Near-optimal One-sided Scheduling for Coded Segmented Network Coding

Bin Tang, Shenghao Yang, Member, IEEE, Baoliu Ye, Member, IEEE, Song Guo, Senior Member, IEEE, and Sanglu Lu, Member, IEEE

Abstract—As a variation of random linear network coding, segmented network coding (SNC) has attracted great interest in data dissemination over lossy networks due to its low computational cost. In order to guarantee the success of decoding, SNC can adopt a feedbackless FEC (forward error correction) approach by applying a linear block code to the input packets before segmentation at the source node. In particular, if the empirical rank distribution of transfer matrices of segments is known in advance, several classes of coded SNC can achieve close-to-optimal decoding performance. However, the empirical rank distribution in the absence of feedback has been little investigated yet, making the whole performance of the FEC approach unknown. To close this gap, in this paper, we present the first comprehensive study on the transmission scheduling issue for the FEC approach, aiming at optimizing the rank distribution of transfer matrices with little control overhead. We propose an efficient adaptive scheduling framework for coded SNC in lossy unicast networks. This framework is one-sided (i.e., each network node forwards the segments adaptively only according to its own state) and scalable (i.e., its buffer cost will not keep on growing when the number of input packets goes to infinity). The performance of the framework is further optimized based on a linear programming approach. Extensive numerical results show that our framework performs near-optimally with respect to the empirical rank distribution.

Index Terms—Random linear network coding, segmented network coding, forward error correction, scheduling strategy.

1 INTRODUCTION

1.1 Background

Random linear network coding (RLNC) [1]–[4] is a new paradigm for data dissemination over lossy communication networks. With RLNC, all participating network nodes keep all packets received so far in their buffer, and forward linear combinations of these packets with random coefficients. Due to its random nature, RLNC is rateless and can be easily implemented in a distributed fashion. Moreover, it is asymptotically capacity-achieving for networks with packet loss in a wide range of scenarios [5]–[7].

Despite these attractive properties, one major issue in implementing RLNC is its high computational complexities. Consider the dissemination of $k$ input packets, each consisting of $L$ symbols from a finite field. RLNC requires $\mathcal{O}(k^2 L)$ and $\mathcal{O}(k^3 + k^2 L)$ finite field operations in encoding and decoding all $k$ packets, respectively, both of which could be prohibitive for common network devices [8]. Besides, RLNC incurs a coefficient vector overhead of $k$ symbols per coded packet, which could reduce the transmission rate significantly.

As a practical variation of RLNC, segmented network coding (SNC) has been proposed to bring down the computational cost [9], [10]. By restricting RLNC operations within each segment (a.k.a. generation [9], chunk [11], class [12], and batch [13], etc.) with $m$ packets, SNC only incurs $\mathcal{O}(mL)$ and $\mathcal{O}(m^2 + mL)$ operations per packet for encoding and decoding, respectively. Therefore, if $m$ is much smaller than $k$, the computational cost and the coefficient vector overhead of SNC could be much reduced compared to RLNC.

Nevertheless, it is not an easy task to guarantee that the input packets can be decoded reliably when SNC is applied. Traditional approaches employ an automatic repeat-request (ARQ) style with feedback on the basis of segments (e.g., [14]–[16]) or innovative coded packets (e.g., [17], [18]). However, such feedbacks usually result in a degraded system performance mainly due to the non-negligible delay as well as the inherent bandwidth consumption. In particular, in some scenarios such as satellite and deep-space communications, feedbacks are not even available. One natural question is, can we design efficient SNC schemes without the need of feedback?

1.2 Related Work & Motivation

1.2.1 Random Scheduling

A direct approach to eliminate feedback for SNC is random scheduling [10], where each participating node keeps transmitting packets, each encoded over a random
segment, until the destination node can recover the whole packets. Despite its simplicity, random scheduling is one-sided, i.e., the action of each node only depends on its own state, and proven to be asymptotically capacity-achieving when $m = \Omega(\log k)$ [10]. However, when $m$ is a small constant (e.g., 32) so as to accommodate the computing capabilities of common network devices [19], the performance of random scheduling could be very poor, as will be demonstrated in this paper.

1.2.2 Forward Error Correction

Very recently, the coding theory community proposes a forward error correction (FEC) approach to guarantee the transmission reliability in the absence of feedback [11], [12]. In an FEC approach, a linear block code is applied on the input packets before partitioning the packets into segments at the source node. Such kind of SNC is referred to as coded SNC. With a coded SNC, the packets in decoded segments can help the decoding of other segments via the linear dependencies among segments (such as sharing some common packets [20], [21], parity-check constraints [22], [23], etc.), which are imposed by the linear block code. In this way, the necessity that each segment is separately decodable can be eliminated. To realize this idea, the concept of empirical rank distribution of transfer matrices of segments (see Sec. 2 for a definition) is introduced to model the transfer of segments in the absence of feedback [13] [24]. When the empirical rank distribution is known in advance, several classes of coded SNC have been developed with linear encoding and decoding complexities including EC codes [21], [24], L-chunked codes [23], BATS codes [13], etc., which have close-to-optimal decoding performance. Among these, the BATS codes, as a matrix generalization of fountain codes [25], [26], can generate an infinite number of segments, preserving the ratelessness of fountain codes and RLNC.

Despite these remarkable designs of coded SNC from an end-to-end point of view, the performance of the FEC approach remains unknown. For example, can we schedule the transmission of segments efficiently in the absence of feedback? Specifically, can the corresponding empirical rank distribution of transfer matrices be 1) exactly characterized so as to facilitate the design of close-to-optimal coded SNC, and 2) optimized with respect to the used network resources?

1.3 Our Results & Paper Organization

To the best of our knowledge, this paper presents the first comprehensive study on the transmission scheduling issue for coded SNC. We propose a simple yet very efficient scheduling framework for coded SNC in lossy unicast communication networks. Under this framework, the communication network is decomposed into multiple virtual line networks. Segments are then assigned to different virtual line networks along which segments are transmitted in a sequential manner. Each intermediate node forwards the segment probabilistically and adaptively according to the number of received innovative packets, which is further optimized via a linear programming based approach. Our framework has several salient features:

- It is one-sided. Thus the framework can be implemented easily in a distributed manner with little control overhead.
- It is scalable and stable, i.e., the buffer cost of each node will not keep on growing when the number of input packets goes to infinity.
- It performs near-optimally over a large range of network settings.

Therefore, together with existing designs of coded SNC, our framework demonstrates the promise of the FEC approach. Specifically, the input packets could be transmitted reliably with a transmission rate close to the value of the min-cut that separates the source node and the destination node.

The main contribution of the paper is summarized as follows:

- We first establish an upper bound on the efficiency of the random scheduling strategy, which theoretically demonstrates that random scheduling is inefficient when the segment size is a practical constant.
- We then propose an efficient one-sided adaptive scheduling framework, demonstrate its stability, and optimize its performance based on linear programming.
- We finally conduct extensive performance evaluations of our adaptive scheduling framework. Numerical results show that our proposed adaptive scheduling scheme performs near-optimally.

The rest of the paper is organized as follows. In Sec. 2, we introduce the basic model and the design objectives of scheduling for coded SNC. In Sec. 3, we present an analysis to the random scheduling strategy. In Sec 4, we propose our adaptive one-sided scheduling framework in the context of line networks, and in Sec. 5, we extend the framework to general unicast networks. Sec. 6 presents the extensive numerical results. Finally, Sec. 7 gives the concluding remarks.

2 THE MODEL AND DESIGN OBJECTIVES

2.1 Network Model

We consider a packet erasure network modeled as a directed graph $\mathcal{G} = (V, E)$ with a node set $V$ and an edge set $E$. As the transmission unit, each packet consists of $L$ symbols from a finite field $\mathbb{F}_q$ with size $q$, which will be regarded as a column vector in $\mathbb{F}_q^L$. Each link $(u, v) \in E$ is associated with two parameters:

- rate constraint $r_{uv} > 0$, denoting the number of packets that node $u$ can send to $v$ per time unit, and
• erasure probability $\varepsilon_{uv}$, meaning that each packet transmitted over the edge will be erased with probability $\varepsilon_{uv}$, which is independent with other packets over any links.

For simplicity, we do not take link delay into account so that each packet transmission is instantaneous. In the following, we focus on the unicast problem where a source node $s \in \mathcal{V}$ wants to send $k$ input packets, say $p_1, p_2, \ldots, p_k$, to a destination node $d \in \mathcal{V}$.

2.2 Segmented Network Coding

Definition 1 (Segmented network code). A segment is a set of packets each of which is a linear combination of the input packets, and a segmented network code (SNC) is a collection of segments. In other words, an SNC is a linear block code with symbols in the codewords separated into segments. It is worth mentioning that, by employing a linear code on the input packets before segmentation, such definition generalizes the most widely used SNC, whose segments are just disjoint subsets of input packets [14]. Same as existing work (e.g., [21], [27]), we assume that all segments in an SNC have the same cardinality $m$, which is referred to as segment size.

2.2.1 Transmission with SNC

A general use of an SNC for transmission can be described as follows. Consider an SNC with $n$ segments, say $S_1, S_2, \ldots, S_n$, each of which is also treated as a matrix formed by juxtaposing all the packets in the segment. With this SNC, each transmitted packet in the network is of the form $(j, c, p)$, where $c \in \mathbb{F}_q^m$ is the coefficient vector, and $p = S_j c$ is a linear combination of packets in $S_j$. For convenience, we refer to such a packet as a $j$-packet. When a transmission opportunity over a link $(u, v)$ arises, node $u$ will perform the following operations:

1) Select a segment $S_j$ according to some certain scheduling strategy.

2) Generate a new $j$-packet by combining all $j$-packets in hand linearly with random coefficients. Specifically, denote the $j$-packets in hand as $(j, c^i, p^i)$, $i = 1, \ldots, h$, and the new $j$-packet as $(j, c, p)$. We have

$$c = \sum_{i=1}^{h} \phi_i c^i, \quad \text{and} \quad p = \sum_{i=1}^{h} \phi_i p^i, \quad (1)$$

where $\phi_i$, $i = 1, \ldots, h$ are chosen from $\mathbb{F}_q$ independently and uniformly at random.

3) Transmit the newly generated $j$-packet to node $v$.

Note that in (1), we only need to combine linearly independent $j$-packets. So the encoding cost of a coded packet is $O(mL)$ finite field operations. Besides, when a node receives a $j$-packet that is linearly dependent with those $j$-packets in hand, it will drop the $j$-packet for saving the buffer cost.

2.2.2 Encoding & Decoding of SNC

Now we introduce some basic concepts towards characterizing the transmission of segments, which play fundamental roles in designing efficient SNC.

Definition 2 (Transfer matrix). Let $Y_j$ denote the matrix whose columns correspond to all coded packets over $S_j$ received by the destination node after the whole transmission is completed, i.e.,

$$Y_j = S_j T_j, \quad (2)$$

where $T_j$ is said to be the transfer matrix of segment $S_j$. In other words, $T_j$ is the matrix whose columns correspond to the coefficient vectors of all $j$-packets received by the destination node.

Definition 3 (Empirical rank distribution). Given a scheduling strategy, the empirical rank distribution of all transfer matrices is a probability vector $(t_0, t_1, \ldots, t_m)$, if exists, such that

$$\left| \left\{ j : \text{rk}(T_j) = i \right\} \right| \xrightarrow{p} t_i, \quad \text{for } i = 0, 1, \ldots, m$$

where $\text{rk}(\cdot)$ denotes the rank of a matrix and $\xrightarrow{p}$ denotes convergence in probability as $n$ goes to infinity. In other words, $t_i$ is the proportion of segments whose transfer matrices have rank $i$.

For a given empirical rank distribution $(t_0, t_1, \ldots, t_m)$, let

$$\bar{t} = \sum_{i=1}^{m} i t_i \quad (3)$$

be the average rank of transfer matrices. There have been efficient schemes to generate a coded SNC (e.g., [13], [23]) with the following properties.\(^2\)

- **Linear-time complexity:** both encoding and decoding operations of SNC have $O(k)$ time complexity.

- **Reliably decoding:** the destination node can recover all $k$ input packets with high probability (i.e., $1 - O(k^{-c})$ for a constant $c > 0$) after the transmission is accomplished.

- **Near-optimality:** the number of segments is $(1+\delta)k/\bar{t}$, where $\delta$ is usually a very small constant, e.g., 0.01. Note that $k/\bar{t}$ is a lower bound on the number of segments in an SNC so that all $k$ input packets can be reliably decoded [24].

In order to give a full view of how an SNC is encoded and decoded, in the following, we provide a brief introduction of BATS codes [13] as an example. Given an empirical rank distribution $(t_0, t_1, \ldots, t_m)$, a BATS code is encoded in the following way. First, a degree distribution $\Phi = (\Phi_1, \Phi_2, \ldots, \Phi_k)$ is specified according to the empirical rank distribution. Then, for the generation

\(^2\) For some schemes, e.g., [23], $t_m > 0$ is additionally required.
of a segment $S_j$, the source node first samples the distribution $\Phi$ which returns a number $D_j$ with probability $\Phi_{D_j}$, then picks a set of $D_j$ input packets, denoted by $B_j$, uniformly at random from all input packets, and finally generates the segment $S_j$ as

$$S_j = B_jG_j,$$

(4)

where $G_j$ is a $D_j \times m$ totally random matrix. Specifically, each entry of $G_j$ is chosen from $\mathbb{F}_q$ independently and uniformly at random. It can be generated by a pseudo-random number generator and thus can be recovered at the destination node by the same generator.

A BATS code can be efficiently decoded using a decoding-substitution algorithm. By combining (2) and (4), we have the linear system

$$Y_j = B_j(G_jT_j).$$

(5)

If there exists some linear system of $B_j$ decodable, then all the packets in $B_j$ can be recovered by Gaussian elimination. In particular, if $rk(G_jT_j) = D_j$, then $B_j$ can be decoded directly by solving (5). Once decoded, the values of packets in $S_j$ are then substituted into every linear system of $B_i$, $i \neq j$, where $B_i \cap B_j \neq \emptyset$, to reduce the number of unknowns in the linear system. The decoding-substitution procedure is performed iteratively until all $B_i$ are decoded or no more $B_j$ is decodable.

## 2.3 Scheduling Strategy

In this paper, we focus on the design of efficient scheduling strategies for coded SNC. The primary objective of the design includes two aspects:

1) **Exact characterization of the empirical rank distribution exactly**, which is required to facilitate the design of a proper and efficient coded SNC.

2) **Maximizing the scheduling efficiency**: Let $D_n$ denote the time cost for transmitting $n$ segments under a certain scheduling strategy. Then the efficiency of the scheduling strategy, denoted by $\Psi$, is defined as

$$\Psi = \liminf_{n \to \infty} \mathbb{E} \left[ \frac{nl}{D_n} \right],$$

(6)

where $\mathbb{E}[\cdot]$ denotes the expectation of a random variable. In short, the scheduling efficiency is just the **effective** (concerning linearly independent coded packets) end-to-end transmission throughput. This implies that the whole $k$ input packets can be successfully transmitted in $(1 + \delta + o(1))k/\Psi$ time using SNC. Evidently, the efficiency $\Psi$ is bounded above by $\text{maxflow}(s, d)$, the value of a maximum flow from $s$ to $d$ over $G$, where each link $(u, v) \in \mathcal{E}$ has a capacity $(1 - \epsilon_{uv})r_{uv}$.

From the system perspective, there are some additional requirements.

3. As demonstrated in [28], when the encoding complexity of intermediate nodes is $O(1)$, the end-to-end capacity is smaller than the value of the maximum flow. Thus, the maximum flow upper bound is not tight.

### 3 Random Scheduling is Inefficient

In this section, we demonstrate the inefficiency of random scheduling strategy [10], [30], [31] when the segment size is set as a practical value.

**Definition 4** (Random scheduling). In random scheduling strategy, when generating a coded packet for transmission, every node always picks a segment from all $n$ segments uniformly at random.

Random scheduling has the feature of ratelessness, i.e., nodes can keep on transmitting until the destination node can recover all the input packets. It has also been proved to be asymptotically optimal in terms of scheduling efficiency when $m = \Omega(\log k)$. In contrast, we will show that random scheduling performs poorly in line networks when $m$ is a small constant as required for practical use.

**Definition 5** (Line network). A network is said to be a line network of length $\Lambda$ if it is just formed by $\Lambda$ tandem links $(v_i, v_{i+1})$, $i = 0, 1, \ldots, \Lambda - 1$ where $v_0 = s$ and $v_{\Lambda-1} = d$. In this case, the terms $r_i$ and $\epsilon_i$ are shortened to $r_i$ and $\epsilon_i$, respectively.

See Fig. 1 for an illustration.

**Theorem 1.** The efficiency of random scheduling strategy in a line network of length $\Lambda$ is upper bounded by

$$\max_{\lambda > 0} \sum_{i=1}^\Lambda \left( 1 - \sum_{h=0}^{i-1} e^{-\lambda} \frac{\lambda^h}{h!} \right),$$

(8)

where $\lambda_i = (1 - \epsilon_i)r_i\lambda$. 

**Fig. 1.** A line network of length three. Node $s$ is the source node, node $d$ is the destination node, and nodes $v_1$ and $v_2$ are the intermediate nodes that do not demand the input packets.
Proof: Without loss of generality, assume that the whole transmission terminates at time $D_n = \lambda n$ for some constant $\lambda > 0$. Let $X_j^{(i)}$, $i = 0, \ldots, \Lambda - 1$, denote the whole number of $j$-packets successfully received by $v_{i+1}$. Then according to the random scheduling strategy as well as the packet erasure model, $X_j^{(i)}$ follows a binomial distribution

$$X_j^{(i)} \sim \text{Binom} \left( \lfloor \lambda nr_j \rfloor, (1 - \varepsilon_i) \cdot \frac{1}{n} \right) \text{ Pois}(\lambda_i). \quad (9)$$

A key observation is on the min-cut bound, i.e., $rk(T_j) \leq X_j^{(i)}$ for any $i = 0, 1, \ldots, \Lambda - 1$. Also, by definition, $rk(T_j) \leq m$. Therefore,

$$rk(T_j) \leq \min \left\{ X_j^{(0)}, X_j^{(1)}, \ldots, X_j^{(\Lambda - 1)}, m \right\}. \quad (10)$$

Due to the symmetry of random scheduling over all segments, $rk(T_j)$, $j = 1, \ldots, n$ are identically distributed. By the linearity of expectation, we now have

$$i = \frac{1}{n} E \left[ \sum_{j=1}^{n} rk(T_j) \right]$$

$$= E [rk(T_j)]$$

$$\leq E \left[ \min \left\{ X_j^{(0)}, X_j^{(1)}, \ldots, X_j^{(\Lambda - 1)}, m \right\} \right]$$

$$= \sum_{l=1}^{m} \text{Pr} \left\{ \min \left\{ X_j^{(0)}, X_j^{(1)}, \ldots, X_j^{(\Lambda - 1)}, m \right\} \geq l \right\}$$

$$= \sum_{l=1}^{m} \text{Pr} \left\{ X_j^{(0)} \geq l, X_j^{(1)} \geq l, \ldots, X_j^{(\Lambda - 1)} \geq l \right\}$$

$$= \sum_{l=1}^{m} \prod_{i=0}^{\Lambda - 1} \left( 1 - \sum_{h=0}^{l-1} \frac{e^{-\lambda_i} \lambda_i^h}{h!} \right),$$

where (a) follows according to the formula for computing the expectation of a non-negative integer-valued random variable [32], (b) follows since under random scheduling, $X_j^{(i)}$ are mutually independent, and (c) follows according to Eq. (9). Therefore, no matter how the termination time $D_n$ is chosen, the upper bound given in (8) must hold.

According to Theorem 1, we can show that random scheduling is inefficient when $m$ is small (see the numerical results in Sec. 6). The underlying reason is that, under random scheduling, $X_j^{(i)}$, $i = 0, 1, \ldots, \Lambda - 1$, are independent. Hence, it is very likely that some $X_j^{(i)}$ is small, leading to a low rank of transfer matrix by (10).4 Besides, the upper bound given in Theorem 1 is inevitably untight as the min-cut bound in (10) cannot always be achieved. Take Fig. 1 as an example. We suppose that node $v_1$ received exactly two $j$-packets at time $b_1$ and $b_2$, and node $v_2$ also received two $j$-packets at time $b_3$ and $b_4$, where $b_1 < b_3 < b_4 < b_2$. In this case, the two $j$-packets received by $v_2$ must be linearly dependent. Hence, $rk(T_j) \leq 1$, although the min-cut bound is 2.

In the next section, we propose an efficient scheduling strategy, which can eliminate the drawbacks of random scheduling by

1) transmitting the segments in a sequential manner to reduce the transmissions of linearly dependent packets, and
2) transmitting the segments adaptively according to the number of successful transmissions over the previous link so that the numbers of successful transmissions over different links could be balanced.

4 ADAPTIVE SCHEDULING: LINE NETWORK

In this section, we first introduce an adaptive scheduling framework with desired properties in the context of line networks, and then optimize its performance via linear programming. The extension of the framework to general unicast networks is referred to the next section.

Let $G$ be a line network with length $\Lambda$. In the adaptive scheduling framework, the $n$ segments $S_1, S_2, \ldots, S_n$ are transmitted along the network in a sequential manner according to the order of their indices. For each segment $S_j$, $j = 1, \ldots, n$, the framework works as follows.

- Source node $v_0$: it generates and forwards $M_j^{(0)}$ $j$-packets to node $v_1$, where $M_j^{(0)}$ is an integer-valued random variable. Once finished, it starts the transmission of next segment $S_{j+1}$ if $j < n$, or sends an ending signal to node $v_1$ if $j = n$, which is assumed to be reliable.

- Intermediate node $v_i$, $i = 1, 2, \ldots, \Lambda - 1$: it starts the transmission of $S_j$ if both of the following conditions are satisfied:
  - it has completed the transmission of segment $S_{i-1}$, and also
  - it has received some $j'$-packet with $j' > j$, or the ending signal from its preceding node $v_{i-1}$, or has received $m$ linearly independent $j$-packets.

Then it generates and forwards $M_j^{(i)}$ $j$-packets to node $v_{i+1}$, where $M_j^{(i)}$ is an integer-valued random variable chosen adaptively according to the $j$-packets received from node $i - 1$ (see Sec. 4.2). Once $v_i$ finishes the transmission of $S_j$, it will drop all $j$-packets in hand. If $j = n$, it will also send an ending signal to node $v_{i+1}$, which is assumed to be reliable.

For the setting of random variables $M_j^{(i)}$, the framework introduces two parameters, a real number $\bar{M} > 0$ and an arbitrarily small constant $0 < \Delta < 1$, such that all $M_j^{(i)}$, $j = 1, 2, \ldots, n$, are independent and identically distributed (i.i.d.), and

$$E \left[ M_j^{(i)} \right] = \begin{cases} \bar{M} & i = 0 \\ (1 - \Delta) \bar{M} \cdot \frac{\bar{M}}{\rho_0} & i > 0 \end{cases}$$

(11)
Besides, there is a maximum integer \( \hat{M}^{(i)} \) that \( M_j^{(i)} \) can take. Before optimizing these random variables, we first demonstrate the basic properties of the framework.

### 4.1 Properties

A direct fact about the scheduling framework is as follows.

**Fact 2.** Under the adaptive scheduling framework, the transmissions of all segments are stochastically the same.

Now we show that the line network is stable under the adaptive scheduling framework, regardless of the probability distributions of \( M^{(i)}, i = 0, 1, \ldots, \Lambda - 1 \), as long as Eq. (11) holds.

**Theorem 3.** The line network \( G \) is stable under the adaptive scheduling framework.

**Proof:** In order to prove the theorem, we model the buffer of each intermediate node \( v_i \), \( i = 1, 2, \ldots, \Lambda - 1 \), as a \( G/G/1 \) queueing system: each segment is viewed as a customer; we say a customer \( S_j \) enters the queue of \( v_i \) if the transmission of \( S_{j-1} \) from node \( v_{i-1} \) is finished, and leaves the queue of \( v_i \) if \( v_i \) completes the transmission of \( S_j \). Thus, the network forms a system of queues in series. Let the inter-arrival time sequence of the first queue of node \( v_1 \) be \( \{ T_j^{(1)} \} \) and the service time sequence of the queue of node \( v_i \) as \( \{ S_j^{(i)} \} \), both indexed by \( j \). According to Fact 2, the sequence \( \{ T_j^{(1)}, S_j^{(2)}, S_j^{(3)}, \ldots, S_j^{(\Lambda - 1)} \} \) forms an i.i.d. sequence. Therefore, this sequence is a strictly stationary metrically transitive sequence [33]. Moreover, for any \( i = 1, 2, \ldots, \Lambda - 1 \),

\[
E\left[ S_j^{(i)} \right] = E\left[ M_j^{(i)}/r_i \right] = E\left[ M_j^{(i)} \right]/r_i = (1 - \Delta)\bar{M}/r_0,
\]

and

\[
E\left[ T_j^{(1)} \right] = \bar{M}/r_0,
\]

implying

\[
E\left[ S_j^{(i)} \right] < E\left[ T_j^{(1)} \right].
\]

Therefore, by Loynes’ theorem [29, Thm. 7], we conclude that the system of queues in series is stable, i.e., the line network is stable.

**Corollary 4.** The time cost \( D_n \) for transmitting all \( n \) segments satisfies \( D_n/n \xrightarrow{p} \bar{M}/r_0 \).

**Proof:** Let \( D^{(i)} \) be the length of time duration between the starting time and the time that node \( i \) completes the transmission of all \( n \) segments. Then, we have \( D_n = D^{(\Lambda - 1)} \). By the weak law of large numbers, \( D^{(i)}/n \xrightarrow{p} \bar{M}/r_0 \) is achieved. Let \( B^{(i)} \) denote the number of segments that have not been transmitted by node \( v_i \) when node \( v_{i-1} \) finishes the transmission of all \( n \) segments. Then for each \( i = 1, 2, \ldots, \Lambda - 1 \), we have

\[
D^{(i)} - D^{(i-1)} \leq M^{(i)}B^{(i)}/r_i.
\]

Since \( B^{(i)} < \infty \) when \( n \to \infty \) according to Theorem 3, the above inequality leads to \( (D^{(i)} - D^{(i-1)})/n \xrightarrow{p} 0 \), or equivalently, \( D^{(i)}/n \xrightarrow{p} D^{(i-1)}/n \). Finally, we have

\[
D^{(\Lambda - 1)}/n \xrightarrow{p} \bar{M}/r_0.
\]

**Corollary 5.** The efficiency of the scheduling framework is given by \( E[\text{rk}(T_j)]/\bar{M} \).

**Proof:** According to the definition of scheduling efficiency, this result follows from Fact 2 and Corollary 4 directly.

**Remark 1.** When the buffer size of each node is bounded by a certain constant, our adaptive scheduling framework is still applicable if the rateless BATS codes are used. Specifically, we assume that there can be an upper bound to the number of segments arrived but unprocessed in node \( i \) at any time. When the buffer is full and coded packets of other segments \( S_j \) arrive, node \( i \) will discard all \( j \)-packets and skip the transmission of \( S_j \). In this way, the rank distribution of the transfer matrix of a segment that has arrived at the destination node successfully will not be changed due to any buffer overflow, and hence the generation of BATS codes will not be affected. Also, due to the ratelessness of BATS codes, once any of \( n \) segments arrived at the destination node successfully, all the input packets can be recovered reliably, albeit the scheduling efficiency under such finite buffer constraint requires to be further investigated.

### 4.2 Performance Optimization

Now we optimize the performance of the framework by optimizing the parameters involved.

According to Corollary 5, when \( \bar{M} \) is fixed, the scheduling efficiency could be optimized by maximizing \( E[\text{rk}(T_j)] \) for any \( j \). To simplify the notation, in this subsection, we omit the subscript of \( j \). Let \( T^{(i)} \) denote the matrix formed by the coefficient vectors of all \( j \)-packets received by node \( v_i \) where \( T^{(0)} \) is an \( m \times m \) identity matrix, and \( T^{(\Lambda)} = I \). In the following, we will maximize \( E[\text{rk}(T)] \) by optimizing \( P_{M^{(i)}|\text{rk}(T^{(i)})} \).

We employ a hop-by-hop approach for optimization, i.e., given \( p_{\text{rk}(T^{(i)})} \), the distribution of \( \text{rk}(T^{(i)}) \), we maximize \( E[\text{rk}(T^{(i+1)})] \) by optimizing \( P_{M^{(i)}|\text{rk}(T^{(i)})} \). As a byproduct of the solution, \( p_{\text{rk}(T^{(i+1)})} \) will be used to optimize the next hop transmission. The flow chart of the whole optimization is illustrated in Fig. 2.

Now we focus on the maximization of \( E[\text{rk}(T^{(i+1)})] \) by choosing a proper \( P_{M^{(i)}|\text{rk}(T^{(i)})} \). To establish the relationship between \( p_{\text{rk}(T^{(i+1)})} \) and \( P_{M^{(i)}|\text{rk}(T^{(i)})} \) as well as \( p_{\text{rk}(T^{(i)})} \), we first introduce some notations.

Define

\[
\zeta_{a,b,h}^{b,h} = \frac{\zeta_{a,b,h}^{b,h}}{s_a^{b,h}} = \frac{\zeta_{a,b,h}^{b,h}}{s_a^{b,h}}.
\]

where

\[
s_a^{b,h} = \begin{cases} (1 - q^{-b})(1 - q^{-b+1}) \cdots (1 - q^{-b+a-1}) & a > 0, \\ 1 & a = 0. \end{cases}
\]
Then we have the following result.

**Lemma 6.** For any \( a = 0, 1, \ldots, m, \)

\[
p_{rk(T^{(i+1)})}(a) = \sum_{b=0}^{m} \sum_{w=0}^{M^{(i)}} \sum_{h=0}^{w} p_{M^{(i)}|rk(T^{(i)})}(w|b)p_{rk(T^{(i)})}(b) \cdot p_{M^{(i)}|rk(T^{(i)})}(w|b) \cdot a_p^{b,h} \left( \frac{w}{h} \right) (1 - \varepsilon_i)^h (\varepsilon_i)^{w-h}.
\]

(12)

**Proof:** According to the encoding operations of SNC at node \( v_i \) and the packet erasure model, we have

\[
T^{(i+1)} = T^{(i)} R^{(i)} E^{(i)},
\]

(13)

where

- \( R^{(i)} \) is a totally random matrix with \( M^{(i)} \) columns that corresponds to the encoding operations at node \( v_i \) and
- \( E^{(i)} \) is an \( M^{(i)} \times M^{(i)} \) random diagonal matrix with independent diagonal entries, where each diagonal entry is 0 with probability \( \varepsilon_i \) and 1 with probability \( 1 - \varepsilon_i \), modeling the random erasures over link \( (v_i, v_{i+1}) \).

Clearly,

\[
Pr\{rk(E^{(i)}) = h|M^{(i)} = w\} = \left( \frac{w}{h} \right) (1 - \varepsilon_i)^h (\varepsilon_i)^{w-h}.
\]

Also, according to [13], we have

\[
Pr\{rk(T^{(i+1)}) = a|rk(T^{(i)}) = b, rk(E^{(i)}) = h\} = \zeta^a_{b,h}.
\]

Therefore, by the law of total probability, we have, for any \( a = 0, 1, \ldots, m, \)

\[
p_{rk(T^{(i+1)})}(a) = \sum_{b=0}^{m} \sum_{w=0}^{M^{(i)}} \sum_{h=0}^{w} p_{M^{(i)}|rk(T^{(i)})}(w|b)p_{rk(T^{(i)})}(b) \cdot p_{M^{(i)}|rk(T^{(i)})}(w|b) \cdot a_p^{b,h} \left( \frac{w}{h} \right) (1 - \varepsilon_i)^h (\varepsilon_i)^{w-h}.
\]

(12)

Besides,

\[
\mathbb{E} \left[ M^{(i)} \right] = \sum_{b=0}^{M^{(i)}} \sum_{w=0}^{b} w p_{rk(T^{(i)})}(b)p_{M^{(i)}|rk(T^{(i)})}(w|b) = \begin{cases} M^{(i)} & i = 0 \\ (1 - \Delta)M^{(i)} \cdot \frac{\varepsilon_i}{\hat{\varepsilon}_i} & i > 0. \end{cases}
\]

(15)

Both Eq. (14) and Eq. (15) are linear functions of \( p_{M^{(i)}|rk(T^{(i)})}(w|b) \). So we establish the following linear programming (LP) formulation to maximize \( \mathbb{E}[rk(T^{(i+1)})] \):

maximize Eq.(14)

subject to Eq.(15)

\[
\sum_{w=0}^{M^{(i)}} p_{M^{(i)}|rk(T^{(i)})}(w|b) = 1, \quad b = 0, \ldots, m
\]

\[
p_{M^{(i)}|rk(T^{(i)})}(w|b) \geq 0, \quad w = 0, \ldots, M^{(i)}, \quad b = 0, \ldots, m.
\]

The above LP could be efficiently solved by the simplex algorithm or the interior point method [34], which also gives \( p_{rk(T^{(i+1)})} \) as a by-product according to Lemma 6. Note that in the LP, setting a larger \( M^{(i)} \) may achieve a higher objective, but also increase the computational cost. In practice, we can set \( M^{(i)} \) to be \[ \left\lfloor \frac{2m}{1 - \varepsilon_i} \right\rfloor \], because, when \( M^{(i)} \) \( j \)-packets are transmitted,
the probability that node $v_{i+1}$ receives at least $m$ $j$-packets is very close to 1, i.e., the probability
\[
\Pr \left\{ \text{rk} \left( T^{(i+1)} \right) = \text{rk} \left( T^{(i)} \right) \left| M^{(i)} = \left\lceil \frac{2m}{1-\varepsilon_i} \right\rceil \right\} \right\}
\]
is very close to 1.

For each fixed $M$, we can use the above hop-by-hop optimization to get the scheduling efficiency $E_2(\text{rk}(T^{(M)}))/M$. So we can pick the optimal value of $M$ to maximize the scheduling efficiency. To be demonstrated in Sec. 6, when $M$ grows from $m$, the efficiency increases firstly, and then decreases afterwards. Therefore, the optimal $M$ could be efficiently found by the golden section search method [35].

5 ADAPTIVE SCHEDULING: GENERAL NETWORKS

In this section, we extend the adaptive scheduling framework for line networks to general unicast networks. The basic idea consists of two steps:

1) first decompose the unicast network into multiple virtual line networks such that the sum of the values of maximum flows in virtual line networks is equal to the value of maximum flow in the original unicast network;
2) then apply the adaptive scheduling strategy to each virtual line network.

In the following, we show how to do the two steps in details.

5.1 Graph Decomposition

The graph decomposition consists of two main steps: 1) find a maximum flow of $G$, and 2) decompose the flow into multiple sub-flows over line networks. For the first step, there have been many efficient algorithms to find a maximum flow; see [36] for a new and excellent survey. In each iteration of the second step, we find a flow path from $s$ to $d$, and subtract the corresponding sub-flow from the whole flow. The details of the second step are provided in Alg. 1.

**Theorem 7.** Alg. 1 is well defined and always terminates with all found paths $P_1, P_2, \ldots, P_g$ satisfying
\[
\sum_{i=1}^{g} C_{P_i} = \text{maxflow}(s, d).
\]

**Proof:** During each iteration (step 4-step 12), a sub-flow corresponding to a path $P$ with value $C_{P}$ is subtracted from $f$. Therefore, one invariant during the iteration is that $f$ is always a flow. In particular, when the value of flow $f$ is equal to 0, $f(u, v) = 0$ holds for any $(u, v) \in \mathcal{E}$, implying $\mathcal{E}' = \emptyset$. Therefore, if $\mathcal{E}' = \emptyset$, then the value of $f$ must be larger than 0, and thus there must be a path from $s$ to $d$ in $\mathcal{G}'$. So the algorithm is well defined. Furthermore, in each iteration, there must be some edges deleted from $\mathcal{E}'$, and therefore the algorithm always terminates. The result of (16) is straightforward. □

**Theorem 8.** Alg. 1 can run in $O(|\mathcal{V}||\mathcal{E}|^2)$ time.

**Proof:** If we use the well-known EdmondsCKarp algorithm to find a maximum flow, then steps 1-2 cost $O(|\mathcal{V}||\mathcal{E}|^2)$ time. For each iteration (step 4-step 12), the run time is dominated by the BFS which costs $O(|\mathcal{V}| + |\mathcal{E}|)$. Since the number of iteration is at most $|\mathcal{E}|$, therefore, the whole time cost of iterations is $O((|\mathcal{V}| + |\mathcal{E}|)|\mathcal{E}|)$. By summing up the costs, we complete the proof. □

5.2 Scheduling in Virtual Line Networks

For each virtual line network $P_i$, we associate each edge $(u, v) \in P_i$ with a rate constraint $C_{P_i} \sum_{j \in \mathcal{E}(u, v) \in P_i} r_{ij}^e$ and an erasure probability $\varepsilon_{uv}$. Such virtualization could be realized by the following scheme: when a transmission opportunity over a link $(u, v)$ arises, it performs the transmission task of $P_i$ with probability $\sum_{j \in \mathcal{E}(u, v) \in P_i} C_{P_i} \varepsilon_{ij}$. Now for each virtual line network $P_i$, we can find its optimized adaptive scheduling strategy given in Sec. 4. Let $r_{0i}^{(i)}$ be the rate constraint of the first link in $P_i$ and $M_i$ be the value of $M$ for $P_i$. Then we assign $n_i$ segments to each $P_i$ such that $n_i \propto r_{0i}^{(i)}/M_i$, i.e.,
\[
n_i = \frac{r_{0i}^{(i)}/M_i}{\sum_{j=1}^{g} C_{P_j} \varepsilon_{ij}} n_i.
\]

Evidently, this could be directly done in a deterministic way. Alternatively, this could also be done stochastically by assigning each segment to a line network $P_i$ with probability $r_{0i}^{(i)}/M_i / \sum_{j=1}^{g} r_{0j}^{(j)}/M_j$. The latter approach preserves the i.i.d. property of all the rank distributions of transfer matrices, which may provide some more convenience for SNC design [24].
Theorem 9. Under the above scheduling strategy, the unicast network \( G \) is stable.

Proof: The stability result is inherited from Theorem 3 for line network, as each node in \( G \) can only belong to a finite number of virtual line networks. \( \square \)

Theorem 10. The proposed scheduling scheme satisfies \( \Psi = \sum_{i=1}^{g} \Psi_i \), where \( \Psi_i \) and \( \Psi \) denote the scheduling efficiency over the virtual line network \( P_i \), and the general unicast network \( G \), respectively.

Proof: Let \( D^{(i)} \) denote the time cost for transmission over virtual line network \( P_i \). Then according to Corollary 4, we have

\[
\frac{D^{(i)}}{n} = \frac{r_0^{(i)} / \bar{M}_i}{\sum_{j=1}^{g} r_0^{(j)} / M_j} \cdot \frac{D^{(i)}}{n} = \frac{1}{\sum_{j=1}^{g} r_0^{(j)} / M_j}. \tag{17}
\]

This implies that the transmissions over different virtual line networks terminate at almost the same time. In other words, the whole time cost \( D_n = \max\{D^{(1)}, D^{(2)}, \ldots, D^{(g)}\} \) satisfies \( D_n/n \to 1/\sum_{j=1}^{g} r_0^{(j)} / M_j \). Based on this property, it is straightforward to show \( \Psi = \sum_{i=1}^{g} \Psi_i \) by the definition of scheduling efficiency and Corollary 5. \( \square \)

From Theorem 7 and Theorem 10, we can see that if the scheduling efficiency over each line network is near-optimal, i.e., \( \Psi_i \) is very close to \( \bar{C}_{P_i} \), then the total scheduling efficiency over the original network is also near-optimal.

6 Numerical Evaluations

In this section, we evaluate the proposed adaptive scheduling framework numerically. In order to show the performance of our framework is near-optimal, it is sufficient to evaluate the performance only in line networks according to Theorem 10. Throughout the evaluations, we always set \( q = 2^8 \), a mostly used size of the finite field, and set \( \Delta \) to be 0.001. Also, without loss of generality, we set the capacities of the links to be equal by first picking erasure probabilities artificially, and then obtaining the corresponding rate constraints. Otherwise, the relative gap between the transmission efficiency and the upper bound (i.e., maxflow(s, d)) can only be smaller.

To evaluate the effect of different parameters, we start with the setting that all the links have the same erasure probability \( \varepsilon \) and rate constraint 1. We first evaluate the effect of network length \( \Lambda \) and the erasure probability \( \varepsilon \) and compare our adaptive scheduling (AS) scheme against random scheduling (RS) with \( m = 32 \). The scheduling efficiencies are shown in Fig. 3. From this figure, we observe the followings:

- Adaptive scheduling can achieve significantly higher efficiency than random scheduling under the same setting.
- When the network length grows, the efficiency of both scheduling schemes decreases, but adaptive scheduling has a slower decreasing speed. Moreover, the efficiency of adaptive scheduling keeps close to the max-flow bound \( 1 - \varepsilon \).

- When the erasure probability becomes larger, the efficiency decreasing speed of adaptive scheduling with respect to network length also becomes larger. Here is an intuitive explanation. When the network length is fixed, \( \bar{M} \cdot (1 - \varepsilon) \) is usually very close for different \( \varepsilon \). Here \( \bar{M} \) is the optimal value of \( M \) (cf. Fig 5). Now consider \( M \) transmissions over a link with erasure probability \( \varepsilon \). When \( M \cdot (1 - \varepsilon) \) is fixed, the variance of the number of successful transmissions, i.e., \( M \varepsilon \cdot (1 - \varepsilon), \) becomes larger when \( \varepsilon \) increases. This is harmful for optimizing the transmission scheme over next links.

We then evaluate the effect of segment size. The results for \( m = 16, 32, 64 \) are plotted in Fig. 4. We observe that when a larger segment size is used, adaptive scheduling can achieve a higher efficiency and have a slower efficiency decreasing speed with respect to network length. The reason is that, when \( m \) is larger, more benefits of network coding can be gained.

We also evaluate the effects of \( \bar{M} \). The relationship between \( \bar{M} \) and the scheduling efficiency is plotted in Fig. 5 in a line network of length 5 under different erasure probabilities. From this figure, we observe the followings:

- The scheduling efficiency increases smoothly when \( \bar{M} \) increases from a small number, and then decreases very fast after \( \bar{M} \) reaches an optimal value, implying that the optimal \( \bar{M} \) could be efficiently found using golden section search method. At the increasing stage, larger \( \bar{M} \) can increase the average number of linearly independent coded packets received by intermediate nodes, which is helpful to

5. In [28], it is also demonstrated that under the encoding cost constraint of \( \mathcal{O}(1) \) at intermediate nodes, the end-to-end capacity becomes smaller when the line network becomes longer.
gain more benefits of network coding over later links. However, when $\bar{M}$ becomes much larger, there could be a great probability that more than $m$ packets of a segment can be successfully transmitted over a link, leading to a waste of transmissions.

- When the erasure probability $\varepsilon$ becomes larger, the optimal $\bar{M}^*$ also becomes larger. But $\bar{M}^*(1 - \varepsilon)$ for different erasure probabilities are very close.

We also plot the relationship between $\bar{M}$ and the scheduling efficiency in line networks with different lengths but the same erasure probability $\varepsilon = 0.2$ in Fig. 6. We find that the optimal value of $\bar{M}$ becomes larger when the network length is larger.

Finally, we evaluate the performance of adaptive scheduling when the links have random erasure probabilities. We set $m = 32$ and $\Lambda = 2, 6, 10$. For each setting, we generate 100 random instances, and in each instance, the erasure probability of each link is chosen from $(0, 0.5)$ independently and uniformly at random. We plot the empirical cumulative distribution function (eCDF) of the ratio between the transmission efficiency and the max-flow upper bound in Fig. 7. From this figure, we find that our adaptive scheduling scheme performs close to optimal for each instance. It can achieve a median ratio of 0.96 and 0.88 when $\Lambda = 2$ and $\Lambda = 10$, respectively.

In summary, our framework performs very close-to-optimal over a large range of network settings. Together with the existing efficient SNC designs, this demonstrates the promise of the FEC approach for SNC.

7 CONCLUDING REMARKS

In this paper, we presented a comprehensive study on the design of scheduling strategy for coded SNC. We first showed that random scheduling is inefficient when the segment size is a practical constant. We then proposed an efficient one-sided adaptive scheduling framework over lossy unicast networks, and demonstrated its supreme performance via extensive numerical evaluations.

There are several directions to be further explored. One direction is to generalize the adaptive scheduling framework to networks with broadcast links (e.g.,
wireless networks). In such networks, the optimization becomes much harder. For instance, a node can send a packet to two nodes, \( u \) and \( v \), simultaneously. When node \( u \) receives the packet, it does not know whether \( v \) also gets the packet, making its decision hard. Another direction is to generalize existing results for unicast networks to multicast networks. Different from unicast networks, it is even unknown how to guide the design of scheduling strategy in multicast networks as there is still no practical and efficient SNC that can achieve a close-to-optimal decoding performance for every destination node.

**REFERENCES**


