,2}, \ldots, \sum_{j=1}^{d_i(t)} c'_j c_{i,j,k} \right) \tag{4}$$

is referred to as the global encoding vector of $\hat{\mathbf{y}}_{i,t}$. Hence, when the master node receives enough packets that have k linearly independent global encoding vectors, it can recover the desired results $\mathbf{A}_1\mathbf{x}, \mathbf{A}_2\mathbf{x}, \ldots, \mathbf{A}_k\mathbf{x}$ by Gaussian elimination.

Overhead: Our RLNC approach suffers from its high encoding and decoding complexities, just like RLNC for communication. More specifically, in our approach, the encoding cost per submatrix is $\mathcal{O}(k \cdot m/k \cdot d) = \mathcal{O}(md)$, and the total decoding cost is $\mathcal{O}(k^3 + k^2 \cdot m/k) = \mathcal{O}(k^3 + mk)$. We can see that the encoding cost is high, but the encoding can been done before any computation and just once, which can be used for computing $\mathbf{Ax}$ as many times as possible with different $\mathbf{x}$. Meanwhile, the decoding cost is also high when k is large, but it is independent of d, the number of columns of $\mathbf{A}$. Thus, when d is very large, the decoding cost at the master node can be much lower than the computation cost at each worker node. In addition, the decoding at the master node can be done in an incremental fashion using Gauss–Jordan elimination, which can further reduce the decoding latency.

Note that the global encoding vector is required by the master node for decoding. To achieve this efficiently, we use a pseudo-random number generator to generate the local encoding vector for each transmitted packet and append the random seed. The number of local results are computed for the packet. Then, the master node can get the global encoding vectors according to (3). In this way, the coefficient overhead is negligible, which is opposite to the traditional RLNC for communication networks.

Remark 1. *Lin et al. [22] have also applied RLNC in distributed training on mobile devices. They used RLNC to create coded data partitions among mobile devices so as to tolerate computational uncertainties, and their main purpose is to reduce the need to exchange data partitions across mobile devices. Differently from [22], the use of RLNC in this paper is for straggler mitigation and packet-loss tolerance in a joint manner, while leveraging the computation and communication capabilities of all worker nodes.*

Remark 2. *Since random linear network coding is performed over the field of real numbers as opposed to a finite field, the entries of generated matrices could be very large numbers, leading the whole computation to be numerically unstable. In fact, this issue is present in any coded distributed computation over the field of real numbers and is not just limited to our approaches. There are two basic approaches to dealing with this issue. One is to use very small coefficients to avoid the emergence of large numbers, which is possible, as the encoding operations are also linear with these coefficients in our proposed approach. This is significantly different from the Reed–Solomon-code/polynomial-code-based approaches which have been widely adopted in coded distributed computation (see, e.g., [11,23]), as the coefficients are powers of evaluation points. In particular, the numerical instability issue for the RLNC approach is much less severe than that for Reed–Solomon-code/polynomial-code-based approaches, since Vandermonde matrices have exponentially large condition numbers. The other is to employ the finite field embedding technique [24,25],*

where the entries are quantized into number of finite digits and then embedded into a finite field. Nevertheless, both approaches incur numerical errors. How to guarantee numerical stability in coded distributed computation is still an open problem and requires further study.

3.2. Latency Analysis

Let $r_i = \frac{1}{\theta_i + 1/\mu_i}$, and $r'_i = \min\{1/\tau_i, r_i(1-\varepsilon_i)\}$. Define

$$T_0 = \frac{k}{\sum_{i=1}^{n} r'_i} + \frac{\sum_{i=1}^{n} r'_i X_i}{\sum_{i=1}^{n} r'_i}. \tag{5}$$

The following result characterizes a upper bound of the latency of the proposed RLNC-based approach.

Theorem 1. *For any constant $\delta > 0$, the latency of the proposed RLNC-based approach, denoted by T_{RLNC}, satisfies*

$$\lim_{k\to\infty} \Pr(T_{\text{RLNC}} \leq (1+\delta)T_0) = 1. \tag{6}$$

The following result establishes a lower bound on the latency of any scheme under the coding framework.

Theorem 2. *For any scheme under the coding framework, the probability that its latency T_{any} is less than T_0 decays exponentially with k; i.e., for any constant $\delta > 0$, there exists some constant $\eta > 1$ that does not depend on k, such that*

$$\Pr(T_{\text{any}} \geq (1-\delta)T_0) = 1 - \mathcal{O}(\eta^{-k}). \tag{7}$$

From Theorems 1 and 2, it is straightforward to see that the proposed RLNC-based approach is asymptotically optimal. In the following, we will formally prove Theorems 1 and 2 by a connectivity analysis of a continuous-time trellis, which models the computation and communication processes.

For any scheme under the coding framework, as illustrated in Figure 2, we model the computation and communication processes of each worker node w_i up to time t using a continuous-time trellis ($G_i^{(t)}$) [26], where edges are classified into three types: computation edges, transmission edges and memory edges. Each computation edge models the computation of a submatrix-vector product. Suppose w_i computes a submatrix-vector product from time t_0 to $t_0 + \tau_i \leq t$. Then, two nodes, $w_i(t_0)$ and $w'_i(t_0 + \tau_i)$, will be introduced, and there is a computation edge from $w_i(t_0)$ to $w'_i(t_0 + \tau_i)$. Similarly, suppose a packet is transmitted from w_i at time t_0 and received successfully by the master node at time $t_1 \leq t$. Then, two nodes $w'_i(t_0)$ and $m(t_1)$, if they do not exist, will be introduced, and there is a transmission edge from $w'_i(t_0)$ to $m_i(t_1)$. We also introduce nodes $w_i(0)$ and a node $m_i(t)$. Nodes $\{w_i(\cdot)\}$ are connected through the timeline, so are nodes $\{w'_i(\cdot)\}$ and nodes $\{m_i(\cdot)\}$. The edges for such connections are called memory edges. Each computation edge and each transmission edge is associated with unit capacity, and each memory edge is associated with an infinity capacity. Finally, we construct a global continuous-time trellis $G^{(t)}$, which includes the union of all $G_i^{(t)}$ and two auxiliary nodes $w(0)$ and $m(t)$. In addition, there is an edge from $w(0)$ to each $w_i(0)$ with an infinity capacity, and there is an edge from each $m_i(t)$ to $m(t)$ with an infinity capacity.

The usefulness of the continuous-time trellis model is summarized in the following result.

Proposition 1. *For any scheme that achieves latency of T, then the maximum flow from $w(0)$ to $m(T)$ in its continuous-time trellis $G^{(T)}$ must be least k. Moreover, for our RLNC approach, if the maximum flow from $w(0)$ to $m(T)$ in its continuous-time trellis $G^{(T)}$ is at least k, then the master node can recover the desired computation result at time T with probability one.*

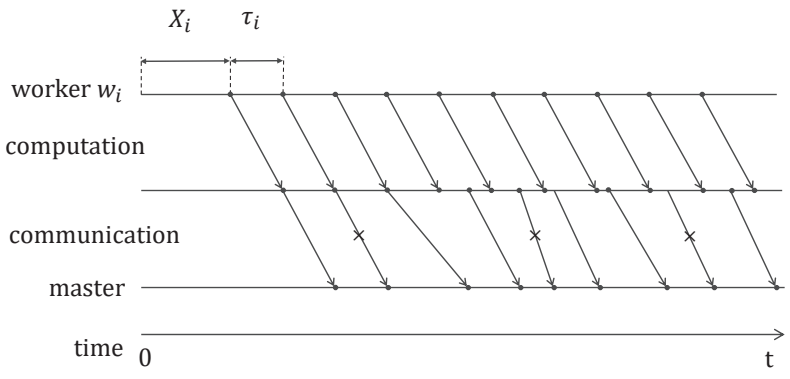


Figure 2. Illustration of a continuous-time trellis, $G_i^{(t)}$.

Proof. It is straightforward to see that the first part holds. The second part is inherited from the optimality of RLNC in communication networks [19] and the fact that all the operations are over the real field $\mathbb{R}$. □

Now, we proceed to prove Theorems 1 and 2. We start by presenting some concentration results regarding the communication between worker nodes and the master node.

Lemma 1. *Suppose $Y_1, Y_2, \ldots$ follow a shifted exponential distribution with rate μ and shift parameter θ independently. Then, for any constant $\delta > 0$, there exists some constant $\eta_1 > 1$, such that*

$$\Pr\left(\left|\sum_{i=1}^{s} Y_i - (\theta + \mu^{-1})s\right| > \delta(\theta + \mu^{-1})s\right) = \mathcal{O}(\eta_1^{-s}). \tag{8}$$

Proof. The result can be proved by a Chernoff-like argument based on moment generating function [27].

The moment generating function of Y_i is

$$\mathbb{E}[e^{-hY_i}] = \frac{\mu}{\mu + h} e^{-h\theta} \tag{9}$$

Hence,

$$\begin{aligned}
&\Pr\left(\sum_{i=1}^{s} Y_i < (1-\delta)(\theta + \mu^{-1})s\right) \\
&= \Pr\left(e^{-h\sum_{i=1}^{s} Y_i} > e^{-h(1-\delta)(\theta + \mu^{-1})s}\right) \\
&\leq \frac{\mathrm{E}\left[e^{-h\sum_{i=1}^{s} Y_i}\right]}{e^{-h(1-\delta)(\theta + \mu^{-1})s}} \\
&= \frac{\prod_{i=1}^{s} \mathrm{E}\left[e^{-hY_i}\right]}{e^{-h(1-\delta)(\theta + \mu^{-1})s}} \\
&= \frac{\left(\frac{\mu}{\mu + h} e^{-h\theta}\right)^s}{e^{-h(1-\delta)(\theta + \mu^{-1})s}}
\end{aligned} \tag{10}$$

where the inequality holds by applying the Markov's inequality. Let $h = \frac{1}{(1-\delta)(\theta + \mu^{-1}) - \theta} - \mu$. We then have

$$\begin{aligned}
&\Pr\left(\sum_{i=1}^{s} Y_i < (1-\delta)(\theta + \mu^{-1})s\right) \\
&\leq \left(\frac{e^{\mu(1-\delta)(\theta+\mu^{-1})-\theta}-1}{\mu(1-\delta)(\theta+\mu^{-1})-\theta}\right)^{-s} \\
&\leq \left(\frac{e^{1-\delta(1+\theta\mu)}-1}{1-\delta(1+\theta\mu)}\right)^{-s}
\end{aligned} \tag{11}$$

By setting $\eta_1 = \frac{e^{1-\delta(1+\theta\mu)}-1}{1-\delta(1+\theta\mu)}$, we get the desired result. □

For a scheme, let $N_i(t)$ ($N_i'(t)$, resp.) be the number of packet transmissions (successful packet transmissions, resp.) from worker node w_i to the master node during the time interval $(X_i, X_i + t)$.

Lemma 2. *For any scheme and any constant $\delta > 0$, there exists some constant $\eta_2 > 1$, such that*

$$\Pr(N_i(t) \geq (1+\delta) r_i t) = \mathcal{O}(\eta_2^{-t}). \tag{12}$$

Proof. Let $Y_1, Y_2, \ldots, Y_{N_i(t)}$ be i.i.d. shifted exponential random variables with rate μ_i and shift parameter θ_i, and $s = \lceil (1+\delta) r_i t \rceil$. According to Lemma 1, there exist some constant $s = \lceil (1+\delta) r_i t \rceil \eta_1 > 1$ and $\eta_2 = \eta_1^{(1+\delta) r_i}$ such that

$$\begin{aligned}
&\Pr(N_i(t) \geq (1+\delta) r_i t) \\
&\leq \Pr\left(\sum_{j=1}^{s} Y_j \leq t\right) \\
&\leq \Pr\left(\sum_{j=1}^{s} Y_j \leq \left(1 - \frac{\delta}{1+\delta}\right)\left(\theta_i + \mu_i^{-1}\right)s\right) \\
&\leq \eta_1^{-s} = \mathcal{O}\left(\eta_2^{-t}\right)
\end{aligned} \tag{13}$$

□

Lemma 3. *For any scheme and any constant $\delta > 0$, there exists some constant $\eta_3 > 1$ such that*

$$\Pr(N_i'(t) \geq (1+\delta) r_i (1-\varepsilon_i) t) = \mathcal{O}(\eta_3^{-t}). \tag{14}$$

Proof. Let A denote the event that $N_i(t) \geq (1 + \delta/2) r_i t$. By the total law of probability,

$$\begin{aligned}
&\Pr\left(N_i'(t) \geq (1+\delta) r_i (1-\varepsilon_i) t\right) \\
&= \Pr\left(N_i'(t) \geq (1+\delta) r_i (1-\varepsilon_i) t \mid A\right) \Pr(A) \\
&\quad + \Pr\left(N_i'(t) \geq (1+\delta) r_i (1-\varepsilon_i) t \mid \bar{A}\right) \Pr(\bar{A}) \\
&\leq \Pr(A) + \Pr\left(N_i'(t) \geq (1+\delta) r_i (1-\varepsilon_i) t \mid \bar{A}\right)
\end{aligned} \tag{15}$$

According to Lemma 2, there exists some constant $\eta_2' > 1$ such that $\Pr(A) = \mathcal{O}\left(\eta_2^{-t}\right)$. Let N be a binomial random variable with parameters $(1 + \delta/2) r_i t$ and $1 - \varepsilon_i$. Then, there exists some constant $\eta_3' > 1$ such that

$$\begin{aligned}
&\Pr\left(N_i'(t) \geq (1+\delta) r_i (1-\varepsilon_i) t \mid \bar{A}\right) \\
&\leq \Pr(N \geq (1+\delta) r_i (1-\varepsilon_i) t) \\
&= \mathcal{O}\left(\eta_3'^{-t}\right)
\end{aligned} \tag{16}$$

where the second step follows by applying the Chernoff bound for a binomial random variable [27]. Finally, by letting $\min\{\eta_3' = \eta_2', \eta_3'\}$, we have

$$\Pr\big(N_i'(t) \geq (1 + \delta) r_i (1 - \varepsilon_i) t\big) = \mathcal{O}\big(\eta_3^{-t}\big) \tag{17}$$

□

Lemma 4. *For any scheme, let $F_i(t)$ be the maximum flow from $w_i(0)$ to $m(t)$ in its continuous-time trellis $G^{(t)}$. Then, for any constant $\delta > 0$, there exists some constant $\eta_4 > 1$ such that*

$$\Pr\left(F_i((1-\delta)T_0) \geq \frac{r_i' k}{\sum_{j=1}^n r_j'}\right) \leq \mathcal{O}(\eta_4^{-k}). \tag{18}$$

Proof. Let B be the event that $(1-\delta)T_0 - X_i > (1-\delta/2)\frac{k}{\sum_{j=1}^n r_j'}$. Then

$$\begin{aligned}
\Pr(B) &\leq \Pr\left((1-\delta)T_0 > (1-\delta/2)\frac{k}{\sum_{i=1}^n r_i'}\right) \\
&= \Pr\left(\sum_{i=1}^n r_i' X_i > \frac{\delta}{2(1-\delta)} k\right) \\
&\leq \Pr\left(\exists i \text{ s. t. } r_i' X_i > \frac{\delta}{2n(1-\delta)} k\right) \\
&\leq \sum_{i=1}^n \Pr\left(r_i' X_i > \frac{\delta}{2n(1-\delta)} k\right) \\
&= \sum_{i=1}^n e^{-\frac{\lambda_i \delta}{2n(1-\delta) r_i'} k} = \mathcal{O}\left(\eta_4'^{-k}\right)
\end{aligned} \tag{19}$$

for some constant $\eta_4' > 1$. By the total law of probability,

$$\begin{aligned}
&\Pr\left(F_i((1-\delta)T_0) \geq \frac{r_i' k}{\sum_{j=1}^n r_j'}\right) \\
&= \Pr\left(F_i((1-\delta)T_0) \geq \frac{r_i' k}{\sum_{j=1}^n r_j'} \mid A\right) \Pr(A) \\
&\quad + \Pr\left(F_i((1-\delta)T_0) \geq \frac{r_i' k}{\sum_{j=1}^n r_j'} \mid \bar{A}\right) \Pr(\bar{A}) \\
&\leq \Pr(A) + \Pr\left(F_i((1-\delta)T_0) \geq \frac{r_i' k}{\sum_{j=1}^n r_j'} \mid \bar{A}\right)
\end{aligned} \tag{20}$$

We consider two cases. In the first case, $\frac{1}{\tau_i} \leq r_i(1-\varepsilon_i)$. Thus, $r_i' = \frac{1}{\tau_i}$. Since $F_i(t)$ cannot exceed the number of computation edges $\frac{t - X_i}{\tau_i}$, it is straightforward to check that

$$\Pr\left(F_i((1-\delta)T_0) \geq \frac{r_i' k}{\sum_{j=1}^n r_j'} \mid \overline{A}\right) = 0. \tag{21}$$

Thus, $\Pr\left(F_i((1-\delta)T_0) \geq \frac{r_i' k}{\sum_{j=1}^n r_j'}\right) = \mathcal{O}(\eta_4'^{-k})$. In the second case, $\frac{1}{\tau_i} > r_i(1-\varepsilon_i)$. Thus, $r_i' = r_i(1-\varepsilon_i)$. Since $F_i((1-\delta)T_0) \leq N_i'((1-\delta)T_0 - X_i)$,

$$\begin{aligned}
&\Pr\left(F_i((1-\delta)T_0) \geq \frac{r_i'k}{\sum_{j=1}^n r_j'} \mid \overline{A}\right) \\
&\leq \Pr\left(N_i'((1-\delta)T_0 - X_i) \geq \frac{r_i'k}{\sum_{j=1}^n r_j'} \mid \overline{A}\right) \\
&\leq \Pr\left(N_i'\left((1-\delta/2)\frac{k}{\sum_{j=1}^n r_j'}\right) \geq \frac{r_i'k}{\sum_{j=1}^n r_j'}\right) \\
&= \mathcal{O}(\eta_5^{-k})
\end{aligned} \tag{22}$$

for some constant $\eta_5 > 1$, where the last step follows from Lemma 3. Thus, we can show that $\Pr\left(F_i((1-\delta)T_0) \geq \frac{r_i'k}{\sum_{j=1}^n r_j'}\right) = \mathcal{O}(\eta_4^{-k})$ for constant $\eta_4 = \min\{\eta_4', \eta_5\}$. □

Now we are ready to prove Theorem 2.

Proof of Theorem 2. For any scheme, since the maximum flow from $w(0)$ to $m((1-\delta)T_0)$ in its continuous-time trellis $G^{((1-\delta)T_0)}$ is equal to $\sum_{i=1}^n F_i((1-\delta)T_0)$, according to Proposition 1, its latency T_{any} satisfies

$$\begin{aligned}
&\Pr(T_{\text{any}} \leq (1-\delta)T_0) \\
&\leq \Pr\left(\sum_{i=1}^n F_i((1-\delta)T_0) \geq k\right) \\
&\leq \Pr\left(\exists i \text{ s.t. } F_i((1-\delta)T_0) \geq \frac{r_i'k}{\sum_{j=1}^n r_j'}\right) \\
&\leq \sum_{i=1}^n \Pr\left(F_i((1-\delta)T_0) \geq \frac{r_i'k}{\sum_{j=1}^n r_j'}\right) \\
&= \mathcal{O}(\eta_4^{-k})
\end{aligned} \tag{23}$$

where the last step follows from Lemma 4. □

Next, we turn to prove Theorem 1. For the RLNC approach and $t \geq X_i$, let $N_i(t, t+\Delta t)$ be the number of successful packet transmissions from worker node w_i to the master node during the time interval $(t, t+\Delta t)$. We have the following result.

Lemma 5. *For any $t \geq X_i$,*

$$\frac{N_i(t, t+\Delta t)}{\Delta t} \xrightarrow{P} r_i(1-\varepsilon_i), \quad \text{as } \Delta t \to \infty; \tag{24}$$

i.e., $N_i(t, t+\Delta t)/\Delta t$ converges to $r_i(1-\varepsilon_i)$ in probability when Δt goes to infinity, or equivalently, for any constant $\epsilon > 0$.

Proof. The result can be shown similarly to that of Lemma 3. □

Lemma 6. *Let $F_i(t)$ be the maximum flow from $w_i(0)$ to $m(t)$ in the continuous-time trellis $G^{(t)}$ of the RLNC approach. Then,*

$$\frac{F_i(t)}{t - X_i} \xrightarrow{P} \min\left\{\frac{1}{\tau_i}, r_i(1-\varepsilon_i)\right\} = r_i', \quad \text{as } t \to \infty, \tag{25}$$

Proof. According to Theorem 1 of [26], Lemma 5 implies this result immediately. □

Now we can prove Theorem 1.

Proof of Theorem 1. According to Lemma 6, $F_i(T_0) \xrightarrow{P} (T_0 - X_i)r'_i$, as $k \to \infty$. Hence, $\sum_{i=1}^{n} F_i(T_0) \xrightarrow{P} k$ as $k \to \infty$. Since,

$$\begin{aligned} &\frac{F_i((1+\delta)T_0)}{F_i(T_0)} \\ &\geq \frac{F_i((1+\delta)T_0)}{(1+\delta)T_0 - X_i} \cdot \frac{(1+\delta)(T_0 - X_i)}{F_i(T_0)} \xrightarrow{P} 1+\delta, \end{aligned} \tag{26}$$

it is straightforward to check that

$$\lim_{k\to\infty} \Pr\left(\sum_{i=1}^{n} F_i((1+\delta)T_0) < k\right) = 0. \tag{27}$$

According to Proposition 1, this implies that

$$\lim_{k\to\infty} \Pr\{T_{\text{RLNC}} \geq (1+\delta)T_0\} = 0. \tag{28}$$

The proof is accomplished. □

4. BATS-Code-Based Approach

As mentioned earlier, despite its optimality, RLNC based approach suffers from its high encoding and decoding overheads. In this section, we propose a new approach based on batched sparse (BATS) code [20], which is a variation of RLNC having low encoding and decoding overheads.

4.1. Description

In the BATS-code-based approach, the k submatrices $\mathbf{A}_1, \ldots, \mathbf{A}_k$ are first encoded into $\mathbf{A}_1, \ldots, \mathbf{A}_k, \mathbf{A}_{k+1}, \ldots, \mathbf{A}_{k'}$ using a fixed-rate systematic erasure code (called a precode), where $k' = (1+\epsilon)k$ and ϵ is a small positive constant (e.g., 0.02). BATS codes are rateless, as an infinite number of batches can be generated. The generation of each batch is as follows:

- Sample a degree *deg* according to a given degree distribution $\Psi = (\Psi_1, \ldots, \Psi_D)$, where D is the maximum degree;
- Select *deg* distinct submatrices uniformly at random from $\mathbf{A}_1, \ldots, \mathbf{A}_k, \mathbf{A}_{k+1}, \ldots, \mathbf{A}_{k'}$;
- Generate M random linear combinations of the *deg* submatrices, which are referred to as a batch.

Based on BATS code, batches of submatrices are assigned to worker nodes, and each worker node performs the local computation on the basis of a batch, which consists of M submatrix-vector multiplications. In order to forward the computational result of a batch to the master node, each worker node will generate a number of packets, each of which is a random linear combination of the M submatrix-vector products corresponding to the batch. For decoding, the master node first recovers $\mathbf{A}_1\mathbf{x}, \ldots, \mathbf{A}_k\mathbf{x}, \mathbf{A}_{k+1}\mathbf{x}, \ldots, \mathbf{A}_{k'}\mathbf{x}$ using Gaussian-elimination-based belief propagation (BP) decoding, and once any k or slightly more than k of $\mathbf{A}_1\mathbf{x}, \ldots, \mathbf{A}_k\mathbf{x}, \mathbf{A}_{k+1}\mathbf{x}, \ldots, \mathbf{A}_{k'}\mathbf{x}$ are recovered, the master node can recover all these $\mathbf{A}_1\mathbf{x}, \ldots, \mathbf{A}_k\mathbf{x}$ by decoding the precode. See [20] for more details.

Overhead: In the BATS-code-based approach, the encoding cost per submatrix is $\mathcal{O}(deg \cdot \frac{m}{k} \cdot d) = \mathcal{O}(\frac{md}{k})$, and the total decoding cost is $\mathcal{O}((M^3 + M^2\frac{m}{k}) \cdot \frac{k}{M}) = \mathcal{O}(M^2k + Mm)$. Clearly, both the encoding cost and decoding cost are much lower than for the RLNC approach, especially when M is a small constant (e.g., 8 or 16). As for the RLNC approach, the decoding cost is independent of d, and the coefficient overhead is negligible when leveraging the pseudo-random-number-generator-based approach.

Remark 3. *There have been many other sparse variants of random linear network coding, including chunked codes (e.g., [28,29]), tunable sparse network coding (e.g., [30,31], and sliding-window coding (e.g., [32–36]). While many of these codes can also be applied, BATS codes are more suitable*

for this distributed computing scenario. On the one hand, BATS codes are rateless. Thus, all the worker nodes can keep on computing and forwarding local results to the master node before the whole computation is completed, as long as enough batches are placed on each worker node. In contrast, chunked codes (e.g., [28,29]) usually have fixed coding rates or require a lot of feedback from the master node. On the other hand, as mentioned in Section 2, in many applications, the step of encoding before computation is required to be performed before the arrival of any input $\mathbf{x}$*. In other words, this encoding step should be irrelevant to the uncertain computation and communication processes of worker nodes. However, differently from BATS codes, sliding-window codes are often generated on-the-fly and are not as suitable as BATS codes.*

4.2. Performance Optimization

The performance of BATS code heavily depends on how the M computation results of each batch are transmitted to the master node, and which degree distribution is used.

Suppose that worker node w_i sends Z_i coded packets to the master nodes for the computation results of each batch B_j. Let $\mathbf{H}_j$ be a $Z_i \times M$ matrix, where each row corresponds to a transmitted packet. If the packet is successfully received by the master node, then the row is the local encoding vector. Otherwise, the row is zero-vector. Let $\mathbf{h}_i = (h_{i,0}, \ldots, h_{i,M})$ denote the rank distribution of $\mathbf{H}_j$, where $h_{i,r}$ is the probability that $\mathbf{H}_j$ has rank r. We can show that

$$h_{i,r} = \begin{cases} \sum_{\ell=r}^{ub} \Pr(Z_i = \ell) \binom{\ell}{r} (1-\varepsilon_i)^r \varepsilon_i^{\ell-r}, & r \leq M-1 \\ \sum_{\ell=M}^{ub} \Pr(Z_i = \ell) \sum_{s=M}^{\ell} \binom{\ell}{s} (1-\varepsilon_i)^s \varepsilon_i^{\ell-s}, & r = M. \end{cases} \tag{29}$$

where ub is an upper bound of Z_i. In order to maximize the transmission efficiency for BATS code, we apply the linear programming method [37] to optimize the distribution of Z_i:

$$\begin{aligned} \max \ & \sum_{r=1}^{M} r h_{i,r} \\ \text{s.t.} \ & \sum_{\ell=0}^{ub} \Pr(Z_i = \ell) \ell \left(\theta_i + \mu_i^{-1} \right) \leq M \tau_i \\ & \sum_{\ell=0}^{ub} \Pr(Z_i = \ell) = 1 \\ & 0 \leq \Pr(Z_i = \ell) \leq 1, \quad \ell = 0, \ldots, ub \end{aligned} \tag{30}$$

Here, the objective is to maximize the expected rank. The first constraint stands for the expected time for transmitting Z_j packets to the master node being no larger than the time for computing M submatrix-vector multiplications, and the last two constraints stand for $\Pr(Z_i = \ell), \ell = 0, \ldots, ub$ being a probability distribution.

When the time goes to infinity, we can see that the proportion of batches whose computation results have been sent to the master node by worker node w_i is $\frac{1/\tau_i}{\sum_{j=1}^{n} 1/\tau_j}$. Hence, we can derive the empirical rank distribution $\mathbf{h}$ over all the batches done by worker nodes as

$$\mathbf{h} = \sum_{i=1}^{n} \frac{1/\tau_i}{\sum_{j=1}^{n} 1/\tau_j} \mathbf{h}_i. \tag{31}$$

Based on the empirical rank distribution, we can find a good degree distribution Ψ such that the BATS code can achieve a coding rate close to $\bar{h}/M$, where $\bar{h}$ is the expected value corresponding to the empirical rank distribution (c.f. [20]).

5. Performance Evaluation

In this section, we first evaluate the decoding cost incurred by our proposed approaches, and then we present simulations conducted to evaluate the overall computational performances of these approaches in comparison to some state-of-the-art approaches.

We first ran some experiments on a computer with an Intel(R) Core(TM) i7-10700 CPU 2.90 GHz and Python 3.7. In these experiments, the matrix $\mathbf{A}$ was $50{,}000 \times d$, where d ranged from 1000 to 16,000. Matrix $\mathbf{A}$ was split into 1000 sub-matrices of the same size, and each submatrix consisted of 50 rows so that each transmitted packet consisted of 50 real numbers. In the BATS-code-based approach, the batch size was set to eight. We simulated the decoding process and evaluated the decoding delays (in terms of second) of both the RLNC based approach and the BATS-code-based approach. The delay for the original matrix multiplication was also evaluated. The results are presented in Table 1.

Table 1. The decoding delays (in terms of second) of our proposed approaches in comparison with the delay of original matrix multiplication.

$d =$	**1000**	**2000**	**4000**	**8000**	**16,000**	**32,000**
matrix multiplication delay	34.16	69.59	138.69	280.86	550.43	1116.84
decoding delay (RLNC)	34.51	34.51	34.51	34.51	34.51	34.51
decoding delay (BATS)	0.54	0.54	0.54	0.54	0.54	0.54

Note that the decoding latencies of both the RLNC based approach and the BATS-code-based approach are irrelevant to d, and the latency for the original matrix multiplication grows linearly with d. From this table, we can see that even when $d = 1000$, the decoding latency of the BATS-code-based approach is only about 1.58% of the latency of original computation, and when d grows larger, this latency becomes negligible. In contrast, when $d = 1000$ or $d = 2000$, the decoding cost of the RLNC based approach is prohibitive.

We also conducted simulations to evaluate the performances of our proposed approaches. In our simulations, the number of worker nodes was 10, and the settings of matrix $\mathbf{A}$ remained the same as above, except that the number of columns d was irrelevant in our simulations. We simulated four scenarios. In the first three scenarios, worker nodes were homogeneous, and the size relationship between computation time per submatrix-vector product and average communication time of a packet varied among these scenarios. In the last scenario, worker nodes were heterogeneous. The involved parameters of these scenarios are given as follows.

- Scenario I, where $(\lambda_i, \tau_i) = (0.1, 0.2)$, $(\mu_i, \theta_i) = (20, 0.05)$ and $\varepsilon_i = 0.2$;
- Scenario II, where $(\lambda_i, \tau_i) = (0.1, 0.15)$, $(\mu_i, \theta_i) = (10, 0.05)$ and $\varepsilon_i = 0.2$;
- Scenario III, where $(\lambda_i, \tau_i) = (0.1, 0.1)$, $(\mu_i, \theta_i) = (10, 0.1)$ and $\varepsilon_i = 0.2$;
- Scenario IV, where for each worker i, parameters λ_i, τ_i, μ_i, θ_i and ε_i were uniformly distributed at random over intervals [0.07, 0.2], [0.1, 0.3], [10, 20], [0.05, 0.2] and [0.1, 0.4], respectively.

For these scenarios, we evaluated the following five methods.

- **Uniform uncoded**, where the divided sub-matrices were equally assigned to 10 worker nodes—i.e., each worker node computed 100 sub-matrices.
- **Two-Replication**, where the divided sub-matrices were equally assigned to five worker nodes, and the computing tasks of these worker nodes were replicated at another five worker nodes.
- $(10, 8)$ **MDS code**, where the divided 1000 sub-matrices were encoded into 1250 sub-matrices and then equally assigned to 10 worker nodes.
- **LT code** [14], where the 1000 original sub-matrices were encoded using LT codes, and an infinite number of coded sub-matrices was assigned to each worker node.
- **RLNC**: The details are introduced in Section 3. The time cost of recoding and decoding operations was ignored.

- **BATS code**: The details are introduced in Section 4, and a batch size of eight was used.

While our proposed schemes tackle the packet-loss issue, the first four of the above schemes do not consider this issue at all. For these schemes, we used an ideal retransmission (IR) scheme for the first four schemes, where the worker nodes know whether a transmitted packet is lost or not immediately. This leads these schemes to perform better. In the following, we refer to the first four schemes as **Uncoded + IR**, **Rep + IR**, **(10,8)MDS + IR** and **LT + IR**, respectively.

The latency performance levels of these approaches under the four scenarios are plotted in Figure 3, where the decoding latency at the master node is ignored. From this figure, we observe the following.

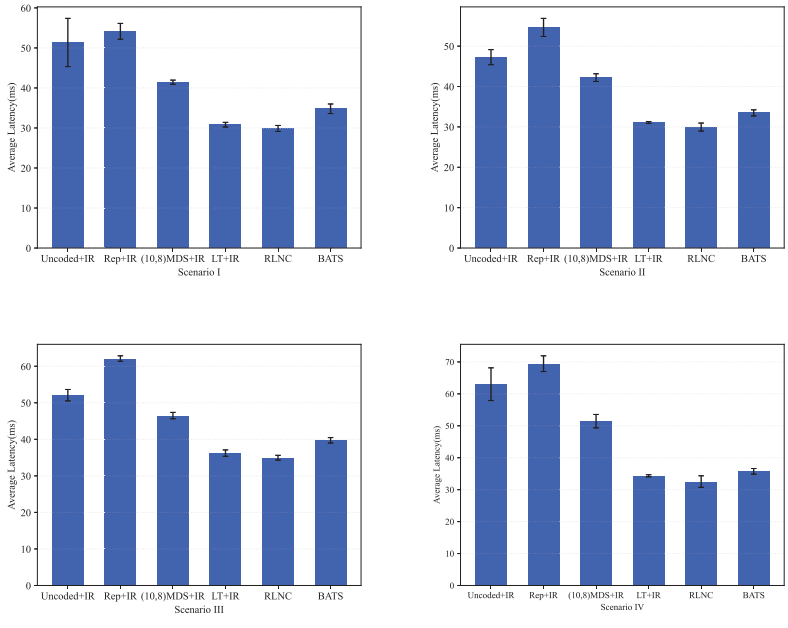


Figure 3. The latency performances of different approaches under four scenarios, where the error bar indicates the standard deviation.

- Among the first four schemes, LT + IR achieved the best performance for all four scenarios. Note that IR eliminates the packet-loss issue, and this result has also been demonstrated in [14], where only the straggler issue was considered. This is because LT codes can achieve near-perfect load balance among the worker nodes in the presence of stragglers.
- For all these scenarios, the proposed RLNC approach achieved the best latency performance among all these schemes. In particular, the performance of the RLNC approach was slightly better than that of LT + IR. Just like LT + IR, our RLNC approach also achieved near-perfect load balance among the worker nodes. Meanwhile, LT + IR incurred a small precode overhead, whereas the RLNC approach did not. This result also demonstrates the near-optimality of the RLNC approach.
- Our BATS approach performed much better than Uncoded + IR, Rep + IR, and (10,8) MDS + IR in all these scenarios, but slightly worse than LT + IR and RLNC. Since LT + IR assumes an ideal retransmission scheme, which is impractical, and the RLNC approach incurs high encoding and decoding costs, the BATS approach is much more practical.

In summary, both our RLNC approach and our BATS approach can overcome both the straggler issue and the packet-loss issue effectively and can achieve near-optimal performance in different scenarios when the number of columns d is large enough.

6. Conclusions

In this paper, we focused on addressing the straggler issue and the packet-loss issue jointly for distributed matrix multiplication in wireless distributed computing systems. We proposed an RLNC approach and proved its asymptotical optimality using a continuous-time-trellis-based argument. We further proposed a more practical variation of the RLNC approach based on BATS code. The effectiveness of both approaches was demonstrated through numerical simulations.

Author Contributions: Methodology, B.F., B.T. and Z.Q.; Validation, B.F.; Formal analysis, B.T.; Writing—original draft, B.F. and B.T.; Writing—review & editing, Z.Q. and B.Y.; Supervision, B.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This paper is supported by the Water Conservancy Project of Jiangsu Province under Grant No. 2021053, the National Natural Science Foundation of China under Grant No. 61872171, the Fundamental Research Funds for the Central Universities under Grant No. B210201053, the Natural Science Foundation of Jiangsu Province under Grant No. BK20190058, and the Future Network Scientific Research Fund Project under Grant No. FNSRFP-2021-ZD-07.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Zhao, S. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* **2021**, *14*, 2179–2217. [CrossRef]
2. Drolia, U.; Guo, K.; Narasimhan, P. Precog: Prefetching for image recognition applications at the edge. In Proceedings of the Second ACM/IEEE Symposium on Edge Computing, San Jose, CA, USA, 12–14 October 2017; pp. 1–13.
3. Datla, D.; Chen, X.; Tsou, T.; Raghunandan, S.; Hasan, S.S.; Reed, J.H.; Kim, J.H. Wireless distributed computing: A survey of research challenges. *IEEE Commun. Mag.* **2012**, *50*, 144–152. [CrossRef]
4. Li, S.; Yu, Q.; Maddah-Ali, M.A.; Avestimehr, A.S. A scalable framework for wireless distributed computing. *IEEE-ACM Trans. Netw.* **2017**, *25*, 2643–2654. [CrossRef]
5. Dean, J.; Barroso, L.A. The tail at scale. *Commun. ACM* **2013**, *56*, 74–80. [CrossRef]
6. Zaharia, M.; Konwinski, A.; Joseph, A.D.; Katz, R.H.; Stoica, I. Improving MapReduce performance in heterogeneous environments. In Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation, San Diego, CA, USA, 8–10 December 2008; pp. 7–21.
7. Wang, D.; Joshi, G.; Wornell, G. Efficient task replication for fast response times in parallel computation. In Proceedings of the 2014 ACM International Conference on Measurement and Modeling of Computer Systems, Austin, TX, USA, 16–20 June 2014; pp. 599–600.
8. Wang, D.; Joshi, G.; Wornell, G. Using straggler replication to reduce latency in large-scale parallel computing. *ACM Sigmetrics Perform. Eval. Rev.* **2015**, *43*, 7–11. [CrossRef]
9. Dean, J.; Ghemawat, S. MapReduce: Simplified data processing on large clusters. *Comm. ACM* **2008**, *51*, 107–113. [CrossRef]
10. Zaharia, M.; Chowdhury, M.; Franklin, M.J.; Shenker, S.; Stoica, I. Spark: Cluster computing with working sets. *HotCloud* **2010**, *10*, 10.
11. Lee, K.; Lam, M.; Pedarsani, R.; Papailiopoulos, D.; Ramchandran, K. Speeding up distributed machine learning using codes. *IEEE Trans. Inf. Theory* **2017**, *64*, 1514–1529. [CrossRef]
12. Ferdinand, N.; Draper, S.C. Hierarchical coded computation. In Proceedings of the 2018 IEEE International Symposium on Information Theory, Vail, CO, USA, 17–22 June 2018; pp. 1620–1624.
13. Kiani, S.; Ferdinand, N.; Draper, S.C. Exploitation of stragglers in coded computation. In Proceedings of the 2018 IEEE International Symposium on Information Theory, Vail, CO, USA, 17–22 June 2018; pp. 1988–1992.
14. Mallick, A.; Chaudhari, M.; Sheth, U.; Palanikumar, G.; Joshi, G. Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication. *Commun. ACM* **2022**, *65*, 111–118. [CrossRef]
15. Luby, M. LT codes. In Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, Vancouver, BC, Canada, 16–19 November 2002; pp. 271–282.
16. Shokrollahi, A. Raptor codes. *IEEE Trans. Inf. Theory* **2006**, *52*, 2551–2567. [CrossRef]
17. Han, D.J.; Sohn, J.Y.; Moon, J. Coded Wireless Distributed Computing With Packet Losses and Retransmissions. *IEEE Trans. Wirel. Commun* **2021**, *20*, 8204–8217. [CrossRef]
18. Ahlswede, R.; Cai, N.; Li, S.Y.; Yeung, R.W. Network information flow. *IEEE Trans. Inf. Theory* **2000**, *46*, 1204–1216. [CrossRef]

19. 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. Inf. Theory* **2006**, *52*, 4413–4430. [CrossRef]
20. Yang, S.; Yeung, R.W. Batched sparse codes. *IEEE Trans. Inf. Theory* **2014**, *60*, 5322–5346. [CrossRef]
21. Park, H.; Lee, K.; Sohn, J.Y.; Suh, C.; Moon, J. Hierarchical coding for distributed computing. In Proceedings of the 2018 IEEE International Symposium on Information Theory, Vail, CO, USA, 17–22 June 2018; pp. 1630–1634.
22. Lin, Z.; Narra, K.G.; Yu, M.; Avestimehr, S.; Annavaram, M. Train where the data is: A case for bandwidth efficient coded training. *arXiv* **2019**, arXiv:1910.10283.
23. Yu, Q.; Maddah-Ali, M.; Avestimehr, S. Polynomial codes: An optimal design for high-dimensional coded matrix multiplication. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
24. Ramamoorthy, A.; Tang, L.; Vontobel, P.O. Universally decodable matrices for distributed matrix-vector multiplication. In Proceedings of the 2019 IEEE International Symposium on Information Theory (ISIT), Paris, France, 7–12 July 2019; pp. 1777–1781.
25. Ramamoorthy, A.; Tang, L. Numerically stable coded matrix computations via circulant and rotation matrix embeddings. *IEEE Trans. Inf. Theory* **2022**, *68*, 2684–2703. [CrossRef]
26. Wu, Y. A trellis connectivity analysis of random linear network coding with buffering. In Proceedings of the IEEE International Symposium on Information Theory, Seattle, WA, USA, 9–14 July 2006; pp. 768–772.
27. Motwani, R.; Raghavan, P. *Randomized Algorithms*; Cambridge University Press: Cambridge, UK, 1995.
28. Tang, B.; Yang, S.; Ye, B.; Yin, Y.; Lu, S. Expander chunked codes. *EURASIP J. Adv. Signal Process.* **2015**, *1*, 106. [CrossRef]
29. Tang, B.; Yang, S. An LDPC approach for chunked network codes. *IEEE ACM Trans. Netw.* **2018**, *26*, 605–617. [CrossRef]
30. Feizi, S.; Lucani, D.E.; Médard, M. Tunable sparse network coding. In Proceedings of the 22th International Zurich Seminar on Communications (IZS), Zürich, Switzerland, 29 February–2 March 2012.
31. Garrido, P.; Sørensen, C.W.; Lucani, D.E.; Agüero, R. Performance and complexity of tunable sparse network coding with gradual growing tuning functions over wireless networks. In Proceedings of the 2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), Valencia, Spain, 4–8 September 2016.
32. Garrido, P.; Gómez, D.; Lanza, J.; Agüero, R. Exploiting sparse coding: A sliding window enhancement of a random linear network coding scheme. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 22–27 May 2016.
33. Wunderlich, S.; Gabriel, F.; Pandi, S.; Fitzek, F.H.; Reisslein, M. Caterpillar RLNC (CRLNC): A practical finite sliding window RLNC approach. *IEEE Access* **2017**, *5*, 20183–20197. [CrossRef]
34. Yang, J.; Shi, Z.P.; Wang, C.X.; Ji, J.B. Design of optimized sliding-window BATS codes. *IEEE Commun. Lett.* **2019**, *23*, 410–413. [CrossRef]
35. Karetsi, F.; Papapetrou, E. Lightweight network-coded ARQ: An approach for ultra-reliable low latency communication. *Comput. Commun.* **2022**, *185*, 118–129. [CrossRef]
36. Tasdemir, E.; Nguyen, V.; Nguyen, G.T.; Fitzek, F.H.; Reisslein, M. FSW: Fulcrum sliding window coding for low-latency communication. *IEEE Access* **2022**, *10*, 54276–54290. [CrossRef]
37. Tang, B.; Yang, S.; Ye, B.; Guo, S.; Lu, S. Near-optimal one-sided scheduling for coded segmented network coding. *IEEE Trans. Comput.* **2015**, *65*, 929–939. [CrossRef]

entropy

MDPI

Article

BAR: Blockwise Adaptive Recoding for Batched Network Coding †

Hoover H. F. Yin [1,2,*], Shenghao Yang [3,*], Qiaoqiao Zhou [4], Lily M. L. Yung [5] and Ka Hei Ng [5]

[1] Department of Information Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong, China
[2] Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China
[3] School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, Shenzhen 518172, China
[4] Department of Computer Science, School of Computing, National University of Singapore, Singapore 119391, Singapore; zhouqq@comp.nus.edu.sg
[5] Independent Researcher, Hong Kong, China; lily@link.cuhk.edu.hk (L.M.L.Y.); kaheicanaan@link.cuhk.edu.hk (K.H.N.)
* Correspondence: hfyin@ie.cuhk.edu.hk (H.H.F.Y.); shyang@cuhk.edu.cn (S.Y.)
† This paper is an extended version of our papers published in Yin, H.H.F.; Yang, S.; Zhou, Q.; Yung, L.M.L. Adaptive Recoding for BATS Codes. In Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT), Barcelona, Spain, 10–15 July 2016; pp. 2349–2353; and Yin, H.H.F.; Ng, K.H. Impact of Packet Loss Rate Estimation on Blockwise Adaptive Recoding for Batched Network Coding. In Proceedings of the 2021 IEEE International Symposium on Information Theory (ISIT), Melbourne, VIC, Australia, 12–20 July 2021; pp. 1415–1420.

Abstract: Multi-hop networks have become popular network topologies in various emerging Internet of Things (IoT) applications. Batched network coding (BNC) is a solution to reliable communications in such networks with packet loss. By grouping packets into small batches and restricting recoding to the packets belonging to the same batch; BNC has much smaller computational and storage requirements at intermediate nodes compared with direct application of random linear network coding. In this paper, we discuss a practical recoding scheme called blockwise adaptive recoding (BAR) which learns the latest channel knowledge from short observations so that BAR can adapt to fluctuations in channel conditions. Due to the low computational power of remote IoT devices, we focus on investigating practical concerns such as how to implement efficient BAR algorithms. We also design and investigate feedback schemes for BAR under imperfect feedback systems. Our numerical evaluations show that BAR has significant throughput gain for small batch sizes compared with existing baseline recoding schemes. More importantly, this gain is insensitive to inaccurate channel knowledge. This encouraging result suggests that BAR is suitable to be used in practice as the exact channel model and its parameters could be unknown and subject to changes from time to time.

Keywords: linear network coding; batched network coding; adaptive recoding

Citation: Yin, H.H.F.; Yang, S.; Zhou, Q.; Yung, L.M.L.; Ng, K.H. BAR: Blockwise Adaptive Recoding for Batched Network Coding. *Entropy* **2023**, *25*, 1054. https://doi.org/10.3390/e25071054

Academic Editor: Boris Ryabko

Received: 31 May 2023
Revised: 23 June 2023
Accepted: 28 June 2023
Published: 13 July 2023

1. Introduction

Noise, interference and congestion are common causes of packet loss in network communications. Usually, a packet has to travel through multiple hops before it can arrive at the destination node. Traditionally, the intermediate nodes apply the store-and-forward strategy. In order to maintain a reliable communication, retransmission is a common practice. A feedback mechanism is applied so that a network node can acknowledge that a packet is lost. However, due to the delay and bandwidth consumption of the feedback packets, retransmission schemes come with a cost of degraded system performance.

In the era of the Internet of things (IoT), there is a diversity of devices and network topologies. Embedded devices or microcomputers have been heavily deployed due to their mobility, lightweight design, and low power consumption. Multi-hop wireless networks

have become a common network topology, highlighting the issues in reliable transmission as wireless links are more vulnerable to packet loss. The packet loss at each link accumulates and the chance of successfully receiving a packet at the destination drops exponentially. Fountain codes, such as in [1–3], can recover the lost packets without the need for feedback because of their ratelessness property. However, the throughput still degrades quickly if there is packet loss at each network link unless link-by-link feedback and retransmission are adopted.

1.1. Network Coding-Based Approaches

Random linear network coding (RLNC) [4,5], a simple realization of network coding [6–8], can achieve the capacity of multi-hop networks with packet loss even without the need for feedback [9,10]. Unfortunately, a direct application of RLNC induces an enormous overhead for the coefficient vectors, as well as high computational and storage costs in network coding operations at intermediate nodes, where the intermediate nodes are usually routers or embedded devices with low computational power and storage space.

Generation-based RLNC was proposed in [11] to resolve these issues. The input packets of the file are partitioned into multiple subsets called the generations, and RLNC is applied to each generation independently. This approach, however, cannot achieve an optimal theoretical rate. Practical concerns and solutions have been further investigated to improve this type of RLNC, such as decoding delay and complexity [12–20], packet size [21–25], and coefficient overhead [26–28].

Instead of partitioning into disjoint subsets, overlapped subsets were investigated in [29–32]. To further reduce the computational costs, the use of RLNC was restricted to small subsets of coded packets generated from the input packets in [33–38]. Example codes used for generating the coded packets include LDPC and fountain codes. This combination of coding theory and network coding is another variant of RLNC called batched network coding (BNC). BATS codes [38,39], a class of BNC, have a close-to-optimal achievable rate where the achievable rate is upper bounded by the expectation of the rank distribution of the batch transfer matrices that model the end-to-end network operations (packet erasures, network coding operations, etc.) on the batches [40]. This hints that the network coding operations, also known as recoding, have an impact on the throughput of BNC.

1.2. Challenges of Recoding in Practice

Baseline recoding is the simplest recoding scheme which generates the same number of recoded packets for every batch. Due to its simple and deterministic structure, baseline recoding appears in many BNC designs and analyses, such as [41–46]. However, the throughput of baseline recoding is not optimal with finite batch sizes [47]. The idea of adaptive recoding, aiming to outperform baseline recoding by generating different numbers of recoded packets for different batches, was proposed in [47] without truly optimizing the numbers. Two adaptive recoding optimization models for independent packet loss channels were then formulated independently in [48,49]. A unified adaptive recoding framework was proposed in [50], subsuming both optimization models and supporting other channel models under certain conditions.

Although adaptive recoding can be applied distributively with local network information, it is a challenge to obtain accurate local information when deploying adaptive recoding in real-world scenarios. Adaptive recoding requires two pieces of information: information distribution remaining in the received batches and the channel condition of the outgoing link.

The first piece of information may change over time if the channel condition of the incoming link varies. One reason for the variation is that the link quality can be affected by interference from users of other networks around the network node. We proposed a simple way to adapt to this variation in [49], grouping a few batches into a block and observing the distribution of received batches in this block. This approach was later called blockwise adaptive recoding (BAR) in [51,52].

The second piece of information may also vary from time to time. In some scenarios, such as deep-space [53–55] and underwater communications [56–58], feedback can be expensive or is not available; therefore, a feedbackless network is preferred. Without feedback, we cannot update our knowledge on the channel condition of the outgoing link. Although we may assume an unchanged channel condition and measure information such as the packet loss rate of the channel beforehand, this measurement, however, can be inaccurate due to observational errors or precision limits.

1.3. Contributions

In this paper, we focus on the practical design of applying BAR in real-world applications. Specifically, we answer the following questions in this paper:

1. How does the block size affect the throughput?
2. Is BAR sensitive to an inaccurate channel condition?
3. How can one calculate the components of BAR and solve the optimization efficiently?
4. How can one make use of link-by-link feedback if it is available?

The first question is related to the trade-off between throughput and latency: a larger block induces a longer delay but gives a higher throughput. We show by numerical evaluations that a small block size can already give a significant throughput gain compared with baseline recoding.

For the second question, we demonstrate that BAR performs very well with an independent packet loss model on channels with dependent packet loss. We also show that BAR is insensitive to an inaccurate packet loss rate. This is an encouraging result as this suggests that it is feasible to apply BAR in real-world applications.

The third question is important in practice as BAR is suppose to run at network nodes, usually routers or IoT devices with limited computational power, but also they may need to handle a huge amount of network traffic. Furthermore, by updating the knowledge of the incoming link from a short observation, we need to recalculate the components of BAR and solve the optimization problem again. In light of this, we want to reduce the number of computations to improve the reaction time and reduce the stress of congestion. We answer this question by proposing an on-demand dynamic programming approach to build the components and some implementation techniques to speed up the algorithm for BAR.

Lastly, for the fourth question, we consider both a perfect feedback system (e.g., the feedback passes through a side-channel with no packet loss) and a lossy feedback system (e.g., the feedback uses the reverse direction of the lossy channel for data transmission). We investigate a few ways to estimate the packet loss rate and show that the throughput can be further boosted by using feedback. Furthermore, a rough estimation is sufficient to catch up the variation in the channel condition. In other words, unless there is another application which requires a more accurate estimation on the packet loss rate, we may consider using an estimation with low computational cost, e.g., the maximum likelihood estimator.

1.4. Paper Organization and Nomenclature

The paper is organized as follows. We first formulate BAR in Section 2. Then, we discuss the implementation techniques for solving BAR efficiently and evaluate the throughput of different block sizes in Section 3. In Section 4, we demonstrate that BAR is insensitive to inaccurate channel models and investigate the use of feedback mechanisms. Lastly, we conclude the paper in Section 5.

Some specific terminology and notations appear frequently throughout the paper. We summarize some of the important terminology and frequently used notations in Tables 1 and 2, respectively.

Table 1. Terminology used for batched network coding (BNC).

Terminology	Description
Batch	A small set of coded packets.
Batch size	The number of coded packets in a batch.
Rank of a batch	A measure of "useful" information (linearly independent packets) retained in the batch.
Expected rank of a batch	The expectation of the rank of the batch at the next network node.
Incoming rank distribution	The distribution of the ranks of the batches arriving at the current network node.
Throughput	The expectation of the rank distribution of the batches arriving at the destination node.
Recoding	The network coding operations restricted to the packets belonging to the same batch.
Recoded packets	The coded packets generated by recoding.
Recoder	The module that performs recoding.
Baseline recoding	A strategy that generates the same number of recoded packets per batch.
Adaptive recoding	A strategy that generates different number of recoded packets per batch.
Block	A set of batches.
Blockwise adaptive recoding	Applying adaptive recoding block by block.

Table 2. Frequently used notations in this paper.

Notation	Description
M	Batch size.
r_b	The rank of the batch b.
t_b	The number of recoded packets for batch b.
$E(r_b, t_b)$	The expected rank of batch b when its rank is r_b at the current node and t_b recoded packets are sent.
$(h_0, \ldots, h_M)$	The incoming rank distribution.
p	The packet loss rate in the independent packet loss model.
$\mathcal{L}$	A block.
$t^{\mathcal{L}}_{\max}$	The total number of recoded packets in block $\mathcal{L}$.
$t^{b}_{\max}$	The maximum number of recoded packets allowed for batch b.
$\mathsf{Binom}(n, p)$	The binomial distribution.
$B_p(t, i)$	The probability mass function of the binomial distribution $\mathsf{Binom}(t, 1-p)$.
$\beta_p(t, r)$	The sum of the first r probability masses of $\mathsf{Binom}(t, 1-p)$.
$\mathsf{Beta}(a, b)$	The beta distribution.
$I_x(a, b)$	The regularized incomplete beta function.

2. Blockwise Adaptive Recoding

In this section, we briefly introduce BNC and then formulate BAR.

2.1. Network Model

As some intermediate nodes may be hardware-implemented routers or not easily reachable for an upgrade, it is not required to deploy a BNC recoder at every intermediate node. The nodes that do not deploy a recoder are transparent to the BNC network as no network coding operations are performed. In the following text, we only consider intermediate nodes that have deployed BNC recoders.

It is not practical to assume every intermediate node knows the information of the whole network; thus, a distributed scheme that only requires local information is desirable. For example, the statistics of the incoming batches, the channel condition of the outgoing link, etc. In a general network, there may be more than one possible outgoing link to reach the destination. We can assign one recoder or one management unit for each outgoing link at an intermediate node [59,60]. In this way, we need a constraint to limit the number of recoded packets of certain batches sent via the outgoing links. The details are discussed in Section 2.4. In other words, we consider each route from the source to the destination separately as a line network.

Line networks are the fundamental building blocks of a general network. Conversely, a recoding scheme for line networks can be extended to general unicast networks and certain multicast networks [38,48]. A line network is a sequence of network nodes where the network links only exist between two neighbouring nodes. An example of a line network is illustrated in Figure 1. In this paper, we only consider line networks in our numerical evaluations.

Figure 1. A three-hop line network. Network links only exist between two neighbouring nodes.

2.2. Batched Network Coding

Suppose we want to send a file from a source node to a destination node through a multi-hop network. The file is divided into multiple input packets, where each packet is regarded as a vector over a fixed finite field. A BNC has three main components: the encoder, the recoder and the decoder.

The encoder of a BNC is applied at the source node to generate batches from the input packets, where each batch consists of a small number of coded packets. Recently, a reinforcement learning approach to optimize the generation of batches was proposed in [61]. Nevertheless, batches are commonly generated using the traditional approach as follows. To generate a batch, the encoder samples a predefined degree distribution to obtain a degree, where the degree is the number of input packets that constitute the batch. Depending on the application, there are various ways to formulate the degree distribution [62–65]. According to the degree, a set of packets is chosen randomly from the input packets. The size of the input packets may be obtained via certain optimizations, such as in [66], to minimize the overhead. Each packet in the batch is formed by taking random linear combinations on the chosen set of packets. The encoder generates M packets per batch, where M is known as the batch size.

Each packet in a batch has a coefficient vector attached to it. Two packets in a batch are defined as linearly independent of each other if and only if their coefficient vectors are linearly independent from each other. Immediately after a batch is generated, the packets within it are assigned as linearly independent from each other. This is accomplished by suitably choosing the initial coefficient vectors [59,67].

A recoder is applied at each intermediate node, performing network coding operations on the received batches to generate recoded packets. This procedure is known as recoding. Some packets of a batch may be lost when they pass through a network link. Each recoded packet of a batch is formed by taking a random linear combination of the received packets in a given batch. The number of recoded packets depends on the recoding scheme. For example, baseline recoding generates the same number of recoded packets for every batch. Optionally, we can also apply a recoder at the source node so that we can have more than M packets per batch at the beginning. After recoding, the recoded packets are sent to the next network node.

At the destination node, a decoder is applied to recover the input packets. Depending on the specific BNC, we can use different decoding algorithms, such as Gaussian elimination, belief propagation and inactivation [68,69].

2.3. Expected Rank Functions

The rank of a batch at a network node is defined by the number of linearly independent packets remaining in the batch, a measure of the amount of information carried by the batch. Adaptive recoding aims to maximize the sum of the expected value of the rank distribution of each batch arriving at the next network node. For simplicity, we called this expected value the expected rank.

For batch b, we denote its rank by r_b and the number of recoded packets to be generated by t_b. The expectation of r_b at the next network node, denoted as $E(r_b, t_b)$, is known as the expected rank function. We have

$$E(r,t) = \sum_{i=0}^{t} \Pr(X_t = i) \sum_{j=0}^{\min\{i,r\}} j\zeta_j^{i,r}, \tag{1}$$

where X_t is the random variable of the number of packets of a batch received by the next network node when we send t packets for this batch at the current node, and $\zeta_j^{i,r}$ is the probability that a batch of rank r at the current node with i received packets at the next network node has rank j at the next network node. The exact formulation of $\zeta_j^{i,r}$ can be found in [38], which is $\zeta_j^{i,r} = \frac{\zeta_j^i \zeta_j^r}{\zeta_j^j q^{(i-j)(r-j)}}$, where q is the field size for the linear algebra operations and $\zeta_j^m = \prod_{k=0}^{j-1}(1-q^{-m+k})$. It is convenient to use $q = 2^8$ in practice as each symbol in this field can be represented by 1 byte. For a sufficiently large field size, say $q = 2^8$, $\zeta_j^{i,r}$ is very close to 1 if $j = \min\{i,r\}$, and is very close to 0 otherwise. That is, we can approximate $\zeta_j^{i,r}$ by $\delta_{j,\min\{i,r\}}$ where $\delta_{\cdot,\cdot}$ is the Kronecker delta. This approximation has also been used in the literature, see, e.g., [45,70–76].

Besides generating all recoded packets by taking random linear combinations, systematic recoding [39,47,59,67], which concerns received packets as recoded packets, can be applied to save computational time. Systematic recoding can achieve a nearly indistinguishable performance compared with methods which generate all recoded packets by taking random linear combinations [39]. Therefore, we can also use (1) to approximate the expected rank functions for systematic recoding accurately.

For the independent packet loss model with packet loss rate p, we have $X_t \sim \mathsf{Binom}(t, 1-p)$, a binomial distribution. If $p = 1$, then a store-and-forward technique can guarantee the maximal expected rank. If $p = 0$, then no matter how many packets we transmit, the next network node must receive no packets. Thus, we assume $0 < p < 1$ in this paper. It is easy to prove that the results in this paper are also valid for $p = 0$ or 1 when we define $0^0 := 1$, which is a convention in combinatorics such that $\mathsf{Binom}(t, 0)$ and $\mathsf{Binom}(t, 1)$ are well-defined with correct interpretation. In the remaining text, we assume $\zeta_j^{i,r} = \delta_{j,\min\{i,r\}}$. That is, for the independent packet loss model, we have

$$E_{\mathsf{indep}}(r,t) = \sum_{i=0}^{t} \binom{t}{i} (1-p)^i p^{t-i} \min\{i,r\}. \tag{2}$$

A demonstration of the accuracy of the approximation $\zeta_j^{i,r} \approx \delta_{j,\min\{i,r\}}$ can be found in Appendix A.

We also consider the expected rank functions for burst packet loss channels modelled by Gilbert–Elliott (GE) models [77,78], where the GE model was also used in other BNC literature such as [52,55,70]. A GE model is a two-state Markov chain, as illustrated in Figure 2. In each state, there is an independent event to decide whether a packet is lost or not. We define $f(s,i,t) := \Pr(S_t = s, X_t = i)$, where S_t is the random variable of the state of the GE model after sending t packets of a batch. By exploiting the structure of the GE model, computation of f can be performed by dynamic programming. Then, we have

$$E_{\mathsf{GE}}(r,t) = \sum_{i=0}^{t} (f(\mathbf{G},i,t) + f(\mathbf{B},i,t)) \min\{i,r\}. \tag{3}$$

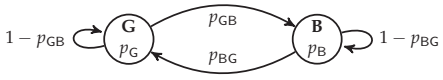


Figure 2. A Gilbert–Elliott (GE) model. In each state, there is an independent event to decide whether a packet is lost or not.

It is easy to see that it would take more steps to compute (3) than (2). Therefore, a natural question to ask is that for burst packet loss channels, is the throughput gap small between adaptive recoding with (2) and (3)? We demonstrate in Section 4.2 that the gap is small so we use (2) any time a nice throughput is received. Therefore, in our investigation we mainly focus on (2).

In the rest of this paper, we refer to $E(r,t)$ as $E_{\mathsf{indep}}(r,t)$ unless otherwise specified. From [50], we know that when the loss pattern follows a stationary stochastic process, the expected rank function $E(r,t)$ is a non-negative, monotonically increasing concave function with respect to t, which is valid for arbitrary field sizes. Further, $E(r,0)=0$ for all r. However, we need to calculate the values of $E(r,t)$ or its supergradients to apply adaptive recoding in practice. To cope with this issue, we first investigate the recursive formula for $E(r,t)$.

We define the probability mass function of the binomial distribution $\mathsf{Binom}(t,1-p)$ by

$$B_p(t,i)=\begin{cases}\binom{t}{i}(1-p)^i p^{t-i} & \text{if } 0\le i\le t,\\ 0 & \text{otherwise.}\end{cases} \tag{4}$$

For integers $r\ge 0$ and $t\ge -1$, we define

$$\beta_p(t,r)=\begin{cases}1 & \text{if } t\le r-1,\\ \sum_{i=0}^{r-1}\binom{t}{i}(1-p)^i p^{t-i}=\sum_{i=0}^{r-1}B_p(t,i) & \text{otherwise.}\end{cases} \tag{5}$$

When $t\ge 0$, the function $\beta_p(t,r)$ is the partial sum of the probability masses of a binomial distribution $\mathsf{Binom}(t,1-p)$. The case where $t=-1$ is used in the approximation scheme in Section 3 and is discussed in that section.

The regularized incomplete beta function, defined as $I_x(a,b):=\frac{\int_0^x t^{a-1}(1-t)^{b-1}\,dt}{\int_0^1 t^{a-1}(1-t)^{b-1}\,dt}$ ([79], Equation 8.17.2), can be used to express the partial sum of the probability masses of a binomial distribution. When $t\ge r>0$, we can apply ([79], Equation 8.17.5) and obtain

$$\beta_p(t,r)=\sum_{i=0}^{r-1}\binom{t}{i}(1-p)^i p^{t-i}=I_p(t-r+1,r). \tag{6}$$

There are different implementations of $I_p(\cdot,\cdot)$ available for different languages. For example, the GNU Scientific Library [80] for C and C++, or the built-in function `betainc` in MATLAB. However, most available implementations consider non-negative real parameters and calculate different queries independently. This consideration is too general for our application, as we only need to query the integral points efficiently. In other words, this formula may be sufficient for prototyping or simulation, but it is not efficient enough for real-time deployment on devices with limited computational power. Nevertheless, this formula is useful for proving the following properties:

Lemma 1. *Assume $0<p<1$. Let Λ be an index set.*

(a) $B_p(t+1,i)=(1-p)B_p(t,i-1)+pB_p(t,i)$ *for* $i=0,1,\dots,t$;
(b) $\beta_p(t+1,r)\le\beta_p(t,r)$ *where the equality holds if and only if* $t+1<r$ *or* $t\ge r=0$;
(c) $\beta_p(t,r)\le\beta_p(t+1,r+1)$ *where the equality holds if and only if* $t<r$;
(d) $\beta_p(t,r+1)\ge\beta_p(t,r)$ *where the equality holds if and only if* $t<r$;
(e) $1\ge\max_{b\in\Lambda}\beta_p(t_b,r_b)\ge\beta_p(t_a+s,r_a)$ *for all* $a\in\Lambda$ *and any non-negative integer* s;

(f) $0 \leq \min_{b\in\Lambda} \beta_p(t_b, r_b) \leq \beta_p(t_a - s, r_a)$ *for all* $a \in \Lambda$ *and any non-negative integer* s *such that* $t_a - s \geq -1$.

Proof. See Appendix B. □

With the notation of $\beta_p(t,r)$, we can now write the recursive formula for $E(r,t)$.

Lemma 2. $E(r, t+1) = E(r,t) + (1-p)\beta_p(t,r)$, *where* t *and* r *are non-negative integers.*

Proof. Let Y_i be independent and identically distributed Bernoulli random variables, where $\Pr(Y_i = 1) = 1 - p$ for all i. When $Y_i = 1$, the i-th packet is received by the next hop.

When we transmit one more packet at the current node, Y_{t+1} indicates whether this packet is received by the next network node or not. If $Y_{t+1} = 0$, i.e., the packet is lost, then the expected rank will not change. If $Y_{t+1} = 1$, then the packet is linearly independent from all the already received packets at the next network node if the number of received packets at the next network node is less than r. That is, the rank of this batch at the next network node increases by 1 if $\sum_{i=1}^{t} Y_i < r$. Therefore, the increment of $E(r,t)$ is $\Pr(Y_{t+1} = 1, \sum_{i=1}^{t} Y_i < r)$. Note that $\sum_{i=1}^{t} Y_i \sim \mathsf{Binom}(t, 1-p)$. As Y_i are all independent and identically distributed, we have $\Pr\big(Y_{t+1} = 1, \sum_{i=1}^{t} Y_i < r\big) = (1-p)\beta_p(t,r)$. □

The formula shown in Lemma 2 can be interpreted as a newly received packet that is linearly independent of all the already received packets with a probability tends to 1 unless the rank has already reached r. This can also be interpreted as $\zeta_j^{i,r} = \delta_{j,\min\{i,r\}}$ with a probability tends to 1. The above lemma can be rewritten in a more useful form as stated below.

Lemma 3. *Let* t *and* r *be non-negative integers.*

(a) $E(r, t+1) = E(r,t) + (1-p)$ *if* $t < r$;

(b) $E(r,t) = (1-p)\sum_{j=0}^{t-1} \beta_p(j,r) = (1-p)\Big(\min\{r,t\} + \sum_{j=r}^{t-1} \beta_p(j,r)\Big)$.

Proof. See Appendix C. □

2.4. Blockwise Adaptive Recoding

The idea of adaptive recoding was presented in [47], and then independently formulated in [48,49]. The former formulation imposes an artificial upper bound on the number of recoded packets and then applies a probabilistic approach to avoid integer programming. The latter investigates the properties of the integer programming problem and proposed efficient algorithms to directly tackle this problem. These two formulations were unified in [50] as a general recoding framework for BNC. This framework requires the distribution of the ranks of the incoming batches, also called the incoming rank distribution. This distribution, however, is not known in advance, and can continually change due to environmental factors. A rank distribution inference approach was proposed in [81], but the long solving time hinders its application in real-time scenarios.

A more direct way to obtain up-to-date statistics is to use the ranks of the few latest batches, a trade-off between a latency of a few batches and the throughput of the whole transmission. This approach was proposed in [49], and later called BAR in [51,52]. In other words, BAR is a recoding scheme which groups batches into blocks and jointly optimizes the number of recoded packets for each batch in the block.

We first describe the adaptive recoding framework and its relation to BAR. We fix an intermediate network node. Let $(h_0, h_1, \ldots, h_M)$ be the incoming rank distribution, t_r the number of recoded packets to be sent for a batch of rank r, and t_{avg} the average number of recoded packets to be sent per batch. The value of t_r is a non-negative real number that is interpreted as follows. Let $\epsilon = t_r - \lfloor t_r \rfloor$ be the fractional part of t_r. There is an ϵ chance to transmit $\lfloor t_r \rfloor + 1$ recoded packets, and a $1 - \epsilon$ chance to transmit $\lfloor t_r \rfloor$ packets. That is, the

fraction is the probability of transmitting one more packet. Similarly, $E(r, t_r)$ is defined as the linear interpolation by $(1-\epsilon)E(r, \lfloor t_r \rfloor) + \epsilon E(r, \lfloor t_r \rfloor + 1)$. The framework maximizes the expected rank of the batches at the next node, which is the optimization problem:

$$\max_{t_r \geq 0, \forall r \in \{0,1,\ldots,M\}} \sum_{r=0}^{M} h_r E(r, t_r) \quad \text{s.t.} \quad \sum_{r=0}^{M} h_r t_r = t_{\text{avg}}. \tag{7}$$

For BAR, the incoming rank distribution is obtained from the recently received few batches. Let a block be a set of batches. We assume that the blocks at a network node are mutually disjoint. Suppose the node receives a block $\mathcal{L}$. For each batch $b \in \mathcal{L}$, let r_b and t_b be the rank of b and the number of recoded packets to be generated for b, respectively. Let $t^{\mathcal{L}}_{\max} = t_{\text{avg}}/|\mathcal{L}|$ be the number of recoded packets to be transmitted for the block $\mathcal{L}$. The batches of the same rank are considered individually with the notations r_b and t_b, and the total number of packets to be transmitted for a block is finite; therefore, we assume t_b for each $b \in \mathcal{L}$ is a non-negative integer, and $t^{\mathcal{L}}_{\max}$ is a positive integer. By dividing both the objective and the constraint of the framework by $|\mathcal{L}|$, we obtain the simplest formulation of BAR:

$$\max_{t_b \in \{0,1,2,\ldots\}, \forall b \in \mathcal{L}} \sum_{b \in \mathcal{L}} E(r_b, t_b) \quad \text{s.t.} \quad \sum_{b \in \mathcal{L}} t_b = t^{\mathcal{L}}_{\max}. \tag{8}$$

To support scenarios with multiple outgoing links for the same batch, e.g., load balancing, we may impose an upper bound on the number of recoded packets per batch. Let $t^b_{\max}$ be a non-negative integer that represents the maximum number of recoded packets allowed to be transmitted for the batch b. This value may depend on the rank of b at the node. Subsequently, we can formulate the following optimization problem based on (8):

$$\begin{aligned} \max_{t_b \in \{0,1,2,\ldots\}, \forall b \in \mathcal{L}} \quad & \sum_{b \in \mathcal{L}} E(r_b, t_b) \\ \text{s.t.} \quad & \sum_{b \in \mathcal{L}} t_b = t^{\mathcal{L}}_{\max} \\ & t_b \leq t^b_{\max}, \forall b \in \mathcal{L}. \end{aligned} \tag{9}$$

Note that we must have $\sum_{b \in \mathcal{L}} t^b_{\max} \geq t^{\mathcal{L}}_{\max}$. In the case where this inequality does not hold, we can include more batches in the block to resolve this issue. When $t^b_{\max}$ is sufficiently large for all $b \in \mathcal{L}$, (9) degenerates into (8).

The above optimization only depends on the local knowledge at the node. The batch rank r_b can be known from the coefficient vectors of the received packets of batch b. As a remark, the value of $t^{\mathcal{L}}_{\max}$ can affect the stability of the packet buffer. For a general network transmission scenario with multiple transmission sessions, the value of $t^{\mathcal{L}}_{\max}$ can be determined by optimizing the utility of certain local network transmissions [82,83].

Though we do not discuss such optimizations in this paper, we consider solving BAR with a general value of $t^{\mathcal{L}}_{\max}$.

On the other hand, note that the solution to (9) may not be unique. We only need to obtain one solution for recoding purpose. In general, (9) is a non-linear integer programming problem. A linear programming variant of (9) can be formulated by using a technique in [81]. However, such a formulation has a huge amount of constraints and requires the values of $E(r_b, t)$ for all $b \in \mathcal{L}$ and all possible t to be calculated beforehand. We defer the discussion of this formulation to Appendix H.

A network node will keep receiving packets until it has received enough batches to form a block $\mathcal{L}$. A packet buffer is used to store the received packets. Then, the node solves (9) to obtain the number of recoded packets for each batch in the block, i.e., $\{t_b\}_{b \in \mathcal{L}}$. The node then generates and transmits t_b-recoded packets for every batch $b \in \mathcal{L}$. At the same time, the network node continually receives new packets. After all the recoded packets for the block $\mathcal{L}$ are transmitted, the node drops the block from its packet buffer and then repeats the procedure by considering another block.

We do not specify the transmission order of the packets. Within the same block, the ordering of packets can be shuffled to combat burst loss, e.g., [43,44,52]. Such shuffling can reduce the burst error length experienced by each batch so that the packet loss events are more "independent" from each other. On the other hand, we do not specify the rate control mechanism, as it should be separated as another module in the system. This can be reflected in BAR by choosing suitable expected rank functions, e.g., modifying the parameters in the GE model. BAR is only responsible for deciding the number of recoded packets per batch.

The size of a block depends on its application. For example, if an interleaver is applied to L batches, we can group the L batches as a block. When $|\mathcal{L}| = 1$, the only solution is $t_b = t^{\mathcal{L}}_{\max}$, which degenerates into baseline recoding. Therefore, we need to use a block size of at least 2 in order to utilize the throughput enhancement of BAR. Intuitively, it is better to optimize (9) with a larger block size. However, the block size is related to the transmission latency as well as the computational and storage burdens at the network nodes. Note that we cannot conclude the exact rank of each batch in a block until the previous network node finishes sending all the packets of this block. That is, we need to wait for the previous network node to send the packets of all the batches in a block until we can solve the optimization problem. Numerical evaluations in Section 3.5 show that $|\mathcal{L}| = 2$ already has obvious advantage over $|\mathcal{L}| = 1$, and it may not be necessary to use a block size larger than eight.

3. Implementation Techniques for Blockwise Adaptive Recoding

In this paper, we focus on the implementation and performance of BAR. Due to the non-linear integer programming structure of (9), we need to make use of certain properties of the model in order to solve it efficiently. The authors of [49] proposed greedy algorithms to solve (9), which were then generalized in [50] to solve (7). The greedy algorithms in [50] have an potential issue when certain probability masses in the incoming rank distribution are too small, as they may take too many iterations to find a feasible solution. The number of iterations is in the order of $\sum_{r=0}^{M} t_r$, depending on the solution to (7). That is, we cannot establish a bound on the time complexity as the incoming rank distribution can be arbitrary.

For BAR, we do not have this issue because the number of recoded packets in a block, $t^{\mathcal{L}}_{\max}$, is fixed.

In this section, we first discuss the greedy algorithm to solve (9) in Section 3.1. Then, we propose an approximation scheme in Section 3.2, and discuss its application to speed up the solver for practical implementations in Section 3.3. The algorithms in Sections 3.1 and 3.3 are similar to that in [50], but they are modified to optimize BAR. Note that the algorithms in [50] are generalized from [49], so the correctness of the aforementioned modified algorithms is inherited directly from the generalized proofs in [50]. For the approximation scheme in Section 3.2, which did not appear in [50], a more detailed discussion is provided in this section.

The algorithms in this section frequently query and compare the values of $(1-p)\beta_p(t,r)$ for different $t \in \{-1, 0, 1, \ldots, t^{\mathcal{L}}_{\max}\}$ and $r \in \{0, 1, 2, \ldots, M\}$. We suppose a lookup table is constructed so that the queries can be performed in $\mathcal{O}(1)$ time. The table is reusable if the packet loss rate of the outgoing link is unchanged. We only consider the subset $\{-1, 0, 1, \ldots, t^{\mathcal{L}}_{\max}\} \times \{0, 1, 2, \ldots, M\}$ of the β_p domain because

1. the maximum rank of a batch is M;
2. any t_b cannot exceed $t^{\mathcal{L}}_{\max}$ as $\sum_{b\in\mathcal{L}} t_b = t^{\mathcal{L}}_{\max}$.

The case $t = -1$ will be used by our approximation scheme so we keep it in the lookup table. We can build the table on-demand by dynamic programming, discussed in Section 3.4.

3.1. Greedy Algorithm

We first discuss the case $t^{\mathcal{L}}_{\max} \leq \sum_{b\in\mathcal{L}} \min\{r_b, t^b_{\max}\}$. This condition means that the value of $t^{\mathcal{L}}_{\max}$ is too small such that the node has just enough or even not enough time to forward the linearly independent packets received. It is trivial that every $\{t_b\}_{b\in\mathcal{L}}$ satisfying $0 \leq t_b \leq \min\{r_b, t^b_{\max}\}$ and $\sum_{b\in\mathcal{L}} t_b = t^{\mathcal{L}}_{\max}$ is a solution to (9), because every such recoded

packet gains $1-p$ to the expected rank by Lemma 3(a),where this gain is maximal according to the definition of $\beta_p(t,r)$.

For $t^{\mathcal{L}}_{\max} > \sum_{b\in\mathcal{L}} \min\{r_b, t^b_{\max}\}$, we can initialize t_b by $\min\{r_b, t^b_{\max}\}$ for every $b \in \mathcal{L}$ as every such recoded packet gains the maximal value $1-p$ to the expected rank. After this, the algorithm chooses the batch that can gain the most expected rank by sending one more recoded packet, and assigns one more recoded packet to it. The correctness is due to the concavity of the expected rank functions.

The above initialization reduces most iterations in the algorithm, as in practice, the difference between the number of recoded packets and the rank of the batch is not huge. Algorithm 1 is the improved greedy algorithm. Unlike the version in [50], the complexity of Algorithm 1 does not depend on the solution.

Algorithm 1: Solver for BAR.

Data: $t^{\mathcal{L}}_{\max}; \{r_b\}_{b\in\mathcal{L}}$
Result: An assignment $\{t^*_b\}_{b\in\mathcal{L}}$ solving (9)
$t \leftarrow t^{\mathcal{L}}_{\max}$; $t_b \leftarrow 0, \forall b \in \mathcal{L}$;
foreach $b \in \mathcal{L}$ **do**
  if $\min\{r_b, t^b_{max}\} \geq t$ **then**
    $t_b \leftarrow t$;
    return The assignment $\{t_b\}_{b\in\mathcal{L}}$;
  else
    $t_b \leftarrow \min\{r_b, t^b_{\max}\}$; $t \leftarrow t - t_b$;
while $t > 0$ **do**
  $T \leftarrow \{b \in \mathcal{L} : t_b = t^b_{\max}\}$;
  $b \leftarrow$ an element in $\arg\max_{b\in\mathcal{L}\setminus T} \beta_p(t_b, r_b)$;
  $t_b \leftarrow t_b + 1$;
  $t \leftarrow t - 1$;
return The assignment $\{t_b\}_{b\in\mathcal{L}}$;

Theorem 1. *Algorithm 1 can be ran in* $\mathcal{O}(|\mathcal{L}| + \max\{0, t^{\mathcal{L}}_{max} - \sum_{b\in\mathcal{L}} \min\{r_b, t^b_{max}\}\} \log|\mathcal{L}|)$ *time.*

Proof. There are totally $\max\{0, t^{\mathcal{L}}_{\max} - \sum_{b\in\mathcal{L}} \min\{r_b, t^b_{\max}\}\}$ iterations in the **while** loop. The query of $\mu = \arg\max_{b\in\mathcal{L}\setminus T} \beta_p(t_b, r_b)$ can be implemented by using a binary heap. The initialization of the heap, i.e., heapify, takes $\mathcal{O}(|\mathcal{L}|)$ time, which can be performed outside the loop. Each query in the loop takes $\mathcal{O}(1)$ time. The update from $\beta_p(t_\mu, r_\mu)$ into $\beta_p(t_\mu + 1, r_\mu)$, if $t_\mu < t^\mu_{\max} - 1$, takes $\mathcal{O}(\log|\mathcal{L}|)$ time. For $t_\mu = t^\mu_{\max} - 1$, we may remove the entry from the heap, taking the same time complexity as the update above. As $\sum_{b\in\mathcal{L}} \min\{r_b, t^b_{\max}\} \geq t^{\mathcal{L}}_{\max}$ by assumption, the algorithm will not query an empty heap. Therefore, the overall time complexity is $\mathcal{O}(|\mathcal{L}| + \max\{0, t^{\mathcal{L}}_{\max} - \sum_{b\in\mathcal{L}} \min\{r_b, t^b_{\max}\}\} \log|\mathcal{L}|)$. □

In the algorithm, we assume that a lookup table for $\beta_p(t,r)$ is pre-computed. The table can be reused unless there is an update on the outgoing channel condition. Nevertheless, we will discuss an efficient way to construct the lookup table in Section 3.4, and the insignificance of the measurement or prediction errors of the loss probability of the outgoing channel in Section 4.

As the query $\arg\max_{b\in\mathcal{L}\setminus T} \beta_p(t_b, r_b)$ is run repeatedly and an update is performed after every query, we can use a binary heap as described in the proof in real implementation. Note that by Lemma 1(b), $\beta_p(t_\mu, r_\mu) \geq \beta_p(t_\mu + 1, r_\mu)$, so the update is a decrease key operation in a max-heap. In other words, a Fibonacci heap [84] cannot benefit from this operation here.

3.2. Equal Opportunity Approximation Scheme

Algorithm 1 increases t_b step by step. From a geometric perspective, the algorithm finds a path from the interior of a compact convex polytope that models the feasible solu-

tions to the facet $\mathcal{H}\colon \sum_{b\in\mathcal{L}} t_b = t^{\mathcal{L}}_{\max}$. If we have a method to move a non-optimal feasible point on $\mathcal{H}$ towards an optimal point, together with a fast and accurate approximation to (8) or (9), then we can combine them to solve (9) faster than using Algorithm 1 directly. This idea is illustrated in Figure 3. A generalized tuning scheme can be found in [50] based on the algorithm in [49]. However, there is no approximation scheme proposed in [50].

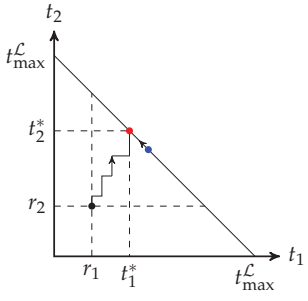


Figure 3. This figure illustrates the idea of modifying the output of an approximation scheme with two batches, where $\mathcal{L} = \{1, 2\}$, $t^{\mathcal{L}}_{\max} \geq r_1 + r_2$ and $t^1_{\max}, t^2_{\max} \geq t^{\mathcal{L}}_{\max}$. The red and blue dots represent the optimal and approximate solutions on the facet $t_1 + t_2 = t^{\mathcal{L}}_{\max}$, respectively. Algorithm 1 starts the search from an interior point (r_1, r_2), while a modification approach starts the search from the blue dot.

We first give an approximation scheme in this subsection. The approximation is based on an observation of the solution for (8) that does not impose an upper boundary on t_b: A batch of higher rank should have more recoded packets transmitted than a batch of lower rank. Unless most $t^b_{\max}$ are too small, the approximation for (8) is also a good approximation for (9).

Theorem 2. *Let $\mathcal{L}$ be a block where $|\mathcal{L}| \geq 2$. If $\{t_b\}_{b\in\mathcal{L}}$ solves* (8) *and $t_b \geq r_b$ for all $b \in \mathcal{L}$, then $t_m < t_n$ for all $m, n \in \mathcal{L}$ such that $r_m < r_n$.*

Proof. See Appendix D. □

As we cannot generate any linearly independent packets for a batch of rank 0, we have $E(0, \cdot) = 0$. Therefore, we can exclude batches of rank 0 from $\mathcal{L}$ before we start the approximation. We define $\mathfrak{L} = \{b \in \mathcal{L}\colon r_b > 0\} \subseteq \mathcal{L}$. When $t^{\mathcal{L}}_{\max} > \sum_{b\in\mathcal{L}} r_b$, we have $t_b \geq r_b$ for all $b \in \mathcal{L}$. An easy way to obtain an approximation is to assign $\{t_b\}_{b\in\mathcal{L}}$ following the guidelines given in Theorem 2 by:

- $t_b = 0$ for all $b \in \mathcal{L} \setminus \mathfrak{L}$;
- $t_b = r_b + \ell$ for all $b \in \mathfrak{L}$.

where $\ell = (t^{\mathcal{L}}_{\max} - \sum_{b\in\mathcal{L}} r_b)/|\mathfrak{L}|$. In the case where ℓ is not an integer, we can round it up for batches with higher ranks and round it down for those with lower ranks.

The above rules allocate the unassigned packets to batches equally after r_b packets have been assigned to each batch b. Thus, we call this approach the equal opportunity approximation scheme. The steps of this scheme are summarized in Algorithm 2.

Note that we do not need to know the packet loss rate p to apply this approximation. That is, if we do not know the value of p, we can still apply this approximation to outperform baseline recoding.

Theorem 3. *Algorithm 2 approximates* (8) *in $\mathcal{O}(|\mathcal{L}|)$ time. If $t^{\mathcal{L}}_{max} \leq \sum_{b\in\mathcal{L}} r_b$, then the algorithm solves* (8).

Algorithm 2: Equal opportunity approximation scheme.

Data: $t^{\mathcal{L}}_{\max}$; $\{r_b\}_{b\in\mathcal{L}}$
Result: An assignment $\{t_b\}_{b\in\mathcal{L}}$ approximating (8)
$\ell \leftarrow t^{\mathcal{L}}_{\max}$; $t_b \leftarrow 0, \forall b \in \mathcal{L}$; $L \leftarrow 0$;
foreach $b \in \mathcal{L}$ **do**
  if $r_b \geq \ell$ **then**
    $t_b \leftarrow \ell$;
    return The assignment $\{t_b\}_{b\in\mathcal{L}}$;
  else if $r_b > 0$ **then**
    $t_b \leftarrow r_b$; $\ell \leftarrow \ell - r_b$; $L \leftarrow L + 1$;
if $L = 0$ **then**
  return An arbitrary feasible solution $\{t_b\}_{b\in\mathcal{L}}$;
$r \leftarrow \ell \bmod L$; $\ell \leftarrow \lfloor \ell / L \rfloor$;
foreach $b \in \mathcal{L}$ *s.t.* $r_b > 0$ **do**
  $t_b \leftarrow r_b + \ell$;
for *the r elements which have the largest* $r_b, b \in \mathcal{L}$ **do**
  $t_b \leftarrow t_b + 1$;
return The assignment $\{t_b\}_{b\in\mathcal{L}}$;

Proof. It is easy to see that Algorithm 2 outputs $\{t_b\}_{b\in\mathcal{L}}$ which satisfies $\sum_{b\in\mathcal{L}} t_b = t^{\mathcal{L}}_{\max}$. That is, the output is a feasible solution of (8). Note that $|\mathfrak{L}| \leq |\mathcal{L}|$, so the assignments and the branches before the last **for** loop take $\mathcal{O}(|\mathcal{L}|)$ time in total. The variable L after the first **foreach** loop equals $|\mathfrak{L}|$. Adding one to the number of recoded packets for $r = \ell \bmod L$ batches with the highest ranks can be performed in $\mathcal{O}(|\mathcal{L}|)$ time. Therefore, the overall running time is $\mathcal{O}(|\mathcal{L}|)$.

If $\mathfrak{L} = \emptyset$, i.e., the whole block is lost, then any feasible $\{t_b\}_{b\in\mathcal{L}}$ is a solution, and the optimal objective value is 0. If $t^{\mathcal{L}}_{\max} \leq \sum_{b\in\mathcal{L}} r_b$, then the algorithm terminates with an output satisfying $t_b \leq r_b$ for all $b \in \mathcal{L}$, which is an optimal solution. □

For the step that adds 1 to the number of recoded packets for $r = \ell \bmod L$ batches with the highest ranks in the algorithm, the worst linear time case can be achieved by using introselect [85] (which is quickselect [86] with a random pivot, but changes to use the median of medians [87] pivot strategy when the complexity grows). We use the selection algorithm to find the r-th largest element, making use of its intermediate steps. During an iteration, one of the following three cases will occur. If the algorithm decides to search a part larger than the pivot, then the discarded part does not contain the largest r elements. If a part smaller than the pivot is selected, then the discarded part is part of the largest r elements. If the pivot is exactly the r-th largest element, then the part larger than the pivot together with the pivot are part of the largest r elements.

In practice, the batch size M is small. We can search these r batches with the highest ranks in $\mathcal{O}(|\mathcal{L}| + M)$ time using a counting technique as an efficient alternative. The technique is to use part of the counting algorithm [88]. We first compute a histogram of the number of times each rank occurs, taking $\mathcal{O}(M)$ time for initialization and $\mathcal{O}(|\mathcal{L}|)$ time to scan the block. Then, we can scan and count the frequencies of the histogram from the highest rank, and eliminate the part where the count exceeds $\ell \bmod |\mathcal{L}|$. This takes $\mathcal{O}(M)$ time. Lastly, we scan the ranks of the batches again in $\mathcal{O}(|\mathcal{L}|)$ time. If included in the modified histogram, we add 1 to the corresponding t_b and minus 1 to the corresponding frequency in the histogram.

Algorithm 2 is a $(1-p)$-approximation algorithm, although the relative performance guarantee factor $1-p$ is not tight in general. However, this suggests that the smaller the packet loss rate p, the more accurate the output the algorithm gives. We defer this discussion to Appendix E.

3.3. Speed-Up via Approximation

In this subsection, we discuss the implementation that corrects an approximate solution to an optimal solution for (9). Algorithm 3 is a greedy algorithm that uses any feasible solution of (8) as a starting point. The **foreach** loop removes the exceeding recoded packets, assigning the released slot to another batch following the iterations in Algorithm 1. This can be regarded as mimicking a replacement of $\beta_p(\cdot,\cdot)$ with the smallest possible value for the batch b that violates the constraint $t_b \le t^b_{\max}$. After this, the intermediate solution is a feasible solution to (9). Then, the last loop finds an increase to the objective by reassigning some slots among the batches.

Algorithm 3: Solver for BAR via approximation.

Data: $t^{\mathcal{L}}_{\max}; r_b, b \in \mathcal{L}$
Result: An assignment $\{t^*_b\}_{b\in\mathcal{L}}$ solving (9)
$t_b \leftarrow 0, \forall b \in \mathcal{L}$;
Run an approximation to get $t_b, b \in \{a \in \mathcal{L} \colon r_a > 0\}$;
$T \leftarrow \{b \in \mathcal{L} \colon t_b \ge t^b_{\max}\}$;
foreach $a \in \mathcal{L}$ *s.t.* $t_a > t^a_{max}$ **do**
  while $t_a > t^a_{max}$ **do**
    $b \leftarrow$ an element in $\arg\max_{b\in\mathcal{L}\setminus T} \beta_p(t_b, r_b)$;
    $t_b \leftarrow t_b + 1$; $t_a \leftarrow t_a - 1$; $T \leftarrow \{a \in \mathcal{L} \colon t_a \ge t^a_{\max}\}$;
while $\min_{a\in\mathcal{L}} \beta_p(t_a - 1, r_a) < \max_{b\in\mathcal{L}\setminus T} \beta_p(t_b, r_b)$ **do**
  $a \leftarrow$ an element in $\arg\min_{a\in\mathcal{L}} \beta_p(t_a - 1, r_a)$;
  $b \leftarrow$ an element in $\arg\max_{b\in\mathcal{L}\setminus T} \beta_p(t_b, r_b)$;
  $t_a \leftarrow t_a - 1$; $t_b \leftarrow t_b + 1$; $T \leftarrow \{a \in \mathcal{L} \colon t_a \ge t^a_{\max}\}$;
return The assignment $\{t_b\}_{b\in\mathcal{L}}$;

Note that the algorithm may query $\beta_p(t_a - 1, r_a)$ for $a \in \mathcal{L}$. If $t_a = 0$, then it has access to the value $\beta_p(-1, r_a)$. Recall that we defined $\beta_p(-1,\cdot) = 1$, the upper bound of $\beta_p(\cdot,\cdot)$ by (10). Therefore, these values act as barriers to prevent outputting a negative number of recoded packets.

Theorem 4. *Let $\{t_b\}_{b\in\mathcal{L}}$ be an approximate solution of* (8) *computed in $\mathcal{O}(T_{approx})$ time. Algorithm 3 can be run in $\mathcal{O}(T_{approx} + |\mathcal{L}| + \sum_{b\in\mathcal{L}} |t^*_b - t_b| \log |\mathcal{L}|)$ time.*

Proof. The assignments before the **foreach** loop takes $\mathcal{O}(T_{\text{approx}} + |\mathcal{L}|)$ time. There are a total of $\sum_{b\in\mathcal{L}} |t^*_b - t_b|/2$ iterations in the loops. The queries for the minimum and maximum values can be implemented using a min-heap and a max-heap, respectively. Similar to Algorithm 1, we can use binary heaps, taking $\mathcal{O}(|\mathcal{L}|)$ initialization time, $\mathcal{O}(1)$ query time, and $\mathcal{O}(\log |\mathcal{L}|)$ update time. Each iteration contains at most two heap queries and four heap updates. The update of the set T can be performed implicitly by setting $\beta_p(t_a, r_a)$ to 0 during the heap updates for $a \in T$. The overall time complexity is then $\mathcal{O}(T_{\text{approx}} + |\mathcal{L}| + \sum_{b\in\mathcal{L}} |t^*_b - t_b| \log |\mathcal{L}|)$. □

In the last **while** loop, we need to query the minimum of $\beta_p(t_a - 1, r_a)$ and the maximum of $\beta_p(t_b, r_b)$. It is clear that we need to decrease the key $\beta_p(t_b, r_b)$ to $\beta_p(t_b + 1, r_b)$ in the max-heap, and increase the key $\beta_p(t_a - 1, r_a)$ to $\beta_p(t_a - 2, r_a)$ in the min-heap. However, we can omit the updates for batches a and b in the max-heap and min-heap, respectively, i.e., reduce from four heap updates to two heap updates. We defer this discussion to Appendix G.

3.4. Construction of the Lookup Table

In the above algorithms, we assume that we have a lookup table for the function $\beta_p(\cdot,\cdot)$ so that we can query its values quickly. In this subsection, we propose an on-demand approach to construct a lookup table by dynamic programming.

Due to the fact that $\sum_{i=0}^{t} B_p(t,i) = 1$, we have

$$0 \le \beta_p(t,r) \le 1. \tag{10}$$

Furthermore, it is easy to see that

$$\beta_p(t,r) = 0 \text{ if and only if } r = 0 \text{ and } t \ge 0; \tag{11}$$
$$\beta_p(t,r) = 1 \text{ if and only if } t \le r-1. \tag{12}$$

A tabular form of β_p is illustrated in Figure 4 after introducing the boundaries 0 and 1 s.

1	1	1	1	1
0	1	1	1	1
0	$\beta_p(1,1)$	1	1	1
0	$\beta_p(2,1)$	$\beta_p(2,2)$	1	1
0	$\beta_p(3,1)$	$\beta_p(3,2)$	$\beta_p(3,3)$	1
0	$\beta_p(4,1)$	$\beta_p(4,2)$	$\beta_p(4,3)$	$\beta_p(4,4)$

Figure 4. The tabular appearance of the function $\beta_p(t,r)$ after introducing boundaries 0 and 1 s. The rows and columns correspond to $t = -1, 0, 1, \ldots$ and $r = 0, 1, 2, \ldots$, respectively. The row above the line is $\beta_p(-1, \cdot)$.

Being a dynamic programming approach, we need the following recursive relations:

$$B_p(t+1,r) = (1-p)B_p(t,r-1) + pB_p(t,r) \quad \text{for } 0 \le r \le t; \tag{13}$$
$$B_p(r,r) = (1-p)B_p(r-1,r-1) \quad \text{for } r > 0; \tag{14}$$
$$\beta_p(t,r) = \beta_p(t,r-1) + B_p(t,r-1) \quad \text{for } 1 < r \le t+1, \tag{15}$$

where (13) is stated in Lemma 1(a); and (14) and (15) are by the definitions of $B_p(t,r)$ and $\beta_p(t,r)$, respectively. The boundary conditions are $B_p(0,0) = 1$, $B_p(i,-1) = 0$, and $\beta_p(i,1) = B_p(i,0)$ for $i = 0, 1, \ldots$. The table can be built in-place in two stages. The first stage fills in $B_p(y, x-1)$ at the (y,x) position of the table. The second stage finishes the table by using (15). Figure 5 illustrates the two stages where the arrows represent the recursive relations (13)–(15). As $\beta_p(0,1) = \beta_p(1,2) = \ldots = 1$, the corresponding entries can be substituted in directly.

1	1	1	1	1
0	$B_p(0,0)$	1	1	1
0	$B_p(1,0)$	$B_p(1,1)$	1	1
0	$B_p(2,0)$	$B_p(2,1)$	$B_p(2,2)$	1
0	$B_p(3,0)$	$B_p(3,1)$	$B_p(3,2)$	$B_p(3,3)$
0	$B_p(4,0)$	$B_p(4,1)$	$B_p(4,2)$	$B_p(4,3)$

(**a**)

1	1	1	1	1
0	$\beta_p(0,1)$	1	1	1
0	$\beta_p(1,1) \to \beta_p(1,2)$		1	1
0	$\beta_p(2,1) \to \beta_p(2,2) \to \beta_p(2,3)$			1
0	$\beta_p(3,1) \to \beta_p(3,2) \to \beta_p(3,3) \to \beta_p(3,4)$			
0	$\beta_p(4,1) \to \beta_p(4,2) \to \beta_p(4,3) \to \beta_p(4,4)$			

(**b**)

Figure 5. The figures illustrate the two stages of the table generation. The indices start from $(-1, 0)$. The first row has the index $y = -1$, which is the row above the line. Compared to Figure 4, $\beta_p(0,1) = \beta_p(1,2) = \ldots = 1$ can be substituted in directly without using the relation (15). (**a**) The first stage of the table generation. The 1 and 0 s paddings are generated first. The solid and dashed arrows represent (13) and (14), respectively. (**b**) The second stage of the table generation. The 1 and 0 s paddings are kept. The arrows represent the recursive relation (15) with the B_p function in-place.

We can compute the values in the table on-demand. Suppose we have $\{t_b\}_{b \in \mathcal{L}}$ in an iteration of Algorithm 1 so that we need the values of $\beta_p(t_b, r_b)$ for $b \in \mathcal{L}$. Let b' be an element in $\arg\max_{b \in \mathcal{L}} r_b$. The table has $r_{b'} + 1$ columns. By the criteria of selecting b in

Algorithm 1 and by Lemmas 1(c) and (d),we have $\max_{b\in\mathcal{L}} t_b = t_{b'}$. From Figure 5, we know we have to calculate all rows of $\beta(t,r)$ for $t \le t_{b'}$. Furthermore, the recursive relations on a row only depend on the previous row; thus, we need to prepare the values of B_p in the next row so that we have the values to compute β_p in the next row. As an example, Figure 6 illustrates the values we have prepared when $t_{b'} = 2$.

1	1	1	1	1
0	$\beta_p(0,1)$	1	1	1
0	$\beta_p(1,1)$	$\beta_p(1,2)$	1	1
0	$\beta_p(2,1)$	$\beta_p(2,2)$	$\beta_p(2,3)$	1
0	$B_p(3,0)$	$B_p(3,1)$	$B_p(3,2)$	$B_p(3,3)$
*	*	*	*	*

Figure 6. The values prepared when $t_{b'} = 2$. The asterisks represent the values that are not yet initialized.

Each entry in the table is modified at most twice during the two stages. Each assignment takes $\mathcal{O}(1)$ time. Therefore, the time and space complexities for building the table are both $\mathcal{O}(MR)$, where R is the number of rows we want to construct. When restricted by the block size, we know that $R \le t^{\mathcal{L}}_{\max}$. The worst case is that we only receive one rank-M batch for the whole block, which is unlikely to occur. In this case, we have the worst case complexity $\mathcal{O}(Mt^{\mathcal{L}}_{\max})$.

Note that we can use fixed-point numbers instead of floating point numbers for a more efficient table construction. Furthermore, the numerical values in the table are not important as long as the orders for any pair of values in the table are the same.

3.5. Throughput Evaluations

We now evaluate the performance of BAR in a feedbackless multi-hop network. Note that baseline recoding is a special case of BAR with block size 1. Our main goal here is to show the throughput gain of BAR among different block sizes. In the evaluation, all (recoded) packets of a batch are sent before sending those of another batch.

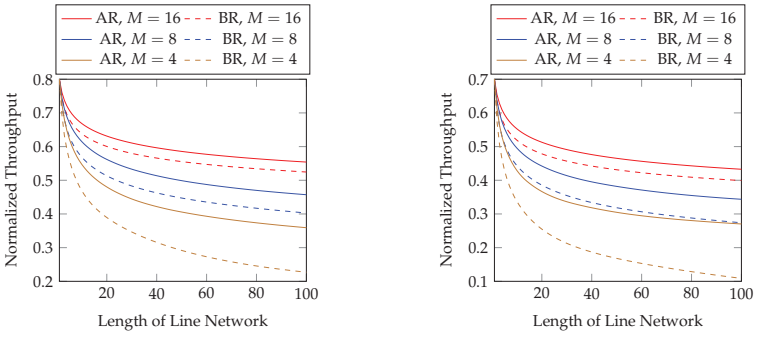
Let $(h_0, h_1, \ldots, h_M)$ be the incoming rank distribution of batches arriving at a network node. The normalized throughput at a network node is defined as the average rank of the received batches divided by the batch size, i.e., $\sum_{i=0}^{M} ih_i/M$. In our evaluations in this subsection, we set $t^{\mathcal{L}}_{\max} = M|\mathcal{L}|$ for every block $\mathcal{L}$. That is, the source node transmits M packets per batch. We assume that every link in the line network has independent packet loss with the same packet loss rate p. In this topology, we set a sufficiently large $t^{b}_{\max}$ for every batch, say, $t^{b}_{\max} = t^{\mathcal{L}}_{\max}$.

We first evaluate the normalized throughput with different batch sizes and packet loss rates. Figure 7 compares adaptive recoding (AR) and baseline recoding (BR) when we know the rank distribution of the batches arriving at each network node before the node applies BAR. In other words, Figure 7 shows the best possible throughput of AR. We compare the effect of block sizes later. We observe that

1. AR has a higher throughput than BR under the same setting;
2. the difference in throughput between AR and BR is larger when the batch size is smaller, the packet loss probability is larger, or the length of the line network is longer.

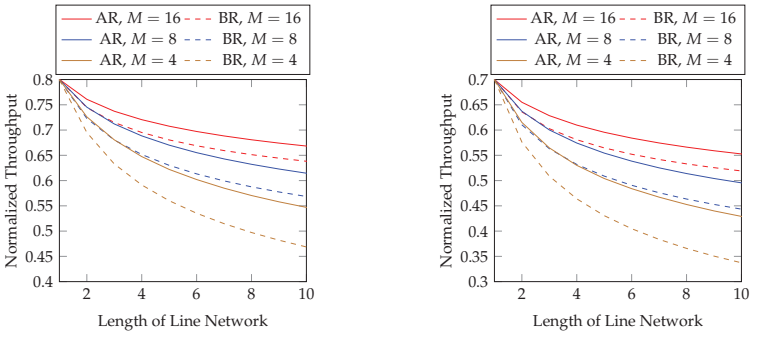
In terms of throughput, the percentage gains of AR over BR using $M = 4$ and $p = 0.2$ are 23.3 and 33.7% at the 20-th and 40-th hops, respectively. They become 43.8 and 70.3%, respectively, when $p = 0.3$.

Although the above figure shows that the throughput of BNC with AR maintains a good performance when the length of the line network is long, many applications use a much shorter line network. We zoom into the figure for the first 10 hops in Figure 8 for practical purposes.



(**a**) $p = 0.2$ (**b**) $p = 0.3$

Figure 7. Adaptive recoding (AR) vs. baseline recoding (BR) in line networks of different lengths, batch sizes and packet loss rates.

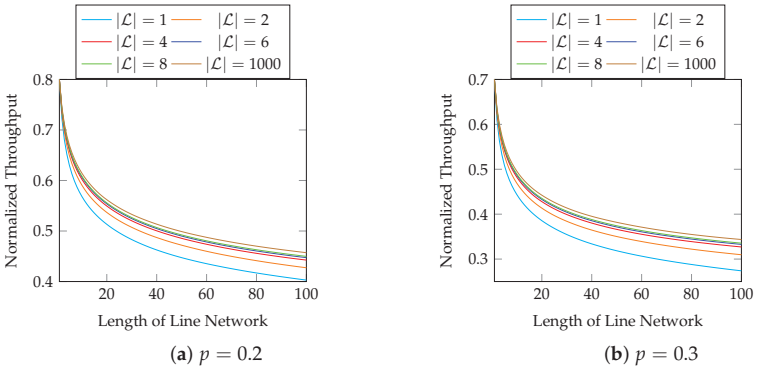


(**a**) $p = 0.2$ (**b**) $p = 0.3$

Figure 8. The first 10 hops in Figure 7.

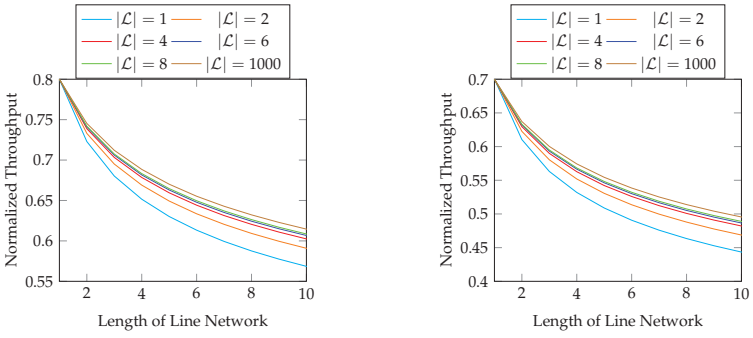
Now, we consider the effect of different block sizes. Figure 9 shows the normalized throughput of different $|\mathcal{L}|$ and p with $M = 8$. The first 10 hops in Figure 9 are zoomed in in Figure 10. We observe that

1. a larger $|\mathcal{L}|$ results a better throughput;
2. using $|\mathcal{L}| = 2$ already gives a much larger throughput than using $|\mathcal{L}| = 1$;
3. using $|\mathcal{L}| > 8$ gives little extra gain in terms of throughput.



(**a**) $p = 0.2$ (**b**) $p = 0.3$

Figure 9. The effect of different block sizes with $M = 8$.

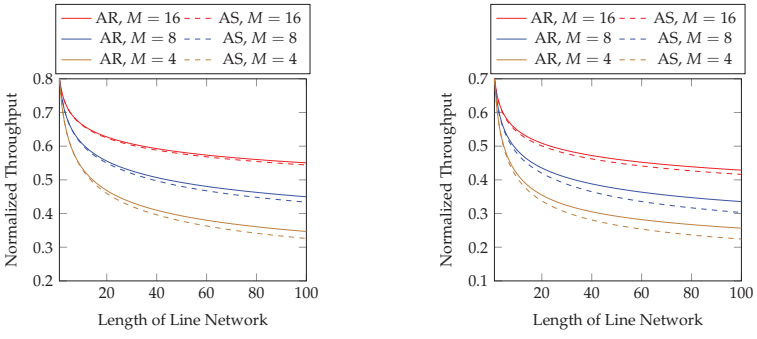


(**a**) $p = 0.2$ (**b**) $p = 0.3$

Figure 10. The first 10 hops in Figure 9.

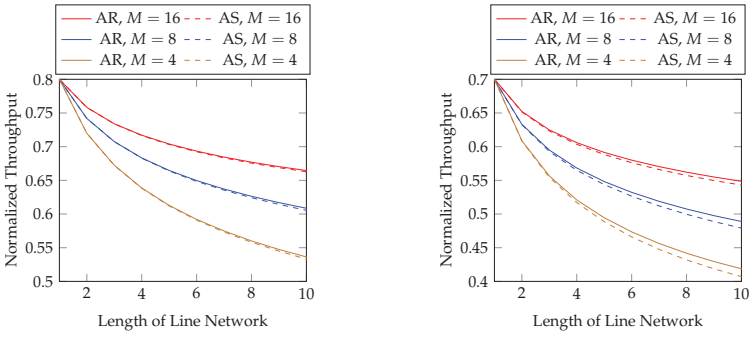
Next, we show the performance of the equal opportunity approximation scheme. Figure 11 compares the normalized throughput achieved by Algorithm 2 (AS) and the true optimal throughput (AR). We compare the best possible throughput of AR here, i.e., the same setting as in Figure 7. The first 10 hops in Figure 11 are zoomed in in Figure 12. We observe that

1. the approximation is close to the optimal solution;
2. the gap in the normalized throughput is smaller when the batch size is larger, the packet loss probability is smaller, or the length of the line network is shorter.



(**a**) $p = 0.2$ (**b**) $p = 0.3$

Figure 11. Approximation vs. optimal AR in line networks of different lengths, batch sizes and packet loss rates.



(**a**) $p = 0.2$ (**b**) $p = 0.3$

Figure 12. The first 10 hops in Figure 11.

4. Impact of Inaccurate Channel Models

In this section, we first demonstrate that the throughput of BAR is insensitive to inaccurate channel models and packet loss rates. Then, we investigate the feedback design and show that although feedback can enhance the throughput, the benefit is insignificant. In other words, BAR works very well without the need of feedback.

4.1. Sensitivity of $\beta_p(t,r)$

We can see that our algorithms only depend on the order of the values of $\beta_p(\cdot,\cdot)$; therefore, it is possible that the optimal $\{t_b\}_{b\in\mathcal{L}}$ for an incorrect p is the same for a correct p. As shown in Figure 4, the boundaries 0 and 1 s are unaffected by $p \in (0,1)$. That is, we only need to investigate the stability of $\beta_p(t,r)$ for $t \geq r > 0$. We calculate values of $\beta_p(t,r)$ corrected to four digital places in Figure 13 for $M = 4$, and $p = 0.1, 0.45$ and their 1% relative changes. We can see that the order of the values are mostly the same when we slightly change p.

0.1000	1	1	1
0.0100	0.1900	1	1
0.0010	0.0280	0.2710	1
0.0001	0.0037	0.0523	0.3439
0.0000	0.0005	0.0086	0.0815
0.0000	0.0001	0.0013	0.0159
0.0000	0.0000	0.0002	0.0027

(**a**) $p = 0.1$.

0.1010	1	1	1
0.0102	0.1918	1	1
0.0010	0.0285	0.2734	1
0.0001	0.0038	0.0533	0.3468
0.0000	0.0005	0.0088	0.0829
0.0000	0.0001	0.0013	0.0163
0.0000	0.0000	0.0002	0.0028

(**b**) $p = 0.101$.

0.0990	1	1	1
0.0098	0.1882	1	1
0.0010	0.0275	0.2686	1
0.0001	0.0036	0.0513	0.3410
0.0000	0.0004	0.0083	0.0800
0.0000	0.0001	0.0012	0.0154
0.0000	0.0000	0.0002	0.0026

(**c**) $p = 0.099$.

0.4500	1	1	1
0.2025	0.6975	1	1
0.0911	0.4252	0.8336	1
0.0410	0.2415	0.6090	0.9085
0.0185	0.1312	0.4069	0.7438
0.0083	0.0692	0.2553	0.5585
0.0037	0.0357	0.1529	0.3917

(**d**) $p = 0.45$.

0.4545	1	1	1
0.2066	0.7024	1	1
0.0939	0.4319	0.8377	1
0.0427	0.2475	0.6163	0.9115
0.0194	0.1358	0.4152	0.7505
0.0088	0.0723	0.2628	0.5676
0.0040	0.0377	0.1589	0.4013

(**e**) $p = 0.4545$.

0.4455	1	1	1
0.1985	0.6925	1	1
0.0884	0.4186	0.8295	1
0.0394	0.2355	0.6016	0.9055
0.0175	0.1268	0.3986	0.7370
0.0078	0.0662	0.2479	0.5494
0.0035	0.0338	0.1471	0.3822

(**f**) $p = 0.4455$.

Figure 13. The values of $\beta_p(t,r)$ for $r = 1,2,3,4$ and $t = 1,2,\ldots$ with different p. The coloured numbers are the largest eight values smaller than 1.

We can also check with the condition number [89] to verify the stability. Roughly speaking, the relative change in the function output is approximately equal to the condition number times the relative change in the function input. A small condition number of $\beta_p(t,r)$ means that the effect of the inaccurate p is small. As shown in Figure 13, the values of $\beta_p(t,r)$ drop quickly when t increases. In the view of the throughput, which is proportional to the sum of these values, we can tolerate a larger relative change, i.e., a larger condition number, when $\beta_p(t,r)$ is small. We calculate condition numbers of $\beta_p(t,r)$ in Figure 14 by the formula stated in Theorem 5.

1.0000	-	-	-
2.0000	0.9474	-	-
3.0000	1.9286	0.8967	-
4.0000	2.9189	1.8585	0.8479
5.0000	3.9130	2.8388	1.7898
6.0000	4.9091	3.8268	2.7596
7.0000	5.9062	4.8187	3.7412

(**a**) $p = 0.1$

1.0000	-	-	-
2.0000	0.7097	-	-
3.0000	1.5714	0.4899	-
4.0000	2.4906	1.2070	0.3296
5.0000	3.4375	2.0325	0.9059
6.0000	4.4000	2.9157	1.6288
7.0000	5.3721	3.8326	2.4384

(**b**) $p = 0.45$

Figure 14. The condition numbers of $\beta_p(t,r)$ for $r = 1,2,3,4$ and $t = 1,2,\ldots$.

Theorem 5. *Let $p \in (0,1)$ and $t \geq r > 0$. The condition number of $\beta_p(t,r)$ with respect to p is $\frac{p^{t-r+1}(1-p)^{r-1}t!}{I_p(t-r+1,r)(t-r)!(r-1)!}$, or equivalently, $\frac{\sum_{j=0}^{r-1}(-1)^j\binom{r-1}{j}p^{t-r+j+1}}{\sum_{j=0}^{r-1}(-1)^j\binom{r-1}{j}p^{t-r+j+1}/(t-r+j+1)}$.*

Proof. See Appendix F. □

4.2. Impact of Inaccurate Channel Models

To demonstrate the impact of an inaccurate channel model, we consider three different channels to present our observations.

- ch1: independent packet loss with constant loss rate $p = 0.45$.
- ch2: burst packet loss modelled by the GE model illustrated in Figure 2 with the parameters used in [70], namely $p_{\mathsf{GB}} = p_{\mathsf{BG}} = p_{\mathsf{G}} = 0.1$, $p_{\mathsf{B}} = 0.8$.
- ch3: independent packet loss with varying loss rates $p = 0.45 + 0.3\sin(2\pi c/1280)$, where c is the number of transmitted batches.

All the three channels have the same average packet loss rate of 0.45. The formula of ch3 is for demonstration purpose only.

We now demonstrate the impact of inaccurate p on the throughput. We consider a line network where all the links use the same channel (ch1, ch2, or ch3). In this topology, we set a sufficiently large $t^b_{\max}$ for every batch, say, $t^b_{\max} = t^{\mathcal{L}}_{\max}$. Similar to the previous evaluation, all (recoded) packets of a batch are sent before sending those of another batch. Furthermore, we set $t^{\mathcal{L}}_{\max} = M|\mathcal{L}|$ for every block $\mathcal{L}$.

In Figure 15 we plot the normalized throughput of the first 80 received blocks at the fourth hop where $|\mathcal{L}| = M = 4$ or 8. We use BAR with (2) for each network although ch2 is a bursty channel. The black curves with BAR are the throughput of BAR where the loss rate is known. For ch1 and ch2, this loss rate p is a constant of 0.45. The red and blue curves are the throughput of BAR when we guess $p = 0.65$ and 0.25, respectively, which is ± 0.2 from the average loss rate of 0.45. As there is no feedback, we do not change our guess on p for these curves. We can see that the throughput is actually very close to the corresponding black curves. This suggests that in the view of the throughput, BAR is not sensitive to p. Even with a wild guess on p, BAR still outperforms BR, as illustrated by the green curves. Regarding ch2, we also plot the orange curve with GE BAR, which is the throughput achieved by BAR with (3). We can see that the gap between the throughput achieved by BAR with (2) and (3) is very small. As a summary of our demonstration:

1. We can use BAR with (2) for bursty channels and the loss in throughput is insignificant.
2. BAR with an inaccurate constant p can achieve a throughput close to the one when we have the exact real-time loss rate.
3. We can see a significant throughput gain from BR by using BAR even with inaccurate channel models.

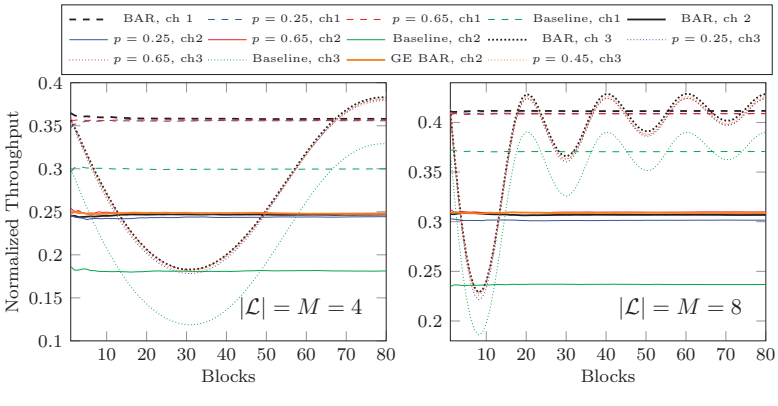


Figure 15. Throughput with inaccurate channel conditions.

4.3. Feedback Design

Although an inaccurate p can give an acceptable throughput, we can further enhance the throughput by adapting the varying p values. To achieve this goal, we need to use feedback.

We adopt a simple feedback strategy which lets the next node return the number of received packets of the batches for the current node to estimate p. Although the next node does not know the number of lost packets per batch, it knows the number of received packets per batch. Therefore, we do not need to introduce more overhead to the transmitted packets by the current node.

When we estimate p, we have to know the number of packets lost during a certain time frame. If the time frame is too small, the estimation is too sensitive so the estimated p changes rapidly and unpredictably. If the time frame is too long, we captured too much out-dated information about the channel so the estimated p changes too slowly and may not be able to adapt to the real loss rate. Recall that the block size is not large as we want to keep the delay small. We use a block as an atomic unit of the time frame. The next node gives feedback on the number of received packets per block. The current node uses the feedback of the blocks in the time frame to estimate p. We perform an estimation of p per received feedback. In this way, the estimated p is the same for each block so that we can apply BAR with (2).

If the feedback is sent via a reliable side channel, then we can assume that the current node can always receive the feedback. However, if the feedback is sent via an unreliable channel, say, the reverse direction of the same channel the data packets were sent from, then we need to consider feedback loss. Let Λ be a set of blocks in a time frame with received feedback. We handle the case of feedback loss by considering the total number of packets transmitted for the blocks in Λ as the total number of packets transmitted during the time frame. In this way, we can also start the estimation before a node sends enough blocks to fill up a time frame. Suppose no feedback is received for every block in a time frame, then we reuse the previously estimated p for BAR.

At the beginning of the transmission, we have no feedback yet so we have no information to estimate p. To outperform BR without the knowledge of p, we can use the approximation of BAR given by Algorithm 2. Once we have received at least one feedback, we can then start estimating p.

4.4. Estimators

Let x and n be the total number of packets received by the current node and the total number of packets transmitted by the previous node, respectively, in a time frame for observation. That is, the number of packets lost in the time frame is $n - x$. We introduce three types of estimators for our numerical evaluations.

(1) Maximum likelihood estimator (MLE): The MLE, denoted by $\hat{p}_{\mathsf{MLE}}$, estimates p by maximizing the likelihood function. $\hat{p}_{\mathsf{MLE}} = (n - x)/n$ is a well-known result which can be obtained via derivative tests. This form collides with the sample average, so by the law of large numbers, $\hat{p}_{\mathsf{MLE}} \to p$ when $n \to \infty$ if p does not change over time.

(2) Minimax estimator: The minimax estimator achieves the smallest maximum risk among all estimators. With the popular mean squared error (MSE) as the risk function, it is a Bayes estimator with respect to the least favourable prior distribution. As studied in [90,91], such prior distribution is a beta distribution $\mathsf{Beta}(\sqrt{n}/2, \sqrt{n}/2)$. The minimax estimator of p, denoted by $\hat{p}_{\mathsf{MM}}$, is the posterior mean, which is $\frac{\sqrt{n}}{1+\sqrt{n}} \frac{n-x}{n} + \frac{1}{1+\sqrt{n}} \frac{1}{2}$, or equivalently, $\frac{n-x+0.5\sqrt{n}}{n+\sqrt{n}}$.

(3) Weighted Bayesian update: Suppose the prior distribution is $\mathsf{Beta}(a, b)$, where the hyperparameters can be interpreted as a pseudo-observation having a successes and b failures. Given a sample of s successes and f failures from a binomial distribution, the posterior distribution is $\mathsf{Beta}(a + s, b + f)$. To fade out the old samples captured by the hyperparameters, we introduce a scaling factor $0 \leq \gamma \leq 1$ and let the posterior distribution be $\mathsf{Beta}(\gamma a + s, \gamma b + f)$. This factor can also prevent the hyperparameters from growing indefinitely. The estimation of p, denoted by $\hat{p}_{\mathsf{Bayes}}$, is the posterior mean with $s = n - x$ and $f = x$, which is $\frac{\gamma a + n - x}{\gamma(a+b)+n}$. To prevent a bias when there are insufficient samples, we

select a non-informative prior as the initial hyperparameters. Specifically, we use the Jeffreys prior, which is $\mathsf{Beta}(1/2, 1/2)$.

We first show the estimation of p by different schemes in Figure 16. We use BAR with (2) and $|\mathcal{L}| = M = 4$. The size of the time frame is W blocks. For $\hat{p}_{\mathsf{MLE}}$ and $\hat{p}_{\mathsf{MM}}$, the observations in the whole time frame have the same weight. For $\hat{p}_{\mathsf{Bayes}}$, the effect of each observation deceases exponentially faster. We consider an observation is out of the time frame when it is scaled into 10% of the original value. That is, we define the scaling factor by $\gamma = \sqrt[W]{0.1}$. In each subplot, the black curve is the real-time p. The red and blue curves are for the estimation without and with feedback loss, respectively. In each case, the two curves are the 25 and 75% percentiles from 1000 runs, respectively.

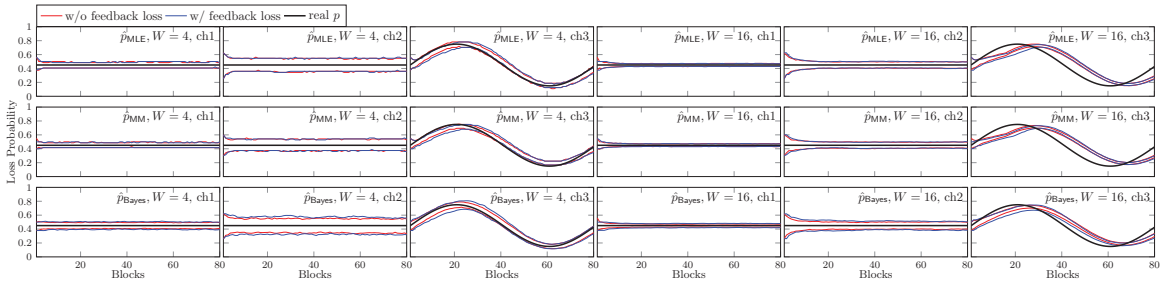


Figure 16. The 25 and 75% percentiles of the estimation of p by different schemes where $|\mathcal{L}| = M = 4$ in 1000 runs.

We can see that a larger W has a slower response to the change in p in ch3. Among the estimators, $\hat{p}_{\mathsf{Bayes}}$ has the fastest response speed as its observations in the time frame are not fairly weighted. Furthermore, although ch1 and ch2 have the same average loss rate, the estimation has a larger variance when the channel is bursty.

4.5. Throughput Evaluations

As discussed in Section 4.2, the guessed p values have an insignificant impact on the throughput. We now show the throughput achieved by the estimation schemes in Figure 17. The parameters for the networks and BAR are the same as in Section 4.2. We do not wildly guess p here so it is no surprise that we can achieve nearly the same throughput as when we know the real p for ch1 and ch2. If we look closely, we can see from Figure 15 that for ch3, there is a small gap between the throughput of BAR when we know the real-time p and the one of BAR when using a constant p. Although the estimation may not be accurate at all times, we can now adapt to the change in p to finally achieve a throughput nearly the same as when we know the real-time p. On the other hand, whether the feedback is lost or not, the plots shown in Figure 17 are basically the same.

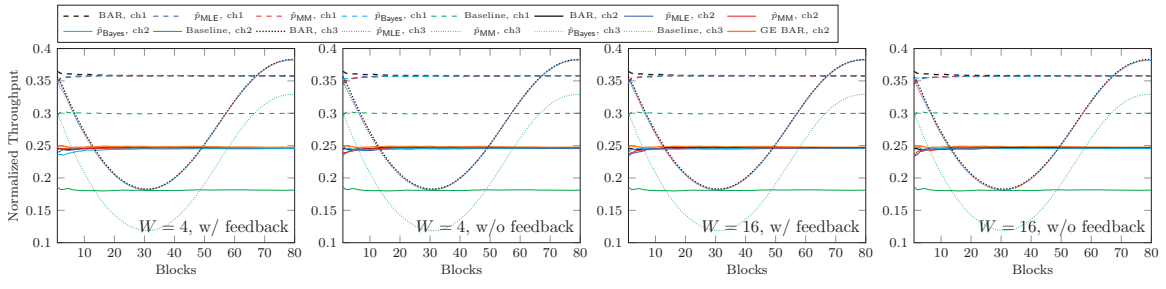


Figure 17. Throughput with estimated p via feedback where $|\mathcal{L}| = M = 4$.

5. Conclusions

We proposed BAR in this paper which can adapt to variations in the incoming channel condition. In a practical perspective, we discussed how to calculate the components of BAR and how to solve BAR efficiently. We also investigated the impact of an inaccurate channel model on the throughput achieved by BAR. Our evaluations showed that

1. BAR is insensitive to the channel model: guessing the loss rate still outperforms BR.
2. For bursty channels, the throughput achieved by BAR with an independent loss model is nearly identical to one with the real channel model. That is, we can use the independent loss model for BAR in practice and apply the techniques in this paper to reduce the computational costs of BAR.
3. Feedback can slightly enhance the throughput for channels with a dynamic loss rate. This suggests that BAR works very well without the need of feedback. On the other hand, feedback loss barely affects the throughput of BAR. Therefore, we can send the feedback through a lossy channel without the need of retransmission. Unless we need to use an accurate estimated loss rate in other applications, we can use MLE with a small time frame for BAR to reduce the computational time.

These encouraging results suggest that BAR is suitable to be deployed in real-world applications.

One drawback of our proposed scheme is that we need to change the default behaviour of some intermediate network nodes, which can be a practical problem in existing networks. In fact, this is a common issue for all network coding schemes. Some routers have hard-wired circuits to efficiently handle heavy traffic, so it is unfeasible to deploy other schemes on them without replacing the hardware. For these heavy-loaded nodes, one may consider producing a hardware to speed up the network coding operations, e.g., [92,93], inducing extra costs on the deployment. On the other hand, the protocol for BNC is not standardized yet, meaning two parties may adopt BAR with incompatible protocols, thus restricting the application of BNC in public networks. However, it is not easy to build a consensus on the protocol, because there are still many research directions to improve the performance of BNC so the protocol design is subject to change in the near future.

6. Patents

The algorithms in Section 3 are variants of those that can be found in the U.S. patent 10,425,192 granted on 24 September 2019 [94]. The linear programming-based algorithm for BAR in Appendix H can be found in the U.S. patent 11,452,003 granted on 20 September 2022 [95].

Author Contributions: Conceptualization, H.H.F.Y. and S.Y.; methodology, H.H.F.Y.; software, H.H.F.Y. and L.M.L.Y.; validation, H.H.F.Y., Q.Z. and K.H.N.; formal analysis, H.H.F.Y.; investigation, H.H.F.Y., S.Y. and Q.Z.; writing—original draft preparation, H.H.F.Y.; writing—review and editing, H.H.F.Y., S.Y. and Q.Z.; visualization, H.H.F.Y.; supervision, H.H.F.Y. and S.Y.; project administration, H.H.F.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 12141108 and 62171399.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Part of the work of Hoover H. F. Yin, Shanghao Yang and Qiaoqiao Zhou was conducted when they were with the Institute of Network Coding, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong. The work of Lily M. L. Yung was performed when she was with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong. The work of Ka Hei Ng was performed when he was with the Department of Physics, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong. Some results in this paper were included in the thesis of Hoover H. F. Yin [96].

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BAR	Blockwise adaptive recoding
IoT	Internet of Things
BNC	Batched network coding
RLNC	Random linear network coding
LDPC	Low-density parity-check code
BATS code	Batched sparse code
GE model	Gilbert–Elliott model
MLE	Maximum likelihood estimator
MSE	Mean squared error

Appendix A. Accuracy of the Approximation $\zeta_j^{i,r} \approx \delta_{j,\min\{i,r\}}$

We demonstrate the accuracy of the approximation $\zeta_j^{i,r} \approx \delta_{j,\min\{i,r\}}$ by showing the percentage error of the expected rank function corrected to three decimal places when $q = 2^8$, $p = 0.2$ and $X_t \sim \mathsf{Binom}(t, 1-p)$ in Table A1. That is, the table shows the values

$$\frac{100\left|\sum_{i=0}^{t}\binom{t}{i}(1-p)^i p^{t-i}(\min\{r,i\} - \sum_{j=0}^{\min\{i,r\}} j\zeta_j^{i,r})\right|}{\sum_{i=0}^{t}\binom{t}{i}(1-p)^i p^{t-i}\sum_{j=0}^{\min\{i,r\}} j\zeta_j^{i,r}}$$

for different r and t.

From the table, we can see that only three pairs of (r,t) have percentage errors larger than 0.1%, where they occur when $r, t \leq 2$. For all the other cases, the percentage errors are less than 0.1%. Therefore, such an approximation is accurate enough for practical applications.

Table A1. Percentage error when approximating expected rank functions.

t	$r=1$	$r=2$	$r=3$	$r=4$	$r=5$	$r=6$	$r=7$	$r=8$	$r=9$	$r=10$	$r=11$	$r=12$	$r=13$	$r=14$	$r=15$	$r=16$
1	0.39216	0.00153	0.00001	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2	0.13140	0.15741	0.00061	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
3	0.03841	0.08032	0.08397	0.00033	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
4	0.01025	0.03091	0.05791	0.05042	0.00020	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
5	0.00258	0.01024	0.02761	0.04398	0.03229	0.00013	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
6	0.00062	0.00308	0.01091	0.02502	0.03416	0.02155	0.00008	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
7	0.00015	0.00087	0.00382	0.01147	0.02258	0.02685	0.01479	0.00006	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
8	0.00003	0.00023	0.00122	0.00457	0.01178	0.02023	0.02123	0.01036	0.00004	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
9	0.00001	0.00006	0.00037	0.00165	0.00526	0.01182	0.01798	0.01686	0.00738	0.00003	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
10	0.00000	0.00002	0.00011	0.00055	0.00210	0.00585	0.01163	0.01585	0.01342	0.00532	0.00002	0.00000	0.00000	0.00000	0.00000	0.00000
11	0.00000	0.00000	0.00003	0.00017	0.00077	0.00257	0.00632	0.01125	0.01388	0.01070	0.00387	0.00002	0.00000	0.00000	0.00000	0.00000
12	0.00000	0.00000	0.00001	0.00005	0.00027	0.00103	0.00302	0.00665	0.01072	0.01208	0.00854	0.00284	0.00001	0.00000	0.00000	0.00000
13	0.00000	0.00000	0.00000	0.00002	0.00009	0.00038	0.00131	0.00344	0.00685	0.01009	0.01046	0.00682	0.00210	0.00001	0.00000	0.00000
14	0.00000	0.00000	0.00000	0.00000	0.00003	0.00013	0.00052	0.00160	0.00381	0.00693	0.00940	0.00901	0.00545	0.00156	0.00001	0.00000
15	0.00000	0.00000	0.00000	0.00000	0.00001	0.00004	0.00020	0.00069	0.00190	0.00412	0.00689	0.00866	0.00773	0.00436	0.00117	0.00000
16	0.00000	0.00000	0.00000	0.00000	0.00000	0.00001	0.00007	0.00028	0.00087	0.00219	0.00436	0.00677	0.00792	0.00660	0.00349	0.00088
17	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00002	0.00010	0.00037	0.00106	0.00246	0.00454	0.00656	0.00718	0.00562	0.00279
18	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00001	0.00004	0.00015	0.00048	0.00126	0.00271	0.00466	0.00629	0.00647	0.00476
19	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00001	0.00006	0.00020	0.00060	0.00147	0.00293	0.00471	0.00598	0.00579
20	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00002	0.00008	0.00027	0.00073	0.00167	0.00311	0.00470	0.00563

Appendix B. Proof of Lemma 1

We have the following properties when a and b are positive integers and $0 \leq x \leq 1$ ([79], Equations 8.17.20 and 8.17.21):

$$I_x(a,b) - I_x(a+1,b) = \binom{a+b-1}{a} x^a (1-x)^b; \tag{A1}$$

$$I_x(a,b+1) - I_x(a,b) = \binom{a+b-1}{b} x^a (1-x)^b. \tag{A2}$$

Proof of Lemma 1(a). It is trivial for $i = 0$. For $i > 0$, recall the recursive formula of binomial coefficients ([79], Equation 1.2.7):

$$\binom{t+1}{i} = \binom{t}{i-1} + \binom{t}{i}, i = 1, 2, \ldots, t.$$

Applying the formula, we have

$$\begin{aligned}
& B_p(t+1,i) \\
&= \binom{t+1}{i}(1-p)^i p^{t+1-i} \\
&= (1-p)\binom{t}{i-1}(1-p)^{i-1}p^{t-(i-1)} + p\binom{t}{i}(1-p)^i p^{t-i} \\
&= (1-p)B_p(t,i-1) + pB_p(t,i).
\end{aligned}$$

□

Proof of Lemma 1(b). Case I: $t < r$. By (10) and (12), $\beta_p(t+1,r) \leq 1 = \beta_p(t,r)$, and the equality holds if and only if $t+1 \leq r-1 < r$.

Case II: $t \geq r > 0$. By (6) and (A1),

$$\begin{aligned}
& \beta_p(t,r) - \beta_p(t+1,r) \\
&= I_p(t-r+1,r) - I_p(t-r+2,r) \\
&= \binom{t}{t-r+1} p^{t-r+1}(1-p)^r \\
&> 0.
\end{aligned}$$

Case III: $t \geq r = 0$. By (11), the equality always hold. □

Proof of Lemma 1(c). Case I: $t < r$. By (12), the equality always hold.

Case II: $t \geq r > 0$. By (6) and (A2),

$$\begin{aligned}
& \beta_p(t+1,r+1) - \beta_p(t,r) \\
&= I_p(t-r+1,r+1) - I_p(t-r+1,r) \\
&= \binom{t}{r} p^{t-r+1}(1-p)^r \\
&> 0.
\end{aligned}$$

Case III: $t \geq r = 0$. By (10) and (11), $\beta_p(t,r) = 0 < \beta_p(t+1,r+1)$. □

Proof of Lemma 1(d). Case I: $t = -1$. By definition, $\beta_p(t,r+1) = \beta_p(t,r) = 1$.

Case II: $t \geq 0$. Recall that $\beta_p(t,r)$ is the partial sum of the probability mass of the binomial distribution $\mathsf{Binom}(t, 1-p)$. By summing one more term, i.e., $\beta_p(t,r+1)$, the partial sum must be larger than or equal to $\beta_p(t,r)$. Note that $B_p(t,i) \neq 0$ when $0 \leq i \leq t$, so the equality holds if and only if $\beta_p(t,r) = 1$ and if $t < r$ by (12). □

Proof of Lemma 1(e) and (f). Inductively by Lemma 1(b), we have

$$\beta_p(t_a + u, r_a) \leq \beta_p(t_a, r_a) \leq \beta_p(t_a - v, r_a) \tag{A3}$$

for all $a \in \Lambda$ where u, v are non-negative integers such that $t_a - v \geq -1$. By (10),

$$0 \leq \min_{b \in \Lambda} \beta_p(t_b, r_b) \leq \beta_p(t_a, r_a) \leq \max_{b \in \Lambda} \beta_p(t_b, r_b) \leq 1 \tag{A4}$$

for all $a \in \Lambda$. Combining (A3) and (A4), the proof is complete. □

Appendix C. Proof of Lemma 3

By Lemma 3(a), we have $E(r, t+1) = E(r, t) + (1-p)\beta_p(t, r)$. If $t < r$, we have

$$\beta_p(t, r) = \sum_{i=0}^{r-1} B_p(t, i) = 1,$$

which proves Lemma 3(a).

For Lemma 3(b), note that we have the initial condition

$$E(r, 0) = B_p(0, 0) \min\{r, 0\} = 0 = (1-p) \sum_{j=0}^{(0)-1} \beta_p(j, r).$$

We can evaluate Lemma 2 recursively and obtain the first equality in Lemma 3(b). By Lemma 3(a), we can show that when $t < r$, we have

$$E(r, t) = t(1-p). \tag{A5}$$

This implies that when $t \geq r$, we have

$$E(r, t) = (1-p)\left(r + \sum_{j=r}^{t-1} \beta_p(j, r)\right). \tag{A6}$$

When $t < r$, the summation term $\sum_{j=r}^{t-1} \beta_p(j, r)$ in (A6) equals 0. So, we can combine (A5) and (A6) and give

$$E(r, t) = (1-p)\left(\min\{r, t\} + \sum_{j=r}^{t-1} \beta_p(j, r)\right).$$

Appendix D. Proof of Theorem 2

Suppose $t_m > t_n$ for some $r_m < r_n$, i.e.,

$$t_m > t_n \geq r_n > r_m. \tag{A7}$$

We define

$$t'_b = \begin{cases} t_m & \text{if } b = n, \\ t_n & \text{if } b = m, \\ t_b & \text{otherwise} \end{cases}$$

for all $b \in \mathcal{L}$. We consider the difference of

$$\begin{aligned} &\sum_{b \in \mathcal{L}} E(r_b, t'_b) - \sum_{b \in \mathcal{L}} E(r_b, t_b) \\ &= [E(r_m, t_n) + E(r_n, t_m)] - [E(r_m, t_m) + E(r_n, t_n)] \\ &= [E(r_n, t_m) - E(r_n, t_n)] + [E(r_m, t_n) - E(r_m, t_m)] \\ &= (1-p)\left[\left(\sum_{j=0}^{t_m-1} \beta_p(j, r_n) - \sum_{j=0}^{t_n-1} \beta_p(j, r_n)\right) + \left(\sum_{j=0}^{t_n-1} \beta_p(j, r_m) - \sum_{j=0}^{t_m-1} \beta_p(j, r_m)\right)\right] \end{aligned} \tag{A8}$$

$$
\begin{aligned}
&= (1-p)\left[\sum_{j=t_n}^{t_m-1}\beta_p(j,r_n) - \sum_{j=t_n}^{t_m-1}\beta_p(j,r_m)\right] \\
&= (1-p)\sum_{j=t_n}^{t_m-1}\left(\beta_p(j,r_n) - \beta_p(j,r_m)\right) \\
&> 0, \qquad \text{(A9)}
\end{aligned}
$$

where

- (A8) follows Lemma 3(b);
- (A9) follows Lemma 1(d) together with (A7).

The above result contradicts that $\{t_b\}_{b\in\mathcal{L}}$ solves (8), which gives us that $t_m \le t_n$ for all $r_m < r_n$.

Next, we suppose $t_m = t_n$ for some $r_m < r_n$, i.e.,

$$t_m = t_n \ge r_n > r_m. \qquad \text{(A10)}$$

We define

$$t''_b = \begin{cases} t_n + 1 & \text{if } b = n, \\ t_m - 1 & \text{if } b = m, \\ t_b & \text{otherwise} \end{cases}$$

for all $b \in \mathcal{L}$. Moreover, we compare the difference of

$$
\begin{aligned}
&\sum_{b\in\mathcal{L}} E(r_i, t''_i) - \sum_{b\in\mathcal{L}} E(r_i, t_i) \\
&= [E(r_n, t_n+1) + E(r_m, t_m-1)] - [E(r_n, t_n) + E(r_m, t_m)] \\
&= [E(r_n, t_n+1) - E(r_n, t_n)] - [E(r_m, t_m) - E(r_m, t_m-1)] \\
&= (1-p)[\beta_p(t_n, r_n) - \beta_p(t_m-1, r_m)] \qquad \text{(A11)} \\
&\ge (1-p)[\beta_p(t_m, r_m+1) - \beta_p(t_m-1, r_m)] \qquad \text{(A12)} \\
&> 0, \qquad \text{(A13)}
\end{aligned}
$$

where

- (A11) follows Lemma 3(a)
- (A12) follows (A10) and Lemma 1(d)
- (A13) follows Lemma 1(c) together with (A10).

This contradicts $\{t_b\}_{b\in\mathcal{L}}$ and solves (8). Therefore, we have $t_m \ne t_n$ for all $r_m < r_n$. Combining the two cases, the proof is complete.

Appendix E. Performance Guarantee and Bounded Error of Algorithm 2

We start the discussion with the following theorem.

Theorem A1. *Let SOL and OPT be the solution given by Algorithm 2 and the optimal solution of* (8)*, respectively, then*

$$
\begin{cases}
SOL \ge (1-p)OPT, \\
OPT - SOL \le (1-p)\sum\limits_{b\in\mathfrak{L}}\sum\limits_{j=r_b+\ell}^{r_b+|\mathfrak{L}|\ell'-1}\beta_p(j,r_b),
\end{cases}
$$

where $\ell' = (t^{\mathcal{L}}_{max} - \sum_{b\in\mathcal{L}} r_b)/|\mathfrak{L}|$ *and* $\ell = \lfloor \ell' \rfloor$.

Proof. We first show that the algorithm has a relative performance guarantee factor of $1-p$. As stated in Theorem 3, when $t^{\mathcal{L}}_{\max} \le \sum_{b\in\mathcal{L}} r_b$, the algorithm guarantees an optimal

solution. Therefore, we only consider $t^{\mathcal{L}}_{\max} > \sum_{b\in\mathcal{L}} r_b$. Let $\{t_b\}_{b\in\mathcal{L}}$ be the approximation given by the algorithm.

Note that any linear combinations of r-independent vectors cannot obtain more than r-independent vectors. Therefore, the expected rank of a batch at the next hop must be no larger than the rank of the batch at the current hop, and, be non-negative. That is,

$$0 \le E(r_b, t) \le r_b, \forall t \ge 0, b \in \mathcal{L}. \tag{A14}$$

This gives a bound of the optimal solution by

$$0 \le \mathrm{OPT} \le \sum_{b\in\mathcal{L}} r_b. \tag{A15}$$

We consider the exact formula of the approximation:

$$\begin{aligned}
\mathrm{SOL} &= (1-p)\sum_{b\in\mathcal{L}} r_b + (1-p)\sum_{b\in\mathcal{L}}\sum_{j=r_b}^{t_b-1}\beta_p(j, r_b) && \text{(A16)}\\
&\ge (1-p)\sum_{b\in\mathcal{L}} r_b && \text{(A17)}\\
&\ge (1-p)\mathrm{OPT}, && \text{(A18)}
\end{aligned}$$

where

- (A16) is stated in Lemma 3(b)
- (A17) holds as $\beta_p(j, r_b) \ge 0$ for all j, r_b, which is by (10);
- (A18) follows the inequality (A15).

Lastly, we show the bounded error. Let $\{t^*_b\}$ be a solution to (8). We write $t^*_b = r_b + \ell_b$ where $\ell_b \ge 0$ for all $b \in \mathcal{L}$. Note that the constraint of (8), i.e., $\sum_{b\in\mathcal{L}} t^*_b = t^{\mathcal{L}}_{\max}$, suggests that

$$\ell_b \le t^{\mathcal{L}}_{\max} - \sum_{b\in\mathcal{L}} r_b = |\mathfrak{L}|\ell'. \tag{A19}$$

On the other hand, it is easy to see that the approximation must either give $t_b = r_b + \ell$ or $t_b = r_b + \ell + 1$. That is, we have $t_b \ge r_b + \ell$. By Lemma 3(b),we have

$$\mathrm{SOL} \ge (1-p)\sum_{b\in\mathfrak{L}}\left[r_b + \sum_{j=r_b}^{r_b+\ell-1}\beta_p(j, r_b)\right]. \tag{A20}$$

We consider the difference between OPT and SOL:

$$\begin{aligned}
&\mathrm{OPT} - \mathrm{SOL}\\
&\le (1-p)\sum_{b\in\mathfrak{L}}\left(\sum_{j=r_b}^{r_b+\ell_b-1}\beta_p(j, r_b) - \sum_{j=r_b}^{r_b+\ell-1}\beta_p(j, r_b)\right) && \text{(A21)}\\
&= (1-p)\sum_{b\in\mathfrak{L}}\left(\sum_{\substack{j=r_b+\ell,\\ \ell_b>\ell}}^{r_b+\ell_b-1}\beta_p(j, r_b) - \sum_{\substack{j=r_b+\ell_b,\\ \ell_b<\ell}}^{r_b+\ell-1}\beta_p(j, r_b)\right)\\
&\le (1-p)\sum_{b\in\mathfrak{L}}\sum_{\substack{j=r_b+\ell,\\ \ell_b>\ell}}^{r_b+\ell_b-1}\beta_p(j, r_b) && \text{(A22)}\\
&\le (1-p)\sum_{b\in\mathfrak{L}}\sum_{j=r_b+\ell}^{r_b+|\mathfrak{L}|\ell'-1}\beta_p(j, r_b), && \text{(A23)}
\end{aligned}$$

where

- (A21) is the difference between the exact form of OPT by Lemma 3(b) after substituting the lower bound of SOL shown in (A20);
- the condition $\ell_b > \ell$ in the summation of (A22) can be removed, as we have $r_b + \ell_b - 1 < r_b + \ell$ if $\ell_b \le \ell$;
- (A23) follows (A19) and the fact shown in (10) that the extra $\beta_p(j, r_b)$ terms are non-negative.

The proof is done. □

If the relative performance guarantee factor of $1 - p$ is tight, we need both equalities in (A17) and (A18) to hold. First, by (10), we know that $\beta_p(j, r_b)$ is always non-negative. The equality in (A17) holds if and only if $\sum_{j=r_b}^{t_b-1} \beta_p(j, r_b) = 0$ for all $b \in \mathcal{L}$. The sum equals 0 only when

- $r_b = 0$ and $t_b \ge 0$ according to (11); or
- $t_b - 1 < r_b$ which forms an empty sum.

The equality in (A18) holds if and only if $\text{OPT} = \sum_{b\in\mathcal{L}} E(r_b, t_b^*) = \sum_{b\in\mathcal{L}} r_b$. Note that (A14) shows that $E(r_b, t_b^*)$ is upper bounded by r_b. This implies that we need $E(r_b, t_b^*) = r_b$ for all $b \in \mathcal{L}$. When $t_b^* \le r_b$, we can apply Lemma 3(a) to obtain $E(r_b, t_b^*) = (1-p)t_b^*$, which equals r_b if and only if $r_b = 0$, as we assumed $0 < p < 1$ in this paper. By Lemma 2, $E(r_b, t)$ is a monotonic increasing function in terms of t for all $r_b \ge 0$. Therefore, when $r_b \ne 0$ we need $t_b^* > r_b$, which implies that $t_{\max}^{\mathcal{L}} > \sum_{b\in\mathcal{L}} r_b$. Then, the approximation will also give $t_b > r_b$ for some $b \in \mathcal{L}$ in this case, and the equality in (A17) does not hold.

That is, we have $\text{SOL} = (1-p)\text{OPT}$ only when $r_b = 0$ for all $b \in \mathcal{L}$. In this case, we have $\text{SOL} = \text{OPT} = 0$. In practice, the probability of having $r_b = 0$ for all $b \in \mathcal{L}$ is very small. Therefore, we can consider that the bound is not tight in most cases but it guarantees that the approximation is good when the packet loss probability is small.

Appendix F. Proof of Theorem 5

Let $\mathbb{B}(a, b; y) := \int_0^y x^{a-1}(1-x)^{b-1}\,dx$ be the incomplete beta function. We have the beta function $\mathbb{B}(a, b) := \mathbb{B}(a, b; 1)$.

From (6), we have $\beta_p(t, r) = I_p(t - r + 1, r) = \frac{\mathbb{B}(t-r+1,r;p)}{\mathbb{B}(t-r+1,r;1)}$. By direct calculation, the condition number is

$$\begin{aligned}
\left|\frac{p\frac{d\beta_p(t,r)}{dp}}{\beta_p(t,r)}\right| &= \left|\frac{p\frac{d}{dp}\int_0^p x^{t-r}(1-x)^{r-1}\,dx}{\mathbb{B}(t-r+1,r;1)I_p(t-r+1,r)}\right| \\
&= \frac{p^{t-r+1}(1-p)^{r-1}}{\mathbb{B}(t-r+1,r;1)I_p(t-r+1,r)} \\
&= \frac{p^{t-r+1}(1-p)^{r-1}}{\int_0^p x^{t-r}(1-x)^{r-1}\,dx} \\
&= \frac{p^{t-r+1}\sum_{j=0}^{r-1}(-1)^j\binom{r-1}{j}p^j}{\int_0^p x^{t-r}\sum_{j=0}^{r-1}(-1)^j\binom{r-1}{j}x^j\,dx} \\
&= \frac{\sum_{j=0}^{r-1}(-1)^j\binom{r-1}{j}p^{t-r+j+1}}{\sum_{j=0}^{r-1}(-1)^j\binom{r-1}{j}\int_0^p x^{t-r+j}\,dx} \\
&= \frac{\sum_{j=0}^{r-1}(-1)^j\binom{r-1}{j}p^{t-r+j+1}}{\sum_{j=0}^{r-1}(-1)^j\binom{r-1}{j}p^{t-r+j+1}/(t-r+j+1)},
\end{aligned} \tag{A24}$$

where the absolute value disappears as both numerator and denominator are non-negative. The first form of the condition number can be obtained by substituting $\mathbb{B}(t-r+1, r; 1) = \frac{(t-r)!(r-1)!}{t!}$ into (A24).

Appendix G. Corrupted Heaps

In this appendix, we explain why we can omit two of the heap updates in Algorithm 3.

Before we start, we need some mathematical descriptions of the optimal solutions of BAR. Then, we will introduce our lazy evaluation technique on a modified heap that we called the corrupted heap.

To simplify the notations, we redefine $\beta_p(t_b, r_b)$ as

$$\beta_p(t,r) = \begin{cases} 0 & \text{if } t_b \geq t^b_{\max}, \\ 1 & \text{if } t_b \leq \min\{r_b, t^b_{\max}\} - 1, \\ \sum_{i=0}^{r_b-1} \binom{t_b}{i}(1-p)^i p^{t_b-i} & \text{otherwise.} \end{cases}$$

in this appendix. In other words, $\beta_p(\cdot,\cdot)$ is now a function of b. When $t_b \geq t^b_{\max}$, $\beta_p(t_b, r_b)$ is the smallest possible value in the image of $\beta_p(\cdot,\cdot)$. Therefore, the algorithms in this paper will not assign more recoded packets to the batch b.

Appendix G.1. Optimality Properties of BAR

First, we introduce the following theorem that states a condition for non-optimality (or optimality after taking contraposition).

Theorem A2. *Let $\{t_b\}_{b\in\mathcal{L}}$ be a feasible solution of* (9)*. Then, $\{t_b\}_{b\in\mathcal{L}}$ is not an optimal solution of* (9) *if and only if there exists two distinct batches κ and ρ with $t_\rho \geq 1$ such that $(1-p)\beta_p(t_\kappa, r_\kappa) > (1-p)\beta_p(t_\rho - 1, r_\rho)$.*

Proof. We first prove the sufficient condition. If $\{t_b\}_{b\in\mathcal{L}}$ does not solve (9), then it means that there exists another configuration $\{t'_b\}_{b\in\mathcal{L}}$ which can give a higher objective value. Since $\sum_{b\in\mathcal{L}} t_b = \sum_{b\in\mathcal{L}} t'_b = t^{\mathcal{L}}_{\max}$, there exists distinct $\kappa, \rho \in \mathcal{L}$ such that $t'_\kappa > t_\kappa$ and $t'_\rho < t_\rho$. Note that $t'_\rho \geq 0$ so we must have $t_\rho \geq 1$. We define

$$\Theta = \{\kappa \colon t'_\kappa > t_\kappa\} \quad \text{and} \quad \Phi = \{\rho \colon t'_\rho < t_\rho\},$$

where

$$\sum_{\theta\in\Theta} (t'_\theta - t_\theta) = \sum_{\phi\in\Phi} (t_\phi - t'_\phi) > 0. \tag{A25}$$

Using the fact that $\{t'_b\}_{b\in\mathcal{L}}$ gives a larger objective value and by Lemma 3(b),we have

$$\sum_{\theta\in\Theta} \sum_{t=t_\theta}^{t'_\theta-1} (1-p)\beta_p(t, r_\theta) > \sum_{\phi\in\Phi} \sum_{t=t'_\phi}^{t_\phi-1} (1-p)\beta_p(t, r_\phi). \tag{A26}$$

Now, we fix κ and ρ such that

$$\kappa \in \arg\max_{\theta\in\Theta} \beta_p(t_\theta, r_\theta) \quad \text{and} \quad \rho \in \arg\min_{\phi\in\Phi} \beta_p(t_\phi - 1, r_\phi).$$

We have

$$\begin{aligned} &\sum_{\theta\in\Theta} (t'_\theta - t_\theta)(1-p)\beta_p(t_\kappa, r_\kappa) \\ &\geq \sum_{\theta\in\Theta} (t'_\theta - t_\theta)(1-p)\beta_p(t_\theta, r_\theta) \\ &\geq \sum_{\theta\in\Theta} \sum_{t=t_\theta}^{t'_\theta-1} (1-p)\beta_p(t, r_\theta) \end{aligned} \tag{A27}$$

$$> \sum_{\phi\in\Phi}\sum_{t=t'_\phi}^{t_\phi-1}(1-p)\beta_p(t,r_\phi) \tag{A28}$$
$$\geq \sum_{\phi\in\Phi}(t_\phi-t'_\phi)(1-p)\beta_p(t_\phi-1,r_\phi) \tag{A29}$$
$$\geq \sum_{\phi\in\Phi}(t_\phi-t'_\phi)(1-p)\beta_p(t_\rho-1,r_\rho),$$

where

- (A27) and (A29) follows Lemma 1(b);
- (A28) is the inequality shown in (A26).

Applying (A25), we have

$$(1-p)\beta_p(t_\kappa,r_\kappa) > (1-p)\beta_p(t_\rho-1,r_\rho),$$

which proves the sufficient condition.

Now we consider the necessary condition, where we have

$$(1-p)\beta_p(t_\kappa,r_\kappa) > (1-p)\beta_p(t_\rho-1,r_\rho) \tag{A30}$$

for some distinct κ and ρ. Let

$$t'_b=\begin{cases} t_\kappa+1 & \text{if } b=\kappa,\\ t_\rho-1 & \text{if } b=\rho,\\ t_b & \text{otherwise}\end{cases}$$

for all $b\in\mathcal{L}$. Then, we consider the following:

$$\begin{aligned}
&\sum_{b\in\mathcal{L}}E(r_b,t'_b)\\
&=\sum_{b\in\mathcal{L}\setminus\{\kappa,\rho\}}E(r_b,t_b)+E(r_\kappa,t_\kappa+1)+E(r_\rho,t_\rho-1)\\
&=\sum_{b\in\mathcal{L}\setminus\{\kappa,\rho\}}E(r_b,t_b)+E(r_\rho,t_\rho-1)+[E(r_\kappa,t_\kappa)+(1-p)\beta_p(t_\kappa,r_\kappa)] &\text{(A31)}\\
&>\sum_{b\in\mathcal{L}\setminus\{\kappa,\rho\}}E(r_b,t_b)+E(r_\kappa,t_\kappa)+[E(r_\rho,t_\rho-1)+(1-p)\beta_p(t_\rho-1,r_\rho)] &\text{(A32)}\\
&=\sum_{b\in\mathcal{L}\setminus\{\rho\}}E(r_b,t_b)+E(r_\rho,t_\rho) &\text{(A33)}\\
&=\sum_{b\in\mathcal{L}}E(r_b,t_b),
\end{aligned}$$

where

- (A31) and (A33) follow Lemma 3(a)
- (A32) follows (A30).

meaning that $\{t_b\}_{b\in\mathcal{L}}$ is not an optimal solution of (9). □

Next, we define the sub-problems (A34) of (9) for $k\in\{0,1,\ldots,t^{\mathcal{L}}_{\max}\}$, which present the optimal substructure of (9).

$$\begin{aligned}
\max_{t_b\in\{0,1,2,\ldots\},\forall b\in\mathcal{L}}\quad & \sum_{b\in\mathcal{L}}E(r_b,t_b)\\
\text{s.t.}\quad & \sum_{b\in\mathcal{L}}t_b=k\\
& t_b\leq t^b_{\max},\forall b\in\mathcal{L}.
\end{aligned} \tag{A34}$$

Note that we assume $\sum_{b\in\mathcal{L}} t^b_{\max} \geq t^{\mathcal{L}}_{\max} \geq k$, or otherwise we should include more batches in the block.

We define a multiset Ω_r that collects the value of $(1-p)\beta_p(t,r)$ for all integers $t \geq 0$, i.e.,

$$\Omega_r = \{(1-p)\beta_p(t,r)\colon t \in \{0,1,2,\ldots\}\}.$$

By Lemma 3(b),we have $E(r,t) = (1-p)\sum_{j=0}^{t-1}\beta_p(j,r)$. As $E(r,t)$ is concave with respect to t, $\beta_p(t,r)$ is a monotonic decreasing function on t (also stated in Lemma 1(b)).

This implies the following lemma.

Lemma A1. *$E(r,t)$ equals the sum of the largest t elements in Ω_r.*

Proof. By Lemma 3(b), $E(r,t) = (1-p)\sum_{j=0}^{t-1}\beta_p(j,r)$. By Lemma 1(b), $\beta_p(t,r)$ is a monotonic decreasing function on t. By (10) and (12), we have $\beta_p(0,r) = 1 \geq \beta_p(t,r)$ for all positive integers t. Therefore, $E(r,t)$ is the sum of the largest t elements in Ω_r. □

We fix a block $\mathcal{L}$ and define a multiset

$$\Omega := \biguplus_{b\in\mathcal{L}} \Omega_{r_b} = \{(1-p)\beta_p(t_b,r_b)\colon t_b \in \{0,1,2,\ldots,t^b_{\max}-1\}, b\in\mathcal{L}\}.$$

If two batches $a,b \in \mathcal{L}$ have the same rank, i.e., $r_a = r_b$, then for all t, we have $(1-p)\beta_p(t,r_a) = (1-p)\beta_p(t,r_b)$. As Ω is a multiset, the duplicated values are not eliminated. Now, we have the following lemma to connect (A34) and Ω.

Lemma A2. *The optimal value of* (A34) *is the sum of the largest k elements in Ω.*

Proof. Let $\{t_b\}_{b\in\mathcal{L}}$ solves (A34). We suppose the optimal value is not the sum of the largest k elements in Ω. However, Lemma A1 states that $E(r_b,t_b)$ equals the sum of the largest t_b elements in Ω_{r_b} for all $b \in \mathcal{L}$. This means that there exists two distinct batches $\kappa,\rho \in \mathcal{L}$ with $t_\kappa \leq t^b_{\max}-1$ and $t_\rho \geq 1$ such that $(1-p)\beta_p(t_\kappa,r_\kappa) > (1-p)\beta_p(t_\rho-1,r_\rho)$.

By setting $t^{\mathcal{L}}_{\max} = k$, we can apply Theorem A2 which gives that $\{t_b\}_{b\in\mathcal{L}}$ is not an optimal solution of (A34). The proof is completed by contradiction. □

We define a multiset $\Omega'_{t^{\mathcal{L}}_{\max}}$ which is a collection of the largest $t^{\mathcal{L}}_{\max}$ elements in Ω. By Lemma A2, $\sum_{\omega\in\Omega'_{t^{\mathcal{L}}_{\max}}}\omega$ is the optimal value of (9). For any non-optimal solution $(t^{(k)}_b)_{b\in\mathcal{L}}$, we define a multiset

$$\mho_k := \{(1-p)\beta_p(t,r_b)\colon t \in \{0,1,\ldots,t^{(k)}_b-1\}, b\in\mathcal{L}\},$$

where the value $k \in \{0,1,\ldots,t^{\mathcal{L}}_{\max}-1\}$ is the number of elements in $\Omega'_{t^{\mathcal{L}}_{\max}}$ which are also contained in $\mho_k$.

Appendix G.2. Lazy Evaluations

We consider an iteration in the last **while** loop in Algorithm 3. Suppose we choose to increase t_b by 1 and decrease t_a by 1.

Lemma A3. *If batch a is selected by the max-heap or batch b is selected by the min-heap in any future iteration, then the optimal solution is reached.*

Proof. Suppose batch a with key $\mathcal{A}$ is selected by the max-heap in a future iteration. Note that $\mathcal{A}$ was once the smallest element in $\mho_k$ for some k. Therefore, at the current state where $k' > k$, every element in $\mho_{k'}$ must be no smaller than $\mathcal{A}$. Equivalently, we have $(1-p)\beta_p(t_\kappa,r_\kappa) \leq (1-p)\beta_p(t_\rho-1,r_\rho)$ for all $\kappa,\rho \in \mathcal{L}$. By Theorem A2, the optimal solution is reached. The min-heap counterpart can be proved in a similar fashion. □

Suppose we omit the update for the batch ρ in the heap. We call the key of the batch ρ a corrupted key, or the key of the batch ρ is corrupted. A key which is not corrupted is called an uncorrupted key. A heap with corrupted keys is called a corrupted heap. In other words, the key of a batch is corrupted in a corrupted max-heap if and only if the same batch was once the minimum of the corresponding original min-heap, and vice versa. As a remark, we do not have a guaranteed maximum portion of corrupted keys as an input. Furthermore, we do not adopt the carpooling technique. This suggests that the heap here is not a soft heap [97].

Lemma A4. *If the root of a corrupted heap is a corrupted key, then the optimal solution is reached.*

Proof. We only consider a corrupted max-heap in the proof. We can use similar arguments to show that a corrupted min-heap also works.

In a future iteration, suppose batch a is selected by the corrupted max-heap. We consider the real maximum in the original max-heap. There are three cases.

Case I: batch a is also the root of the original max-heap. As the key of a is corrupted, it means that the batch was once selected by the corresponding min-heap. By Lemma A3, the optimal solution is reached.

Case II: the root of the original max-heap is batch a' where the key of a' is also corrupted. Similar to Case I, batch a' was once selected by the corresponding min-heap, and we can apply Lemma A3 to finish this case.

Case III: the root of the original max-heap is batch a'' where the key of a'' is not corrupted. In this case, the uncorrupted key of a'' is also in the corrupted max-heap. Note that the corrupted key of a is no larger than the actual key of a in the original max-heap. This means that the key of a, a'' and the corrupted key of a have the same value. It is equivalent to let the original max-heap select batch a, as every element in $\mho_{k'}$ must be no smaller than the key of a'', where k' represents the state of the current iteration. Then, the problem is reduced to Case I.

Combining the three cases, the proof is completed. □

Theorem A3. *The updates for batch a in the max-heap and batch b in the min-heap can be omitted.*

Proof. When we omit the updates, the heap itself becomes a corrupted heap. We have to make sure that when a batch with corrupted key is selected, the termination condition of the algorithm is also met.

We can express the key of batch π in a corrupted max-heap and min-heap by $\beta_p(t_\pi + s_\pi, r_\pi)$ and $\beta_p(t_\pi - 1 - u_\pi, r_\pi)$, respectively, where s_π, u_π are non-negative integers. When s_π or u_π is 0, the key is uncorrupted in the corresponding corrupted heap. By Lemma 1(b), we have

$$\begin{aligned}\beta_p(t_\pi + s_\pi, r_\pi) &\le \beta_p(t_\pi, r_\pi),\\ \beta_p(t_\pi - 1, r_\pi) &\le \beta_p(t_\pi - 1 - u_\pi, r_\pi).\end{aligned}$$

That is, the root of the corrupted max-heap is no larger than the root of the original max-heap. Similar for the min-heap. Mathematically, we have

$$\max_{\pi\in\mathcal{L}} \beta_p(t_\pi + s_\pi, r_\pi) \le \max_{\pi\in\mathcal{L}} \beta_p(t_\pi, r_\pi), \tag{A35}$$

$$\min_{\pi\in\mathcal{L}} \beta_p(t_\pi - 1, r_\pi) \le \min_{\pi\in\mathcal{L}} \beta_p(t_\pi - 1 - u_\pi, r_\pi). \tag{A36}$$

Suppose a corrupted key is selected. By Lemma A4, we know that the optimal solution is reached. Therefore, we can apply the contrapositive of Theorem A2 and know that

$$(1-p)\beta_p(t_\kappa, r_\kappa) \le (1-p)\beta_p(t_\rho - 1, r_\rho) \tag{A37}$$

for all $\kappa, \rho \in \mathcal{L}$. We can omit the condition $t_\rho \geq 1$ because by (10) and (12), we have $\beta_p(-1, \cdot) = 1 \geq \beta_p(\cdot, \cdot)$. The inequality (A37) is equivalent to

$$\max_{\pi \in \mathcal{L}} \beta_p(t_\pi, r_\pi) \leq \min_{\pi \in \mathcal{L}} \beta_p(t_\pi - 1, r_\pi).$$

We can mix this inequality with (A35) and (A36) to show that when a corrupted key is selected, we have

$$\max_{\pi \in \mathcal{L}} \beta_p(t_\pi + s_\pi, r_\pi) \leq \min_{\pi \in \mathcal{L}} \beta_p(t_\pi - 1 - u_\pi, r_\pi),$$

which is the termination condition shown in Algorithm 3 after we replaced the heaps into corrupted heaps.

We just showed that once a corrupted key selected, the termination condition is reached. In the other words, before a corrupted key is selected, every previous selection must be an uncorrupted key. That is, the details inside the iterations are unaffected. If an uncorrupted key is selected where it also satisfies the termination condition, then no corrupted key is touched, and the corrupted heap still acts as a normal heap at this point.

The correctness of the algorithm when using a corrupted heap is proven. Moreover, we do not need to mark which key is corrupted. This is, we can omitted the mentioned heap updates for a normal heap. □

We do not need to mark down which key is corrupted while the algorithm still works, so we can simply omit the mentioned updates as lazy evaluations. As there are two heaps in algorithm, we can reduce from four to two heap updates.

Appendix H. Linear Programming Formulation of BAR

In [81], a distributionally robust optimization [98] for AR is formulated as a linear programming problem. It is based on an observation that when the expected rank function $E(r, t)$ is concave with respect to t, we can reformulate it by

$$E(r, t) = \min_{i \in \{0, 1, \ldots, \bar{\imath}\}} (\Delta_{r,i} t + \xi_{r,i})$$

if we fix an artificial upper bound $t \leq \bar{\imath}$, where $\Delta_{r,t} := E(r, i+1) - E(r, i)$ and $\xi_{r,i} := E(r, i) - i\Delta_{r,i}$. In (9), we implicitly have $t \leq t^{\mathcal{L}}_{\max}$, so we can make use of this expression to write (9) as

$$\begin{aligned} \max_{t_b, e_b \geq 0, \forall b \in \mathcal{L}} \quad & \sum_{b \in \mathcal{L}} e_b \\ \text{s.t.} \quad & \sum_{b \in \mathcal{L}} t_b = t^{\mathcal{L}}_{\max} \\ & t_b \leq t^b_{\max}, \forall b \in \mathcal{L} \\ & e_b \leq E(r_b, i) + (E(r_b, i+1) - E(r_b, i))(t_b - i), \forall b \in \mathcal{L}, \forall i \in \{0, 1, \ldots, t^{\mathcal{L}}_{\max}\}, \end{aligned}$$

where t_b is allowed to be a non-integer. A non-integer t_b means that we first generate $\lfloor t_b \rfloor$ recoded packets, then we generate one more recoded packet with probability $t_b - \lfloor t_b \rfloor$. Note that there are $|\mathcal{L}| t^{\mathcal{L}}_{\max}$ constraints for e_b.

To turn such a non-deterministic solution into a deterministic one, we perform the following steps:

1. Collect the batches with non-integer recoded packets into a set S.
2. Calculate $R = \sum_{b \in S} (t_b - \lfloor t_b \rfloor)$. Note that R is an integer for BAR.
3. For every $b \in S$, remove the fractional part of t_b.
4. Randomly select R batches from S and add one recoded packet to each of these batches.

We have an integer R because $\sum_{b \in \mathcal{L}} t_b = t^{\mathcal{L}}_{\max}$. Furthermore, we have $R < |S|$. Referring to the idea of Algorithm 1, we have the same value of $\Delta_{r_b, \lfloor t_b \rfloor}$ for all $b \in S$. After

removing the fractional part of t_b for all $b \in S$, it becomes the sub-problem (A34) (defined in Appendix G.1) with $k = t^{\mathcal{L}}_{\max} - R$. The last step follows Algorithm 1 such that the output is a solution to (9) where t_b for all $b \in \mathcal{L}$ are all integers.

References

1. Luby, M. LT Codes. In Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, Vancouver, BC, Canada, 19 November 2002; pp. 271–282.
2. Shokrollahi, A. Raptor Codes. *IEEE Trans. Inf. Theory* **2006**, *52*, 2551–2567. [CrossRef]
3. Maymounkov, P. *Online Codes*; Technical Report; New York University: New York, NY, USA, 2002.
4. Ho, T.; Koetter, R.; Médard, M.; Karger, D.R.; Effros, M. The Benefits of Coding over Routing in a Randomized Setting. In Proceedings of the 2003 IEEE International Symposium on Information Theory (ISIT), Yokohama, Japan, 29 June–4 July 2003; p. 442.
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. Inf. Theory* **2006**, *52*, 4413–4430. [CrossRef]
6. Ahlswede, R.; Cai, N.; Li, S.Y.R.; Yeung, R.W. Network Information Flow. *IEEE Trans. Inf. Theory* **2000**, *46*, 1204–1216.
7. Koetter, R.; Médard, M. An Algebraic Approach to Network Coding. *IEEE/ACM Trans. Netw.* **2003**, *11*, 782–795. [CrossRef]
8. Li, S.Y.R.; Yeung, R.W.; Cai, N. Linear Network Coding. *IEEE Trans. Inf. Theory* **2003**, *49*, 371–381. [CrossRef]
9. Wu, Y. A Trellis Connectivity Analysis of Random Linear Network Coding with Buffering. In Proceedings of the 2006 IEEE International Symposium on Information Theory (ISIT), Seattle, WA, USA, 9–14 July 2006; pp. 768–772.
10. Lun, D.S.; Médard, M.; Koetter, R.; Effros, M. On coding for reliable communication over packet networks. *Phys. Commun.* **2008**, *1*, 3–20. [CrossRef]
11. Chou, P.A.; Wu, Y.; Jain, K. Practical Network Coding. In Proceedings of the Annual Allerton Conference on Communication Control and Computing, Monticello, IL, USA, 1–3 October 2003; Volume 41, pp. 40–49.
12. 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]
13. 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]
14. 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]
15. 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]
16. 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]
17. 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]
18. Fu, A.; Sadeghi, P.; Médard, M. Dynamic rate adaptation for improved throughput and delay in wireless network coded broadcast. *IEEE/ACM Trans. Netw.* **2014**, *22*, 1715–1728. [CrossRef]
19. Chatzigeorgiou, I.; Tassi, A. Decoding delay performance of random linear network coding for broadcast. *IEEE Trans. Veh. Technol.* **2017**, *66*, 7050–7060. [CrossRef]
20. Yazdani, N.; Lucani, D.E. Revolving codes: Overhead and computational complexity analysis. *IEEE Commun. Lett.* **2021**, *25*, 374–378. [CrossRef]
21. Torres Compta, P.; 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]
22. Torres Compta, P.; Fitzek, F.H.P.; Lucani, D.E. On the Effects of Heterogeneous Packet Lengths on Network Coding. In Proceedings of the European Wireless 2014, Barcelona, Spain, 14–16 May 2014; pp. 385–390.
23. Taghouti, M.; Lucani, D.E.; Cabrera, J.A.; Reisslein, M.; Pedersen, M.V.; Fitzek, F.H.P. Reduction of Padding Overhead for RLNC Media Distribution with Variable Size Packets. *IEEE Trans. Broadcast.* **2019**, *65*, 558–576. [CrossRef]
24. Taghouti, M.; Tömösközi, M.; Howeler, M.; Lucani, D.E.; Fitzek, F.H.; Bouallegue, A.; Ekler, P. Implementation of Network Coding with Recoding for Unequal-Sized and Header Compressed Traffic. In Proceedings of the 2019 IEEE Wireless Communications and Networking Conference (WCNC), Marrakech, Morocco, 15–19 April 2019.
25. Schütz, B.; Aschenbruck, N. Packet-Preserving Network Coding Schemes for Padding Overhead Reduction. In Proceedings of the 2019 IEEE 44th Conference on Local Computer Networks (LCN), Osnabrueck, Germany, 14–17 October 2019; pp. 447–454.
26. 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.
27. Silva, D. Minimum-Overhead Network Coding in the Short Packet Regime. In Proceedings of the 2012 International Symposium on Network Coding (NetCod), Boston, MA, USA, 29–30 June 2012; pp. 173–178.
28. 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.

29. 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, Lausanne, Switzerland, 15–16 June 2009; pp. 74–79.
30. Heidarzadeh, A.; Banihashemi, A.H. Overlapped Chunked Network Coding. In Proceedings of the 2010 IEEE Information Theory Workshop on Information Theory (ITW), Dublin, Ireland, 30 August–3 September 2010; pp. 1–5.
31. Li, Y.; Soljanin, E.; Spasojevic, P. Effects of the Generation Size and Overlap on Throughput and Complexity in Randomized Linear Network Coding. *IEEE Trans. Inf. Theory* **2011**, *57*, 1111–1123. [CrossRef]
32. Tang, B.; Yang, S.; Yin, Y.; Ye, B.; Lu, S. Expander graph based overlapped chunked codes. In Proceedings of the 2012 IEEE International Symposium on Information Theory (ISIT), Cambridge, MA, USA, 1–6 July 2012; pp. 2451–2455.
33. 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.
34. Mahdaviani, K.; Yazdani, R.; Ardakani, M. Linear-Complexity Overhead-Optimized Random Linear Network Codes. *arXiv* **2013**, arXiv:1311.2123.
35. Yang, S.; Tang, B. From LDPC to chunked network codes. In Proceedings of the 2014 IEEE Information Theory Workshop on Information Theory (ITW), Hobart, TAS, Australia, 2–5 November 2014; pp. 406–410.
36. Tang, B.; Yang, S. An LDPC Approach for Chunked Network Codes. *IEEE/ACM Trans. Netw.* **2018**, *26*, 605–617. [CrossRef]
37. Yang, S.; Yeung, R.W. Coding for a network coded fountain. In Proceedings of the 2011 IEEE International Symposium on Information Theory (ISIT), St. Petersburg, Russia, 31 July–5 August 2011; pp. 2647–2651.
38. Yang, S.; Yeung, R.W. Batched Sparse Codes. *IEEE Trans. Inf. Theory* **2014**, *60*, 5322–5346. [CrossRef]
39. Yang, S.; Yeung, R.W. *BATS Codes: Theory and Practice*; Synthesis Lectures on Communication Networks; Morgan & Claypool Publishers: San Rafael, CA, USA, 2017.
40. Yang, S.; Ho, S.W.; Meng, J.; Yang, E.H. Capacity Analysis of Linear Operator Channels Over Finite Fields. *IEEE Trans. Inf. Theory* **2014**, *60*, 4880–4901. [CrossRef]
41. Zhou, Q.; Yang, S.; Yin, H.H.F.; Tang, B. On BATS Codes with Variable Batch Sizes. *IEEE Commun. Lett.* **2017**, *21*, 1917–1920. [CrossRef]
42. Huang, Q.; Sun, K.; Li, X.; Wu, D.O. Just FUN: A Joint Fountain Coding and Network Coding Approach to Loss-Tolerant Information Spreading. In Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing, Philadelphia, PA, USA, 11–14 August 2014; pp. 83–92.
43. Yin, H.H.F.; Ng, K.H.; Wang, X.; Cao, Q. On the Minimum Delay of Block Interleaver for Batched Network Codes. In Proceedings of the 2019 IEEE International Symposium on Information Theory (ISIT), Paris, France, 7–12 July 2019; pp. 1957–1961.
44. Yin, H.H.F.; Ng, K.H.; Wang, X.; Cao, Q.; Ng, L.K.L. On the Memory Requirements of Block Interleaver for Batched Network Codes. In Proceedings of the 2020 IEEE International Symposium on Information Theory (ISIT), Angeles, CA, USA, 21–26 June 2020; pp. 1658–1663.
45. Zhou, Z.; Li, C.; Yang, S.; Guang, X. Practical Inner Codes for BATS Codes in Multi-Hop Wireless Networks. *IEEE Trans. Veh. Technol.* **2019**, *68*, 2751–2762. [CrossRef]
46. Zhou, Z.; Kang, J.; Zhou, L. Joint BATS Code and Periodic Scheduling in Multihop Wireless Networks. *IEEE Access* **2020**, *8*, 29690–29701. [CrossRef]
47. Yang, S.; Yeung, R.W.; Cheung, J.H.F.; Yin, H.H.F. BATS: Network Coding in Action. In Proceedings of the Annual Allerton Conference on Communication Control and Computing, Monticello, IL, USA, 30 September–3 October 2014; pp. 1204–1211.
48. Tang, B.; Yang, S.; Ye, B.; Guo, S.; Lu, S. Near-Optimal One-Sided Scheduling for Coded Segmented Network Coding. *IEEE Trans. Comput.* **2016**, *65*, 929–939. [CrossRef]
49. Yin, H.H.F.; Yang, S.; Zhou, Q.; Yung, L.M.L. Adaptive Recoding for BATS Codes. In Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT), Barcelona, Spain, 10–15 July 2016; pp. 2349–2353.
50. Yin, H.H.F.; Tang, B.; Ng, K.H.; Yang, S.; Wang, X.; Zhou, Q. A Unified Adaptive Recoding Framework for Batched Network Coding. *IEEE J. Sel. Areas Inf. Theory* **2021**, *2*, 1150–1164. [CrossRef]
51. Yin, H.H.F.; Ng, K.H. Impact of Packet Loss Rate Estimation on Blockwise Adaptive Recoding for Batched Network Coding. In Proceedings of the 2021 IEEE International Symposium on Information Theory (ISIT), Melbourne, VIC, Australia, 12–20 July 2021; pp. 1415–1420.
52. Yin, H.H.F.; Ng, K.H.; Zhong, A.Z.; Yeung, R.W.; Yang, S.; Chan, I.Y.Y. Intrablock Interleaving for Batched Network Coding with Blockwise Adaptive Recoding. *IEEE J. Sel. Areas Inf. Theory* **2021**, *2*, 1135–1149. [CrossRef]
53. Breidenthal, J.C. The Merits of Multi-Hop Communication in Deep Space. In Proceedings of the 2000 IEEE Aerospace Conference, Big Sky, MT, USA, 18–25 March 2000; Volume 1, pp. 211–222.
54. Zhao, H.; Dong, G.; Li, H. Simplified BATS Codes for Deep Space Multihop Networks. In Proceedings of the 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference, Chongqing, China, 20–22 May 2016; pp. 311–314.
55. Yeung, R.W.; Dong, G.; Zhu, J.; Li, H.; Yang, S.; Chen, C. Space Communication and BATS Codes: A Marriage Made in Heaven. *J. Deep. Space Explor.* **2018**, *5*, 129–139.
56. Sozer, E.M.; Stojanovic, M.; Proakis, J.G. Underwater Acoustic Networks. *IEEE J. Ocean. Eng.* **2000**, *25*, 72–83. [CrossRef]

57. Yang, S.; Ma, J.; Huang, X. Multi-Hop Underwater Acoustic Networks Based on BATS Codes. In Proceedings of the 13th International Conference on Underwater Networks & Systems, Shenzhen, China, 3–5 December 2018; pp. 30:1–30:5.
58. Sprea, N.; Bashir, M.; Truhachev, D.; Srinivas, K.V.; Schlegel, C.; Sacchi, C. BATS Coding for Underwater Acoustic Communication Networks. In Proceedings of the OCEANS 2019, Marseille, France, 17–20 June 2019; pp. 1–10.
59. Yin, H.H.F.; Yeung, R.W.; Yang, S. A Protocol Design Paradigm for Batched Sparse Codes. *Entropy* **2020**, *22*. 790. [CrossRef] [PubMed]
60. Yin, H.H.F.; Tahernia, M. On the Design of Timeout-Based Recoders for Batched Network Codes in the MU-MIMO Regime. In Proceedings of the 2022 IEEE Region 10 Conference (TENCON), Hong Kong, China, 1–4 November 2022.
61. Qing, J.; Yin, H.H.F.; Yeung, R.W. Enhancing the Decoding Rates of BATS Codes by Learning with Guided Information. In Proceedings of the 2022 IEEE International Symposium on Information Theory (ISIT), Espoo, Finland, 26 June–1 July 2022; pp. 37–42.
62. Yang, S.; Zhou, Q. Tree Analysis of BATS Codes. *IEEE Commun. Lett.* **2016**, *20*, 37–40. [CrossRef]
63. Yang, S.; Ng, T.C.; Yeung, R.W. Finite-Length Analysis of BATS Codes. *IEEE Trans. Inf. Theory* **2018**, *64*, 322–348. [CrossRef]
64. Yang, J.; Shi, Z.; Wang, C.; Ji, J. Design of Optimized Sliding-Window BATS Codes. *IEEE Commun. Lett.* **2019**, *23*, 410–413. [CrossRef]
65. Xu, X.; Zeng, Y.; Guan, Y.L.; Yuan, L. Expanding-Window BATS Code for Scalable Video Multicasting Over Erasure Networks. *IEEE Trans. Multimed.* **2018**, *20*, 271–281. [CrossRef]
66. Yin, H.H.F.; Wong, H.W.H.; Tahernia, M.; Qing, J. Packet Size Optimization for Batched Network Coding. In Proceedings of the 2022 IEEE International Symposium on Information Theory (ISIT), Espoo, Finland, 26 June–1 July 2022; pp. 1584–1589.
67. Yang, S.; Yeung, R.W. Network Communication Protocol Design from the Perspective of Batched Network Coding. *IEEE Commun. Mag.* **2022**, *60*, 89–93. [CrossRef]
68. Shokrollahi, A.; Luby, M. Raptor Codes. In *Foundations and Trends in Communications and Information Theory*; Now Publishers Inc.: Hanover, MA, USA, 2011; Volume 6.
69. Shokrollahi, A.; Lassen, S.; Karp, R. Systems and Processes for Decoding Chain Reaction Codes through Inactivation. U.S. Patent 6,856,263, 15 February 2005.
70. Xu, X.; Guan, Y.L.; Zeng, Y. Batched Network Coding with Adaptive Recoding for Multi-Hop Erasure Channels with Memory. *IEEE Trans. Commun.* **2018**, *66*, 1042–1052. [CrossRef]
71. Yin, H.H.F.; Tahernia, M. Multi-Phase Recoding for Batched Network Coding. In Proceedings of the 2022 IEEE Information Theory Workshop on Information Theory (ITW), Mumbai, India, 6–9 November 2022; pp. 25–30.
72. Ye, F.; Roy, S.; Wang, H. Efficient Data Dissemination in Vehicular Ad Hoc Networks. *IEEE J. Sel. Areas Commun. (JSAC)* **2012**, *30*, 769–779. [CrossRef]
73. Yin, H.H.F.; Xu, X.; Ng, K.H.; Guan, Y.L.; Yeung, R.W. Packet Efficiency of BATS Coding on Wireless Relay Network with Overhearing. In Proceedings of the 2019 IEEE International Symposium on Information Theory (ISIT), Paris, France, 7–12 July 2019; pp. 1967–1971.
74. Lucani, D.E.; Médard, M.; Stojanovic, M. Random Linear Network Coding for Time-Division Duplexing: Field Size Considerations. In Proceedings of the 2009 IEEE Global Telecommunications Conference, Honolulu, HI, USA, 30 November–4 December 2009; pp. 1–6.
75. Yin, H.H.F.; Xu, X.; Ng, K.H.; Guan, Y.L.; Yeung, R.W. Analysis of Innovative Rank of Batched Network Codes for Wireless Relay Networks. In Proceedings of the 2021 IEEE Information Theory Workshop on Information Theory (ITW), Kanazawa, Japan, 17–21 October 2021.
76. Zhang, C.; Tang, B.; Ye, B.; Lu, S. An efficient chunked network code based transmission scheme in wireless networks. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6.
77. Gilbert, E.N. Capacity of a Burst-Noise Channel. *Bell Syst. Tech. J.* **1960**, *39*, 1253–1265. [CrossRef]
78. Elliott, E.O. Estimates of Error Rates for Codes on Burst-Noise Channels. *Bell Syst. Tech. J.* **1963**, *42*, 1977–1997. [CrossRef]
79. NIST Digital Library of Mathematical Functions. Release 1.1.0. Available online: http://dlmf.nist.gov/ (accessed on 15 December 2020).
80. Galassi, M.; Davies, J.; Theiler, J.; Gough, B.; Jungman, G.; Booth, M.; Rossi, F. *GNU Scientific Library Reference Manual*, 3rd ed.; Network Theory Ltd.: London, UK, 2002.
81. Wang, J.; Jia, Z.; Yin, H.H.F.; Yang, S. Small-Sample Inferred Adaptive Recoding for Batched Network Coding. In Proceedings of the 2021 IEEE International Symposium on Information Theory (ISIT), Melbourne, VIC, Australia, 12–20 July 2021; pp. 1427–1432.
82. Dong, Y.; Jin, S.; Yang, S.; Yin, H.H.F. Network Utility Maximization for BATS Code Enabled Multihop Wireless Networks. In Proceedings of the 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020.
83. Dong, Y.; Jin, S.; Chen, Y.; Yang, S.; Yin, H.H.F. Utility Maximization for Multihop Networks Employing BATS Codes with Adaptive Recoding. *IEEE J. Sel. Areas Inf. Theory* **2021**, *2*, 1120–1134. [CrossRef]
84. Fredman, M.L.; Tarjan, R.E. Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. *J. ACM (JACM)* **1987**, *34*, 596–615. [CrossRef]
85. Musser, D. Introspective Sorting and Selection Algorithms. *Softw. Pract. Exp.* **1997**, *27*, 983–993. [CrossRef]
86. Hoare, C.A.R. Algorithm 65: Find. *Commun. ACM* **1961**, *4*, 321–322. [CrossRef]

87. Blum, M.; Floyd, R.W.; Pratt, V.; Rivest, R.L.; Tarjan, R.E. Time Bounds for Selection. *J. Comput. Syst. Sci.* **1973**, *7*, 448–461. [CrossRef]
88. Seward, H.H. Information Sorting in the Application of Electronic Digital Computers to Business Operations. Master's Thesis, MIT Digital Computer Laboratory, Cambridge, UK, 1954; Report R-232.
89. Higham, N.J. *Accuracy and Stability of Numerical Algorithms: Second Edition*; Other Titles in Applied Mathematics; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2002.
90. Hodges, J.L., Jr.; Lehmann, E.L. Some Problems in Minimax Point Estimation. *Ann. Math. Stat.* **1950**, *21*, 182–197. [CrossRef]
91. Steinhaus, H. The Problem of Estimation. *Ann. Math. Stat.* **1957**, *28*, 633–648. [CrossRef]
92. Qing, J.; Leong, P.H.W.; Yeung, R.W. Performance Analysis and Optimal Design of BATS Code: A Hardware Perspective. *IEEE Trans. Veh. Technol.* **2023**, 1–14. [CrossRef]
93. Yang, S.; Yeung, W.H.; Chao, T.I.; Lee, K.H.; Ho, C.I. Hardware Acceleration for Batched Sparse Codes. U.S. Patent 10,237,782, 19 March 2019.
94. Yin, H.F.H.; Yang, S.; Yeung, W.H.R. Loss-Resilient Protocols for Communication Networks. U.S. Patent 10,425,192, 24 September 2019.
95. Yin, H.F.H.; Ng, K.H.; Zhong, Z.; Yeung, R.W.H.; Yang, S. Compatible Packet Separation for Communication Networks. U.S. Patent 11,452,003, 20 September 2022.
96. Yin, H.F.H. Recoding Optimizations in Batched Sparse Codes. Ph.D. Thesis, The Chinese University of Hong Kong, Hong Kong, China, 2019.
97. Chazelle, B. The Soft Heap: An Approximate Priority Queue with Optimal Error Rate. *J. ACM (JACM)* **2000**, *47*, 16. [CrossRef]
98. Gao, R.; Kleywegt, A.J. Distributionally Robust Stochastic Optimization with Wasserstein Distance. *Math. Oper. Res.* **2023**, *48*, 603–655. [CrossRef]

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.

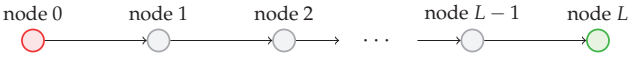


Figure 1. A line network of length L. Node 0 is the source node, and node L is the destination node. The direct edge from node i to node $i+1$ ($i = 0, 1, \ldots, L-1$) illustrates the network link.

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 $\mathrm{rank}(\mathbf{H}_i^{(u)})$ 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]:

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

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

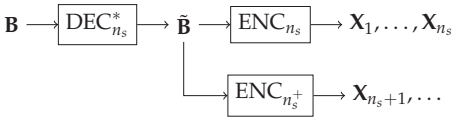
We justify that the above encoding process is systematic by showing that the first n_s batches include all the message packets. Denote by $\mathrm{ENC}^*_{n_s}$ the encoder that generates only the M_i non-redundant coded packets in the ith batch, where $i = 1, \ldots, n_s$. For any K packets $\mathbf{B}$, $\mathrm{DEC}^*_{n_s}\big(\mathrm{ENC}^*_{n_s}(\mathbf{B})\big) = \mathrm{DEC}_{n_s}\big(\mathrm{ENC}_{n_s}(\mathbf{B})\big) = \mathbf{B}$. 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., $\mathrm{ENC}^*_{n_s}\big(\mathrm{DEC}^*_{n_s}(\mathbf{B})\big) = \mathrm{ENC}^*_{n_s}(\tilde{\mathbf{B}}) = \mathbf{B}$.

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 $\mathbf{X}_i = \tilde{\mathbf{X}}_i \cup \bar{\mathbf{X}}_i$.

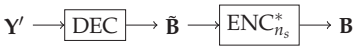
The batches generated by the above systematic encoding process will be further transmitted through a network and processed by the inner code. Let $\mathbf{Y}'$ be the coded

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 $\mathrm{DEC}^*_{n_s}$, 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.

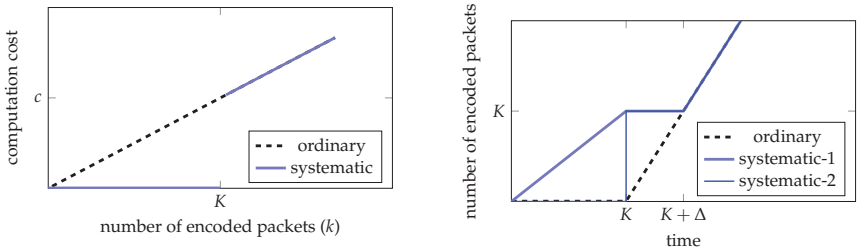


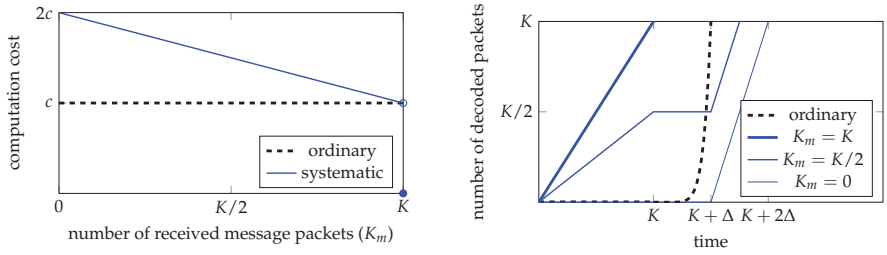
Figure 3. Illustration of the encoding computation cost for the ordinary outer code and the systematic outer code. (**a**) shows the encoding computation cost of generating the first k coded packets. For the ordinary outer code, the computation cost increases linearly with k. For the systematic outer code, the computation cost is 0 when $k \leq K$. The jump in the computation cost after time K is used to execute $\mathrm{DEC}^*_{n_s}$. (**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 $\mathrm{ENC}^*_{n_s}$, which takes computation cost b on average to generate a packet. As K_m message packets have been received, we only need to use $\mathrm{ENC}^*_{n_s}$ 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 $\mathrm{ENC}^*_{n_s}$ 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.



(**a**) (**b**)

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 (ENC, DEC). In the following part of this section, we discuss the traditional random approach to designing a consistent (ENC, DEC). In the next section, we discuss a new approach that can design a consistent (ENC, 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 $\text{ENC}_{n_s^+}$ and DEC for a consistent outer code (ENC, DEC).

The ordinary outer code is random, but we need a deterministic encoder–decoder pair $(\text{ENC}_{n_s}, \text{DEC}_{n_s})$ 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 $(\text{ENC}_{n_s}, \text{DEC}_{n_s})$ 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 $\text{ENC}_{n_s^+}$ 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.

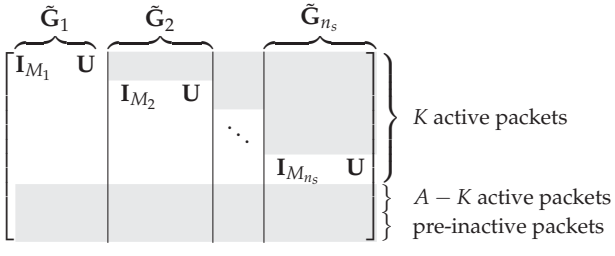


Figure 5. Illustration of encoding using triangular embedding. The gray part contains non-zero entries and the white part contains only zero.

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.

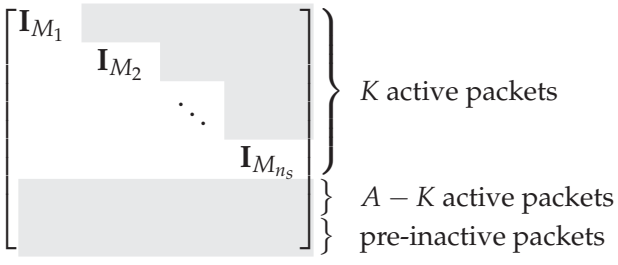


Figure 6. Illustration of decoding using triangular embedding. The gray part contains non-zero entries and the white part contains only zero.

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.

	$\boldsymbol{K=10M}$	$\boldsymbol{K=100M}$	$\boldsymbol{K=1000M}$
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* i*th 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 $\mathrm{Col}(\mathbf{H}^{(u+)}) = \mathrm{Col}(\mathbf{H}^{(u)}\mathbf{\Phi}^{(u)}\mathbf{E}^{(u)}) \subseteq \mathrm{Col}(\mathbf{H}^{(u)})$, $\mathbf{e}_i \notin \mathrm{Col}(\mathbf{H}^{(u+)})$ 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 $r = \text{rank}(\mathbf{H}^{(u)})$ 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 $\text{rank}(\mathbf{H}^{(u)})$. 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* $N_{u+} < r = \text{rank}(\mathbf{H}^{(u)})$. *Under the condition that* $\mathbf{e}_i \in \text{Col}(\mathbf{H}^{(u)})$, *the probability that* $\mathbf{e}_i \in \text{Col}(\mathbf{H}^{(u+)})$ *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.

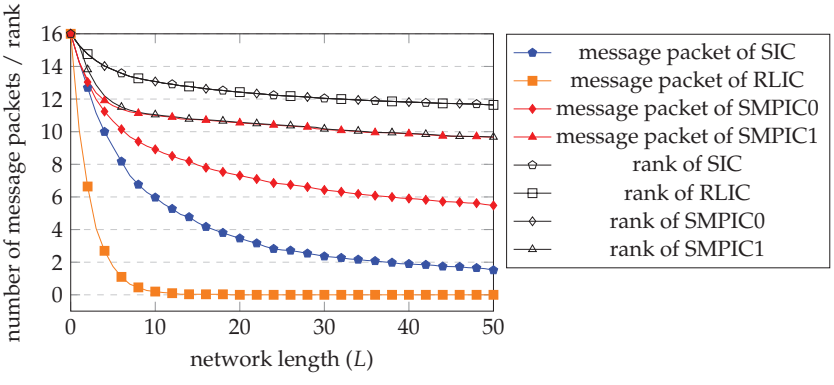


Figure 7. The average number of recovered message packets and the average rank at node 0 to 50 for SIC, RLIC, SMPIC with $w = 0$ (denoted by SMPIC0) and SMPIC with $w = N'_u - s$ (denoted by SMPIC1), each with 500 trials.

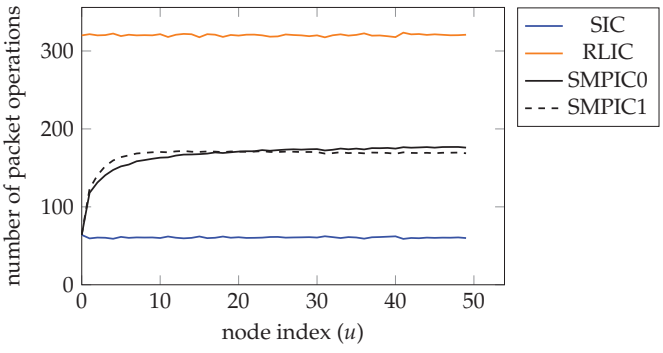


Figure 8. The average number of linear combination operations at node 0 to node 49 for SIC, RLIC, SMPIC with $w = 0$ (denoted by SMPIC0) and SMPIC with $w = N'_u - s$ (denoted by SMPIC1), each with 500 trials.

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:

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

Observe that

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

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 $\mathbf{\Psi}$ asymptotically optimized for the rank-M rank distribution. The non-zero entries of $\mathbf{\Psi}$ 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.

$\mathbf{\Psi}_{17}$	$\mathbf{\Psi}_{18}$	$\mathbf{\Psi}_{19}$	$\mathbf{\Psi}_{20}$	$\mathbf{\Psi}_{23}$	$\mathbf{\Psi}_{27}$	$\mathbf{\Psi}_{31}$	$\mathbf{\Psi}_{35}$
0.0588	0.0571	0.0245	0.0899	0.1170	0.0921	0.0678	0.0679
$\mathbf{\Psi}_{43}$	$\mathbf{\Psi}_{45}$	$\mathbf{\Psi}_{63}$	$\mathbf{\Psi}_{73}$	$\mathbf{\Psi}_{123}$	$\mathbf{\Psi}_{126}$	$\mathbf{\Psi}_{239}$	
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:

- $\mathbf{B}$: all the input packets.
- range: a range of index of batches.

Output:

- $\mathbf{X}$: an array of the generated batches.

1: **procedure** ENC($\mathbf{B}$, 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** $\mathbf{X}$

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:

- $\mathbf{B}$: 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** $\mathbf{b}$ in $\mathbf{B}_i$ **do**
10: Mark $\mathbf{b}$ 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 $\mathbf{b}$ from that equation.
13: Remove i from Ω.
14: $\begin{bmatrix}\mathbf{B}, \mathbf{B}_p\end{bmatrix} \leftarrow$ Decode the precode using the decoded packets.
15: **return** $\mathbf{B}$

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).

entropy

MDPI

Article

Multiple Linear-Combination Security Network Coding

Yang Bai [1,*,†], Xuan Guang [1,†] and Raymond W. Yeung [2,†]

[1] School of Mathematical Sciences and LPMC, Nankai University, Tianjin 300071, China
[2] Institute of Network Coding and the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong SAR, China
* Correspondence: bbbyang@mail.nankai.edu.cn
† These authors contributed equally to this work.

Abstract: In this paper, we put forward the model of multiple linear-combination security multicast network coding, where the wiretapper desires to obtain some information about a predefined set of multiple linear combinations of the source symbols by eavesdropping any one (but not more than one) channel subset up to a certain size r, referred to as the *security level*. For this model, the security capacity is defined as the maximum average number of source symbols that can be securely multicast to all sink nodes for one use of the network under the linear-combination security constraint. For any security level and any linear-combination security constraint, we fully characterize the security capacity in terms of the ratio of the rank of the linear-combination security constraint to the number of source symbols. Also, we develop a general construction of linear security network codes. Finally, we investigate the asymptotic behavior of the security capacity for a sequence of linear-combination security models and discuss the asymptotic optimality of our code construction.

Keywords: information-theoretical security; linear-combination security; network coding; secure network coding; security capacity; code construction; asymptotic behavior

Citation: Bai, Y.; Guang, X.; Yeung, R.W. Multiple Linear-Combination Security Network Coding. *Entropy* **2023**, *25*, 1135. https://doi.org/10.3390/e25081135

Academic Editor: Syed A. Jafar

Received: 29 May 2023
Revised: 11 July 2023
Accepted: 20 July 2023
Published: 28 July 2023

1. Introduction

In 2000, Ahlswede et al. [1] proposed the general concept of network coding. In particular, they investigated the single-source multicast network coding problem, where the source symbols generated by a single source node are required to multicast to multiple sink nodes through a noiseless network while the nodes in the network are allowed to process the received information. It was proven in [1] that if coding is applied at the intermediate nodes (rather than routing only), the source node can multicast source symbols to all the sink nodes at the theoretically maximum rate, i.e., the smallest minimum cut capacity separating a sink node from the source node, as the alphabet size of both the information source and the channel transmission symbol tends toward infinity. In 2003, Li et al. [2] proved that linear network coding over a finite alphabet is sufficient for optimal multicast by means of a vector space approach. Independently, Koetter and Médard [3] developed an algebraic characterization of linear network coding by means of a matrix approach. Jaggi et al. [4] further presented a deterministic polynomial-time algorithm for constructing a linear network code. For comprehensive discussions of network coding, we refer the reader to [5–10].

In the paradigm of network coding, information-theoretic security in the presence of a wiretapper is naturally considered (cf. [11–28]), called the *secure network coding* problem. In the model of secure network coding over a wiretap network, (i) the source node multicasts the source symbols to all the sink nodes, which, as legal users, are required to correctly decode the source symbols; and (ii) the wiretapper, who can access any one but not more than one wiretap set of communication channels, is not allowed to obtain any information about the source symbols. The classical information-theoretically secure models, e.g., Shannon's cipher system [29], secret sharing [30,31] and the wiretap channel II [32], can

be regarded as special cases of the secure network coding model. In particular, a wiretap network is called an *r-wiretap network* if the wiretapper can fully access an arbitrary subset of, at most, r edges, where the non-negative integer r is called the *security level*.

In the model of secure network coding, to guarantee the required information-theoretic security, it is necessary to randomize the source symbols to combat the wiretapper. Cai and Yeung [11] presented a code construction for the r-wiretap network. El Rouayheb et al. [12] further showed that the Cai–Yeung code construction can be viewed as a network generalization of the code construction for wiretap channel II in [32]. Motivated by El Rouayheb et al., Silva and Kschischang [13] proposed a universal design of security network codes based on rank-metric codes. For the construction of security network codes in [11–13]. However, the existing upper bounds on the minimum required alphabet size may be too large for implementation for certain applications in terms of computational complexity and storage requirement. Feldman et al. [33] showed that for a given security level, the alphabet size can be reduced by sacrificing a small fraction of the information rate. However, if the information rate is not sacrificed, whether it is possible to reduce the required alphabet size is considered an open problem [12,17]. Recently, Guang and Yeung [18] developed a systematic graph-theoretic approach to improve the upper bound on the minimum required alphabet size for the existence of secure network codes, achieving an improvement of general significance. Subsequently, in order to tackle the problem of secure network coding when the information rate and the secure level may vary over time, Guang et al. [19] put forward local-encoding-preserving secure network coding, where a family of secure linear network codes is called local-encoding-preserving if all the codes in this family use a common local encoding operation at each intermediate node in the network. They also constructed a family of local-encoding-preserving secure linear network codes applicable for all possible pairs of rate and security level. We note that the variable-rate linear network coding problem without security consideration was previously investigated by Fong and Yeung [34].

In this paper, we put forward the model of multiple linear-combination security network coding, where multiple linear combinations (containing single linear combination as a special case) of the source symbols are required to be protected from the wiretapper. More precisely, in this model over an r-wiretap network, (i) the single source node generates source symbols over a finite field F, and all the source symbols are required to be correctly decoded at all the sink nodes; and (ii) for a predefined set of linear combinations of the source symbols, the wiretapper, who can fully access any channel subset of a size not larger than r, is not allowed to obtain any information about these linear combinations. For the above security model with security level r, the (linear-combination) security capacity is defined as the maximum average number of source symbols that can be securely multicast to all the sink nodes for one use of the network under the above linear-combination security constraint. A model related to the current work is that considered by Bhattad and Narayanan [23], which contains weakly secure network coding as a special case. The relation between the current work and that of Bhattad and Narayanan [23] is discussed in Appendix A.

In this paper, we investigate the security capacity and the code construction for this model and analyze the asymptotic behavior of the security capacity and code construction for a sequence of linear-combination security models. The main contributions and organization of this paper are as follows:

- In Section 2, we formally present the model of linear-combination security network coding and the preliminaries, including the necessary notation and definitions.
- In Section 3, we characterize the security capacity by considering different cases of the security level r. We first prove that $C_{\min} - 1$ is the maximum security level such that the source symbols can be securely multicast to all sink nodes with a positive rate, where $C_{\min}$ is the smallest minimum cut capacity separating a sink node from the source node. Therefore, the security capacity is zero for $r \geq C_{\min}$. For any nontrivial security level $1 \leq r \leq C_{\min} - 1$, we prove upper bounds on the security capacity in

terms of the ratio τ of the rank of the linear-combination security constraint to the number of source symbols.
We further develop a systematic construction of linear security network codes, which is applicable to an arbitrary linear-combination security model. Based on the obtained upper bounds and the developed code construction, we fully characterize the security capacity for any possible pair of the number of the source symbols and the linear-combination security constraint. We also determine the threshold value τ_0 such that there is no penalty on the security capacity compared with the capacity without any security consideration when the ratio τ is not larger than τ_0.

- In Section 4, we fully characterize the asymptotic behavior of the security capacity for a sequence of linear-combination security models and prove that our code construction is asymptotically optimal.
- We conclude in Section 5 with a summary of our results.

2. Preliminaries

Consider a communication network whose communication channels are point-to-point. The network is represented by a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are finite sets of nodes and edges, respectively. Here, an edge in the graph $\mathcal{G}$ corresponds to a point-to-point channel in the network. In the graph $\mathcal{G}$, multiple edges between two nodes are allowed. We assume that an element in a finite field F can be reliably transmitted on each edge for each use. We use $\mathrm{tail}(e)$ and $\mathrm{head}(e)$ to denote the *tail* node and the *head* node of an edge e, respectively. For a node $v \in \mathcal{V}$, we let $\mathrm{In}(v) = \{e \in \mathcal{E} : \mathrm{head}(e) = v\}$ and $\mathrm{Out}(v) = \{e \in \mathcal{E} : \mathrm{tail}(e) = v\}$, i.e., $\mathrm{In}(v)$ and $\mathrm{Out}(v)$ are the set of input edges and the set of output edges, respectively. Furthermore, a sequence of edges $(e_1, e_2, \cdots, e_m)$ is called a (directed) *path* from the node $\mathrm{tail}(e_1)$ to the node $\mathrm{head}(e_m)$ if $\mathrm{tail}(e_i) = \mathrm{head}(e_{i-1})$ for $i = 2, 3, \cdots, m$. For two nodes u and v with $u \neq v$, an edge subset $C \subseteq \mathcal{E}$ is called a *cut* separating v from u if no path exists from u to v upon removing the edges in C. The *capacity* of a cut separating v from u is defined as the size of this cut. A cut C separating v from u is called a *minimum cut* separating v from u if there does not exist a cut C' separating v from u such that $|C'| < |C|$. The capacity of a minimum cut separating v from u is called the *minimum cut capacity* separating v from u, as denoted by $\mathrm{mincut}(u, v)$. There is a single *source node* $s \in \mathcal{V}$ and a set of *sink nodes* $T \subseteq \mathcal{V} \setminus \{s\}$ on the graph $\mathcal{G}$. Without loss of generality, we assume that the source node s has no input edges and that every sink node $t \in T$ has no output edges, i.e., $\mathrm{In}(s) = \mathrm{Out}(t) = \emptyset$, $\forall\, t \in T$. The graph $\mathcal{G}$, together with s and T, forms a *network* $\mathcal{N}$ denoted by $\mathcal{N} = (\mathcal{G}, s, T)$.

The source node s generates L *source symbols* $B_1, B_2, \cdots, B_L$ that are independent and identically distributed (i.i.d.) random variables with a uniform distribution on the finite field F. All the source symbols are required to be multicast to every sink node t in T by using the network $\mathcal{N}$ multiple times, i.e., transmitting multiple elements in F on each edge by using the edge multiple times. There is a wiretapper who can eavesdrop any edge subset of a size up to the security level r, while, for a positive integer m_L, the m_L linear combinations of the source symbols

$$\sum_{i=1}^{L} a_{i,j} \cdot B_i, \quad j = 1, 2, \cdots, m_L \tag{1}$$

over the finite field F are required to be protected from the wiretapper, where $a_{i,j}$, $1 \leq i \leq L, 1 \leq j \leq m_L$ are constants in F; that is, the wiretapper is not allowed to obtain any information about the multiple linear combinations of the source symbols given in (1). Furthermore, we let $\mathbf{B} = (B_1, B_2, \cdots, B_L)$, and

$$M_L = \left[a_{i,j}\right]_{1 \leq i \leq L,\, 1 \leq j \leq m_L},$$

is an $L \times m_L$ matrix. Then, the m_L linear combinations in (1) can be written as $\mathbf{B} \cdot M_L$. Without loss of generality, we assume that $m_L \leq L$ and that the matrix M_L has full column rank, i.e.,

$$\text{Rank}(M_L) = m_L.$$

In this model, the security level r is known by the source node and sink nodes, but which edge subset is eavesdropped by the wiretapper is unknown. It suffices to consider only the wiretap sets of a size exactly equal to r. Then, we let

$$\mathcal{W}_r \triangleq \{W \subseteq \mathcal{E} : |W| = r\},$$

where each edge subset $W \in \mathcal{W}_r$ is called a *wiretap set*. We use $\{(L, M_L), r\}$ to denote the above linear-combination security model.

Next, we define a *(linear-combination) security network code* for the security model $\{(L, M_L), r\}$. In order to combat the wiretapper, we may need randomness to randomize the source symbols. However, as we show, it is not always necessary to randomize the source symbols. As part of the code to be defined, we assume that the *key* $\mathbf{K}$ is a random variable uniformly distributed over a finite set $\mathcal{K}$, which is available only at the source node s. The key $\mathbf{K}$ and the source symbols B_i, $i = 1, 2, \cdots, L$ are assumed to be mutually independent. A (L, M_L) security network code is defined as follows. First, we let $b_i \in F$ and $\mathbf{k} \in \mathcal{K}$ be arbitrary outputs of the source symbol B_i and the key $\mathbf{K}$, respectively, $i = 1, 2, \cdots, L$. We further let $\mathbf{b} = (b_1, b_2, \cdots, b_L)$, which is the output of the vector of source symbols $\mathbf{B} = (B_1, B_2, \cdots, B_L)$. A (L, M_L) security network code $\widehat{\mathbf{C}}$ consists of:

- A *local encoding function* $\widehat{\theta}_e$ for each edge $e \in \mathcal{E}$, where

$$\widehat{\theta}_e : \begin{cases} F^L \times \mathcal{K} \to \text{Im}(\widehat{\theta}_e), & \text{if tail}(e) = s; \\ \prod\limits_{d \in \text{In}(\text{tail}(e))} \text{Im}(\widehat{\theta}_d) \to \text{Im}(\widehat{\theta}_e), & \text{otherwise;} \end{cases} \tag{2}$$

  with $\text{Im}(\widehat{\theta}_e)$ denoting the image set of $\widehat{\theta}_e$;
- A decoding function for each sink node $t \in T$:

$$\widehat{\varphi}_t : \prod_{e \in \text{In}(t)} \text{Im}(\widehat{\theta}_e) \to F^L$$

  to decode the source symbols $b_1, b_2, \cdots, b_L$ at t.

Furthermore, we use $y_e \in \text{Im}(\widehat{\theta}_e)$ to denote the message transmitted on each edge $e \in \mathcal{E}$ by using the code $\widehat{\mathbf{C}}$ under $\mathbf{b}$ and $\mathbf{k}$. With the encoding mechanism described in (2), we readily see that y_e is a function of $\mathbf{b}$ and $\mathbf{k}$, as denoted by $\widehat{h}_e(\mathbf{b}\ \mathbf{k})$ (i.e., $y_e = \widehat{h}_e(\mathbf{b}\ \mathbf{k})$), where $\widehat{h}_e$ can be obtained by recursively applying the local encoding functions $\widehat{\theta}_e, e \in \mathcal{E}$ according to any ancestral order of the edges in $\mathcal{E}$. More precisely, for each $e \in \mathcal{E}$, we have

$$\widehat{h}_e(\mathbf{b}\ \mathbf{k}) = \begin{cases} \widehat{\theta}_e(\mathbf{b}\ \mathbf{k}), & \text{if tail}(e) = s; \\ \widehat{\theta}_e\big(\widehat{h}_{\text{In}(u)}(\mathbf{b}\ \mathbf{k})\big), & \text{otherwise;} \end{cases}$$

where $u = \text{tail}(e)$ and $\widehat{h}_E(\mathbf{b}\ \mathbf{k}) \triangleq \big(\widehat{h}_e(\mathbf{b}\ \mathbf{k}) : e \in E\big)$ for an edge subset $E \subseteq \mathcal{E}$ so that $\widehat{h}_{\text{In}(u)}(\mathbf{b}\ \mathbf{k}) = \big(\widehat{h}_e(\mathbf{b}\ \mathbf{k}) : e \in \text{In}(u)\big)$. We call $\widehat{h}_e$ the *global encoding function* of the edge e for the code $\widehat{\mathbf{C}}$.

For the security model $\{(L, M_L), r\}$, a (L, M_L) security network code $\widehat{\mathbf{C}} = \{\widehat{\theta}_e : e \in \mathcal{E};\ \widehat{\varphi}_t : t \in T\}$ is *admissible* if the following *decoding* and *security conditions* are satisfied:

- ***Decoding condition***: All the source symbols are correctly decoded for each sink node $t \in T$, i.e., for each $t \in T$,

$$\widehat{\varphi}_t\big(\widehat{h}_{\text{In}(t)}(\mathbf{b}\ \mathbf{k})\big) = \mathbf{b}, \quad \forall\ \mathbf{b} \in F^L \ \text{ and } \ \forall\ \mathbf{k} \in \mathcal{K}; \tag{3}$$

- ***Security condition***: for each wiretap set, $W \in \mathcal{W}_r$,

$$I\Big(\mathbf{Y}_W;\ \mathbf{B}\cdot M_L\Big) = 0, \tag{4}$$

where $\mathbf{Y}_W \triangleq (Y_e : e \in W)$, and $Y_e \triangleq \widehat{h}_e(\mathbf{B}, \mathbf{K})$ is the random variable transmitted on the edge e.

For an admissible (L, M_L) security network code $\widehat{\mathbf{C}} = \{\widehat{\theta}_e : e \in \mathcal{E};\ \widehat{\varphi}_t : t \in T\}$, we let

$$n_e = \lceil \log_{|F|} |\mathrm{Im}(\widehat{\theta}_e)| \rceil$$

for each edge e in $\mathcal{E}$, which is regarded as the number of times the edge e is used for transmission when applying the code $\widehat{\mathbf{C}}$. We further let $n(\widehat{\mathbf{C}}) \triangleq \max_{e\in\mathcal{E}} n_e$. Then, the *rate* of $\widehat{\mathbf{C}}$ is defined by

$$R(\widehat{\mathbf{C}}) = \frac{L}{n(\widehat{\mathbf{C}})}, \tag{5}$$

which is the average number of source symbols that can be securely multicast to all the sink nodes for one use of the network using the code $\widehat{\mathbf{C}}$.

Furthermore, the *security capacity* for this model $\{(L, M_L), r\}$ is defined as the maximum rate of all admissible (L, M_L) security network codes, i.e.,

$$\mathcal{C} = \max\Big\{R(\widehat{\mathbf{C}}) : \widehat{\mathbf{C}} \text{ is an admissible } (L, M_L) \text{ security network code for } \{(L, M_L), r\}\Big\}.$$

According to the definition of the rate in (5), characterizing the security capacity $\mathcal{C}$ is equivalent to determining the minimum $n(\widehat{\mathbf{C}})$ over all the admissible (L, M_L) security network codes, i.e.,

$$n^* \triangleq \min\Big\{n(\widehat{\mathbf{C}}) : \widehat{\mathbf{C}} \text{ is an admissible } (L, M_L) \text{ security network code for } \{(L, M_L), r\}\Big\}.$$

For instance, a special case of the linear-combination security model $\{(L, M_L), r\}$ is algebraic-sum security network coding, as elaborated below. In this model, the source node s generates L source symbols $B_1, B_2, \cdots, B_L$, which are required to be multicast to every sink node $t \in T$, and the wiretapper, who can eavesdrop any edge subset of size r, is not allowed to obtain any information about the m algebraic sums of the source symbols:

$$\sum_{\substack{i\in[L]:\\ i\equiv j (\mathrm{mod}\ m)}} B_i,\ \ j = 1, 2, \cdots, m, \tag{6}$$

where $1 \le m \le L$, and $[L] \triangleq \{1, 2, \cdots, L\}$. For this algebraic-sum security model, when $m = 1$, we adopt the convention that $i \equiv 1 (\mathrm{mod}\ 1)$ for all $i = 1, 2, \cdots, L$. Then, Equation (6) becomes $\sum_{i=1}^{L} B_i$, i.e., the algebraic sum $\sum_{i=1}^{L} B_i$ of all the L source symbols is required to be protected from the wiretapper. When $m = L$, we have $i \not\equiv i' (\mathrm{mod}\ m)$, $\forall\ i, i' \in [L]$, where $i \ne i'$; thus, all the source symbols $B_1, B_2, \cdots, B_L$ are required to be protected from the wiretapper. This is the standard model of secure network coding, which has been widely studied in the literature, e.g., [11–24].

An example scenario for the application of the linear-combination security model is as follows. A predefined set of linear combinations of the source symbols is required to be protected from the wiretapper, while other linear combinations are unprotected. The source node s generates L source symbols $B_1, B_2, \cdots, B_L$ in the finite field F, which are required to be multicast to every sink node $t \in T$. The $L \times m_L$ matrix M_L is regarded as an F-valued parity-check matrix. We denote the solution space of the system of linear equations $\vec{b} \cdot M_L = \vec{0}$ over F as $V(\vec{0})$, i.e.,

$$V(\vec{0}) = \{\vec{b} \in F^L :\ \vec{b} \cdot M_L = \vec{0}\},$$

where $\vec{0}$ is the zero row m_L-vector. According to the value of $\vec{b} \cdot M_L$, for every output $\vec{b} \in F^L$ of $\mathbf{B} = (B_1, B_2, \cdots, B_L)$, the vector space F^L can be partitioned into $|F|^{m_L}$ cosets of the solution space given by $\{V(\vec{a}) : \vec{a} \in F^{m_L}\}$, where $V(\vec{a}) \triangleq \{\vec{b} \in F^L : \vec{b} \cdot M_L = \vec{a}\}$. In this scenario, we desire to protect the information as to which coset $V(\vec{a})$ the vector $\vec{b}$ lies in, which may contain some useful information for the wiretapper. In other words, the information about the specified linear combinations $\mathbf{B} \cdot M_L$ needs to be protected from the wiretapper, while other linear combinations are unprotected.

3. Characterization of the Capacity ofthe Security Model $\{(L, M_L), r\}$

3.1. Upper Bounds on the Security Capacity

Consider a linear-combination security model $\{(L, M_L), r\}$. We first consider the trivial case of $r \geq C_{\min}$, where $C_{\min} \triangleq \min_{t \in T} \text{mincut}(s, t)$. In this case, for a sink node $t \in T$ such that $\text{mincut}(s, t) = C_{\min}$, the wiretapper is able to decode the source symbols, provided that the sink node t correctly decodes them. This shows that the security capacity is $\mathcal{C} = 0$ for $r \geq C_{\min}$, which implies that $C_{\min} - 1$ is an upper bound on the maximum security level for which the source symbols can be multicast with a positive rate. For another trivial case $r = 0$, the security model $\{(L, M_L), 0\}$ becomes a single-source multicast network coding problem without any security consideration. Given the fact that the maximum rate at which the source symbols can be correctly multicast to all the sink nodes is $C_{\min}$ (cf. [1,6]), we thus obtain that

$$n^* = \left\lceil \frac{L}{C_{\min}} \right\rceil,$$

or, equivalently,

$$\mathcal{C} = \frac{L}{n^*} = \frac{L}{\lceil L/C_{\min} \rceil}.$$

Next, we consider $0 < r < C_{\min}$. We readily see that an admissible (L, M_L) security network code $\widehat{\mathbf{C}}$ is also a network code such that all the L source symbols can be correctly decoded at each $t \in T$. This immediately implies that n^* can be lower-bounded by $\lceil L/C_{\min} \rceil$ for any security level $0 < r < C_{\min}$, i.e.,

$$n^* \geq \left\lceil \frac{L}{C_{\min}} \right\rceil. \tag{7}$$

Furthermore, we present the following lemma, which asserts a non-trivial lower bound on n^*.

Lemma 1. *Consider a linear-combination security model* $\{(L, M_L), r\}$ *with a security level of* $0 < r < C_{\min}$, *where* $\text{Rank}(M_L) = m_L$. *Let* $\tau = m_L/L$. *Then,*

$$n^* \geq \left\lceil \frac{\tau L}{C_{\min} - r} \right\rceil. \tag{8}$$

Proof. First, we claim that

$$H(\mathbf{B} \cdot M_L) = \tau L \cdot \log |F|, \tag{9}$$

where $\tau L = m_L$. To see this, we consider an arbitrary row vector $\vec{x} \in F^{\tau L}$ and obtain

$$\begin{aligned} \Pr\left(\mathbf{B} \cdot M_L = \vec{x}\right) &= \sum_{\mathbf{b} \in F^L:\, \mathbf{b} \cdot M_L = \vec{x}} \Pr\left(\mathbf{B} = \mathbf{b}\right) \\ &= \#\left\{\mathbf{b} \in F^L : \mathbf{b} \cdot M_L = \vec{x}\right\} \cdot \frac{1}{|F|^L} = \frac{1}{|F|^{\tau L}}, \end{aligned} \tag{10}$$

where the equality $\Pr\left(\mathbf{B} = \mathbf{b}\right) = \frac{1}{|F|^L}$ holds because the source symbols $B_i, 1 \leq i \leq L$ are i.i.d. with the uniform distribution on F.

We now consider an arbitrary admissible (L, M_L) security network code: $\widehat{\mathbf{C}} = \{\widehat{\theta}_e : e \in \mathcal{E};\ \widehat{\varphi}_t : t \in T\}$. For an edge subset C that separates a sink node $t \in T$ from the source node s, it follows from the decoding condition (3) that $H(\mathbf{B}|\mathbf{Y}_C) = 0$. This immediately implies that

$$H\big(\mathbf{B} \cdot M_L \big| \mathbf{Y}_C\big) = 0. \tag{11}$$

Furthermore, for any wiretap set $W \in \mathcal{W}_r$ with $W \subseteq C$, it follows from the security condition (4) that

$$H\big(\mathbf{B} \cdot M_L\big) = H\big(\mathbf{B} \cdot M_L \big| \mathbf{Y}_W\big). \tag{12}$$

Combining (11) and (12), we obtain

$$\begin{aligned}
H\big(\mathbf{B} \cdot M_L\big) &= H\big(\mathbf{B} \cdot M_L \big| \mathbf{Y}_W\big) - H\big(\mathbf{B} \cdot M_L \big| \mathbf{Y}_C\big) \\
&= I\big(\mathbf{B} \cdot M_L;\ \mathbf{Y}_{C\setminus W} \big| \mathbf{Y}_W\big) \\
&\le H\big(\mathbf{Y}_{C\setminus W} \big| \mathbf{Y}_W\big) \\
&\le H\big(\mathbf{Y}_{C\setminus W}\big) \\
&\le \sum_{e\in C\setminus W} H(Y_e) \\
&\le \sum_{e\in C\setminus W} \log\big|\mathrm{Im}(\widehat{\theta}_e)\big| \qquad (13) \\
&\le \sum_{e\in C\setminus W} n_e \cdot \log|F| \qquad (14) \\
&\le n(\widehat{\mathbf{C}}) \cdot |C \setminus W| \cdot \log|F|, \qquad (15)
\end{aligned}$$

where the inequality (13) holds because Y_e takes values in $\mathrm{Im}(\widehat{\theta}_e)$, and the inequality (14) follows from

$$n_e = \big\lceil \log_{|F|} \big|\mathrm{Im}(\widehat{\theta}_e)\big| \big\rceil \ge \log_{|F|} \big|\mathrm{Im}(\widehat{\theta}_e)\big|,$$

and the inequality (15) follows from $n(\widehat{\mathbf{C}}) = \max_{e\in\mathcal{E}} n_e$.

Combining (9) and (15), we obtain

$$n(\widehat{\mathbf{C}}) \ge \frac{H\big(\mathbf{B} \cdot M_L\big)}{|C \setminus W| \cdot \log|F|} = \frac{\tau L}{|C \setminus W|}.$$

Note that the above inequality is true for each sink node $t \in T$ and all the pairs (C, W) of the cut C separating t from s and the wiretap set $W \in \mathcal{W}_r$ such that $W \subseteq C$. We thus obtain

$$n(\widehat{\mathbf{C}}) \ge \max_{t\in T} \max_{\substack{(W,C)\in\mathcal{W}_r\times\Lambda_t:\\ W\subseteq C}} \frac{\tau L}{|C \setminus W|},$$

where $\Lambda_t \triangleq \big\{C \subseteq \mathcal{E} : C \text{ is a cut separating } t \text{ from } s\big\}$. For each $t \in T$, we have

$$|C \setminus W| \ge C_{\min} - r, \quad \forall\ (W, C) \in \mathcal{W}_r \times \Lambda_t \text{ with } W \subseteq C.$$

According to the definition of $C_{\min}$, this lower bound is achievable for some $t \in T$ and $(W, C) \in \mathcal{W}_r \times \Lambda_t$ such that $W \subseteq C$. It then follows that

$$n(\widehat{\mathbf{C}}) \ge \frac{\tau L}{C_{\min} - r}.$$

Furthermore, since $n(\widehat{\mathbf{C}})$ is an integer, we have

$$n(\widehat{\mathbf{C}}) \geq \left\lceil \frac{\tau L}{C_{\min} - r} \right\rceil. \tag{16}$$

In addition, because the above lower bound (16) on $n(\widehat{\mathbf{C}})$ is valid for any admissible (L, M_L) security network code $\widehat{\mathbf{C}}$, we obtain

$$n^* \geq \left\lceil \frac{\tau L}{C_{\min} - r} \right\rceil.$$

The lemma is thus proven. □

The lower bounds in (7) and (8) on n^* apply to all $0 < r < C_{\min}$. For a specific value of τ, one of them can be tighter than the other. By comparing these bounds, we can readily obtain the upper bounds on the security capacity $\mathcal{C}$ as stated in the following theorem.

Theorem 1. *Consider a linear-combination security model* $\{(L, M_L), r\}$ *with a security level of* $0 < r < C_{\min}$, *where* $\mathrm{Rank}(M_L) = m_L$. *Let*

$$\tau = \frac{m_L}{L} \text{ and } \tau_0 = \frac{C_{\min} - r}{C_{\min}}.$$

- *If* $0 \leq \tau \leq \tau_0$, *then*
$$\mathcal{C} \leq \frac{L}{\lceil L/C_{\min} \rceil}.$$
- *If* $\tau_0 < \tau \leq 1$, *then*
$$\mathcal{C} \leq \frac{L}{\lceil \tau L/(C_{\min} - r) \rceil}.$$

Proof. By comparing the lower bounds ((7) and (8)) on n^*, we immediately obtain

- if $0 \leq \tau \leq \tau_0$, then
$$n^* \geq \left\lceil \frac{L}{C_{\min}} \right\rceil \geq \left\lceil \frac{\tau L}{C_{\min} - r} \right\rceil \tag{17}$$
implying that
$$\mathcal{C} \leq \frac{L}{\lceil L/C_{\min} \rceil};$$
- If $\tau_0 < \tau \leq 1$, we have
$$n^* \geq \left\lceil \frac{\tau L}{C_{\min} - r} \right\rceil \geq \left\lceil \frac{L}{C_{\min}} \right\rceil \tag{18}$$
implying that
$$\mathcal{C} \leq \frac{L}{\lceil \tau L/C_{\min} \rceil}.$$

We have thus proven the theorem. □

3.2. Characterization of the Security Capacity

Next, we present a code construction for the security model $\{(L, M_L), r\}$ with $0 < r < C_{\min}$, which shows that the upper bounds in Theorem 1 for both cases of τ are tight. We thus obtain a full characterization of the security capacity for the security model $\{(L, M_L), r\}$, as stated in the following theorem.

Theorem 2. *Consider a linear-combination security model* $\{(L, M_L), r\}$ *over a finite field* F*, where* $0 < r < C_{\min}$ *and* $|F| > \max\{|T|, \binom{|\mathcal{E}|}{r}\}$*. Let*

$$\tau = \frac{m_L}{L} \text{ and } \tau_0 = \frac{C_{\min} - r}{C_{\min}}.$$

- *If* $0 \leq \tau \leq \tau_0$*, then*

$$\mathcal{C} = \frac{L}{\lceil L/C_{\min} \rceil}. \tag{19}$$

- *If* $\tau_0 < \tau \leq 1$*, then*

$$\mathcal{C} = \frac{L}{\lceil \tau L/(C_{\min} - r) \rceil}. \tag{20}$$

This theorem reveals the somewhat surprising fact that for the case of $0 \leq \tau \leq \tau_0$, there is no penalty on the security capacity compared with the capacity without any security consideration. In Section 4, we further investigate the asymptotic behavior of the security capacity for a sequence of the security models as L tends toward infinity. We not only characterize the asymptotic behavior of the security capacity but also show the asymptotic optimality of our construction.

We first define a linear security network code for the security model $\{(L, M_L), r\}$. Briefly, a (L, M_L) security network code $\widehat{\mathbf{C}}$ is *linear* if the local encoding functions for all the edges are linear. Specifically, we recall that $\mathbf{b} = (b_1, b_2, \cdots, b_L) \in F^L$ is an arbitrary output of the vector of source symbols $\mathbf{B} = (B_1, B_2, \cdots, B_L)$. Let $\mathcal{K} = F^z$, where the non-negative integer z is specified later. Then, the key $\mathbf{K}$ is a random row vector uniformly distributed on F^z. We further let $\mathbf{k} \in F^z$ be an arbitrary output of $\mathbf{K}$. Consequently, for a (L, M_L) linear security network code $\widehat{\mathbf{C}}$, all the global encoding functions $\widehat{h}_e$, $e \in \mathcal{E}$ are linear functions of $\mathbf{b}$ and $\mathbf{k}$. Therefore, there exists an F-valued $(L + z) \times n$ matrix $H_e = \left[\vec{h}_e^{(1)}\ \vec{h}_e^{(2)}\ \cdots\ \vec{h}_e^{(n)}\right]$ for each $e \in \mathcal{E}$ such that

$$\widehat{h}_e(\mathbf{b}\ \mathbf{k}) = (\mathbf{b}\ \mathbf{k}) \cdot H_e,$$

where $n \triangleq n(\widehat{\mathbf{C}})$, and H_e is called the *global encoding matrix* of the edge e for the code $\widehat{\mathbf{C}}$. In particular, if $n(\widehat{\mathbf{C}}) = 1$, then the code $\widehat{\mathbf{C}}$ is called a (L, M_L) *scalar-linear* security network code.

In the following, for the nontrivial case of a security model $\{(L, M_L), r\}$ with a security level of $0 < r < C_{\min}$, we develop a construction of admissible (L, M_L) linear security network codes that can be applied to any pair (L, M_L). This code construction shows that the upper bounds in Theorem 1 for both cases of τ are tight, which we state in the following theorem.

Theorem 3. *Consider a linear-combination security model* $\{(L, M_L), r\}$ *over a finite field* F*, where* $\mathrm{Rank}(M_L) = m_L$*,* $0 < r < C_{\min}$ *and* $|F| > \max\{|T|, \binom{|\mathcal{E}|}{r}\}$*. Let*

$$\tau = \frac{m_L}{L} \text{ and } \tau_0 = \frac{C_{\min} - r}{C_{\min}}.$$

Then, there exists an admissible (L, M_L) *linear security network code* $\widehat{\mathbf{C}}$ *such that*

- *If* $0 \leq \tau \leq \tau_0$*, then*

$$n(\widehat{\mathbf{C}}) = \left\lceil \frac{L}{C_{\min}} \right\rceil; \tag{21}$$

- *if* $\tau_0 < \tau \leq 1$*, then*

$$n(\widehat{\mathbf{C}}) = \left\lceil \frac{\tau L}{C_{\min} - r} \right\rceil. \tag{22}$$

3.3. Proof of Theorem 3

In this subsection, we provide the proof of Theorem 3, which includes three parts: code construction, verification of the decoding condition and verification of the security condition.

► **Code construction**:

We consider a linear-combination security model $\{(L, M_L), r\}$ over a finite field F, where $0 < r < C_{\min}$ and $|F| > \max\left\{|T|, \binom{|\mathcal{E}|}{r}\right\}$. In the following, we construct an admissible (L, M_L) linear security network code such that the L source symbols can be securely multicast to all the sink nodes by transmitting n symbols on each edge, i.e., using the network n times, where

$$n = \begin{cases} \left\lceil \frac{L}{C_{\min}} \right\rceil, & \text{if } 0 \le \tau \le \tau_0, \\ \left\lceil \frac{\tau L}{C_{\min} - r} \right\rceil. & \text{if } \tau_0 < \tau \le 1, \end{cases} \tag{23}$$

(cf. (21) and (22)). For any $0 \le \tau \le 1$, we let

$$z = \begin{cases} 0, & \text{if } L \ge nr + \tau L, \\ nr + \tau L - L, & \text{if } L < nr + \tau L, \end{cases} \tag{24}$$

i.e.,

$$\mathcal{K} = \begin{cases} \emptyset, & \text{if } L \ge nr + \tau L, \\ F^{nr+\tau L-L}, & \text{if } L < nr + \tau L. \end{cases}$$

According to (24), when $L \ge nr + \tau L$, it is unnecessary to randomize the source symbols to guarantee linear-combination security. Furthermore, for any pair (L, z) satisfying (24), we observe that

$$nr + \tau L \le L + z \le nC_{\min}. \tag{25}$$

The first inequality in (25) is straightforward. To prove the second inequality, we consider two cases below.

Case 1: $L \ge nr + \tau L$.

According to (24) we have $z = 0$, and thus:

$$L + z = L. \tag{26}$$

Furthermore, it follows from (23) that for $0 \le \tau \le \tau_0$,

$$n = \left\lceil \frac{L}{C_{\min}} \right\rceil \ge \frac{L}{C_{\min}};$$

and for $\tau_0 < \tau \le 1$,

$$n = \left\lceil \frac{\tau L}{C_{\min} - r} \right\rceil \ge \left\lceil \frac{L}{C_{\min}} \right\rceil \ge \frac{L}{C_{\min}}$$

(cf. (18) for the first inequality in the above equation). Together with (26), we immediately prove that $L + z = L \le nC_{\min}$ for this case.

Case 2: $L < nr + \tau L$.

According to (24), we have

$$L + z = nr + \tau L. \tag{27}$$

Furthermore, it follows from (23) that for $0 \le \tau \le \tau_0$,

$$n = \left\lceil \frac{L}{C_{\min}} \right\rceil \ge \left\lceil \frac{\tau L}{C_{\min} - r} \right\rceil \ge \frac{\tau L}{C_{\min} - r}$$

(cf. (17) for the first inequality in the above equation), and for $\tau_0 < \tau \leq 1$,

$$n = \left\lceil \frac{\tau L}{C_{\min} - r} \right\rceil \geq \frac{\tau L}{C_{\min} - r}.$$

Together with (27), we immediately obtain that $L + z = nr + \tau L \leq nC_{\min}$ for this case. Combining the two cases, we have proven the second inequality in (25).

According to (25), we have $L + z \leq nC_{\min}$. This implies that the $L + z$ symbols in F generated by the source node s, which contain the L source symbols and the key of z symbols, can be multicast to all the sink nodes in T by using the network n times. To elaborate this, we first claim that

$$L + z > (n-1)C_{\min}. \tag{28}$$

- When $0 \leq \tau \leq \tau_0$, it follows from (23) that

$$\begin{aligned}(n-1)C_{\min} &= \left(\left\lceil \frac{L}{C_{\min}} \right\rceil - 1 \right) \cdot C_{\min} \\ &< \frac{L}{C_{\min}} \cdot C_{\min} = L \leq L + z.\end{aligned}$$

- When $\tau_0 < \tau \leq 1$, according to (23), we obtain

$$\begin{aligned}(n-1)C_{\min} &= \left(\left\lceil \frac{\tau L}{C_{\min} - r} \right\rceil - 1 \right) \cdot C_{\min} \\ &< \frac{\tau L}{C_{\min} - r} \cdot C_{\min} \\ &= \tau L + \frac{\tau L}{C_{\min} - r} \cdot r \\ &\leq \tau L + nr \leq L + z,\end{aligned}$$

  where the last two inequalities follow from (23) and (25), respectively.

  Thus, we have proven (28).

Now, we let $b'_1, b'_2, \cdots, b'_{L+z}$ be the $L + z$ source symbols, and divide them into n groups $\mathbf{b}'_1, \mathbf{b}'_2, \cdots, \mathbf{b}'_{n-1}$ and $\mathbf{b}'_n$, where for $i = 1, 2, \cdots, n-1$, $\mathbf{b}'_i$ contains $C_{\min}$ source symbols, and $\mathbf{b}'_n$ contains the remaining $L + z - (n-1)C_{\min}$ source symbols. Here, we note from (25) and (28) that

$$1 \leq L + z - (n-1)C_{\min} \leq C_{\min}.$$

Thus, it suffices to construct, at most, 2 scalar-linear network codes of dimensions $C_{\min}$ and $\omega \triangleq L + z - (n-1)C_{\min}$, respectively, to multicast the $L + z$ source symbols to all the sink nodes.

Let $\mathbf{C}_1$ be a $C_{\min}$-dimensional scalar-linear network code in the network $\mathcal{N}$, of which the global encoding vectors are column vectors $\vec{f}_e$ in $F^{C_{\min}}$ for all $e \in \mathcal{E}$, and let $\mathbf{C}_2$ be an ω-dimensional scalar-linear network code on $\mathcal{N}$, of which the global encoding vectors are column vectors $\vec{f}'_e$ in F^{ω} for all $e \in \mathcal{E}$ (cf. [1,2] and [6]). We use two codes $\mathbf{C}_1$ and $\mathbf{C}_2$ to construct an $(L + z)$-dimensional (vector-) linear network code $\mathbf{C}$ on the network $\mathcal{N}$ such that n symbols are transmitted on each edge $e \in \mathcal{E}$. Specifically, for each $e \in \mathcal{E}$, we let

$$G_e = \begin{bmatrix} \vec{g}_e^{(1)} & \vec{g}_e^{(2)} & \cdots & \vec{g}_e^{(n)} \end{bmatrix}$$

$$= \begin{bmatrix} \vec{f}_e & \vec{0} & \cdots & \vec{0} & \vec{0} \\ \vec{0} & \vec{f}_e & \cdots & \vec{0} & \vec{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vec{0} & \vec{0} & \cdots & \vec{f}_e & \vec{0} \\ \vec{0} & \vec{0} & \cdots & \vec{0} & \vec{f}'_e \end{bmatrix},$$

which is an F-valued $(L+z)\times n$ matrix regarded as the global encoding matrix for the code $\mathbf{C}$.

Next, for an edge $e \in \mathcal{E}$, we use $\langle G_e \rangle$ to denote the vector space spanned by the column vectors of the matrix G_e, i.e.,

$$\langle G_e \rangle \triangleq \langle \vec{g}_e^{(1)}, \vec{g}_e^{(2)}, \cdots, \vec{g}_e^{(n)} \rangle.$$

Furthermore, for a wiretap set $W \in \mathcal{W}_r$, we use G_W to denote the $(L+z)\times nr$ matrix whose column vectors are the column vectors of G_e for all the edges $e \in W$, i.e.,

$$G_W = \Big[G_e : \ e \in W \Big] = \Big[\vec{g}_e^{(1)} \ \ \vec{g}_e^{(2)} \ \ \cdots \ \ \vec{g}_e^{(n)} : \ e \in W \Big],$$

Then, similarly, we use $\langle G_W \rangle$ to denote the vector space spanned by the column vectors of the matrix G_W, i.e.,

$$\langle G_W \rangle \triangleq \langle \vec{g}_e^{(1)}, \vec{g}_e^{(2)}, \cdots, \vec{g}_e^{(n)} : \ e \in W \rangle.$$

Hence, we readily see that

$$\langle G_W \rangle = \sum_{e \in W} \langle G_e \rangle.$$

Now, we claim that there exist F-valued column $(L+z)$-vectors $\vec{u}_i$, $i = 1, 2, \cdots, \tau L$ such that

$$\langle \vec{u}_i : 1 \le i \le \tau L \rangle \bigcap \langle G_W \rangle = \{\vec{0}\}, \ \ \forall\, W \in \mathcal{W}_r. \tag{29}$$

To show this, we prove by induction on $1 \le j \le \tau L$ that if we have $j-1$ linearly independent column vectors $\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_{j-1}$ in F^{L+z} such that

$$\langle \vec{u}_i : 1 \le i \le j-1 \rangle \bigcap \langle G_W \rangle = \{\vec{0}\}, \ \ \forall\, W \in \mathcal{W}_r,$$

then we can choose a column vector $\vec{u}_j \in F^{L+z} \setminus \langle \vec{u}_i : 1 \le i \le j-1 \rangle$ such that

$$\langle \vec{u}_i : 1 \le i \le j \rangle \bigcap \langle G_W \rangle = \{\vec{0}\}, \ \ \forall\, W \in \mathcal{W}_r,$$

provided that $|F| > \binom{|\mathcal{E}|}{r}$. We consider

$$\begin{aligned} &\Big| F^{L+z} \setminus \bigcup_{W \in \mathcal{W}_r} \langle G_W, \ \vec{u}_1, \vec{u}_2, \cdots, \vec{u}_{j-1} \rangle \Big| \\ &\ge |F|^{L+z} - |\mathcal{W}_r| \cdot |F|^{nr+j-1} && (30) \\ &\ge |F|^{nr+\tau L} - |\mathcal{W}_r| \cdot |F|^{nr+\tau L-1} && (31) \\ &\ge |F|^{nr+\tau L-1} \cdot \big(|F| - |\mathcal{W}_r| \big) > 0, && (32) \end{aligned}$$

where the inequality (30) follows because

$$\begin{aligned} \dim \big(\langle G_W, \ \vec{u}_1, \vec{u}_2, \cdots, \vec{u}_{j-1} \rangle \big) &\le \dim \big(\langle G_W \rangle \big) + j - 1 \\ &\le n|W| + j - 1 = nr + j - 1, \ \ \forall\, W \in \mathcal{W}_r; \end{aligned}$$

inequality (31) follows from $L+z \geq nr+\tau L$ according to (25) and inequality (32) follows from

$$|F| > \binom{|\mathcal{E}|}{r} = |\mathcal{W}_r|.$$

Thus, we have proven the existence of such vectors $\vec{u}_i$, $1 \leq i \leq \tau L$ that satisfy the condition (29).

With the vectors $\vec{u}_i$, $1 \leq i \leq \tau L$, we let U be an F-valued $(L+z) \times (L+z)$ invertible matrix such that $\vec{u}_i$, $1 \leq i \leq \tau L$ are the first τL column vectors of U. Furthermore, we consider an $(L+z) \times \tau L$ matrix

$$\widehat{M}_L \triangleq \begin{bmatrix} M_L \\ \mathbf{0} \end{bmatrix}, \tag{33}$$

where $\mathbf{0}$ is the $z \times \tau L$ zero matrix. In particular, when $z = 0$ (cf. (24)), $\widehat{M}_L = M_L$. Recalling that M_L has full column rank, we readily see that $\widehat{M}_L$ also has full column rank. With the full-column-rank matrix $\widehat{M}_L$, we let Γ be an F-valued $(L+z) \times (L+z)$ invertible matrix such that the column vectors of $\widehat{M}_L$ are the first τL column vectors of Γ. Then, we define the matrix

$$Q \triangleq \Gamma \cdot U^{-1}, \tag{34}$$

which is of size $(L+z) \times (L+z)$ and also invertible over F.

Now, we consider the transformation $Q \cdot \mathbf{C}$ of the code $\mathbf{C}$ by the matrix Q, i.e., $Q \cdot \mathbf{C}$ is an F-valued $(L+z)$-dimensional linear network code on the network $\mathcal{N}$, of which all the global encoding matrices are

$$H_e \triangleq Q \cdot G_e, \ \forall\, e \in \mathcal{E},$$

(cf. the transformation of a scalar-linear network code in [6], Section 19.3.1 and [19], Theorem 2). Next, we show that $\widehat{\mathbf{C}} \triangleq Q \cdot \mathbf{C}$ is an admissible F-valued (L, M_L) linear security network code for the security model $\{(L, M_L), r\}$ by verifying the decoding and security conditions.

Remark 1. *We now discuss the computational complexity of our code construction. Our code construction consists of two parts: (i) constructing the two linear network codes* $\mathbf{C}_1$ *and* $\mathbf{C}_2$ *of different dimensions, which are used to multicast all the* $L+z$ *symbols to the sink nodes; and (ii) constructing the transformation matrix* Q*, or equivalently, constructing the* τL *column* $(L+z)$*-vectors* $\vec{u}_i$, $1 \leq i \leq \tau L$ *that satisfy the condition* (29)*. We analyze the complexity of the two parts as follows.*

- *The linear network codes* $\mathbf{C}_1$ *and* $\mathbf{C}_2$ *can be constructed in polynomial time (cf. [4,6,7]);*
- *To obtain the column* $(L+z)$*-vectors* $\vec{u}_i$, $1 \leq i \leq \tau L$ *that satisfy* (29)*, we, in turn, choose* τL *vectors* $\vec{u}_i$ *as follows:*

$$\vec{u}_i \in F^{L+z} \setminus \bigcup_{W \in \mathcal{W}_r} \langle G_W,\ \vec{u}_1, \vec{u}_2, \cdots, \vec{u}_{i-1} \rangle.$$

According to ([35], Lemma 11), the vectors $\vec{u}_i$, $1 \leq i \leq \tau L$ *can be found in*

$$\mathcal{O}\big(\tau L (L+z)^3 |\mathcal{W}_r| + \tau L (L+z) |\mathcal{W}_r|^2\big).$$

By combining the above analysis, our code construction can be implemented in polynomial time.

► **Verification of the decoding condition**:

We continue to consider the output of the source $(\mathbf{b}, \mathbf{k})$, where $\mathbf{b} \in F^L$ is the vector of source symbols, and $\mathbf{k} \in F^z$ is the key. In using the code $\widehat{\mathbf{C}}$, the implementation of the global encoding matrices H_e, $e \in \mathcal{E}$ is equivalent to linearly transforming $(\mathbf{b}\ \mathbf{k})$ into $\boldsymbol{x} \triangleq (\mathbf{b}\ \mathbf{k}) \cdot Q$, then using the code $\mathbf{C}$ to multicast $\boldsymbol{x}$ to all the sink nodes in T.

Since the vector $\boldsymbol{x}$ can be correctly decoded at each $t \in T$ when applying the code $\mathbf{C}$, $(\mathbf{b} \ \ \mathbf{k})$ can be also correctly decoded at each $t \in T$, as can the vector $\mathbf{b}$ of source symbols. Thus, we have verified the decoding condition.

► **Verification of the security condition**:

In order to verify the security condition (4), we need the next lemma, which plays a crucial role in our code construction. This lemma provides a necessary and sufficient condition for a linear security network code to satisfy the security condition (4). For an edge $e \in \mathcal{E}$, $\langle H_e \rangle$ denotes the vector space spanned by the column vectors of H_e, i.e.,

$$\langle H_e \rangle \triangleq \langle \vec{h}_e^{(1)}, \vec{h}_e^{(2)}, \cdots, \vec{h}_e^{(n)} \rangle.$$

Furthermore, for a wiretap set $W \in \mathcal{W}_r$, we let H_W be the $(L+z) \times nr$ matrix that contains all the column vectors of the global encoding matrices H_e for all the edges $e \in W$, i.e.,

$$H_W = \Big[H_e : \ e \in W \Big] = \Big[\vec{h}_e^{(1)} \ \ \vec{h}_e^{(2)} \ \ \cdots \ \ \vec{h}_e^{(n)} : \ e \in W \Big].$$

We let

$$\langle H_W \rangle \triangleq \langle \vec{h}_e^{(1)}, \vec{h}_e^{(2)}, \cdots, \vec{h}_e^{(n)} : \ e \in W \rangle$$

be the vector space spanned by the column vectors of H_W. Evidently,

$$\langle H_W \rangle = \sum_{e \in W} \langle H_e \rangle.$$

Lemma 2. *For the security model $\{(L, M_L), r\}$ over a finite field F with $0 < r < C_{\min}$, let $\widehat{\mathbf{C}}$ be an F-valued (L, M_L) linear security network code, of which the global encoding matrices are $(L+z) \times n$ matrices $H_e = \Big[\vec{h}_e^{(1)} \ \ \vec{h}_e^{(2)} \ \ \cdots \ \ \vec{h}_e^{(n)} \Big]$, $e \in \mathcal{E}$. Then, for the code $\widehat{\mathbf{C}}$, the security condition* (4) *is satisfied if and only if*

$$\langle \widehat{M}_L \rangle \bigcap \langle H_W \rangle = \{\vec{0}\}, \ \ \forall \, W \in \mathcal{W}_r, \tag{35}$$

where $\widehat{M}_L = \left[\begin{smallmatrix} M_L \\ \mathbf{0} \end{smallmatrix} \right]$ is an $(L+z) \times \tau L$ matrix as defined in (33).

Proof. See Appendix B. □

Now, we start to verify the security condition for our code construction. Toward this end, according to Lemma 2, it suffices to verify (35). For the constructed (L, M_L) linear security network code $\widehat{\mathbf{C}}$, we have

$$\langle \vec{u}_i : \ 1 \le i \le \tau L \rangle \bigcap \langle G_W \rangle = \{\vec{0}\}, \ \ \forall \, W \in \mathcal{W}_r \tag{36}$$

(cf. (29)). We recall (34) that $Q = \Gamma \cdot U^{-1}$ is an $(L+z) \times (L+z)$ invertible matrix. Then, according to (36), we immediately obtain

$$\langle Q \cdot \vec{u}_i : \ 1 \le i \le \tau L \rangle \bigcap \langle Q \cdot G_W \rangle = \{\vec{0}\}, \ \ \forall \, W \in \mathcal{W}_r. \tag{37}$$

We note that

$$H_W = \Big[H_e : \ e \in W \Big] = Q \cdot \Big[G_e : \ e \in W \Big] = Q \cdot G_W, \ \ \forall \, W \in \mathcal{W}_r. \tag{38}$$

Furthermore, we write

$$\Big[\vec{u}_1 \ \ \vec{u}_2 \ \ \cdots \ \ \vec{u}_{\tau L} \Big] = U \cdot \begin{bmatrix} I_{\tau L} \\ \mathbf{0} \end{bmatrix},$$

where we recall that $\vec{u}_i$, $1 \leq i \leq \tau L$ are the first τL column vectors of U, $I_{\tau L}$ is the $\tau L \times \tau L$ identity matrix and $\mathbf{0}$ is the $(L + z - \tau L) \times \tau L$ zero matrix. Then, we can see that

$$\begin{aligned} Q \cdot \left[\vec{u}_1 \ \vec{u}_2 \ \cdots \ \vec{u}_{\tau L}\right] &= Q \cdot U \cdot \begin{bmatrix} I_{\tau L} \\ \mathbf{0} \end{bmatrix} \\ &= \Gamma \cdot U^{-1} \cdot U \cdot \begin{bmatrix} I_{\tau L} \\ \mathbf{0} \end{bmatrix} \qquad (39) \\ &= \Gamma \cdot \begin{bmatrix} I_{\tau L} \\ \mathbf{0} \end{bmatrix} \\ &= \widehat{M}_L, \qquad (40) \end{aligned}$$

where (39) follows from $Q = \Gamma \cdot U^{-1}$ (cf. (34)), and (40) follows because the column vectors of $\widehat{M}_L$ are the first τL column vectors of Γ. Combining (38) and (40) with (37), we immediately prove that

$$\left\langle \widehat{M}_L \right\rangle \bigcap \left\langle H_W \right\rangle = \{\vec{0}\}, \ \ \forall\, W \in \mathcal{W}_r.$$

Thus, according to Lemma 2, we have verified the security condition. Combining all the above, Theorem 3 has been proven.

3.4. An Example to Illustrate Our Code Construction

Let $\mathcal{N} = (\mathcal{G}, s, T = \{t_1, t_2\})$ be the butterfly network as depicted in Figure 1. For the security model $r = 1$, we consider two linear-combination security models $\{(2, M_2), 1\}$ and $\{(3, M_3), 1\}$ over the field $\mathbf{F}_3 = \{0, 1, 2\}$, where

$$M_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \ \text{ and } \ M_3 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}. \qquad (41)$$

Namely, in the $\{(2, M_2), 1\}$ security model, the algebraic sum $B_1 + B_2$ of the two source symbols is required to be protected from the wiretapper, and in the $\{(3, M_3), 1\}$ security model, the algebraic sums $B_1 + B_2$ and $B_2 + B_3$ of the source symbols are required to be protected from the wiretapper.

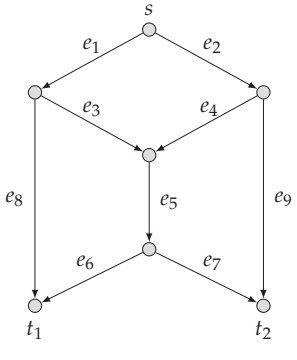


Figure 1. The butterfly network: $\mathcal{N} = (\mathcal{G}, s, T = \{t_1, t_2\})$.

- The security model: $\{(2, M_2), 1\}$.

In this model, the source node s generates two source symbols b_1 and b_2 in $\mathbf{F}_3$, and the algebraic sum $b_1 + b_2$ needs to be protected. According to (41), we have

$$m_2 = \text{Rank}(M_2) = 1, \quad \text{and} \quad \tau = \frac{m_2}{2} = \frac{1}{2} = \tau_0 = \frac{C_{\min} - r}{C_{\min}}.$$

Therefore, we have $0 \leq \tau \leq \tau_0$, i.e., the first case in Theorem 2. Next, we construct an optimal $\mathbf{F}_3$-valued $(2, M_2)$ linear security network code for the $\{(2, M_2), 1\}$ security model, which achieves a security capacity of 2.

According to our code construction, it follows from (23) and (24) that we take

$$n = \left\lceil \frac{L}{C_{\min}} \right\rceil = 1$$

and $z = 0$ because $L = 2 \geq nr + \tau L = 2$. We first consider an $\mathbf{F}_3$-valued two-dimensional scalar-linear network code $\mathbf{C}_1$ on the network $\mathcal{N}$, which is used to multicast two source symbols b_1 and b_2 in $\mathbf{F}_3$ to sink nodes t_1 and t_2. The global encoding matrices (vectors) of $\mathbf{C}_1$ are

$$G_{e_1} = G_{e_3} = G_{e_8} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad G_{e_2} = G_{e_4} = G_{e_9} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \text{ and } G_{e_5} = G_{e_6} = G_{e_7} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Clearly, the code $\mathbf{C}_1$ is not secure for the algebraic sum $b_1 + b_2$ because the wiretapper can obtain $b_1 + b_2$ by accessing the edge e_5 on which $b_1 + b_2$ is transmitted. Based on the code $\mathbf{C}_1$, we now construct a $(2, M_2)$ scalar-linear security network code for the $\{(2, M_2), 1\}$ security model .

Next, we let $\vec{u}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, an $\mathbf{F}_3$-valued column 2-vector such that $\vec{u}_1 \notin \langle G_{e_i} \rangle$, $\forall\, 1 \leq i \leq 9$ (cf. (29)). Then, let $U = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}$, a 2×2 invertible matrix on $\mathbf{F}_3$ such that $\vec{u}_1$ is the first column vector of U. Furthermore, since $z = 0$, we have $\widehat{M}_2 = M_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ (cf. (33)) and let $\Gamma = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, which is a 2×2 invertible matrix on $\mathbf{F}_3$ such that $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ is the first column vector of Γ. According to (34), we calculate $Q = \Gamma \cdot U^{-1} = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}$. Now, we obtain an admissible $\mathbf{F}_3$-valued $(2, M_2)$ scalar-linear security network code $\widehat{\mathbf{C}}_1 = Q \cdot \mathbf{C}_1$, of which the global encoding matrices (vectors) are $H_{e_i} = Q \cdot G_{e_i}$, $1 \leq i \leq 9$. Specifically,

$$H_{e_1} = H_{e_3} = H_{e_8} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad H_{e_2} = H_{e_4} = H_{e_9} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \text{ and } H_{e_5} = H_{e_6} = H_{e_7} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

We use y_{e_i}, which takes values in $\mathbb{F}_3$, to denote the message transmitted on each edge e_i, $1 \leq i \leq 9$. According to the above global encoding matrices of $\widehat{\mathbf{C}}_1$, the messages y_{e_i} $(= (b_1, b_2) \cdot H_e)$ transmitted on the edges e_i, $1 \leq i \leq 9$ are

$$y_{e_1} = y_{e_3} = y_{e_8} = b_1 + 2b_2, \quad y_{e_2} = y_{e_4} = y_{e_9} = b_2, \text{ and } y_{e_5} = y_{e_6} = y_{e_7} = b_1,$$

as depicted in Figure 2. We can readily verify the decoding and security conditions for the code $\widehat{\mathbf{C}}_1$. In particular, in this case, we see that although no randomness is used to randomize the source symbols, the wiretapper cannot obtain any information about the algebraic sum $b_1 + b_2$ when any one edge is eavesdropped.

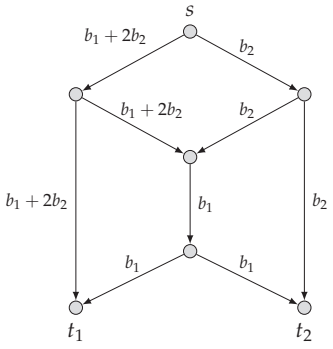


Figure 2. An $\mathbf{F}_3$-valued $(2, M_2)$ scalar-linear security network code for $\{(2, M_2), 1\}$.

- The security model: $\{(3, M_3), 1\}$.

In this model, the source node s generates three source symbols b_1, b_2 and b_3 in $\mathbf{F}_3$, and two algebraic sums $b_1 + b_2$ and $b_2 + b_3$ need to be protected. According to (41), we note that $m_3 = \text{Rank}(M_3) = 2$; thus,

$$\tau = \frac{m_3}{3} = \frac{2}{3} > \tau_0 = \frac{C_{\min} - r}{C_{\min}} = \frac{1}{2}.$$

Therefore, we have $\tau_0 < \tau \leq 1$, i.e., the second case in Theorem 2. Next, we construct an optimal $\mathbf{F}_3$-valued $(3, M_3)$ linear security network code for the $\{(3, M_3), 1\}$ security model, which achieves a security capacity of $3/2$.

According to our code construction, it follows from (23) and (24) that we take

$$n = \left\lceil \frac{\tau L}{C_{\min} - r} \right\rceil = 2$$

and $z = 1$ because $L < nr + \tau L$ according to $L = 3$ and $nr + \tau L = 4$. We consider an $\mathbf{F}_3$-valued four-dimensional (where $4 = L + z$) linear network code $\mathbf{C}_2$ of rate 2, which is used to multicast the three source symbols b_1, b_2 and b_3 and a key k in $\mathbf{F}_3$ to the sink nodes t_1 and t_2. The 4×2 global encoding matrices of $\mathbf{C}_2$ are

$$G_{e_1} = G_{e_3} = G_{e_8} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \; G_{e_2} = G_{e_4} = G_{e_9} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, \text{ and } G_{e_5} = G_{e_6} = G_{e_7} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

We note that the code $\mathbf{C}_2$ is not secure because the wiretapper can obtain some information about $b_1 + b_2$ by accessing the edge e_5 on which $b_1 + b_2$ and $b_3 + k$ are transmitted. Based on the code $\mathbf{C}_2$, we now construct a linear secure network code for the $\{(3, M_3), 1\}$ security model.

Let

$$\vec{u}_1 = \begin{bmatrix} 1 \\ 2 \\ 0 \\ 0 \end{bmatrix} \text{ and } \vec{u}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \end{bmatrix}$$

be two $\mathbf{F}_3$-valued column 4-vectors such that $\langle \vec{u}_1, \vec{u}_2 \rangle \bigcap \langle G_{e_i} \rangle = \{\vec{0}\}$, $\forall\, 1 \leq i \leq 9$ (cf. (29)). Then, let

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix}$$

be a 4×4 invertible matrix on $\mathbf{F}_3$ such that $\vec{u}_1$ and $\vec{u}_2$ are the first two column vectors of U. Furthermore, since $z = 1$, as mentioned above, we have

$$\widehat{M}_3 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

(cf. (33)). Also let

$$\Gamma = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

be a 4×4 invertible matrix on $\mathbf{F}_3$ such that the column vectors of $\widehat{M}_L$ are the first two column vectors of Γ. According to (34), we calculate

$$Q = \Gamma \cdot U^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix},$$

Now, we obtain an admissible $\mathbf{F}_3$-valued $(3, M_3)$ linear security network code $\widehat{\mathbf{C}}_2 = Q \cdot \mathbf{C}_2$, of which the 4×2 global encoding matrices are $H_{e_i} = Q \cdot G_{e_i}$, $1 \le i \le 9$; specifically,

$$H_{e_1} = H_{e_3} = H_{e_8} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix},\ H_{e_2} = H_{e_4} = H_{e_9} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix},\ \text{and } H_{e_5} = H_{e_6} = H_{e_7} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 2 & 1 \\ 0 & 2 \end{bmatrix}.$$

We use y_{e_i}, which takes values in $\mathbb{F}_3^2$, to denote the message transmitted on each edge e_i, $1 \le i \le 9$. According to the above global encoding matrices of $\widehat{\mathbf{C}}_2$, the messages y_{e_i} $(= (b_1, b_2, b_3, k) \cdot H_e)$ transmitted on the edges e_i, $1 \le i \le 9$ are

$$\begin{aligned} y_{e_1} = y_{e_3} = y_{e_8} &= (b_1 + b_2 + b_3,\ b_2 + b_3 + k), \\ y_{e_2} = y_{e_4} = y_{e_9} &= (b_3,\ k),\ \text{ and } \ y_{e_5} = y_{e_6} = y_{e_7} = (b_1 + b_2 + 2b_3,\ b_2 + b_3 + 2k), \end{aligned}$$

as depicted in Figure 3.

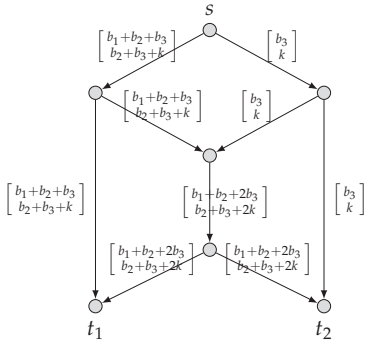


Figure 3. An $\mathbf{F}_3$-valued $(3, M_3)$ linear-security network code for $\{(3, M_3), 1\}$.

For the $\{(2, M_2), 1\}$ and $\{(3, M_3), 1\}$ security models, as discussed in the above example, according to Theorem 3, admissible linear security network codes with rates of 2 and 3/2, respectively, can be constructed if the field size is $|F| > \max\{|T|, \binom{|\mathcal{E}|}{r}\} = 9$. However,

we see in the example that the field $\mathbf{F}_3$, of size 3 is sufficient for our code construction. This implies that the $\max\left\{|T|, \binom{|\mathcal{E}|}{r}\right\}$ bound in Theorem 3 on the field size is only sufficient but not necessary for our code construction.

4. Asymptotic Behavior of the Security Capacity

In this section, we investigate the asymptotic behavior of the security capacity. For a fixed network $\mathcal{N}$ and a security level r, we consider a sequence of the $\{(L, M_L), r\}$, $L = 1, 2, \cdots$ security models. The following theorem characterizes the asymptotic behavior of the security capacity for a sequence of security models $\{(L, M_L), r\}$, $L = 1, 2, \cdots$.

Theorem 4. *Consider a sequence of linear-combination security models $\{(L, M_L), r\}$ over a finite field F for $L = 1, 2, \cdots$, where $0 < r < C_{\min}$ and $|F| > \max\left\{|T|, \binom{|\mathcal{E}|}{r}\right\}$. $\mathcal{C}_{L,M_L}$ denotes the security capacity for each model $\{(L, M_L), r\}$. Let*

$$\tau_L = \frac{m_L}{L},\ L = 1, 2, \cdots \text{ and } \tau_0 = \frac{C_{\min} - r}{C_{\min}},$$

where $m_L = \mathrm{Rank}(M_L)$ for $L = 1, 2, \cdots$.

- *If $\tau_L \leq \tau_0 + o(1)$, then,*
$$\lim_{L\to\infty} \mathcal{C}_{L,M_L} = C_{\min}.$$
- *If $\tau_L = \kappa + o(1)$, with κ satisfying $\tau_0 < \kappa \leq 1$, then,*
$$\lim_{L\to\infty} \mathcal{C}_{L,M_L} = \kappa^{-1} \cdot (C_{\min} - r).$$

Proof. We first consider the case of $\tau_L \leq \tau_0 + o(1)$. Then, there exists a non-negative sequence, a_L, $L = 1, 2, \cdots$ with $\lim_{L\to\infty} a_L = 0$, such that

$$\tau_L \leq \tau_0 + a_L, \quad L = 1, 2, \cdots. \tag{42}$$

We now use Theorem 2 to show that

$$\mathcal{C}_{L,M_L} \geq \frac{L}{\lceil (\tau_0 + a_L) \cdot L/(C_{\min} - r) \rceil}. \tag{43}$$

To show this, consider the following two cases:

- If $0 \leq \tau_L \leq \tau_0$, it follows from (19) that
$$\mathcal{C}_{L,M_L} = \frac{L}{\lceil L/C_{\min} \rceil} = \frac{L}{\lceil \tau_0 \cdot L/(C_{\min} - r) \rceil} \geq \frac{L}{\lceil (\tau_0 + a_L) \cdot L/(C_{\min} - r) \rceil};$$
- If $\tau_0 < \tau_L \leq 1$, then we obtain
$$\mathcal{C}_{L,M_L} = \frac{L}{\lceil \tau_L \cdot L/(C_{\min} - r) \rceil} \geq \frac{L}{\lceil (\tau_0 + a_L) \cdot L/(C_{\min} - r) \rceil},$$
where the equality follows from (20), and the inequality follows from (42).

Combining (43) and (7) with Lemma 1, we further obtain that for each pair (L, M_L),

$$\frac{L}{\lceil (\tau_0 + a_L) \cdot L/(C_{\min} - r) \rceil} \leq \mathcal{C}_{L,M_L} \leq \frac{L}{\lceil L/C_{\min} \rceil} \leq C_{\min}. \tag{44}$$

We note that

$$\lim_{L\to\infty} \frac{L}{\lceil (\tau_0 + a_L) \cdot L/(C_{\min} - r) \rceil} = C_{\min}.$$

Together with (44), we have thus proven that

$$\lim_{L\to\infty} \mathcal{C}_{L,M_L} = C_{\min}.$$

Next, we consider a case in which $\tau_L = \kappa + o(1)$, where $\tau_0 < \kappa \leq 1$. Then, there exists a sequence b_L, $L = 1, 2, \cdots$ satisfying $\lim_{L\to\infty} b_L = 0$ such that

$$\tau_L = \kappa + b_L, \quad L = 1, 2, \cdots.$$

Here, we note that b_L may be negative. Together with $\kappa > \tau_0$ and $\lim_{L\to\infty} b_L = 0$, there exists a positive integer L_0 such that for each $L \geq L_0$,

$$|b_L| < \kappa - \tau_0, \text{ i.e., } \tau_0 - \kappa < b_L < \kappa - \tau_0,$$

which implies that

$$\tau_L = \kappa + b_L > \tau_0, \quad \forall\, L \geq L_0.$$

According to (20) in Theorem 2, we have

$$\mathcal{C}_{L,M_L} = \frac{L}{\lceil (\kappa + b_L) \cdot L/(C_{\min} - r) \rceil},$$

so that

$$\lim_{L\to\infty} \mathcal{C}_{L,M_L} = \kappa^{-1} \cdot (C_{\min} - r).$$

Thus, the theorem is proven. □

According to Theorem 4, we can see that for a sequence of security models $\{(L, M_L), r\}$, $L = 1, 2, \cdots$ that satisfies $\tau_L \leq \tau_0 + o(1)$ or $\tau_L = \kappa + o(1)$, where $\tau_0 < \kappa \leq 1$, our code construction is *asymptotically optimal*, i.e.,

$$\lim_{L\to\infty} R(\widehat{\mathbf{C}}_{L,M_L}) = \lim_{L\to\infty} \mathcal{C}_{L,M_L}, \tag{45}$$

where $\widehat{\mathbf{C}}_{L,M_L}$ is the code constructed for each model $\{(L, M_L), r\}$ by our code construction. To illustrate this, in the following, we consider several specific sequences of security models.

First, we consider a sequence of security models $\{(L, M_L), r\}$, $L = 1, 2, \cdots$ in which all the ranks $\text{Rank}(M_L)$, $L = 1, 2, \cdots$ are upper-bounded by a constant, such as m, e.g., the security constraint of multiple algebraic sums,

$$\sum_{\substack{i\in[L]:\\ i\equiv j (\text{mod } m)}} B_i, \quad j = 1, 2, \cdots, m$$

as discussed in the last paragraph of Section 2. With this, we have

$$\lim_{L\to\infty} \frac{m_L}{L} = 0,$$

which implies the inequality $\tau_L = m_L/L \leq \tau_0 + o(1)$. It then follows from the first case of Theorem 4 that

$$\lim_{L\to\infty} \mathcal{C}_{L,M_L} = C_{\min}.$$

Next, we show that our code construction is asymptotically optimal. We first note that

$$\tau_L = \frac{m_L}{L} \leq \frac{C_{\min} - r}{C_{\min}} = \tau_0, \quad \forall\, L \geq C_{\min} \cdot \left\lceil \frac{m}{C_{\min} - r} \right\rceil.$$

Together with the first case of Theorem 3 (cf. (21)), the constructed code $\widehat{\mathbf{C}}_{L,M_L}$ achieves a rate of $R(\widehat{\mathbf{C}}_{L,M_L}) = \frac{L}{\lceil L/C_{\min} \rceil}$. This immediately implies that the equality (45) is satisfied, namely that our code construction is asymptotically optimal for this example.

Next, we consider a sequence of security models $\{(L, M_L), r\},\ L = 1, 2, \cdots$ in which all the ranks $m_L = \mathrm{Rank}(M_L)$ satisfy

$$m_L = \lceil \kappa \cdot L \rceil,\ \ L = 1, 2, \cdots .$$

We note that the sequence of $m_L,\ L = 1, 2, \cdots$ is not upper-bounded. According to Theorem 4, we can obtain the asymptotic behavior of the security capacity for the sequence of models $\{(L, M_L), r\},\ L = 1, 2, \cdots$ as follows:

$$\lim_{L\to\infty} \mathcal{C}_{L,M_L} = \begin{cases} C_{\min}, & \text{if } 0 < \kappa \le \tau_0, \\ \kappa^{-1} \cdot (C_{\min} - r), & \text{if } \tau_0 < \kappa < 1. \end{cases} \tag{46}$$

Furthermore, it follows from Theorem 3 that

$$\lim_{L\to\infty} R(\widehat{\mathbf{C}}_{L,M_L}) = \begin{cases} C_{\min}, & \text{if } 0 < \kappa \le \tau_0, \\ \kappa^{-1} \cdot (C_{\min} - r), & \text{if } \tau_0 < \kappa < 1, \end{cases} \tag{47}$$

where $\widehat{\mathbf{C}}_{L,M_L}$ is the code constructed for each model $\{(L, M_L), r\}$ by the code construction. Comparing (46) and (47), we immediately see that the equality (45) holds, which shows that our code construction is asymptotically optimal for this example.

Finally, we consider the special sequence of security models $\{(L, M_L), r\}$ for $L = 1, 2, \cdots$, where $m_L = L$, i.e., $\tau_L = m_L/L = 1$ for all $L = 1, 2, \cdots$. This linear-combination security constraint is equivalent to protecting all the source symbols from the wiretapper, so each model $\{(L, M_L), r\}$ is equivalent to the standard secure-network coding model. Thus, we have

$$\lim_{L\to\infty} \mathcal{C}_{L,M_L} = C_{\min} - r. \tag{48}$$

On the other hand, for each pair (L, M_L), it follows from $\tau_L = 1$ and Theorem 3 that the (L, M_L) linear security network code $\widehat{\mathbf{C}}_{L,M_L}$ constructed by our code construction has a rate of

$$R(\widehat{\mathbf{C}}_{L,M_L}) = \frac{L}{\lceil L/(C_{\min} - r) \rceil}.$$

This implies that

$$\lim_{L\to\infty} R(\widehat{\mathbf{C}}_{L,M_L}) = C_{\min} - r. \tag{49}$$

Combining (48) and (49), we see that the equality (45) holds, and thus, our code construction is also asymptotically optimal for this example.

5. Conclusions

In this paper, we put forward the model of multiple linear-combination security network coding, which is specified by the security level, the number of source symbols and the linear-combination security constraint. We fully characterized the security capacity for any such security model in terms of the ratio τ of the rank of the linear-combination security constraint to the number of source symbols. Also, we developed a construction of linear security network codes. The code construction is applicable to any security model, and the constructed code achieves the security capacity. We also determined a threshold value τ_0 such that there is no penalty on the security capacity compared with the capacity without any security consideration when the ratio τ is not larger than τ_0. Finally, we analyzed the asymptotic behavior of the security capacity for a sequence of linear-combination security

models and fully characterized the asymptotic behavior of the security capacity. We also showed that our code construction is asymptotically optimal.

Author Contributions: All authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: The work of Y. Bai and X. Guang was supported in part by the National Key Research and Development Program of China (grant number 2022YFA1005000), the Natural Science Foundation of China (grant number 12141108), the Natural Science Foundation of Tianjin, China (grant number 20JCYBJC01390), and the Fundamental Research Funds for the Central Universities of China (grant number NKU 050-63233070). The work of R. W. Yeung was supported in part by a fellowship award from the Research Grants Council of the Hong Kong Special Administrative Region, China (grant number CUHK SRFS2223-4S03)..

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: A special case of the results presented in this paper was discussed in our submission to the 2023 IEEE Information Theory Workshop. We thank an anonymous reviewer for pointing out the relation between our submission and the work of Bhattad and Narayanan [23].

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. A Related Work by Bhattad and Narayanan

A model related to the current work is that considered by Bhattad and Narayanan [23], the general case of which is given as follows. On the network $\mathcal{N}$, the single source node s generates L $L \leq C_{\min}$ source symbols, as denoted by $X_1, X_2, \cdots, X_L$, over a finite field F, which are required to be multicast to all the sink nodes in T. Let U_p, $1 \leq p \leq P$ be P subsets of the L source symbols and G_p, $1 \leq p \leq P$ be another P subsets of the L source symbols. The security requirement is specified by the P pairs (U_p, G_p), $1 \leq p \leq P$ as follows. The wiretapper, who can access any one wiretap set W in a collection $\mathcal{W}$ of wiretap sets, is not allowed to obtain any information about U_p, given G_p for each $p = 1, 2, \cdots, P$, i.e., for each $p = 1, 2, \cdots, P$,

$$I\big(U_p; Y_W \big| G_p\big) = 0 \quad \text{or} \quad H\big(U_p \big| G_p\big) = H\big(U_p \big| Y_W, G_p\big), \quad \forall\, W \in \mathcal{W}, \tag{A1}$$

where $Y_W = (Y_e : \ e \in W)$, with Y_e being the random variable transmitted on the edge e. In particular, when taking $P = L$, $U_p = \{X_p\}$ and $G_p = \emptyset$ for $1 \leq p \leq P$, the security requirement (A1) becomes

$$I\big(X_p; Y_W\big) = 0, \ \ \forall\, 1 \leq p \leq P \ \text{ and } \ W \in \mathcal{W}.$$

This type of security requirement is called *weak security* in [23].

For the above model, the main focus in [23] is on how to find a suitable linear transformation of the L source symbols for a given linear network code to obtain a secure linear network code such that the security requirement (A1) is satisfied. Theorem 3 in [23], the most general result presented in the paper, asserts the existence of such a linear transformation when a given condition is satisfied. We state this theorem as follows.

Theorem A1 ([23], Theorem 3)**.** *Consider an L-dimensional $L \leq C_{\min}$ network code* **C** *over a finite field F and a collection of wiretap sets $\mathcal{W}$ in which $r = \max_{W \in \mathcal{W}} |W|$. Let (U_p, G_p), $1 \leq p \leq P$ be P pairs of subsets U_p and G_p of the L source symbols, which specify the security requirement. If*

$$\max_{1 \leq p \leq P} \big(|U_p| + |G_p|\big) \leq L - r, \tag{A2}$$

then there exists a linear transformation of the source symbols as a precoding on the linear network code **C** *such that the security requirement* (A1) *is satisfied.*

We now go back to the linear-combination security model $\{(L, M_L), r\}$ discussed in the current paper. Consider the first case $0 \leq \tau \leq \tau_0$ in Theorem 2, where we recall that $\tau = m_L/L$, where $m_L = \text{Rank}(M_L)$ and $\tau_0 = (C_{\min} - r)/C_{\min}$. Then, we can apply the approach of the linear transformation in Theorem A1 (cf. [23] for details) to obtain a (L, M_L) linear security network code, provided that the following two additional conditions on the model parameters are satisfied:

$$0 \leq \tau \leq \frac{L-r}{L} \ (\leq \tau_0) \quad \text{and} \quad L \leq C_{\min}. \tag{A3}$$

Specifically, we consider a linear-combination security model $\{(L, M_L), r\}$ satisfying the conditions (A3), where the L source symbols $B_1, B_2, \cdots, B_L$ are required to be multicast to all the sink nodes in T, and the multiple linear combinations $\mathbf{B} \cdot M_L$, where $\mathbf{B} = (B_1, B_2, \cdots, B_L)$ are required to be protected from the wiretapper. We first linearly transform $\mathbf{B} = (B_1, B_2, \cdots, B_L)$ to $(X_1, X_2, \cdots, X_L)$ using an $L \times L$ invertible matrix M whose left $L \times m_L$ submatrix is equal to M_L. Then, we have

$$(X_1, X_2, \cdots, X_L) = (B_1, B_2, \cdots, B_L) \cdot M,$$

where

$$(X_1, X_2, \cdots, X_{m_L}) = (B_1, B_2, \cdots, B_L) \cdot M_L.$$

We now apply Theorem A1 as follows. Take $X_1, X_2, \cdots, X_L$ as the source symbols. Let $U = \{X_1, X_2, \cdots, X_{m_L}\}$, $G = \emptyset$ and $\mathcal{W} = \mathcal{W}_r$. According to $\tau \leq (L-r)/L$, we see that $|U| + |G| = m_L \leq L - r$, which satisfies the condition (A2) in Theorem A1. It therefore follows from Theorem A1 that we can construct a linear secure network code such that $X_1, X_2, \cdots, X_L$ can be multicast to all the sink nodes in T, and the wiretapper cannot obtain any information about U, i.e.,

$$I\big(X_1, X_2, \cdots, X_{m_L}; Y_W\big) = 0, \ \ \forall\, W \in \mathcal{W}_r,$$

or, equivalently,

$$I\big(\mathbf{B} \cdot M_L; Y_W\big) = 0, \ \ \forall\, W \in \mathcal{W}_r.$$

Hence, we obtain an admissible (L, M_L) linear security network code for the security model $\{(L, M_L), r\}$.

However, the second case $\tau_0 < \tau \leq 1$ in Theorem 2 cannot be handled by the approach proposed in [23]. Specifically, according to $\tau > \tau_0$, we have

$$\frac{m_L}{L} = \tau > \tau_0 = \frac{C_{\min} - r}{C_{\min}} \geq \frac{L-r}{L}.$$

This implies that $|U| + |G| = m_L > L - r$, which does not satisfy the condition (A2) in Theorem A1. Hence, we cannot apply the linear transformation approach for the case of $\tau_0 < \tau \leq 1$.

Appendix B. Proof of Lemma 2

We first prove the "only if" part by contradiction. Suppose, on the contrary, that there exists a wiretap set $W \in \mathcal{W}_r$ such that

$$\langle \widehat{M}_L \rangle \bigcap \langle H_W \rangle \neq \{\vec{0}\}. \tag{A4}$$

In the following, we prove that

$$I\big(\mathbf{B} \cdot M_L;\ \mathbf{Y}_W\big) > 0, \tag{A5}$$

which contradicts the security condition (4) for the code $\widehat{\mathbf{C}}$.

According (A4), there exist two non-zero column vectors $\vec{w} \in F^{n|W|}$ $(= F^{nr})$ and $\vec{u} \in F^{\tau L}$ such that

$$H_W \cdot \vec{w} = \widehat{M}_L \cdot \vec{u} \neq \vec{0}, \tag{A6}$$

where $\vec{0}$ is the zero-column $(L+z)$-vector. Then, we obtain

$$\begin{aligned}
&I\Big(\mathbf{B}\cdot M_L;\ \mathbf{Y}_W\Big)\\
&= I\Big(\mathbf{B}\cdot M_L;\ (\mathbf{B}\ \mathbf{K})\cdot H_W\Big) &\text{(A7)}\\
&= H\Big(\mathbf{B}\cdot M_L\Big) - H\Big(\mathbf{B}\cdot M_L\big|(\mathbf{B}\ \mathbf{K})\cdot H_W\Big)\\
&= H\Big(\mathbf{B}\cdot M_L\Big) - H\Big(\mathbf{B}\cdot M_L\cdot\big|(\mathbf{B}\ \mathbf{K})\cdot H_W, (\mathbf{B}\ \mathbf{K})\cdot H_W\cdot\vec{w}\Big)\\
&\geq H\Big(\mathbf{B}\cdot M_L\Big) - H\Big(\mathbf{B}\cdot M_L\big|(\mathbf{B}\ \mathbf{K})\cdot H_W\cdot\vec{w}\Big)\\
&= I\Big(\mathbf{B}\cdot M_L;\ (\mathbf{B}\ \mathbf{K})\cdot H_W\cdot\vec{w}\Big)\\
&= I\Big(\mathbf{B}\cdot M_L;\ (\mathbf{B}\ \mathbf{K})\cdot\widehat{M}_L\cdot\vec{u}\Big) &\text{(A8)}\\
&= I\Big(\mathbf{B}\cdot M_L;\ \mathbf{B}\cdot M_L\cdot\vec{u}\Big)\\
&= H\Big(\mathbf{B}\cdot M_L\cdot\vec{u}\Big) - H\Big(\mathbf{B}\cdot M_L\cdot\vec{u}\big|\mathbf{B}\cdot M_L\Big)\\
&= H\Big(\mathbf{B}\cdot M_L\cdot\vec{u}\Big) > 0, &\text{(A9)}
\end{aligned}$$

where the equality in (A7) follows from $\mathbf{Y}_W = (\mathbf{B}\ \mathbf{K})\cdot H_W$, the equality in (A8) follows from (A6), the equality in (A9) follows from $H\Big(\mathbf{B}\cdot M_L\cdot\vec{u}\big|\mathbf{B}\cdot M_L\Big) = 0$ and the inequality in (A9) follows from $M_L\cdot\vec{u} \neq \vec{0}$ because $\vec{u} \neq \vec{0}$, and M_L has full column rank. Thus, the inequality in (A5) is proven.

Next, we prove the "if" part. According to the security condition (4), we prove that

$$H\Big(\mathbf{B}\cdot M_L\big|\mathbf{Y}_W\Big) = H\Big(\mathbf{B}\cdot M_L\Big),\ \ \forall\, W \in \mathcal{W}_r \tag{A10}$$

if the condition in (35) is satisfied. To prove (A10), it suffices to show that for each $W \in \mathcal{W}_r$, the equality

$$\Pr\Big(\mathbf{B}\cdot M_L = \vec{x}\big|\mathbf{Y}_W = \vec{y}\Big) = \Pr\Big(\mathbf{B}\cdot M_L = \vec{x}\Big) \tag{A11}$$

is satisfied for any $\vec{x} \in F^{\tau L}$ row vector and any $\vec{y} \in F^{nr}$ row vector such that $\Pr\left(\mathbf{Y}_W = \vec{y}\right) > 0$, i.e., there exists a pair $(\mathbf{b}\ \mathbf{k})$ of a vectors of source symbols $\mathbf{b} \in F^L$ and a key $\mathbf{k} \in F^z$ such that $(\mathbf{b}\ \mathbf{k})\cdot H_W = \vec{y}$.

We recall that

$$\Pr\Big(\mathbf{B}\cdot M_L = \vec{x}\Big) = \frac{1}{|F|^{\tau L}},\ \ \forall\, \vec{x} \in F^{\tau L}$$

(cf. (10)). Thus, we only need to prove that for each $W \in \mathcal{W}_r$,

$$\Pr\Big(\mathbf{B}\cdot M_L = \vec{x}\big|\mathbf{Y}_W = \vec{y}\Big) = \frac{1}{|F|^{\tau L}}$$

for any $\vec{x} \in F^{\tau L}$ and $\vec{y} \in F^{nr}$ such that $\Pr\left(\mathbf{Y}_W = \vec{y}\right) > 0$. We now consider

$$\begin{aligned}
&\Pr\Big(\mathbf{B}\cdot M_L = \vec{x}\big|\mathbf{Y}_W = \vec{y}\Big)\\
&= \frac{\Pr\left(\mathbf{B}\cdot M_L = \vec{x}, \mathbf{Y}_W = \vec{y}\right)}{\Pr\left(\mathbf{Y}_W = \vec{y}\right)}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{\Pr\left((\mathbf{B}\ \mathbf{K})\cdot\widehat{M}_L = \vec{x},\ (\mathbf{B}\ \mathbf{K})\cdot H_W = \vec{y}\right)}{\Pr\left((\mathbf{B}\ \mathbf{K})\cdot H_W = \vec{y}\right)} \\
&= \frac{\Pr\left((\mathbf{B}\ \mathbf{K})\cdot\left[\widehat{M}_L\ H_W\right] = (\vec{x}\ \ \vec{y})\right)}{\Pr\left((\mathbf{B}\ \mathbf{K})\cdot H_W = \vec{y}\right)} \\
&= \frac{\sum_{(\mathbf{b}\ \mathbf{k})\in F^L\times F^z:\ (\mathbf{b}\ \mathbf{k})[\widehat{M}_L\ H_W]=(\vec{x}\ \vec{y})} \Pr\left(\mathbf{B}=\mathbf{b},\ \mathbf{K}=\mathbf{k}\right)}{\sum_{(\mathbf{b}'\ \mathbf{k}')\in F^L\times F^z:\ (\mathbf{b}'\ \mathbf{k}')H_W=\vec{y}} \Pr\left(\mathbf{B}=\mathbf{b}',\ \mathbf{K}=\mathbf{k}'\right)} \\
&= \frac{\#\left\{(\mathbf{b}\ \mathbf{k})\in F^L\times F^z : (\mathbf{b}\ \mathbf{k})\cdot\left[\widehat{M}_L\ H_W\right] = (\vec{x}\ \ \vec{y})\right\}}{\#\left\{(\mathbf{b}'\ \mathbf{k}')\in F^L\times F^z : (\mathbf{b}'\ \mathbf{k}')\cdot H_W = \vec{y}\right\}}, \qquad \text{(A12)}
\end{aligned}
$$

where we use "#$\{\cdot\}$" to denote the cardinality of the set, and the equality (A12) follows because $\mathbf{B}$ and $\mathbf{K}$ are independently and uniformly distributed on F^L and F^z, respectively. Furthermore,

- For the denominator in (A12), we have

$$\#\left\{(\mathbf{b}'\ \mathbf{k}')\in F^L\times F^z : (\mathbf{b}'\ \mathbf{k}')\cdot H_W = \vec{y}\right\} = |F|^{L+z-\mathrm{Rank}(H_W)}; \qquad \text{(A13)}$$

- For the numerator in (A12), we have

$$
\begin{aligned}
&\#\left\{(\mathbf{b}\ \mathbf{k})\in F^L\times F^z : (\mathbf{b}\ \mathbf{k})\cdot\left[\widehat{M}_L\ H_W\right] = (\vec{x}\ \ \vec{y})\right\} \\
&= |F|^{L+z-\mathrm{Rank}\left(\left[\widehat{M}_L\ H_W\right]\right)} \\
&= |F|^{L+z-\mathrm{Rank}(H_W)-\tau L}, \qquad \text{(A14)}
\end{aligned}
$$

where the equality (A14) follows from the following condition: $\left\langle\widehat{M}_L\right\rangle\bigcap\left\langle H_W\right\rangle = \{\vec{0}\}$ (cf. (35)).

Combining (A13) and (A14) with (A12), we immediately prove that

$$\Pr\left(\mathbf{B}\cdot M_L = \vec{x}\middle|\mathbf{Y}_W = \vec{y}\right) = \frac{1}{|F|^{\tau L}},$$

which implies the equality in (A11). The "if" part is also proven. We have thus proven the lemma.

References

1. Ahlswede, R.; Cai, N.; Li, S.-Y.; Yeung, R.W. Network Information Flow. *IEEE Trans. Inf. Theory* **2000**, *46*, 1204–1216. [CrossRef]
2. Li, S.-Y.R.; Yeung, R.W.; Cai, N. Linear Network Coding. *IEEE Trans. Inf. Theory* **2003**, *49*, 371–381. [CrossRef]
3. Koetter, R.; Médard, M. An Algebraic Approach to Network Coding. *IEEE/ACM Trans. Netw.* **2003**, *11*, 782–795. [CrossRef]
4. Jaggi, S.; Sanders, P.; Chou, P.A.; Effros, M.; Egner, S.; Jain, K.; Tolhuizen, L.M. Polynomial Time Algorithms for Multicast Network Code Construction. *IEEE Trans. Inf. Theory* **2005**, *51*, 1973–1982. [CrossRef]
5. Ho, T.; Lun, D. *Network Coding: An Introduction*; Cambridge University Press: Cambridge, UK, 2008.
6. Yeung, R.W. *Information Theory and Network Coding*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2008.
7. Yeung, R.W.; Li, S.-Y.R.; Cai, N.; Zhang, Z. Network Coding Theory Part I: Single Source. *Found. Trends Commun. Inf. Theory* **2006**, *2*, 241–329. [CrossRef]
8. Yeung, R.W.; Li, S.-Y.R.; Cai, N.; Zhang, Z. Network Coding Theory Part II: Multiple Source. *Found. Trends Commun. Inf. Theory* **2006**, *2*, 330–381. [CrossRef]
9. Fragouli, C.; Soljanin, E. Network Coding Fundamentals. *Found. Trends Netw.* **2007**, *2*, 1–133. [CrossRef]
10. Fragouli, C.; Soljanin, E. Network Coding Applications. *Found. Trends Netw.* **2008**, *2*, 135–269. [CrossRef]
11. Cai, N.; Yeung, R.W. Secure Network Coding on a Wiretap Network. *IEEE Trans. Inf. Theory* **2011**, *57*, 424–435. [CrossRef]
12. El Rouayheb, S.; Soljanin, E.; Sprintson, A. Secure Network Coding for Wiretap Networks of Type II. *IEEE Trans. Inf. Theory* **2012**, *58*, 1361–1371. [CrossRef]
13. Silva, D.; Kschischang, F.R. Universal Secure Network Coding via Rank-Metric Codes. *IEEE Trans. Inf. Theory* **2011**, *57*, 1124–1135. [CrossRef]
14. Cai, N.; Chan, T. Theory of Secure Network Coding. *Proc. IEEE* **2011**, *99*, 421–437.

15. Cui, T.; Ho, T.; Kliewer, J. On Secure Network Coding With Nonuniform or Restricted Wiretap Sets. *IEEE Trans. Inf. Theory* **2013**, *59*, 166–176. [CrossRef]
16. Cheng, F.; Yeung, R.W. Performance Bounds on a Wiretap Network with Arbitrary Wiretap Sets. *IEEE Trans. Inf. Theory* **2014**, *60*, 3345–3358. [CrossRef]
17. Fragouli, C.; Soljanin, E. (Secure) Linear Network Coding Multicast. *Des. Codes Cryptogr.* **2016**, *78*, 269–310. [CrossRef]
18. Guang, X.; Yeung, R.W. Alphabet Size Reduction for Secure Network Coding: A Graph Theoretic Approach. *IEEE Trans. Inf. Theory* **2018**, *64*, 4513–4529. [CrossRef]
19. Guang, X.; Yeung, R.W.; Fu, F.-W. Local-Encoding-Preserving Secure Network Coding. *IEEE Trans. Inf. Theory* **2020**, *66*, 5965–5994. [CrossRef]
20. Cai, N.; Yeung, R.W. A Security Condition for Multi-Source Linear Network Coding. In Proceedings of the 2007 IEEE International Symposium on Information Theory, Nice, France, 24–29 June 2007; pp. 561–565.
21. Chan, T.; Grant, A. Capacity Bounds for Secure Network Coding. In Proceedings of the 2008 Australian Communications Theory Workshop, Christchurch, New Zealand, 30 January–1 February 2008; pp. 95–100.
22. Zhang, Z.; Yeung, R.W. A General Security Condition for Multi-Source Linear Network Coding. In Proceedings of the 2009 IEEE International Symposium on Information Theory, Seoul, Republic of Korea, 28 June–3 July 2009; pp. 1155–1158.
23. Bhattad, K.; Narayanan, K.R. Weakly Secure Network Coding. In Proceedings of the First Workshop on Network Coding, Theory and Applications, Riva del Garda, Italy, 7 April 2005 ; pp. 8–20.
24. Harada, K.; Yamamoto, H. Strongly Secure Linear Network Coding. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2008**, *91*, 2720–2728. [CrossRef]
25. Cai, N.; Hayashi, M. Secure Network Code for Adaptive and Active Attacks with No-randomness in Intermediate Nodes. *IEEE Trans. Inf. Theory* **2020**, *66*, 1428–1448. [CrossRef]
26. Hayashi, M.; Cai, N. Secure Non-Linear Network Code over A One-Hop Relay Network. *IEEE Trans. Inf. Theory* **2021**, *2*, 296–305. [CrossRef]
27. Agarwal, G. K.; Cardone, M.; Fragouli, C. On Secure Network Coding for Multiple Unicast Traffic. *IEEE Trans. Inf. Theory* **2020**, *66*, 5204–5227. [CrossRef]
28. Zhou, H.; El Gamal, A. Network Information Theoretic Security With Omnipresent Eavesdropping. *IEEE Trans. Inf. Theory* **2021**, *67*, 8280–8299. [CrossRef]
29. Shannon, C.E. Communication Theory of Secrecy Systems. *Bell Syst. Tech. J.* **1949**, 28, 656–715. [CrossRef]
30. Blakley, G.R. Safeguarding Cryptographic Keys. In Proceedings of the Managing Requirements Knowledge, International Workshop on, New York, NY, USA, 4–7 June 1979; p. 313.
31. Shamir, A. How to Share a Secret. *Commun. ACM* **1979**, 22, 612–613. [CrossRef]
32. Ozarow, L.H.; Wyner, A.D. Wire-Tap Channel II. *AT&T Bell Lab. Tech. J.* **1984**, 63, 2135–2157.
33. Feldman, J.; Malkin, T.; Servedio, R.A.; Stein, C. On the Capacity of Secure Network Coding. In Proceedings of 42nd Annual Allerton Conference on Communication, Control, and Computing, Monticello, VA, USA, 29 September–1 October 2004.
34. Fong S.-L.; Yeung, R.W. Variable-Rate Linear Network Coding. *IEEE Trans. Inf. Theory* **2010**, 56, 2618–2625. [CrossRef]
35. Yang, S.; Yeung, R.W.; Ngai C.K. Refined Coding Bounds and Code Constructions for Coherent Network Error Correction. *IEEE Trans. Inf. Theory* **2011**, 57, 1409–1424. [CrossRef]

entropy

MDPI

Article

Efficient Communications in V2V Networks with Two-Way Lanes Based on Random Linear Network Coding

Yiqian Zhang [1,2], Tiantian Zhu [1] and Congduan Li [1,2,*]

[1] School of Electronics and Communication Engineering, Sun Yat-sen University, Shenzhen 518107, China; zhangyq75@mail2.sysu.edu.cn (Y.Z.); 18705168317@163.com (T.Z.)
[2] Shenzhen Key Laboratory of Navigation and Communication Integration, Shenzhen 518107, China
* Correspondence: licongd@mail.sysu.edu.cn

Abstract: Vehicle-to-vehicle (V2V) communication has gained significant attention in the field of intelligent transportation systems. In this paper, we focus on communication scenarios involving vehicles moving in the same and opposite directions. Specifically, we model a V2V network as a dynamic multi-source single-sink network with two-way lanes. To address rapid changes in network topology, we employ random linear network coding (RLNC), which eliminates the need for knowledge of the network topology. We begin by deriving the lower bound for the generation probability. Through simulations, we analyzed the probability distribution and cumulative probability distribution of latency under varying packet loss rates and batch sizes. Our results demonstrated that our RLNC scheme significantly reduced the communication latency, even under challenging channel conditions, when compared to the non-coding case.

Keywords: random linear network coding; vehicle-to-vehicle communication; dynamic topology; latency reduction

Citation: Zhang, Y.; Zhu, T.; Li, C. Efficient Communications in V2V Networks with Two-Way Lanes Based on Random Linear Network Coding. *Entropy* **2023**, *25*, 1454. https://doi.org/10.3390/e25101454

Academic Editor: Syed A. Jafar

Received: 19 September 2023
Revised: 9 October 2023
Accepted: 13 October 2023
Published: 17 October 2023

1. Introduction

Vehicles serve as vital means of transportation in urban cities, necessitating increased intelligence, as the intelligence of a single car falls short of meeting the requirements for road safety, path planning, decision-making, and traffic efficiency. To address these challenges, the Internet of Vehicles (IoV) has been introduced, enabling communication and collaboration among vehicles, and playing a crucial role in slow-vehicle warnings, intersection collision warnings [1], as well as congestion alleviation, emission reduction, and time saving [2]. However, these networks face obstacles in terms of mobility and occlusion. Notably, when a vehicle is traveling at a speed of 120 km/h, a mere 1-second latency can result in a driving distance of 33 m, potentially leading to severe consequences. Initially, communication among vehicles relied on dedicated short-range communication (DSRC) technology [3], which facilitated short-range communication with low latency [4]. Nevertheless, in high-speed scenarios, vehicles often move out of the communication range, rendering DSRC inadequate. The advent of 5G technology has introduced the concept of ultra-reliable and low-latency communications (uRLLC) [5], enabling vehicles to maintain communication even during high-speed mobility scenarios. To minimize long-term content access costs in vehicle-to-vehicle (V2V) networks, ref. [6] considered a distributed multi-agent reinforcement learning (MARL)-based edge caching method and proposed a distributed MARL-based edge caching method (DMRE), where every agent adaptively learns optimal caching strategies in collaboration with others. Additionally, they integrated the advantages of deep Q-Networks into DMRE, resulting in a computationally efficient method named DeepDMRE, which utilizes neural networks to approximate Nash equilibria. Such deep Q-Networks were also considered in [7], to explore the integration of reconfigurable intelligent surfaces (RIS) with unmanned aerial vehicles (UAVs) in the

downlink of non-orthogonal multiple-access (NOMA) networks. They proposed a joint optimization scheme using deep Q-networks to maximize system capacity, while considering UAV energy constraints and demonstrating significant improvements in system capacity.

Network coding [8] offers a promising solution for enhancing the performance of communication systems in V2V networks. By employing network coding techniques, intermediate nodes in the network can encode the received messages before transmitting them to the next hop, and the sink node decodes the received messages to reconstruct the original information. Ref. [9] proposed the use of XOR network coding in fault-tolerant dynamic scheduling and routing algorithms for time-sensitive in-vehicle networks (IVNs), to increase throughput, reliability, and robustness. Experimental results demonstrated that the XOR network coding scheme outperformed the frame replication and elimination for reliability (FRER) mechanism in terms of schedulability, flow, and response time, because the FRER mechanism tends to over-utilize the available bandwidth, whereas XOR network coding provides a better performance without excessive bandwidth usage. Ref. [10] expanded upon the security and privacy considerations in V2V networks as the number of vehicles accessing the network increases and proposed a comprehensive scheme that combines network coding, relay collaboration, and homomorphic encryption. The scheme ensures that the original information remains inaccessible to relay nodes, except for the intended target vehicle node. It also protects against potential collusion attacks, preventing conspiratorial attackers or multiple relay nodes from recovering the original information. Theoretically, such schemes guarantee the confidentiality, privacy protection, and anti-collusion capabilities of V2V networks. In [11], F. Ye et al. adopted network coding in vehicular ad hoc networks (VANETs) by modeling platoon vehicles driving in the same direction on a highway as a 1-D lattice network, in which a single source node aims to disseminate messages to all other vehicles. They analyzed the theoretical upper bound of the benefits achieved through network coding and conducted simulations to demonstrate the performance superiority over random broadcasting using Rayleigh fading wireless channels. F. Liu et al. [12] extended the data dissemination in VANETs in [11] to a two-way lane scenario by modeling the network as two separate 1-D lattice networks, corresponding to the two directions of traffic flow. They divided the dissemination into the *encountering* phase and the *separated* phase, determined by whether the broadcasting coverage areas of the two disseminators overlapped, which means vehicles traveling in both direction can communicate with both disseminators simultaneously. They analyzed the impact of the opposite direction over the traditional one-way lane model and showed that two disseminators traveling in opposite directions can enhance the speed of data dissemination. Ref. [13] compared three methods in a highway data mulling scenario, with vehicles from the opposite direction as data mutes to transmit large multimedia files, modeled as a *coupon collector* problem, and among which the network-coding-based strategy outperformed erasure-coding and repetition-coding strategies. The literature [14–16] shows that network coding can improve reliability and throughput, but it fails to deal with dynamic situations where the vehicle volume increases rapidly and the network structure becomes complex. Therefore, random linear network coding (RLNC) [17] has garnered significant attention, particularly for its ability to operate without prior knowledge of the network topology. RLNC involves random coding coefficient selection from a finite field and performing linear operations on the packets. As the vehicular scale increases, the random selection of RLNC encoding coefficients within a finite field obviates the need to account for variations in node quantity and network topology within this method. By receiving a sufficient number of packets with independent coefficients at the sink, the original information can be decoded at source, which enables transmitting content over wireless vehicle communications with lossy links and that are highly dynamic. Ref. [18] proposed a RLNC scheme for data transmission in a one-way lane V2V network, modeled as a multi-source multi-relay single-sink broadcasting network, to reduce latency and enhance the network robustness. In this one-way lane V2V communication scenario, the leading vehicles relay the detected road conditions and critical safety alerts to those following behind, affording

them sufficient time for well-informed decision-making. This type of information, with its small data payload, facilitates swift transmission with no node departures in multi-round communication processes, as assumed in [18], and implies a static and unchanging network topology. This may not align with the evolving landscape of intelligent transportation. In particular, with the increasing demand for in-vehicle entertainment experiences, expediting the transmission of large-scale data from nearby vehicles has become essential. Given the significant data volume involved, this study explores the utilization of vehicles in the opposite lane to establish a framework for bidirectional V2V large-scale data transmission over an extended period. During prolonged communication sessions for large-scale data transmission, nodes at high speeds tend to exit the communicable range of receiving vehicles, leading to dynamic changes in the network topology over multiple rounds. In this extended two-way lane large-scale data transmission scenario, the network is modeled as a multi-source single-sink network with a dynamic topology, where cars may enter or leave the communication range, resulting in a varying number of sources each round. The destination car node receives information from cars traveling in both the same and opposite directions. By utilizing RLNC in this dynamic two-way lane model, the proposed scheme enhances throughput and robustness, without relying on a specific network topology.

The main contributions of this paper are as follows:

- it extends the one-way lane model proposed in [18] to incorporate two-way lanes, thereby creating a dynamic network model;
- the paper provides a lower bound on the generation probability, demonstrating the feasibility and effectiveness of the RLNC scheme;
- it evaluates the performance of the RLNC scheme under frequently changing network conditions and poor channel conditions. The results demonstrate that RLNC significantly reduces latency compared to non-coding schemes.

The rest of this paper is organized as follows: Section 2 provides a brief overview of RLNC and compares our work with the related literature. Section 3 presents a system model, detailing the two-way lane RLNC transmission scheme and conducting an analysis of the generation probability and time delay. In Section 4, we analyze the simulation performance for communication delays under different packet loss rates and batch sizes, and then compare the coding and non-coding schemes. Section 5 concludes the paper.

2. Related Work

In this section, we give details about RLNC and introduce the reasons why RLNC is used. Then, the recent related literature for RLNC in V2V networks is compared with our work.

2.1. Brief of RLNC

We first give a brief introduction to RLNC. Li et al. proposed linear network coding (LNC) in [19], where they allowed intermediate nodes in the network to perform operations on the incoming packets, combining them linearly before forwarding. At the receiving end, the nodes can then decode the received combinations, to retrieve the original information. Ho et al. then proposed LNC in a randomized setting [20], where the coding coefficients are randomly chosen in a fixed-size finite field. Figure 1 illustrates a straightforward application of RLNC in a butterfly network. Source node s is tasked with sending messages X_1 and X_2 to sinks t_1 and t_2. Each channel can transmit only one message during a given time slot. Node s sends the linearly encoded X_1 and X_2 with the randomly selected coefficients (ξ_1, ξ_2), resulting in $\xi_1 X_1 + \xi_2 X_2$, to node 1. This information is then forwarded to nodes 3 and t_1. Similar operations occur at node 2, with randomly chosen coefficients (ξ_3, ξ_4). Given that node 3 receives two messages but can only utilize one channel to communicate with node 4, it becomes imperative to perform linear network coding at node 3 using

randomly chosen coefficients (ξ_5, ξ_6). Subsequently, node 4 forwards the encoded message to both sinks. The messages received at t_1 are denoted as Y_{11} and Y_{12}, which is

$$\begin{bmatrix} Y_{11} \\ Y_{12} \end{bmatrix} = \begin{bmatrix} \xi_1 & \xi_2 \\ \xi_5\xi_1 + \xi_6\xi_3 & \xi_5\xi_2 + \xi_6\xi_4 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}, \tag{1}$$

and the messages received at t_2 are denoted as Y_{21} and Y_{22}, which is

$$\begin{bmatrix} Y_{21} \\ Y_{22} \end{bmatrix} = \begin{bmatrix} \xi_3 & \xi_4 \\ \xi_5\xi_1 + \xi_6\xi_3 & \xi_5\xi_2 + \xi_6\xi_4 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}, \tag{2}$$

With invertible coefficient matrices, the original X_1 and X_2 can be decoded.

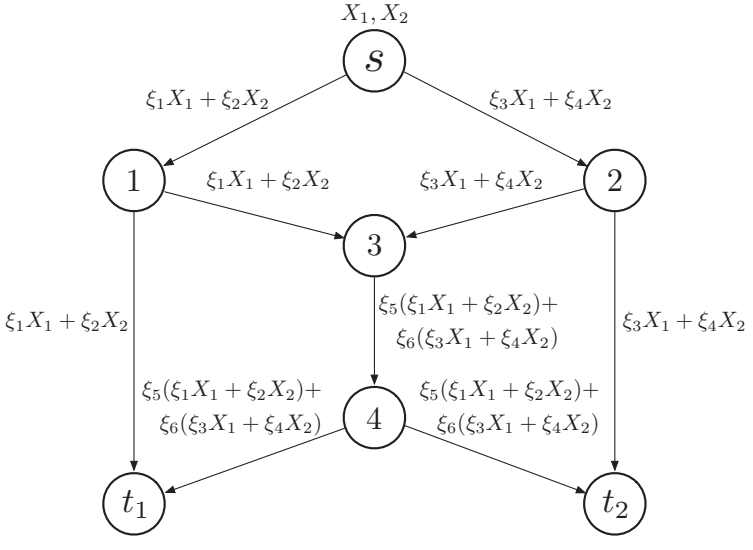


Figure 1. RLNC model in a butterfly network.

2.2. RLNC in V2V

Many works have introduced RLNC to V2V communication scenarios. Considering massive gigabit content transmission in millimeter-wave networks, ref. [21] applied symbol-level network coding (SLNC); that is, RLNC at the symbol scale, and utilized a cooperative concurrent distribution strategy in the scenario of highway network topology, where roadside units (RSU) encode the original packets and then forward them to vehicles. The proposed scheme enables collaborative V2V and vehicle-to-infrastructure (V2I) mmWave communications through a greedy network coding strategy based on a graph-theoretic approach. The scheme achieves a low latency, high efficiency, error resilience, and reliability. In ref. [22], E. Tasdemir et al. implemented a dynamic systematic sliding window RLNC scheme for end-to-end communication in vehicle platooning scenarios, where the platooning leader generates packets that are transmitted hop-by-hop to the platooning members. The coding process only involves packets within the dynamically sliding window, which moves forward to include new packets and is closed through feedback, and these packets are combined linearly to generate coded packets using RLNC techniques. This coding scheme was shown through simulation to provide resilience and low latency. To address challenges like transmission collisions and channel fading, ref. [23] proposed a hybrid medium access control (MAC) protocol for basic safety messages (BSMs) dissemination within the DSRC framework. Additionally, this protocol, with three sessions, a MAC setup session, CSMA session, and PNC session integrating physical-layer network coding and RLNC, further enhances the reliability and efficiency of BSM dissemination.

Ref. [24] further analyzed packet delivery ratio performance theoretically and through a comprehensive simulation. Our proposed method is compared with the recent literature works in a comparative table, as Table 1.

Table 1. Comparative table of our proposed method and recent literature.

	Ours	**[21]**	**[22]**	**[23]**	**[24]**
With the help of RSU	✗	✓	✗	✓	✓
The level of RLNC	Packet level	Symbol level	Packet level	Packet level	Packet level
One/two-way lane	Two-way lane	One-way lane	One-way lane	Two-way lane	Two-way lane and intersection scenario
Data scale	Large scale	Large scale	Large scale	BSM	BSM

3. System Model and RLNC Algorithm

In this section, we give the system model and introduce the RLNC algorithm.

3.1. System Model

First, we build the two-way lane V2V model based on real vehicle road scenarios, as illustrated in Figure 2. In the model, the car receiving messages, denoted as R and travels at a speed of v_R. We have m cars, denoted as $A_1, A_2, \ldots, A_m$, traveling in the same direction as R at constant speeds of $v_{A_1}, v_{A_2}, \ldots, v_{A_m}$, respectively. Additionally, there are w cars, denoted as $B_1, B_2, \ldots, B_w$, traveling in the opposite direction at constant speeds of $v_{B_1}, v_{B_2}, \ldots, v_{B_w}$, respectively. For each $i \in 1, 2, \ldots, m$ and $j \in 1, 2, \ldots, w$, cars A_i and B_j store M identical raw packets to be transmitted. These M raw packets collectively form a *generation*. Once the sink node R receives (or decodes) all M raw packets, the raw packets are updated to transmit the next generation.

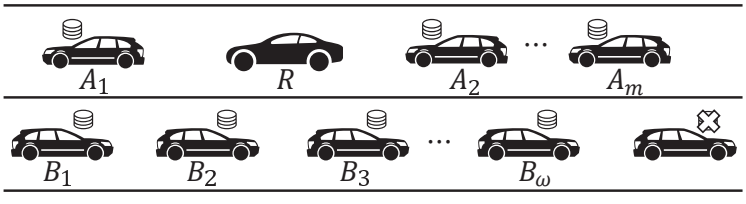


Figure 2. Vehicle road model.

R only communicates with cars within its communication range d. Specifically, R and A_i establish contact only when the distance between them, denoted as $d_{(A_i,R)}$, satisfies $d_{(A_i,R)} < d$. In the case where A_i is positioned ahead of R, the communication between R and A_i can be maintained for

$$t = \frac{d + d_{(A_i,R)}}{v_R - v_{A_i}}; \tag{3}$$

When $v_{A_i} > v_R$, the communication between R and A_i can be maintained for

$$t = \frac{d - d_{(A_i,R)}}{v_{A_i} - v_R}. \tag{4}$$

If $v_{A_i} = v_R$, they can always communicate with each other.

In addition, when A_i is positioned behind R, when $v_{A_i} < v_R$, the communication between R and A_i can be maintained for

$$t = \frac{d - d_{(A_i,R)}}{v_R - v_{A_i}}; \tag{5}$$

When $v_{A_i} > v_R$, the communication between R and A_i can be maintained for

$$t = \frac{d + d_{(A_i,R)}}{v_{A_i} - v_R}. \tag{6}$$

If $v_{A_i} = v_R$, they can always communicate with each other.

Regarding the opposite lane, if car B_i is moving towards R, then

$$t = \frac{d - d_{(B_i,R)}}{v_R + v_{B_i}}; \tag{7}$$

If car B_i is traveling in the opposite direction and is positioned behind car R, R and B_i can keep in touch for

$$t = \frac{d + d_{(B_i,R)}}{v_R + v_{B_i}}. \tag{8}$$

We further extract the model as a multi-source single-sink network, as shown in Figure 3. In this model, the sink node is denoted R. The cars traveling in the same direction are A_{α_t}, where $\alpha_t = 0, 1, 2, \ldots, m$.Similarly, the cars traveling in the opposite direction are denoted as B_{β_t}, with $\beta_t = 0, 1, 2, \ldots, w$. Both the cars in the same direction and those in the opposite direction possess identical sets of M raw data packets, collectively referred to as a generation. These packets are organized into batches to be transmitted. It is important to note that the number of source nodes, denoted by α_t and β_t, may vary in each round.

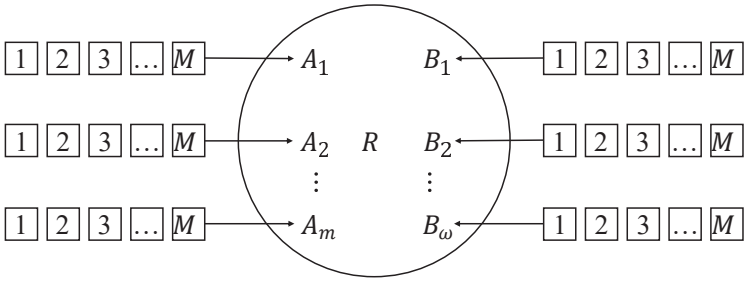


Figure 3. A sketch of a two-way lane model.

3.2. RLNC Algorithm

We now implement the RLNC algorithm, analyze the probability of generation, and present the corresponding algorithms. To implement RLNC, we select encoding coefficients from the finite field $GF(q)$, where the size of the finite field is denoted q. Consequently, we obtain the encoded data packet $\mathbf{\Gamma}_{A_i}$ transmitted by the source node A_i from the same direction as

$$\mathbf{\Gamma}_{A_i} = a_i^1 \boldsymbol{r}_1 + a_i^2 \boldsymbol{r}_2 + a_i^3 \boldsymbol{r}_3 + \cdots + a_i^M \boldsymbol{r}_M, \tag{9}$$

where $\boldsymbol{r}_k$ represents the kth data packet, and $a_i^k \in GF(q)$ denotes the encoding coefficient associated with $\boldsymbol{r}_k$ in the encoded data packet $\mathbf{\Gamma}_{A_i}$. Here, i ranges from 0 to α_t (the number of source nodes in the same direction), and k ranges from 1 to M (the total number of data packets in a generation). Similarly, the encoded data packet $\mathbf{\Gamma}_{B_j}$ from the opposite source B_j is

$$\mathbf{\Gamma}_{B_j} = b_j^1 \boldsymbol{r}_1 + b_j^2 \boldsymbol{r}_2 + b_j^3 \boldsymbol{r}_3 + \cdots + b_j^M \boldsymbol{r}_M, \tag{10}$$

where $b_j^k \in GF(q)$ is the encoding coefficient of $\boldsymbol{r}_k$ in $\mathbf{\Gamma}_{B_i}$, $j = 0, 1, 2, \ldots, \beta_t$, $k = 1, 2, \ldots, M$.

In each time slot, the source nodes collectively transmit the $\alpha_t + \beta_t$ packets that have been encoded using RLNC, which can be represented in matrix form as follows:

$$\begin{bmatrix} \boldsymbol{\Gamma}_{A_1} \\ \boldsymbol{\Gamma}_{A_2} \\ \cdots \\ \boldsymbol{\Gamma}_{A_{\alpha_t}} \\ \boldsymbol{\Gamma}_{B_1} \\ \boldsymbol{\Gamma}_{B_2} \\ \cdots \\ \boldsymbol{\Gamma}_{B_{\beta_t}} \end{bmatrix} = \begin{bmatrix} a_2^1 & a_2^2 & \cdots & a_2^M \\ a_1^1 & a_1^2 & \cdots & a_1^M \\ \cdots & & & \cdots \\ a_{\alpha_t}^1 & a_{\alpha_t}^3 & \cdots & a_{\alpha_t}^M \\ b_1^1 & b_1^2 & \cdots & b_1^M \\ b_2^1 & b_2^2 & \cdots & b_2^M \\ \cdots & & & \cdots \\ b_{\beta_t}^1 & b_{\beta_t}^2 & \cdots & b_{\beta_t}^M \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \boldsymbol{r}_3 \\ \cdots \\ \boldsymbol{r}_{M-1} \\ \boldsymbol{r}_M \end{bmatrix} \tag{11}$$

$$= \boldsymbol{C}_{(\alpha_t+\beta_t)\times M} \boldsymbol{R}_{M\times 1}, \tag{12}$$

where $\boldsymbol{C}_{(\alpha_t+\beta_t)\times M}$ is the coefficient matrix and $\boldsymbol{R}_{M\times 1}$ is the raw packet matrix.

3.2.1. Generation Probability

In order to decode M raw packets, the sink node R needs to receive M linearly independent encoded packets. If the encoding coefficient vector of a packet is linearly independent of the encoding data packets previously received, then this packet contributes to the decoding process. We define the number of linearly independent packets received by the sink node as the "sink's state", denoted as S_R. The generation probability, which represents the probability that the encoding packets are linearly independent, takes different forms based on the sink's state. Specifically, it depends on whether S_R is greater than or less than $M - \alpha_t - \beta_t$. To address these scenarios, we give Lemmas 1 and 2.

Lemma 1. *When the sink's state $S_R \leq M - \alpha_t - \beta_t$, generation probability is of the form*

$$P_{ge=\alpha_t+\beta_t} = \prod_{l=i}^{i+\alpha_t+\beta_t-1} (1 - \frac{1}{q^{M-l}}), \tag{13}$$

where q is the Galois field size, M is the number of raw packets in each batch, and i is the current state of the sink node, $\alpha_t = 0, 1, 2, \cdots, m$, $\beta_t = 0, 1, 2, \cdots, w$, $i = 0, 1, 2, \cdots, M - \alpha_t - \beta_t$.

Proof. Similarly to the proof of Theorem 1 in [18] but with $n = \alpha_t + \beta_t$, source nodes send $\alpha_t + \beta_t$ data packets in a time slot and the n packets are regarded as a group. n sources send n data packets $\boldsymbol{\Gamma}_{Y_\eta}$ in one time slot. There are $q_M - 1$ choices, resulting in

$$C_{(q^M-1)+\alpha_t+\beta_t-1}^{\alpha_t+\beta_t} = C_{q^M+\alpha_t+\beta_t-2}^{\alpha_t+\beta_t} \tag{14}$$

kinds of combination.

If all the n packets are linearly independent, then there are

$$\frac{\prod_{j=0}^{\alpha_t+\beta_t-1} C_{q^M-q^{i+j}}^1}{A_{\alpha_t+\beta_t}^{\alpha_t+\beta_t}} \tag{15}$$

kinds of combination.

Therefore, the generation probability is

$$P_{ge=\alpha_t+\beta_t} = \frac{\frac{\prod_{j=0}^{\alpha_t+\beta_t-1} C_{q^M-q^{i+j}}^1}{A_{\alpha_t+\beta_t}^{\alpha_t+\beta_t}}}{C_{q^M+\alpha_t+\beta_t-2}^{\alpha_t+\beta_t}} = \frac{\prod_{j=0}^{\alpha_t+\beta_t-1} C_{q^M-q^{i+j}}^1}{A_{q^M+\alpha_t+\beta_t-2}^{\alpha_t+\beta_t}}. \tag{16}$$

After simplifying, we can prove this. □

Lemma 2. *When the sink's state is $M > S_R > M - \alpha_t - \beta_t$, that is, $S_R = M - \alpha_t - \beta_t + 1, M - \alpha_t - \beta_t + 2, \ldots, M - 1$, the generation probability is of the form:*

$$P_{ge=M-i} = \prod_{l=i}^{M-1} (1 - \frac{1}{q^{M-l}}), \tag{17}$$

where q is the Galois field size, M is the number of raw packets in each batch, and i is the current sink's state, $i = M - \alpha_t - \beta_t + 1, M - \alpha_t - \beta_t + 2, \ldots, M - 1$.

Proof. Similarly to the proof of Theorem 2 in [18] but with $n = \alpha_t + \beta_t$, the generation probability is

$$P_{ge=M-i} = \frac{\prod_{j=0}^{M-i-1} C^1_{q^{M}-q^{i+j}}}{A^{M-i}_{M-i}} = \frac{\prod_{j=0}^{M-i-1} C^1_{q^{M}-q^{i+j}}}{A^{M-i}_{q^M+M-i-2}}. \tag{18}$$

After simplifying, we can prove this. □

The generation probability in this context exhibits similarities to the generation probability discussed in [18]. However, in the context of dynamic topology, the generation probability in each round is influenced, not only by the sink's state, but also by the number of currently communicable source nodes. Specifically, the sink's state in time slot t can be expressed as $S_R > M - \alpha_t - \beta_t$, indicating that the generation probability aligns with the conditions stated in Lemma 2. Conversely, prior to the initiation of the $(t+1)$th round of communication, due to multiple vehicles departing the communicable range, in time slot $t+1$, we have $S_R \leq M - \alpha_{t+1} - \beta_{t+1}$. In such cases, the generation probability adheres to the conditions specified in Lemma 1. This distinction arises from the changes in the sink's state and the varying number of communicable source nodes as a consequence of the dynamic topology in the network.

We focus on determining the lower bound of the generation probability. The lower bound is acquired when the sink's state is $M - m - w$ and sources send $m + w$ data packets. According to Appendix B in [18], let $n = m + w$, and we establish the lower bound of generation probability

$$\min P_{ge} = \prod_{l=M-m-w}^{M-1} (1 - \frac{1}{q^{M-l}}), \tag{19}$$

which is equivalent to

$$\min P_{ge} = \prod_{\mu=1}^{m+w} (1 - \frac{1}{q^{\mu}}), \tag{20}$$

Thus, the lower bound of the generation probability depends on the total number of sources at the beginning of communication $m + w$ and the Galois field size q.

Figure 4 illustrates the lower bound of P_{ge}, as given by Equation (20), where n represents the total number of sources ($n = m + w$). As the finite field size increases, the lower bound of generation probability also increases. For example, when considering a finite field size of $GF(256)$, the minimum generation probability exceeds 0.996. Consequently, with a sufficiently large finite field size, it is reasonable to assume that every packet transmitted to the sink is valid, and the generation probability approaches 1. Assuming that, after ζ rounds of communication, the sink node has received M data packets, the decoding probability can be expressed as

$$P_d > \prod_{\mu=1}^{m+w} (1 - \frac{1}{q^{\mu}})^{\zeta}. \tag{21}$$

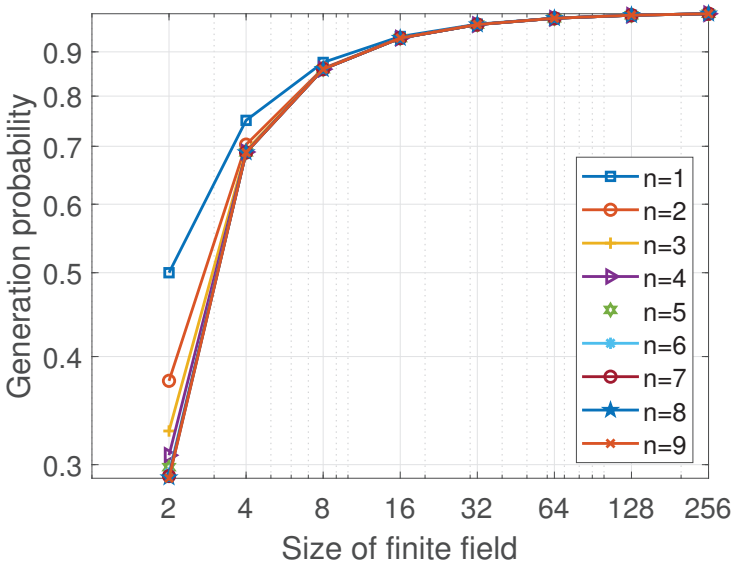


Figure 4. Lower bound of the generation probability with different sources.

3.2.2. Time Delay Analysis

In our analysis of the time delay, we consider the dynamic nature of the participating sources in each round, which differs from the one-way lane scenario described in [18]. To address a two-way lane scenario, we first determine the number of source nodes within the communication range of the sink during each round. This is performed based on the position, speed, and initial distance to the sink, as outlined in Algorithm 1. In Algorithm 1, we utilize an indicator variable **f**. When $\mathbf{f} = 1$, this indicates that the source node is initially positioned ahead of the sink node. Conversely, when $\mathbf{f} = 2$, this indicates that the source node is initially located behind the sink node. The algorithm utilizes this indicator to determine the number of source nodes present in each round, considering their relative positions with respect to the sink node. We tally the number of same direction sources engaged in each communication round. This is contingent upon whether the source is positioned ahead or behind the destination, as well as the relative speeds of the source and destination vehicles, and the relative distance between them. As for the count of counter-directional sources, this hinges on whether the source is located ahead or behind the destination, along with the relative distance between them.

Based on the number of sources participating in each round, $\alpha_t + \beta_t$, we obtain the binomial distribution for the state transition of the sink node. When the sink's state is $S_R = 0, 1, 2, \ldots, M - \alpha_t - \beta_t$, $\alpha_t + \beta_t$ source nodes collectively send $\alpha_t + \beta_t$ valid data packets. The probability of the sink node receiving k valid data packets in this round, which corresponds to a transition to k states in time slot t, can be calculated as

$$P_{mov}(t,k) = C^{k}_{\alpha_t+\beta_t}(1-p_e)^k p_e^{\alpha_t+\beta_t-k}, \tag{22}$$

where p_e denotes the packet loss rate.

Algorithm 1 # of source nodes within communication range in time slot t

Input: # of source nodes from the same direction m
# of source nodes from the reverse direction w
communication range d_b
speed of sink v_b
position of source nodes in the same direction **f**, speed **v**, distance **d**
position of source nodes in the reverse direction **fo**, speed **vo**, distance **do**
rounds of communication N

Initialize # of same direction nodes $\mathbf{Q} = 0$
$\mathbf{Q}(0) = m$
for $i = 1 \to N$ **do**
  count=0
  for $j = 0 \to m - 1$ **do**
    if $\mathbf{f}(j) = 1 \&\& \mathbf{v}(j) < v_b \&\& j(v_b - \mathbf{v}(j)) - (d_b + \mathbf{d}(j)) > 0$ **then**
      count=count-1
    end if
    if $\mathbf{f}(j) = 1 \&\& \mathbf{v}(j) > v_b \&\& j(\mathbf{v}(j) - v_b) - (d_b - \mathbf{d}(j)) > 0$ **then**
      count=count-1
    end if
    if $\mathbf{f}(j) = 2 \&\& \mathbf{v}(j) < v_b \&\& j(v_b - \mathbf{v}(j)) - (d_b - \mathbf{d}(j)) > 0$ **then**
      count=count-1
    end if
    if $\mathbf{f}(j) = 2 \&\& \mathbf{v}(j) > v_b \&\& j(\mathbf{v}(j) - v_b) - (d_b + \mathbf{d}(j)) > 0$ **then**
      count=count-1
    end if
  end for
  $\mathbf{Q}(i) = m$+count
end for
Initialize # of reverse direction nodes $\mathbf{Qo} = \mathbf{0}$
$\mathbf{Qo}(0) = w$
for $i = 1 \to N$ **do**
  count=0
  for $j = 0 \to w - 1$ **do**
    if $\mathbf{fo}(j) = 1 \&\& j(v_b + \mathbf{v}(j)) - (d_b + \mathbf{d}(j)) > 0$ **then**
      count=count-1
    end if
    if $\mathbf{f}(j) = 2 \&\& j(v_b + \mathbf{v}(j)) - (d_b - \mathbf{d}(j)) > 0$ **then**
      count=count-1
    end if
  end for
  $\mathbf{Qo}(i) = w$+count
end for
total number of the source nodes $\mathbb{Q} = \mathbf{Q} + \mathbf{Qo}$

Output: # of source nodes within communication range

When the state of the sink is $S_R = M - \alpha_t - \beta_t + 1, M - \alpha_t - \beta_t + 2, \cdots, M - 1$, the $M - S_R$ source nodes will send $M - S_R$ valid data packets. The probability of sink node transits k states in time slot t is

$$P_{mov}(t,k) = C_{M-S_R}^{k}(1-p_e)^k p_e^{M-S_R-k}. \tag{23}$$

The state matrix of the sink in the first time slot is

$$\mathbb{S}_1 = [B_{n(1)}(0)\ B_{n(1)}(1) \cdots\ B_{n(1)}(n(1))] \tag{24}$$

$$= [P_{0,0}^1\ P_{0,1}^1 \cdots\ P_{0,n(1)}^1], \tag{25}$$

where the binomial distribution $B_{n(t)}(\alpha)$ represents the probability of receiving α valid data packets out of the data packets sent by $n(t)$ source nodes in time slot t. Here, $n(t)$ represents the total number of source nodes in time slot t, which is initially $m+w$. $P_{i,j}^{k}$ denotes the probability of the sink state transitioning from i to j in the kth time slot. To determine the state matrix of the sink node after time slot t, denoted as $\mathbb{S}t$, we need to first solve for the state matrix after time slot $t-1$. The state matrix after time slot t is denoted $\mathbb{S}t$ and is represented by Equation (26).

The probability $P_t(M)$, which represents the sink node receiving M valid data packets after t time slots, can be calculated by summing the probabilities $P_{i,M}^{t}$ over all possible states i in the state matrix $\mathbb{S}t$. Mathematically, this can be expressed as $P_t(M) = \sum P_{i,M}^{t}$. The solution for the state matrix $\mathbb{S}_t$ depends on the state matrix $\mathbb{S}_{t-1}$ from the previous time slot and the number of source nodes $\mathbb{Q}(t)$ derived from Algorithm 1. Algorithm 2 provides a solution for calculating the probability P_t, which represents the sink node being in different states after time slot t. This probability is dependent on the values of P_{t-1} and $\mathbb{Q}(t-1)$. Specifically, $P_t(M)$ represents the completion probability of time slot t, which is the probability that sink node receives M data packets after t time slots. The recursive relationship between P_t and P_{t-1} is expressed as $P_t = \Phi(P_{t-1}, B_{\mathbb{Q}(t-1)})$, where $B_{\mathbb{Q}(t-1)}$ represents the binomial distribution of the number of received packets in time slot t when $\mathbb{Q}(t-1)$ data packets are sent. Such a solution for the recursive relation is provided in Algorithm 3. In Algorithm 3, the first step is to determine the number of source nodes participating in each round of communication. If this is larger than the needed number of packets M, then only M nodes will participate in the communication. Otherwise, all the nodes will participate. For each sink state i, the probability distribution at time slot t is computed by calculating the probability of receiving $k = i - j$ messages correctly after having received j messages (see line 19 in Algorithm 3). By utilizing Algorithm 2 and Algorithm 3, we can calculate the completion probability $P_t(M)$ of the sink node after t time slots. Subsequently, we will conduct an analysis of the delay probability distribution, taking into consideration varying packet loss rates p_e and packet batch sizes M.

$$
\begin{aligned}
\mathbb{S}_t &= \begin{bmatrix}
B_{n(t)}(0)\sum P_{\eta,0}^{t-1} & B_{n(t)}(1)\sum P_{\eta,0}^{t-1} & B_{n(t)}(2)\sum P_{\eta,0}^{t-1} & \cdots & B_{n(t)}(n(t))\sum P_{\eta,0}^{t-1} \\
B_{n(t)}(0)\sum P_{\eta,1}^{t-1} & B_{n(t)}(1)\sum P_{\eta,1}^{t-1} & B_{n(t)}(2)\sum P_{\eta,1}^{t-1} & \cdots & B_{n(t)}(n(t))\sum P_{\eta,1}^{t-1} \\
\cdots & & & & \\
B_{n(t)}(0)\sum P_{\eta,M-2}^{t-1} & B_{n(t)}(1)\sum P_{\eta,M-2}^{t-1} & B_{n(t)}(2)\sum P_{\eta,M-2}^{t-1} & \cdots & 0 \\
B_{n(t)}(0)\sum P_{\eta,M-1}^{t-1} & B_{n(t)}(1)\sum P_{\eta,M-1}^{t-1} & 0 & \cdots & 0
\end{bmatrix} \\
&= \begin{bmatrix}
P_{0,0}^{t} & P_{0,1}^{t} & P_{0,2}^{t} & \cdots & P_{0,n}^{t} \\
P_{1,1}^{t} & P_{1,2}^{t} & P_{1,3}^{t} & \cdots & P_{1,1+n}^{t} \\
\cdots & & & & \\
P_{M-2,M-2}^{t} & P_{M-2,M-1}^{t} & P_{M-2,M}^{t} & \cdots & 0 \\
P_{M-1,M-1}^{t} & P_{M-1,M}^{t} & 0 & \cdots & 0
\end{bmatrix}.
\end{aligned} \tag{26}
$$

Algorithm 2 Completion probability of sink at time slot t

Input: # of raw data packets M, packet loss rate p_e, communication times N, # of source nodes $\mathbb{Q}$

1: $B_n(i) = C_n^i(1-p_e)^i p_e^{n-i}$
2: **for** $i = 0 \rightarrow \mathbb{Q}(0)$ **do**
3: $\quad P_1(i) = B_{\mathbb{Q}(0)}(i)$
4: **end for**
5: **for** $t = 2 \rightarrow N$ **do**
6: $\quad P_t = \Phi(P_{t-1}, B_{\mathbb{Q}_{(t-1)}})$
7: **end for**

Output: Completion probability of sink at time slot t: $P_t(M)$

Algorithm 3 The state distribution probability of sink after time slot t:$P_t = \Phi(P_{t-1}, B_{\mathbb{Q}(t)})$

Input: The distribution probability of sink state last time slot P_{t-1},# of current source nodes $\mathbb{Q}(t-1)$

sumQ(t)=0
for $i = 0 \rightarrow t-1$ **do**
  sumQ(t)=sumQ(t)+$\mathbb{Q}(i)$
end for
if sumQ(t) $> M$ **then**
  maxNum = M
else
  maxNum=sumQ(t)
end if
sumQ(t-1)=0
for $i = 0 \rightarrow t-2$ **do**
  sumQ(t-1)=sumQ(t-1)+$\mathbb{Q}(i)$
end for
for $i = 0 \rightarrow$ maxNum **do**
  $P_t(i) = 0$
  for $j = 0 \rightarrow$ sumQ(t-1) **do**
    for $k = 0 \rightarrow \mathbb{Q}(t-1)$ **do**
      if $j + k = i$&&$j! = M$ **then**
        $P_t(i) = P_t(i) + P_t(j) \times B_{\mathbb{Q}(t-1)}(k)$
      end if
    end for
  end for
end for

Output: The state probability distribution P_t of sink after time slot t

4. Simulation Performance

In this section, we present a comprehensive analysis of the performance of the proposed scheme through Matlab simulations. Our main focus was on minimizing the latency, which was quantified by the number of time slots required to complete the transmission. We paid particular attention to two key factors that impact the latency: the packet loss rate, and the batch size. In addition, we conducted an analysis on the impact of varying vehicle communication ranges and different arrival rates following a stochastic arrival process, the Poisson process. Furthermore, we compared the coding and non-coding schemes. By comparing their performance, we could evaluate the effectiveness of the coding scheme in reducing the latency and improving the overall efficiency of the system.

4.1. Packet Loss Rate

To analyze the completion probability distribution under varying packet loss rates, we set up the following parameters:

- The initial distance of the source node to the sink node was randomly generated within the range of 0 to 150 m. This was because the current communication range of intelligent cars is 150–300 m. We stipulated that the communication range of the sink node was 150 m in front and behind; that is, the sink node could communicate with vehicles within a distance of 150 m:
- The speed of each node was randomly assigned within a range of 60 to 120 km/h considering the highway scenario;
- The initial position of each car was randomly generated, either in front of or behind the sink node;
- Each time slot was set to a duration of 100 ms, resulting in 10 rounds of communication per second.

With these parameters in place, we calculated the completion probability of the sink node at time slot t, which represents the likelihood of receiving M packets after t time slots.

Figure 5 shows the results of the simulation conducted with varying packet loss rates of $m = 2$, $w = 3$, and $M = 100$. In Figure 5a, we can observe that as the packet loss rate increased, the sink node required more time to receive M data packets, resulting in a more dispersed probability distribution of completion delay. This was because of the decrease in the number of data packets received by the sink node during each round, as a result of the high packet loss rate.Additionally, as the time slots progress, more sources may move out of the sink's communication range, resulting in fewer sources participating in the communication process and increasing the delay. Figure 5b demonstrates the correlation between the packet loss rate and the slope of the cumulative completion probability distribution curve. As the packet loss rate decreases, the curve becomes steeper, indicating a higher probability of timely completion. This implies that a lower packet loss rate leads to more efficient and reliable completion of the transmission process.

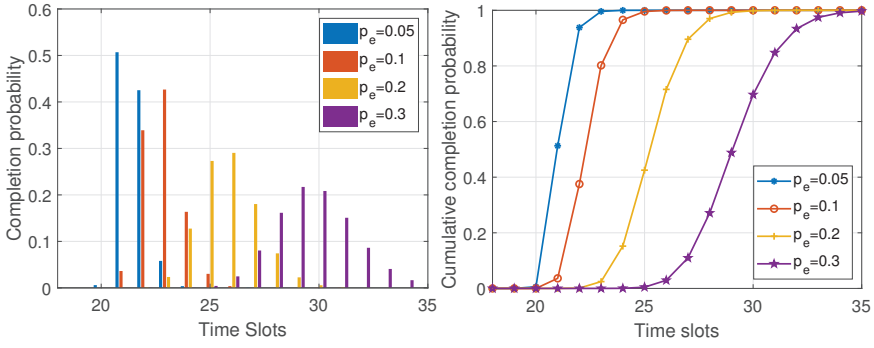


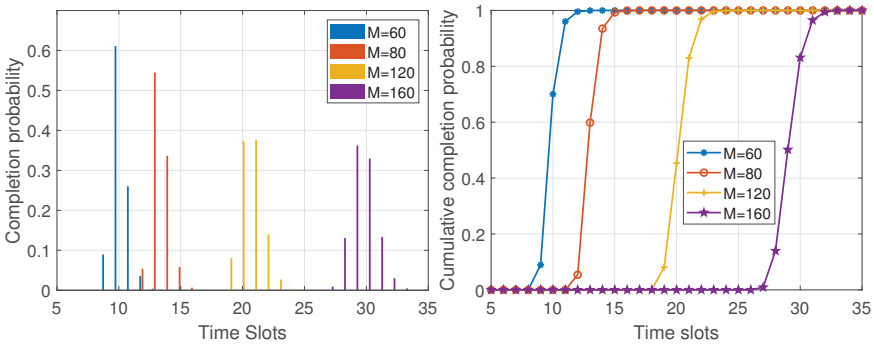
Figure 5. Completion probability distribution and cumulative completion probability distribution of the delay under different packet loss rates ($m = 2$, $w = 3$, $M = 100$). (**a**) Completion probability distribution; (**b**) cumulative completion probability distribution.

4.2. Batch Size

The delay also depends on the batch size. Figure 6 presents the simulation results under different batch sizes, with parameters set as $m = 3$, $w = 4$, $p_e = 0.1$. In Figure 6a, we can observe the completion probability distribution, and in Figure 6b, we can observe the cumulative distribution of the probability of delay. As the batch size increased, the number of time slots required to transmit a batch also increased. This resulted in a broader probability distribution of completion delay, indicating that larger batch sizes require more time to complete the transmission process. The increased delay was attributed to the larger number of data packets that had to be transmitted within each batch.

Table 2 presents the average delay and unit delay of the first batch of packets transmitted under varying batch sizes. It can be observed that as the batch size increased, the average delay consistently increased. However, contrary to the findings in the one-way lane scenario [18], the unit delay did not always decrease in the two-way lanes scenario. In the two-way lane scenario, after applying RLNC to M raw data packets and transmitting them to the sink node, the sink node needed to receive M valid data packets to perform decoding. Therefore, with larger batch sizes, more rounds of transmission were required to complete the transmission process, even with the same number of initial sources. Consequently, the duration of the communication process was prolonged, leading to a higher likelihood of source nodes moving out of the communication range of the sink node. Thus, the subsequent rounds witnessed a decrease in the number of sources and the number of packets received in each time slot. It is worth noting that for smaller network sizes,

the communication delay became larger, which was due to the limited number of sources available for transmission.



(**a**) (**b**)

Figure 6. Completion probability distribution and cumulative completion probability distribution of delay under different batch size ($m = 3$, $w = 4$, $p_e = 0.1$). (**a**) Completion probability distribution; (**b**) cumulative completion probability distribution.

The dynamic nature of the network topology needs to be considered when analyzing the impact of batch size. Merely focusing on the average and unit delay for the first batch is not sufficient. This is because, as subsequent batches are sent, the number of source nodes changes over time, influencing the overall communication delay. Therefore, we analyzed the total delay for sending a total of Q data packets, while varying the batch size M, under the condition that the total quantity of data packets Q remained constant. Table 3 presents the cumulative delay incurred during the transmission of all packets, under varying batch sizes. It can be observed that, as the batch size increased, the total delay exhibited a decreasing trend. This was because, with a smaller batch size, it took more rounds of communication to send the subsequent batches, and the source nodes may have left the communication range, resulting in more time slots. This implies that increasing the batch size can mitigate communication delays and improve the overall efficiency of the transmission process.To reduce the communication delay in two-way lane V2V communication using RLNC, increasing the batch size within the storage and computing capabilities of the source and sink nodes can effectively minimize the communication delay.

Table 2. Unit delay under different batch sizes (in time slots).

$M(m = 3, w = 4, p_e = 0.1)$	E_T	$\overline{T}$
60	10.2545	0.1709
80	13.4206	0.1678
120	20.6697	0.1722
160	29.5586	0.1847

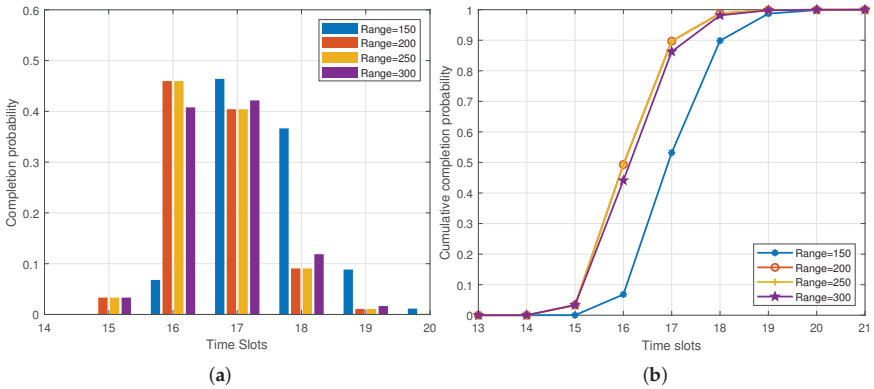
Table 3. Total delay under different batch sizes (in time slots).

$M(Q = 480, m = 3, w = 4, p_e = 0.1)$	E_T	$M(Q = 600, m = 4, w = 5, p_e = 0.1)$	E_T
60	258.0652	75	138.5758
80	255.6062	100	136.7833
120	252.2668	120	136.0340
160	250.0318	150	135.4871

4.3. Vehicle Range

Next, we examined the influence of varying the vehicle ranges on the time slots required for transmission, as illustrated in Figure 7. Figure 7a shows the completion proba-

bility distribution, and Figure 7b shows the cumulative completion probability distribution. It is evident that with a smaller communication range, more time slots were needed to complete the transmission. However, as the communication range increased, there was minimal impact on the transmission process. This was attributed to the fact that with a sufficiently large communication range, vehicles remained within the communication range until the transmission was complete.



(**a**) (**b**)

Figure 7. Completion probability distribution and cumulative completion probability distribution of delay under different range of vehicles ($m = 2$, $w = 3$, $p_e = 0.1$, $M = 100$). (**a**) Completion probability distribution; (**b**) cumulative completion probability distribution.

4.4. Poisson Arrival Process

The Poisson process approximates the stochastic arrival process of vehicles well. In the following analysis, we delved into the influence of varying arrival rates on the transmission, utilizing a Poisson process model to simulate vehicle arrivals. As depicted in the Figure 8, we initiated the network with three vehicles in the same direction and four in the opposite direction, assuming a packet loss rate of 0.1. λ represents the Poisson parameter, indicating the arrival rate of vehicles. For each task involving the transmission of 100 data packets, we determined the time slots required for 50 cases. At the beginning of the transmission, there was a period during which the model with $\lambda = 1$ (represented by blue triangles) required a significantly large number of time slots. This was because the vehicles transmitting initially gradually leave, but due to the relatively low arrival of vehicles, more time slots were needed to complete the transmission.

Notably, after multiple cases, we observed a stabilizing trend in the time required for each task. Once it reached a steady state, a higher vehicle arrival rate (signified by a larger λ) corresponded to a reduced number of time slots being required to complete a transmission. This was because, when there were more vehicles within communication range, more vehicles could participate in the transmission task, enabling faster reception of a sufficient number of decoded data packets.

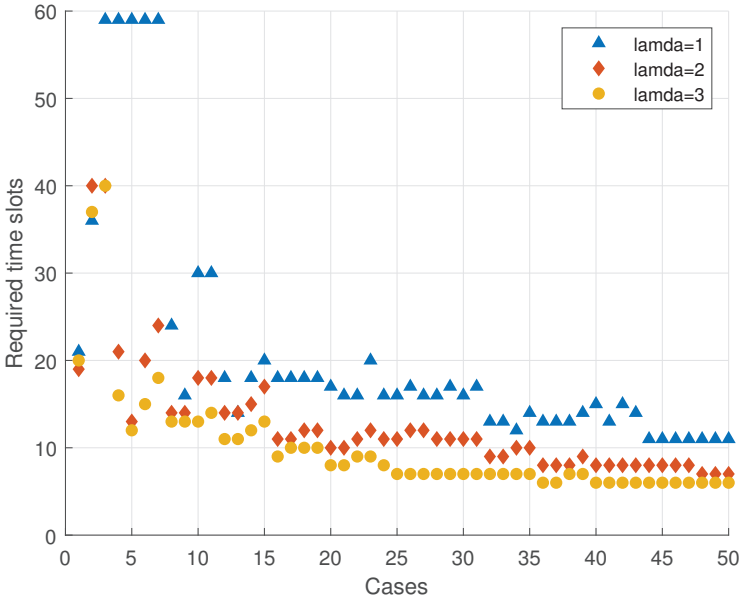


Figure 8. The time slots required with different λ in the Poisson process ($m = 2$, $w = 3$, $p_e = 0.1$, $M = 100$).

4.5. Coding vs. Non-Coding

By employing RLNC, M raw data packets were encoded in batches at the source nodes and transmitted to the sink node, which required the sink node to receive M valid data packets for decoding. Through multiple batches, the sink node could recover Q original data packets. Without coding, each source node randomly selected and sent one packet to the sink node in each round, until all packets had been received. However, this approach does not guarantee that the randomly selected data packets from different sources will be distinct, potentially leading to the sink node receiving duplicate data packets. Table 4 provides a comparison between the coding and non-coding schemes. It shows that as the packet loss rate increased, the communication latency also increased. However, it is worth noting that doubling the packet loss rate did not result in a significant increase in communication delay. The adoption of RLNC technology improved the communication robustness, enhancing the resistance against channel degradation and reducing communication delay.

In terms of RLNC coding overheads, this hinged on both the finite field size q and the number of original packets N involved in the coding process. Storing the coefficients in a single coded packet necessitated $(N-1) \cdot \log_2 q$ bits. It is evident that, with a small packet size, the performance was significantly hindered due to this substantial overhead. An approach involving the attachment of the seed of the random coefficients generator to the coded packets was employed in [14,15] for network coding, effectively reducing the overheads to $\log_2 q$, regardless of the number of combined packets. This idea, initially proposed in [25], is worth considering for adoption to reduce overheads in future research.

Table 4. Comparison of coding and non-coding schemes (in time slots).

Case	E_T ($Q = 400$, $M = 50$, $m = 2$, $w = 3$)	E_T ($Q = 600$, $M = 60$, $m = 4$, $w = 5$)
Coding ($p_e = 0.1$)	270.9284	180.3904
Coding ($p_e = 0.2$)	327.5296	225.0866
Coding ($p_e = 0.3$)	399.8219	282.0798
Non-coding ($p_e = 0$)	514.86	474.73

5. Conclusions

In this paper, we proposed an RLNC scheme for efficient and low-latency transmission of large-scale data in V2V communication with two-way lanes. We introduced a dynamic multi-source single-sink model specifically designed for the two-way lane V2V communication scenario and derived the lower bound of the generation probability for the RLNC scheme. Our analysis revealed that reducing the packet loss rate and increasing the sending batch size properly can effectively decrease the communication delay. By conducting a comparative analysis with a non-RLNC scheme, we demonstrated the superior performance of the RLNC scheme in reducing the communication delay and enhancing network robustness in V2V networks with a dynamic topology with two-way lanes.

Author Contributions: Conceptualization, T.Z. and Y.Z.; methodology, T.Z.; software, T.Z. and Y.Z.; validation, Y.Z., T.Z. and C.L.; formal analysis, Y.Z.; investigation, C.L.; writing—original draft preparation, Y.Z.; writing—review and editing, Y.Z. and C.L.; visualization, Y.Z.; supervision, C.L.; project administration, C.L.; funding acquisition, C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Science Foundation of China (NSFC) with grant Nos. 62271514 and the Science, Technology and Innovation Commission of Shenzhen Municipality with grant Nos. JCYJ20210324120002007, and ZDSYS20210623091807023.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Papadimitratos, P.; Fortelle, A.L.; Evenssen, K.; Brignolo, R.; Cosenza, S. Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation. *IEEE Commun. Mag.* **2009**, *47*, 84–95. [CrossRef]
2. Rios-Torres, J.; Malikopoulos, A.A. A Survey on the Coordination of Connected and Automated Vehicles at Intersections and Merging at Highway On-Ramps. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 1066–1077. [CrossRef]
3. Abboud, K.; Omar, H.A.; Zhuang, W. Interworking of DSRC and Cellular Network Technologies for V2X Communications: A Survey. *IEEE Trans. Veh. Technol.* **2016**, *65*, 9457–9470. [CrossRef]
4. Kenney, J.B. Dedicated Short-Range Communications (DSRC) Standards in the United States. *Proc. IEEE* **2011**, *99*, 1162–1182. [CrossRef]
5. Popovski, P.; Nielsen, J.J.; Stefanovic, C.; de Carvalho, E.; Strom, E.; Trillingsgaard, K.F.; Bana, A.S.; Kim, D.M.; Kotaba, R.; Park, J.; et al. Wireless Access for Ultra-Reliable Low-Latency Communication: Principles and Building Blocks. *IEEE Netw.* **2018**, *32*, 16–23. [CrossRef]
6. Zhou, H.; Jiang, K.; He, S.; Min, G.; Wu, J. Distributed Deep Multi-Agent Reinforcement Learning for Cooperative Edge Caching in Internet-of-Vehicles. *IEEE Trans. Wirel. Commun.* **2023**, 1. [CrossRef]
7. Zhang, H.; Huang, M.; Zhou, H.; Wang, X.; Wang, N.; Long, K. Capacity Maximization in RIS-UAV Networks: A DDQN-Based Trajectory and Phase Shift Optimization Approach. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 2583–2591. [CrossRef]
8. Ahlswede, R.; Cai, N.; Li, S.Y.; Yeung, R. Network information flow. *IEEE Trans. Inf. Theory* **2000**, *46*, 1204–1216. [CrossRef]
9. Syed, A.A.; Ayaz, S.; Leinmüller, T.; Chandra, M. Network Coding Based Fault-Tolerant Dynamic Scheduling and Routing for In-Vehicle Networks. *J. Netw. Syst. Manag.* **2023**, *31*. [CrossRef]
10. Sun, Y.; Yin, L.; Ma, Y.; Wang, C. IoV-SDCM: An IoV Secure Data Communication Model Based on Network Encoding and Relay Collaboration. *Secur. Commun. Netw.* **2022**, *2022*, 6546004. [CrossRef]
11. Ye, F.; Roy, S.; Wang, H. Efficient Data Dissemination in Vehicular Ad Hoc Networks. *IEEE J. Sel. Areas Commun.* **2012**, *30*, 769–779. [CrossRef]
12. Liu, F.; Chen, Z.; Xia, B. Data Dissemination With Network Coding in Two-Way Vehicle-to-Vehicle Networks. *IEEE Trans. Veh. Technol.* **2016**, *65*, 2445–2456. [CrossRef]
13. Park, J.S.; Lee, U.; Oh, S.; Gerla, M.; Lun, D.; Ro, W.W.; Park, J. Delay Analysis of Car-to-Car Reliable Data Delivery Strategies Based on Data Mulling with Network Coding. *IEICE Trans. Inf. Syst.* **2008**, *E91-D*, 2524–2527. [CrossRef]
14. Hassanabadi, B.; Valaee, S. Reliable Periodic Safety Message Broadcasting in VANETs Using Network Coding. *IEEE Trans. Wirel. Commun.* **2014**, *13*, 1284–1297. [CrossRef]
15. Gao, Y.; Ali, G.G.M.N.; Chong, P.H.J.; Guan, Y.L. Network Coding Based BSM Broadcasting at Road Intersection in V2V Communication. In Proceedings of the 2016 IEEE 84th Vehicular Technology Conference (VTC-Fall), Montreal, QC, Canada, 18–21 September 2016; pp. 1–5. [CrossRef]

16. Gao, Y.; Chong, P.H.J.; Guan, Y.L. BSM dissemination with network coded relaying in VANETs at NLOS intersections. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6. [CrossRef]
17. Ho, T.; Medard, M.; Koetter, R.; Karger, D.; Effros, M.; Shi, J.; Leong, B. A Random Linear Network Coding Approach to Multicast. *IEEE Trans. Inf. Theory* **2006**, *52*, 4413–4430. [CrossRef]
18. Zhu, T.; Li, C.; Tang, Y.; Luo, Z. On latency reductions in vehicle-to-vehicle networks by random linear network coding. *China Commun.* **2021**, *18*, 24–38. [CrossRef]
19. Li, S.Y.; Yeung, R.; Cai, N. Linear network coding. *IEEE Trans. Inf. Theory* **2003**, *49*, 371–381. [CrossRef]
20. Ho, T.; Koetter, R.; Medard, M.; Karger, D.; Effros, M. The benefits of coding over routing in a randomized setting. In Proceedings of the IEEE International Symposium on Information Theory, 2003. Proceedings, Yokohama, Japan, 29 June 2003–4 July 2003. [CrossRef]
21. Pan, S.; Zhang, X.M. Cooperative Gigabit Content Distribution with Network Coding for mmWave Vehicular Networks. *IEEE Trans. Mob. Comput.* **2023**. [CrossRef]
22. Tasdemir, E.; Lehmann, C.; Nophut, D.; Gabriel, F.; Fitzek, F.H.P. Vehicle Platooning: Sliding Window RLNC for Low Latency and High Resilience. In Proceedings of the 2020 IEEE International Conference on Communications Workshops (ICC Workshops), Dublin, Ireland, 7–11 June 2020. [CrossRef]
23. Zhang, M.; Chong, P.H.J.; Seet, B.C.; Rehman, S.U.; Kumar, A. Integrating PNC and RLNC for BSM dissemination in VANETs. In Proceedings of the 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), Montreal, QC, Canada, 8–13 October 2017; pp. 1–5. [CrossRef]
24. Zhang, M.; Ali, G.G.M.N.; Chong, P.H.J.; Seet, B.C.; Kumar, A. A Novel Hybrid MAC Protocol for Basic Safety Message Broadcasting in Vehicular Networks. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 4269–4282. [CrossRef]
25. Liu, Z.; Wu, C.; Li, B.; Zhao, S. UUSee: Large-Scale Operational On-Demand Streaming with Random Network Coding. In Proceedings of the 2010 Proceedings IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010; pp. 1–9. [CrossRef]

entropy

MDPI

Article

Fundamental Limits of Coded Caching in Request-Robust D2D Communication Networks †

Wuqu Wang [1], Zhe Tao [2], Nan Liu [1,*] and Wei Kang [3]

[1] National Mobile Communications Research Laboratory, Southeast University, Nanjing 211189, China; wuquwang@seu.edu.cn
[2] Huawei Technologies, Nanjing 210012, China
[3] School of Information Science and Engineering, Southeast University, Nanjing 211189, China
* Correspondence: nanliu@seu.edu.cn
† This article was presented in part at the IEEE International Symposium on Networks, Computers and Communications (ISNCC 2023), Doha, Qatar, 23–26 October 2023.

Abstract: D2D coded caching, originally introduced by Ji, Caire, and Molisch, significantly improves communication efficiency by applying the multi-cast technology proposed by Maddah-Ali and Niesen to the D2D network. Most prior works on D2D coded caching are based on the assumption that all users will request content at the beginning of the delivery phase. However, in practice, this is often not the case. Motivated by this consideration, this paper formulates a new problem called *request-robust D2D coded caching*. The considered problem includes K users and a content server with access to N files. Only r users, known as requesters, request a file each at the beginning of the delivery phase. The objective is to minimize the average and worst-case delivery rate, i.e., the average and worst-case number of broadcast bits from all users among all possible demands. For this novel D2D coded caching problem, we propose a scheme based on uncoded cache placement and exploiting common demands and one-shot delivery. We also propose information-theoretic converse results under the assumption of uncoded cache placement. Furthermore, we adapt the scheme proposed by Yapar et al. for uncoded cache placement and one-shot delivery to the request-robust D2D coded caching problem and prove that the performance of the adapted scheme is order optimal within a factor of two under uncoded cache placement and within a factor of four in general. Finally, through numerical evaluations, we show that the proposed scheme outperforms known D2D coded caching schemes applied to the request-robust scenario for most cache size ranges.

Keywords: coded caching; device-to-device; request-robust; order-optimal scheme

Citation: Wang, W.; Tao, Z.; Liu, N.; Kang, W. Fundamental Limits of Coded Caching in Request-Robust D2D Communication Networks. *Entropy* **2024**, *26*, 250. https://doi.org/10.3390/e26030250

Academic Editors: Shenghao Yang and Kenneth Shum

Received: 4 February 2024
Revised: 1 March 2024
Accepted: 7 March 2024
Published: 12 March 2024

1. Introduction

In recent years, the demand for user throughput has greatly increased by applications based on fifth-generation (5G) mobile networks [1], such as short videos, self-driving vehicles, the Metaverse, etc. Fortunately, the data of such applications can be pre-stored in the user's storage during low-network consumption periods, preventing network congestion during peak hours. This approach is known as *caching* [2]. There are typically two phases in the caching process [3]. Before knowing any user requests, the server fills the users' caches in the *placement phase* during off-peak hours. The *delivery phase* follows in peak hours. The delivery signals will be designed and transmitted from terminals like the server or the users to satisfy all user demands when they are revealed. Technology for caching has advanced quickly in recent years, and it is currently regarded as one of the effective methods for relieving the congestion of wireless networks.

Traditional caching ignores the processing capability of the users, and therefore, the contents cached by the users and the signals transmitted by the server are both uncoded. In contrast to traditional caching, *coded caching* [3], proposed by Maddah-Ali and Niesen, uses a combination of coded multi-casting and device caching to simultaneously fulfill

multiple requests through coded transmissions. The coded caching strategy works for the prototypical network topology called the *single shared-link network*, e.g., vehicular networks [4]. Both the cache contents of the users and the delivery signal from the server are allowed to be coded in the coded caching problem. The aim is to design a caching and delivery scheme that minimizes the *average* and *worst-case* delivery rate, which is defined as the average and worst-case number of broadcast bits among all possible user requests. When the optimal caching and delivery scheme that achieves the lowest worst-case delivery rate can be found for any user cache size, the optimal tradeoff between rate and memory for the system is determined. If each user directly stores a subset of the files' bits in its cache without coding, it is referred to as an *uncoded* cache placement scheme; otherwise, it is referred to as a *coded* cache placement scheme. The original problem [3] studied in coded caching is *centralized*, assuming that all users present during the placement phase will each make a request for a file at the beginning of the delivery phase. *Decentralized coded caching* [5–7] also considers the possibility of users leaving or turning off during the delivery phase and explores less coordinated caching strategies.

Taking self-driving vehicles as an example: one promising approach to improve the communication efficiency is through the use of device-to-device (D2D) communication, which allows the users to directly exchange information with each other without the need for a server like a base station. This can be particularly useful in situations where the traditional infrastructure is limited or unavailable, such as in remote or rural areas. To solve the coded caching problem in these scenarios, a framework is proposed by Ji et al. in [8] for D2D coded caching. In the placement phase, similar to coded caching [3], the server fills the users' caches before the users make any requests. In the delivery phase, when the users reveal their demands, the server is disconnected from the users and it is up to the users to communicate with each other so that each user can decode the file it requested using the signals transmitted by the other users and the contents of its local cache. For the centralized D2D coded caching problem, the caching strategy of [3] (Algorithm 1), which is uncoded, is widely used in the placement phase, e.g., in [8,9] and so on. In [8], a novel delivery scheme was provided that is appropriate for the D2D scenario. Additionally, a well-known D2D coded caching converse was proposed in [8], and it has been demonstrated that, when the memory size is large, the proposed D2D caching and delivery scheme is order optimal within a constant factor. It is difficult to find the optimal caching and delivery scheme and the corresponding optimal rate–memory tradeoff for the centralized D2D coded caching problem. However, there are many researchers who try to find the fundamental limits of the centralized D2D coded caching problems under certain assumptions or additional constraints, e.g., [9,10].

With concern to the timeliness of the communication, *one-shot delivery*, which is defined to satisfy the condition that each user can decode any bit of its requested file from its own cache and the transmitted signal from at most one other user, is proposed in [9] for the centralized D2D coded caching problem. For example, one self-driving vehicle may quickly decode the requested map data after receiving the signals transmitted by another self-driving vehicle, without waiting for all the considered vehicles to complete the transmission of signals. The proposed caching and delivery scheme in [9] is optimal under the constraint of uncoded cache placement and one-shot delivery, and it is order optimal within a factor of two if the converse of the shared-link coded caching problem with uncoded cache placement [11] is used as the lower bound and order optimal within a factor of four compared to the general D2D coded caching converse results.

In addition to [9], many other researchers study variants of the D2D coded caching problems, such as allowing for coded placement with three users [10], private caching [12], private caching with a trusted server [13,14], distinct cache sizes [15], finite file packetizations [16], finite-length analysis [17], secure coded caching [18], secure delivery [19], wireless multi-hop D2D networks [20,21], partially cooperative D2D communication networks [22,23], constructions of placement delivery arrays (PDAs) [24], and so on. Among these papers, most of them assume that all users will request content at the begin-

ning of the delivery phase. However, in practice, this may not be true. For example, when assisted self-driving vehicles within a certain range carry out D2D communication, they may not request at the same time or some of them may be driven manually and do not need to access high-definition map data. In these situations, waiting for all users to request content will waste time, and setting the requests of the users who do not request some arbitrary file demand will waste communication resources. Note that in these scenarios, even though the users may not request data, they are still available to participate in the delivery phase by transmitting signals that are functions of their cached contents.

Hence, in this paper, we propose and study a new problem called *request-robust D2D coded caching*, where in the delivery phase, though all users in the placement phase are still present and may help with the transmission, some of them do not request any files. It is not known in the placement phase the number or identity of the users who do not request files. This problem is not the same as the decentralized D2D coded caching problem [8], where users who leave or turn off during the delivery phase do not make file requests, nor do they participate in the delivery. Note that this problem is similar to the user inactivity problem in the D2D caching setting [9,22], where each user may independently have a probability of being inactive, i.e., they do not make a file request at the beginning of the delivery phase. However, in the request-robust D2D coded caching problem, inactive users still help in the delivery phase by transmitting signals, whereas in the user inactivity problem, they do not.

1.1. Main Contributions

The main contributions of the paper can be summarized as follows:

(1) For the request-robust D2D coded caching problem, we adapt the scheme from [9] for uncoded cache placement and one-shot delivery and call the adapted scheme the *adapted Yapar–Wan–Schaefer–Caire (YWSC) scheme*.

(2) In order to find better performance, we present a new achievable scheme based on the uncoded cache placement and exploiting common demands [11] and one-shot delivery [9]. The caching strategy is the same as that proposed by Maddah-Ali and Niesen in [3] (Algorithm 1), while the delivery strategy divides the sub-files into three categories, and different delivery signals are designed for each category. We call the new scheme the *three-category-based scheme*. This scheme was presented in the conference version of this paper [25].

(3) We propose an information-theoretic lower bound under uncoded cache placement based on seeking the converse of a problem called coded caching with inactive users. The problem of coded caching with inactive users was proposed in [26], where users are inactive with a certain probability in the traditional coded caching problem of [3]. Hence, the converse for the problem of coded caching with inactive users can serve as a converse for the request-robust D2D coded caching problem. Note that [26] only considers the optimization of the cache replication parameter and does not provide a converse for the caching and delivery scheme.

(4) We prove that the performance of the adapted YWSC scheme is order optimal within a factor of two under the assumption of uncoded cache placement and within a factor of four in general.

(5) Through numerical evaluation, we show that the three-category-based scheme outperforms the adapted YWSC scheme, as well as other known D2D coded caching schemes [3] applied to the request-robust scenario.

1.2. Notations

Throughout this paper, $H(\cdot)$ represents the entropy of random variables, $|\cdot|$ represents the cardinality of a set, $\oplus$ denotes finite field addition, we let $\mathcal{X}\backslash\mathcal{Y} \triangleq \{x \in \mathcal{X} | x \notin \mathcal{Y}\}$, $[x:y:z] \triangleq \{x, x+y, x+2y, ..., z\}$, $[x:y] = [x:1:y]$ and $[n] = [1:n]$. For two integers x, and y, if $x < y$ or $x \leq 0$, we let $\binom{y}{x} = 0$.

2. System Model and Related Background

2.1. System Model

We study the *request-robust D2D coded caching* problem, which is defined in the following. We consider a D2D coded caching system (see Figure 1) where a server is connected to a fixed content file database of N files, $\mathcal{W} \triangleq (W_1, ..., W_N)$. Each file consists of F bits. There are K users in the system, each with a cache of size MF bits. We focus on the non-trivial scenario where $M \leq N$. Let $\mathcal{K}$ be the set of user indices, i.e., $\mathcal{K} \triangleq [K]$.

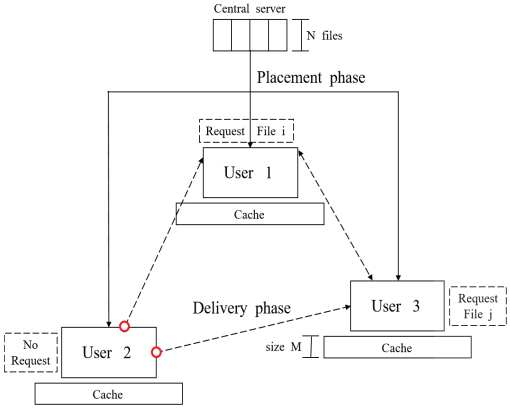


Figure 1. System model for request-robust D2D coded caching problem when there are 3 users. In this realization, User 2 does not request. Solid and dotted lines indicate placement and delivery phases, respectively.

The system operates in two phases. In the placement phase, each user's cache is filled by the central server, which does not know the number of users or the identities of the users requesting files in the other phase. Denote the content in the cache of User k as Z_k, $k \in \mathcal{K}$. In the delivery phase, some of the K users will make file requests while others will not. We denote the set of users making file requests as $\mathcal{R}$, $\mathcal{R} \subseteq \mathcal{K}$. Each user in $\mathcal{R}$ will request a single file. Let r denote the number of users requesting files, i.e., $r \triangleq |\mathcal{R}|$, and we assume that the file requests, i.e., which user requests which file and which users are not requesting any files, are known to all K users. Each of the K users will send a signal that will be received by the users in $\mathcal{R}$. It is required that each user in $\mathcal{R}$ can decode its requested file by using the signals received and its own cache content. Note that Figure 1 is different from [9] (Figure 1), i.e., there exists a user who does not request any file in Figure 1, while in [9] (Figure 1), all users request files.

More specifically, a caching and delivery scheme for this system consists of

1. K caching functions
$$\varphi_k : [2^F]^N \to [2^{MF}], \quad k \in \mathcal{K},$$
which map the N files into cache contents of the users, denoted by $Z_k = \varphi_k(W_1, ..., W_N)$, $k \in \mathcal{K}$. Thus, we have the following entropy constraint:
$$H(Z_k | W_1, W_2, ..., W_N) = 0, \quad k \in \mathcal{K}.$$

2. $K \sum_{r=1}^{K} \binom{K}{r} N^r$ encoding functions
$$\phi_k^{D_\mathcal{R}} : [2^{MF}] \to [2^{R_k F}], \quad k \in \mathcal{K}.$$
where $D_\mathcal{R}$ is the set of file requests made by the users in $\mathcal{R}$. For example, if there are $K = 4$ users, and Users 1 and 3 do not request files during the delivery phase,

the request vector is $D_{\{2,4\}} = (d_2, d_4)$. The encoding function $\phi_k^{D_\mathcal{R}}$ denotes the mapping of User k from its cached content to the signal it transmits, which is denoted as $X_k^{D_\mathcal{R}}$, i.e., $X_k^{D_\mathcal{R}} \triangleq \phi_k^{D_\mathcal{R}}(Z_k)$. Thus, we have

$$H(X_k^{D_\mathcal{R}}|Z_k) = 0, \quad k \in \mathcal{K}. \tag{1}$$

We assume the signal $X_k^{D_\mathcal{R}}$ consists of $R_k^{D_\mathcal{R}} F$ bits. The signals transmitted by all K users consist of $R^{D_\mathcal{R}} F$ bits, i.e., $R^{D_\mathcal{R}} = \sum_{k=1}^K R_k^{D_\mathcal{R}}$.

3. $\sum_{r=1}^K \binom{K}{r} r N^r$ decoding functions

$$\psi_k^{D_\mathcal{R}} : [2^{MF}] \times [2^{F\sum_{u\in\mathcal{K}\setminus\{k\}} R_u^{D_\mathcal{R}}}] \to [2^F], \quad k \in \mathcal{R},$$

which is the decoding function used by User k. For example, if there are $K = 4$ users, and Users 1 and 3 do not request any file during the delivery phase, the decoded files at Users 2 and 4 are $\hat{W}_{d_2} = \psi_2^{(d_2,d_4)}(Z_2, X_1^{(d_2,d_4)}, X_3^{(d_2,d_4)}, X_4^{(d_2,d_4)})$ and $\hat{W}_{d_4} = \psi_4^{(d_2,d_4)}(Z_4, X_1^{(d_2,d_4)}, X_2^{(d_2,d_4)}, X_3^{(d_2,d_4)})$, respectively.

Correct decoding by the users requesting files is given by $\hat{W}_{d_k} = W_{d_k}, k \in \mathcal{R}$, or in other words,

$$H(W_{d_k}|Z_k, X_{[K]\setminus\{k\}}^{D_\mathcal{R}}) = 0, \qquad k \in \mathcal{R}, \tag{2}$$

which is called the decodability constraint. We find that by combing (1) and (2), one can decode any file by knowing the cache of all users, i.e.,

$$H(W_{[N]}|Z_{[K]}) = 0. \tag{3}$$

which implies that we are interested in the case where $KM \geq N$.

For any caching and delivery scheme that satisfies the decodability constraint, for a fixed $\mathcal{R}$ with size r, we define $\mathcal{D}^\mathcal{R}$ as the set of all possible demands $\{1, \cdots, N\}^r$. We are interested in two performance metrics: one is with respect to the average performance, and the other is with respect to the worst performance. More specifically, the average performance is defined as follows: we assume that the request vector $D_\mathcal{R}$ is uniformly distributed on $\mathcal{D}^\mathcal{R}$. Then, the average delivery rate with respect to the uniform demand $R_{\text{ave,req-rob}}^\mathcal{R}$ is defined as

$$R_{\text{ave,req-rob}}^\mathcal{R} = \mathbb{E}_{D_\mathcal{R}}[R^{D_\mathcal{R}}].$$

For a given r, we define the maximum average delivery rate with respect to the uniform demand $R_{\text{ave,req-rob}}^r$, where the maximum is over all request sets $\mathcal{R}$ with size r, i.e.,

$$R_{\text{ave,req-rob}}^r = \max_{\mathcal{R}:|\mathcal{R}|=r} R_{\text{ave,req-rob}}^\mathcal{R}$$

The worst-case performance is defined as follows: first, the worst-case delivery rate $R_{\text{worst,req-rob}}^\mathcal{R}$ is defined as

$$R_{\text{worst,req-rob}}^\mathcal{R} = \max_{D_\mathcal{R}} R^{D_\mathcal{R}}.$$

For a given r, we define the maximum worst-case delivery rate $R_{\text{worst,req-rob}}^r$, where the maximum is over all request sets $\mathcal{R}$ with size r, i.e.,

$$R_{\text{worst,req-rob}}^r = \max_{\mathcal{R}:|\mathcal{R}|=r} R_{\text{worst,req-rob}}^\mathcal{R}.$$

We would like to design caching and delivery schemes such that $R^r_{\text{ave,req-rob}}$ and $R^r_{\text{worst,req-rob}}$ are both the smallest for every $r = 1, 2, \cdots, K$. As can be seen, this is a multi-objective optimization.

For easy presentability of the results, following the notation of [11], we denote $N_e(D_{\mathcal{R}})$ as the number of distinct files in a request vector $D_{\mathcal{R}}$. $D_{\mathcal{R}\setminus\{k\}}$ and $N_e(D_{\mathcal{R}\setminus\{k\}})$ are denoted as the request vector of users $\mathcal{R} \setminus \{k\}$ and the number of distinct files requested by all requesters but User k, respectively.

2.2. Preliminaries

In this subsection, we briefly summarize the related approaches, namely the uncoded symmetric placement scheme in [3] and the problem of coded caching with inactive users, which are critical for building our results for the request-robust D2D coded caching problem.

2.2.1. Uncoded Symmetric Placement Scheme

First, we introduce the uncoded symmetric placement scheme, which is useful for our scheme proposed in Section 4.

Definition 1. *(Maddah-Ali Niesen [MAN] Uncoded Symmetric Placement Scheme): Define t as $t = KM/N$. When t is an integer, we have the MAN uncoded symmetric placement scheme as follows: Each file W_n is divided into $\binom{K}{t}$ disjoint sub-files denoted by $W_{n,\mathcal{T}}$, where $n \in [N]$, $\mathcal{T} \subseteq K$, $|\mathcal{T}| = t$, and $H(W_{n,\mathcal{T}}) = F/\binom{K}{t}$. Each user k caches all the bits of the sub-files $W_{n,\mathcal{T}}$, $n \in [N]$, for all $\mathcal{T} \ni k$. Since each file includes $\binom{K-1}{t-1}$ sub-files with $\mathcal{T} \ni k$, each user k satisfies the memory constraint $H(Z_k) = NF\binom{K-1}{t-1}H(W_{n,\mathcal{T}}) = NFt/K = MF$.*

For the convenience of understanding and reference, we give the algorithm of this scheme in Algorithm 1.

Algorithm 1 MAN Uncoded Symmetric Placement Scheme $(N, K, M, W_{[N]})$

$t \leftarrow KM/N$
$\mathcal{T} \leftarrow \{\mathcal{T} \subseteq [K] : |\mathcal{T}| = t\}$
for $n \in N$ **do**
  Divide file W_n into disjoint sub-files $(W_{n,\mathcal{T}} : \mathcal{T} \in \mathcal{T})$ with equal size
end for
for $k \in [K]$ **do**
  $Z_k \leftarrow (W_{n,\mathcal{T}} : n \in [N], \mathcal{T} \in \mathcal{T}, k \in \mathcal{T})$
end for

The uncoded symmetric placement scheme is the optimal achievable placement scheme both for the shared-link model with uncoded cache placement [3] and D2D work with uncoded cache placement and one-shot delivery [9], which reveals that regardless of the number of users, using the uncoded symmetric placement scheme can satisfy the optimal rate of these models in all cases. Due to the superiority of the uncoded symmetric placement scheme, we use the scheme as the placement scheme in our scheme proposed in Section 4.

2.2.2. Problem of Coded Caching with Inactive Users

We denote our original D2D model with r users of K users requesting files independently in $\mathcal{W}$ as System 1. In order to derive the converse of System 1, we consider another system model named *coded caching with inactive users*, denoted as System 2. This is the model where a central server responds to the users' requests, and some of the users do not request any files in the delivery phase. The central server connects to the whole file database.

The placement phase of System 2 is exactly the same as that of System 1. Thus, in System 2, Equation (3) is still satisfied. The delivery phase of System 2 is different from

that of System 1. Specifically, in System 1, the codewords are transmitted by the users, while in System 2, the codewords are transmitted by the central server. Since the central server has the whole database and connects to all K users, while in System 1, each user only caches a subset of the whole database and only connects to other $K-1$ users, the optimal maximum average and worst-case delivery rate in System 2, denoted as $R^{r*}_{\text{ave,inactive}}$ and $R^{r*}_{\text{worst,inactive}}$, can not be larger than the delivery rate in System 1, respectively. In other words, we have the following inequality:

$$R^{r*}_{\text{ave,req-rob}} \geq R^{r*}_{\text{ave,inactive}}, \tag{4}$$

$$R^{r*}_{\text{worst,req-rob}} \geq R^{r*}_{\text{worst,inactive}}. \tag{5}$$

3. Main Result

In this section, we present the main results of this work. We propose two achievable schemes for the request-robust D2D coded caching problem in Theorems 1 and 2. We further propose a converse for the problem of coded caching with inactive users in Theorem 3, which also serves as a converse result to the request-robust D2D coded caching problem. Theorem 4 compares the performance gap between the achievability result in Theorem 1 and the converse result in Theorem 3 and shows that they are within a multiplicative gap.

The first achievable scheme is obtained by adapting the achievable scheme in [9] with uncoded cache placement and one-shot delivery to the request-robust D2D coded caching problem. More specifically, the adaptation is performed by assigning the users, who do not request, a demand that is most requested by the requesters. We call the adapted scheme the *adapted Yapar–Wan–Schaefer–Caire (YWSC) scheme*. We denote the adapted request vector as $D'_{\mathcal{K}}$ and the adapted request vector of users $\mathcal{K}\setminus\{k\}$ as $D'_{\mathcal{K}\setminus\{k\}}$. Hence, we have $N_e(D'_{\mathcal{K}}) = N_e(D_{\mathcal{R}})$ and obtain Theorem 1 as follows:

Theorem 1. *For the request-robust D2D coded caching problem, the optimal maximum average delivery rate with respect to the uniform demand is upper bounded by*

$$R^{r*}_{ave,req\text{-}rob} \leq \mathbb{E}_{D'_{\mathcal{K}}}\left\{\frac{K\binom{K-1}{t} - \sum_{i=1}^{K}\binom{K-1-N_e(D'_{\mathcal{K}\setminus\{i\}})}{t} - f\left[\binom{K-r}{t} - \binom{K-r-1}{t}\right]}{t\binom{K}{t}}\right\}, \tag{6}$$

when $t = \frac{KM}{N}$ is an integer in $[K]$, where f is an integer equal to one if and only if each requester demands a distinct file, i.e., $N_e(D_{\mathcal{R}}) = r$; otherwise, $f = 0$. When $t \notin [K]$, $R^{r*}_{ave,req\text{-}rob}$ is upper bounded by the lower convex envelope of the values in (6) for integer values of $t \in [K]$.

For the maximum worst-case delivery rate, we have

$$R^{r*}_{worst,req\text{-}rob} \leq \max_{D'_{\mathcal{K}}}\left\{\frac{K\binom{K-1}{t} - \sum_{i=1}^{K}\binom{K-1-N_e(D'_{\mathcal{K}\setminus\{i\}})}{t} - f\left[\binom{K-r}{t} - \binom{K-r-1}{t}\right]}{t\binom{K}{t}}\right\}, \tag{7}$$

$$= \begin{cases} \frac{K\binom{K-1}{t} - (K-r)\binom{K-r-1}{t} - r\binom{K-r}{t}}{t\binom{K}{t}}, & r \leq N, \\ \frac{K\binom{K-1}{t} - (2N-r)\binom{K-N}{t} - (K+r-2N)\binom{K-1-N}{t}}{t\binom{K}{t}}, & otherwise, \\ \frac{K\left[\binom{K-1}{t} - \binom{K-1-N}{t}\right]}{t\binom{K}{t}}, & r \geq 2N, \end{cases} \tag{8}$$

where $t = \frac{KM}{N}$ is an integer in $[K]$. When $t \notin [K]$, $R^{r*}_{worst,req\text{-}rob}$ is upper bounded by the lower convex envelope of the values in (7) for integer values of $t \in [K]$.

Proof. The proof of Theorem 1 is provided in Appendix A. □

Theorem 1 is a simple adaptation of an existing scheme; in order to improve its performance, we propose a new scheme, called the *three-category-based scheme*, and obtain Theorem 2 as follows:

Theorem 2. *For the request-robust D2D coded caching problem, the optimal maximum average delivery rate with respect to the uniform demand is upper bounded by*

$$R^{r*}_{ave,req\text{-}rob} \leq \mathbb{E}_{D_{\mathcal{R}}} \left\{ \frac{\sum_{i=\max\{1,t+r-K\}}^{\min\{t-1,r-1\}} \binom{K-r}{t-i} \left[\binom{r}{i+1} - \binom{r-N_e(D_{\mathcal{R}})}{i+1} \right]}{\binom{K}{t}} + \frac{\binom{K-r}{t} N_e(D_{\mathcal{R}})}{\binom{K}{t}} + \frac{r\binom{r-1}{t} - \sum_{i\in\mathcal{R}} \binom{r-1-N_e(D_{\mathcal{R}\setminus\{i\}})}{t}}{t\binom{K}{t}} \right\}, \tag{9}$$

where $t = \frac{KM}{N}$ is an integer in $[K]$. When $t \notin [K]$, $R^{r*}_{ave,req\text{-}rob}$ is upper bounded by the lower convex envelope of the values in (9) for integer values of $t \in [K]$.

Then, for the maximum worst-case delivery rate, we have

$$R^{r*}_{worst,req\text{-}rob} \leq \max_{D_{\mathcal{R}}} \left\{ \frac{\sum_{i=\max\{1,t+r-K\}}^{\min\{t-1,r-1\}} \binom{K-r}{t-i} \left[\binom{r}{i+1} - \binom{r-N_e(D_{\mathcal{R}})}{i+1} \right]}{\binom{K}{t}} + \frac{\binom{K-r}{t} N_e(D_{\mathcal{R}})}{\binom{K}{t}} + \frac{r\binom{r-1}{t} - \sum_{i\in\mathcal{R}} \binom{r-1-N_e(D_{\mathcal{R}\setminus\{i\}})}{t}}{t\binom{K}{t}} \right\}, \tag{10}$$

$$= \begin{cases} \frac{\sum_{i=\max\{1,t+r-K\}}^{\min\{t-1,r-1\}} \binom{K-r}{t-i}\binom{r}{i+1}}{\binom{K}{t}} + \frac{r\binom{K-r}{t}}{\binom{K}{t}} + \frac{r\binom{r-1}{t}}{t\binom{K}{t}}, & r \leq N, \\ \frac{\sum_{i=\max\{1,t+r-K\}}^{\min\{t-1,r-1\}} \binom{K-r}{t-i}\left[\binom{r}{i+1} - \binom{r-N}{i+1}\right]}{\binom{K}{t}} + \frac{N\binom{K-r}{t}}{\binom{K}{t}} & \\ \quad + \frac{r\binom{r-1}{t} - (2N-r)\binom{r-N}{t} - 2(r-N)\binom{r-1-N}{t}}{t\binom{K}{t}}, & otherwise, \\ \frac{\sum_{i=\max\{1,t+r-K\}}^{\min\{t-1,r-1\}} \binom{K-r}{t-i}\left[\binom{r}{i+1} - \binom{r-N}{i+1}\right]}{\binom{K}{t}} + \frac{N\binom{K-r}{t}}{\binom{K}{t}} & \\ \quad + \frac{r\left[\binom{r-1}{t} - \binom{r-1-N}{t}\right]}{t\binom{K}{t}}, & r \geq 2N, \end{cases} \tag{11}$$

where $t = \frac{KM}{N}$ is an integer in $[K]$. When $t \notin [K]$, $R^{r*}_{worst,req\text{-}rob}$ is upper bounded by the lower convex envelope of the values in (10) for integer values of $t \in [K]$.

Proof. In proving Theorem 2, we propose a new scheme, i.e., the three-category-based scheme, where the sub-files are divided into three categories and different delivery signals are designed for each category. The detailed proof can be found in Section 4. □

Remark 1. *In Section 5, we numerically compare the delivery rates of the three-category-based scheme and the adapted YWSC scheme, and it can be seen that the proposed three-category-based scheme outperforms the adapted YWSC scheme in all cases cited (see Section 5 for the cited cases and comparison results).*

In the following theorem, we characterize a converse for the request-robust D2D coded caching problem.

Theorem 3. *For the request-robust D2D coded caching problem, the optimal maximum average delivery rate with respect to the uniform demand and under the constraint of uncoded placement is lower bounded by:*

$$R^{r*}_{ave,req\text{-}rob} \geq \mathbb{E}_{D_{\mathcal{R}}} \left[\frac{\binom{K}{t+1} - \binom{K-N_e(D_{\mathcal{R}})}{t+1}}{\binom{K}{t}} \right], \tag{12}$$

where $t = \frac{KM}{N}$ *is an integer in* $[K]$*, where* $D_{\mathcal{R}}$ *is uniformly distributed over* $\mathcal{D}^{\mathcal{R}}$*. When* $t \notin [K]$*,* $R^{r*}_{ave,req\text{-}rob}$ *is lower bounded by the lower convex envelope of the values in (12) for integer values of* $t \in [K]$*.*

Then, for the maximum worst-case delivery rate, we have that the optimal maximum worst-case delivery rate under the constraint of uncoded placement is lower bounded by

$$R^{r*}_{worst,req\text{-}rob} \geq \frac{\binom{K}{t+1} - \binom{K-\min\{r,N\}}{t+1}}{\binom{K}{t}}, \tag{13}$$

where $t = \frac{KM}{N}$ *is an integer in* $[K]$*. When* $t \notin [K]$*,* $R^{r*}_{worst,req\text{-}rob}$ *is lower bounded by the lower convex envelope of the values in (13) for integer values of* $t \in [K]$*.*

Proof. Similar to [9], we use the converse of the central server version, i.e., System 2, as a converse for the request-robust D2D coded caching problem when the converse is under the constraint of uncoded placement. The detailed proof is provided in Appendix B. □

We compare the rate achieved by the adapted YWSC scheme from Theorem 1 and the converse present in Theorem 3 and obtain a multiplicative gap result of Theorem 4 as follows:

Theorem 4. *For the request-robust D2D coded caching problem, the upper bounds of the optimal maximum average and worst-case rates from Theorem 1 are order optimal within a factor of two under the constraint of uncoded cache placement and within a factor of four in general.*

Proof. The proof of Theorem 4 is given in Appendix D. □

Remark 2. *It is hard to analytically prove that the rate achieved by the three-category-based scheme from Theorem 2 outperforms the rate achieved by the adapted YWSC scheme from Theorem 1 in all cases. However, since in the numerical comparisons in Section 5 the three-category-based scheme performs better than the adapted YWSC scheme in all cases cited, we conjecture that the rate achieved by the three-category-based scheme from Theorem 2 and the converse present in Theorem 3 also follow the multiplicative gap characterized by Theorem 4.*

4. A Novel Achievable Scheme, i.e., Proof of Theorem 2

In this section, we present an achievable scheme for the request-robust D2D coded caching problem. The scheme achieves the rate stated in Theorem 2. We will first provide a general achievable scheme, which is based on uncoded cache placement and exploiting common demands [11] and one-shot delivery [9]. Then, we will characterize the performance of the proposed scheme and show that for any requester set $\mathcal{R}$ and corresponding request vector $D_{\mathcal{R}}$, the proposed three-category-based scheme achieves the rate

$$\begin{aligned} R^{D_{\mathcal{R}}}_{\text{req-rob}} =& \frac{\sum_{i=\max\{1,t+r-K\}}^{\min\{t-1,r-1\}} \binom{K-r}{t-i}\left[\binom{r}{i+1} - \binom{r-N_e(D_{\mathcal{R}})}{i+1}\right]}{\binom{K}{t}} \\ &+ \frac{\binom{K-r}{t}N_e(D_{\mathcal{R}})}{\binom{K}{t}} + \frac{r\binom{r-1}{t} - \sum_{i\in\mathcal{R}}\binom{r-1-N_e(D_{\mathcal{R}\setminus\{i\}})}{t}}{t\binom{K}{t}}, \end{aligned} \tag{14}$$

which, with the explicit characterization of the maximum worst-case delivery rate in Section 4.2, immediately proves Theorem 2. Finally, we will provide an example to aid in a better understanding of the proposed three-category-based scheme.

4.1. General Scheme

For the placement phase, because the central server does not know the number of requesters r, we use the highly adaptable MAN uncoded symmetric placement scheme

described in Definition 1, denoted as $\mathcal{M}_{\text{MAN}}$. In the following, we restrict to integer values of $t \in [1:K]$. For cache size M where $t = KM/N$ is not an integer, memory-sharing will be performed [3,8].

For the delivery phase, let the set of requesters be $\mathcal{R}$ with size r. The r requesters each demand a single file. The delivery strategy is divided into the following steps:

(a) **Determining the leading requesters**: Each user $k \in \mathcal{K} \setminus \mathcal{R}$ who does not request arbitrarily selects a subset of $N_e(D_{\mathcal{R}})$ requesters, denoted by $\mathcal{U}'^k = \{u_1'^k, ..., u_{N_e(D_{\mathcal{R}})}'^k\} \subseteq \mathcal{R}$, that request $N_e(D_{\mathcal{R}})$ distinct files. Using the idea of leaders from [11], we name these requesters as the *leading requesters of User k*.

(b) **Splitting the sub-files into three categories**: Recall that each sub-file is denoted as $W_{n,\mathcal{T}}$ and is cached by only users in $\mathcal{T}$. If $\mathcal{T} \subseteq \mathcal{K} \setminus \mathcal{R}$, then this sub-file belongs to the first category, which is the set of sub-files that are only cached by users who do not request any files. If $\mathcal{T}$ contains some elements from $\mathcal{R}$ and some elements from $\mathcal{K} \setminus \mathcal{R}$, then this sub-file belongs to the second category, which is the set of sub-files that are cached by both requesters and non-requesters. Finally, if $\mathcal{T} \subseteq \mathcal{R}$, then this sub-file belongs to the third category, which is the set of sub-files that are only cached by users who make file requests.

The three categories may not all exist or be required by requesters, and it depends on the value of r and t. When $t \in [K-r+1, K]$ or $r = K$, the first category does not exist. When $t = 1$ or $r = K$, the second category does not exist, and when $r = 1$ or $t = K$, the second category is not required. When $t \in [r+1, K]$, the third category does not exist, and when $t = r$ or $r = 1$, the third category is not required.

(c) **Transmitting signals for the sub-files in the three categories**: we will discuss the delivery scheme for the sub-files in each of the three categories.

(i) For the sub-files in the first category needed by Requester $k, k \in \mathcal{R}$, since these sub-files are not cached in any of the requesters, the users in $\mathcal{K} \setminus \mathcal{R}$ who cache these sub-files transmit them in an uncoded form. Suppose $W_{d_k,\mathcal{T}}$ is requested by User $k \in \mathcal{R}$, $\mathcal{T} \subseteq \mathcal{K} \setminus \mathcal{R}$, any of the t users in $\mathcal{T}$ can transmit the sub-file in an uncoded form. However, we adopt the file-splitting strategy in [9] and allow each user in $\mathcal{T}$ to transmit $1/t$ part of the sub-file, i.e., $W_{d_k,\mathcal{T}}$ is divided into t pieces, each consisting of $\frac{F}{t\binom{K}{t}}$ numbers of bits. The pieces are denoted as $W_{d_k,\mathcal{T},a}$, $a \in \mathcal{T}$, and User a transmits

$$X_a^{1\text{st},d_k,\mathcal{T}} = W_{d_k,\mathcal{T},a}. \tag{15}$$

We notice that $W_{d_k,\mathcal{T},a}$ may be needed by other requesters, i.e., there may be other requesters that request file d_k also. Hence, we let each user a transmit in sequence $X_a^{1\text{st},d_k,\mathcal{T}}$ for all $k \in \mathcal{U}'^a$. Hence, the rate of transmitting total bits for the sub-files in the first category is

$$R^{1\text{st}} = \frac{t\binom{K-r}{t}N_e(D_{\mathcal{R}})}{t\binom{K}{t}} = \frac{\binom{K-r}{t}N_e(D_{\mathcal{R}})}{\binom{K}{t}}, \tag{16}$$

because for each $\mathcal{T}$, every user $a \in \mathcal{T}$ transmits $N_e(D_{\mathcal{R}})\frac{F}{t\binom{K}{t}}$ bits, and there are t users in each $\mathcal{T}$, and a total of $\binom{K-r}{t}$ number of $\mathcal{T}$ that are subsets of $\mathcal{K} \setminus \mathcal{R}$.

(ii) Consider a sub-file in the second category needed by Requester $k, k \in \mathcal{R}$, denoted as $W_{d_k,\mathcal{T}}$, where $k \notin \mathcal{T}$. Denote the set of elements in $\mathcal{T}$ that are in $\mathcal{R}$ as $\mathcal{B}$, whose size is denoted as i, and we have $1 \le i \le r-1$, because we know at least User k who is requesting a file is not in $\mathcal{T}$. Further denote the set of elements of $\mathcal{T}$ that are in $\mathcal{K} \setminus \mathcal{R}$ as $\hat{\mathcal{B}}$, whose size is $t-i$, and we have $1 \le t-i \le K-r$. Hence, $\mathcal{T}$ can be written as $\mathcal{T} = \mathcal{B} \bigcup \hat{\mathcal{B}}$. Furthermore, i must satisfy $i \in [\max\{1, t+r-K\}, \min\{t-1, r-1\}]$.

Consider the set $\bar{\mathcal{B}} \triangleq \{k\} \bigcup \mathcal{B}$, which is a set consisting of $i+1$ requesters. The sub-file $W_{d_x,\hat{\mathcal{B}}\cup\bar{\mathcal{B}}\setminus\{x\}}$ is needed for $x \in \bar{\mathcal{B}}$. We ask that this sub-file be transmitted by the $t-i$ non-requesters, and $W_{d_x,\hat{\mathcal{B}}\cup\bar{\mathcal{B}}\setminus\{x\}}$ for any $x \in \mathcal{B}$ be divided into $t-i$ equal-length disjoint sub-pieces of $\frac{F}{(t-i)\binom{K}{t}}$ bits, which are denoted by $W_{d_x,\hat{\mathcal{B}}\cup\bar{\mathcal{B}}\setminus\{x\},b'}$ where $b \in \hat{\mathcal{B}}$. Hence, if User $b \in \hat{\mathcal{B}}$ transmits

$$X_b^{\text{2nd},d_x,\hat{\mathcal{B}}\cup\bar{\mathcal{B}}\setminus\{x\}} = \bigoplus_{x\in\bar{\mathcal{B}}} W_{d_x,\hat{\mathcal{B}}\cup\bar{\mathcal{B}}\setminus\{x\},b'} \tag{17}$$

the sub-piece retrieval can be accomplished for each requester in $\bar{\mathcal{B}}$ since User x has all the sub-pieces on the RHS of (17), except for $W_{d_x,\hat{\mathcal{B}}\cup\bar{\mathcal{B}}\setminus\{x\},b'}$.

We ask each user $b \in \mathcal{K}\setminus\mathcal{R}$ to transmit $X_b^{\text{2nd},d_x,\hat{\mathcal{B}}\cup\bar{\mathcal{B}}\setminus\{x\}}$ in sequence, only if $\bar{\mathcal{B}} \cap \mathcal{U}'^b \neq \emptyset$, i.e., user b will not transmit if the set $\bar{\mathcal{B}}$ consists of only non-leading requesters. We now count the amount of transmission for the second category. For a fixed i, the number of $\hat{\mathcal{B}}$, which is of size $t-i$, is $\binom{K-r}{t-i}$. For each $u \in \hat{\mathcal{B}}$, the number of transmitted bits is $\binom{r}{i+1} - \binom{r-N_e(D_{\mathcal{R}})}{i+1}$ times the size of a sub-piece, and there are a total of $t-i$ users in $\hat{\mathcal{B}}$. Hence, the rate of transmitting all the bits for the sub-files in the second category is

$$R^{\text{2nd}} = \frac{\sum_{i=\max\{1,t+r-K\}}^{\min\{t-1,r-1\}} \binom{K-r}{t-i}\left[\binom{r}{i+1} - \binom{r-N_e(D_{\mathcal{R}})}{i+1}\right]}{\binom{K}{t}}. \tag{18}$$

The next lemma shows that the scheme proposed satisfies the decodability constraint, even for the non-leading requesters.

Lemma 1. *The scheme proposed for the sub-files in the second category satisfies the decodability constraint, i.e., (2).*

The proof of Lemma 1 is based on showing the equivalence of the designed scheme and that in [11]. Hence, we state the following remark first:

Remark 3. *Notice that when the sub-files are in the second category and the parameter t is fixed, for each partition integer i, a user $b \in \hat{\mathcal{B}}$ generates its codewords exclusively from the sub-pieces $W_{d_k,\hat{\mathcal{B}}\cup\mathcal{B},b'}$ and there exist $\binom{r}{i}$ such sub-pieces in its cache. In addition, for any $c \in \hat{\mathcal{B}}\setminus\{b\}$, we have $W_{d_k,\hat{\mathcal{B}}\cup\mathcal{V}^b,b} \cap W_{d_l,\hat{\mathcal{B}}\cup\mathcal{V}^c,c} = \emptyset$ for any $\mathcal{V}^b, \mathcal{V}^c \subseteq \mathcal{R}, |\mathcal{V}^b| = |\mathcal{V}^c| = i, k \in \mathcal{V}^b, l \in \mathcal{V}^c$. That is to say, users in $\hat{\mathcal{B}}$ generate their codewords based on non-overlapping libraries of size $(t-i)N\binom{r}{i}\frac{F}{(t-i)\binom{K}{t}} = N\binom{r}{i}F/\binom{K}{t}$ bits. Also, observe that the cache of requester k contains $(t-i)\binom{r-1}{i-1}$ such $W_{d_k,\hat{\mathcal{B}}\cup\mathcal{V}^b,b}$ sub-pieces, which amount to $N(t-i)\binom{r-1}{i-1}\frac{F}{(t-i)\binom{K}{t}} = N\binom{r-1}{i-1}\frac{F}{\binom{K}{t}} = Ni\binom{r}{t}\frac{F}{r\binom{K}{t}}$ bits.*

Therefore, the proposed scheme is in fact composed of $(t-i)$ shared-link models [3] each with N files of size $F' = \binom{r}{i}F/\binom{K}{t}$ bits and $K' = r$ users with caches of size $M' = Ni/r$ units each. The corresponding parameter for each model is found to be $t' = \frac{K'M'}{N} = i$. To ensure the existence of sub-files in the second category, the partition integer must satisfy $i \in [\max\{1, t+r-K\}, \min\{t-1, r-1\}]$. Hence, for every $i \in [\max\{1, t+r-K\}, \min\{t-1, r-1\}]$, summing the achievable rates R_{sl}, which is defined as follows from [11]:

$$R_{sl} = \frac{\binom{K}{t+1} - \binom{K-N_e(\boldsymbol{d})}{t+1}}{\binom{K}{t}}, \tag{19}$$

of each $b \in \hat{\mathcal{B}}$ shared-link sub-system and replacing the shared-link system parameters F, K, M, t, and $N_e(\boldsymbol{d})$ with F', K', M', t', and $N_e(D_{\mathcal{R}})$, respectively, we obtain (18).

We now prove that for each partition integer $i \in [\max\{1, t+r-K\}, \min\{t-1, r-1\}]$, each requester k is able to decode the needed sub-files in the second category with the partition integer i upon receiving the codewords $X_b^{2\text{nd},d_k,\hat{\mathcal{B}}\cup\bar{\mathcal{B}}\setminus\{k\}}$ for all $b \in \hat{\mathcal{B}}$.

When k is a leading requester of the user b who does not request, i.e., $k \in \mathcal{U}'^b$, it can decode any required sub-piece $W_{d_k,\hat{\mathcal{B}}\cup\mathcal{P}^k,b}$, where $\mathcal{P}^k \subseteq \mathcal{R}\setminus\{k\}$, $|\mathcal{P}^k| = i$, from $X_b^{2\text{nd},d_k,\hat{\mathcal{B}}\cup\mathcal{P}^k}$, which is broadcast from user b by performing

$$W_{d_k,\hat{\mathcal{B}}\cup\mathcal{P}^k,b} = \left(\bigoplus_{x\in\mathcal{P}^k} W_{d_x,\hat{\mathcal{B}}\cup\mathcal{P}^k\cup\{k\}\setminus\{x\},b}\right) \bigoplus X_b^{2\text{nd},d_k,\hat{\mathcal{B}}\cup\mathcal{P}^k},$$

as can be seen from (17).

When $k \notin \mathcal{U}'^b$, it is less straightforward for the non-leading requester k to decode the needed sub-files, because not all of the corresponding codewords $X_b^{2\text{nd},d_k,\hat{\mathcal{B}}\cup\mathcal{P}^k}$ for its required sub-pieces $W_{d_k,\hat{\mathcal{B}}\cup\mathcal{P}^k,b}$ are directly broadcast from user b. However, Requester k can generate these codewords simply based on the codewords received. To show this, we reformulate the following Lemma 2 from [11] (Lemma 1), which is applied to the codewords broadcast by the user b with the partition integer i.

Lemma 2. *Given an integer t, a partition integer i, a subset $\hat{\mathcal{B}} \subseteq \mathcal{K}\setminus\mathcal{R}$ of size $t-i$, a user $b \in \hat{\mathcal{B}}$, and a set of leading requesters $\mathcal{U}'^b$, for any subset $\mathcal{C}^b \subseteq \mathcal{R}$ that includes $\mathcal{U}'^b$, let $\mathcal{V}_F^b$ be family of all subsets $\mathcal{V}^b$ of $\mathcal{C}^b$ such that each requested file in $D_{\mathcal{R}}$ is requested by exactly one user in $\mathcal{V}^b$. The following equation holds:*

$$\bigoplus_{\mathcal{V}^b\in\mathcal{V}_F^b} X_b^{2nd,d_k,\hat{\mathcal{B}}\cup\{\mathcal{C}^b\setminus\mathcal{V}^b\}\setminus\{k\}} = 0, \tag{20}$$

if each $X_b^{2nd,d_k,\hat{\mathcal{B}}\cup\{\mathcal{C}^b\setminus\mathcal{V}^b\}\setminus\{k\}}$ is defined in (17).

Proof. As we mentioned in Remark 3, for the sub-files needed in the second category, when the parameters t and i are fixed, the proposed scheme, in fact, corresponds to $(t-i)$ shared-link schemes. Thus, Ref. [11] (Lemma 1) can directly be applied to each b-th shared-link scheme. □

Let us now consider any subset $\bar{\mathcal{B}}$ of $i+1$ non-leading requesters of user b such that $\bar{\mathcal{B}} \cap \mathcal{U}'^b = \emptyset$. Using (20), the following equation can be derived:

$$X_b^{2\text{nd},d_k,\hat{\mathcal{B}}\cup\bar{\mathcal{B}}\setminus\{k\}} = \bigoplus_{\mathcal{V}^b\in\mathcal{V}_F^b\setminus\{\mathcal{U}'^b\}} X_b^{2\text{nd},d_k,\hat{\mathcal{B}}\cup\{\mathcal{C}^b\setminus\mathcal{V}^b\}\setminus\{k\}}, \tag{21}$$

where $\mathcal{C}^b = \mathcal{B}'^i \cup \mathcal{U}'^b$. Equation (21) shows that the codeword $X_b^{2\text{nd},d_k,\hat{\mathcal{B}}\cup\bar{\mathcal{B}}\setminus\{k\}}$ can be directly computed from the broadcast codewords transmitted to all the leading requesters of b, because all codewords on the RHS of (21) are directly broadcasted by user b. Hence, each requester k can obtain the value $X_b^{2\text{nd},d_k,\hat{\mathcal{B}}\cup\bar{\mathcal{B}}\setminus\{k\}}$ for any subset $\bar{\mathcal{B}}$ of $i+1$ requesters and can decode its demanded sub-pieces as discussed before. Hence, Lemma 1 is proved.

(iii) Lastly, we consider the sub-files in the third category. Since all the sub-files needed for delivery are only cached in the requesters, the transmission will happen only among requesters. This is equivalent to the D2D coded caching model considered in [9], and we adopt its achievable scheme with uncoded cache placement and one-shot delivery, which we call the *Yapar–Wan–Schaefer–Caire (YWSC) scheme*.

More specifically, during the delivery phase, each sub-file is divided into t equal-length disjoint sub-pieces of $\frac{F}{t\binom{K}{i}}$ bits, which are denoted by $W_{n,\mathcal{T},i}$, $i \in \mathcal{T}$. Further, each user i

in requester set $\mathcal{R}$, where $|\mathcal{R}| = r$, selects an arbitrary subset of $N_e(D_{\mathcal{R}\setminus\{i\}})$ users from $\mathcal{R}\setminus\{i\}$, denoted by $\mathcal{U}^i = \{u_1^i, ..., u_{N_e(D_{\mathcal{R}\setminus\{i\}})}^i\}$, which request $N_e(D_{\mathcal{R}\setminus\{i\}}))$ distinct files and are referred to as *leading demanders of user i*. Then, for all subsets $\mathcal{E}^i \subseteq \mathcal{R}\setminus\{i\}$ of t users, each user i transmits

$$X_i^{\text{3rd}} = \{Y_{\mathcal{E}^i}^i\}_{\mathcal{E}^i\cap\mathcal{U}^i\neq\emptyset}, \tag{22}$$

where

$$Y_{\mathcal{E}^i}^i = \bigoplus_{k\in\mathcal{E}^i} W_{d_k,\{\mathcal{E}^i\cup\{i\}\}\setminus\{k\},i}. \tag{23}$$

In other words, since all users $k \in \mathcal{E}^i$ shall retrieve the needed sub-pieces $W_{d_k,\{\mathcal{E}^i\cup\{i\}\}\setminus\{k\},i}$ from the transmissions of User i, using the idea of leaders from [11], each user i only needs to transmit in sequence the codewords $Y_{\mathcal{E}^i}^i$ for all subsets $\mathcal{E}^i$ such that $\mathcal{E}^i\cap\mathcal{U}^i \neq \emptyset$, i.e., X_i^{3rd}.

As a result, when User k is a leading demander for User i, i.e., $k \in \mathcal{U}^i$, it can decode any needed sub-piece $W_{d_k,\mathcal{G}^k\cup\{i\},i}$, where $\mathcal{G}^k \subseteq \mathcal{R}\setminus\{i,k\}$, $|\mathcal{G}^k| = t-1$, from $Y_{\mathcal{G}^k\cup\{k\}}^i$, which is transmitted from User i by performing

$$W_{d_k,\mathcal{G}^k\cup\{i\},i} = \left(\bigoplus_{x\in\mathcal{G}^k} W_{d_x,\{\mathcal{G}^k\cup\{i,k\}\}\setminus\{x\},i}\right) \bigoplus Y_{\mathcal{G}^k\cup\{k\}}^i.$$

When User k is not a leading demander for User i, using the equation

$$\bigoplus_{\mathcal{V}^i\in\mathcal{V}_F^i} Y_{\mathcal{C}^i\setminus\mathcal{V}^i}^i = 0,$$

proved in [9] (Lemma 1), where subset $\mathcal{C}^i \subseteq \mathcal{R}\setminus\{i\}$ includes $\mathcal{U}^i$ and $\mathcal{V}_F^i$ denotes the family of all subsets $\mathcal{V}^i$ of $\mathcal{C}^i$ such that each requested file in $D_{\mathcal{R}\setminus\{i\}}$ is requested by exactly one user in $\mathcal{V}^i$, each user k can decode its requested sub-piece through obtaining the value $Y_{\mathcal{E}^i}^i$, for any subset $\mathcal{E}^i$ of t users such that $\mathcal{E}^i\cap\mathcal{U}^i = \emptyset$, from the broadcast codewords by the following equation:

$$Y_{\mathcal{E}^i}^i = \bigoplus_{\mathcal{V}^i\in\mathcal{V}_F^i\setminus\{\mathcal{U}^i\}} Y_{\mathcal{C}^i\setminus\mathcal{V}^i}^i, \tag{24}$$

where $\mathcal{C}^i = \mathcal{E}^i\cup\mathcal{U}^i$. To sum up, for each $i \in \mathcal{R}\setminus\{k\}$, User k decodes its requested sub-pieces by following either one of the strategies above, depending on whether it is a leading demander of User i or not.

For each user $i \in \mathcal{R}$, the size of the transmitted signal amounts to $\binom{r-1}{t} - \binom{r-1-N_e(D_{\mathcal{R}\setminus\{i\}})}{t}$ times the size of a sub-piece. Hence, the rate of transmitting all the bits for the sub-files in the third category is

$$R^{\text{3rd}} = \frac{r\binom{r-1}{t} - \sum_{i\in\mathcal{R}}\binom{r-1-N_e(D_{\mathcal{R}\setminus\{i\}})}{t}}{t\binom{K}{t}}, \tag{25}$$

In other words, the equivalence of the above scheme and that of [9] can be seen in the following remark:

Remark 4. *For the sub-files in the third category, the proposed scheme is equivalent to that of [9]. The central server has a library of N files, $\{W_{n,\mathcal{T}}|\mathcal{T}\subset\mathcal{R}, |\mathcal{T}| = t\}$, and each file has $\tilde{F} \triangleq \binom{r}{t}\frac{F}{\binom{K}{t}}$ bits. There are r D2D users in the system, each requesting a single file. Based on the MAN uncoded symmetric placement scheme adopted, the placement is the same as [9], where each sub-file has the*

size of $\frac{F}{\binom{K}{t}}$*, which is equal to* $\frac{\tilde{F}}{\binom{\tilde{K}}{t}}$*, and sub-file* $W_{n,\mathcal{T}}$ *is placed in users in* $\mathcal{T}$*. Thus, each user caches a total of* $N\binom{r-1}{t-1}$ *sub-files. Hence, each user has a memory of*

$$\tilde{M} = N\binom{r-1}{t-1}\frac{F}{\binom{K}{t}}$$

bits, and we can check that

$$\frac{r\tilde{M}}{N\tilde{F}} = t,$$

just as in [9]. Hence, for the sub-files in the third category, the equivalence between the proposed cache placement scheme and that of [9] is established. In other words, corresponding to the parameters $(F, N, K, MF, t, and\ Ne(\mathbf{d}))$ *in [9], which denote the file size, number of files, number of users, cache size of each user, the parameter defined as* $t = \frac{KMF}{NF}$*, and the number of users requesting different files, respectively, we have* $\left(\tilde{F}, N, r, \tilde{M}, t, and\ \tilde{Ne}(D_{\mathcal{R}})\right)$ *in our problem.*

Summing up the three sub-schemes mentioned above, every requester can decode its requested file. Meanwhile, combining the rate from (16), (18), and (25), the total rate of this delivery scheme is $R^{D_{\mathcal{R}}}_{\text{req-rob}} = R^{1\text{st}} + R^{2\text{nd}} + R^{3\text{rd}}$, which results in the rate stated in (14). The description of this subsection with the explicit characterization of the maximum worst-case delivery rate in Section 4.2, directly proves Theorem 2. More specifically, since the proposed scheme is symmetric with respect to the users, any $\mathcal{R}$ with $|\mathcal{R}| = r$ will offer the same average delivery rate $R^{\mathcal{R}}_{\text{ave,req-rob}}$ and the same worst-case delivery rate $R^{\mathcal{R}}_{\text{worst,req-rob}}$.

Remark 5. *The difference in transmitting the sub-files in the three categories is that the transmissions of the sub-files in the first and third categories both adopt the one-shot delivery scheme in [9], while the transmission of the sub-files in the second category adopts the common demands scheme in [11]. Moreover, the transmission of the sub-files in the first category is in uncoded form, while the transmissions of the sub-files in the second and third categories are both with coded multi-casting opportunity.*

We formally write the three-category-based scheme in Algorithm 2.

Algorithm 2 Three-category-based Scheme (N, K, M)

procedure PLACEMENT($W_1, ..., W_N$)
Apply Algorithm 1 MAN Uncoded Symmetric Placement Scheme $(N, K, M, W_{[N]})$
end procedure
procedure DELIVERY($\mathcal{R}, D_{\mathcal{R}}$)
$r \leftarrow |D_{\mathcal{R}}|$
$t \leftarrow KM/N$
$N_e(D_{\mathcal{R}}) \leftarrow$ the number of distinct elements in $D_{\mathcal{R}}$
for $k \in [K] \setminus \mathcal{R}$ **do**
  $\mathcal{U}'^k \leftarrow \{u'^k_1, ..., u'^k_{N_e(D_{\mathcal{R}})}\}$
end for
(i) For sub-files in the first category:
$\mathcal{T} \leftarrow \{\mathcal{T} \subseteq [K] \setminus \mathcal{R} : |\mathcal{T}| = t\}$
for $\mathcal{T} \in \mathcal{T}$ **do**
  for $n \in [N]$ **do**
    Divide sub-file $W_{n,\mathcal{T}}$ into t disjoint sub-pieces $(W_{n,\mathcal{T},a} : a \in \mathcal{T})$ with equal size
  end for
  for $a \in \mathcal{T}$ **do**
    for $s \in \mathcal{U}'^a$ **do**
      User a transmit $X^{1\text{st},d_s,\mathcal{T}}_a = W_{d_s,\mathcal{T},a}$

end for
end for
end for
(ii) For sub-files in the second category:
for $\in [\max\{1, t+r-K\}, \min\{t-1, r-1\}]$ **do**
  $\mathcal{B} \leftarrow \{\hat{\mathcal{B}} \subseteq [K] \setminus \mathcal{R} : |\hat{\mathcal{B}}| = t-i\}$
  for $\hat{\mathcal{B}} \in \mathcal{B}$ **do**
    for $\mathcal{B} \subset \mathcal{R} : |\mathcal{B}| = i$ **do**
      for $n \in [N]$ **do**
        Divide sub-file $W_{n,\mathcal{B}\cup\hat{\mathcal{B}}}$ into $t-i$ disjoint sub-pieces $(W_{n,\mathcal{B}\cup\hat{\mathcal{B}},b} : b \in \hat{\mathcal{B}})$ with equal size
      end for
    end for
    for $b \in \hat{\mathcal{B}}$ **do**
      for $\bar{\mathcal{B}} \subseteq \mathcal{R} : |\bar{\mathcal{B}}| = i+1$ **do**
        if $\bar{\mathcal{B}} \cap \mathcal{U}'^b == \emptyset$ **then**
          continue
        else
          User b transmits $X_b^{\text{2nd},d_x,\hat{\mathcal{B}}\cup\bar{\mathcal{B}}\setminus\{x\}} = \bigoplus_{x\in\bar{\mathcal{B}}} W_{d_x,\hat{\mathcal{B}}\cup\bar{\mathcal{B}}\setminus\{x\},b}$
        end if
      end for
    end for
  end for
end for
(iii) For sub-files in the third category:
$\mathcal{G} \leftarrow \{\mathcal{G} \subseteq [K] : |\mathcal{G}| = t\}$
for $n \in [N]$ **do**
  for $\mathcal{G} \in \mathcal{G}$ **do**
    Divide sub-file $W_{n,\mathcal{G}}$ into t disjoint sub-pieces $(W_{n,\mathcal{G},i} : i \in \mathcal{G})$ with equal size
  end for
end for
for $i \in \mathcal{R}$ **do**
  $N_e(D_{\mathcal{R}\setminus\{i\}}) \leftarrow$ the number of distinct elements in $D_{\mathcal{R}\setminus\{i\}}$
  $\mathcal{U}^i \leftarrow \{u_1^i, ..., u_{N_e(D_{\mathcal{R}\setminus\{i\}})}^i\}$
  for $\mathcal{E}^i \subseteq \mathcal{R}\setminus\{i\} : |\mathcal{E}^i| = t$ users **do**
    if $\mathcal{E}^i \cap \mathcal{U}^i == \emptyset$ **then**
      continue
    else
      User i transmits $Y_{\mathcal{E}^i}^i = \bigoplus_{k\in\mathcal{E}^i} W_{d_k,\{\mathcal{E}^i\cup\{i\}\}\setminus\{k\},i}$
    end if
  end for
end for
end procedure

4.2. The Maximum Worst-Case Delivery Rate

In this subsection, we characterize the performance of the proposed three-category-based scheme for the maximum worst-case delivery rate. The characterization is based on the observation that the binomial coefficient $\binom{n}{m}$ demonstrates a strictly ascending pattern with respect to n.

For the request-robust D2D coded caching problem, when $N, K, M, r, and\ \mathcal{R}$ do not change, since the upper bound rate $R_{\text{req-rob}}^{D_\mathcal{R}}$ from (14) decreases as $N_e(D_\mathcal{R})$ decreases,

the upper bound rate for the maximum worst-case delivery rate is the one at the maximum value of $N_e(D_{\mathcal{R}})$, i.e.,

$$N_e(D_{\mathcal{R}}) = \min\{r, N\}. \tag{26}$$

Then, for $r \geq 2N$, each file can be requested by at least two requesters, which leads to $N_e(D_{\mathcal{R}\setminus\{i\}}), \forall i \in \mathcal{R}$ having the maximum value of N. Hence, this case maximizes the upper bound rate $R^{D_{\mathcal{R}}}_{\text{req-rob}}$.

For $r < 2N$, a requester i may be the only user requesting a file or not, which leads to $N_e(D_{\mathcal{R}\setminus\{i\}}) = N_e(D_{\mathcal{R}}) - 1$ and $N_e(D_{\mathcal{R}\setminus\{i\}}) = N_e(D_{\mathcal{R}})$, respectively. Hence, due to (26), for $r \leq N$, we have $\binom{r-1-N_e(D_{\mathcal{R}\setminus\{i\}})}{t} = 0$, which proves the case where $r \leq N$.

For $N < r < 2N$, we have $N_e(D_{\mathcal{R}}) = N$ and then obtain that each file cannot be requested by more than two requesters. Thus, due to a total of r requesters, there are $2N - r$ requesters, each of which are the only users requesting a file while each of the remaining $2(r - N)$ requesters are not. Thus, we prove the case where $N < r < 2N$.

4.3. Example

To aid in better understanding, we provide an example to illustrate the proposed scheme in Section 4.1.

Let us consider a case where $N = 2, K = 6, and\ M = 2/3$. Hence, $t = KM/N = 2$. In the placement phase, each file is divided into $\binom{6}{2} = 15$ sub-files, and each sub-file's index is in $\mathcal{T}$, where $\mathcal{T}$ is the family of all sets $\mathcal{T}$ such that $\mathcal{T} \subset [6], |\mathcal{T}| = 2$. The user $k \in [6]$ caches the following sub-files for each $n \in \{1, 2\}$:

$$Z_k = \{W_{n,\mathcal{T}} | \mathcal{T} \in \mathcal{T}, k \in \mathcal{T}\}.$$

In the delivery phase, without a loss of generality, we consider only Users $1, 2, 3, and\ 4$ as requesters, each requesting a single file, i.e., $\mathcal{R} = \{1, 2, 3, 4\}$, and the request vector is $D_{\{1,2,3,4\}} = (1, 2, 1, 1)$. Notice that $r = 4$ and $N_e(D_{\{1,2,3,4\}}) = 2$. Requesters $1, 2, 3, and\ 4$ need the following missing sub-files:

$$\begin{aligned}
W_1 \setminus Z_1 &= \{W_{1,\{2,3\}}, W_{1,\{2,4\}}, W_{1,\{2,5\}}, W_{1,\{2,6\}}, W_{1,\{3,4\}}, W_{1,\{3,5\}}, W_{1,\{3,6\}}, W_{1,\{4,5\}}, W_{1,\{4,6\}}, W_{1,\{5,6\}}\},\\
W_2 \setminus Z_2 &= \{W_{2,\{1,3\}}, W_{2,\{1,4\}}, W_{2,\{1,5\}}, W_{2,\{1,6\}}, W_{2,\{3,4\}}, W_{2,\{3,5\}}, W_{2,\{3,6\}}, W_{2,\{4,5\}}, W_{2,\{4,6\}}, W_{2,\{5,6\}}\},\\
W_1 \setminus Z_3 &= \{W_{1,\{1,2\}}, W_{1,\{1,4\}}, W_{1,\{1,5\}}, W_{1,\{1,6\}}, W_{1,\{2,4\}}, W_{1,\{2,5\}}, W_{1,\{2,6\}}, W_{1,\{4,5\}}, W_{1,\{4,6\}}, W_{1,\{5,6\}}\},\\
W_1 \setminus Z_4 &= \{W_{1,\{1,2\}}, W_{1,\{1,3\}}, W_{1,\{1,5\}}, W_{1,\{1,6\}}, W_{1,\{2,3\}}, W_{1,\{2,5\}}, W_{1,\{2,6\}}, W_{1,\{3,5\}}, W_{1,\{3,6\}}, W_{1,\{5,6\}}\}.
\end{aligned}$$

In Step (a), determining the leading requesters without a loss of generality, we assume that User 5 picks Users $1 and\ 2$ and User 6 picks Users $2 and\ 3$ as the leading requesters, i.e., $\mathcal{U}'^5 = \{1, 2\}, \mathcal{U}'^6 = \{2, 3\}$.

In Step (b), we split the sub-files into three categories. More specifically, $W_{n,\{5,6\}}$, $n \in [N]$ belong to the first category; $W_{n,\{1,5\}}$, $W_{n,\{1,6\}}$, $W_{n,\{2,5\}}$, $W_{n,\{2,6\}}$, $W_{n,\{3,5\}}$, $W_{n,\{3,6\}}$, $W_{n,\{4,5\}}$, $W_{n,\{4,6\}}$, $n \in [N]$ belong to the second category; and $W_{n,\{1,2\}}$, $W_{n,\{1,3\}}$, $W_{n,\{1,4\}}$, $W_{n,\{2,3\}}$, $W_{n,\{2,4\}}$, $W_{n,\{3,4\}}$ belong to the third category.

In Step (c) of delivering signals, for the sub-files belonging to the first category, i.e., sub-files only cached by Users 5 and 6, after splitting these sub-files into two equal-length sub-pieces, from (15), we know that User 5 transmits

$$W_{1,\{5,6\},5}, W_{2,\{5,6\},5},$$

and User 6 transmits

$$W_{1,\{5,6\},6}, W_{2,\{5,6\},6},$$

which are all directly needed by the requesters. The rate $R^{1st} = 1/30 \times 2 \times 2 = 2/15$, which coincides with (16).

For the sub-files in the second category, since i must satisfy $i \in [\max\{1, t+r-K\}, \min\{t-1, r-1\}]$, in this example, we only need to consider $i = 1$, which means that $t - i = 1$. The fact that $t - i = 1$ means that none of these sub-files need to be further split. Take, for example, User 1 in $\mathcal{R}$; it requires the sub-file $W_{1,\{2,5\}}$. Thus, $\mathcal{T} = \{2,5\} = \mathcal{B} \bigcup \hat{\mathcal{B}}$, where $\mathcal{B} = \{2\}$ and $\hat{\mathcal{B}} = \{5\}$. Consider the set $\bar{\mathcal{B}} = \{1\} \bigcup \mathcal{B} = \{1,2\}$ of $i+1 = 2$ users. Following from (17), if User $\{5\}$ transmits

$$W_{1,\{2,5\}} \oplus W_{2,\{1,5\}},$$

both Users 1 and 2 will be able to obtain the sub-file they want, i.e., $W_{1,\{2,5\}}$ and $W_{2,\{1,5\}}$, respectively. Similarly, we may consider all four users in $\mathcal{R}$ and the sub-files that each of them need. We find that if User $\{5\}$ transmits

$$\begin{aligned} &W_{1,\{2,5\}} \oplus W_{2,\{1,5\}}, W_{1,\{3,5\}} \oplus W_{1,\{1,5\}}, \\ &W_{1,\{4,5\}} \oplus W_{1,\{1,5\}}, W_{2,\{3,5\}} \oplus W_{1,\{2,5\}}, \\ &W_{2,\{4,5\}} \oplus W_{1,\{2,5\}}, W_{1,\{4,5\}} \oplus W_{1,\{3,5\}}, \end{aligned}$$

and User $\{6\}$ transmits

$$\begin{aligned} &W_{1,\{2,6\}} \oplus W_{2,\{1,6\}}, W_{1,\{3,6\}} \oplus W_{1,\{1,6\}}, \\ &W_{1,\{4,6\}} \oplus W_{1,\{1,6\}}, W_{2,\{3,6\}} \oplus W_{1,\{2,6\}}, \\ &W_{2,\{4,6\}} \oplus W_{1,\{2,6\}}, W_{1,\{4,6\}} \oplus W_{1,\{3,6\}}, \end{aligned}$$

all the requesting users will be able to decode the necessary sub-files of the second category. However, recall that $\mathcal{U}'^5 = \{1,2\}$ and $\mathcal{U}'^6 = \{2,3\}$. Hence, the signal $W_{1,\{4,5\}} \oplus W_{1,\{3,5\}}$ corresponds to $\bar{\mathcal{B}} = \{3,4,5\}$, which has zero intersection with $\mathcal{U}'^5$. Hence, $W_{1,\{4,5\}} \oplus W_{1,\{3,5\}}$ need not be transmitted and can be calculated due to the fact that

$$(W_{1,\{3,5\}} \oplus W_{1,\{1,5\}}) \oplus (W_{1,\{4,5\}} \oplus W_{1,\{1,5\}}) \oplus (W_{1,\{4,5\}} \oplus W_{1,\{3,5\}}) = 0.$$

Similarly, $W_{1,\{4,6\}} \oplus W_{1,\{1,6\}}$ need not be transmitted due to the fact that

$$(W_{1,\{3,6\}} \oplus W_{1,\{1,6\}}) \oplus (W_{1,\{4,6\}} \oplus W_{1,\{3,6\}}) \oplus (W_{1,\{4,6\}} \oplus W_{1,\{1,6\}}) = 0.$$

Hence, the rate $R^{2nd} = 1/15 \times 5 \times 2 = 2/3$, which coincides with (18).

For the sub-files in the third category, User 1 has $\mathcal{R} \setminus \{1\} = \{2,3,4\}$, which means $N_e(D_{\mathcal{R}\setminus\{1\}}) = 2$. Suppose User 1 picks the leading demanders as $\mathcal{U}^1 = \{2,4\}$. Similarly, User 2 has $\mathcal{R} \setminus \{2\} = \{1,3,4\}$, which means $N_e(D_{\mathcal{R}\setminus\{1\}}) = 1$. Suppose User 2 picks the leading demanders as $\mathcal{U}^2 = \{3\}$. User 3 has $\mathcal{R} \setminus \{3\} = \{1,2,4\}$, which means $N_e(D_{\mathcal{R}\setminus\{1\}}) = 2$. Suppose User 3 picks the leading demanders as $\mathcal{U}^3 = \{1,2\}$, and User 4 has $\mathcal{R} \setminus \{4\} = \{1,2,3\}$, which means $N_e(D_{\mathcal{R}\setminus\{1\}}) = 2$. Suppose User 4 picks the leading demanders as $\mathcal{U}^1 = \{2,3\}$.

Since $t = 2$, we split the sub-files in the third category into two equal-length sub-pieces. For User 1, $\mathcal{E}^1$ is of size $t = 2$ and is a subset of $\mathcal{R} \setminus \{1\} = \{2,3,4\}$. Hence, possible $\mathcal{E}^1$s can be $\{2,3\}, \{2,4\}, \textit{and } \{3,4\}$, which all satisfy the condition of non-zero intersection with $\mathcal{U}^1 = \{2,4\}$. Hence, from (22) and (23), User 1 transmits

$$Y^1_{\{2,3\}} = W_{2,\{1,3\},1} \oplus W_{1,\{1,2\},1}, Y^1_{\{2,4\}} = W_{2,\{1,4\},1} \oplus W_{1,\{1,2\},1}, Y^1_{\{3,4\}} = W_{1,\{1,4\},1} \oplus W_{1,\{1,3\},1}.$$

Similarly, User 3 transmits

$$Y^3_{\{1,2\}} = W_{1,\{2,3\},3} \oplus W_{2,\{1,3\},3}, Y^3_{\{1,4\}} = W_{1,\{3,4\},3} \oplus W_{1,\{1,3\},3}, Y^3_{\{2,4\}} = W_{2,\{3,4\},3} \oplus W_{1,\{2,3\},3},$$

and User 4 transmits

$$Y^4_{\{1,2\}} = W_{1,\{2,4\},4} \oplus W_{2,\{1,4\},4}, Y^4_{\{1,3\}} = W_{1,\{3,4\},4} \oplus W_{1,\{1,4\},4}, Y^4_{\{2,3\}} = W_{2,\{3,4\},4} \oplus W_{1,\{2,4\},4}.$$

As for User 2, $\mathcal{E}^2$ is of size $t = 2$ and is a subset of $\mathcal{R} \setminus \{2\} = \{1,3,4\}$. Hence, possible $\mathcal{E}^2$s can be $\{1,3\}, \{1,4\}, and\ \{3,4\}$. Recall that $\mathcal{U}^2 = \{3\}$; hence, only $\{1,3\}$ and $\{3,4\}$ satisfy the condition of non-zero intersection with $\mathcal{U}^2 = \{3\}$. Hence, from (22) and (23), User 2 transmits

$$Y^2_{\{1,3\}} = W_{1,\{2,3\},2} \oplus W_{1,\{1,2\},2}, \quad Y^2_{\{3,4\}} = W_{1,\{2,4\},2} \oplus W_{1,\{2,3\},2}.$$

The signal $Y^2_{\{1,4\}} = W_{1,\{2,4\},2} \oplus W_{1,\{1,2\},2}$ does not need to be transmitted and can be calculated since $Y^2_{\{1,3\}} \oplus Y^2_{\{3,4\}} \oplus Y^2_{\{1,4\}} = 0$ (cf. (24)). Hence, the rate $R^{\text{3rd}} = 1/30 \times (3 \times 3 + 2) = 11/30$, which coincides with (25).

Thus, the total delivery rate achieved by the proposed scheme for the case $\mathcal{R} = \{1,2,3,4\}$ and the request vector is $D_{\{1,2,3,4\}} = (1,2,1,1)$ is $\frac{2}{15} + \frac{2}{3} + \frac{11}{30} = \frac{7}{6}$. In this case, if we directly use the YWSC scheme [9] by assigning Users 5 and 6 a demand that is the same as Users 1, 3, and 4, i.e., all six users make file requests, and the demand vector is $\mathbf{d} = \{1,2,1,1,1,1\}$, then according to [9] (Equation (14)), the delivery rate is $\frac{6\times10-(3\times5+6)}{30} = \frac{13}{10}$. If we assign Users 5 and 6 with some other demands, the delivery rate will be even higher. Hence, we see that for the request-robust D2D coded caching problem, the proposed scheme performs better than directly applying the YWSC scheme for this example.

The reason why the proposed three-category-based scheme has better performance is that directly applying the YWSC scheme may contain information that is useful for users who do not request files, which is useless for requesters and difficult to be excluded. We provide more details in the following remarks:

Remark 6. *When each user requests a single file ($r = K$), our proposed scheme corresponds to the one originally presented in [9]. The improvement in our scheme is that when $r < K$, we take full advantage of the users who do not request files so that the broadcast codewords are only useful to the requesters. Moreover, through the numerical comparison in Section 5, we find that letting the users who do not request files broadcast pieces of sub-files in the first and second categories, i.e., sub-files in $\{W_{n,\mathcal{A}} | \mathcal{A} \ni \{k\}, k \in \mathcal{K} \setminus \mathcal{R}\}$, incurs a much smaller rate than letting all the users participate in broadcasting the required pieces of sub-files in all three categories.*

Remark 7. *The proposed scheme is symmetric in the placement phase. As mentioned in [9] (Remark 6), under the constraints of uncoded cache placement, the shared link models in [11,27] showed the optimality of symmetry in the placement phase [3]. This symmetry happens in the placement phase before the requesters are identified and reveal their demands, and any asymmetry in the placement will certainly not result in a better worst-case rate. However, due to the file-categorization step (i.e., Step (b)), the delivery phase of the proposed scheme is* asymmetric, *while if the value of $N_e(D_{\mathcal{R}\setminus\{i\}})$ is the same for every $i \in \mathcal{R}$, the delivery phase of directly applying the YWSC scheme (i.e., the adapted YWSC scheme (see Appendix A for the specific scheme)) is symmetric. Interestingly, the asymmetric delivery phase of the proposed scheme outperforms the possibly symmetric delivery phase of directly applying the YWSC scheme both for the maximum average and worst-case delivery rate in all cases cited, as shown in Section 5.*

Remark 8. *The delivery phase of the proposed scheme is actually one-shot, which is defined in [9] as meaning that each user k can recover the i-th needed bit denoted as $W^k_{d_k}(i)$ from its own cache and the transmission of a single other user whose index is $j_k(i)$, i.e., $H(W^k_{d_k}(i) | X_{j_k(i)}, Z_k) = 0$ holds. One-shot delivery allows all users to participate in the transmission without causing users to repeatedly broadcast the same codeword. However, it is difficult to confirm whether the proposed*

scheme is optimal under the constraint of uncoded cache placement and one-shot delivery since the delivery scheme as mentioned in Remark 7 is asymmetric.

Remark 9. *The rate achieved by the three-category-based scheme from (14) outperforms the rate achieved by the adapted YWSC scheme from (A1) in some specific cases. For example, for $r = 2$ and the maximum worst-case delivery rate, when $t \in [2, K-1], r \leq N$, we have*

$$\begin{aligned} R^{D_{\mathcal{R}}}_{\textit{adapted-YWSC}}|_{r=2,t\in[2,K-1],r\leq N} &= \frac{K\binom{K-1}{t} - (K-2)\binom{K-2-1}{t} - 2\binom{K-2}{t}}{t\binom{K}{t}} \qquad (27)\\ &= \frac{(2K-t-1)\binom{K-2}{t-1}}{t\binom{K}{t}} \\ &> \frac{(2K-t-2)\binom{K-2}{t-1}}{t\binom{K}{t}} \\ &= \frac{\binom{K-2}{t-1}\binom{2}{1+1}}{\binom{K}{t}} + \frac{2\binom{K-2}{t}}{\binom{K}{t}} \\ &= R^{D_{\mathcal{R}}}_{\textit{req-rob}}|_{r=2,t\in[2,K-1],r\leq N}, \qquad (28) \end{aligned}$$

where (27) is from (8) and (28) is from (11). Meanwhile, when $t \in [2, K-1], r > N$, i.e., $N = 1$, we have

$$\begin{aligned} R^{D_{\mathcal{R}}}_{\textit{adapted-YWSC}}|_{r=2,t\in[2,K-1],N=1} &= \frac{K\left[\binom{K-1}{t} - \binom{K-2}{t}\right]}{t\binom{K}{t}} \qquad (29)\\ &= \frac{1}{\binom{K}{t}} \cdot \frac{K(K-2)!}{t!(K-t-1)!} \\ &> \frac{1}{\binom{K}{t}} \cdot \frac{(K-1)!}{t!(K-t-1)!} \\ &= \frac{\binom{K-2}{t-1}\binom{2}{1+1}}{\binom{K}{t}} + \frac{\binom{K-2}{t}}{\binom{K}{t}} \\ &= R^{D_{\mathcal{R}}}_{\textit{req-rob}}|_{r=2,t\in[2,K-1],N=1}, \qquad (30) \end{aligned}$$

where (29) is from (8) and (30) is from (11). Moreover, $t \geq K$, i.e., $M \geq N$, is trivial, and when $t = 1$, the three-category-based scheme and the adapted YWSC have the same performance. Hence, for $r = 2$ and the maximum worst-case delivery rate, the three-category-based scheme outperforms the adapted YWSC scheme for all values of K, N, t.

5. Numerical Evaluations

In this section, we compare the rate–memory tradeoff of the three-category-based scheme, the adapted YWSC scheme, and the achievable schemes in [8], adapted to the request-robust D2D coded caching scenario. The adaptation is performed by assigning the users, who do not request, a demand that is most requested by the requesters. We also plot the converse bounds on the optimal average and worst-case delivery rate of the request-robust D2D coded caching problem in Theorem 3.

We consider the cases where the value of K is from 1 to 60. For a fixed K, we consider that the value of N is from 1 to K. The performance metrics are the maximum average delivery rate with respect to the uniform demand and the maximum worst-case delivery rate, i.e., $R^r_{\text{ave,req-rob}}$ and $R^r_{\text{worst,req-rob}}$. We find that, in these cases, our proposed three-category-based scheme outperforms the adapted YWSC scheme and the adapted schemes of [8] for all possible r.

Take the example where $N = 10, K = 30$. As shown in Figure 2, the performance of the proposed scheme is given by the red solid line when $r = 20$ and the purple solid line with dots when $r = 5$. These lines are plotted according to the right-hand side (RHS) of (9) and (10). The performance of the adapted YWSC scheme is given by a blue dash–dot line when $r = 20$ and a cyan dash–dot line with dots when $r = 5$. These lines are plotted according to the RHS of (6) and (7). The proposed converse is given by the black dotted line when $r = 20$ and the orange dotted line with asterisks when $r = 5$. Since the achievable rate in [8] is independent of the demand, the performance of the adapted scheme from [8] does not change with the value of r and is given by the green dashed line. For the maximum worst-case rate, we also provide the lower bound in [8] adapted to the request-robust D2D coded caching scenario with the brown dashed line with dots. It can be seen that our proposed scheme outperforms the adapted YWSC scheme and adapted scheme of [8] in this case; meanwhile, the proposed converse is rather tight compared to the adapted lower bound in [8].

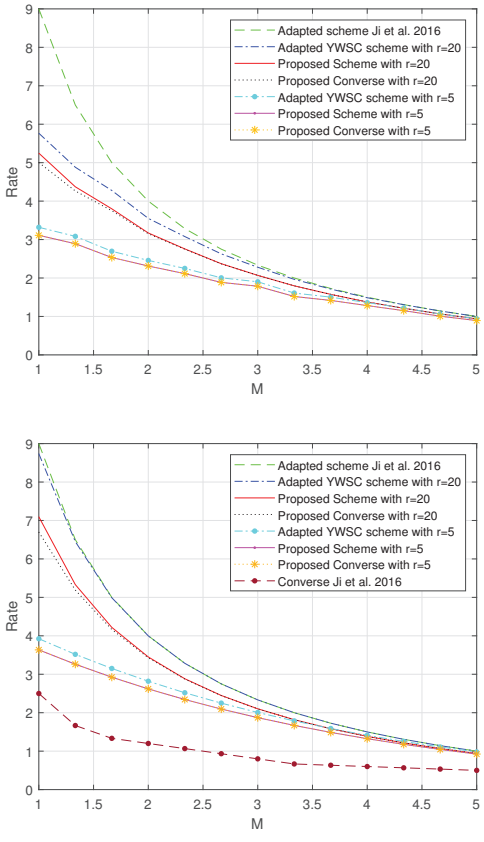


Figure 2. Consider the request-robust D2D coded caching problem from Section 2.1 where $N = 10$ *and* $K = 30$. The figure above is for the tradeoff between memory size and the maximum worst-case delivery rate for different requester numbers. The figure below shows the tradeoff between memory size and the maximum average delivery rate under uniform demand for different requester numbers. The scheme and converse proposed by Ji et al. [8] are both adapted to this request-robust D2D scenario.

6. Conclusions

In this paper, we propose a new problem called request-robust D2D coded caching, where in the delivery phase, though all users in the placement phase are still present, some of them do not request any files. We presented an achievable scheme for this problem based on uncoded cache placement and exploiting common demands and one-shot delivery. The caching strategy is the same as that proposed by Maddah-Ali and Niesen, while the delivery strategy divides the sub-files into three categories, and different delivery signals are designed for each category. We also characterized information-theoretic lower bounds for the request-robust D2D coded caching problem under the constraint of uncoded cache placement. The lower bounds are both for the maximum average delivery rate under uniform demand and the maximum worst-case delivery rate. We adapt the scheme proposed by Yapar et al. for uncoded cache placement and one-shot delivery to the request-robust D2D coded caching problem. The adaptation is performed by assigning the users, who do not request, a demand that is the most requested by the requesters. The performance of the adapted scheme is proved to be order optimal within a factor of two under uncoded cache placement and within a factor of four in general. Finally, by numerical evaluation, we show that the proposed scheme outperforms the known D2D coded caching schemes applied to the request-robust scenario.

Author Contributions: Conceptualization, W.W., Z.T., N.L. and W.K.; methodology, W.W., Z.T., N.L. and W.K.; formal analysis, W.W., Z.T., N.L. and W.K.; writing—original draft preparation, W.W.; writing—review and editing, W.W., N.L. and W.K.; supervision, N.L. and W.K.; funding acquisition, N.L. and W.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the National Natural Science Foundation of China under Grants 62361146853, 62071115, and 62371129 and the Research Fund of the National Mobile Communications Research Laboratory, Southeast University (no. 2024A03).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: Author Zhe Tao was employed by the company Huawei Technologies. The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of the data; in the writing of the manuscript; or in the decision to publish the results.

Appendix A. Adapted Yapar–Wan–Schaefer–Caire Scheme, i.e., Proof of Theorem 1

In this section, we present another achievable scheme that adapts the YWSC scheme from [9] for the request-robust D2D coded caching problem. The scheme achieves the rate stated in Theorem 1 .

We will first provide the general achievable scheme, which is based on the uncoded cache placement and exploiting one-shot delivery [9]. Then, we will characterize the performance of the proposed scheme and show that for any requester set $\mathcal{R}$ and corresponding request vector $D_{\mathcal{R}}$, the adapted YWSC scheme achieves the rate

$$R^{D_{\mathcal{R}}}_{\text{adapted-YWSC}} = \frac{K\binom{K-1}{t} - \sum_{i=1}^{K}\binom{K-1-N_e(D'_{\mathcal{K}\setminus\{i\}})}{t} - f\left[\binom{K-r}{t} - \binom{K-r-1}{t}\right]}{t\binom{K}{t}}, \tag{A1}$$

where $D'_{\mathcal{K}}$ is the request vector of all the users after adaption, and f is an integer equal to one if and only if each requester demands a distinct file; otherwise, $f = 0$. The rate $R^{D_{\mathcal{R}}}_{\text{adapted-YWSC}}$ with the explicit characterization of the maximum worst-case delivery rate in Appendix A.2 immediately proves Theorem 1. Finally, we will provide an example to aid in a better understanding of the adapted YWSC scheme.

Appendix A.1. General Scheme

Similar to Section 4.1, for the placement phase, we still use $\mathcal{M}_{\text{MAN}}$ described in Definition 1. In the following, we restrict to integer values of $t \in [1:K]$. For cache size M where $t = KM/N$ is not an integer, memory-sharing will be performed [3,8].

For the delivery phase, let the set of requesters be $\mathcal{R}$ with size r. The r requesters each demand a single file. We find that the YWSC scheme from [9] is highly adaptable for the request-robust D2D coded caching scenario. The adaptation is performed by assigning the users, who do not request, a demand that is the most requested by the requesters. If there are multiple demands that are the most requested, the users, who do not request, would be assigned one of the demands. Recall that we denote the adapted request vector as $D'_{\mathcal{K}}$, and the adapted request vector of users $\mathcal{K}\setminus\{k\}$ as $D'_{\mathcal{K}\setminus\{k\}}$. For example, if $K = N = 4$, $\mathcal{R} = \{1,2\}, D_{\{1,2\}} = \{1,4\}$, we have $D'_{\{1,2,3,4\}} = \{1,4,1,1\}$ or $D'_{\{1,2,3,4\}} = \{1,4,4,4\}$. Obviously, $N_e(D'_{\mathcal{K}}) = N_e(D_{\mathcal{R}})$ and $N_e(D'_{\mathcal{K}\setminus\{k\}}) \geq N_e(D_{\mathcal{R}\setminus\{k\}})$ for each $k \in \mathcal{R}$.

Next, to adapt the YWSC scheme for the system model, the delivery strategy is divided into the following steps:

(a) **Determining the leading demanders**: Recall that each sub-file is denoted as $W_{n,\mathcal{T}}$ and is cached by only users in $\mathcal{T}$. Each sub-file is divided into t equal-length disjoint sub-pieces of $\frac{F}{t\binom{K}{t}}$ bits, which are denoted by $W_{n,\mathcal{T},i}, i \in \mathcal{T}$. Further, each user i selects an arbitrary subset of $N_e(D'_{\mathcal{K}\setminus\{i\}})$ users from $\mathcal{K}\setminus\{i\}$, denoted by $\mathcal{U}^i = \{u^i_1, ..., u^i_{N_e(D'_{\mathcal{K}\setminus\{i\}})}\}$, which request $N_e(D'_{\mathcal{K}\setminus\{i\}}))$ distinct files and are referred to as *leading demanders of user i*.

Without a loss of generality, we assume that the leading demanders of each user i are determined from the requesters as much as possible and are denoted by $\bar{\mathcal{U}}^i$. For example, for $K = N = 3, \mathcal{R} = \{1,2\}, D_{\{1,2\}} = \{1,3\}, D'_{\{1,2,3\}} = \{1,3,1\}$, the leading demander of User 2 in the assumption is $\bar{\mathcal{U}}^2 = \{1\}$ and should not be $\{3\}$, while the leading demander of User 1 is $\bar{\mathcal{U}}^1 = \{2,3\}$.

(b) **Pre-transmitting signals**: Then, for all subset $\mathcal{E}^i \subseteq \mathcal{K}\setminus\{i\}$ of t users, each user i prepares for transmitting:

$$X_i^{\text{YWSC}} = \{Y^i_{\mathcal{E}^i}\}_{\mathcal{E}^i \cap \mathcal{U}^i \neq \emptyset}, \tag{A2}$$

where

$$Y^i_{\mathcal{E}^i} = \bigoplus_{k\in\mathcal{E}^i} W_{d_k,\{\mathcal{E}^i\cup\{i\}\}\setminus\{k\},i}, \tag{A3}$$

which is the same as (23). In other words, since all users $k \in \mathcal{E}^i$ shall retrieve the needed sub-pieces $W_{d_k,\{\mathcal{E}^i\cup\{i\}\}\setminus\{k\},i}$ from the transmissions of User i, using the idea of leaders from [11], each user i only need to transmit in sequence the codewords $Y^i_{\mathcal{E}^i}$ for all subsets $\mathcal{E}^i$ such that $\mathcal{E}^i \cap \mathcal{U}^i \neq \emptyset$, i.e., X_i^{YWSC}.

(c) **Removing unwanted codewords**: We notice that the codewords that are only useful for the users, who do not request, are unwanted codewords and do not need to be transmitted. Next, we will remove the unwanted codewords.

Recall that each user i only prepares for transmitting the codewords $Y^i_{\mathcal{E}^i}$ where $\mathcal{E}^i \cap \bar{\mathcal{U}}^i \neq \emptyset$. For user i, when $\bar{\mathcal{U}}^i \cap \mathcal{K}\setminus\mathcal{R} = \emptyset$, the signal X_i^{YWSC} does not have unwanted codewords. When $\bar{\mathcal{U}}^i \cap \mathcal{K}\setminus\mathcal{R} \neq \emptyset$, the signal X_i^{YWSC} may have the codewords $Y^i_{\mathcal{E}^i}$ where $\mathcal{E}^i \subseteq \mathcal{K}\setminus\mathcal{R}$, which are only useful for the users, who do not request, and should be removed. Obviously, if and only if each requester demands a distinct file, we have $\bar{\mathcal{U}}^i \cap \mathcal{K}\setminus\mathcal{R} \neq \emptyset$

for only one requester i in $\mathcal{R}$. Hence, for all subsets $\mathcal{E}^i \subseteq \mathcal{K} \setminus \{i\}$ of t users, each user i transmits

$$X_i^{\text{adapted-YWSC}} = \{Y^i_{\mathcal{E}^i}\}_{\mathcal{E}^i \cap \bar{\mathcal{U}}^i \neq \emptyset, \mathcal{E}^i \nsubseteq \mathcal{K} \setminus \mathcal{R}},$$

where $Y^i_{\mathcal{E}^i}$ is defined in (A3).

As a result, when Requester k is a leading demander for User i, i.e., $k \in \mathcal{U}^i$, it can decode any needed sub-piece $W_{d_k, \mathcal{G}^k \cup \{i\}, i}$, where $\mathcal{G}^k \subseteq \mathcal{K} \setminus \{i,k\}$, $|\mathcal{G}^k| = t-1$, from $Y^i_{\mathcal{G}^k \cup \{k\}}$, which is transmitted from User i, by performing

$$W_{d_k, \mathcal{G}^k \cup \{i\}, i} = \left(\bigoplus_{x \in \mathcal{G}^k} W_{d_x, \{\mathcal{G}^k \cup \{i,k\}\} \setminus \{x\}, i} \right) \bigoplus Y^i_{\mathcal{G}^k \cup \{k\}}.$$

When Requester k is not a leading demander for User i, using the equation

$$\bigoplus_{\mathcal{V}^i \in \mathcal{V}^i_F} Y^i_{\mathcal{C}^i \setminus \mathcal{V}^i} = 0,$$

proved in [9] (Lemma 1), where subset $\mathcal{C}^i \subseteq \mathcal{K} \setminus \{i\}$ includes $\bar{\mathcal{U}}^i$ and $\mathcal{V}^i_F$ denotes the family of all subsets $\mathcal{V}^i$ of $\mathcal{C}^i$ such that each requested file in $D'_{\mathcal{K} \setminus \{i\}}$ is requested by exactly one user in $\mathcal{V}^i$, each user k can decode its requested sub-piece through obtaining the value $Y^i_{\mathcal{E}^i}$ for any subset $\mathcal{E}^i$ of t users such that $\mathcal{E}^i \cap \bar{\mathcal{U}}^i = \emptyset$ from the broadcast codewords by the following equation:

$$Y^i_{\mathcal{E}^i} = \bigoplus_{\mathcal{V}^i \in \mathcal{V}^i_F \setminus \{\bar{\mathcal{U}}^i\}} Y^i_{\mathcal{C}^i \setminus \mathcal{V}^i},$$

where $\mathcal{C}^i = \mathcal{E}^i \cup \bar{\mathcal{U}}^i$. To sum up, for each $i \in \mathcal{K} \setminus \{k\}$, Requester k decodes its requested sub-pieces by following either one of the strategies above depending on whether it is a leading demander of User i or not.

For each user $i \in \mathcal{K}$, the size of the transmitted signal amounts to $\binom{K-1}{t} - \binom{K-1-N_e(D'_{\mathcal{K} \setminus \{i\}})}{t}$ times the size of a sub-piece. Meanwhile, if we have $\bar{\mathcal{U}}^i \cap \mathcal{K} \setminus \mathcal{R} \neq \emptyset$ for only one requester i in $\mathcal{R}$, the size of the removed unwanted codewords amounts to $\binom{K-r}{t} - \binom{K-r-1}{t}$ times the size of a sub-piece. Hence, the rate of transmitting total bits coincides (A1).

For the convenience of understanding, we give the algorithm of the adapted YWSC scheme in Algorithm A1.

Algorithm A1 Adapted YWSC Scheme (N, K, M)

procedure PLACEMENT($W_1, ..., W_N$)
Apply Algorithm 1 MAN Uncoded Symmetric Placement Scheme($N, K, M, W_{[N]}$)
end procedure
procedure DELIVERY($\mathcal{R}, D_{\mathcal{R}}$)
$r \leftarrow |D_{\mathcal{R}}|$
$t \leftarrow KM/N$
Assign the users, who do not request, a demand that is the most requested by the requesters and has $D'_{\mathcal{K}} \leftarrow D_{\mathcal{R}}$
$\mathcal{G} \leftarrow \{\mathcal{G} \subseteq [K] : |\mathcal{G}| = t\}$
for $n \in [N]$ **do**
  for $\mathcal{G} \in \mathcal{G}$ **do**
    Divide sub-file $W_{n,\mathcal{G}}$ into t disjoint sub-pieces $(W_{n,\mathcal{G},i} : i \in \mathcal{G})$ with equal size
  end for
end for
for $i \in \mathcal{K}$ **do**
  $N_e(D'_{\mathcal{K} \setminus \{i\}}) \leftarrow$ the number of distinct elements in $D'_{\mathcal{K} \setminus \{i\}}$

$\bar{\mathcal{U}}^i \leftarrow \{u^i_1, ..., u^i_{N_e(D'_{\mathcal{K}\setminus\{i\}})}\}$ from $\mathcal{R}$ as much as possible
for $\mathcal{E}^i \subseteq \mathcal{K} \setminus \{i\} : |\mathcal{E}^i| = t$ users **do**
if $\mathcal{E}^i \cap \bar{\mathcal{U}}^i == \emptyset$ **then**
continue
else if $\mathcal{E}^i \subseteq \mathcal{K} \setminus \mathcal{R}$ **then**
continue
else
User i transmits $Y^i_{\mathcal{E}^i} = \bigoplus_{k \in \mathcal{E}^i} W_{d_k, \{\mathcal{E}^i \cup \{i\}\} \setminus \{k\}, i}$
end if
end for
end for
end procedure

Appendix A.2. The Maximum Worst-Case Delivery Rate

In this section, we characterize the performance of the adapted YWSC scheme for the maximum worst-case delivery rate. The characterization is based on the observation that the binomial coefficient $\binom{n}{m}$ demonstrates a strictly ascending pattern with respect to n.

Again, the adaptation is performed by assigning the users, who do not request, a demand that is the most requested by the requesters, and we have $N_e(D'_{\mathcal{K}}) = N_e(D_{\mathcal{R}})$. Meanwhile, f from (A1) is an integer equal to one if and only if each requester demands a distinct file, i.e., $N_e(D_{\mathcal{R}}) = r$; otherwise, $f = 0$. Hence, for the request-robust D2D coded caching problem, when $N, K, M, r, and\ \mathcal{R}$ do not change, since the upper bound rate $R^{D_{\mathcal{R}}}_{\text{adapted-YWSC}}$ from (A1) decreases as $N_e(D'_{\mathcal{K}})$ decreases, the upper bound rate for the maximum worst-case delivery rate is the one at the maximum value of $N_e(D'_{\mathcal{K}})$, i.e., $N_e(D'_{\mathcal{K}}) = \min\{r, N\}$.

Then, for $r \geq 2N$, each file can be requested by at least two users, which leads to $N_e(D'_{\mathcal{K}\setminus\{k\}}), \forall k \in \mathcal{K}$ having the maximum value of N. Meanwhile, in this case, $f = 0$. Hence, this case maximizes the upper bound rate $R^{D_{\mathcal{R}}}_{\text{adapted-YWSC}}$.

For $r < 2N$, User k may be the only user requesting a file or not, which leads to $N_e(D'_{\mathcal{K}\setminus\{k\}}) = N_e(D'_{\mathcal{K}}) - 1$ and $N_e(D'_{\mathcal{K}\setminus\{k\}}) = N_e(D'_{\mathcal{K}})$, respectively. Due to the adaption, for $r \leq N$, there are $r - 1$ users, each of which are the only users requesting a file while each of the remaining $K - r + 1$ users are not. Meanwhile, in the case $r \leq N$, $N_e(D'_{\mathcal{K}}) = r$ and $f = 1$. Thus, we prove the case where $r \leq N$.

For $N < r < 2N$, we have $N_e(D'_{\mathcal{K}}) = N$. Then, for $D_{\mathcal{R}}$, we obtain that each file cannot be requested by more than two requesters. Thus, due to a total of K users, there are $2N - r$ users, each of which are the only users requesting a file while each of the remaining $K + r - 2N$ users are not. Meanwhile, in this case, $f = 0$. Thus, we prove the case where $N < r < 2N$.

Appendix A.3. Example

To aid in better understanding, we provide an example to illustrate the adapted YWSC scheme in Appendix A.1.

Let us consider a case where $N = 2, K = 4, and\ M = 1$. Hence, $t = KM/N = 2$. In the placement phase, each file is divided into $\binom{4}{2} = 6$ sub-files, and each sub-file's index is in $\mathcal{T}$, where $\mathcal{T}$ is the family of all sets $\mathcal{T}$ such that $\mathcal{T} \subset [4], |\mathcal{T}| = 2$. The user $k \in [6]$ caches the following sub-files for each $n \in \{1, 2\}$:

$$Z_k = \{W_{n,\mathcal{T}} | \mathcal{T} \in \mathcal{T}, k \in \mathcal{T}\}.$$

In the delivery phase, without a loss of generality, we only consider Users 1 *and* 2, as requesters, each requesting a single file, i.e., $\mathcal{R} = \{1, 2\}$, and the request vector is $D_{\{1,2\}} = (1, 2)$. Hence, we consider that the request vector after adaption is

$D'_{\{1,2,3,4\}} = (1,2,1,1)$. Notice that $r = 4$ and $N_e(D'_{\{1,2,3,4\}}) = 2$. Requesters 1 *and* 2 need the following missing sub-files:

$$W_1 \setminus Z_1 = \{W_{1,\{2,3\}}, W_{1,\{2,4\}}, W_{1,\{3,4\}}\},$$
$$W_2 \setminus Z_2 = \{W_{2,\{1,3\}}, W_{2,\{1,4\}}, W_{2,\{3,4\}}\}.$$

In Step (a), since $t = 2$, we split the sub-files into two equal-length sub-pieces. Further, determining the leading demanders from the requesters as much as possible, without a loss of generality, we assume that User 1 picks Users 1 *and* 3, User 2 picks User 1, User 3 picks Users 1 *and* 2, and User 4 picks Users 1 *and* 2 as the leading demanders, i.e., $\tilde{\mathcal{U}}^1 = \{2,3\}$, $\tilde{\mathcal{U}}^2 = \{1\}, \tilde{\mathcal{U}}^3 = \{1,2\}$, *and* $\tilde{\mathcal{U}}^4 = \{1,2\}$. Notice that $N_e(D'_{\mathcal{K}\setminus\{1\}}) = 2, N_e(D'_{\mathcal{K}\setminus\{2\}}) = 1$, $N_e(D'_{\mathcal{K}\setminus\{3\}}) = 2$, *and* $N_e(D'_{\mathcal{K}\setminus\{4\}}) = 2$.

In Step (b), for User 1, $\mathcal{E}^1$ is of size $t = 2$ and is a subset of $\mathcal{K} \setminus \{1\} = \{2,3,4\}$. Hence, possible $\mathcal{E}^1$s can be $\{2,3\}, \{2,4\}$, *and* $\{3,4\}$, which all satisfy the condition of non-zero intersection with $\tilde{\mathcal{U}}^1 = \{2,3\}$. Hence, from (22) and (23), User 1 pre-transmits

$$Y^1_{\{2,3\}} = W_{2,\{1,3\},1} \oplus W_{1,\{1,2\},1}, Y^1_{\{2,4\}} = W_{2,\{1,4\},1} \oplus W_{1,\{1,2\},1}, Y^1_{\{3,4\}} = W_{1,\{1,4\},1} \oplus W_{1,\{1,3\},1}.$$

Similarly, User 3 pre-transmits

$$Y^3_{\{1,2\}} = W_{1,\{2,3\},3} \oplus W_{2,\{1,3\},3}, Y^3_{\{1,4\}} = W_{1,\{3,4\},3} \oplus W_{1,\{1,3\},3}, Y^3_{\{2,4\}} = W_{2,\{3,4\},3} \oplus W_{1,\{2,3\},3},$$

and User 4 pre-transmits

$$Y^4_{\{1,2\}} = W_{1,\{2,4\},4} \oplus W_{2,\{1,4\},4}, Y^4_{\{1,3\}} = W_{1,\{3,4\},4} \oplus W_{1,\{1,4\},4}, Y^4_{\{2,3\}} = W_{2,\{3,4\},4} \oplus W_{1,\{2,4\},4}.$$

As for User 2, $\mathcal{E}^2$ is of size $t = 2$ and is a subset of $\mathcal{K} \setminus \{2\} = \{1,3,4\}$. Hence, possible $\mathcal{E}^2$s can be $\{1,3\}, \{1,4\}$, *and* $\{3,4\}$. Recall that $\tilde{\mathcal{U}}^2 = \{1\}$; hence, only $\{1,3\}$ and $\{1,4\}$ satisfy the condition of non-zero intersection with $\mathcal{U}^2 = \{1\}$. Hence, from (A2) and (A3), User 2 pre-transmits

$$Y^2_{\{1,3\}} = W_{1,\{2,3\},2} \oplus W_{1,\{1,2\},2}, \quad Y^2_{\{1,4\}} = W_{1,\{2,4\},2} \oplus W_{1,\{1,2\},2}.$$

The signal $Y^2_{\{3,4\}} = W_{1,\{2,3\},2} \oplus W_{1,\{2,4\},2}$ does not need to be pre-transmitted and can be calculated since $Y^2_{\{1,3\}} \oplus Y^2_{\{1,4\}} \oplus Y^2_{\{3,4\}} = 0$ (cf. (24)).

In Step (c), removing unwanted codewords, we notice that the codeword $Y^1_{\{3,4\}}$ is only useful for the users, who do not request, and hence are removed. The other codewords are transmitted in the way of Step (b).

Thus, the total delivery rate achieved by the adapted YWSC scheme for the case $\mathcal{R} = \{1,2\}$ and $D'_{\{1,2,3,4\}} = (1,2,1,1)$ is $R^{\{1,2\}}_{\text{adapted-YWSC}} = \frac{2+3+3+2}{12} = \frac{5}{6}$, which coincides with (A1). In this case, if we directly use the YWSC scheme [9] by assigning Users 3 and 4 a demand that is the same as User 1, then according to [9] (Equation (14)), the delivery rate is $\frac{4\times 3-1}{12} = \frac{11}{12}$. If we assign Users 3 and 4 with some other demands, the delivery rate will be the same. However, if we use the proposed three-category-based scheme from Section 4.1, then according to (A1), the delivery rate is $R^{\{1,2\}}_{\text{req-rob}} = \frac{2}{3}$. Hence, we see that for the request-robust D2D coded caching problem, the proposed three-category-based scheme performs better than the adapted YWSC scheme for this example.

The reason why the proposed three-category-based scheme and the adapted YWSC scheme have better performance is in the following remark:

Remark A1. *The achievable rate $R^{D_{\mathcal{R}}}_{adapted\text{-}YWSC}$ of our model is similar to the achievable rate $R^*(\mathbf{d}, \mathcal{M}_{MAN})$ of Yapar's model [9]. The main difference is that the value of $N_e(\mathbf{d}_{\setminus\{i\}})$ in $R^*(\mathbf{d}, \mathcal{M}_{MAN})$ depends on all the users' request vector $\mathbf{d}$, while the value of $N_e(D'_{\mathcal{K}\setminus\{i\}})$ in*

$R^{D_{\mathcal{R}}}_{\textit{adapted-YWSC}}$ *depends on the r requesters' request vector* $D_{\mathcal{R}}$. *When* $N, K,$ *and* M *is the same, for User i, there is* $N_e(D'_{\mathcal{K}\setminus\{i\}}) \leq N_e(\mathbf{d}_{\setminus\{i\}})$, *and hence* $R^{D_{\mathcal{R}}}_{\textit{adapted-YWSC}} \leq R^*(\mathbf{d}, \mathcal{M}_{MAN})$ *for uniform demand distribution and the worst case. However, using the adapted YWSC scheme, the signal* $X_i^{\textit{adapted-YWSC}}$ *may contain useless information. For example, in Appendix A.3, the signal* $Y^1_{\{2,3\}}$ *contains the useless sub-piece* $W_{1,\{1,2\},1}$, *which is transmitted for User* 3, *who does not request. We find that removing useless sub-pieces does not affect the size of the rate but makes signals contain too small uncoded sub-pieces. The proposed three-category-based scheme in Section 4 does not contain useless information and maximizes the global gain brought from the code by utilizing the users, who do not request, to transmit signals as much as possible.*

Appendix B. Proof of Theorem 3

Appendix B.1. The Maximum Average Delivery Rate

In this subsection, we propose the converse bound given in Theorem 3 for the maximum average delivery rate of the request-robust D2D coded caching problem. Let us consider the problem of coded caching with inactive users defined in Section 2.2.2 first. For a problem of coded caching with inactive users with the rate R_{inact}, given the same placement Z and requester demand vector $D_{\mathcal{R}}$ as in Section 2.1, the encoding function on the central server is

$$\phi'^{D_{\mathcal{R}}} : [2^{NF}] \to [2^{R_{\text{inact}}F}],$$

and the delivery information X' is denoted as $X' = \phi'^{D_{\mathcal{R}}}(W_1, ..., W_N)$. The r requesters decode the request message according to caching content and delivery information, and the decoding function is defined as

$$\psi'^{D_{\mathcal{R}}}_k : [2^{M_k F}] \times [2^{R_{\text{inact}}F}] \to [2^F], \quad k \in \mathcal{R}.$$

The file decoded by User k is denoted as $\hat{W}_{d_k} = \psi'^{D_{\mathcal{R}}}_k(X', Z_k)$. If the error probability of this system satisfies

$$\mathbb{P}(\hat{W}_{d_k} \neq W_{d_k}) \leq \epsilon,$$

we call the system ϵ-*achievable*.

Given $D_{\mathcal{R}}$ and Z, we define the minimum achievable rate of the ϵ-*achievable* system as $R^{D_{\mathcal{R}}*}_{\epsilon,\text{inact}}(Z)$. For a fixed $\mathcal{R}$ with size r, $\mathcal{D}^{\mathcal{R}}$ is defined as the set of all possible demands $\{1, \cdots, N\}^r$. When $D_{\mathcal{R}}$ is uniformly distributed on $\mathcal{D}^{\mathcal{R}}$, given placement Z, the average delivery rate with respect to the uniform demand $R^{\mathcal{R}*}_{\epsilon,\text{ave,inact}}(Z)$ is defined as

$$R^{\mathcal{R}*}_{\epsilon,\text{ave,inact}}(Z) = \mathbb{E}_{D_{\mathcal{R}}}[R^{D_{\mathcal{R}}*}_{\epsilon,\text{inact}}(Z)].$$

For a given r, we define the maximum average delivery rate with respect to the uniform demand, denoted as $R^r_{\text{ave,inact}}$, where the maximum is over all the request sets $\mathcal{R}$ with size r. Then, the optimal maximum average delivery rate of the memory–rate tradeoff is essentially the maximum average delivery rate for the minimum value given an arbitrary cache size M, and at this rate, the user can decode the requested file with a sufficiently large size without error; that is,

$$R^{r*}_{\text{ave,inact}} = \sup_{\epsilon>0} \lim_{F\to+\infty} \sup \min_{Z} \max_{\mathcal{R}:|\mathcal{R}|=r} R^{\mathcal{R}*}_{\epsilon,\text{ave,inact}}(Z).$$

Similarly, for a fixed $\mathcal{R}$ with size r, given placement Z, the worst-case delivery rate is

$$R^{\mathcal{R}*}_{\epsilon,\text{worst,inact}}(Z) = \max_{D_{\mathcal{R}}} R^{D_{\mathcal{R}}*}_{\epsilon,\text{inact}}(Z),$$

and for a given r, the optimal maximum worst-case delivery rate we want to find is

$$R^{r*}_{\text{worst,inact}} = \sup_{\epsilon>0} \lim_{F\to+\infty} \sup \min_{Z} \max_{\mathcal{R}:|\mathcal{R}|=r} R^{\mathcal{R}*}_{\epsilon,\text{worst,inact}}(Z).$$

For the problem of coded caching with inactive users, we have the following results:

Theorem A1. *For the problem of coded caching with inactive users with K users, a database of N files, uncoded cache sizes of M files at each user, only r users as requesters demanding files during the delivery phase, and parameter $t = \frac{KM}{N}$, we have*

$$R^{r*}_{ave,inact} = \mathbb{E}_{D_{\mathcal{R}}}\left[\frac{\binom{K}{t+1} - \binom{K-N_e(D_{\mathcal{R}})}{t+1}}{\binom{K}{t}}\right], \tag{A4}$$

for $t \in \mathcal{K}$, where $D_{\mathcal{R}}$ is uniformly random on $\mathcal{D}^{\mathcal{R}} = \{1, ..., N\}^r$ and $N_e(D_{\mathcal{R}})$ denotes the number of distinct requests in $D_{\mathcal{R}}$. When $t \notin \mathcal{K}$, $R^{r*}_{ave,inact}$ equals the lower convex envelope of the values in (A4) for integer values of $t \in \mathcal{K}$.

Moreover, for the worst-case rate, we have

$$R^{r*}_{worst,inact} = \frac{\binom{K}{t+1} - \binom{K-\min\{r,N\}}{t+1}}{\binom{K}{t}}, \tag{A5}$$

for $t \in \mathcal{K}$. When $t \notin \mathcal{K}$, $R^{r*}_{worst,inact}$ equals the lower convex envelope of the values in (A5) for integer values of $t \in \mathcal{K}$.

Proof. The tight lower bounds of the average delivery rate and worst-case delivery rate are derived in the rest of this section. The caching and delivery scheme that achieves the optimal maximum average and worst-case rates is described in Appendix C. □

Due to inequality (4), to prove Theorem 3, we just need to prove Theorem A1. Motivated by [11], to lower bound the achievable average rate, we first introduce the concept of demand-type division. For the problem of coded caching with inactive users with the number of initial users K, the number of requesters r, and the fixed request set $\mathcal{R}$, given the request vector $D_{\mathcal{R}}$, we define $\boldsymbol{s}_{\mathcal{R}}(D_{\mathcal{R}})$ as its *statistics* such that the i^{th} element of $\boldsymbol{s}_{\mathcal{R}}(D_{\mathcal{R}})$ is equal to the number of the i-th most requested file. For example, when $K = 6$ *and* $N = 4$, and during the delivery phase $r = 4$, $\mathcal{R} = \{1, 2, 3, 4\}$, *and* $D_{\mathcal{R}} = \{1, 1, 3, 4\}$, the statistic for $D_{\mathcal{R}}$ is $\boldsymbol{s}_{\{1,2,3,4\}}(D_{\mathcal{R}}) = (2, 1, 1, 0)$. For convenience, we simply the statistics $\boldsymbol{s}_{\mathcal{R}}(D_{\mathcal{R}})$ as $\boldsymbol{s}_{\mathcal{R}}$. Then, we denote the set of all possible statistics by $\mathcal{S}_{\mathcal{R}}$. Through this statistical method, the set of request vectors $\mathcal{D}^{\mathcal{R}}$ can be divided into some subsets as *type*, and type $\mathcal{D}^{\boldsymbol{s}_{\mathcal{R}}}$ is defined as the set of queries with statistics $\boldsymbol{s}_{\mathcal{R}}$.

Note that for each request vector $D_{\mathcal{R}}$, the value of $N_e(D_{\mathcal{R}})$ only depends on its statistic $\boldsymbol{s}_{\mathcal{R}}(D_{\mathcal{R}})$, so for the same type of request vector, the rate is the same. For convenience, the $N_e(D_{\mathcal{R}})$ of the type $\mathcal{D}^{\boldsymbol{s}_{\mathcal{R}}}$ is called $N_e(\boldsymbol{s}_{\mathcal{R}})$.

Given the number of requesters r, the request set $\mathcal{R}$, and the placement Z, for each type $\mathcal{D}^{\boldsymbol{s}_{\mathcal{R}}}$ in the problem of coded caching with inactive users, the average delivery rate $R^{\boldsymbol{s}_{\mathcal{R}}*}_{\epsilon,\text{ave,inact}}(Z)$ is defined as

$$R^{\boldsymbol{s}_{\mathcal{R}}*}_{\epsilon,\text{ave,inact}}(Z) = \frac{1}{|\mathcal{D}^{\boldsymbol{s}_{\mathcal{R}}}|} \sum_{D_{\mathcal{R}} \in \mathcal{D}^{\boldsymbol{s}_{\mathcal{R}}}} R^{D_{\mathcal{R}}*}_{\epsilon,\text{inact}}(Z),$$

For all types of request vectors, we have

$$R^{r*}_{\text{ave,inact}} = \sup_{\epsilon>0} \lim_{F\to+\infty} \sup \min_{Z} \max_{\mathcal{R}:|\mathcal{R}|=r} \mathbb{E}_{\boldsymbol{s}_{\mathcal{R}}}\left[R^{\boldsymbol{s}_{\mathcal{R}}*}_{\epsilon,\text{ave,inact}}(Z)\right]$$

$$\geq \sup_{\epsilon>0} \lim_{F\to+\infty} \sup \max_{\mathcal{R}:|\mathcal{R}|=r} \mathbb{E}_{\boldsymbol{s}_{\mathcal{R}}} \left[\min_{Z} R^{\boldsymbol{s}_{\mathcal{R}}*}_{\epsilon,\text{ave,inact}}(Z) \right]. \tag{A6}$$

Hence, the lower bound of $R^{r*}_{\text{ave,inact}}$ can be derived through bounding the minimum value of $R^{\boldsymbol{s}_{\mathcal{R}}*}_{\epsilon,\text{ave,inact}}(Z)$ for each type $\mathcal{D}^{\boldsymbol{s}_{\mathcal{R}}}$ individually. We use the following lemma to lower bound the average rates within each type:

Lemma A1. *Consider the problem of coded caching with inactive users with K initial users, N files, MF cache sizes, r requesters, and request set $\mathcal{R}$ during the delivery phase; for each type $\mathcal{D}^{\boldsymbol{s}_{\mathcal{R}}}$, the minimum value of $R^{\boldsymbol{s}_{\mathcal{R}}*}_{\epsilon,ave,inact}(Z)$ is lower bounded by*

$$\min_{Z} R^{\boldsymbol{s}_{\mathcal{R}}*}_{\epsilon,ave,inact}(Z) \geq \text{Conv}\left(\frac{\binom{K}{t+1} - \binom{K-N_e(\boldsymbol{s}_{\mathcal{R}})}{t+1}}{\binom{K}{t}} \right) - \left(\frac{1}{F} + N_e^2(\boldsymbol{s}_{\mathcal{R}})\epsilon \right), \tag{A7}$$

where $\text{Conv}(f(t))$ *denotes the lower convex envelope of the following points:* $\{(t, f(t))|t \in \{0, 1, ..., K\}\}$.

Proof. We notice that solving the fundamental limits of the problem of coded caching with inactive users is essentially the caching problem defined in [11]. The only difference between the problem of coded caching with inactive users and the caching problem defined in [11] is the size of the request vector. Therefore, applying [11] (Lemma 2) and replacing the caching problem parameters demand $\boldsymbol{d}$; statistics of demand $\boldsymbol{s}$; set of all possible statistics $\mathcal{S}$; set of all possible demands $\mathcal{D}$; and type of demand $\mathcal{D}_{\boldsymbol{s}}$ with $D_{\mathcal{R}}, \boldsymbol{s}_{\mathcal{R}}(D_{\mathcal{R}}), \mathcal{S}_{\mathcal{R}}, \mathcal{D}^{\mathcal{R}}$, *and* $\mathcal{D}^{\boldsymbol{s}_{\mathcal{R}}}$, we obtain (A7). □

From (A6) and Lemma A1, the lower bound of $R^{r*}_{\text{ave,inact}}$ can be further derived as

$$R^{r*}_{\text{ave,inact}} \geq \mathbb{E}_{\boldsymbol{s}_{\mathcal{R}}} \left[\text{Conv}\left(\frac{\binom{K}{t+1} - \binom{K-N_e(\boldsymbol{s}_{\mathcal{R}})}{t+1}}{\binom{K}{t}} \right) \right]. \tag{A8}$$

Because the sequence

$$c_n = \frac{\binom{K}{n+1} - \binom{K-N_e(\boldsymbol{s}_{\mathcal{R}})}{n+1}}{\binom{K}{n}},$$

is convex, the order of the expectation and the Conv in (A8) can be switched. Therefore, $R^{r*}_{\text{ave,inact}}$ is lower bounded by the rate defined in Theorem A1, which immediately proves Theorem 3 for the maximum average delivery rate.

Appendix B.2. The Maximum Worst-Case Delivery Rate

Due to (5), to prove Theorem 3 for the maximum worst-case delivery rate, we just need to prove Theorem A1 for the maximum worst-case delivery rate. Given the number of requesters r, the request set $\mathcal{R}$, and placement Z, for each type $\mathcal{D}^{\boldsymbol{s}_{\mathcal{R}}}$ in the problem of coded caching with inactive users, we define the worst-case delivery rate $R^{\boldsymbol{s}_{\mathcal{R}}*}_{\epsilon,\text{worst,inact}}(Z)$ as

$$R^{\boldsymbol{s}_{\mathcal{R}}*}_{\epsilon,\text{worst,inact}}(Z) = \max_{D_{\mathcal{R}} \in \mathcal{D}^{\boldsymbol{s}_{\mathcal{R}}}} R^{D_{\mathcal{R}}*}_{\epsilon,\text{inact}}(Z).$$

Note that

$$\begin{aligned} R^{r*}_{\text{worst,inact}} &= \sup_{\epsilon>0} \lim_{F\to+\infty} \sup \min_{Z} \max_{\mathcal{R}:|\mathcal{R}|=r} \max_{\boldsymbol{s}_{\mathcal{R}}} R^{\boldsymbol{s}_{\mathcal{R}}*}_{\epsilon,\text{worst,inact}}(Z) \\ &\geq \sup_{\epsilon>0} \lim_{F\to+\infty} \sup \max_{\mathcal{R}:|\mathcal{R}|=r} \max_{\boldsymbol{s}_{\mathcal{R}}} \min_{Z} R^{\boldsymbol{s}_{\mathcal{R}}*}_{\epsilon,\text{worst,inact}}(Z). \end{aligned}$$

For each $\boldsymbol{s}_{\mathcal{R}} \in \mathcal{S}_{\mathcal{R}}$, using Lemma A1, we can know that

$$\begin{aligned}\min_{Z} R^{\boldsymbol{s}_{\mathcal{R}}*}_{\epsilon,\text{worst,inact}}(Z) &\geq \min_{Z} R^{\boldsymbol{s}_{\mathcal{R}}*}_{\epsilon,\text{ave,inact}}(Z) \\ &\geq \text{Conv}\left(\frac{\binom{K}{t+1} - \binom{K-N_e(\boldsymbol{s}_{\mathcal{R}})}{t+1}}{\binom{K}{t}}\right) - \left(\frac{1}{F} + N_e^2(\boldsymbol{s}_{\mathcal{R}})\epsilon\right).\end{aligned}$$

When $N, K, M, r, and\ \mathcal{R}$ do not change, since the worst-case delivery rate $R^{\boldsymbol{s}_{\mathcal{R}}*}_{\epsilon,\text{worst,inact}}(Z)$ decreases as $N_e(\boldsymbol{s}_{\mathcal{R}})$ decreases, the optimal worst-case delivery rate is the one at the maximum value of $N_e(\boldsymbol{s}_{\mathcal{R}})$. Meanwhile, there is

$$N_e(\boldsymbol{s}_{\mathcal{R}}) \leq \min\{r, N\}. \tag{A9}$$

Consequently,

$$\begin{aligned}R^{r*}_{\text{worst,inact}} &\geq \sup_{\epsilon>0} \lim_{F\to+\infty} \sup \max_{\mathcal{R}:|\mathcal{R}|=r} \max_{\boldsymbol{s}_{\mathcal{R}}} \text{Conv}\left(\frac{\binom{K}{t+1} - \binom{K-N_e(\boldsymbol{s}_{\mathcal{R}})}{t+1}}{\binom{K}{t}}\right) - \left(\frac{1}{F} + N_e^2(\boldsymbol{s}_{\mathcal{R}})\epsilon\right) \\ &= \text{Conv}\left(\frac{\binom{K}{t+1} - \binom{K-\min\{r,N\}}{t+1}}{\binom{K}{t}}\right).\end{aligned}$$

Therefore, $R^{r*}_{\text{worst,inact}}$ is lower bounded by the rate defined in Theorem A1 for the maximum worst-case delivery rate, which immediately proves Theorem 3 for the maximum worst-case delivery rate.

Appendix C. Optimal Scheme for Problem of Coded Caching with Inactive Users Achieved in Theorem A1

In this section, we present the optimal scheme for the problem of coded caching with inactive users. The scheme achieves the rate stated in Theorem A1. We will characterize the performance of the proposed scheme and show that for any requester set $\mathcal{R}$ and corresponding request vector $D_{\mathcal{R}}$, the optimal scheme achieves the rate

$$R_{\text{inact}} = \frac{\binom{K}{t+1} - \binom{K-N_e(D_{\mathcal{R}})}{t+1}}{\binom{K}{t}}. \tag{A10}$$

The rate R_{inact} with the explicit characterization of the maximum worst-case delivery rate in (A9) immediately proves Theorem A1.

Similar to Section 4.1, we restrict to integer values of $t \in \mathcal{K}$ and use the MAN uncoded symmetric placement scheme given in Definition 1 in the placement phase. For non-integer values of t, the pair (M, R_{inact}) achieves the lower convex envelope of the achievable points for integer values of $t \in \mathcal{K}$.

For the delivery phase, the central server arbitrarily selects a subset of $N_e(D_{\mathcal{R}})$ requesters, denoted by $\hat{\mathcal{U}} = \{\hat{u}_1, ..., \hat{u}_{N_e(D_{\mathcal{R}})}\}$, that requests $N_e(D_{\mathcal{R}})$ distinct files, $\hat{\mathcal{U}} \subseteq \mathcal{R}$. Following the idea of leaders from [11], we name these requesters as *leaders*.

Given an arbitrary subset $\mathcal{H}$ of $t+1$ users, each requester $k \in \mathcal{H} \cap \mathcal{R}$ needs the sub-file $W_{d_k,\mathcal{H}\backslash\{k\}}$, which is known by all other users in $\mathcal{H}$. Precisely, all the requesters $k \in \mathcal{H} \cap \mathcal{R}$ shall retrieve the needed sub-files $W_{d_k,\mathcal{H}\backslash\{k\}}$ from the transmissions of the central server. By letting the central server broadcast the codeword

$$Y_{\mathcal{H}} = \bigoplus_{x\in\mathcal{H}\cap\mathcal{R}} W_{d_x,\mathcal{H}\backslash\{x\}}, \qquad \text{where} \quad \mathcal{H}\cap\mathcal{R} \neq \emptyset, \tag{A11}$$

this sub-file retrieval can be accomplished since each requester $k \in \mathcal{H} \cap \mathcal{R}$ has all the sub-files on the RHS of (A11) except for $W_{d_k,\mathcal{H}\backslash\{k\}}$.

In order to achieve the rate in (A10), the central server only needs to transmit the codeword $Y_{\mathcal{H}}$, which is useful for at least one leader, i.e., $X^{\text{inact}} = \{Y_{\mathcal{H}}\}_{\mathcal{H}\cap\hat{\mathcal{U}}\neq\emptyset}$. The above delivery scheme totally transmits $\binom{K}{t+1} - \binom{K-N_e(D_{\mathcal{R}})}{t+1}$ codewords each with a size of $F/\binom{K}{t}$ bits, which achieves the rate in (A10).

Remark A2. *We notice that the proposed scheme is in fact the shared-link model [3] with the same file number N, file size F, user number K, cache size M, and corresponding parameter t. The only difference is the size of the request vector. Replacing the shared-link system parameter $N_e(\boldsymbol{d})$ in (19) with $N_e(D_{\mathcal{R}})$, we obtain (A10).*

We now prove that each requester can decode the file requested by the above delivery scheme. When k is a leader, i.e., $k \in \hat{\mathcal{U}}$, it can decode any required sub-file $W_{d_k,\mathcal{H}'}$, where $\mathcal{H}' \not\ni k, |\mathcal{H}'| = t$, from $Y_{\mathcal{H}'\cup\{k\}}$ by performing

$$W_{d_k,\mathcal{H}'} = \left(\bigoplus_{x\in\mathcal{H}'} W_{d_x,\mathcal{H}'\cup\{k\}\setminus\{x\}}\right) \bigoplus Y_{\mathcal{H}'\cup\{k\}}, \tag{A12}$$

where $Y_{\mathcal{H}'\cup\{k\}}$ is defined in (A11).

When $k \in \mathcal{R}\setminus\hat{\mathcal{U}}$, not all codewords $Y_{\mathcal{H}'\cup\{k\}}$ are transmitted, and the requester k needs to decode the required codewords not transmitted directly. We use Lemma A2 to explain that the non-leader requester k can also decode all the required sub-files, even if the central server does not transmit $Y_{\mathcal{H}'\cup\{k\}}$, where $\mathcal{H}'\cap\hat{\mathcal{U}} = \emptyset$.

Lemma A2. *Given the request vector $D_{\mathcal{R}}$ and picking a set of leaders $\hat{\mathcal{U}}$, for any set $\mathcal{I} \subseteq \mathcal{K}$, let $\mathcal{V}_F$ be the family of all subsets $\mathcal{V}$ of $\mathcal{I}$ such that each requested file in $D_{\mathcal{R}}$ is requested by exactly one user in $\mathcal{V}$, and we have*

$$\bigoplus_{\mathcal{V}\in\mathcal{V}_F} Y_{\mathcal{I}\setminus\mathcal{V}} = 0, \tag{A13}$$

where $Y_{\mathcal{I}\setminus\mathcal{V}}$ is defined in (A11).

Proof. As mentioned in Remark A2, the proposed scheme for the problem of coded caching with inactive users is actually the shared-link scheme. Thus, Lemma 1 in [11] can be directly applied to the proposed scheme. □

Consider any subset $\mathcal{H}$ of $t+1$ non-leader users but containing requesters. From Lemma A2, the codeword $Y_{\mathcal{H}}$ can be directly computed from the transmitted codewords by using the following equation:

$$Y_{\mathcal{H}} = \bigoplus_{\mathcal{V}\in\mathcal{V}_F\setminus\{\hat{\mathcal{U}}\}} Y_{\mathcal{I}\setminus\mathcal{V}}, \tag{A14}$$

where $\mathcal{I} = \mathcal{H}\cup\hat{\mathcal{U}}$, given the fact that all codewords on the RHS of (A14) are broadcast, because each $\mathcal{I}\setminus\mathcal{V}$ has a size of $t+1$ and contains at least one leader. Hence, each requester k can obtain the value $Y_{\mathcal{H}}$ for any subset $\mathcal{H}$, where $\mathcal{H}\cap\mathcal{R} \neq \emptyset$ of $t+1$ users, and can decode its demanded files as discussed in (A11). The proposed scheme for the problem of coded caching with inactive users achieves the rate in (A10), which proves the achievability of Theorem A1.

For the convenience of understanding, we give the algorithm of the caching and delivery scheme with inactive users in Algorithm A2.

Algorithm A2 Caching and Delivery Scheme with Inactive Users(N, K, M)

procedure PLACEMENT$(W_1, ..., W_N)$
1: Apply Algorithm 1 MAN Uncoded Symmetric Placement Scheme$(N, K, M, W_{[N]})$
end procedure

procedure DELIVERY($\mathcal{R}, D_{\mathcal{R}}$)
$t \leftarrow KM/N$
$N_e(D_{\mathcal{R}}) \leftarrow$ the number of distinct elements in $D_{\mathcal{R}}$
$\hat{\mathcal{U}} \leftarrow \{\hat{u}_1, ..., \hat{u}_{N_e(D_{\mathcal{R}})}\}$
for $\mathcal{H} \subseteq [K] : |\mathcal{H}| = t+1$ **do**
  if $\mathcal{H} \cap \hat{\mathcal{U}} == \emptyset$ **then**
    continue
  else
    Central server transmits $Y_{\mathcal{H}} = \bigoplus_{x \in \mathcal{H}: d_x \in D_{\mathcal{R}}} W_{d_x, \mathcal{H}\setminus\{x\}}$
  end if
end for
end procedure

We provide an example to explain how the proposed scheme works.

Let us consider a case that $N = 2, K = 3, M = 2/3$, *and* $t = KM/N = 1$. In the placement phase, each file is divided into $\binom{3}{1} = 3$ sub-files. The users $k \in [3]$ cache the following sub-files:

$$Z_1 = \{W_{1,\{1\}}, W_{2,\{1\}}\}, \qquad Z_2 = \{W_{1,\{2\}}, W_{2,\{2\}}\}, \qquad Z_3 = \{W_{1,\{3\}}, W_{2,\{3\}}\}.$$

In the delivery phase, only User 1 *and* 2 as requesters request a single file, i.e., $\mathcal{R} = \{1, 2\}$, and the request vector is $D_{\{1,2\}} = (1, 1)$. Notice that $r = 2$ and $N_e(D_{\{1,2\}}) = 1$.

Without a loss of generality, assume that the central server picks User 1 as the leader, i.e., $\hat{\mathcal{U}} = \{1\}$. Then, the central server transmits the following codewords:

$$Y_{\{1,2\}} = W_{1,\{2\}} \oplus W_{1,\{1\}}, \qquad Y_{\{1,3\}} = W_{1,\{3\}},$$

without transmitting $Y_{\{2,3\}} = W_{1,\{3\}}$, this will cause $Y_{\{2,3\}} \oplus Y_{\{1,3\}} = 0$ from (A13). No matter what the leader central server picks, it can transmit one less codeword, denoted as $Y_{\mathcal{H}}$, where $\mathcal{H} \cap \hat{\mathcal{U}} = \emptyset$, through this method.

From the transmitted codewords just mentioned, all the requesters can decode all their needed files by performing (A12). The rate $R_{\text{inact}} = 1/3 \times 2 = 2/3$ and could be directly calculated by (A10).

Remark A3. *The proposed scheme for the problem of coded caching with inactive users is with secure delivery [28] for $t \geq 1$ and $r \geq 2$, where the users who do not request and the external wiretapper cannot decode any files since the transmitted codewords only consist of the sub-files needed by requesters, and the sub-files whose index only consists of requesters would not be directly transmitted and cannot be decoded.*

Appendix D. Order Optimality of the Adapted YWSC Scheme in Appendix A.1 i.e., Proof of Theorem 4

As discussed in Remark 1, in this section, under the constraint of the uncoded cache placement, we only compare the achievable rate $R^{D_{\mathcal{R}}}_{\text{adapted-YWSC}}$ from (A1) with the lower bound rate for $R^{r*}_{\text{ave,req-rob}}$ and $R^{r*}_{\text{worst,req-rob}}$ from (12) and (13), respectively.

For the maximum average delivery rate, from (12), (A4), and (A10), we have that $\mathbb{E}_{D_{\mathcal{R}}}[R^{D_{\mathcal{R}}}_{\text{adapted-YWSC}}] \geq R^{r*}_{\text{ave,req-rob}} \geq R^{r*}_{\text{ave,inact}} = \mathbb{E}_{D_{\mathcal{R}}}[R_{\text{inact}}]$. Furthermore, we observe that $R_{\text{inact}} \geq \frac{t}{t+1} R^{D_{\mathcal{R}}}_{\text{adapted-YWSC}}$ by the following:

$$\frac{t}{t+1} R^{D_{\mathcal{R}}}_{\text{adapted-YWSC}} \leq \frac{\frac{1}{t+1} K \binom{K-1}{t} - \frac{1}{t+1} \sum_{i=1}^{K} \binom{K-1-N_e(D'_{\mathcal{K}\setminus\{i\}})}{t}}{\binom{K}{t}}$$

$$
\begin{aligned}
&= \frac{\binom{K}{t+1} - \sum_{i=1}^{K} \frac{1}{K-N_e(D'_{\mathcal{K}\setminus\{i\}})} \binom{K-N_e(D'_{\mathcal{K}\setminus\{i\}})}{t+1}}{\binom{K}{t}} \\
&\leq \frac{\binom{K}{t+1} - \min_i \frac{K}{K-N_e(D'_{\mathcal{K}\setminus\{i\}})} \binom{K-N_e(D'_{\mathcal{K}\setminus\{i\}})}{t+1}}{\binom{K}{t}} \\
&\leq \frac{\binom{K}{t+1} - \binom{K-N_e(D_{\mathcal{R}})}{t+1}}{\binom{K}{t}} \qquad \text{(A15)} \\
&= R_{\text{inact}},
\end{aligned}
$$

where (A15) is because $1 \leq N_e(D_{\mathcal{R}\setminus\{i\}}) \leq N_e(D'_{\mathcal{K}\setminus\{i\}}) \leq N_e(D'_{\mathcal{K}}) = N_e(D_{\mathcal{R}})$ for all $i \in [K]$.

Therefore, we have that

$$
\begin{aligned}
&\mathbb{E}_{D_{\mathcal{R}}}[R^{D_{\mathcal{R}}}_{\text{adapted-YWSC}}] \geq R^{r*}_{\text{ave,req-rob}} \geq R^{r*}_{\text{ave,inact}} = \mathbb{E}_{D_{\mathcal{R}}}[R_{\text{inact}}] \\
\geq &\frac{t}{t+1}\mathbb{E}_{D_{\mathcal{R}}}[R^{D_{\mathcal{R}}}_{\text{adapted-YWSC}}] \geq \frac{1}{2}\mathbb{E}_{D_{\mathcal{R}}}[R^{D_{\mathcal{R}}}_{\text{adapted-YWSC}}].
\end{aligned}
$$

gives the same result, which is

$$
\begin{aligned}
&\max_{D_{\mathcal{R}}} R^{D_{\mathcal{R}}}_{\text{adapted-YWSC}} \geq R^{r*}_{\text{worst,req-rob}} \geq R^{r*}_{\text{worst,inact}} = \max_{D_{\mathcal{R}}} R_{\text{inact}} \\
\geq &\frac{t}{t+1}\max_{D_{\mathcal{R}}} R^{D_{\mathcal{R}}}_{\text{adapted-YWSC}} \geq \frac{1}{2}\max_{D_{\mathcal{R}}} R^{D_{\mathcal{R}}}_{\text{adapted-YWSC}},
\end{aligned}
$$

which can be achieved for the maximum worst-case delivery rate by using similar steps. These results prove that the achievable rate $R^{D_{\mathcal{R}}}_{\text{adapted-YWSC}}$ is order optimal within a factor of two under the constraint of uncoded cache placement. In addition, by the proved order optimality of the shared-link scheme within a factor of two [29] for allowing coded placement, and as discussed in Remark A2 that the only difference between the delivery scheme with inactive users and a shared-link scheme is the size of the request vector, it immediately proves that the achieved rate is within a factor of four in general and completes the proof of Theorem 4.

References

1. Akpakwu, G.A.; Silva, B.J.; Hancke, G.P.; Abu-Mahfouz, A.M. A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges. *IEEE Access* **2018**, *6*, 3619–3647. [CrossRef]
2. Golrezaei, N.; Molisch, A.F.; Dimakis, A.G.; Caire, G. Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution. *IEEE Commun. Mag.* **2013**, *51*, 142–149. [CrossRef]
3. Maddah-Ali, M.A.; Niesen, U. Fundamental Limits of Caching. *IEEE Trans. Inf. Theory* **2014**, *60*, 2856–2867. [CrossRef]
4. Wei, Q.; Wang, L.; Xu, L.; Tian, Z.; Han, Z. Hierarchical Coded Caching for Multiscale Content Sharing in Heterogeneous Vehicular Networks. *IEEE Trans. Veh. Technol.* **2022**, *71*, 5770–5786. [CrossRef]
5. Maddah-Ali, M.A.; Niesen, U. Decentralized Coded Caching Attains Order-Optimal Memory-Rate Tradeoff. *IEEE/ACM Trans. Netw.* **2015**, *23*, 1029–1040. [CrossRef]
6. Cheng, H.; Li, C.; Xiong, H.; Frossard, P. Optimal decentralized coded caching for heterogeneous files. In Proceedings of the 2017 25th European Signal Processing Conference (EUSIPCO), Kos, Greece, 28 August–2 September 2017; pp. 2531–2535. [CrossRef]
7. Zheng, L.; Chen, Q.; Yan, Q.; Tang, X. Decentralized Coded Caching Scheme With Heterogeneous File Sizes. *IEEE Trans. Veh. Technol.* **2020**, *69*, 818–827. [CrossRef]
8. Ji, M.; Caire, G.; Molisch, A.F. Fundamental Limits of Caching in Wireless D2D Networks. *IEEE Trans. Inf. Theory* **2016**, *62*, 849–869. [CrossRef]
9. Yapar, C.; Wan, K.; Schaefer, R.F.; Caire, G. On the Optimality of D2D Coded Caching With Uncoded Cache Placement and One-Shot Delivery. *IEEE Trans. Commun.* **2019**, *67*, 8179–8192. [CrossRef]
10. Wang, W.; Liu, N.; Kang, W. Three-User D2D Coded Caching With Two Random Requesters and One Sender. *IEEE Trans. Commun.* **2023**, *71*, 6967–6978. [CrossRef]
11. Yu, Q.; Maddah-Ali, M.A.; Avestimehr, A.S. The Exact Rate-Memory Tradeoff for Caching With Uncoded Prefetching. *IEEE Trans. Inf. Theory* **2018**, *64*, 1281–1296. [CrossRef]

12. Wan, K.; Sun, H.; Ji, M.; Tuninetti, D.; Caire, G. On the Fundamental Limits of Device-to-Device Private Caching under Uncoded Cache Placement and User Collusion. *IEEE Trans. Inf. Theory* **2022**, *68*, 5701–5729. [CrossRef]
13. Wan, K.; Sun, H.; Ji, M.; Tuninetti, D.; Caire, G. Device-to-Device Private Caching with Trusted Server. In Proceedings of the ICC 2020–2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6. [CrossRef]
14. Wan, K.; Sun, H.; Ji, M.; Tuninetti, D.; Caire, G. Novel Converse for Device-to-Device Demand-Private Caching with a Trusted Server. In Proceedings of the 2020 IEEE International Symposium on Information Theory (ISIT), Los Angeles, CA, USA , 21–26 June 2020; pp. 1705–1710. [CrossRef]
15. Ibrahim, A.M.; Zewail, A.A.; Yener, A. Device-to-Device Coded-Caching With Distinct Cache Sizes. *IEEE Trans. Commun.* **2020**, *68*, 2748–2762. [CrossRef]
16. Woolsey, N.; Chen, R.R.; Ji, M. Towards Finite File Packetizations in Wireless Device-to-Device Caching Networks. *IEEE Trans. Commun.* **2020**, *68*, 5283–5298. [CrossRef]
17. Zhang, X.; Ji, M. Finite-length Analysis of D2D Coded Caching via Exploiting Asymmetry in Delivery. In Proceedings of the 2022 IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication (SPAWC), Oulu, Finland, 4–6 July 2022; pp. 1–5. [CrossRef]
18. Zewail, A.A.; Yener, A. Device-to-Device Secure Coded Caching. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 1513–1524. [CrossRef]
19. Awan, Z.H.; Sezgin, A. Fundamental limits of caching in D2D networks with secure delivery. In Proceedings of the 2015 IEEE International Conference on Communication Workshop (ICCW), London, UK, 8–12 June 2015; pp. 464–469. [CrossRef]
20. Ji, M.; Chen, R.R.; Caire, G.; Molisch, A.F. Fundamental limits of distributed caching in multihop D2D wireless networks. In Proceedings of the 2017 IEEE International Symposium on Information Theory (ISIT), Aachen, Germany, 25–30 June 2017; pp. 2950–2954. [CrossRef]
21. Lee, M.C.; Ji, M.; Molisch, A.F. Optimal Throughput-Outage Analysis of Cache-Aided Wireless Multi-Hop D2D Networks. *IEEE Trans. Commun.* **2021**, *69*, 2489–2504. [CrossRef]
22. Tebbi, A.; Sung, C.W. Coded caching in partially cooperative D2D communication networks. In Proceedings of the 2017 9th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Munich, Germany, 6–8 November 2017; pp. 148–153. [CrossRef]
23. Phatak, A.; Varanasi, M.K. An improved coded caching scheme for partially cooperative D2D networks. In Proceedings of the 2022 IEEE International Symposium on Information Theory (ISIT), Espoo, Finland, 26 June–1 July 2022; pp. 1121–1126. [CrossRef]
24. Li, J.; Chang, Y. New Constructions of D2D Placement Delivery Arrays. *IEEE Commun. Lett.* **2023**, *27*, 85–89. [CrossRef]
25. Wang, W.; Liu, N.; Kang, W. Coded Caching in Request-robust D2D Communication Networks. In Proceedings of the 2023 International Symposium on Networks, Computers and Communications (ISNCC), Doha, Qatar, 23–26 October 2023; pp. 1–6. [CrossRef]
26. Liao, J.; Tirkkonen, O. Coded Caching in Presence of User Inactivity. In Proceedings of the 2022 IEEE Wireless Communications and Networking Conference (WCNC), Austin, TX, USA, 10–13 April 2022; pp. 1437–1442. [CrossRef]
27. Wan, K.; Tuninetti, D.; Piantanida, P. On the optimality of uncoded cache placement. In Proceedings of the 2016 IEEE Information Theory Workshop (ITW), Cambridge, UK, 11–14 September 2016; pp. 161–165. [CrossRef]
28. Sengupta, A.; Tandon, R.; Clancy, T.C. Fundamental Limits of Caching With Secure Delivery. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 355–370. [CrossRef]
29. Yu, Q.; Maddah-Ali, M.A.; Avestimehr, A.S. Characterizing the Rate-Memory Tradeoff in Cache Networks Within a Factor of 2. *IEEE Trans. Inf. Theory* **2019**, *65*, 647–663. [CrossRef]

entropy

MDPI

Article

On Matrix Representation of Extension Field GF(p^L) and Its Application in Vector Linear Network Coding

Hanqi Tang, Heping Liu, Sheng Jin, Wenli Liu and Qifu Sun *

School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China; tanghanqi1009@ustb.edu.cn (H.T.); 13121778655@163.com (H.L.); 13735677295@163.com (S.J.); lilyaccount00@163.com (W.L.)
* Correspondence: qfsun@ustb.edu.cn

Abstract: For a finite field GF(p^L) with prime p and $L > 1$, one of the standard representations is $L \times L$ matrices over GF(p) so that the arithmetic of GF(p^L) can be realized by the arithmetic among these matrices over GF(p). Based on the matrix representation of GF(p^L), a conventional linear network coding scheme over GF(p^L) can be transformed to an L-dimensional vector LNC scheme over GF(p). Recently, a few real implementations of coding schemes over GF(2^L), such as the Reed–Solomon (RS) codes in the ISA-L library and the Cauchy-RS codes in the Longhair library, are built upon the classical result to achieve matrix representation, which focuses more on the structure of every individual matrix but does not shed light on the inherent correlation among matrices which corresponds to different elements. In this paper, we first generalize this classical result from over GF(2^L) to over GF(p^L) and paraphrase it from the perspective of matrices with different powers to make the inherent correlation among these matrices more transparent. Moreover, motivated by this correlation, we can devise a lookup table to pre-store the matrix representation with a smaller size than the one utilized in current implementations. In addition, this correlation also implies useful theoretical results which can be adopted to further demonstrate the advantages of binary matrix representation in vector LNC. In the following part of this paper, we focus on the study of vector LNC and investigate the applications of matrix representation related to the aspects of random and deterministic vector LNC.

Keywords: vector linear network coding; matrix representation; finite field

Citation: Tang, H.; Liu, H.; Jin, S.; Liu, W.; Sun, Q. On Matrix Representation of Extension Field GF(p^L) and Its Application in Vector Linear Network Coding. *Entropy* **2024**, *26*, 822. https://doi.org/10.3390/e26100822

Academic Editors: Boris Ryabko and Jun Chen

Received: 30 July 2024
Revised: 8 September 2024
Accepted: 24 September 2024
Published: 26 September 2024

1. Introduction

The finite fields GF(p^L) with a prime of p and an integer of $L \geq 1$ have been widely used in modern information coding, information processing, cryptography, and so on. Specifically, in the study of *linear network coding* (LNC), conventional LNC [1] transmits data symbols along the edges over GF(p^L), and every outgoing edge of a node v transmits a data symbol that is a GF(p^L)-linear combination of the incoming data symbols to v. A general LNC framework called *vector* LNC [2] models the *data unit* transmitted along every edge as an L-dimensional vector of data symbols over GF(p). Correspondingly, the coding operations at v involve GF(p)-linear combinations of all data symbols in incoming data unit vectors and are naturally represented by $L \times L$ matrices over GF(p).

Recently, many works [3–7] have shown that vector LNC has the potential to reduce extra coding overheads in networks relative to conventional LNC. In order to achieve vector LNC, a *matrix representation* of GF(p^L) [8] is $L \times L$ matrices over GF(p) so that the arithmetic of GF(p^L) can be realized by the arithmetic among these matrices over GF(p). Based on the matrix representation of GF(p^L), a conventional LNC scheme over GF(p^L) can be transformed to an L-dimensional vector LNC scheme over GF(p). In addition to the theory of LNC, many existing implementations of linear codes, such as the Cauchy-RS codes in the Longhair library [9] and the RS codes in the Jerasure library [10,11] and the latest release of the ISA-L library [12], also practically achieve arithmetic over GF(2^L) using matrix representation.

In order to achieve the matrix representation of GF(p^L), a classical result obtained in [13] relies on polynomial multiplications to describe the corresponding matrix of an element over GF(2^L). A number of current implementations and studies (see, e.g., [9–15]) utilize such a characterization to achieve the matrix representation of GF(2^L). However, the characterization in the present form focuses more on the structure of every individual matrix and does not shed light on the inherent correlation between matrices that corresponds to different elements. As a result, in the aforementioned existing implementations, the corresponding binary matrix is either independently computed on demand or fully stored in a lookup table as an $L \times L$ matrix over GF(2) in advance.

In the first part of this paper, we shall generalize the characterization of matrix representation from over GF(2^L) to over GF(p^L) and paraphrase it from the perspective of matrices with different powers so that the inherent correlation among these matrices will become more transparent. More importantly, this correlation motivates us to devise a lookup table to pre-store the matrix representation with a smaller size. Specifically, compared to the one adopted in the latest release of the ISA-L library [12], the table size is reduced by a factor of $1/L$. Additionally, this correlation also implies useful theoretical results that can be adopted to further demonstrate the advantages of binary matrix representation in vector LNC. In the second part, we focus on the study of vector LNC and show the applications of matrix representation related to the aspects of random and deterministic coding. In random coding, we theoretically analyze the coding complexity of conventional and vector LNC via matrix representation under the same alphabet size 2^L. The comparison results show that vector LNC via matrix representation can reduce at least half of the coding complexity to achieve multiplications. Then, in deterministic LNC, we focus on the special choice of coding operations that can be efficiently implemented. In particular, we illustrate that the choice of primitive polynomial can influence the distributions of matrices with different numbers of non-zero entries and propose an algorithm to obtain a set of sparse matrices that can be good candidates for the coefficients of a practical LNC scheme.

This paper is structured as follows. Section 2 reviews the mathematical fundamentals of representations to an extension field GF(p^L). Section 3 paraphrases the matrix representation from the perspective of matrices in different powers and then devises a lookup table to pre-store the matrix representation with a smaller size. Section 4 focuses on the study of vector LNC and shows the applications of matrix representation related to the aspects of random and deterministic coding. Section 5 summarizes this paper.

Notation. In this paper, every bold symbol represents a vector or a matrix. In particular, $\mathbf{I}_L$ refers to the identity matrix of size L, and $\mathbf{0}$, $\mathbf{1}$, respectively, represent an all-zero or all-one matrix, whose size, if not explicitly explained, can be inferred in the context.

2. Preliminaries

In this section, we review three different approaches to express an extension field GF(p^L) with p^L elements, where p is a prime. The first approach is the standard *polynomial representation*. Let $p(x)$ denote an irreducible polynomial of degree L over GF(p) and α be a root of $p(x)$. Every element of GF(p^L) can be uniquely expressed as a polynomial in α over GF(p) with a degree less than L, and $\{1, \alpha, \alpha^2, \ldots, \alpha^{L-1}\}$ forms a basis GF(p^L) over GF(p). In particular, every $\beta \in \mathrm{GF}(p^L)$ can be uniquely represented in the form of $\sum_{l=0}^{L-1} v_l \alpha^l$ with $v_l \in \mathrm{GF}(p)$. In the polynomial representation, the element $\beta = \sum_{l=0}^{L-1} v_l \alpha^l$ is expressed as the L-dimensional *representative vector* $\mathbf{v}_\beta = [v_0\ v_1\ \ldots\ v_{L-1}]_\beta$ over GF(p). In order to further simplify this expression, $\mathbf{v}_\beta$ can be written as the integer $0 \le d_\beta^{\mathrm{poly}} \le p^L - 1$ such that

$$d_\beta^{\mathrm{poly}} = \sum_{l=0}^{L-1} p^l \hat{v}_l, \tag{1}$$

where $0 \le \hat{v}_l < p$ is the integer representation of v_l, that is, $\sum_{i=1}^{\hat{v}_l} 1 = v_l$ where 1 is to be the multiplicative unit of GF(p).

The second approach is called the *generator representation*, which further requires $p(x)$ to be a primitive polynomial such that α is a primitive element, and all p^L-1 non-zero elements in GF(p^L) can be generated as $\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{p^L-2}$. Thus, every non-zero $\beta \in \mathrm{GF}(p^L)\backslash 0$ is uniquely expressed as the integer $0 \le d_\beta^{\mathrm{gen}} \le p^L-2$ subject to

$$\beta = \alpha^{d_\beta^{\mathrm{gen}}}. \tag{2}$$

The *polynomial representation* clearly specifies the additive structure of GF(p^L) as a vector space or a quotient ring of polynomials over GF(p) while leaving the multiplicative structure hard to determine. Meanwhile, the *generator representation* explicitly illustrates the cyclic multiplicative group structure of GF(p^L)$\backslash\{0\}$ without clearly demonstrating the additive structure. It turns out that the addition operation and its inverse in GF(p^L) are easy to implement based on the *polynomial representation*, while the multiplicative operations and its inverse in GF(p^L) are easy to be implement based on the *generator representation*. In particular, for $\beta_1, \beta_2 \in \mathrm{GF}(p^L)$,

$$d_{\beta_1+\beta_2}^{\mathrm{poly}} = d_{\beta_1}^{\mathrm{poly}} \oplus d_{\beta_2}^{\mathrm{poly}} \text{ or equivalently } \mathbf{v}_{\beta_1+\beta_2} = \mathbf{v}_{\beta_1} + \mathbf{v}_{\beta_2} \tag{3}$$

$$d_{\beta_1\beta_2}^{\mathrm{gen}} = (d_{\beta_1}^{\mathrm{gen}} + d_{\beta_2}^{\mathrm{gen}}) \mod p^L - 1, \tag{4}$$

where the operation $\oplus$ between two integers $d_{\beta_1}^{\mathrm{poly}}$ and $d_{\beta_2}^{\mathrm{poly}}$ means the component-wise p-ary addition $\mathbf{v}_1 + \mathbf{v}_2$ between the p-ary expression $\mathbf{v}_1, \mathbf{v}_2$ of them. This is the key reason that in practice both representations are always adopted interchangeably when conducting operations in GF(p^L).

Unfortunately, except for some special $\beta \in \mathrm{GF}(p^L)$, such as $\alpha^l, 0 \le l < L$, there is not a straightforward way to establish the mapping between d_β^{poly} and d_β^{gen} without computation, and a built-in lookup table is always adopted in practice to establish the mapping between two types of representations. For instance, Table 1 lists the mapping between d_β^{poly} and d_β^{gen} for non-zero elements β in GF(2^3) with $p(x) = x^4 + x + 1$.

Table 1. The mapping between d_β^{poly} and d_β^{gen} for non-zero β in GF(2^4) with $p(x) = x^4 + x + 1$.

d_β^{gen}	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
d_β^{poly}	1	2	4	8	3	6	12	11	5	10	7	14	15	13	9

By convention, elements β in GF(p^L) are represented as d_β^{poly}. It takes L p-ary additions to compute $d_{\beta_1+\beta_2} = d_{\beta_1}^{\mathrm{poly}} \oplus d_{\beta_1}^{\mathrm{poly}}$. Based on the lookup table, it takes 3 lookups (which, respectively, map $d_{\beta_1}^{\mathrm{poly}}, d_{\beta_2}^{\mathrm{poly}}$ to $d_{\beta_1}^{\mathrm{gen}}, d_{\beta_2}^{\mathrm{gen}}$ and $d_{\beta_1\beta_2}^{\mathrm{gen}}$ to $d_{\beta_1\beta_2}^{\mathrm{poly}}$), 1 integer addition, and at most 1 modulo p^L-1 operation to compute $d_{\beta_1\beta_2}^{\mathrm{poly}} = d_{\beta_1}^{\mathrm{poly}} d_{\beta_2}^{\mathrm{poly}}$. Meanwhile, it is worthwhile to note that the calculation of $d_{\beta_1\beta_2}^{\mathrm{poly}} = d_{\beta_1}^{\mathrm{poly}} d_{\beta_2}^{\mathrm{poly}}$ without the table follows the multiplication of polynomials $f_1(x)$ and $f_2(x)$ with coefficient vectors $\mathbf{v}_{\beta_1}$ and $\mathbf{v}_{\beta_2}$, respectively, and finally falls into

$$f_1(x) f_2(x) \text{ modulo } p(x), \tag{5}$$

where the computational complexity compared with the following matrix representation will be fully discussed in Section 4.

The third approach, which is the focus of this paper, is given by means of matrices called the *matrix representation* [8]. Let $\mathbf{C}$ be the $L \times L$ companion matrix of an irreducible polynomial $p(x)$ of degree L over GF(p). In particular, if $p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{L-1} x^{L-1} + x^L$ with $a_0, a_1, \ldots, a_{L-1} \in \mathrm{GF}(p)$,

$$\mathbf{C} = \begin{bmatrix} \mathbf{0} & -a_0 \\ & -a_1 \\ \mathbf{I}_{L-1} & \cdots \\ & -a_{L-1} \end{bmatrix}_{L\times L}. \tag{6}$$

It can be easily verified that $p(x)$ is the characteristic polynomial of $\mathbf{C}$, and according to the Cayley–Hamilton theorem, $p(\mathbf{C}) = \mathbf{0}$. As a result, $\{\mathbf{I}_L, \mathbf{C}, \mathbf{C}^2, \cdots, \mathbf{C}^{L-1}\}$ forms a basis of GF(p^L) over GF(p), and for every $\beta \in$ GF(p^L) with the representative vector $\mathbf{v}_\beta = [v_0\ v_1\ \ldots\ v_{L-1}]^{\mathrm{T}}$ based on the polynomial representation, the matrix representation $\mathbf{M}(\beta)$ of β is defined as

$$\mathbf{M}(\beta) = \textstyle\sum_{i=0}^{L-1} v_i \mathbf{C}^i. \tag{7}$$

If the considered $p(x)$ further qualifies as a primitive polynomial, then similar to the role of the primitive element α defined above, $\mathbf{C}$ is also a multiplicative generator of all non-zero elements in GF(p^L), that is, $\mathbf{M}(\alpha^i) = \mathbf{C}^i$ for all $0 \le i \le p^L - 2$. One advantage for the matrix representation is that all operations in GF(p^L) can be realized by matrix operations over GF(p) among the matrices in $\mathcal{C}$ such that there is no need to interchange between the polynomial and the generator representations in performing field operations. For more detailed discussions of representation of an extension field, please refer to [16].

Based on the polynomial representation and generator representation, even though the arithmetic over GF(p^L) can be efficiently realized by (3), (4) and a lookup table, it requires two different types of calculation systems, i.e., one over GF(p) and the other over integers. This hinders the deployment practicality in applications with resource-constrained edge devices, such as in ad hoc networks or Internet of Things applications. In comparison, the matrix representation of GF(p^L) interprets the arithmetic of GF(p^L) solely over the arithmetic over GF(p), so it is also a good candidate for realization of the efficient implementation of linear codes over GF(p^L) such as in [9–13].

3. Useful Characterization of the Matrix Representation

Let $p(x)$ be a defined irreducible polynomial over GF(p) of degree L and let $\alpha \in$ GF(p^L) be a root of $p(x)$. When $p = 2$, a useful characterization of the matrix representation $\mathbf{M}(\beta)$ of $\beta \in$ GF(p^L) (with respect to $p(x)$) can be deduced based on the following classical result obtained in Construction 4.1 and Lemma 4.2 of [13]: For $1 \le j \le L$, the jth column in $\mathbf{M}(\beta)$ is equal to the binary expression of $\alpha^{j-1}\beta$ based on the polynomial representation. A number of implementations and studies (see, e.g., [9–15]) of linear codes utilize such characterization to achieve the matrix representation of GF(2^L). However, the characterization in the present form relies on polynomial multiplications and focuses more on the structure of every individual $\mathbf{M}(\beta)$. It does not explicitly shed light on the inherent correlation among $\mathbf{M}(\beta)$ of different $\beta \in$ GF(2^L). It turns out that in existing implementations, such as the Cauchy-RS codes in the Longhair library [9] and the RS codes in the Jerasure library [10,11], and the latest release of ISA-L library [12], $\mathbf{M}(\beta)$ is either independently computed on demand or fully stored in a lookup table as an $L \times L$ matrix over GF(2) in advance.

In this section, we shall generalize the characterization of matrix representation from over GF(2^L) to over GF(p^L) and paraphrase it based on the interplay with the generator representation instead of the conventional polynomial representation so that the correlation among $\mathbf{M}(\beta)$ of different $\beta \in$ GF(p^L) will become more transparent. From now on, we assume that $p(x)$ is further qualified to be a primitive polynomial such that α is a primitive element in GF(p^L). For simplicity, let $\mathbf{v}_i$, $0 \le i \le p^L - 2$, denote the representative (column) vector of α^i based on the polynomial representation. Then, the following theorem asserts that the matrix representation $\mathbf{M}(\alpha^i) = \mathbf{C}^i$ consists of L representative vectors with consecutive subscripts.

Theorem 1. *For $0 \le i \le p^L - 2$, the matrix representation $\mathbf{M}(\alpha^i) = \mathbf{C}^i$ can be written as follows:*

$$\mathbf{C}^i = \begin{bmatrix} \mathbf{v}_i & \mathbf{v}_{i+1} & \cdots & \mathbf{v}_{i+L-1} \end{bmatrix}. \tag{8}$$

As $\mathbf{C}^{p^L-1} = \mathbf{I}_L$, we omit the modulo-$(p^L-1)$ expressions on the exponent of $\mathbf{C}$ and subscript of $\mathbf{v}$ throughout this paper for brevity.

Proof. First, the matrix $\mathbf{C}^i$ can be characterized by multiplication iterations based on (6) as follows. When $2 \le i \le L$,

$$\mathbf{C}^i = \begin{bmatrix} & -a_0 & p_0^{(1)} & p_0^{(2)} & \cdots & p_0^{(i-1)} \\ & -a_1 & p_1^{(1)} & p_1^{(2)} & \cdots & p_1^{(i-1)} \\ \mathbf{U} & \vdots & \vdots & \vdots & \ddots & \vdots \\ & -a_{L-2} & p_{L-2}^{(1)} & p_{L-2}^{(2)} & \cdots & p_{L-2}^{(i-1)} \\ & -a_{L-1} & p_{L-1}^{(1)} & p_{L-1}^{(2)} & \cdots & p_{L-1}^{(i-1)} \end{bmatrix}, \tag{9}$$

where $L \times (L-i)$ matrix $\mathbf{U} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_{L-i} \end{bmatrix}$. Further, when $L+1 \le i \le p^L - 2$,

$$\mathbf{C}^i = \begin{bmatrix} p_0^{(i-L)} & p_0^{(i-L+1)} & \cdots & p_0^{(i-1)} \\ p_1^{(i-L)} & p_1^{(i-L+1)} & \cdots & p_1^{(i-1)} \\ \vdots & \vdots & \ddots & \vdots \\ p_{L-2}^{(i-L)} & p_{L-2}^{(i-L+1)} & \cdots & p_{L-2}^{(i-1)} \\ p_{L-1}^{(i-L)} & p_{L-1}^{(i-L+1)} & \cdots & p_{L-1}^{(i-1)} \end{bmatrix}. \tag{10}$$

The entries in (9) and (10) iteratively qualify

$$p_0^{(1)} = -a_0 a_{L-1},\ p_j^{(1)} = a_{j-1} - a_j a_{L-1}, 1 \le j \le L-1 \tag{11}$$

and

$$\begin{aligned} p_0^{(k)} &= -a_0 p_{L-1}^{(k-1)}, \\ p_j^{(k)} &= p_{j-1}^{(k-1)} - a_j p_{L-1}^{(k-1)}, 1 \le j \le L-1, 2 \le k \le i-1. \end{aligned} \tag{12}$$

When $i = 0$, it can be easily checked that each vector in $\{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{L-1}\}$ is a unit vector such that the only non-zero entry 1 of $\mathbf{v}_i$ locates at $(i+1)$th row. Therefore, $\mathbf{C}^0 = \mathbf{I}_L = [\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{L-1}]$, and (8) holds. When $i = 1$, consider $\mathbf{v}_L$ with $p(\mathbf{C}) = \mathbf{0}$, i.e.,

$$a_0 \mathbf{I}_L + a_1 \mathbf{C} + a_2 \mathbf{C}^2 + \ldots + a_{L-1} \mathbf{C}^{L-1} + \mathbf{C}^L = \mathbf{0}. \tag{13}$$

Obviously, $\mathbf{v}_L = \begin{bmatrix} -a_0 & -a_1 & \ldots & -a_{L-1} \end{bmatrix}^{\mathrm{T}}$ and (8) holds.

Assume when $i = m$, (8) holds, i.e., $\mathbf{C}^m = \begin{bmatrix} \mathbf{v}_m & \mathbf{v}_{m+1} & \cdots & \mathbf{v}_{m+L-1} \end{bmatrix}$. The Lth column vector $\begin{bmatrix} p_0^{(m-1)} & p_1^{(m-1)} & \ldots & p_{L-1}^{(m-1)} \end{bmatrix}^{\mathrm{T}}$ of $\mathbf{C}^m$ based on (9) corresponds to the representative vector of $\mathbf{C}^{m+L-1}$, that is, the matrix $\mathbf{C}^{m+L-1}$ is equal to

$$p_0^{(m-1)} \mathbf{I}_L + p_1^{(m-1)} \mathbf{C} + \ldots + p_{L-2}^{(m-1)} \mathbf{C}^{L-2} + p_{L-1}^{(m-1)} \mathbf{C}^{L-1} \tag{14}$$

It remains to prove, by induction, that $\mathbf{C}^{m+1} = \begin{bmatrix} \mathbf{v}_{m+1} & \mathbf{v}_{m+2} & \cdots & \mathbf{v}_{m+L} \end{bmatrix}$. As the column vectors indexed from 1th to $(L-1)$th of matrix $\mathbf{C}^{m+1}$ are exactly same as the ones indexed

from 2th to Lth of $\mathbf{C}^m$, it suffices to show that the Lth column vector of $\mathbf{C}^{m+1}$ corresponds to $\mathbf{v}_{m+L}$. The following is based on (13) and (14):

$$\begin{aligned}\mathbf{C}^{m+L} &= p_0^{(m-1)}\mathbf{C} + p_1^{(m-1)}\mathbf{C}^2 + \ldots + p_{L-2}^{(m-1)}\mathbf{C}^{L-1} + p_{L-1}^{(m-1)}\mathbf{C}^L \\ &= p_0^{(m-1)}\mathbf{C} + p_1^{(m-1)}\mathbf{C}^2 + \ldots + p_{L-2}^{(m-1)}\mathbf{C}^{L-1} - p_{L-1}^{(m-1)}(a_0\mathbf{I}_L + \ldots + a_{L-1}\mathbf{C}^{L-1}) \\ &= -a_0 p_{L-1}^{(m-1)}\mathbf{I}_L + (p_0^{(m-1)} - a_1 p_{L-1}^{(m-1)})\mathbf{C} + \ldots + (p_{L-2}^{(m-1)} - a_{L-1}p_{L-1}^{(m-1)})\mathbf{C}^{L-1}.\end{aligned}$$

It can be easily checked that $p_0^{(m)}$ and $p_j^{(m)}$ with $1 \le j \le L-1$ in $\mathbf{C}^{m+1}$ calculated by (12) exactly consist of the representative vector of $\mathbf{C}^{m+L}$, i.e., $\mathbf{v}_{m+L}$. This completes the proof. □

The above theorem draws an interesting conclusion that every non-zero matrix in $\mathcal{C}$ is composed of L representative vectors. Specifically, the first column vector of the matrix representation $\mathbf{C}^i$ is the representative vector of α^i, and its jth column vector, $1 \le j \le L$, corresponds to the representative vector of α^{i+j-1}. For the case $p = 2$, even though the above theorem is essentially same as Construction 4.1 and Lemma 4.2 in [13], its expression with the interplay of generator representation allows us to further devise a lookup table to pre-store the matrix representation with a smaller size.

In this table, we store p^L representative vectors with table size $L \times p^L$ and arrange them based on the power order of α with $0 \le i \le p^L - 2$. Note that the first column of matrix $\mathbf{C}^i$ can be indexed by vector $\mathbf{v}_i$ or $(i+1)$th column in this table, and the remaining columns of $\mathbf{C}^i$ can be obtained via subsequent $L-1$ column vectors based on Theorem 1. As a result, although this table only stores p^L vectors, it contains the whole matrix representations of GF(p^L) due to the inherent correlation among $\mathbf{C}^i$. The following Example 1 shows the explicit lookup table of GF(2^4) as an example.

Example 1. *Consider the field GF(2^4) and primitive polynomial $p(x) = 1 + x + x^4$ over GF(2). The companion matrix $\mathbf{C}$ is written as follows:*

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Then, the lookup table to store matrix representation $\mathbf{C}^i$ with $0 \le i \le 14$ is shown in Figure 1. In this figure, the solid "window" that currently represents the matrix $\mathbf{C}$ can be slid to the right or left to generate $\mathbf{C}^i$ with different i; meanwhile, the dashed box shows the cyclic property based on cyclic group $\{\mathbf{I}_4 = \mathbf{C}^{15}, \mathbf{C}, \mathbf{C}^2, \cdots, \mathbf{C}^{14}\}$.

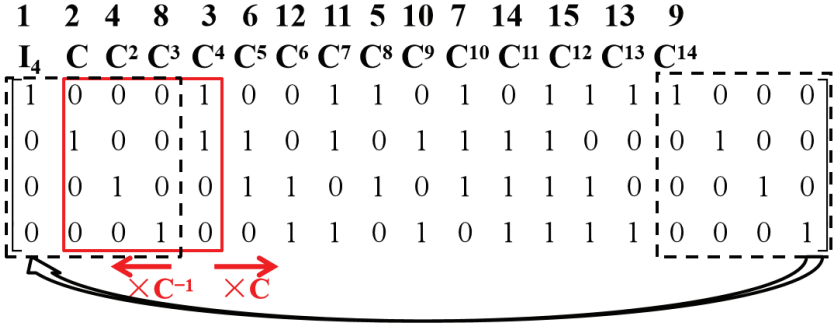


Figure 1. The lookup table to store the matrix representation $\mathbf{C}^i$ with $0 \le i \le 14$ for the field GF(2^4) and primitive polynomial $p(x) = 1 + x + x^4$.

Recall that in the lookup table of the matrix representation adopted in the latest release of the ISA-L library [12], the matrix representation of every element in GF(p^L) needs to be stored, so a total of L^2p^L p-ary elements need to be pre-stored. Compared with that, only an $L \times p^L$ p-ary matrix needs to be stored in the new lookup table, so the table size is reduced by a factor of $1/L$. Moreover, Theorem 1 implies the following useful corollaries of the matrix representation $\mathcal{C} = \{\mathbf{0}, \mathbf{C}^0, \mathbf{C}^1, \cdots, \mathbf{C}^{p^L-2}\}$ of GF(p^L).

Corollary 1. *Every vector in the vector space GF(p)^L exactly occurs L times as a column vector in matrices of $\mathcal{C}$.*

Proof. As $\{\mathbf{I}_L, \mathbf{C}, \mathbf{C}^2, \cdots, \mathbf{C}^{L-1}\}$ forms a polynomial basis of GF(p^L) over GF(p), the representative vectors of matrices in $\mathcal{C}$ are distinct. Consider a function $f : \{\mathbf{C}^i\} \to \{\mathbf{v}_i\}$. It can be easily checked that f is bijective, and $\mathbf{v}_i$ exactly corresponds to the jth column vector of $\mathbf{C}^{i-j+1}$ with $1 \leq j \leq L$. The zero vector of length L simply occurs L times in $L \times L$ matrix $\mathbf{0}$. □

Corollary 2. *For every GF(p^L), regardless of the choice of the primitive polynomial $p(x)$, the total number of zero entries in $\mathcal{C}$ remains unchanged as L^2p^{L-1}.*

The above two corollaries will be adopted to further demonstrate the advantages of binary matrix representation in vector LNC with $\mathcal{C}$.

4. Applications of Matrix Representation in Vector LNC

In this section, we focus on the study of vector LNC with binary matrices $\mathcal{C}$ and show the applications of matrix representation related to the aspects of random and deterministic coding.

4.1. Computational Complexity Comparison in Random LNC

Herein, the coding coefficients of random LNC are randomly selected from GF(2^L), which can provide a distributed and asymptotically optimal approach for information transmission, especially in unreliable or topologically unknown networks, such as wireless broadcast networks [17] or ad hoc network [18]. Recall that in polynomial and generator representations, the multiplication over GF(2^L) based on a lookup table requires two different types of calculation systems, so this table may not be utilized in resource-constrained edge devices. Therefore, under the same alphabet size 2^L, we first theoretically compare the random coding complexity between conventional LNC over $\{\beta = \sum_{l=0}^{L-1} v_l \alpha^l\}$ and vector LNC over $\mathcal{C}$ without lookup table, from the perspective of required binary operations.

To keep the same benchmark for complexity comparison, we adopt the following assumptions.

- We assume that an all-1 binary vector $\mathbf{m}$ as information will multiply $2^L - 1$ non-zero coding coefficients selected from $\{\beta = \sum_{l=0}^{L-1} v_l \alpha^l\}$ and $\mathcal{C}$, which can be simulated as encoding process. The complexity is the total number of binary operations that $2^L - 1$ multiplications take.
- We shall ignore the complexity of a shifting or permutation operation on the binary vector $\mathbf{m}$, which can be efficiently implemented.
- We only consider the standard implementation of multiplication in GF(2^L) by polynomial multiplication modulo and primitive polynomial $p(x) = a_0 + a_1 x + \cdots + a_{L-1}x^{L-1} + x^L$ with η non-zero $a_i, 0 \leq i \leq L-1$, instead of considering other advanced techniques such as the FFT algorithm [19].

We first consider the encoding scheme with coefficients selected from $\{\beta = \sum_{l=0}^{L-1} v_l \alpha^l\}$. Assume that α is a root of $p(x)$ and every element β in GF(2^L) can be expressed as $g(\alpha)$, where $g(x)$ represents a polynomial over GF(2) with a degree less than L. An all-1 binary information vector $\mathbf{m}$ can be expressed as $\alpha^{L-1} + \alpha^{L-2} + \cdots + \alpha^2 + \alpha + 1$. We can divide the whole encoding process into two parts: multiplication and addition. In the multiplication part, the complexity of shifting operations is ignored, and one polynomial $\mathbf{m}\alpha^i$ in $\mathbf{m}g(\alpha)$ will modulo $p(x)$ i times and take $i\eta$ binary operations. Because every $\alpha^i, 1 \leq i \leq L-1$ occurs 2^{L-1} times among all $g(\alpha)$ in GF(2^L), it will take $\sum_{1 \leq i \leq L-1} i\eta \times 2^{L-1}$ binary operations to compute $\mathbf{m}\alpha^i$. In the addition part, it takes $(j-1)L$ binary operations to compute the additions between j binary vectors $\mathbf{m}\alpha^i$ with distinct i. Note that the number of distinct $g(\alpha)$ with j non-zero terms is $\binom{L}{j}$ in GF(2^L). Therefore, the traverse of $g(\alpha)$ will take an extra $\sum_{1 \leq j \leq L} \binom{L}{j} \times (j-1)L$ binary additions to compute $\mathbf{m}g(\alpha)$. In total, the complexity of this scheme is shown as follows:

$$\sum_{1 \leq i \leq L-1} i(\eta \times 2^{L-1} + L \times \binom{L}{i+1}). \tag{15}$$

Next, we consider the encoding scheme with coefficients selected from $\mathcal{C}$, whose complexity of encoding process depends on the total number of 1 in $\mathbf{C}^i$ with $0 \leq i \leq 2^L - 2$. In this framework, it is worthwhile to note that every $\mathbf{C}^i$ in $\mathcal{C}$ is full-rank and can extract a permutation matrix. Since the complexity of permutational operations is ignored, based on Proposition 2, the complexity of encoding process over $\mathcal{C}$ is shown as follows:

$$L^2 \times 2^{L-1} - L \times (2^L - 1) = 2^{L-1}(L^2 - 2L) + L. \tag{16}$$

For any primitive polynomial $p(x)$, $\eta \geq 2$. With $3 \leq L \leq 12$, Table 2 lists the *average* number of binary operations *per symbol* in two schemes. Specifically, every value calculated by Equation (15) and (16) has divided the alphabet size 2^L, and we can find that in random coding, the vector LNC via matrix representation can theoretically reduce at least half of the coding complexity to achieve multiplications under the same alphabet size 2^L.

Table 2. Average number of binary operations per symbol with parameter $\eta = 2$.

L	3	4	5	6	7	8	9	10	11	12
$\mathcal{C}$	1.88	4.25	7.66	12.09	17.55	24.03	31.52	40.01	49.51	60.01
$\sum_{l=0}^{L-1} v_l \alpha^l$	4.88	10.25	17.66	27.09	38.55	52.03	67.52	85.01	104.51	126.01
rate	38.5%	41.5%	43.3%	44.6%	45.5%	46.2%	46.7%	47.1%	47.3%	47.6%

4.2. The Special Choices of Binary $p(x)$ and Sparse $\mathbf{C}^i$

In addition to the random coding, a deterministic LNC where we pay a broader concern to reduce the computational complexity can also carefully design some special coding operations which can be efficiently implemented, such as circular shift [5,6] or permutation [7]. In this subsection, different from random choice of coefficients, we will carefully design the choices of binary primitive polynomial $p(x)$ and sparse matrices $\mathbf{C}^i$ in $\mathcal{C}$ based on the unveiled properties in Sec. III. We illustrate that the choice of $p(x)$ can influence the distributions of matrices $\mathbf{C}^i$ with different numbers of non-zero entries. Then, based on a proper $p(x)$, an algorithm is proposed to obtain a subset of $\mathcal{C}$, which contains a series of relatively sparse matrices in $\mathcal{C}$.

When $p = 2$, the entries in representative vectors based on Equation (11) and (12) will, respectively, degenerate as follows:

$$\begin{aligned} p_0^{(1)} &= a_{L-1}, \\ p_j^{(1)} &= a_{j-1} + a_j a_{L-1} = a_{j-1} + a_j p_0^{(1)}, 1 \le j \le L-1. \end{aligned} \tag{17}$$

and

$$\begin{aligned} p_0^{(k)} &= p_{L-1}^{(k-1)}, \\ p_j^{(k)} &= p_{j-1}^{(k-1)} + a_j p_{L-1}^{(k-1)} = p_{j-1}^{(k-1)} + a_j p_0^{(k)}, 1 \le j \le L-1, 2 \le k \le i-1. \end{aligned} \tag{18}$$

with a_0 must be 1 in $p(x)$. Based on the above two equations, consider two adjacent representative vectors $\mathbf{v}_{k-1}$ and $\mathbf{v}_k$. When the last entry $p_{L-1}^{(k-1)}$ in $\mathbf{v}_{k-1}$ is equal to 0, the entries in $\mathbf{v}_k$ follow

$$p_0^{(k)} = 0,\ p_j^{(k)} = p_{j-1}^{(k-1)}, 1 \le j \le L-1,$$

which means that $\mathbf{v}_k$ can be generated by downward circular shift to $\mathbf{v}_{k-1}$. When the last entry $p_{L-1}^{(k-1)}$ equals 1, the entries in $\mathbf{v}_k$ follow

$$p_0^{(k)} = 1,\ p_j^{(k)} = p_{j-1}^{(k-1)} + a_j, 1 \le j \le L-1.$$

Therefore, the difference between $\mathbf{v}_{k-1}$ and $\mathbf{v}_k$ in Hamming weight is no more than $\eta - 1$, where η represents the number of non-zero $a_i, 0 \le i \le L-1$ in primitive polynomial $p(x)$.

Note that for the matrix representation of every GF(2^L), the total number of 1 in $\mathcal{C}$ is always $L^2 \times 2^{L-1}$ regardless of the choice of binary $p(x)$. However, the value of η will influence the distributions of sparse matrices in $\mathcal{C}$. Based on (17) and (18), we can intuitively deduce that with smaller η, the sparse matrices in $\mathcal{C}$ will be more concentrated distribution. Since the identity matrix $\mathbf{I}_L$ with L non-zero entries is the sparsest full-rank matrix, we utilize Algorithm 1 to choose 2^{L-s} matrices in $\mathcal{C}$, which can be good candidates as coding coefficients of a practical coding scheme over GF(2^L).

Algorithm 1 The choice of sparse matrices over GF(2^L)

Initialize S as an empty set of $L \times L$ binary matrix
  $S \leftarrow \mathbf{0}$
  $S \leftarrow \mathbf{I}_L$
  generate matrix $\mathbf{C}$ based on $p(x)$
  generate matrix $\mathbf{C}^{-1}$ based on Equation (18)
  define matrix $\hat{\mathbf{C}} = \mathbf{C}$
  define matrix $\hat{\mathbf{C}}^{-1} = \mathbf{C}^{-1}$
  define integer $s < L$: the required size of $\mathcal{C}_s$
for $i = 1 : 2^{L-s-1} - 1$
  $S \leftarrow \hat{\mathbf{C}}$
  $S \leftarrow \hat{\mathbf{C}}^{-1}$
  $\hat{\mathbf{C}} = \hat{\mathbf{C}} \times \mathbf{C}$
  $\hat{\mathbf{C}}^{-1} = \hat{\mathbf{C}}^{-1} \times \mathbf{C}^{-1}$
  $i = i + 1$
end
return S

In Algorithm 1, the multiplications using $\mathbf{C}$ or $\mathbf{C}^{-1}$ can be easily achieved by sliding the "window" right or left, respectively, as shown in Figure 1. Let $\mathcal{C}_s$ denote this subset of $\mathcal{C}$ and the 2^{L-s} matrices in $\mathcal{C}_s$ can be written as $\{\mathbf{0}, \mathbf{I}_L, \mathbf{C}^i, \mathbf{C}^{-i}\}$ with $1 \le i \le 2^{L-s-1} - 1$. Then, Table 3 lists the ratio of the total number of 1 between $\mathcal{C}_s$ and $\mathcal{C}$ with $s = 1, 2$. We can find that the 2^{L-s} matrices, which are special choices using Algorithm 1, indeed contain less 1 than the other matrices in $\mathcal{C}$.

Table 3. Ratio of total numbers of 1 between $\mathcal{C}_s$ and $\mathcal{C}$.

L	$p(x)$	η	$s=1$	$s=2$
3	X^3+X+1	2	0.3056	0.0833
4	X^4+X+1	2	0.3438	0.1094
5	X^5+X^2+1	2	0.3800	0.1200
6	X^6+X+1	2	0.4410	0.1372
7	X^7+X+1	2	0.4585	0.1987
8	$X^8+X^4+X^3+X^2+1$	4	0.4635	0.2235
9	X^9+X^4+1	2	0.4777	0.2148
10	$X^{10}+X^3+1$	2	0.4950	0.2382
11	$X^{11}+X^2+1$	2	0.4898	0.2325
12	$X^{12}+X^6+X^4+X+1$	4	0.4977	0.2421
13	$X^{13}+X^4+X^3+X+1$	4	0.4906	0.2469
14	$X^{14}+X^5+X^3+X+1$	4	0.4975	0.2500
15	$X^{15}+X+1$	2	0.4979	0.2462
16	$X^{16}+X^5+X^3+X^2+1$	4	0.4978	0.2490

Moreover, in Figure 2, we numerically analyze the relationship between the number of 1 in each matrix and the corresponding number of matrices under the alphabet size 2^{12}. For all candidates of binary primitive polynomials, we choose four representative $p(x)$ with different $\eta = 4, 6, 8, 10$ and the specific polynomials are shown as follows:

$$
\begin{aligned}
p_1(x) &= x^{12}+x^6+x^4+x^1+1\\
p_2(x) &= x^{12}+x^7+x^6+x^5+x^3+x^1+1\\
p_3(x) &= x^{12}+x^8+x^7+x^6+x^4+x^3+x^2+x^1+1\\
p_4(x) &= x^{12}+x^{10}+x^9+x^8+x^7+x^5+x^4+x^3+x^2+x^1+1
\end{aligned}
$$

As all the matrices in $\mathcal{C}$ are full-rank, the value range of the x-axis should be $[12, 132]$, and we restrict it to $[40, 100]$ to highlight the distributions. These four curves illustrate that with η increasing, the distribution variance of the number of 1 in a matrix will decrease, that is, the number of matrices with an average number of 1, i.e., 70–80, will increase and the number of relatively sparse or dense matrices will decrease. As a result, a smaller η of $p(x)$ not only infers a more concentrated distribution but also more amounts for sparse matrices in $\mathcal{C}$; then, we can select parameter s according to practical requirements and obtain $\mathcal{C}_s$ using Algorithm 1.

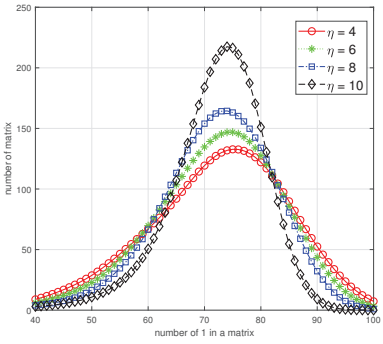


Figure 2. The distribution of sparse matrices in $\mathcal{C}$ with different $\eta = 4, 6, 8, 10$.

5. Conclusions

Compared with the classical result, the paraphrase of matrix representation in this paper focuses more on inherent correlation among matrices and a lookup table to prestore the matrix representation with a smaller size is devised. This work also identifies that the total number of non-zero entries in $\mathcal{C}$ is a constant number, which motivates us to demonstrate the advantages of binary matrix representation in vector LNC. In the applications of matrix representation, we first theoretically demonstrate the vector LNC via matrix representation can reduce at least half of the coding complexity compared with conventional one over GF(2^L). Then, we illustrate the influence of η, i.e., the number of non-zero item in $p(x)$, on the amounts and distributions of sparse matrices in $\mathcal{C}$ and propose an algorithm to obtain sparse matrices which can be good candidates as coding coefficients of a practical vector LNC scheme.

Author Contributions: Methodology, H.T.; Software, S.J.; Writing—original draft, H.T.; Writing—review & editing, H.L.; Visualization, W.L.; Supervision, Q.S.; Funding acquisition, Q.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the National Natural Science Foundation of China under Grants U22A2005, 62101028 and 62271044, and by the Fundamental Research Funds for the Central Universities under Grant FRF-TP-22-041A1.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Li, S.-Y.R.; Yeung, R.W.; Cai, N. Linear network coding. *IEEE Trans. Inf. Theory* **2003**, *49*, 371–381. [CrossRef]
2. Ebrahimi, J.B.; Fragouli, C. Algebraic algorithm for vecor network coding. *IEEE Trans. Inf. Theory* **2011**, *57*, 996–1007. [CrossRef]
3. Sun, Q.T.; Yang, X.; Long, K.; Yin, X.; Li, Z. On vector linear solvability of multicast networks. *IEEE Trans. Commun.* **2016**, *64*, 5096–5107. [CrossRef]
4. Etzion, T.; Wachter-Zeh, A. Vector network coding based on subspace codes outperforms scalar linear network coding. *IEEE Trans. Inf. Theory* **2018**, *64*, 2460–2473. [CrossRef]
5. Tang, H.; Sun, Q.T.; Li, Z.; Yang, X.; Long, K. Circular-shift linear network coding. *IEEE Trans. Inf. Theory* **2019**, *65*, 65–80. [CrossRef]
6. Sun, Q.T.; Tang, H.; Li, Z.; Yang, X.; Long, K. Circular-shift linear network codes with arbitrary odd block lengths. *IEEE Trans. Commun.* **2019**, *67*, 2660–2672. [CrossRef]
7. Tang, H.; Zhai, Z.; Sun, Q.T.; Yang, X. The multicast solvability of permutation linear network coding. *IEEE Commun. Lett.* **2023**, *27*, 105–109. [CrossRef]
8. Wardlaw, W.P. Matrix representation of finite field. *Math. Mag.* **1994**, *67*, 289–293. [CrossRef]
9. Longhair: $O(N^2)$ Cauchy Reed-Solomon Block Erasure Code for Small Data. 2021. Available online: https://github.com/catid/longhair (accessed on 1 July 2024).
10. Plank, J.S.; Simmerman, S.; Schuman, C.D. Jerasure: A Library in c/c++ Facilitating Erasure Coding for Storage Applications, Version 1.2; Technical Report CS-08-627; University of Tennessee: Knoxville, TN, USA, 2008.
11. Luo, J.; Shrestha, M.; Xu, L.; Plank, J.S. Efficient encoding schedules for XOR-based erasure codes. *IEEE Trans. Comput.* **2014**, *63*, 2259–2272. [CrossRef]
12. Intel® Intelligent Storage Acceleration Library. 2024. Available online: https://github.com/intel/isa-l/tree/master/erasurecode (accessed on 1 June 2024).
13. Blomer, J.; Kalfane, M.; Karp, R.; Karpinski, M.; Luby, M.; Zuckerman, D. *An XOR-Based Erasure-Resilient Coding Scheme*; Technical Report TR-95-048; University of California at Berkeley: Berkeley, CA, USA, 1995.
14. Plank, J.S.; Xu, L. Optimizing Cauchy Reed-Solomon codes for fault-tolerant network storage applications. In Proceedings of the Fifth IEEE International Symposium on Network Computing and Applications, Cambridge, MA, USA, 24–26 July 2006; pp. 173–180.
15. Zhou, T.; Tian, C. Fast erasure coding for data storage: A comprehensive study of the acceleration techniques. *ACM Trans. Storage (TOS)* **2020**, *16*, 1–24. [CrossRef]
16. Lidl, R.; Niederreiter, H. *Finite Fields*, 2nd ed.; Cambridge University Press: Cambridge, UK, 1997.

17. Su, R.; Sun, Q.T.; Zhang, Z. Delay-complexity trade-off of random linear network coding in wireless broadcast. *IEEE Trans. Commun.* **2020**, *68*, 5606–5618. [CrossRef]
18. Asterjadhi, A.; Fasolo, E.; Rossi, M.; Widmer, J.; Zorzi, M. Toward network coding-based protocols for data broadcasting in wireless ad hoc networks. *IEEE Trans. Wirel. Commun.* **2010**, *9*, 662–673. [CrossRef]
19. Gao, S.; Mateer, T. Additive fast Fourier transforms over finite fields. *IEEE Trans. Inf. Theory* **2010**, *56*, 6265–6272. [CrossRef]

entropy

MDPI

Article

Finite-Blocklength Analysis of Coded Modulation with Retransmission

Ming Jiang [1,2,*], Yi Wang [1], Fan Ding [1] and Qiushi Xu [1]

[1] National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China; wang_yi@seu.edu.cn (Y.W.); fan_d@seu.edu.cn (F.D.); xuqiushi@seu.edu.cn (Q.X.)
[2] Purple Mountain Laboratories, Nanjing 211100, China
* Correspondence: jiang_ming@seu.edu.cn; Tel.: +86-139-1290-9162

Abstract: The rapid developments of 5G and B5G networks have posed higher demands on retransmission in certain scenarios. This article reviews classical finite-length coding performance prediction formulas and proposes rate prediction formulas for coded modulation retransmission scenarios. Specifically, we demonstrate that a recently proposed model for correcting these prediction formulas also exhibits high accuracy in coded modulation retransmissions. To enhance the generality of this model, we introduce a range variable $\mathbf{P_{final}}$ to unify the predictions with different SNRs. Finally, based on simulation results, the article puts forth recommendations specific to retransmission with a high spectral efficiency.

Keywords: finite blocklength; retransmission scenario; rate prediction

Citation: Jiang, M.; Wang, Y.; Ding, F.; Xu, Q. Finite-Blocklength Analysis of Coded Modulation with Retransmission. *Entropy* **2024**, *26*, 863. https://doi.org/10.3390/e26100863

Academic Editors: Shenghao Yang and Kenneth Shum

Received: 14 September 2024
Revised: 8 October 2024
Accepted: 11 October 2024
Published: 14 October 2024

1. Introduction

New-generation mobile communication systems, 5G NR networks, are worldwide-deployed communication systems. The 5G wireless system, which is not the straightforward evolution of traditional 4G cellular networks, is developed as a multipurpose mobile network with many new service functionalities [1]. 5G networks can provide not only traditional voice and data communication but also numerous new use cases, applications for various industries, and connectivity for devices and applications across society [2,3]. Examples include vehicle-to-vehicle and vehicle-to-infrastructure communication, industrial automation, health services, smart cities, and smart homes [4,5]. Compared to 4G LTE, 5G NR and the future development of B5G systems have introduced a series of technical indicators. To meet these performance requirements, 5G systems will leverage various emerging technologies, such as heterogeneous networks (HetNets) [6,7], massive multiple-input multiple-output (mMIMO) [8], millimeter-wave (mmWave) communication [9,10], device-to-device (D2D) communication [11,12], machine-to-machine (M2M) communication [13], reconfigurable intelligent surfaces (RISs) [14], and network slicing [15], among others.

D2D and M2M communications have many different characteristics compared to the traditional communication services designed for human interaction. For instance, the communication among many sensors and controllers in closed-loop control systems of automated industries requires a maximum latency of 5 ms and a reliable packet error rate ranging from 10^{-2} to 10^{-5} [16]. In terms of traffic safety, the packet error rate cannot exceed 10^{-5}. These typical applications involve short data packets (code length ranges from several hundred to one thousand) and impose very high requirements on latency and reliability. For the applications targeting these machine communications, various solutions have been proposed, including fewer symbols in OFDM signal packets, reducing transmission time. The theoretical limit for the transmission of these short data packets depends on the specific transmission environments and the technologies employed.

Shannon's limit provides the theoretical maximum performance when the encoding blocklength tends towards infinity. However, in practical situations, Shannon's limit

does not apply to the performance of moderate-length codes [17]. The finite-blocklength performance bounds in binary input additive white Gaussian noise (BIAWGN) channels have adequately addressed this issue. Recently, numerous significant advancements related to finite-blocklength analysis have emerged. Ref. [18] leverages the property of joint convexity to address a broad spectrum of use cases, thereby facilitating the efficient resolution of joint optimization problems in multi-user environments in the finite-blocklength regime. In [19], Behrooz Makki derives closed-form expressions for message-decoding probabilities, throughput, expected delay, and error probability in hybrid automatic repeat request (HARQ) configurations. Moreover, the expectation and variance of the maximum achievable rate in a mMIMO system with a finite blocklength are rigorously analyzed [20]. However, the finite-blocklength performance analysis is not suitable for higher-order modulation schemes, for which effective solutions have been proposed in the existing literature [21–23]. Moreover, to ensure transmission reliability while also meeting low-latency constraints, a limited number of retransmissions is typically required in practical wireless networks. Combining rate-compatible coding and incremental redundancy retransmission schemes, the performance analysis of finite-length coded retransmission with high-order modulation is an urgent issue that needs to be addressed.

In this paper, we analyze the performance of finite-length coded modulation in a retransmission scenario when rate-compatible code is modulated and transmitted using different modulation schemes in the first and second transmissions. The remaining structure of this paper is represented as follows. Firstly, we review the theoretical formulas for predicting the performance of finite-length coded modulation and provide a brief explanation of the calculation of key parameters in this formula under retransmission scenarios. Next, we revisit the model for tuning on the theoretical prediction formula and elaborate on the usage of the model. Here, we refine this method to make it more general. Finally, through simulation results, we demonstrate the good adaptability of this calibration model to retransmission scenarios. Based on the simulation results, we also offer some recommendations for the retransmission approach of coded modulation.

2. Preliminaries

2.1. Some Bounds for Finite-Blocklength Coding

Consider a code with a codebook of size M and blocklength n, where the rate R can be expressed as

$$R = \frac{\log_2 M}{n}. \tag{1}$$

Building upon this, [24] has proposed formulas for the upper and lower bounds on the performance of finite-length coding. For example, the upper bound like the converse bound, the lower bound like the random coding union (RCU) bound and the dependence testing (DT) bound. However, these bounds all involve a greater amount of summation and combinatorial operations, leading to a higher overall complexity and potentially imprecise results.

For a binary symmetric channel (BSC) with a crossover probability of δ, when it achieves the block error rate (BLER) ϵ, its RCU bound and DT bound can be calculated by

$$\epsilon \leq \sum_{t=0}^{n} \binom{n}{t} \delta^t (1-\delta)^{n-t} \min\{1, (M-1) \sum_{k=0}^{t} \binom{n}{k} 2^{-n}\}, \tag{2}$$

and

$$\epsilon \leq \sum_{t=0}^{n} \binom{n}{t} \min\{\delta^t (1-\delta)^{n-t}, (M-1)2^{-n-1}\}, \tag{3}$$

respectively. In practical coding, the blocklength n usually amounts to several hundred, or even greater than 1000. When the combinations of $\binom{n}{k}$ are calculated, the computation process becomes slower, and the precision of the results is certainly affected.

The converse bound of a BSC satisfies

$$M \leq \frac{1}{\beta_{1-\epsilon}^{n}}, \tag{4}$$

$$\beta_{\alpha}^{n} = (1-\lambda)\beta_{L} + \lambda\beta_{L+1}, \tag{5}$$

where the β_l in (5) is defined as

$$\beta_{l} = \sum_{k=0}^{l} \binom{n}{k} 2^{-n}, l = L, L+1, \tag{6}$$

and the integer L and variable λ $(0 \leq \lambda \leq 1)$ should be determined by the following equation:

$$\alpha = (1-\lambda)\alpha_{L} + \lambda\alpha_{L+1}, \tag{7}$$

with

$$\alpha_{l} = \sum_{k=0}^{l-1} \binom{n}{k} (1-\delta)^{n-k}\delta^{k}, l = L, L+1. \tag{8}$$

The calculation of the converse bound (4) not only involves the combinatorial operations but also requires solving roots for two parameters in a system of binary equations, making the computation quite complicated. Therefore, a simpler and more efficient calculation method is further explored by normal approximation (NA).

2.2. Normal Approximation Combined with Coded Modulation

Given a finite blocklength n, BLER ϵ, the upper bound of the rate can be predicted by

$$R = C - \sqrt{\frac{V}{n}}Q^{-1}(\epsilon) + \mathcal{O}\left(\frac{\log_2 n}{n}\right), \tag{9}$$

which is called normal approximation [24], where $Q(x) = \int_{x}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2} dt$, C and V are the channel capacity and the channel dispersion, respectively. They are both characteristic parameters of the channel, where the physical quantities do not depend on the encoding scheme. The third-order term $\mathcal{O}\left(\frac{\log_2 n}{n}\right)$ is proven to be $\frac{\log_2 n}{2n}$ in [24].

In different channels, C and V have different calculation methods. For a BSC with a crossover probability of δ and $\delta \notin \{0, \frac{1}{2}, 1\}$, we have

$$C = 1 - h(\delta) \tag{10}$$

$$V = \delta(1-\delta)(\log_2 \frac{1-\delta}{\delta})^2, \tag{11}$$

where $h(x) = -x\log_2 x - (1-x)\log_2(1-x)$.

Meanwhile, for a binary erasure channel (BEC) with an erasure probability of δ, we have

$$C = 1 - \delta \tag{12}$$

$$V = \delta(1-\delta). \tag{13}$$

Here, for more general applications, taking a BIAWGN channel with an SNR of P into consideration, we have

$$C = \frac{1}{2}\log_2(1+P) \tag{14}$$

$$V = \frac{P}{2}\frac{P+2}{(P+1)^2}\log_2^2 e. \tag{15}$$

Although (14) and (15) can be easily calculated, specific modulation methods do not provide the correlation between C, V and constellations.

If the input m points of a constellation, such as m-QAM, follow the discrete uniform distribution, the two parameters C and V can be computed [25] by

$$C_m(P) = \log_2 m - \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\left[\log_2\left(\sum_{j=1}^{m} e^{\|Z\|^2-\|x(i)+Z-x(j)\|^2}\right)\right] \tag{16}$$

$$V_m(P) = \frac{1}{m}\sum_{i=1}^{m}\mathrm{Var}\left[\log_2\left(\sum_{\mathrm{j}=1}^{\mathrm{m}} \mathrm{e}^{\|\mathrm{Z}\|^2-\|\mathrm{x(i)+Z-x(j)}\|^2}\right)\right] \tag{17}$$

where Z is a complex Gaussian variable with a zero mean and unit variance, and $x(i)$ corresponds to a normalized constellation point of m-QAM with a given SNR of P. $\mathbb{E}[\cdot]$ and $\mathrm{Var}[\cdot]$ represent the calculations of the mean and variance, respectively. When the value of m is quite large, the calculations of (16) and (17) suffer a noticeable increase in complexity, but do provide the correlation between C, V and the constellation.

3. Practical Application with Retransmission

In this section, we consider the coded modulation retransmission scenario in incremental redundancy (IR) HARQ and the calculations of key parameters with the theoretical formula and a calibration model proposed by Eva C. Song and Guosen Yue [26], which are easy to use and have extremely good accuracy.

When the first segment of a rate-compatible coding scheme with a high-rate code of length n fails to be received, the transmitter then sends the redundancy version of coded bits with identical length n to the receiving end, resulting in a half-rate code of length $2n$ for decoding. During retransmission, the modulation order is usually lowered according to the specific modulation and coding scheme (MCS), such as the MCS table in 5G NR, thereby better handling errors and enhancing the robustness of transmission.

In [26], the calculations of C and V for the parallel complex Gaussian channels with an m-QAM input are provided by (15) and (16), respectively. Similarly, we can consider the coded modulation in the retransmission scenario as the receiver simultaneously receiving two equal-length coded blocks from a rate-compatible coding scheme with different modulations m_1-QAM and m_2-QAM over the same channel.

Therefore, in this scenario, C and V in (9) are computed by

$$C = \frac{1}{2}(C_{m1}(P) + C_{m2}(P)) \tag{18}$$

$$V = \frac{1}{2}(V_{m1}(P) + V_{m2}(P)), \tag{19}$$

where $C_{m_i}(P)$ and $V_{m_i}(P), i = 1, 2$ can get by (16) and (17) on the constellations of m_1-QAM and m_2-QAM, respectively. The proof of C and V is provided in Appendix A.

In terms of practical coding, ref. [26] proposes the following models:

$$R(P, n, \epsilon) = C(P) - \Delta C(P) - \alpha(P)\sqrt{\frac{V(P)}{n}}Q^{-1}(\epsilon) + \frac{\log_2(n)}{2n}, \tag{20}$$

where $C(P)$ and $V(P)$ can get by (18) and (19), respectively. $\Delta C(P)$ refers to the gap between the theoretical capacity and the rate that practical coding can achieve when the blocklength is finite. $\alpha(P) \geq 1$ is the correction parameter for the channel dispersion V.

We follow the flowchart shown in Figure 1 to calculate the parameters in (20). Firstly, select a targeting BLER ϵ and a sufficiently long blocklength n_{inf} as an approximation for infinite blocklength, where a practical rate-compatible coding scheme is employed for the necessary initial simulation, such as LTE-turbo codes and 5G-LDPC codes. Then, for each specific rate $R_i^{n_{\mathrm{inf}}}, i = 1, 2, 3, \ldots, t_1$, obtain the $P_i^{n_{\mathrm{inf}}}, i = 1, 2, 3, \ldots, t_1$ required to achieve

the BLER ϵ based on simulation. Next, select several short blocklengths $n_1, n_2, \ldots, n_s$ for tuning. For each $n_k, k = 1, \ldots, s$ and specific rate $R_j^{n_k}, j = 1, 2, 3, \ldots, t_2$, obtain the $P_j^{n_k}, j = 1, 2, 3, \ldots, t_2$ required to achieve the BLER ϵ based on simulation.

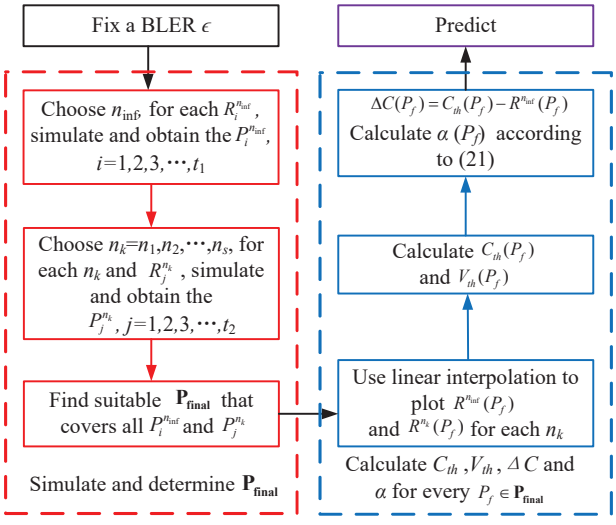


Figure 1. Flowchart of calculation algorithm.

We simulate to obtain the $P_i^{n_{\inf}}$ and $P_j^{n_k}$ variables using the following method: Given the modulation scheme, code blocklength $n, n = n_{\inf}$ or n_k, and rate R, we vary the values of SNR to obtain a set of data for different SNRs and BLERs $(P_m^n, \epsilon_m), m = 1, 2, 3, \ldots$. Then, around the given BLER ϵ, we identify two different BLERs which are the nearest neighbors $\epsilon_{m_1} > \epsilon > \epsilon_{m_2}$ and perform linear interpolation based on their corresponding SNRs $P_{m_1}^n$ and $P_{m_2}^n$ to obtain the SNR P^n corresponding to the desired BLER ϵ. The linear interpolation formula is as follows.

$$\epsilon = \frac{\epsilon_{m_2} - \epsilon_{m_1}}{P_{m_2}^n - P_{m_1}^n}(P^n - P_{m_1}^n) + \epsilon_{m_1}. \tag{21}$$

Let the ϵ be the desired BLER; then, we can get the SNR P^n by (21).

Next, based on the $P_i^{n_{\inf}}$ and $P_j^{n_k}$ obtained from above, determine a smallest range (or slightly larger) of $\mathbf{P}_{\text{final}}$ to cover all the $P_i^{n_{\inf}}$ and $P_j^{n_k}$. For example, if we simulate to obtain $P_i^{n_{\inf}} = 1, 1.5, \ldots, 2.5$ (dB) and $P_j^{n_k} = 1.2, 1.7, 2.2, \ldots, 2.9$ (dB), then we can choose $\mathbf{P}_{\text{final}} = [1, 2.9]$ (dB). After that, use linear interpolation to connect all the $R_i^{n_{\inf}}(P_i^{n_{\inf}})$ and $R_j^{n_k}(P_j^{n_k})$ to get $R^{n_{\inf}}(P_f)$ and $R^{n_k}(P_f)$ in the range $P_f \in \mathbf{P}_{\text{final}}$. Then, calculate the theoretical channel capacity $C_{th}(P_f)$ and the theoretical channel dispersion $V_{th}(P_f)$ according to (18) and (19), respectively. Next, calculate $\Delta C(P_f) = C_{th}(P_f) - R^{n_{\inf}}(P_f)$ and for every $P_f \in \mathbf{P}_{\text{final}}$, find α that minimizes (22) to get $\alpha(P_f)$.

$$\alpha(P_f) = \arg\min_{\alpha} \sum_{k=1}^{s} (C_{th}(P_f) - \Delta C(P_f) - \alpha\sqrt{\frac{V_{th}(P_f)}{n_k}}Q^{-1}(\epsilon) + \frac{\log_2 n_k}{2n_k} - R^{n_k}(P_f))^2 \tag{22}$$

Finally, for any given code length n and $P_f \in \mathbf{P}_{\text{final}}$, compute C and V according to (18) and (19), and obtain ΔC and α from the steps above. Predict R by using (20), which is shown in Algorithm 1.

Algorithm 1: Calculation algorithm of the model to predict R

Input : ϵ, n_{inf}, $R_i^{n_{\text{inf}}}$, $n_1, n_2, \ldots, n_s$, $R_j^{n_k}$
Output: R

Fix a BLER ϵ
Simulate to get $P_i^{n_{\text{inf}}}$ based on n_{inf}, ϵ, $R_i^{n_{\text{inf}}}, i = i = 1, 2, 3, \ldots, t_1$
Simulate to get $P_j^{n_k}$ based on $n_k = n_1, n_2, \ldots, n_s$, ϵ, $R_j^{n_k}, j = 1, 2, 3, \ldots, t_2$
Choose a range $\mathbf{P_{final}}$ that covers all the $P_i^{n_{\text{inf}}}$ and $P_j^{n_k}$
Use linear interpolation to connect all the $R_i^{n_{\text{inf}}}(P_i^{n_{\text{inf}}})$ and $R_j^{n_k}(P_j^{n_k})$ to get $R^{n_{\text{inf}}}(P_f)$ and $R^{n_k}(P_f)$ in the range $P_f \in \mathbf{P_{final}}$
Calculate $C_{th}(P_f)$, $V_{th}(P_f)$ for $P_f \in \mathbf{P_{final}}$
$\Delta C(P_f) = C_{th}(P_f) - R^{n_{\text{inf}}}(P_f)$ for $P_f \in \mathbf{P_{final}}$
For every $P_f \in \mathbf{P_{final}}$, calculate α that minimizes (22) to get $\alpha(P_f)$
return

$$R = C(P_f) - \Delta C(P_f) - \alpha(P_f)\sqrt{\frac{V(P_f)}{n}}Q^{-1}(\epsilon) + \frac{\log_2(n)}{2n}$$

for every $P_f \in \mathbf{P_{final}}$

The above method incorporates some modifications to the method proposed in [26]. When using (22), the SNR required for calculating each α is the same, but the simulated SNR often varies for different selected n_k and $R_j^{n_k}$. Therefore, after obtaining the simulation data points, we select a range $\mathbf{P_{final}}$ to unify the different SNRs obtained from the simulation that required in the formula.

4. Numerical Example

In this section, we demonstrate that the proposed model is also applicable to the scenario of retransmission and we analyze the results with different coded modulation combinations. In the following examples, we always use the rate-compatible coding scheme based on 5G-LDPC codes and BP decoding in the transmitter and receiver. Assume that 16-QAM and QPSK are used in two transmissions, respectively, where the coded bits in the first half and the second half of each encoding segment are modulated by 16-QAM and QPSK, respectively.

In our simulations, the rate R is computed by

$$R = R_c \times \frac{1}{2}(\log_2(m_1) + \log_2(m_2)), \tag{23}$$

where R_c is the original code rate, $\log_2(m_1)$ and $\log_2(m_2)$ refer to the modulation orders for the two segments. In this example, an LDPC code with a code rate of $R_c = \frac{1}{3}$, 16-QAM$_{1st}$ $\log_2(m_1) = 4$, and QPSK$_{2nd}$ $\log_2(m_2) = 2$ are employed in the two transmissions; thus, the rate here is $R = 1$.

Since the number of message bits remains the same after retransmission, the code length becomes twice as long, and the highest code rate of 5G-LDPC codes is $\frac{11}{12}$ in the first transmission. Then, after retransmission with $R_c = \frac{11}{24}$, the highest rate here is $R = \frac{11}{24} \times 3 = \frac{11}{8}$.

We select $n_{\text{inf}} = 7200$ as an approximation for infinite code length, which approaches the maximum length 8448 of information bits in the 5G-LDPC coding scheme, with $n_1 = \frac{1}{20}n_{\text{inf}} = 360$, $n_2 = \frac{3}{20}n_{\text{inf}} = 1080$ for tuning. We choose the code rates like $R_c = \frac{1}{3}, \frac{7}{20}, \frac{11}{30}, \frac{5}{12}, \frac{9}{20}$ and $\frac{11}{24}$ to make the code with length 360 have an integer number of information bits. Then, we predict the retransmission performance of the coded modulation with $n = 3600$.

With this example, let us go through the steps outlined in Algorithm 1. Choose a fixed BLER $\epsilon = 0.1$ and then simulate to obtain Figures 2 and 3a,b. In this example, $\mathbf{P_{final}} = [1.4, 4.65]$ (dB) can cover all the simulation points $P_i^{n_{\inf}}$ and $P_j^{n_k}$. Then, Tables 1 and 2 calculate ΔC and α, respectively. Finally, using (20) and the previously obtained parameters, we can predict R. By repeating the steps mentioned above for $\epsilon = 10^{-2}$ and 10^{-3}, we can get the results shown in Figure 4. As shown in Figure 4, the prediction performance of this model is also very good in the retransmission scenario with a moderate blocklength and different modulations.

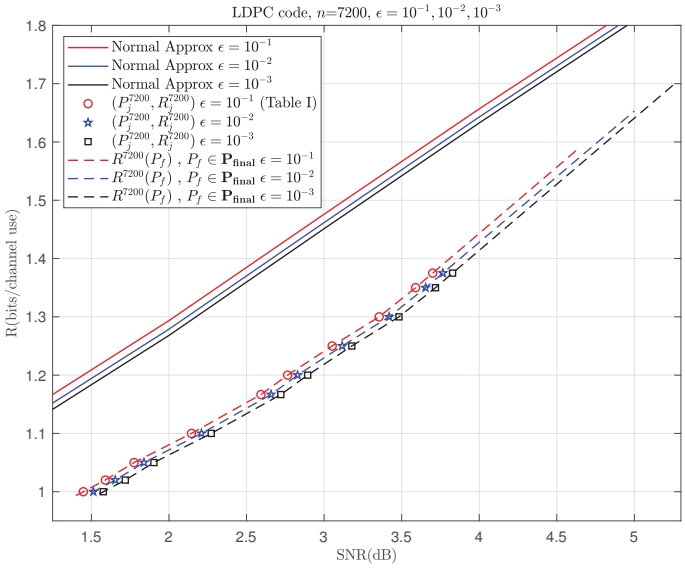
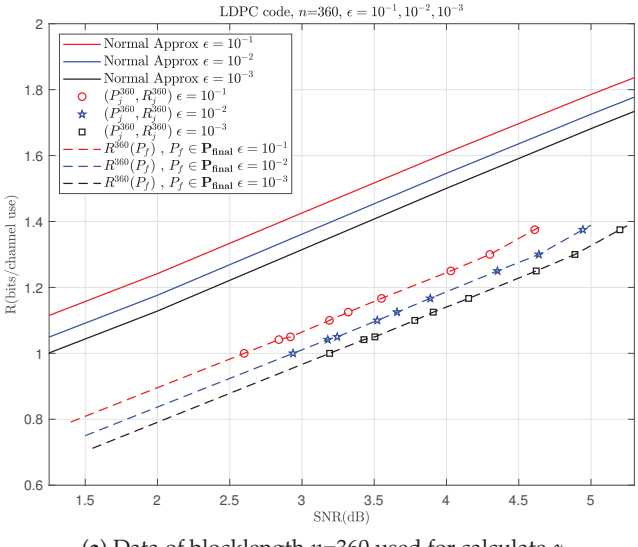
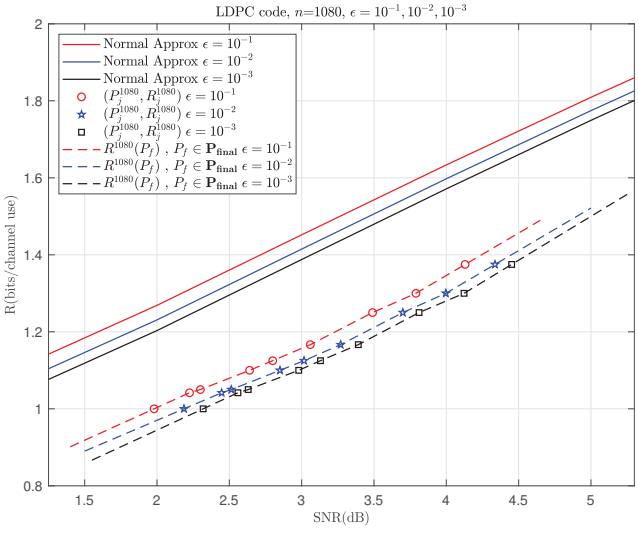


Figure 2. Calculate ΔC for HARQ using 16-QAM$_{1st}$ in the 1st transmission and QPSK$_{2nd}$ in the 2nd one with different BLERs, where the red circles, blue stars and black squares correspond to the points for $\epsilon = 10^{-1}$ (listed in Table 1), 10^{-2} and 10^{-3}, respectively.



(**a**) Data of blocklength n=360 used for calculate α

Figure 3. *Cont.*



(**b**) Data of blocklength n=1080 used for calculate α

Figure 3. Calculate α for HARQ using 16-QAM$_{1st}$ in the 1st transmission and QPSK$_{2nd}$ in the 2nd one with different BLERs, where the red circles, blue stars and black squares correspond to the points for $\epsilon = 10^{-1}$ (listed in Table 2), 10^{-2} and 10^{-3}, respectively.

Table 1. Calculate ΔC for HARQ using 16-QAM$_{1st}$ in the 1st transmission and QPSK$_{2nd}$ in the 2nd one, $n_{\text{inf}} = 7200$, $\epsilon = 10^{-1}$. Here, some points in $\mathbf{P_{final}}$ are shown.

$P_f \in \mathbf{P_{final}}$ (dB)	C_{th}	R^{7200}	ΔC
1.45	1.2	1	0.2
1.775	1.26	1.05	0.21
2.145	1.32	1.1	0.22
2.593	1.4	1.167	0.233
2.765	1.43	1.2	0.23
3.052	1.49	1.25	0.24
3.355	1.54	1.3	0.24
3.59	1.58	1.35	0.23
3.7	1.61	1.375	0.235

Table 2. Calculate α for HARQ using 16-QAM$_{1st}$ in the 1st transmission and QPSK$_{2nd}$ in the 2nd one, $n_1 = 360$, $n_2 = 1080$, $\epsilon = 10^{-1}$. Here, some points in $\mathbf{P_{final}}$ are shown.

$P_f \in \mathbf{P_{final}}$ (dB)	R^{360}	R^{1080}	C_{th}	V_{th}	ΔC	α
2.6	1	1.09	1.42	1.383	0.23	1
2.75	1.026	1.117	1.446	1.375	0.23	1
2.9	1.048	1.141	1.475	1.366	0.23	1
3.05	1.074	1.165	1.501	1.356	0.24	1.1849
3.2	1.102	1.194	1.529	1.344	0.24	1.6132
3.35	1.13	1.223	1.556	1.332	0.24	2.1820
3.5	1.157	1.251	1.583	1.319	0.235	2.5282

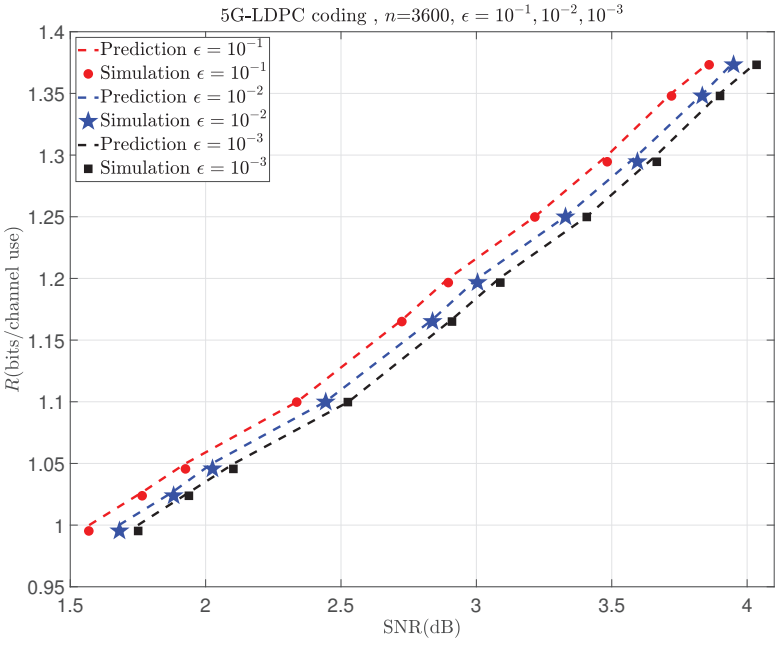


Figure 4. Prediction results of HARQ using 16-QAM$_{1st}$ in the 1st transmission and QPSK$_{2nd}$ in the 2nd one with different BLERs.

To more clearly distinguish between the simulation data points used before prediction and those used to validate the accuracy of the prediction afterward, we plot the simulation data points required before prediction in Figures 2 and 3a,b as hollow points and the simulation data points used for validation after prediction in Figures 4–7 as solid points, respectively. The predicted curves in Figures 4–7 are derived from the initial simulations, Algorithm 1, as well as the analytical formulas. The simulation points on the predicted curves are obtained by selecting certain SNRs and spectral efficiencies within the interval after predicting the performance and then conducting simulations for verification. The comparisons between the simulated points and the predicted curves show very small discrepancies. When the SNR is the same, the simulation value may be a little lower than the prediction curve.

The calculation environments for simulations and predictions are the same. We use MATLAB 2023b to calculate, use MATLAB's built-in functions *ldpcEncode* and *ldpcDecode* for encoding and decoding LDPC codes, use MATLAB's built-in functions *qammod* and *qamdemod* for modulation and demodulation, and we use MATLAB's built-in functions *awgn* to add noise.

The following examples show the prediction results of moderate-blocklength coded retransmission with other modulation schemes, like 1024-QAM$_{1st}$ and 256-QAM$_{2nd}$, in the first and the second transmissions. They are shown in Figures 5–7, respectively.

As shown in these figures, different combinations of modulation schemes can cover different ranges of rates. In Figure 5, we can see that the combination of 64-QAM and QPSK covers the rate range from 1.33 to 1.83 in the SNR from 4.13 dB to 6.96 dB; that of 64-QAM and 16-QAM covers the rate range from 1.67 to 2.29 in the SNR from 5.82 dB to 8.52 dB; and 64-QAM combined with 64-QAM covers the rate range from 2 to 2.75 in the SNR from 7.3 dB to 10.6 dB.

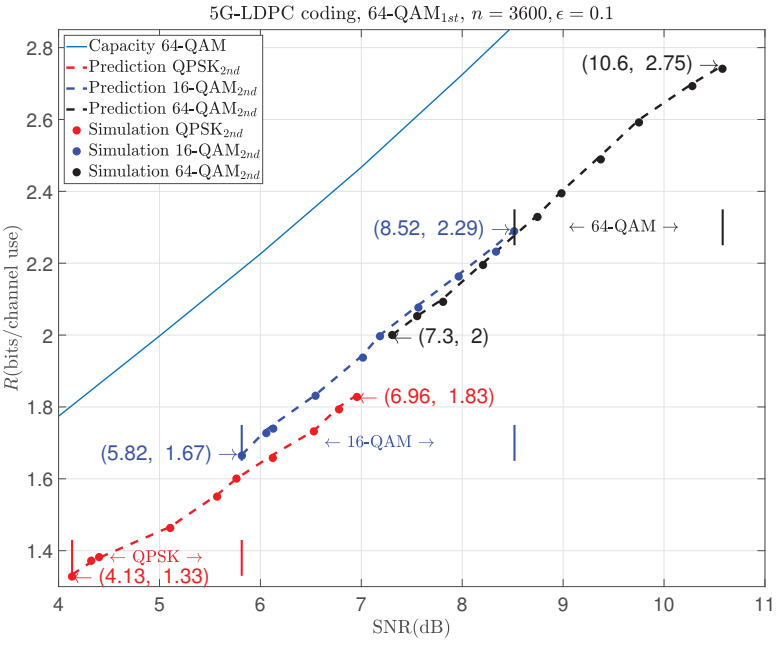


Figure 5. Prediction results of HARQ using 64-QAM$_{1st}$ in the 1*st* transmission and m_2-QAM$_{2nd}$ in the 2*nd* one.

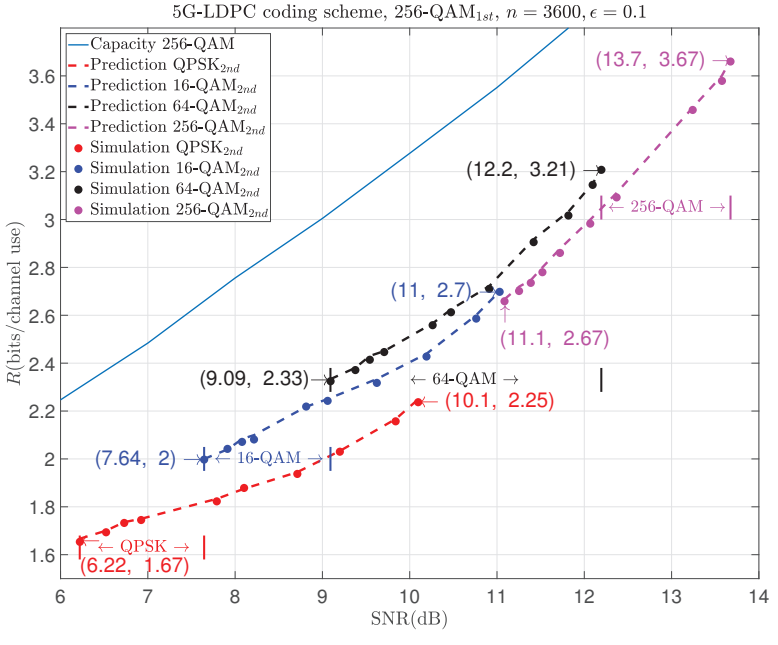


Figure 6. Prediction results of HARQ using 256-QAM$_{1st}$ in the 1*st* transmission and m_2-QAM$_{2nd}$ in the 2*nd* one.

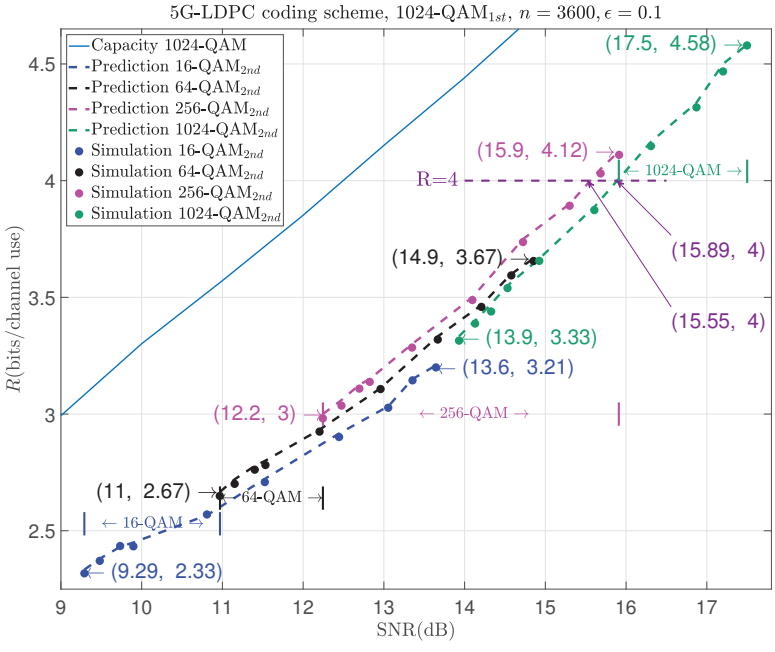


Figure 7. Prediction results of HARQ using 1024-QAM$_{1st}$ in the 1*st* transmission and m_2-QAM$_{2nd}$ in the 2*nd* one.

The results in Figure 6 show that if 256-QAM is used for the initial transmission and if the aim for a retransmission is to achieve the rate range between 2 and 2.25, we can use QPSK or 16-QAM for the retransmission. In this case, the required SNR range is between 7.64 dB and 10.1 dB. Similarly, if 16-QAM or 64-QAM is used for the retransmission, the corresponding SNR range is between 9.09 dB and 11 dB, resulting in a rate range of 2.33 to 2.7. If 256-QAM is still used during retransmission, it will cover a rate range from 2.67 to 3.67 in the SNR from 11.1 dB to 13.7 dB.

However, we can also see in these three figures that using the same modulation scheme during retransmission as before results in poorer performance. For example, in Figure 7, to achieve the same rate like $R = 4$, the combination of 1024-QAM and 1024-QAM performs about 0.34 dB poorer than that of 1024-QAM and 256-QAM.

The results above can guide us in selecting different modulation schemes based on varying data rate requirements during retransmission. For example, as illustrated in Figure 7, when using 1024-QAM for the initial transmission, if the rate is between 2.33 and 2.67, it would be better to use 16-QAM for retransmission. This is because the minimum code rate of LDPC code is 1/3, and the minimum rate of 1024-QAM combined with 64-QAM is $R = \frac{1}{3} \times \frac{1}{2} \times (10 + 6) = 2.67$. Similarly, if the rate is between 3 and 4.12, it would be better to use 256-QAM for retransmission because the maximum code rate after retransmission is $R_c = \frac{11}{12} \times \frac{1}{2} = \frac{11}{24}$, and the maximum rate of 1024-QAM combined with 256-QAM is $R = \frac{11}{24} \times \frac{1}{2} \times (10 + 8) = 4.12$. If the rate is greater than 4.12, we only use 1024-QAM to retransmit. Similar conclusions can be drawn for 64-QAM and 256-QAM in Figures 5 and 6.

5. Efficiency Analysis

Figure 8 takes the MCS of 5G NR with the BG1 matrix as an example, where we select a set of coding parameters with blocklengths ranging from $n = 360$ ($Z = 18$) to the maximum length of $n = 8448$ ($Z = 384$). The ranges of code rates, respectively, cover $[\frac{2}{3}, \frac{11}{12}]$ and $[\frac{1}{3}, \frac{11}{24}]$ in the first transmission and retransmission with a total $n_{R_c} = 11$ different code

rates. Increasing the value of n_{R_c} can further fine-tune the prediction accuracy. For the system-level simulations that are crucial for the design of 5G networks, it is generally required to obtain the link-level BLER performance metrics for all data points in Figure 8 through simulations. Then, for the specific link settings of code rates and blocklengths, the BLER performances can be directly obtained via linear interpolation with the nearby data points. For future mobile communication systems, with wider ranges of code rates and blocklengths and lower BLER targets, the performance evaluations for link-level simulations with multiple retransmissions will significantly increase the computational complexity. Our proposed performance prediction scheme can effectively reduce the computational load while ensuring evaluation accuracy.

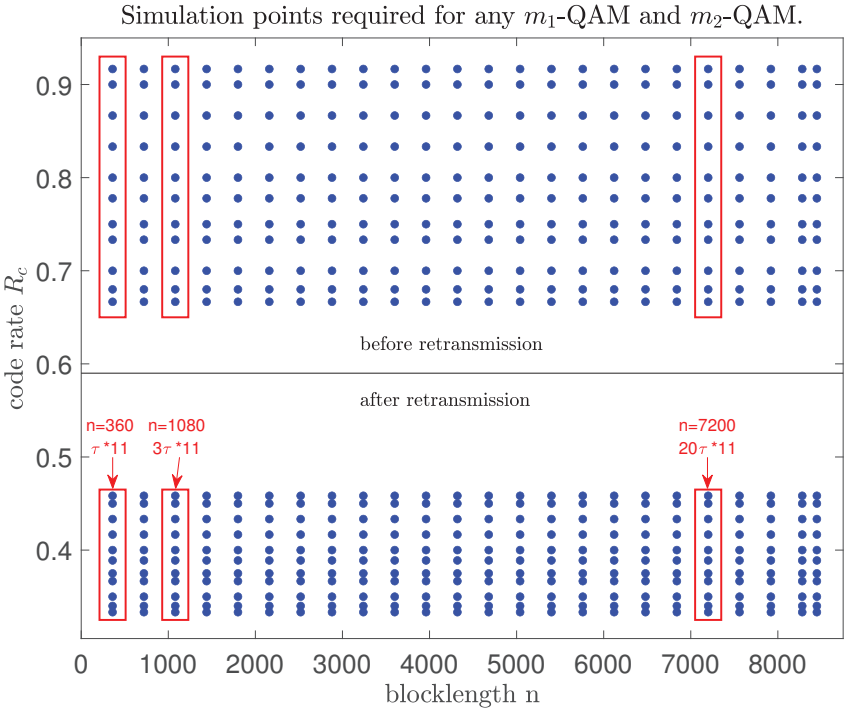


Figure 8. The link settings required for system-level simulations with different code rates and blocklengths, where m_1-QAM and m_2-QAM are used in the first transmission and second transmission, respectively.

In the above example of MCSs shown in Figure 8, once the m_1-QAM, m_2-QAM and BLER are determined before or after retransmission, we only need to simulate all the rate data for three sets of blocklengths (the blue points in three red rectangle boxes in Figure 8) to predict the rates for any other blocklength (other blue points in Figure 8). Hence, when the BLERs and SNRs for the coded modulation combinations with all the different blocklengths and code rates are required for system-level simulations, using our algorithm can significantly improve the efficiency of performance evaluations. As shown in Figure 8, assuming that a single testing of BLER ϵ evaluation for a code with $n = 360$ requires a time of τ, then obtaining one set of data requires a time of $\tau \times 11$ for all code rates considered before or after retransmission. As the blocklength increases, the simulation time will also increase linearly, which means that the simulation testings of performance evaluation for the codes with $n = 1080$ and $n = 7200$ require a computation time of 3τ and 20τ, respectively. Therefore, the total simulation time required to obtain all the data needed for the performance prediction of MCSs with m_1-QAM and m_2-QAM is $(\tau + 3\tau + 20\tau) \times 11 = 264\tau$. Since the time required for the calculations of ΔC and α

is negligible compared to that of simulation tests for performance evaluations, the total computational complexity needed to complete the entire prediction can be approximately evaluated by 264τ. Then, if we need the rate data of codes from $n = 360$ to 8448 according to all the lifting values of 5G-LDPC codes shown in Figure 8, the total computational complexity required for a brute-force Monte Carlo simulation should be about 3300τ, which is clearly greater than 264τ. Since each modulation combination for retransmission requires a separate simulation, this algorithm can significantly reduce the computational complexity when a large amount of SNR-R relationship data corresponding to various blocklengths are needed, given a specific BLER ϵ.

6. Conclusions

In this paper, we have reviewed the theoretical prediction formulas for the performance of finite-length coding and their correction models. Through simulation, we validated the good adaptability of the correction model to the retransmission scenarios. To make this model more general, we introduced a range $\mathbf{P_{final}}$ to unify the different SNRs. Based on the simulation results, we can choose the modulation method for the second transmission according to different bit rate requirements. It is also evident that if the same modulation method is employed in the second transmission as before, its performance is not as effective as some methods involving a reduction in the modulation order during retransmission.

Author Contributions: Software, Y.W.; validation, M.J.; data curation, Y.W., F.D. and Q.X.; writing—original draft preparation, Y.W. and M.J.; writing—review and editing, Y.W. and M.J.; supervision, M.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (NSFC) under grant 62331002.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Random variables and its realization are denoted by capital letters X and its lower case x, respectively. The boldface letter denotes a vector. A sequence $x_1, \ldots, x_n$ is denoted by x^n. A sequence x^n partitioned into 2 blocks with equal blocklength $n_1 = n_2 = \frac{n}{2}$ is denoted by $x^n = [\mathbf{x_1}, \mathbf{x_2}]$, where n is the blocklength of the code, and the ith element of vector $\mathbf{x}_1, \mathbf{x}_2$ is denoted by $\mathbf{x}_{1,i}, \mathbf{x}_{2,i}$. The total variational distance between two distributions D and Q is denoted by $\|D - Q\|_{TV}$. The Radon-Nikodym derivative of D w.r.t. Q is denoted by $\frac{dD}{dQ}$. Expectations and variances taken w.r.t. a distribution D are indicated by $\mathbb{E}_D[\cdot]$ and $\mathrm{Var_D}[\cdot]$, respectively.

Let $(Q_X(x(1)), \ldots, Q_X(x(m)))$ denote an input distribution, where $x(i)$'s are the m-QAM constellation points with average power Pw. Denote by $\widehat{D}_X$ the composition $\left(\widehat{D}_X(x(1)), \ldots, \widehat{D}_X(x(m))\right) = \left(\widehat{Q}_X(x(1)), \ldots, \widehat{Q}_X(x(m))\right)$, if $nQ_X(x)$ is an integer for all x, or otherwise, the type closest to the Q_X in total variational distance, in which the greater probabilities are assigned first to the indices $i \in \{1, \ldots, m\}$ whose corresponding constellation symbols $x(i)$ have individual power equal to Pw (if available), then to indices whose corresponding symbols have individual power less than Pw, and finally to those with corresponding individual power greater than Pw.

In the retransmission scenario, the 2 segments are effectively transmitted over the same BIAWGN channel, hence:

$$\mathbf{Y}_k = \mathbf{X}_k + \mathbf{W}_k, \tag{A1}$$

where $\mathbf{W}_k \sim \mathcal{CN}\left(\mathbf{0}_{n_k}, I_{n_k}\sigma_k^2\right), k = 1, 2$, $\sigma_1 = \sigma_2 = \sigma$, and I_{n_k} is an identity matrix. Therefore, the SNR is

$$P = \frac{Pw}{\sigma}. \tag{A2}$$

The proof of the channel capacity C and the channel dispersion V in the retransmission scenario is an application of Theorem 2 from [25], which is stated as follows.

Theorem A1. *For a channel $D_{Y^n|X^n}(y^n|n^n)$ any input distribution D_{X^n}, and any output distributuion Q_{Y^n}, there exists a code with M codewords in $\mathcal{F}_n$ and average probability of error satisfying*

$$\epsilon \leq D_{X^n} D_{Y^n|X^n}[\widetilde{i}(X^n;Y^n) \leq \log_2\gamma_n] + L_n M D_{X^n} Q_{Y^n}[\widetilde{i}(X^n;Y^n) > \log_2\gamma_n] + D_{X^n}[X^n \notin \mathcal{F}_n] \tag{A3}$$

where

$$\widetilde{i}(X^n;Y^n) = \log_2 \frac{D_{Y^n|X^n}(Y^n|X^n)}{Q_{Y^n}(Y^n)} \tag{A4}$$

and the coefficient L_n is defined as

$$L_n \triangleq \sup_{y^n \in \mathcal{Y}^n} \frac{dD_{Y^n}(y^n)}{dQ_{Y^n}(y^n)} \tag{A5}$$

and γ_n is an arbitary positive threshold whose optimal choice to give the highest rates is $\gamma_n = L_n M$.

This proof follows similar steps as the proof for the single AWGN channel in [25] with the modification of taking into account the retransmission scenario. For the 2 segments of length $n_k = \frac{n}{2}, k = 1, 2$, denote by $\widehat{D}_X^{(k)}$ the composition of segment k and $T_{n_k}\left(\widehat{D}_X^{(k)}\right)$ the corresponding type class. For the different modulation schemes m_1-QAM and m_2-QAM of the 2 segments. We fix

$$Q_X(x(i)) = \frac{1}{m_k}, i = 1, \ldots, m_k, k = 1, 2. \tag{A6}$$

Under this construction, it can be shown that

$$\left\|\widehat{D}_X^{(k)} - Q_X\right\|_{TV} \leq \mathcal{O}\left(\frac{1}{n_k}\right), \tag{A7}$$

and

$$\mathbb{E}_{\widehat{D}_X^{(k)}}\left[\|X\|^2\right] \leq Pw. \tag{A8}$$

We choose the input distribution D_{X^n} as the following:

$$D_{X^n}(x^n) = \prod_{k=1}^{2} \frac{\left\{\mathbf{x}_k \in T_{n_k}\left(\widehat{D}_X^{(k)}\right)\right\}}{\left(n_k \widehat{D}_X^{(k)}(x(1)), \ldots, n_k \widehat{D}_X^{(k)}(x(m_k))\right)}. \tag{A9}$$

Let

$$\mathcal{F}_n = \left\{x^n : \|\mathbf{x}_k\|^2 \leq n_k Pw, k = 1, 2\right\}. \tag{A10}$$

It can be verified that the input distribution from (A7) satisfies the maximal power constraint $\|\mathbf{x}_k\|^2 \leq n_k Pw$. Hence,

$$D_{X^n}[X^n \notin \mathcal{F}_n] = 0. \tag{A11}$$

The output distribution induced by the input distribution D_{X^n} can be written as

$$D_{Y^n}(y^n) = \sum_{x^n \in \mathcal{X}^n} D_{X^n}(x^n) D_{Y^n|X^n}(y^n|x^n) = \prod_{k=1}^{2} \sum_{\mathbf{x}_k \in T_{n_k}\left(\widehat{P}_X^{(k)}\right)} \Gamma_k D_{\mathbf{Y}_k|\mathbf{X}_k}(\mathbf{y}_k|\mathbf{x}_k), \tag{A12}$$

where

$$\Gamma_k = \frac{1}{\binom{n_k}{n_k \widehat{D}_X^{(k)}(x(1)), \ldots, n_k \widehat{D}_X^{(k)}(x(m_k))}} \tag{A13}$$

and

$$D_{\mathbf{Y}_k|\mathbf{X}_k}(\mathbf{y}_k|\mathbf{x}_k) = \prod_{t=1}^{n_k} D_{Y|X}^{(k)}(y_t|x_t) \tag{A14}$$

and $D_{Y|X}^{(k)}$ indicates the channel experienced by segment $k, k = 1, 2$.

Next, we choose the following auxiliary distributions:

$$Q_Y^{(k)}(y) = \sum_{i=1}^{m_k} Q_X(x(i)) D_{Y|X}^{(k)}(y|x(i)), k = 1, 2 \tag{A15}$$

$$Q_{Y^n}(y^n) = \prod_{k=1}^{2} \prod_{t=1}^{n_k} Q_Y^{(k)}(\mathbf{y}_{k,t}) \tag{A16}$$

where Q_X is given in (A6).

It can be shown by applying Proposition 3 of [25] to (A12) and (A16), we have

$$\frac{dP_{Y^n}(y^n)}{dQ_{Y^n}(y^n)} \leq L_n \triangleq \prod_{k=1}^{2} c_k(m_k) n_k^{\frac{m_k-1}{2}} \tag{A17}$$

for sufficiently large n_k's, where $c_k(m_k)$'s are positive constants that depend only on the constellation size m.

We now apply Theorem A1 to the distributions defined above. Let $\gamma_n = L_n M$. For the first term in (A3), it can be verified that

$$\frac{1}{n}\widetilde{i}(x^n; Y^n) = -\frac{1}{n} \sum_{k=1}^{2} \sum_{t=1}^{n_k} \log_2 \left(\sum_{j=1}^{m_k} Q_X(x(j)) e^{\frac{\left\|\mathbf{w}_{k,t}\right\|^2 - \left\|\mathbf{x}_{k,t} + \mathbf{w}_{k,t} - x(j)\right\|^2}{\sigma^2}} \right). \tag{A18}$$

Since $\{\mathbf{W}_{k,t}\}$'s are independent, we can invoke the Berry-Esseen Theorem on $\frac{1}{n}\widetilde{i}(x^n; Y^n)$. Using (A7), the mean can be verified to be

$$\mathbb{E}_{D_{Y^n|X^n=x^n}} \left[\frac{1}{n}\widetilde{i}(x^n; Y^n) \right] = \frac{1}{2}(C_{m1}(P) + C_{m2}(P)) + \mathcal{O}\left(\frac{1}{n}\right) = C + \mathcal{O}\left(\frac{1}{n}\right). \tag{A19}$$

Similarly, the variance can be verified to be

$$\mathrm{Var}_{D_{Y^n|X^n=x^n}} \left[\frac{1}{n}\widetilde{i}(x^n; Y^n) \right] = \frac{1}{n}\left(\frac{1}{2}(V_{m1}(P) + V_{m2}(P))\right) + \mathcal{O}\left(\frac{1}{n}\right) = \frac{V}{n} + \mathcal{O}\left(\frac{1}{n}\right). \tag{A20}$$

Applying the Berry-Esseen Theorem on $\frac{1}{n}\widetilde{i}(x^n; Y^n)$ yields

$$D_{Y^n|X^n}\left[\frac{1}{n}\widetilde{i}(x^n; Y^n) \leq \frac{\log_2(L_n M)}{n} \right] \leq Q\left(\frac{C - \frac{\log_2(L_n M)}{n}}{\sqrt{\frac{V}{n}}} \right) + \frac{B_1}{\sqrt{n}}, \tag{A21}$$

where B_1 is some positive constant. Consequently, by averaging over the input sequences, the first term from (A3) can be bounded by

$$D_{X^n} D_{Y^n|X^n}\left[\frac{1}{n}\widetilde{i}(X^n; Y^n) \leq \frac{\log_2(L_n M)}{n} \right] \leq Q\left(\frac{C - \frac{\log_2(L_n M)}{n}}{\sqrt{\frac{V}{n}}} \right) + \frac{B_1}{\sqrt{n}}. \tag{A22}$$

For the second term in (A3), observe that under the conditional distribution $D_{Y^n|X^n=x^n}$, $\frac{1}{n}\widetilde{i}(x^n;Y^n)$ is a summation of independent random variables with positive variance and finite third absolute moment. Therefore, we can apply the refined large deviation result of Lemma 47 in [24]

$$Q_{Y^n}\left[\widetilde{i}(x^n;Y^n) > \log_2\gamma_n\right] = \mathbb{E}\left[e^{-\widetilde{i}(x^n;Y^n)}\left\{\widetilde{i}(x^n;Y^n) > \log_2(L_nM)\right\}\right] \leq \frac{B_2}{\sqrt{n}}(L_nM)^{-1}, \tag{A23}$$

where the expectation is taken with respect to $D_{Y^n|X^n=x^n}$ and B_2 is a positive constant. Therefore, the second term from (A3) can be bounded as

$$L_nMD_{X^n}Q_{Y^n}\left[\widetilde{i}(X^n;Y^n) > \log_2(L_nM)\right] \leq \frac{B_2}{\sqrt{n}}. \tag{A24}$$

Finally, combining (A22), (A24) and (A11), yields

$$\epsilon \leq Q\left(\frac{\frac{C-\log_2(L_nM)}{n}}{\sqrt{\frac{V}{n}}}\right) + \frac{B}{\sqrt{n}}, \tag{A25}$$

where $B = B_1 + B_2$. Rearranging (A25) and with a bit of analysis yields the result of C and V in retransmission scenario for (9).

References

1. Panwar, N.; Sharma, S.; Singh, A.K. A survey on 5G: The next generation of mobile communication. *Phys. Commun.* **2016**, *18*, 64–84. [CrossRef]
2. El Hattachi, R.; Erfanian, J. 5G white paper. In *Next Generation Mobile Networks*; White Paper; NGMN Alliance: Dusseldorf, Germany, 2015; Volume 1, p. 1.
3. Andrews, J.G.; Buzzi, S.; Choi, W.; Hanly, S.V.; Lozano, A.; Soong, A.C.K.; Zhang, J.C. What Will 5G Be? *IEEE J. Sel. Areas Commun.* **2014**, *32*, 1065–1082. [CrossRef]
4. Dimitrakopoulos, G.; Demestichas, P. Intelligent Transportation Systems. *IEEE Veh. Technol. Mag.* **2010**, *5*, 77–84. [CrossRef]
5. Anttiroiko, A.V. *Electronic Government: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications*; IGI Global: Hershey, PA, USA, 2008; Volume 3.
6. Mughees, A.; Tahir, M.; Sheikh, M.A.; Amphawan, A.; Meng, Y.K.; Ahad, A.; Chamran, K. Energy-efficient joint resource allocation in 5G HetNet using Multi-Agent Parameterized Deep Reinforcement learning. *Phys. Commun.* **2023**, *61*, 102206. [CrossRef]
7. Ghosh, J.; Vargas-Rosales, C.; Mendes, L.L.; Ra, I.H.; Nhan Vo, V.; Aimtongkham, P.; So-In, C. A Novel Transceiver and an Asynchronous Mode for the Hybrid Multiple-Access HetNet Architecture. *IEEE Access* **2023**, *11*, 135609–135625. [CrossRef]
8. Girycki, A.; Rahman, M.A.; Vinogradov, E.; Pollin, S. Learning-Based Precoding-Aware Radio Resource Scheduling for Cell-Free mMIMO Networks. *IEEE Trans. Wirel. Commun.* **2024**, *23*, 4876–4888. [CrossRef]
9. Zhang, J.; Xi, R.; He, Y.; Sun, Y.; Guo, X.; Wang, W.; Na, X.; Liu, Y.; Shi, Z.; Gu, T. A Survey of mmWave-Based Human Sensing: Technology, Platforms and Applications. *IEEE Commun. Surv. Tutor.* **2023**, *25*, 2052–2087. [CrossRef]
10. Xue, Q.; Ji, C.; Ma, S.; Guo, J.; Xu, Y.; Chen, Q.; Zhang, W. A Survey of Beam Management for mmWave and THz Communications towards 6G. *arXiv* **2024**, arXiv:2308.02135. [CrossRef]
11. Lai, W.K.; Wang, Y.C.; Lin, H.C.; Li, J.W. Efficient Resource Allocation and Power Control for LTE-A D2D Communication with Pure D2D Model. *IEEE Trans. Veh. Technol.* **2020**, *69*, 3202–3216. [CrossRef]
12. Gismalla, M.S.M.; Azmi, A.I.; Salim, M.R.B.; Abdullah, M.F.L.; Iqbal, F.; Mabrouk, W.A.; Othman, M.B.; Ashyap, A.Y.I.; Supa'at, A.S.M. Survey on Device to Device (D2D) Communication for 5GB/6G Networks: Concept, Applications, Challenges, and Future Directions. *IEEE Access* **2022**, *10*, 30792–30821. [CrossRef]
13. Mazhar, M.S.; Saleem, Y.; Almogren, A.; Arshad, J.; Jaffery, M.H.; Rehman, A.U.; Shafiq, M.; Hamam, H. Forensic Analysis on Internet of Things (IoT) Device Using Machine-to-Machine (M2M) Framework. *Electronics* **2022**, *11*, 1126. [CrossRef]
14. Pan, C.; Zhou, G.; Zhi, K.; Hong, S.; Wu, T.; Pan, Y.; Ren, H.; Renzo, M.D.; Lee Swindlehurst, A.; Zhang, R.; et al. An Overview of Signal Processing Techniques for RIS/IRS-Aided Wireless Systems. *IEEE J. Sel. Top. Signal Process* **2022**, *16*, 883–917. [CrossRef]
15. Wijethilaka, S.; Liyanage, M. Survey on Network Slicing for Internet of Things Realization in 5G Networks. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 957–994. [CrossRef]
16. Schiessl, S.; Gross, J.; Al-Zubaidy, H. Delay Analysis for Wireless Fading Channels with Finite Blocklength Channel Coding. In Proceedings of the MSWiM—Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'15), Dubrovnik, Croatia, 24–28 August 2015.

17. Mary, P.; Gorce, J.M.; Unsal, A.; Poor, H.V. Finite Blocklength Information Theory: What Is the Practical Impact on Wireless Communications? In Proceedings of the IEEE Globecom Workshops (GC Wkshps 2016), Washington, DC, USA, 4–8 December 2016; IEEE: New York, NY, USA, 2016.
18. Zhu, Y.; Hu, Y.; Yuan, X.; Gursoy, M.C.; Poor, H.V.; Schmeink, A. Joint Convexity of Error Probability in Blocklength and Transmit Power in the Finite Blocklength Regime. *IEEE Trans. Wirel. Commun.* **2022**, *22*, 2409–2423. [CrossRef]
19. Makki, B.; Svensson, T.; Caire, G.; Zorzi, M. Fast HARQ Over Finite Blocklength Codes: A Technique for Low-Latency Reliable Communication. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 194–209. [CrossRef]
20. You, X.; Sheng, B.; Huang, Y.; Xu, W.; Zhang, C.; Wang, D.; Zhu, P.; Ji, C. Closed-Form Approximation for Performance Bound of Finite Blocklength Massive MIMO Transmission. *IEEE Trans. Commun.* **2023**, *71*, 6939–6951. [CrossRef]
21. Valembois, A.; Fossorier, M. Sphere-Packing Bounds Revisited for Moderate Block Lengths. *IEEE Trans. Inf. Theory* **2004**, *50*, 2998–3014. [CrossRef]
22. Lazic, D.; Beth, T.; Egner, S. Constrained capacity of the AWGN channel. In Proceedings of the 1998 IEEE International Symposium on Information Theory (ISIT-98), Cambridge, MA, USA, 16–21 August 1998.
23. Shi, J.; Wesel, R.D. A Study on Universal Codes With Finite Block Lengths. *IEEE Trans. Inf. Theory* **2007**, *53*, 3066–3074. [CrossRef]
24. Polyanskiy, Y.; Poor, H.V.; Verdu, S. Channel Coding Rate in the Finite Blocklength Regime. *IEEE Trans. Inf. Theory* **2010**, *56*, 2307–2359. [CrossRef]
25. MolavianJazi, E. *A Unified Approach to Gaussian Channels with Finite Blocklength*; University of Notre Dame: Notre Dame, IN, USA, 2014.
26. Song, E.C.; Yue, G. Finite Blocklength Analysis for Coded Modulation with Applications to Link Adaptation. In Proceedings of the 2019 IEEE Wireless Communications and Networking Conference (WCNC), Marrakesh, Morocco, 15–18 April 2019.

MDPI AG
Grosspeteranlage 5
4052 Basel
Switzerland
Tel.: +41 61 683 77 34

Entropy Editorial Office
E-mail: entropy@mdpi.com
www.mdpi.com/journal/entropy

Academic Open
Access Publishing

mdpi.com

ISBN 978-3-7258-3878-3