


Article

# Network Coding Approaches for Distributed Computation over Lossy Wireless Networks

Bin Fan <sup>1,2</sup>, Bin Tang <sup>1,2,\*</sup> , Zhihao Qu <sup>1,2</sup> and Baoliu Ye <sup>1,2</sup>

<sup>1</sup> Key Laboratory of Water Big Data Technology of Ministry of Water Resources, Hohai University, Nanjing 211100, China

<sup>2</sup> 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



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 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  $\mathbf{y} = \mathbf{Ax}$ , 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  $\mathbf{A}_1, \dots, \mathbf{A}_k$ , and each worker node is assigned multiple submatrices, each of which is a random linear combination of the  $\mathbf{A}_1, \dots, \mathbf{A}_k$ . 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  $\mathbf{A}\mathbf{x}$  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 remainder 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  $w_1, w_2, \dots, w_n$ , 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.

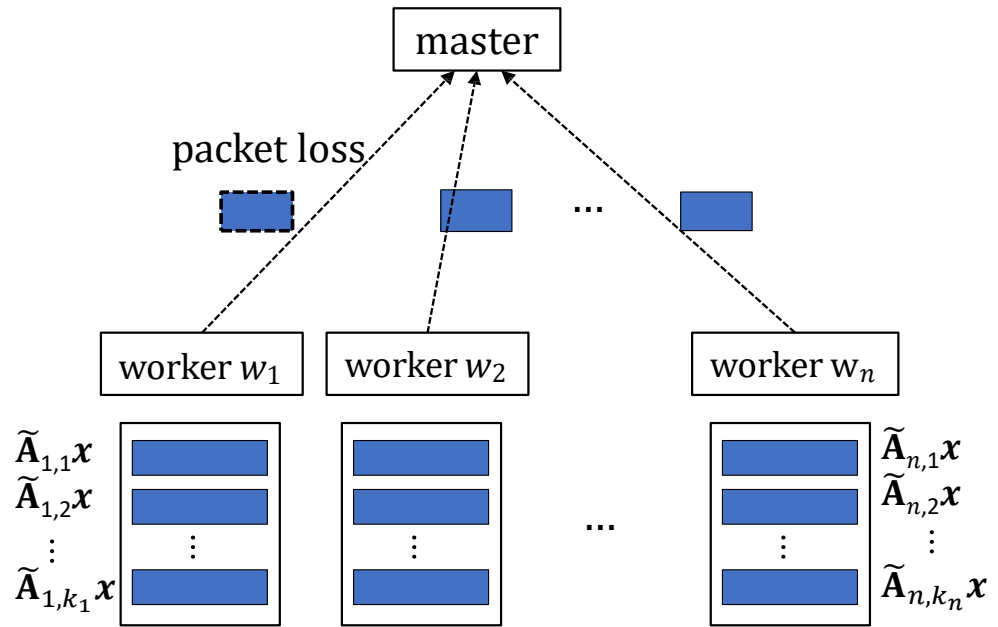


Figure 1. Illustration of the wireless distributed computing system for matrix multiplication.

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  $\mathbf{A}_1, \dots, \mathbf{A}_k$ , i.e.,  $\mathbf{A}^T = [\mathbf{A}_1^T \ \mathbf{A}_2^T \ \dots \ \mathbf{A}_k^T]$ . 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  $\tilde{\mathbf{A}}_{i,1}, \tilde{\mathbf{A}}_{i,2}, \dots, \tilde{\mathbf{A}}_{i,k_i}$ , 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  $\tilde{\mathbf{A}}_{i,1}\mathbf{x}, \tilde{\mathbf{A}}_{i,2}\mathbf{x}, \dots, \tilde{\mathbf{A}}_{i,k_i}\mathbf{x}$  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  $\varepsilon_i$ . 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  $\lambda_i$ . The second is a constant time for calculating each submatrix-vector product, which is denoted by  $\tau_i$ . 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  $\mu_i$  and shift parameter  $\theta_i$ . 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  $\mathbf{A}_1, \dots, \mathbf{A}_k$ ; i.e.,

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

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  $\tilde{\mathbf{y}}_{i,1} = \tilde{\mathbf{A}}_{i,1}\mathbf{x}, \tilde{\mathbf{y}}_{i,2} = \tilde{\mathbf{A}}_{i,2}\mathbf{x}, \dots, \tilde{\mathbf{y}}_{i,k_i} = \tilde{\mathbf{A}}_{i,k_i}\mathbf{x}$ , 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,  $(c'_1, \dots, c'_{d_i(t)})$  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  $\mathbf{A}_1 \mathbf{x}, \mathbf{A}_2 \mathbf{x}, \dots, \mathbf{A}_k \mathbf{x}$ . Here,

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

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  $\mathbf{A}_1 \mathbf{x}, \mathbf{A}_2 \mathbf{x}, \dots, \mathbf{A}_k \mathbf{x}$  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 be done before any computation and just once, which can be used for computing  $\mathbf{A} \mathbf{x}$  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 an 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  $T_{\text{RLNC}}$ , satisfies

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

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  $T_{\text{any}}$  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  $m_i(t_1)$ , 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.



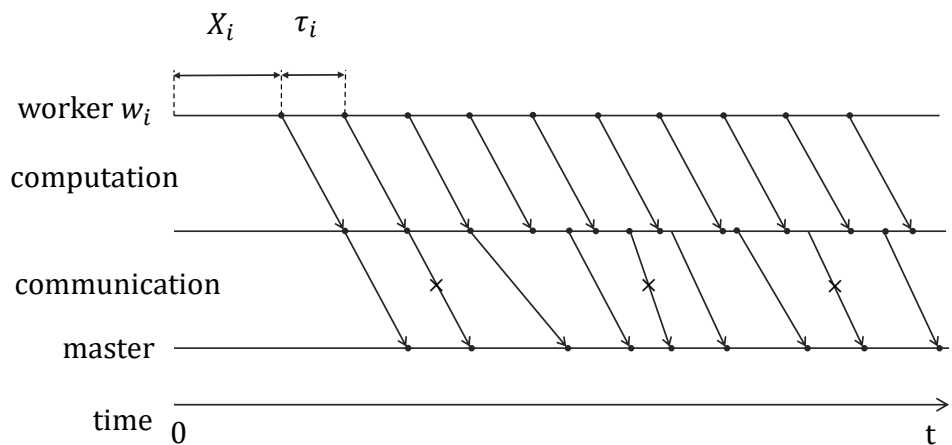


Figure 2. Illustration of a continuous-time trellis,  $G_i^{(t)}$ .

**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  $\mathbb{R}$ .  $\square$

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  $Y_1, Y_2, \dots$  follow a shifted exponential distribution with rate  $\mu$  and shift parameter  $\theta$  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,

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

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.  $\square$

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

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

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

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

$\square$

**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  $N_i(t) \geq (1 + \delta/2)r_it$ . By the total law of probability,

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

According to Lemma 2, there exists some constant  $\eta'_2 > 1$  such that  $\Pr(A) = \mathcal{O}(\eta_2^{-t})$ . Let  $N$  be a binomial random variable with parameters  $(1 + \delta/2)r_it$  and  $1 - \varepsilon_i$ . Then, there exists some constant  $\eta'_3 > 1$  such that

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



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

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

□

**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

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

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

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

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  $F_i((1 - \delta)T_0) \leq N'_i((1 - \delta)T_0 - X_i)$ ,

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

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  $\eta_4 = \min\{\eta'_4, \eta_5\}$ .  $\square$

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  $T_{\text{any}}$  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.  $\square$

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  $t \geq X_i$ ,

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

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

**Proof.** The result can be shown similarly to that of Lemma 3.  $\square$

**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,

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

**Proof.** According to Theorem 1 of [26], Lemma 5 implies this result immediately.  $\square$

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  $k \rightarrow \infty$ . Hence,  $\sum_{i=1}^n F_i(T_0) \xrightarrow{P} k$  as  $k \rightarrow \infty$ . 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

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

According to Proposition 1, this implies that

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

The proof is accomplished.  $\square$

#### 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  $\mathbf{A}_1, \dots, \mathbf{A}_k$  are first encoded into  $\mathbf{A}_1, \dots, \mathbf{A}_k, \mathbf{A}_{k+1}, \dots, \mathbf{A}_{k'}$  using a fixed-rate systematic erasure code (called a precode), where  $k' = (1 + \epsilon)k$  and  $\epsilon$  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  $\Psi = (\Psi_1, \dots, \Psi_D)$ , where  $D$  is the maximum degree;
- Select  $deg$  distinct submatrices uniformly at random from  $\mathbf{A}_1, \dots, \mathbf{A}_k, \mathbf{A}_{k+1}, \dots, \mathbf{A}_{k'}$ ;
- 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  $\mathbf{A}_1\mathbf{x}, \dots, \mathbf{A}_k\mathbf{x}, \mathbf{A}_{k+1}\mathbf{x}, \dots, \mathbf{A}_{k'}\mathbf{x}$  using Gaussian-elimination-based belief propagation (BP) decoding, and once any  $k$  or slightly more than  $k$  of  $\mathbf{A}_1\mathbf{x}, \dots, \mathbf{A}_k\mathbf{x}, \mathbf{A}_{k+1}\mathbf{x}, \dots, \mathbf{A}_{k'}\mathbf{x}$  are recovered, the master node can recover all these  $\mathbf{A}_1\mathbf{x}, \dots, \mathbf{A}_k\mathbf{x}$  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  $\mathbf{h}_i = (h_{i,0}, \dots, h_{i,M})$  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  $Z_i$ :

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

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  $\Pr(Z_i = \ell), \ell = 0, \dots, ub$  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  $\Psi$  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  $\lambda_i$ ,  $\tau_i$ ,  $\mu_i$ ,  $\theta_i$  and  $\varepsilon_i$  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.

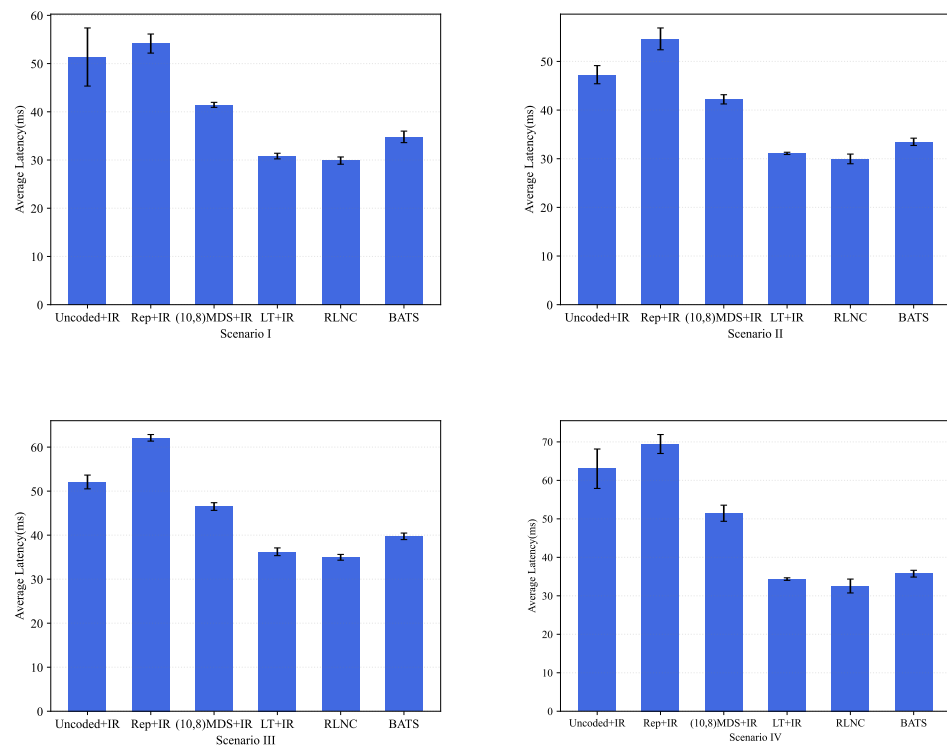
For these scenarios, we evaluated the following five methods.

- **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](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.