On Streaming Codes With Unequal Error Protection

Mahdi Haghifam, M. Nikhil Krishnan, Ashish Khisti, *Member, IEEE*, Xiaoqing Zhu, *Senior Member, IEEE*, Wai-Tian Tan, *Senior Member, IEEE*, and John Apostolopoulos, *Fellow, IEEE*

***Abstract*—Error control codes for real-time interactive applications such as audio and video streaming must operate under strict delay constraints and be resilient to burst losses. Previous works have characterized optimal streaming codes that guarantee perfect and timely recovery of all source packets when the burst loss is below a certain maximum threshold. In this work, we generalize the notion of streaming codes to the unequal error protection (UEP) setting. Toward this end, we define two natural notions of streaming codes; symbol-level UEP and packet-level UEP. In the symbol-level UEP, the symbols within each source packet have varying recoverability requirements. In the packet-level UEP scenario, packets at even time slots and odd time slots have different recovery guarantees. We discuss practical motivations for both settings and develop coding schemes. We establish optimality or near-optimality guarantees through information-theoretic converse bounds. Simulations over Gilbert and Fritchman channels show that our coding schemes outperform baseline schemes over a wide range of channel parameters.**

***Index Terms*—Streaming codes, low-latency communication, unequal error protection.**

I. Introduction

REAL-TIME streaming applications such as interactive audio/video conferencing, online gaming, and augmented reality require high reliability, low latency, and preferably in-order delivery of source packets. In this setup, the encoder must operate on a stream of source packets, in a sequential fashion. Likewise, the decoder must attempt to reconstruct the source packets in-order and by their playback deadlines. There are two main error correction approaches to combat packet losses in communication networks; Automatic Repeat Request (ARQ) and Forward Error Correction (FEC). ARQ is inherently inferior when considering low-latency constraints, especially for long distance communication. For that reason, FEC schemes are considered more appropriate candidates in applications such as interactive voice/video communication and multi-player gaming.

In designing FEC for streaming applications, both fundamental limits and coding schemes can be considerably different from classical systems, which do not have to satisfy low-latency constraints. A new class of error correction codes for streaming applications is introduced in [2]. The encoder observes a semi-infinite source stream — one source packet is revealed in each time slot — and maps it to a coded output stream of rate R. The channel is modeled as a burst-erasure channel, i.e., starting at an arbitrary time, it introduces an erasure-burst of maximum length B. The decoder is required to reconstruct each source packet with a maximum delay T. In [2], a fundamental relationship between R, B and T is established and a novel optimal construction is proposed. We will refer to this family of codes as *streaming codes* in this work. It is worth mentioning that random linear combinations, popularly used in, e.g., network coding, do not attain the optimal performance. In [2]–[5], streaming codes are extended to channels with both burst and isolated erasures. We refer the reader to [6]–[10] and the references therein for various extensions of these works.

To the best of our knowledge, all prior works on streaming codes assume that the source packets are of equal importance. The code constructions discussed in these prior works guarantee perfect recovery of all the source packets when the burst length is below a certain predetermined threshold. However, these constructions fail to recover when the length of the erasure burst exceeds this threshold. In communication channels, when the length of erasure burst is random, such codes suffer from cliff effect as shown in [3]–[5]. In this work, we consider an extension of the streaming codes framework, which allows for unequal error protection (UEP). Inspired by the real-world applications, we introduce two different notions of UEP. The first notion is *symbol-level UEP*, where we assume that each source packet consists of two sub-packets. While both sub-packets have the same recovery deadline, the high-priority sub-packet should have a higher level of error protection. Particularly, we require both sub-packets to be recovered within a delay of T when the length of the erasure burst is no greater than a certain threshold, say B_L. If the length of the erasure burst is greater than B_L, but less than a higher threshold, say $B_I > B_L$, only the high-priority sub-packets need to be recovered within a delay of T. The second notion is *packet-level UEP*. Here we assume that when the burst is below a nominal threshold B_L, all source packets must

Manuscript received May 1, 2021; revised September 4, 2021; accepted November 3, 2021. Date of publication November 9, 2021; date of current version December 23, 2021. This article was presented in part at the 29th Biennial Symposium on Communications (BSC) [1]. *(Mahdi Haghifam and M. Nikhil Krishnan contributed equally to this work.) (Corresponding author: M. Nikhil Krishnan.)*

Mahdi Haghifam and Ashish Khisti are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: mahdi.haghifam@mail.utoronto.ca; akhisti@ece.utoronto.ca).

M. Nikhil Krishnan was with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada. He is now with the International Institute of Information Technology Bangalore, Bengaluru 560100, India (e-mail: nikhilkrishnan.m@gmail.com).

Xiaoqing Zhu, Wai-Tian Tan, and John Apostolopoulos are with Cisco Systems, San Jose, CA 95134 USA (e-mail: xiaoqzhu@cisco.com; dtan2@cisco.com; john.apostolopoulos@gmail.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/JSAIT.2021.3126687, provided by the authors.

Digital Object Identifier 10.1109/JSAIT.2021.3126687

be recovered within a delay of T. When the burst is greater than B_L, but less than B_I, we require that only the packets in even time slots must be recovered, while the packets in odd time slots need not be recovered.

Both symbol-level and packet-level UEP have natural applications. In scalable video coding [11], base layer maps to high-priority sub-packets, while the refinement layer maps to low-priority sub-packets (aligns with the symbol-level UEP setting). The availability of base layer will lead to reconstruction of a lower quality video when the channel introduces a longer erasure burst, whereas the availability of both base and refinement layers will lead to reconstruction of higher quality video. Similarly, for the Internet, packet headers are more important and need better protection to ensure that the actual data gets through. Thus, even though the final objective is delivering the payload data, the physical layer should provide a better protection to such protocol information. As a motivation for packet-level UEP, consider multiple description coding of video, where the frames at even times are compressed as reference I frames while the frames at odd times are compressed using predictive coding as P frames. In the presence of long burst losses, a packet-level UEP will recover I frames to yield a baseline reconstruction, while during periods of nominal burst losses both I and P frames will be recovered.

There has been a long-standing interest in the area of unequal error protection. The simplest approach to UEP is just allocating separate channels for different kinds of data. However, in many systems, we have to use the same underlying channel for transmission. The literature on UEP dates back to at least 1958 [12] and linear codes for UEP were proposed by Masnick and Wolf [13]. In recent literature, a scheme called Priority Encoded Transmission was proposed in [14] for Internet-type channels. We note that these prior works deal with block codes. The authors of [15] consider the problem of designing UEP codes, in the context of video streaming. The setup in [15] is different from what we pursue in this work. For instance, in [15], there is no strict delay requirement, and the code is based on Maximum Distance Separable (MDS) codes. To our knowledge, the present paper is the first work that studies UEP streaming codes.

A. Organization of the Present Paper

We begin with formally defining the problem setup in Section II. In Section III, we propose an achievable scheme for the symbol-level UEP setup, which is applicable for all parameters. Using an information-theoretic converse bound, we conclude our scheme is optimal in the sense that it attains the maximum achievable rate of any symbol-level UEP streaming code. Our code construction is *explicit*, and the required field size for our construction is *two*.

In Section IV, we present our results for the packet-level UEP setup. We provide a code construction for all parameters. Using an information-theoretic converse, we show that the proposed construction is near-optimal and achieves within one unit of the minimum possible delay. We also propose code constructions, which are optimal for specific parameter regimes. Again, all these constructions are *explicit*, and require only a field size of *two*.

In Section V, we present simulation results which indicate that the newly proposed streaming code constructions outperform existing ones for some instances of Gilbert and Fritchman channels.

B. Notation

The set of non-negative integers is denoted by $\mathbb{Z}_+$. The set of natural numbers $\{1, 2, 3, \ldots\}$ is given by $\mathbb{N}$. All the elements of any matrix considered in this paper are drawn from a finite field $\mathbb{F}_q$ with q elements, where 0 and 1 denote the additive and the multiplicative identities, respectively. For all the code constructions that we propose, we have $q = 2$. The set of k-dimensional vectors over $\mathbb{F}_q$ is denoted by $\mathbb{F}_q^k$, and the set of $k \times n$ matrices over $\mathbb{F}_q$ is denoted by $\mathbb{F}_q^{k\times n}$. For $a, b \in \mathbb{Z}_+$, we use $[a : b]$ to denote $\{i \in \mathbb{Z}_+ \mid a \leq i \leq b\}$. Similarly, $[a : b) \triangleq \{i \in \mathbb{Z}_+ \mid a \leq i < b\}$. We reserve boldface, upper case letters to denote matrices; e.g., $\mathbf{G}$, $\mathbf{I}_n$. We use underbars to denote column vectors; for instance, $\underline{s}(t), \underline{x}(t)$.

II. Problem Setting, Preliminaries and Main Results

A. Problem Setting

Let k and n be integers satisfying $0 < k < n$. Consider a scenario that in each (discrete) time-t, a source packet $\underline{s}(t) \triangleq [s_0(t)\ s_1(t)\ \cdots\ s_{k-1}(t)]^\top \in \mathbb{F}_q^{k\times 1}$ arrives at the source, where $t \in [0 : \infty)$. The source has access to a set of systematic encoders $\{\mathcal{E}_t\}_{t\in[0:\infty)}$, where:

$$\mathcal{E}_t : \underbrace{\mathbb{F}_q^{k\times 1} \times \cdots \times \mathbb{F}_q^{k\times 1}}_{t+1 \text{ times}} \to \mathbb{F}_q^{n\times 1}.$$

At time-t, the *systematic* and *causal* encoder $\mathcal{E}_t$ maps $t + 1$ source packets $\{\underline{s}(i)\}_{i=0}^t$ to a coded packet $\underline{x}(t) \in \mathbb{F}_q^{n\times 1}$ of the form (up to a permutation of entries):

$$\underline{x}(t) \triangleq \left[\underline{s}(t)^\top\ \underline{p}(t)^\top\right]^\top. \tag{1}$$

Here, $\underline{p}(t) \triangleq [p_0(t)\ p_1(t)\ \cdots\ p_{n-k-1}(t)]^\top \in \mathbb{F}_q^{n-k}$ will be referred to as the parity packet at time-t. The parity packet $\underline{p}(t)$ is a deterministic function of the source packets $\{\underline{s}(i)\}_{i=0}^t$. The source transmits the coded packets to a receiver over a packet erasure channel. We do not account for propagation delay in our setting and hence the coded packets (if not erased by the channel) instantaneously reach the receiver. Let $\underline{y}(t)$ denote the received packet at time-t which is given by:

$$\underline{y}(t) = \begin{cases} *, & \text{if } \underline{x}(t) \text{ is erased,} \\ \underline{x}(t), & \text{otherwise.} \end{cases}$$

The receiver has access to a set of decoders $\{\mathcal{D}_t\}_{t\in[0:\infty)}$, where:

$$\mathcal{D}_t : \underbrace{\left(\mathbb{F}_q^{n\times 1} \cup \{*\}\right) \times \cdots \times \left(\mathbb{F}_q^{n\times 1} \cup \{*\}\right)}_{t+1+T \text{ times}} \to \mathbb{F}_q^{k\times 1}.$$

At time-$(t + T)$, the decoder $\mathcal{D}_t$ provides a deterministic estimate $\hat{\underline{s}}(t) \triangleq [\hat{s}_0(t)\ \hat{s}_1(t)\ \ldots\ \hat{s}_{k-1}(t)]^\top$ of $\underline{s}(t)$ using $\{\underline{y}(i)\}_{i=0}^{t+T}$. Note that the decoders are delay-constrained as an estimate of each $\underline{s}(t)$ has to be produced by time-$(t + T)$ (or before). Any instance of an $(\{\mathcal{E}_t\}, \{\mathcal{D}_t\})$ pair will be referred to as an

$(n, k, T)_q$ streaming code (or simply, a streaming code). As each coded packet has length n and each source packet has length k, the rate of a streaming code is naturally given by $\frac{k}{n}$.

Definition 1 (Erasure Pattern): An erasure pattern $e^\infty \triangleq \{e_i\}_{i=0}^\infty$ is a binary sequence such that $e_i = 1$ if and only if $\underline{y}(i) = *$.

Fix $i \in [0 : k-1]$ and $t \in [0 : \infty)$. Given an erasure pattern, a particular source symbol $s_i(t)$ is said to be recoverable with delay T if the decoder $\mathcal{D}_t$ is able to precisely recover $s_i(t)$ by time-$(t+T)$, i.e., the estimate $\hat{s}_i(t) = s_i(t)$. Similarly, a source packet $\underline{s}(t)$ is said to be recoverable with delay T if the estimate $\hat{\underline{s}}(t) = \underline{s}(t)$.

Definition 2 (B-Bursty Erasure Pattern): Let $B \in \mathbb{N}$. An erasure pattern e^∞ is said to be a B-bursty erasure pattern if there exists $\ell \in [0 : \infty)$ such that $\{i \mid e_i = 1\} \subseteq [\ell : \ell+B-1]$, i.e., packet erasures are confined to within *at most* B consecutive time slots. Note that we consider not just the "maximal" erasure pattern where packets in all the B consecutive time slots are erased (subsets consisting of fewer erasures are also part of the definition).

Definition 3 [(B, T)-Recoverability of a Source Symbol]: Fix $i \in [0 : k-1]$ and $t \in [0 : \infty)$. Given a streaming code, i.e., $(\{\mathcal{E}_l\}_{l\in[0:\infty)}, \{\mathcal{D}_l\}_{l\in[0:\infty)})$, the source symbol $s_i(t)$ within the source packet $\underline{s}(t)$ is said to be (B, T)-recoverable if $s_i(t)$ is recoverable with delay T in presence of any B-bursty erasure pattern.

Definition 4 [(B, T)-Recoverability of a Source Packet]: Fix $t \in [0 : \infty)$. Given a streaming code, the source packet $\underline{s}(t)$ is said to be (B, T)-recoverable if $\underline{s}(t)$ is recoverable with delay T in presence of any B-bursty erasure pattern.

In this paper, we consider two different recoverability settings. In the first setting, we assume that some symbols within each source packet are of high-priority compared to others. In the second setting, we consider certain source packets to be of high-priority than others. For the high-priority symbols (similarly, packets), we impose a superior, (B_I, T)-recoverability property whereas for the low-priority ones, we impose $(B_L \leq B_I, T)$-recoverability. We will now formally introduce the two settings pursued in this paper.

1) *(γ, B_I, B_L, T)-Symbol-Level Unequal Error Protection:* Let $0 \leq \gamma \leq 1$ be a real number such that γk is an integer, where k, as seen earlier, is the size of the source packet $\underline{s}(t)$. A streaming code is said to possess (γ, B_I, B_L, T)-symbol-level unequal error protection (UEP) if the following recoverability conditions are satisfied for all $t \in [0 : \infty)$.
   - The initial γk symbols of $\underline{s}(t)$, denoted by $\{s_0(t), s_1(t), \ldots, s_{\gamma k-1}(t)\}$, are all high-priority symbols and are (B_I, T)-recoverable.
   - The rest of the source symbols $\{s_{\gamma k}(t), s_{\gamma k+1}(t), \ldots, s_{k-1}(t)\}$ are all (B_L, T)-recoverable.

   For convenience, we will refer to a streaming code possessing (γ, B_I, B_L, T)-symbol-level UEP property simply as a (γ, B_I, B_L, T)-UEP streaming code.
2) *(B_I, B_L, T)-Packet-Level Unequal Error Protection:* A streaming coding is said to possess (B_I, B_L, T)-packet-level UEP if the following conditions hold.
   - The source packets at even time slots (i.e., $\{\underline{s}(2t)\}_{t\in[0:\infty)}$) are all (B_I, T)-recoverable.
   - The rest of the source packets, $\{\underline{s}(2t+1)\}_{t\in[0:\infty)}$, possess (B_L, T)-recoverability.

   We refer to a streaming code possessing (B_I, B_L, T)-packet-level UEP property simply as a (B_I, B_L, T)-UEP streaming code.

Remark 1: At first glance, the error protection provided by (γ, B_I, B_L, T) or (B_I, B_L, T)-UEP streaming codes may appear to be limiting, as these codes consider only a single burst of length either B_I or B_L across all time slots $[0 : \infty)$, as per Definition 2. However, owing to the delay-constrained decoder, these codes can in fact recover from any number of burst erasures. For instance, high-priority source symbols in a (γ, B_I, B_L, T)-UEP streaming code and high-priority source packets in a (B_I, B_L, T)-UEP streaming code are recoverable in presence of any number of bursts of length B_I as long as there is a guard interval (where there are no erasures) of T time slots between the end of a burst and the beginning of a subsequent burst. For instance, consider the following periodic erasure pattern:

$$e^\infty = \left\{ \underbrace{1, \ldots, 1}_{B_I}, \underbrace{0, 0, \ldots, 0}_{T}, 1, \ldots, 1, 0, 0, \ldots, 0, \ldots \right\}.$$

Similarly, the low-priority source symbols or source packets tolerate multiple length-B_L bursts which are separated by at least T time slots.

Remark 2: Note that the parameters B_I, B_L satisfy $B_L \leq B_I$ by definition. In addition, we also have $B_I \leq T$, as otherwise, a burst erasure of length B_I starting at time-$2t$ can erase all the coded packets $\{\underline{x}(2t), \underline{x}(2t+1), \ldots, \underline{x}(2t+T), \ldots, \underline{x}(2t+B_I-1)\}$ making it impossible to recover any of the symbols in $\underline{s}(2t)$ with delay T. Moreover, in the packet-level UEP setting, we also assume $B_I < T$ (strict inequality). This is because if $B_I = T$, in order to recover $\underline{s}(2t)$ consisting of k symbols by time-$(t + B_I)$, the size of parity packet $\underline{p}(2t + B_I)$ has to be at least k. This can be easily accomplished by repeating each source packet as the parity packet after $B_I = T$ time slots, i.e., $\underline{p}(t + B_I) = \underline{s}(t)$.

Remark 3: When $B_I = B_L \triangleq B_{\text{MS}}$, both (γ, B_I, B_L, T) and (B_I, B_L, T) UEP streaming code settings merge into a single setting where all the source packets $\{\underline{s}(t)\}_{t\in[0:\infty)}$ are $(B_{\text{MS}}, T_{\text{MS}})$-recoverable (here, $T_{\text{MS}} \triangleq T$). This setting, where every source symbol (or packet) is treated equally important, is pursued in earlier works [2]–[5]. We will refer to this setting as the Martinian-Sundberg setting. As the fundamental limits and the matching streaming code constructions are already known when $B_I = B_L$, we will henceforth assume that $B_L < B_I \leq T$ for (γ, B_I, B_L, T)-UEP streaming codes and $B_L < B_I < T$ for (B_I, B_L, T)-UEP streaming codes.

Table I summarizes key notation that we use while describing symbol-level and packet-level UEP streaming codes. We will now present an overview of the Martinian-Sundberg setting. The optimal streaming code construction under this setting appearing in [2] forms a key ingredient of most of the code constructions in the present paper.

TABLE I
A SUMMARY OF KEY NOTATION USED WHILE DESCRIBING SYMBOL-LEVEL AND PACKET-LEVEL UEP STREAMING CODES

Notation	Definition
B_I	Maximum length of recoverable burst erasures for high-priority source packets/symbols
B_L	Maximum length of recoverable burst erasures for low-priority source packets/symbols
T	Decoding delay constraint in UEP streaming codes
B_{MS}	Maximum length of recoverable burst erasures in the MS scheme
T_{MS}	Decoding delay constraint in the MS scheme
γ	Fraction of high-priority symbols in a source packet
$\underline{s}(t)$	Source packet (has length k)
$\underline{x}(t)$	Coded packet (has length n)
$\underline{p}(t)$	Parity packet (has length $r \triangleq n-k$)

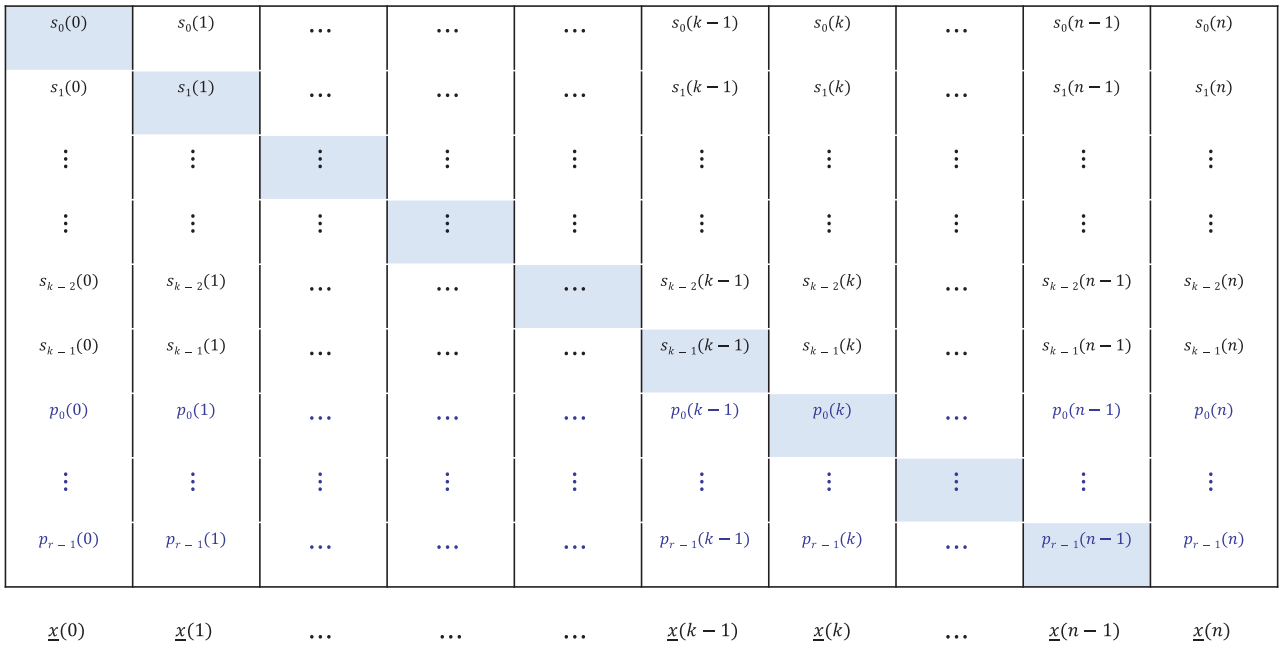


Fig. 1. An illustration of the diagonal interleaving procedure. Each column here depicts a coded packet, where the initial k symbols constitute a source packet. Each diagonal here corresponds to codeword of a systematic $[n, k]$ block code $\mathcal{C}$. In the figure, we have $r \triangleq n-k$.

B. Martinian-Sundberg Setting

Martinian-Sundberg setting is a special case of the two settings that we pursue in this paper, where $B_I = B_L \triangleq B_{\text{MS}}$. We refer to a streaming code under this setting, which allows recovery of all source packets in presence of any B_{MS}-bursty erasure pattern with delay T_{MS}, as a $(B_{\text{MS}}, T_{\text{MS}})$-streaming code. Recall that k and n denote the source and the coded packet sizes (in terms of the field elements in $\mathbb{F}_q$). In [2], the authors present the following upper bound on the rate $R \triangleq \frac{k}{n}$ of a $(B_{\text{MS}}, T_{\text{MS}})$-streaming code:

$$R \leq \frac{T_{\text{MS}}}{T_{\text{MS}} + B_{\text{MS}}}. \tag{2}$$

The paper [2] also presents a novel family of codes namely Maximally Short (MS) codes for a wide range of parameters $\{B_{\text{MS}}, T_{\text{MS}}\}$, which are rate-optimal with respect to (2). In a subsequent paper [16], the authors provide rate-optimal $(B_{\text{MS}}, T_{\text{MS}})$-streaming codes for all parameters $\{B_{\text{MS}}, T_{\text{MS}}\}$.

Let $\mathcal{C}$ denote a systematic $[n, k]$ block code, which has generator matrix of the form $\mathbf{G} \triangleq [\mathbf{I}_k \ \mathbf{P}]$. This code can be utilized to obtain parity packets $\{\underline{p}(t)\}_{t\in[0:\infty)}$ via a certain diagonal interleaving technique, in the following manner:

$$\begin{aligned}&\big[p_0(t)\ p_1(t+1)\ \cdots\ p_{n-k-1}(t+n-k-1)\big]\\ &\quad= \big[s_0(t-k)\ s_1(t-k+1)\ \cdots\ s_{k-1}(t-1)\big]\mathbf{P}.\end{aligned} \tag{3}$$

Fig. 1 illustrates that when the resulting coded packets are arranged as a sequence of columns, each diagonal corresponds to a codeword in $\mathcal{C}$, hence explaining the name "diagonal interleaving". The rate-optimal codes presented in [16] are obtained via diagonally interleaving codewords of carefully designed scalar block codes.

In [16], given the parameters $\{B_{\text{MS}}, T_{\text{MS}}\}$, the generator matrix of $\mathcal{C}$ is chosen as follows:

$$\mathbf{G} = \left[\begin{array}{c|c|c} \mathbf{I}_{B_{\text{MS}}} & \mathbf{0}_{B_{\text{MS}}\times(T_{\text{MS}}-B_{\text{MS}})} & \mathbf{I}_{B_{\text{MS}}} \\ \hline \mathbf{0}_{(T_{\text{MS}}-B_{\text{MS}})\times B} & \multicolumn{2}{c}{\mathbf{G}'} \end{array}\right], \tag{4}$$

where $\mathbf{G}'$ is a $(T_{\text{MS}} - B_{\text{MS}}) \times T_{\text{MS}}$ matrix of the form $[\mathbf{I}_{T_{\text{MS}}-B_{\text{MS}}}\ \mathbf{P}']$ satisfying the following property; any set of $T_{\text{MS}}-B_{\text{MS}}$ consecutive columns (including wrap-around) of $\mathbf{G}'$ should be independent. In [17], the authors provide an explicit construction of $\mathbf{G}'$ over $\mathbb{F}_2$ for any $\{B_{\text{MS}}, T_{\text{MS}}\}$ based on the matrix $\hat{\mathbf{P}}(m, n)$ as described below. For $m, n \in \mathbb{N}$, consider the

following recursive definition of the matrix $\hat{\mathbf{P}}(m,n) \in \mathbb{F}_2^{m\times n}$:

$$\hat{\mathbf{P}}(m,n) = \begin{cases} \begin{bmatrix} \mathbf{I}_m & \hat{\mathbf{P}}(m,n-m) \end{bmatrix} & \text{if } \ m < n, \\ \begin{bmatrix} \mathbf{I}_n \\ \hat{\mathbf{P}}(m-n,n) \end{bmatrix} & \text{else if } m > n, \\ \mathbf{I}_m & \text{else.} \end{cases}$$

The paper [17] chooses $\mathbf{G}' = \hat{\mathbf{P}}(T_{\text{MS}} - B_{\text{MS}}, T_{\text{MS}})$.

Remark 4: After setting $B_I = B_L = B_{\text{MS}}$ and $T = T_{\text{MS}}$, from Remark 2, it follows that the parameters $\{B_{\text{MS}}, T_{\text{MS}}\}$ satisfy $B_{\text{MS}} \leq T_{\text{MS}}$.

C. Main Results

We summarize here the main theoretical results of the present paper. For the symbol-level UEP scenario, we propose a coding scheme and provide a matching rate upper bound (Section III). It is interesting to note that the optimal coding scheme in this scenario, is obtained by encoding the streams of high-priority and low-priority sub-packets *separately* using $(B_{\text{MS}} = B_I, T_{\text{MS}} = T)$ and $(B_{\text{MS}} = B_L, T_{\text{MS}} = T)$ streaming codes, respectively.

Result 1: The capacity of (γ, B_I, B_L, T)-symbol-level UEP streaming codes is given by:

$$\frac{T}{T + B_L + \gamma(B_I - B_L)}.$$

For the packet-level UEP scenario, we propose a coding scheme (Section IV-A) and show its near-optimality by deriving rate upper bounds (Section IV-B).

Result 2: We propose a (B_I, B_L, T)-packet-level UEP streaming code construction with achievable rate $O(1/T)$ away from the rate upper bound. In other words, our construction achieves within one unit of the minimum possible delay.

For some parameter regimes, we also propose optimal packet-level UEP streaming code constructions (Section IV-C) that match the rate outer bound.

Result 3: We provide two rate-optimal code constructions that together cover the regime where B_I is odd and B_L is even.

III. Symbol-Level Unequal Error Protection Codes

The following theorem presents the main result for (γ, B_I, B_L, T)-UEP streaming codes.

Theorem 1: The supremum of rates achievable by (γ, B_I, B_L, T)-UEP streaming codes, i.e., the capacity of (γ, B_I, B_L, T)-UEP streaming codes, is given by:

$$C_{\gamma,B_I,B_L,T} = \frac{T}{T + B_L + \gamma(B_I - B_L)}. \tag{5}$$

Proof: The proof is divided into two parts; Achievability and Converse.

1) *Achievability:* For notational convenience, assume that γk is an integer. Recall from Section II that each source packet is divided into two sub-packets as $\underline{s}(t) \triangleq [\underline{s_I}(t)^\top \ \underline{s_L}(t)^\top]^\top$, where $\underline{s_I}(t) \in \mathbb{F}_q^{\gamma k}$ and $\underline{s_L}(t) \in \mathbb{F}_q^{(1-\gamma)k}$. Here $\underline{s_I}(t)$ and $\underline{s_L}(t)$ consist of high-priority and low-priority source symbols, respectively. The main idea in the code construction is to apply $(B_{\text{MS}}, T_{\text{MS}})$-streaming codes to encode high-priority and low-priority source sub-packets separately. We then concatenate the resulting "coded sub-packets".
   a) Apply $(B_{\text{MS}} = B_I, T_{\text{MS}} = T)$-streaming code (recall the construction described in Section II-B) to the stream of source sub-packets $\{\underline{s_I}(t)\}_{t\in[0:\infty)}$ and denote the resulting stream of "parity sub-packets" as $\{\underline{p_I}(t)\}_{t\in[0:\infty)}$.
   b) Apply $(B_{\text{MS}} = B_L, T_{\text{MS}} = T)$-streaming code to the stream of source sub-packets $\{\underline{s_L}(t)\}_{t\in[0:\infty)}$ and denote the resulting stream of parity sub-packets as $\{\underline{p_L}(t)\}_{t\in[0:\infty)}$.

   Finally, we construct the coded packet at time-t as follows:

$$\underline{x}(t) = \left[\underline{s_I}(t)^\top \ \underline{s_L}(t)^\top \ \underline{p_I}(t)^\top \ \underline{p_L}(t)^\top\right]^\top.$$

   We will show that the construction satisfies all the desired properties of a (γ, B_I, B_L, T)-UEP streaming code. Without loss of generality, assume that the channel introduces a single burst of length B. At the decoder side, conditioned on the length of the burst introduced by channel, we have two cases.
   a) $B \leq B_L$: In this case, $\underline{s_L}(i)$ can be recovered using all non-erased coded sub-packets within the set $\{[\underline{s_L}(t)^\top \ \underline{p_L}(t)^\top]^\top\}_{t\in[0:i+T]}$ due to the properties of $(B_{\text{MS}} = B_L, T_{\text{MS}} = T)$-streaming codes. Similarly, since $B < B_I$, $\underline{s_I}(i)$ can be recovered using all non-erased coded sub-packets within the set $\{[\underline{s_I}(t)^\top \ \underline{p_I}(t)^\top]^\top\}_{t\in[0:i+T]}$.
   b) $B_L < B \leq B_I$: In this case, the decoder discards all non-erased parity sub-packets within $\{\underline{p_L}(t)\}_{t\in[0:i+T]}$ and uses non-erased coded sub-packets within $\{[\underline{s_I}(t)^\top \ \underline{p_I}(t)^\top]^\top\}_{t\in[0:i+T]}$ to recover $\underline{s_I}(i)$.

   Hence, we have the following lower bound on the capacity of (γ, B_I, B_L, T)-UEP streaming codes:

$$\frac{k}{k + (1-\gamma)\frac{kB_L}{T} + \gamma\frac{kB_I}{T}} = \frac{T}{T + B_L + \gamma(B_I - B_L)} \leq C_{\gamma,B_I,B_L,T}.$$

   Here, we have used the fact that the parity packet size of a (B,T)-streaming code is given by $k\frac{B}{T}$.
2) *Converse:* We will begin with providing some intuition behind the converse. Let $r \triangleq n - k$ denote the size of each parity packet $\underline{p}(t)$. Consider the scenario that coded packets $\underline{x}(0), \ldots, \underline{x}(B_I - 1)$ are erased and rest of the coded packets are non-erased. Then, by the definition of (γ, B_I, B_L, T)-UEP streaming codes, the decoder can recover the high-priority source sub-packets $\underline{s_I}(0), \ldots, \underline{s_I}(B_I - B_L - 1)$ using coded packets $\underline{x}(B_I), \ldots, \underline{x}(B_I - B_L - 1 + T)$. Now, assume that the low-priority source sub-packets $\underline{s_L}(0), \ldots, \underline{s_L}(B_I - B_L - 1)$ are "revealed" to the decoder. Together with the knowledge of $\underline{s_I}(0), \ldots, \underline{s_I}(B_I - B_L - 1)$, the decoder can reconstruct coded packets $\underline{x}(0), \ldots, \underline{x}(B_I - B_L - 1)$. In the next step, the decoder essentially deals with a single burst of length B_L spanning time slots $[B_I - B_L : B_I - 1]$. It follows from

the definition of (γ, B_I, B_L, T)-UEP streaming code that both high-priority and low-priority source sub-packets in this interval can be recovered using coded packets $\underline{x}(0), \ldots, \underline{x}(B_I-B_L-1), \underline{x}(B_I), \ldots, \underline{x}(B_I-1+T)$. Coded packets $\underline{x}(0), \ldots, \underline{x}(B_I-1+T)$ contain "information" of $(B_I+T)k$ symbols. As we obtained these (B_I+T) coded packets using coded packets $\underline{x}(B_I), \ldots, \underline{x}(B_I-1+T)$ (contain $T(k+r)$ symbols) and revealed source sub-packets $\underline{s_L}(0), \ldots, \underline{s_L}(B_I-B_L-1)$ (contain $(1-\gamma)(B_I-B_L)k$ symbols), we have $(1-\gamma)(B_I-B_L)k+T(k+r) \geq (B_I+T)k$. As rate is given by $\frac{k}{k+r}$, we thus have an upper bound on the rate. In the following, we formalize these steps. For $j \geq i$, let $\underline{s}_i^j \triangleq \{\underline{s}(t)\}_{t=i}^j$, $\underline{a}_i^j \triangleq \{\underline{s_I}(t)\}_{t=i}^j$, $\underline{b}_i^j \triangleq \{\underline{s_L}(t)\}_{t=i}^j$ and $\underline{x}_i^j \triangleq \{\underline{x}(t)\}_{t=i}^j$. We have:

$$
\begin{aligned}
&(1-\gamma)(B_I-B_L)k + T(k+r) && (6)\\
&\geq H\left(\underline{b}_0^{B_I-B_L-1}\right) + H\left(\underline{x}_{B_I}^{B_I-1+T}\right)\\
&\geq H\left(\underline{b}_0^{B_I-B_L-1}, \underline{x}_{B_I}^{B_I-1+T}\right)\\
&= H\left(\underline{b}_0^{B_I-B_L-1}, \underline{x}_{B_I}^{B_I-1+T}, \underline{a}_0^{B_I-B_L-1}\right) && (7)\\
&= H\left(\underline{x}_0^{B_I-B_L-1}, \underline{x}_{B_I}^{B_I-1+T}\right) && (8)\\
&= H\left(\underline{x}_0^{B_I-B_L-1}, \underline{x}_{B_I}^{B_I-1+T}, \underline{s}_{B_I-B_L}^{B_I-1}\right) && (9)\\
&= H\left(\underline{x}_0^{B_I-1+T}\right) && (10)\\
&= (B_I+T)k. && (11)
\end{aligned}
$$

Here (7) follows from the (B_I, T)-recoverability property of high-priority source sub-packets; (8) is true as, given $\underline{s}_0^{B_I-B_L-1}$, parity packets in time slots $[0 : B_I - B_L - 1]$ can be determined; (9) follows from the (B_L, T)-recoverability property of low-priority source sub-packets and the (B_I, T)-recoverability property of high-priority source sub-packets; (10) is true as parity packets in time slots $[B_I - B_L : B_I - 1]$ is a function of source packets $\underline{s}_0^{B_I-1}$; (11) follows from the fact that source symbols are independent. Hence, for any (γ, B_I, B_L, T)-UEP streaming code, from (6) and (11), we have:

$$R = \frac{k}{k+r} \leq \frac{T}{T+B_L+\gamma(B_I-B_L)}.$$

Thus:

$$C_{\gamma,B_I,B_L,T} \leq \frac{T}{T+B_L+\gamma(B_I-B_L)}.$$

Thus, the proposed code construction is a rate-optimal (γ, B_I, B_L, T)-UEP streaming code. ∎

Remark 5: The construction in Theorem 1 is over $\mathbb{F}_2$, as the constituent codes, i.e., $(B_{\text{MS}} = B_I, T_{\text{MS}} = T)$ and $(B_{\text{MS}} = B_L, T_{\text{MS}} = T)$ streaming codes are both over $\mathbb{F}_2$.

IV. Packet-Level Unequal Error Protection Codes

In a (B_I, B_L, T)-UEP streaming code, source packets in the even time slots, i.e., $\{\underline{s}(2t)\}_{t\in[0:\infty)}$, are of high-priority and hence, have (B_I, T)-recoverability. The remaining packets $\{\underline{s}(2t+1)\}_{t\in[0:\infty)}$ possess (B_L, T)-recoverability. As noted earlier in Remarks 2 and 3, we assume that the parameters $\{B_I, B_L, T\}$ satisfy $0 \leq B_L < B_I < T$. In Section IV-A, we present a (B_I, B_L, T)-UEP streaming code construction for all parameters $\{B_I, B_L, T\}$. In Section IV-B, we provide converse results, i.e., rate upper bounds, which show that the construction is near-optimal. Subsequently, in Section IV-C, we present two *optimal* (B_I, B_L, T)-UEP streaming code constructions which apply for specific parameter ranges.

A. Explicit Construction of (B_I, B_L, T)-UEP/Streaming Codes for all Parameters

We initially provide a construction in Section IV-A1 (which will be referred to as Construction A) for the case $B_L = 0$ and $0 < B_I < T$. We will show later in Section IV-A2 that this construction can easily be extended for $0 \leq B_L < B_I < T$ (Construction B). Both code constructions are over $\mathbb{F}_2$. In Construction A, we set $B_L = 0$ and focus on achieving the (B_I, T)-recoverability property of high-priority source packets. An outline of Construction A is as follows. We divide each high-priority source packet in even time slots into two source sub-packets. This results in two independent source sub-packet streams. We then apply two MS codes separately (one code to each source sub-packet stream) to produce two parity sub-packet streams. The parity sub-packets corresponding to one source sub-packet stream are placed in even time slots, whereas those corresponding to the other source sub-packet stream are placed in odd time slots. As consecutive parity sub-packets within a parity sub-packet stream are two time slots apart, the effective burst length and delay constraint to be tolerated by the two underlying MS codes will be approximately $\frac{B_I}{2}$ and $\frac{T}{2}$, respectively.

1) Construction A ($B_L = 0$, $0 < B_I < T$): As $B_L = 0$, there is no recovery guarantee on source packet $\underline{s}(2t+1)$ if $\underline{x}(2t+1)$ is erased, $t \in [0 : \infty)$. Hence, when $B_L = 0$, we construct each parity packet $\underline{p}(t)$ as a function of only the high-priority source packets $\{\underline{s}(2t') \mid 2t' \leq t\}$. We divide the scenario of $B_L = 0$ into two cases; (i) B_I: even and (ii) B_I: odd.

- *B_I is even:* In this case, we provide a $(B_I, B_L = 0, T)$-UEP streaming code construction with rate $\frac{T-1}{T-1+\frac{B_I}{2}}$. Let the source packet $\underline{s}(t)$ at time-t be of size $k = T - 1$. We partition the source packet into two sub-packets $\underline{s}'(t)$ and $\underline{s}''(t)$ of size $\lceil\frac{T-1}{2}\rceil$ and $\lfloor\frac{T-1}{2}\rfloor$, respectively, in the following manner:

$$
\begin{aligned}
\underline{s}'(t) &\triangleq \left[s_0'(t)\ s_1'(t)\ \cdots\ s_{\lceil\frac{T-1}{2}\rceil-1}'(t)\right]^\top\\
&= \left[s_0(t)\ s_1(t)\ \cdots\ s_{\lceil\frac{T-1}{2}\rceil-1}(t)\right]^\top,\\
\underline{s}''(t) &\triangleq \left[s_0''(t)\ s_1''(t)\ \cdots\ s_{\lfloor\frac{T-1}{2}\rfloor-1}''(t)\right]^\top\\
&= \left[s_{\lceil\frac{T-1}{2}\rceil}(t)\ s_{\lceil\frac{T-1}{2}\rceil+1}(t)\ \cdots\ s_{k-1}(t)\right]^\top.
\end{aligned}
$$

Now consider the following two streams of source sub-packets:

$$\left\{\underline{s}^{(1)}(i)\right\}_{i\in[0:\infty)},\ \left\{\underline{s}^{(2)}(i)\right\}_{i\in[0:\infty)},$$

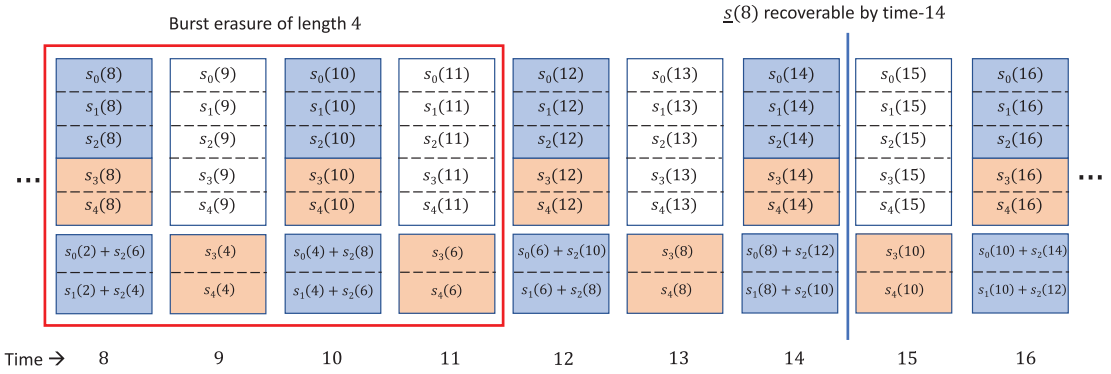


Fig. 2. An illustration of the $(B_I = 4, B_L = 0, T = 6)$-UEP streaming code obtained via Construction A. The symbols marked in blue belong to coded sub-packets produced by a $(B_{\text{MS}} = 2, T_{\text{MS}} = 3)$-streaming code. Symbols in orange belong to coded sub-packets of a $(B_{\text{MS}} = 2, T_{\text{MS}} = 2)$-streaming code.

where $\underline{s}^{(1)}(i) \triangleq \underline{s}'(2i)$, $\underline{s}^{(2)}(i) \triangleq \underline{s}''(2i)$. We apply $(B_{\text{MS}} = \frac{B_I}{2}, T_{\text{MS}} = \lceil \frac{T-1}{2} \rceil)$-streaming code (optimal construction described in Section II-B) to the stream of source sub-packets $\{\underline{s}^{(1)}(i)\}_{i\in[0:\infty)}$ to generate coded sub-packets of the following form:

$$\underline{x}^{(1)}(i) \triangleq \begin{bmatrix} \underline{s}^{(1)}(i) \\ \underline{p}^{(1)}(i) \end{bmatrix}, \tag{12}$$

where $\underline{p}^{(1)}(i) \in \mathbb{F}_2^{\frac{B_I}{2}}$ is a parity sub-packet, which is a function of source sub-packets $\{\underline{s}^{(1)}(i')\}_{i'\in[0:i-1]}$. Similarly, $(B_{\text{MS}} = \frac{B_I}{2}, T_{\text{MS}} = \lfloor \frac{T-1}{2} \rfloor)$-streaming code is applied to source sub-packets $\{\underline{s}^{(2)}(i)\}_{i\in[0:\infty)}$ to generate coded sub-packets of the form:

$$\underline{x}^{(2)}(i) \triangleq \begin{bmatrix} \underline{s}^{(2)}(i) \\ \underline{p}^{(2)}(i) \end{bmatrix}, \tag{13}$$

where parity sub-packet $\underline{p}^{(2)}(i) \in \mathbb{F}_2^{\frac{B_I}{2}}$ is a function of source sub-packets $\{\underline{s}^{(2)}(i')\}_{i'\in[0:i-1]}$. As $B_I < T$ and B_I is even, it follows that $\lfloor \frac{T-1}{2} \rfloor \geq \frac{B_I}{2}$ and hence for both choices of T_{MS}, we have $T_{\text{MS}} \geq B_{\text{MS}}$. Hence, the two required $(B_{\text{MS}}, T_{\text{MS}})$-streaming codes do exist (see Remark 4). As $\{\underline{s}^{(1)}(i')\}_{i'\in[0:i-1]}$ and $\{\underline{s}^{(2)}(i')\}_{i'\in[0:i-1]}$ are derived by partitioning the high-priority source packets $\{\underline{s}(2i')\}_{i'\in[0:i-1]}$, the parity sub-packets $\underline{p}^{(1)}(i)$ and $\underline{p}^{(2)}(i)$ are functions of $\{\underline{s}(2i')\}_{i'\in[0:i-1]}$. The parity packet $\underline{p}(t)$ of the $(B_I, B_L = 0, T)$-UEP streaming code to be constructed is obtained from parity sub-packets $\{\underline{p}^{(1)}(i)\}$ and $\{\underline{p}^{(2)}(i)\}$ as follows:

$$\underline{p}(t) = \begin{cases} \underline{p}^{(1)}\left(\frac{t}{2}\right) & t \text{ is even} \\ \underline{p}^{(2)}\left(\frac{t-1}{2}\right) & t \text{ is odd.} \end{cases} \tag{14}$$

Finally, the coded packet $\underline{x}(t)$ of the $(B_I, B_L = 0, T)$-UEP streaming code is obtained by appending $\underline{p}(t)$ to the source packet $\underline{s}(t)$ as in (1).

Example 1: We discuss here the construction of a $(B_I = 4, B_L = 0, T = 6)$-UEP streaming code. We choose the source packet size to be $k = T - 1 = 5$. Consider the two streams of source sub-packets: $\{\underline{s}^{(1)}(i)\}_{i\in[0:\infty)}$ and $\{\underline{s}^{(2)}(i)\}_{i\in[0:\infty)}$, where $\underline{s}^{(1)}(i) \triangleq \underline{s}'(2i)$, $\underline{s}^{(2)}(i) \triangleq \underline{s}''(2i)$. Source sub-packets $\underline{s}'(2i)$ and $\underline{s}''(2i)$ consist of first three symbols and last two symbols of $\underline{s}(2i)$, respectively. By applying a $(B_{\text{MS}} = 2, T_{\text{MS}} = 3)$-streaming code to $\{\underline{s}^{(1)}(i)\}_{i\in[0:\infty)}$, coded sub-packets $\{\underline{x}^{(1)}(i)\}$ of the form (12) are generated. Similarly, coded sub-packets $\{\underline{x}^{(2)}(i)\}$ (as in (13)) are generated from $\{\underline{s}^{(2)}(i)\}$ by applying a $(B_{\text{MS}} = 2, T_{\text{MS}} = 2)$-streaming code. Let the generator matrices for the underlying block codes which result in $(B_{\text{MS}} = 2, T_{\text{MS}} = 3)$ and $(B_{\text{MS}} = 2, T_{\text{MS}} = 2)$ streaming codes via diagonal interleaving be denoted by $\mathbf{G}^{(1)}$ and $\mathbf{G}^{(2)}$, respectively. From (4), we have:

$$\mathbf{G}^{(1)} \triangleq \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix},$$
$$\mathbf{G}^{(2)} \triangleq \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}.$$

Let $\underline{p}^{(l)}(i) \triangleq [p_0^{(l)}(i)^\top \; p_1^{(l)}(i)^\top]^\top$, for $l = 1, 2$. As seen in (3), the parity symbols within $\{\underline{p}^{(l)}(i)\}$ are obtained as follows:

$$\begin{aligned} &\left[p_0^{(1)}(t) \; p_1^{(1)}(t+1)\right] \\ &\quad = \left[s_0^{(1)}(t-3) \; s_1^{(1)}(t-2) \; s_2^{(1)}(t-1)\right] \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}, \end{aligned} \tag{15}$$

$$\begin{aligned} &\left[p_0^{(2)}(t) \; p_1^{(2)}(t+1)\right] \\ &\quad = \left[s_0^{(2)}(t-2) \; s_1^{(2)}(t-1)\right] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \end{aligned} \tag{16}$$

Using (14) and definitions $\underline{s}^{(1)}(i) \triangleq \underline{s}'(2i)$, $\underline{s}^{(2)}(i) \triangleq \underline{s}''(2i)$, equations (15) and (16) will be transformed to:

$$\begin{aligned} &\left[p_0(2t) \; p_1(2t+2)\right] \\ &\quad = \left[s_0(2t-6) \; s_1(2t-4) \; s_2(2t-2)\right] \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}, \end{aligned}$$

$$\begin{aligned} &\left[p_0(2t+1) \; p_1(2t+3)\right] \\ &\quad = \left[s_3(2t-4) \; s_4(2t-2)\right] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \end{aligned}$$

In Fig. 2, we illustrate the resulting streaming code across time slots $[8 : 16]$. Assume that there is a B_I-bursty erasure pattern which erases coded packets $\underline{x}(8), \ldots, \underline{x}(11)$. All the coded packets transmitted prior to time-8 and after time-11 are non-erased. It can be inferred from the figure that $\underline{s}'(8)$ is recovered by time-14 and $\underline{s}'(10)$ by time-16. Similarly, $\underline{s}''(8)$ and $\underline{s}''(10)$ are recovered by time-13 and time-15, respectively. Hence, the whole source packets $\underline{s}(8)$ and $\underline{s}(10)$ are recovered by time-14 and 16, respectively (delay $T = 6$).

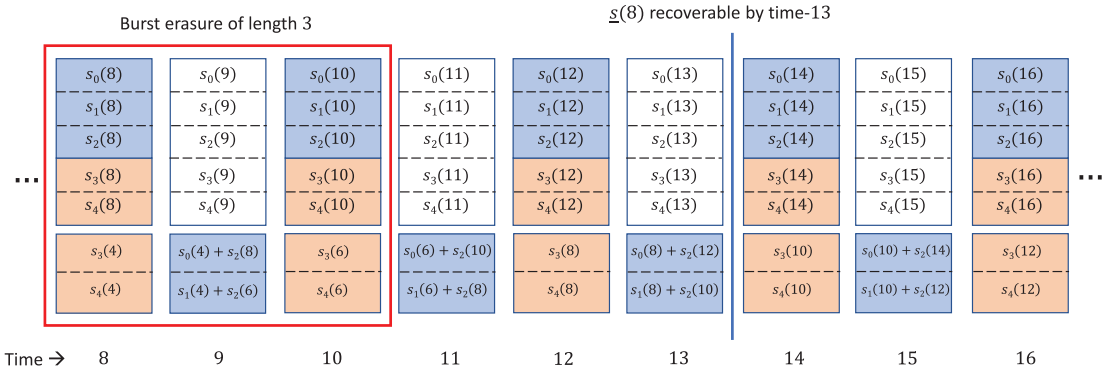


Fig. 3. An illustration of the $(B_I = 3, B_L = 0, T = 5)$-UEP streaming code obtained via Construction A. The symbols marked in blue belong to coded sub-packets of a $(B_{\text{MS}} = 2, T_{\text{MS}} = 3)$-streaming code. Symbols in orange belong to coded sub-packets of a $(B_{\text{MS}} = 2, T_{\text{MS}} = 2)$-streaming code.

- *B_I is odd:* Here, the construction achieves the rate $\frac{T}{T+\lceil \frac{B_I}{2} \rceil}$. Let the source packet $\underline{s}(t)$ be of size $k = T$. We partition each source packet $\underline{s}(t)$ into two sub-packets $\underline{s}'(t)$ and $\underline{s}''(t)$ of size $\lceil \frac{T}{2} \rceil$ and $\lfloor \frac{T}{2} \rfloor$, respectively:

$$\begin{aligned}\underline{s}'(t) &\triangleq \left[s'_0(t)\ s'_1(t)\ \cdots\ s'_{\lceil \frac{T}{2} \rceil - 1}(t)\right]^\top \\ &= \left[s_0(t)\ s_1(t)\ \cdots\ s_{\lceil \frac{T}{2} \rceil - 1}(t)\right]^\top, \\ \underline{s}''(t) &\triangleq \left[s''_0(t)\ s''_1(t)\ \cdots\ s''_{\lfloor \frac{T}{2} \rfloor - 1}(t)\right]^\top \\ &= \left[s_{\lceil \frac{T}{2} \rceil}(t)\ s_{\lceil \frac{T}{2} \rceil + 1}(t)\ \cdots\ s_{k-1}(t)\right]^\top.\end{aligned}$$

As in the previous case, consider source sub-packets $\{\underline{s}^{(1)}(i)\}_{i\in[0:\infty)}$, $\{\underline{s}^{(2)}(i)\}_{i\in[0:\infty)}$, where $\underline{s}^{(1)}(i) \triangleq \underline{s}'(2i), \underline{s}^{(2)}(i) \triangleq \underline{s}''(2i)$. By applying the optimal $(B_{\text{MS}} = \lceil \frac{B_I}{2} \rceil, T_{\text{MS}} = \lceil \frac{T}{2} \rceil)$-streaming code to source sub-packets $\{\underline{s}^{(1)}(i)\}_{i\in[0:\infty)}$, the stream of parity sub-packets $\{\underline{p}^{(1)}(i)\}_{i\in[0:\infty)}$ is obtained. Similarly, the optimal $(B_{\text{MS}} = \lceil \frac{B_I}{2} \rceil, T_{\text{MS}} = \lfloor \frac{T}{2} \rfloor)$-streaming code is applied to source sub-packets $\{\underline{s}^{(2)}(i)\}_{i\in[0:\infty)}$ to obtain $\{\underline{p}^{(2)}(i)\}_{i\in[0:\infty)}$. As $B_I < T$ and B_I is odd, it follows that $\lfloor \frac{T}{2} \rfloor \geq \lceil \frac{B_I}{2} \rceil$ and hence the two required $(B_{\text{MS}}, T_{\text{MS}})$-streaming codes do exist for both choices of T_{MS}. Parity sub-packets $\underline{p}^{(1)}(i) \in \mathbb{F}_2^{\lceil \frac{B_I}{2} \rceil}$ and $\underline{p}^{(2)}(i) \in \mathbb{F}_2^{\lceil \frac{B_I}{2} \rceil}$ are functions of $\{\underline{s}^{(1)}(i')\}_{i'\in[0:i-1]}$ and $\{\underline{s}^{(2)}(i')\}_{i'\in[0:i-1]}$, respectively. The parity packet $\underline{p}(t)$ of the $(B_I, B_L = 0, T)$-UEP streaming code is determined as follows:

$$\underline{p}(t) = \begin{cases} \underline{p}^{(1)}\left(\frac{t+1}{2}\right) & t \text{ is odd} \\ \underline{p}^{(2)}\left(\frac{t}{2}\right) & t \text{ is even.} \end{cases} \tag{17}$$

Finally, coded packet $\underline{x}(t)$ of the $(B_I, B_L = 0, T)$-UEP streaming code is constructed by appending parity packet $\underline{p}(t)$ to the source packet $\underline{s}(t)$.

Example 2: Consider the construction of a $(B_I = 3, B_L = 0, T = 5)$-UEP streaming code. Each source packet $\underline{s}(t)$ consisting of $k = T = 5$ symbols is split to obtain $\underline{s}'(t) \triangleq [s'_0(t)\ s'_1(t)\ s'_2(t)]^\top = [s_0(t)\ s_1(t)\ s_2(t)]^\top$ and $\underline{s}''(t) \triangleq [s''_0(t)\ s''_1(t)]^\top = [s_3(t)\ s_4(t)]^\top$. A $(B_{\text{MS}} = 2, T_{\text{MS}} = 3)$-streaming code is invoked to generate parity sub-packets $\{\underline{p}^{(1)}(i)\}_{i\in[0:\infty)}$ from source sub-packets $\{\underline{s}^{(1)}(i)\}_{i\in[0:\infty)}$, where $\underline{s}^{(1)}(i) \triangleq \underline{s}'(2i)$. Similarly, a $(B_{\text{MS}} = 2, T_{\text{MS}} = 2)$-streaming code is applied to generate $\{\underline{p}^{(2)}(i)\}_{i\in[0:\infty)}$ from $\{\underline{s}^{(2)}(i)\}_{i\in[0:\infty)}$, where $\underline{s}^{(2)}(i) \triangleq \underline{s}''(2i)$. The symbols within parity sub-packets can be written in terms of symbols of source sub-packets precisely as in (15) and (16). However, the way parity sub-packets are mapped to parity packets differ slightly in (17) (compared to (14)). Substituting (17) and definitions $\underline{s}^{(1)}(i) \triangleq \underline{s}'(2i), \underline{s}^{(2)}(i) \triangleq \underline{s}''(2i)$ in (15) and (16), we have:

$$\begin{aligned}\left[p_0(2t-1)\ p_1(2t+1)\right] &= \left[s_0(2t-6)\ s_1(2t-4)\ s_2(2t-2)\right] \\ &\quad \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}, \\ \left[p_0(2t)\ p_1(2t+2)\right] &= \left[s_3(2t-4)\ s_4(2t-2)\right] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.\end{aligned}$$

In Fig. 3, we illustrate the resulting $(B_I = 3, B_L = 0, T = 5)$-UEP streaming code across time slots [8:16]. Assume that there is a B_I-bursty erasure pattern which erases coded packets $\underline{x}(8), \underline{x}(9), \underline{x}(10)$. It can be inferred from the figure that $\underline{s}'(8)$ is recovered by time-13 and $\underline{s}'(10)$ is recovered by time-15. Similarly, $\underline{s}''(8)$ and $\underline{s}''(10)$ are recovered by time-12 and time-14, respectively. Thus, the whole source packets $\underline{s}(8)$ and $\underline{s}(10)$ are recovered by time-13 and 15, respectively (delay $T = 5$).

In the following proposition, we prove that Construction A results in a $(B_I, B_L = 0, T)$-packet-level UEP streaming code, i.e., all source packets of the form $\underline{s}(2t)$, $t \in [0 : \infty)$, are (B_I, T)-recoverable.

Proposition 1: Construction A yields a $(B_I, B_L = 0, T)$-packet-level UEP streaming code.

Proof: Let $\mathcal{C}$ denote the code obtained using Construction A. We show here only the proof for the case when B_I is even. The proof can be extended in an analogous manner when B_I is odd.

As $B_L = 0$, recall that we are not interested in the recovery of source packets belonging to the odd time slots if corresponding coded packets are erased. Hence, in our construction, each parity packet $\underline{p}(t)$ is obtained purely as a function of $\{\underline{s}(2i) \mid 2i \leq t\}$. By design, $\{\underline{x}^{(1)}(i) \triangleq [\underline{s}^{(1)}(i)^\top\ \underline{p}^{(1)}(i)^\top]^\top\}_{i\in[0:\infty)}$ are coded sub-packets of a $(B_{\text{MS}} = \frac{B_I}{2}, T_{\text{MS}} = \lceil \frac{T-1}{2} \rceil)$-streaming code. Let this code be denoted by $\mathcal{C}_e^{(1)}$ (the subscript e here denotes the B_I even case).

TABLE II

A SUMMARY OF PACKETS THAT ARE ERASED WHEN B_I IS EVEN. THE TWO ROWS CORRESPOND TO THE TWO CASES; ERASURE BURST OF LENGTH B_I STARTS AT TIME-$2t$ OR ELSE, TIME-$(2t-1)$. NOTE THAT FOR CONVENIENCE IN EXPOSITION, WE ARE IDENTIFYING SOURCE AND PARITY PACKETS AS SEPARATE ENTITIES, EVEN THOUGH THEY ARE BOTH TRANSMITTED TOGETHER AS A SINGLE CODED PACKET. CONSIDER THE CASE WHEN THE BURST STARTS AT TIME-$(2t-1)$ (ROW-2). ERASED PARITY PACKET $\underline{p}(2t-1) \triangleq \underline{p}^{(2)}(t-1)$ IS A FUNCTION OF NON-ERASED SOURCE SUB-PACKETS $\{\underline{s}^{(2)}(i')\}_{i' \in [0:t-2]} \triangleq \{\underline{s}''(2i')\}_{i' \in [0:t-2]}$ (WHICH ARE PART OF NON-ERASED CODED PACKETS $\{\underline{x}(2i')\}_{i' \in [0:t-2]}$). HENCE, $\underline{p}(2t-1) \triangleq \underline{p}^{(2)}(t-1)$ IS EFFECTIVELY NON-ERASED. THUS, SUB-PACKETS ERASED WHEN THE BURST STARTS AT TIME-$(2t-1)$ FORM A SUBSET OF THOSE IN THE CASE WHEN THE BURST STARTS AT TIME-$2t$

Erased packets of $\mathcal{C}$ (ignoring source packets from odd time slots)	Erased sub-packets of $\mathcal{C}_e^{(1)}$	Erased sub-packets of $\mathcal{C}_e^{(2)}$
$\underline{s}(2t), \underline{s}(2t+2), \ldots, \underline{s}(2t+B_I-2)$ $\underline{p}(2t), \underline{p}(2t+1), \ldots, \underline{p}(2t+B_I-1)$	$\underline{s}^{(1)}(t), \underline{s}^{(1)}(t+1), \ldots, \underline{s}^{(1)}(t+\frac{B_I}{2}-1)$ $\underline{p}^{(1)}(t), \underline{p}^{(1)}(t+1), \ldots, \underline{p}^{(1)}(t+\frac{B_I}{2}-1)$	$\underline{s}^{(2)}(t), \underline{s}^{(2)}(t+1), \ldots, \underline{s}^{(2)}(t+\frac{B_I}{2}-1)$ $\underline{p}^{(2)}(t), \underline{p}^{(2)}(t+1), \ldots, \underline{p}^{(2)}(t+\frac{B_I}{2}-1)$
$\underline{s}(2t), \underline{s}(2t+2), \ldots, \underline{s}(2t+B_I-2)$ $\underline{p}(2t-1), \underline{p}(2t), \ldots, \underline{p}(2t+B_I-2)$	$\underline{s}^{(1)}(t), \underline{s}^{(1)}(t+1), \ldots, \underline{s}^{(1)}(t+\frac{B_I}{2}-1)$ $\underline{p}^{(1)}(t), \underline{p}^{(1)}(t+1), \ldots, \underline{p}^{(1)}(t+\frac{B_I}{2}-1)$	$\underline{s}^{(2)}(t), \underline{s}^{(2)}(t+1), \ldots, \underline{s}^{(2)}(t+\frac{B_I}{2}-1)$ $\underline{p}^{(2)}(t-1), \underline{p}^{(2)}(t), \ldots, \underline{p}^{(2)}(t+\frac{B_I}{2}-2)$

Similarly, let $\mathcal{C}_e^{(2)}$ denote the $(B_{\text{MS}} = \frac{B_I}{2}, T_{\text{MS}} = \lfloor \frac{T-1}{2} \rfloor)$-streaming code generating coded sub-packets $\{\underline{x}^{(2)}(i) \triangleq [\underline{s}^{(2)}(i)^\top \ \underline{p}^{(2)}(i)^\top]^\top\}_{i \in [0:\infty)}$.

Consider an erasure burst of length B_I, which results in the loss of $\frac{B_I}{2}$ coded packets each from even and odd time slots. If the erasure burst is starting at time-$2t$, for some $t \geq 0$, coded packets $\underline{x}(2t), \underline{x}(2t+1), \ldots, \underline{x}(2t+B_I-1)$ are erased. Similarly, if the erasure burst is starting at time-$(2t-1)$, for some $t \geq 1$, coded packets $\underline{x}(2t-1), \underline{x}(2t), \ldots, \underline{x}(2t+B_I-2)$ are erased. In either case, erased source packets from the even time slots, which need to be recovered with a delay of T, are given by $\{\underline{s}(2t), \underline{s}(2t+2), \ldots, \underline{s}(2t+B_I-2)\}$. In Table II, we provide a summary of source sub-packets and parity sub-packets which are erased from $\mathcal{C}_e^{(1)}$ and $\mathcal{C}_e^{(2)}$, as a result of an erasure burst starting at either time-$2t$ or time-$(2t-1)$. If the burst is starting at time-$(2t-1)$, the erased source sub-packets and parity sub-packets are effectively a subset of those in the other case (see the accompanying description of Table II). Hence, from here on, we assume that the erasure burst is starting at time-$2t$. Moreover, it suffices to prove recovery of just $\underline{s}(2t)$ (first source packet in the burst), as the same steps can be repeated to recover the subsequent $(\frac{B_I}{2}-1)$ source packets belonging to even time slots.

In order to recover $\underline{s}^{(1)}(t) \triangleq \underline{s}'(2t)$, the code $\mathcal{C}_e^{(1)}$ which operates on a delay $T_{\text{MS}} = \lceil \frac{T-1}{2} \rceil$ requires access to $\{\underline{x}^{(1)}(i)\}_{i=0}^{t-1} \cup \{\underline{x}^{(1)}(i)\}_{i=t+\frac{B_I}{2}}^{t+\lceil \frac{T-1}{2} \rceil}$. As there are no erasures prior to time-$2t$, clearly, coded sub-packets $\{\underline{x}^{(1)}(i)\}$ are available. It can be verified from Table II that none of the sub-packets among $\{\underline{x}^{(1)}(i)\}_{i=t+\frac{B_I}{2}}^{t+\lceil \frac{T-1}{2} \rceil} \triangleq \{[\underline{s}^{(1)}(i)^\top \ \underline{p}^{(1)}(i)^\top]^\top\}_{i=t+\frac{B_I}{2}}^{t+\lceil \frac{T-1}{2} \rceil}$ are erased. In order to obtain the "last" coded sub-packet:

$$\begin{bmatrix} \underline{s}^{(1)}\Big(t+\lceil \frac{T-1}{2} \rceil\Big) \\ \underline{p}^{(1)}\Big(t+\lceil \frac{T-1}{2} \rceil\Big) \end{bmatrix} \triangleq \begin{bmatrix} \underline{s}\Big(2t+2\lceil \frac{T-1}{2} \rceil\Big) \\ \underline{p}\Big(2t+2\lceil \frac{T-1}{2} \rceil\Big) \end{bmatrix},$$

which helps in recovery of $\underline{s}^{(1)}(t)$, the decoder needs to wait only till time $2(t+\lceil \frac{T-1}{2} \rceil) \leq 2t+T$. Thus $s'(2t) \triangleq \underline{s}^{(1)}(t)$ can be recovered with a delay of at most T.

Similarly, in order to recover $\underline{s}''(2t) \triangleq \underline{s}^{(2)}(t)$, consider the code $\mathcal{C}_e^{(2)}$ which operates on a delay $T_{\text{MS}} = \lfloor \frac{T-1}{2} \rfloor$. Using $\mathcal{C}_e^{(2)}$, the decoder requires access to $\{\underline{x}^{(2)}(i)\}_{i=0}^{t-1} \cup \{\underline{x}^{(2)}(i)\}_{i=t+\frac{B_I}{2}}^{t+\lfloor \frac{T-1}{2} \rfloor}$ for recovering $\underline{s}^{(2)}(t)$. Clearly, $\{\underline{x}^{(2)}(i)\}_{i=0}^{t-1}$ are available as there are no erasure prior to time-$2t$. From Table II, it can be verified that no sub-packets among $\{\underline{x}^{(2)}(i)\}_{i=t+\frac{B_I}{2}}^{t+\lfloor \frac{T-1}{2} \rfloor} \triangleq \{[\underline{s}^{(2)}(i)^\top \ \underline{p}^{(2)}(i)^\top]^\top\}_{i=t+\frac{B_I}{2}}^{t+\lfloor \frac{T-1}{2} \rfloor}$ are erased. In order to obtain the last coded sub-packet:

$$\begin{bmatrix} \underline{s}^{(2)}\Big(t+\lfloor \frac{T-1}{2} \rfloor\Big) \\ \underline{p}^{(2)}\Big(t+\lfloor \frac{T-1}{2} \rfloor\Big) \end{bmatrix} \triangleq \begin{bmatrix} \underline{s}\Big(2t+2\lfloor \frac{T-1}{2} \rfloor\Big) \\ \underline{p}\Big(2t+2\lfloor \frac{T-1}{2} \rfloor+1\Big) \end{bmatrix},$$

the decoder has to wait only till time $2(t+\lfloor \frac{T-1}{2} \rfloor)+1 \leq 2t+T$. Thus $s''(2t)$ can also be recovered with a delay of at most T. ■

2) Construction B ($0 \leq B_L < B_I < T$): Construction A presented in Section IV-A1 can be generalized (which will be referred to as Construction B) in a straightforward manner to any $B_L \geq 0$ by applying it separately across packets in even, odd time slots and concatenating the parity packets.

- *B_L and B_I are both odd:* Let source packets have size $k = T$. Construction A is applied to protect source packets in even time slots from B_I-bursty erasure patterns, which generates parity packets (say, $\{\underline{p}_I(t)\}_{t \in [0:\infty)}$) of size $\lceil \frac{B_I}{2} \rceil$ at each time. Another instance of Construction A will be applied to protect source packets in odd time slots from B_L-bursty erasure patterns, where it generates parity packets (say, $\{\underline{p}_L(t)\}_{t \in [0:\infty)}$) of size $\lceil \frac{B_L}{2} \rceil$ at each time. Concatenation of parity packets generated from these two codes results in parity packets $\{\underline{p}(t)\}_{t \in [0:\infty)}$ of size $\lceil \frac{B_I}{2} \rceil + \lceil \frac{B_L}{2} \rceil$, where $\underline{p}(t) \triangleq [\underline{p}_I(t)^\top \ \underline{p}_L(t)^\top]^\top$. Clearly, the resulting code having coded packets of the form (1) is a (B_I, B_L, T)-UEP streaming code. Rate achievable, in this case, is $\frac{T}{T+\lceil \frac{B_I}{2} \rceil + \lceil \frac{B_L}{2} \rceil}$.
- *B_L is odd, B_I is even:* Let source packets have size $k = T(T-1)$. Source packets in even time slots will be protected from B_I-bursty erasure patterns by applying Construction A, resulting in parity packets of size $T\lceil \frac{B_I}{2} \rceil$. Similarly, application of another instance of Construction A to protect source packets in odd time slots from B_L-bursty erasure patterns, generates parity packets of size $(T-1)\lceil \frac{B_L}{2} \rceil$. As in the previous case, parity packets generated due to source packets in even and odd time

TABLE III
ACHIEVABLE RATES FOR CONSTRUCTION B PRESENTED IN SECTION IV-A2

Condition	Rate
B_L: odd, B_I: odd	$\frac{T}{T+\lceil\frac{B_I}{2}\rceil+\lceil\frac{B_L}{2}\rceil}$
B_L: odd, B_I: even	$\frac{T(T-1)}{T(T-1)+T\lceil\frac{B_I}{2}\rceil+(T-1)\lceil\frac{B_L}{2}\rceil}$
B_L: even, B_I: odd	$\frac{T(T-1)}{T(T-1)+(T-1)\lceil\frac{B_I}{2}\rceil+T\lceil\frac{B_L}{2}\rceil}$
B_L: even, B_I: even	$\frac{T-1}{T-1+\lceil\frac{B_I}{2}\rceil+\lceil\frac{B_L}{2}\rceil}$

slots will be concatenated to get $\{\underline{p}(t)\}_{t\in[0:\infty)}$, where each $\underline{p}(t)$ has size $T\lceil\frac{B_I}{2}\rceil+(T-1)\lceil\frac{B_L}{2}\rceil$. The resulting code is a (B_I, B_L, T)-UEP streaming code having rate $\frac{T(T-1)}{T(T-1)+T\lceil\frac{B_I}{2}\rceil+(T-1)\lceil\frac{B_L}{2}\rceil} > \frac{T-1}{T-1+\lceil\frac{B_I}{2}\rceil+\lceil\frac{B_L}{2}\rceil}$.

In a similar manner, it is possible to obtain constructions for the remaining cases of (B_L: even, B_I: odd) and (B_L: even, B_I: even). The achievable rates are summarized in Table III.

B. Rate Upper Bounds for (B_I, B_L, T)-UEP Streaming Codes

In the following theorem, we upper bound the rate achievable by a (B_I, B_L, T)-UEP streaming code.

Theorem 2: Let $\mathcal{C}$ be a (B_I, B_L, T)-UEP streaming code with rate R. Then R satisfies:

$$R \leq \begin{cases} \frac{T+1}{T+1+\left\lceil\frac{B_L}{2}\right\rceil+\left\lceil\frac{B_I}{2}\right\rceil} & B_L\text{: odd, } B_I\text{: odd} \\ \frac{T}{T+\left\lceil\frac{B_L}{2}\right\rceil+\left\lceil\frac{B_I}{2}\right\rceil} & \text{otherwise.} \end{cases} \tag{18}$$

Proof: For $j \geq i$, recall that $\underline{s}_i^j \triangleq \{\underline{s}(t)\}_{t=i}^j$ and $\underline{x}_i^j \triangleq \{\underline{x}(t)\}_{t=i}^j$. In addition, let $\tilde{\underline{s}}_i^j \triangleq \underline{s}_i^j \cap \{\underline{s}(i+2\ell)\}_{\ell=0}^{\infty}$. We will discuss here only the case where B_I and B_L are both odd. For the remaining cases, the proof can be extended along similar lines. Let $r \triangleq n-k$ denote the size of each parity packet $\underline{p}(t)$. We will begin with providing an outline of the proof. Consider a burst erasure of length B_I erasing coded packets $\underline{x}_0^{B_I-1}$. Owing to the (B_I, T)-recoverability of high-priority source packets, $\tilde{\underline{s}}_0^{B_I-1}$ can be recovered by time-(B_I-1+T). Assume that low-priority source packets $\tilde{\underline{s}}_1^{B_I-B_L-3}$ and parity packet $\underline{p}(B_I-1)$ are "revealed" to the decoder. Together with the high-priority source packets $\tilde{\underline{s}}_0^{B_I-B_L-2}$, which are already recovered, the decoder can reconstruct the coded packets $\underline{x}(0), \ldots, \underline{x}(B_I-B_L-2)$. As $\underline{p}(B_I-1)$ is revealed and $\underline{s}(B_I-1)$ is recovered, coded packet $\underline{x}(B_I-1)$ is also known to the decoder. Hence, the erased coded packets are effectively restricted to within time slots $[B_I-B_L-1 : B_I-2]$ (a burst of length B_I). Owing to the (B_L, T)-recoverability of low-priority source packets, $\tilde{\underline{s}}_{B_I-B_L-1}^{B_I-2}$ can be recovered. In summary, the decoder can retrieve all of $\underline{x}_0^{B_I-1+T}$ (contain "information" of $(B_I+T)k$ symbols) using $\{\underline{x}_{B_I}^{B_I-1+T}, \tilde{\underline{s}}_1^{B_I-B_L-3}, \underline{p}(B_I-1)\}$ (a total of $T(k+r)+(\frac{B_I-B_L-2}{2})k+r$ symbols). Thus, we have $(\frac{B_I-B_L-2}{2})k+r+T(k+r) \geq (B_I+T)k$. As rate is given by $\frac{k}{k+r}$, we thus have an upper bound on the rate. We will now formalize this idea.

$$\left(\frac{B_I-B_L-2}{2}\right)k+r+T(k+r) \tag{19}$$

$$\begin{aligned}
&\geq H\left(\tilde{\underline{s}}_1^{B_I-B_L-3}\right)+H\left(\underline{p}(B_I-1)\right)+H\left(\underline{x}_{B_I}^{B_I+T-1}\right) \\
&\geq H\left(\tilde{\underline{s}}_1^{B_I-B_L-3}, \underline{p}(B_I-1), \underline{x}_{B_I}^{B_I+T-1}\right) \\
&= H\left(\tilde{\underline{s}}_1^{B_I-B_L-3}, \underline{p}(B_I-1), \underline{x}_{B_I}^{B_I+T-1}, \tilde{\underline{s}}_0^{B_I-1}\right) && (20) \\
&= H\Big(\underline{s}_0^{B_I-B_L-2}, \underline{s}(B_I-1), \\
&\qquad \underline{p}(B_I-1), \underline{x}_{B_I}^{B_I+T-1}, \tilde{\underline{s}}_{B_I-B_L}^{B_I-3}\Big) && (21) \\
&= H\left(\underline{x}_0^{B_I-B_L-2}, \underline{x}_{B_I-1}^{B_I-1+T}, \tilde{\underline{s}}_{B_I-B_L}^{B_I-3}\right) && (22) \\
&= H\left(\underline{x}_0^{B_I-B_L-2}, \underline{x}_{B_I-1}^{B_I-1+T}, \tilde{\underline{s}}_{B_I-B_L}^{B_I-3}, \tilde{\underline{s}}_{B_I-B_L-1}^{B_I-2}\right) && (23) \\
&= H\left(\underline{x}_0^{B_I-B_L-2}, \underline{x}_{B_I-1}^{B_I-1+T}, \underline{s}_{B_I-B_L-1}^{B_I-2}\right) && (24) \\
&= H\left(\underline{x}_0^{B_I-1+T}\right) && (25) \\
&= (B_I+T)k, && (26)
\end{aligned}$$

where (20) follows from the (B_I, T)-recoverability property of high-priority source packets; (21) is obtained by rearranging the terms in (20); (22) is true as parity packets in time slots $[0 : B_I-B_L-2]$ can be obtained as a deterministic function of $\underline{s}_0^{B_I-B_L-2}$; (23) follows from the (B_L, T)-recoverability property of low-priority source packets; (24) is obtained by rearranging the terms in (23); (25) follows as parity packets in time slots $[0 : B_I-2]$ can be obtained as a deterministic function of $\underline{s}_0^{B_I-2}$; (26) follows from the fact that source symbols are independent. From (19) and (26), we have:

$$\begin{aligned}
(T+1)r &\geq \left(\frac{B_I+B_L+2}{2}\right)k \\
&= \left(\left\lceil\frac{B_I}{2}\right\rceil+\left\lceil\frac{B_L}{2}\right\rceil\right)k.
\end{aligned} \tag{27}$$

From (27),

$$R = \frac{k}{k+r} \leq \frac{T+1}{T+1+\left\lceil\frac{B_I}{2}\right\rceil+\left\lceil\frac{B_L}{2}\right\rceil}.$$

■

Near-Optimality of Construction B: From Table III and (18), it can be inferred that the rates achievable by Construction B are away within one unit of delay T from the optimal rates.

Remark 6: From (2) and (18), it can be inferred that if $B_L \in \{B_I-1, B_I\}$, $(B_{\text{MS}} = B_I, T_{\text{MS}} = T)$-streaming codes proposed in [16] are optimal.

C. Rate-Optimal (B_I, B_L, T)-UEP Streaming Codes: B_I is Odd and B_L is Even

Recall that in the near-optimal Construction B, the central idea is to apply Construction A separately on high and low priority source packets. We will now present two, more involved, coding schemes (Construction C & D), where we perform *joint coding* across high and low priority source packets. As we will see now, such an approach leads to optimal code rates which meet (18). Let $B_{\text{eff}} \triangleq \lceil\frac{B_I}{2}\rceil+\frac{B_L}{2}$. We assume that $B_L \leq B_I-2$ as optimal codes exist trivially otherwise (see Remark 6). Moreover, as B_I is odd and B_L is even, we have $B_L < B_I-2$. Construction C and Construction D cover regimes

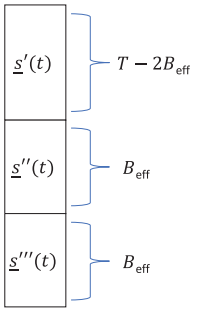


Fig. 4. Each source packet $\underline{s}(t)$ of size T is partitioned into three sub-packets; $\underline{s}'(t)$ of size $T-2B_{\text{eff}}$; $\{\underline{s}''(t), \underline{s}'''(t)\}$ of size B_{eff} each.

$T \geq 2B_{\text{eff}}$ and $T < 2B_{\text{eff}}$, respectively. Both constructions achieve the rate:

$$R_{\text{opt}} = \frac{T}{T+B_{\text{eff}}}, \tag{28}$$

which meets the upper bound in (18).

1) Construction C ($T \geq 2B_{eff}$): From (28), it follows that the condition $T \geq 2B_{\text{eff}}$ results in $R_{\text{opt}} \geq \frac{2}{3}$. Hence, this regime may be referred to as the high-rate regime. Let the source packet size be $k = T$. We partition the source packet $\underline{s}(t)$ into three sub-packets: $\underline{s}'(t) \triangleq [s_0(t)\ s_1(t)\ \ldots\ s_{T-2B_{\text{eff}}-1}(t)]^T$, $\underline{s}''(t) \triangleq [s_{T-2B_{\text{eff}}}(t)\ s_{T-2B_{\text{eff}}+1}(t)\ \cdots\ s_{T-B_{\text{eff}}-1}(t)]^T$ and $\underline{s}'''(t) \triangleq [s_{T-B_{\text{eff}}}(t)\ s_{T-B_{\text{eff}}+1}(t)\ \cdots\ s_{T-1}(t)]^T$ (see Fig. 4). In this construction, each parity packet $\underline{p}(t)$ is composed of three components; $\underline{p}'(t), \underline{p}''(t), \underline{p}'''(t)$:

$$\underline{p}(t) = \underline{p}'(t) + \underline{p}''(t) + \underline{p}'''(t), \tag{29}$$

where $\{\underline{p}(t), \underline{p}'(t), \underline{p}''(t), \underline{p}'''(t)\} \subseteq \mathbb{F}_q^{B_{\text{eff}}}$. The sub-packets $\{\underline{s}''(t)\}, \{\underline{s}'''(t)\}$ are replicated as parity packet components in the following manner:

$$\underline{p}''(t) = \begin{cases} \underline{s}''(t-T+1) & (t-T) \text{ is odd} \\ \underline{s}'''(t-T) & (t-T) \text{ is even,} \end{cases} \tag{30}$$

$$\underline{p}'''(t) = \begin{cases} \underline{s}''(t-B_L) & t \text{ is odd} \\ \underline{s}'''(t-B_L-1) & t \text{ is even.} \end{cases} \tag{31}$$

In other words, (30) and (31) imply the following. For any packet $\underline{s}(t)$ in an even time slot t, we have $\underline{s}''(t)$ replicated as $\underline{p}''(t+T-1) = \underline{s}''(t)$, $\underline{s}'''(t)$ replicated as $\underline{p}''(t+T) = \underline{s}'''(t)$. Similarly, for any packet $\underline{s}(t)$ in an odd time slot t, the sub-packet $\underline{s}''(t)$ is replicated as $\underline{p}'''(t+B_L) = \underline{s}''(t)$ and $\underline{s}'''(t)$ is replicated as $\underline{p}'''(t+B_L+1) = \underline{s}'''(t)$. The remaining parity packet component $p'(t)$ is a function of sub-packets $\{\underline{s}'(t')\}_{t' \in [0:t]}$. In the following, we discuss how $p'(t)$ is determined, which will complete the description of our code construction.

Let $\lambda \triangleq (T-2B_{\text{eff}})B_{\text{eff}}$. Let $J^{(e)} \triangleq 2[0 : \lceil \frac{B_I}{2} \rceil - 1] = \{0, 2, \ldots, 2(\lceil \frac{B_I}{2} \rceil - 1)\}$, i.e., all even numbers in $[0 : B_I - 1]$. For $j \in J^{(e)}$, let $\underline{v}_j^{(e)}(t)$ denote the sum of all sub-packets within the set $\{\underline{s}'(\ell)\}_{\ell \in [0:t-1]}$, which are precisely $B_I + 1$ time slots apart, starting from time-j, i.e.,

$$\underline{v}_j^{(e)}(t) = \sum_{i \in \mathcal{I}_j^{(e)}(t)} \underline{s}'(i), \tag{32}$$

where $\mathcal{I}_j^{(e)}(t) \triangleq \{j, j+(B_I+1), j+2(B_I+1), \ldots\} \cap [0 : t-1]$. Note that all sub-packets involved in (32) belong to even time slots. Similarly, we define $J^{(o)} \triangleq 2[0 : \frac{B_L}{2} - 1] + 1 = \{1, 3, \ldots, 2(\frac{B_L}{2} - 1) + 1\}$, i.e., all odd numbers in $[1 : B_L - 1]$. For $j \in J^{(o)}$, let $\underline{v}_j^{(o)}(t)$ denote the sum of all sub-packets within $\{\underline{s}'(\ell)\}_{\ell \in [0:t-1]}$, which are precisely B_L time slots apart, starting from time-j. i.e.,

$$\underline{v}_j^{(o)}(t) = \sum_{i \in \mathcal{I}_j^{(o)}(t)} \underline{s}'(i), \tag{33}$$

where $\mathcal{I}_j^{(o)}(t) \triangleq \{j, j+B_L, j+2B_L, \ldots\} \cap [0 : t-1]$. All sub-packets involved in (33) belong to odd time slots. Let $\underline{v}(t)$ be a column vector of size $\lambda \times 1$, which is obtained by concatenating all the B_{eff} column vectors $\{\underline{v}_j^{(e)}(t)\}_{j \in J^{(e)}} \cup \{\underline{v}_j^{(o)}(t)\}_{j \in J^{(o)}}$. i.e., $\underline{v}(t)$ takes the form:

$$\underline{v}(t) \triangleq \begin{bmatrix} \underline{v}^{(e)}(t) \\ \underline{v}^{(o)}(t) \end{bmatrix},$$

where:

$$\underline{v}^{(e)}(t) \triangleq \begin{bmatrix} \underline{v}_0^{(e)}(t) \\ \underline{v}_2^{(e)}(t) \\ \vdots \\ \underline{v}_{2\left(\left\lceil \frac{B_I}{2} \right\rceil - 1\right)}^{(e)}(t) \end{bmatrix}, \quad \underline{v}^{(o)}(t) \triangleq \begin{bmatrix} \underline{v}_1^{(o)}(t) \\ \underline{v}_3^{(o)}(t) \\ \vdots \\ \underline{v}_{2\left(\frac{B_L}{2}-1\right)+1}^{(o)}(t) \end{bmatrix}. \tag{34}$$

The parity packet component $\underline{p}'(t)$ consists of B_{eff} consecutive entries of $\underline{v}(t)$ in the following manner; $\underline{p}'(0)$ consists of first B_{eff} entries of $\underline{v}(0)$, $\underline{p}'(1)$ consists of next B_{eff} entries of $\underline{v}(1)$,..., $\underline{p}'(T-2B_{\text{eff}}-1)$ consists of last B_{eff} entries of $\underline{v}(T-2B_{\text{eff}}-1)$, $\underline{p}'(T-2B_{\text{eff}})$ consists of first B_{eff} entries of $\underline{v}(T-2B_{\text{eff}})$ and so on. i.e., in general, if $t' = t \bmod (T-2B_{\text{eff}})$, $\underline{p}'(t)$ consists of entries $[t'B_{\text{eff}} : (t'+1)B_{\text{eff}}-1]$ of $\underline{v}(t)$. This completes the construction. In Appendix A, we show that the construction satisfies all the desired properties of a (B_I, B_L, T)-UEP streaming code.

Example 3: We consider here the construction of a rate-optimal $(B_I = 5, B_L = 2, T = 9)$-UEP streaming code. Here, source packet size $k = T = 9$, $B_{\text{eff}} = 4, \lambda = 4, J^{(e)} = \{0, 2, 4\}, J^{(o)} = \{1\}$. Set $\underline{s}(t) \triangleq \underline{0}$ if $t < 0$. We have: $\underline{p}'(0) = [0\ 0\ 0\ 0]^\top$, $\underline{p}'(1) = [s_0(0)\ 0\ 0\ 0]^\top$, $\underline{p}'(2) = [s_0(0)\ 0\ 0\ s_0(1)]^\top$, $\underline{p}'(3) = [s_0(0)\ s_0(2)\ 0\ s_0(1)]^\top$, $\underline{p}'(4) = [s_0(0)\ s_0(2)\ 0\ s_0(1) + s_0(3)]^\top$, $\underline{p}'(5) = [s_0(0)\ s_0(2)\ s_0(4)\ s_0(1) + s_0(3)]^\top$, $\underline{p}'(6) = [s_0(0)\ s_0(2)\ s_0(4)\ s_0(1) + s_0(3) + s_0(5)]^\top$, $\underline{p}'(7) = [s_0(0) + s_0(6)\ s_0(2)\ s_0(4)\ s_0(1) + s_0(3) + s_0(5)]^\top$ etc. (see Fig. 5).

2) Construction D ($T < 2B_{eff}$): Let the source packet size be $k = \kappa T$, where $\kappa \triangleq \lfloor \frac{T}{2} \rfloor \frac{B_L}{2}$. The parity packets $\{\underline{p}(t)\}$ here will have size κB_{eff}. Let $\mathcal{C}^{(o)}$, $\mathcal{C}^{(e)}$ denote an optimal $(B_{\text{MS}} = \frac{B_L}{2}, T_{\text{MS}} = \lfloor \frac{T}{2} \rfloor)$-streaming code and $(B_{\text{MS}} = \lceil \frac{B_I}{2} \rceil, T_{\text{MS}} = \lfloor \frac{T}{2} \rfloor)$-streaming code, respectively. Each source packet $\underline{s}(t)$ is partitioned into two sub-packets $\{\underline{s}'(t), \underline{s}''(t)\}$, where the partitioning scheme differs based on whether the packet belongs to an even or odd time slot. In even time slots, we have: $\underline{s}'(2t) \triangleq \underbrace{[s_0(2t)\ s_1(2t)\ \ldots\ s_{\kappa B_{\text{eff}}-1}(2t)]^T}_{\kappa B_{\text{eff}} \text{ symbols}}$

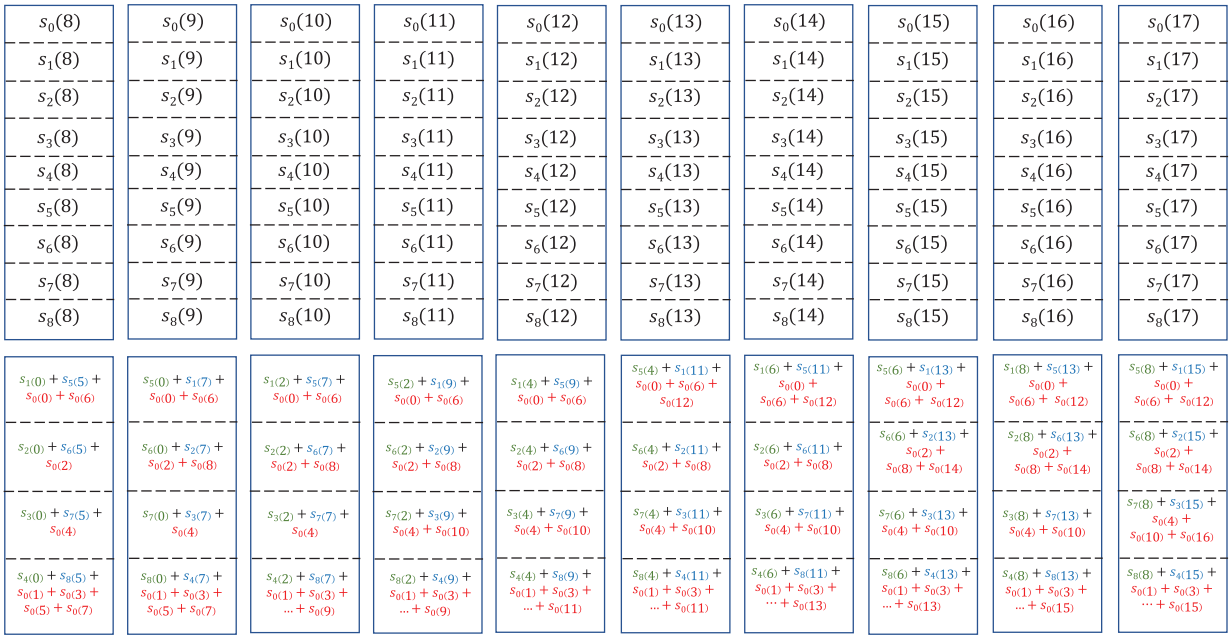


Fig. 5. A rate-optimal $(B_I = 5, B_L = 2, T = 9)$-UEP streaming code obtained via Construction C. Here red, green, blue symbols indicate contributions from $\underline{p}'(t)$, $\underline{p}''(t)$ and $\underline{p}'''(t)$, respectively.

and $\underline{s}''(2t) \triangleq \underbrace{[s_{\kappa B_{\text{eff}}}(2t)\ s_1(2t)\ \ldots\ s_{\kappa T-1}(2t)]^T}_{\kappa(T-B_{\text{eff}})\text{ symbols}}$. For source packets in odd time slots, we have: $\underline{s}'(2t+1) \triangleq \underbrace{[s_0(2t+1)\ s_1(2t+1)\ \ldots\ s_{\kappa\nu-1}(2t+1)]^T}_{\kappa\nu\text{ symbols}}$ and $\underline{s}''(2t+1) \triangleq \underbrace{[s_{\kappa\nu}(2t+1)\ s_1(2t+1)\ \ldots\ s_{\kappa T-1}(2t+1)]^T}_{\kappa(T-\nu)\text{ symbols}}$, where $\nu \triangleq \frac{\lceil \frac{T}{2}\rceil - \lceil \frac{B_I}{2}\rceil}{\frac{B_L}{2}} B_{\text{eff}}$. It can easily be verified that if $T < 2B_{\text{eff}}$, we have $\nu < T$.

We apply $\mathcal{C}^{(e)}$ to $\{\underline{s}''(2t)\}_{t\in[0:\infty)}$ (sub-packets from even time slots) to generate parity sub-packets $\{\underline{p}^{(e)}(i)\}_{i\in[0:\infty)}$. Each parity sub-packet $\underline{p}^{(e)}(i)$ has size $\frac{\kappa(T-B_{\text{eff}})}{\lfloor \frac{T}{2}\rfloor}\lceil \frac{B_I}{2}\rceil$. Similarly, $\mathcal{C}^{(o)}$ is applied to $\{\underline{s}''(2t+1)\}_{t\in[0:\infty)}$ (sub-packets from odd time slots) to generate parity sub-packets $\{\underline{p}^{(o)}(i)\}_{i\in[0:\infty)}$ of size $\frac{\kappa(T-\nu)}{\lfloor \frac{T}{2}\rfloor}\frac{B_L}{2}$ each. It can be verified that $\frac{\kappa(T-B_{\text{eff}})}{\lfloor \frac{T}{2}\rfloor}\lceil \frac{B_I}{2}\rceil + \frac{\kappa(T-\nu)}{\lfloor \frac{T}{2}\rfloor}\frac{B_L}{2} = \kappa B_{\text{eff}}$, i.e., sizes of $\underline{p}^{(e)}(i)$ and $\underline{p}^{(o)}(i)$ add up to κB_{eff}.

If T is even, in odd time slots, we have:

$$\underline{p}(2t+1) = \begin{bmatrix} \underline{p}^{(e)}(t+1) \\ \underline{p}^{(o)}(t) \end{bmatrix},\ t \in [0:\infty).$$

Similarly, if T is odd, in even time slots, we have:

$$\underline{p}(2t) = \begin{bmatrix} \underline{p}^{(e)}(t) \\ \underline{p}^{(o)}(t-1) \end{bmatrix},\ t \in [0:\infty).$$

Parity packets in remaining time slots are obtained as a function of $\underline{s}'(.)$'s in even and odd time slots, in the following manner. Let $\nu' \triangleq \lceil \frac{T}{2}\rceil - \lceil \frac{B_I}{2}\rceil$, $\lambda' \triangleq \kappa\nu\frac{B_L}{2} = \kappa\nu' B_{\text{eff}}$. Consider $\underline{v}_j^{(o)}(t)$ and $\underline{v}^{(o)}(t)$ as defined in (33) and (34). Here, the vector $\underline{v}^{(o)}(t)$ has size $\frac{B_L}{2}\kappa\nu = \lambda'$. For $i \in [0:\nu'-1]$, let $\underline{v}^{(o,i)}(t)$ denote the vector composed of κB_{eff} consecutive entries $[i\kappa B_{\text{eff}} : (i+1)\kappa B_{\text{eff}} - 1]$ of $\underline{v}^{(o)}(t)$. If T is even, in even time slots, we have:

$$\underline{p}(2t) = \underline{v}^{(o,i_t)}(2t) + \underline{s}'(2t-T),\ t \in [0:\infty), \qquad (35)$$

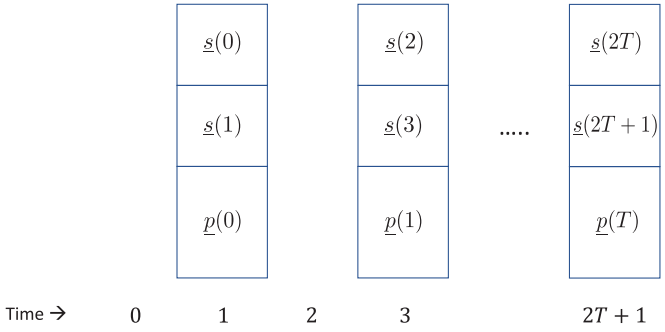


Fig. 6. The source packets $\underline{s}(2t)$ and $\underline{s}(2t+1)$ are combined to form a single long source packet in time-$(2t+1)$. In the presence of an erasure burst of length B_I, the source packet $\underline{s}(2t)$ can only be recovered at time-$(2t+2T+1)$ (delay of $2T+1$ time slots).

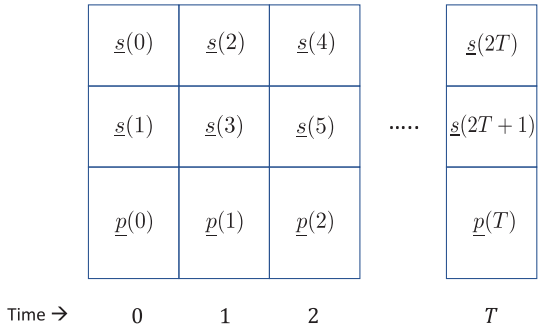
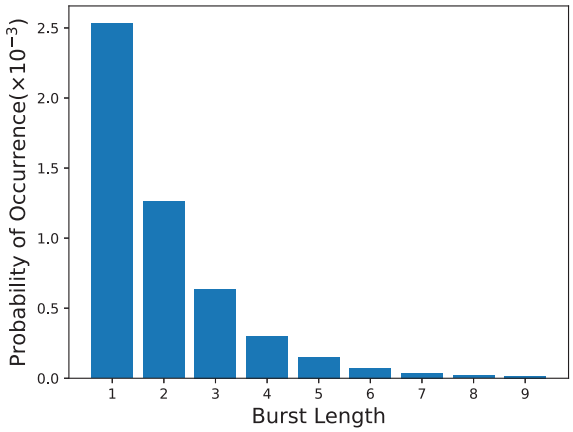
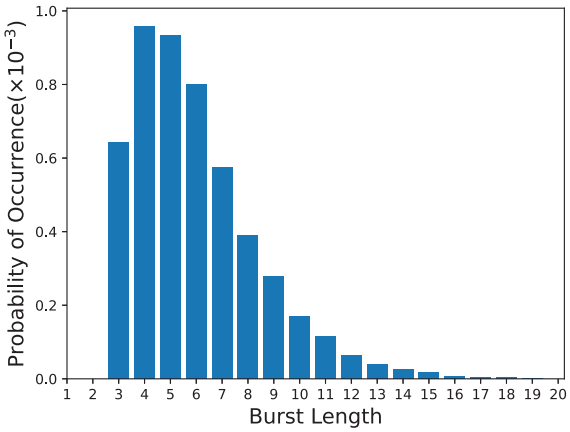


Fig. 7. In time-t, the source packets $\underline{s}(2t)$ and $\underline{s}(2t+1)$ are combined to form a single long source packet. As the source requires knowledge of source packets $\underline{s}(2t)$ and $\underline{s}(2t+1)$ in time-t, this approach becomes increasingly infeasible as t increases.



(a) Gilbert channel with $(\alpha, \beta) = (5 \times 10^{-3}, 0.5)$.



(b) Fritchman channel with $(\alpha, \beta, M) = (5 \times 10^{-3}, 0.5, 3)$.

Fig. 8. Burst histogram.

where $i_t = t \mod \nu'$. Similarly, if T is odd, in odd time slots, we have:

$$\underline{p}(2t+1) = \underline{v}^{(o,i_t)}(2t+1) + \underline{s}'(2t+1-T), \quad t \in [0:\infty). \tag{36}$$

This completes the construction. In Appendix B, we show that the construction satisfies all the desired properties of a (B_I, B_L, T)-UEP streaming code.

Remark 7 (On the requirement of B_I odd and B_L even): Constructions C and D do not generalize if B_I is even, as the (B_I, T)-recoverability requirement of high-priority source packets is not satisfied in this case (we will discuss this in Appendices A and B). If B_I and B_L are both odd, Construction B already achieves the rate $\frac{T}{T+B_{\text{eff}}}$.

D. Need for Packet-Level UEP

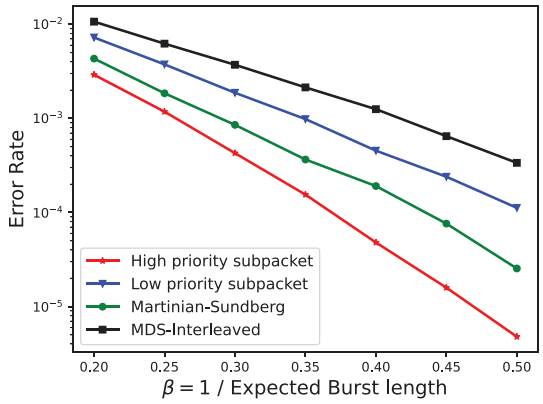
It is quite natural to ask the question whether one can simply combine two adjacent source packets (a high-priority source packet and a low-priority source packet) into a single source packet and then apply symbol-level UEP (with $\gamma = 0.5$) instead of packet-level UEP. The resulting scheme achieves a rate of $\frac{T}{T+\frac{B_I+B_L}{2}}$, which is a strict improvement over the achievable rates in the packet-level UEP setting. Thus, on a first look, it might seem that such a scheme subsumes the need for discussing packet-level UEP schemes. However, it turns out that this is not the case. In the following, we describe two possible ways of combining source packets to apply the symbol-level UEP and highlight the underlying practical issues.

Recall our problem setting (Section II), where it is assumed that a source packet $\underline{s}(t)$ arrives at the source at time-t. In Fig. 6, we consider an approach where in every odd time slot $(2t+1)$, the source packets $\underline{s}(2t)$ and $\underline{s}(2t+1)$ are combined together to form a single double-sized source packet. As can be noted from Fig. 6, using a $(\gamma = 0.5, B_I, B_L, T)$ symbol-level UEP scheme, the source packet $\underline{s}(0)$ is guaranteed to be recovered only by time-$(2T+1)$. Thus, the actual realized delay approximately doubles compared to using a (B_I, B_L, T) packet-level UEP scheme. If we consider the alternative described in Fig. 7, in each time-t, the source requires knowledge of the source packet $\underline{s}(2t)$ (ahead of time). The required lookahead grows with t, which makes the scheme practically infeasible.

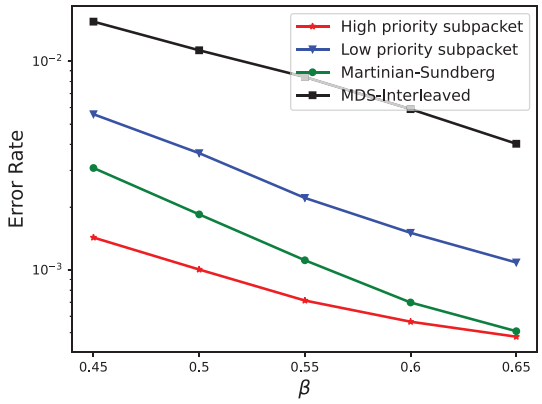
V. Numerical Studies

All the code constructions and optimality results in the paper are stated for deterministic burst erasure channel models. In this section, we study the performance of our proposed coding schemes with respect to the stochastic channel models; *Gilbert channel* and *Fritchman channel*. The Gilbert channel is a Markov model consisting of a good state (G) and a bad state (E). Let α and β denote the transition probabilities from the good state to the bad state and vice versa, respectively. In the good state, the packet-loss probability is zero, whereas in the bad state, each coded packet is lost with probability 1. Figure 8(a) shows the burst length histogram of the Gilbert channel with the parameters $(\alpha, \beta) = (5 \times 10^{-3}, 0.5)$. The Fritchman channel is a generalization of the Gilbert channel and consists of $M+1$ states; one good state denoted by G and M bad states denoted by $E_1, E_2, \ldots, E_M$. If the channel state is G in time-t, then it will either transition to E_1 with probability α or else, stay at state G with probability $1-\alpha$ in time-$(t+1)$. If the channel state is E_i for some $i \in [1:M]$ in time-t, it will transition either to E_{i+1} (let $E_{M+1} \triangleq G$) with probability β or else, stay at state E_i with probability $1-\beta$. As in the Gilbert channel, in the good state, each coded packet is received perfectly, whereas in the bad states each coded packet is lost with probability 1. Fritchman and related higher-order Markov models are commonly used to model fade durations in wireless links. Figure 8(b) shows the burst length histogram of the Fritchman channel with the parameters $(\alpha, \beta, M) = (5 \times 10^{-3}, 0.5, 3)$.

In this section, we compare our UEP code constructions with two baseline schemes; MDS-interleaved and MS coding schemes. In our comparison, we keep delay constraint and rate roughly the same for all the codes. In an MDS-interleaved coding scheme, $[n_{\text{MDS}}, k_{\text{MDS}}]$-MDS codes are diagonally



(a) Gilbert channel with $\alpha = 5 \times 10^{-3}$.

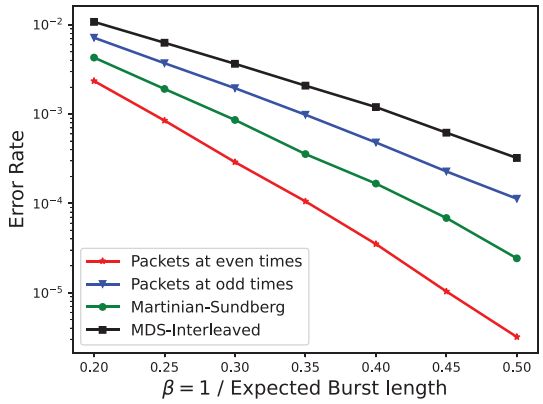


(b) Fritchman channel with $\alpha = 5 \times 10^{-3}$ and $M = 3$.

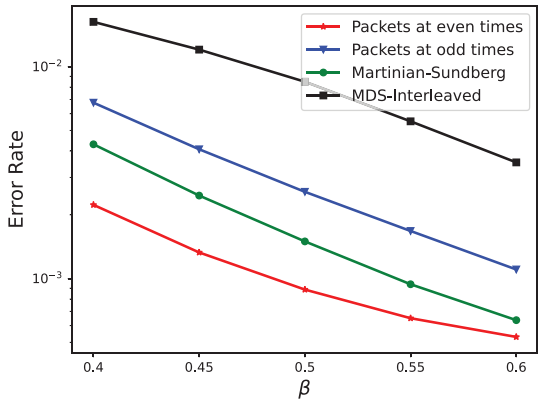
Fig. 9. Symbol-level UEP. The code parameters are given in Table IV.

TABLE IV
THE CODE PARAMETERS USED FOR THE SYMBOL-LEVEL UEP RESULTS

Channel Model	MDS-Interleaved $[n_{\text{MDS}}, k_{\text{MDS}}]$	MS $(B_{\text{MS}}, T_{\text{MS}})$	Symbol-Level UEP (γ, B_I, B_L, T)	Rate	Delay
Gilbert	$[21, 14]$	$(10, 20)$	$(2/5, 13, 8, 20)$	$2/3$	20
Fritchman	$[16, 10]$	$(10, 15)$	$(1/3, 14, 8, 15)$	$\approx 3/5$	15



(a) Gilbert channel with $\alpha = 5 \times 10^{-3}$.



(b) Fritchman channel with $\alpha = 5 \times 10^{-3}$ and $M = 3$.

Fig. 10. Packet-level UEP. The code parameters are given in Table V.

TABLE V
THE CODE PARAMETERS USED FOR THE PACKET-LEVEL UEP RESULTS

Channel Model	MDS-Interleaved $[n_{\text{MDS}}, k_{\text{MDS}}]$	MS $(B_{\text{MS}}, T_{\text{MS}})$	Packet-Level UEP (B_I, B_L, T)	Rate	Delay
Gilbert	$[21, 14]$	$(10, 20)$	$(12, 8, 20)$	$\approx 2/3$	20
Fritchman	$[17, 10]$	$(11, 16)$	$(13, 7, 16)$	$\approx 3/5$	16

interleaved to produce streaming codes. The resulting coding scheme can tolerate upto $n_{\text{MDS}} - k_{\text{MDS}}$ arbitrary packet erasures with a delay of $n_{\text{MDS}} - 1$. It is known that [2] MDS-interleaved codes, in general, do not achieve the best-possible trade-off between delay, rate and burst correction capability. In particular, for a fixed rate and delay, MS codes tolerate larger burst lengths compared to MDS-interleaved codes. A comparison of (2) with either (5) or (18) suggests that for a given rate and delay, UEP streaming codes allow for the choice of $B_L \leq B_{\text{MS}} \leq B_I$. i.e., it is possible to provide improved burst correction capability for high-priority source symbols/packets at the cost of slightly degraded performance for low-priority source symbols/packets.

In Fig. 9(a) and 9(b), we show simulation results under the symbol-level UEP scenario for the Gilbert channel and the Fritchman channel, respectively. The code parameters are summarized in Table IV. In Fig. 10(a) and 10(b), we compare our coding schemes with other baseline schemes under the packet-level UEP scenario (code parameters are provided in Table V). All the numbers reported here are generated by simulating the codes over 10^7 channel uses. As can be seen, under both scenarios, our constructions

achieve lower error rate for the high-priority part of the source stream.

Remark 8: For typical interactive applications, the maximum allowable end-to-end latency should not exceed 150 ms. For instance, in a VoIP application where each audio packet spans 10 or 20 ms of speech, considering 30 to 40 ms propagation delay for coast-to-coast communication, the maximum allowed delay T can be between 5 and 12. For a video application at 2 Mbps and packet size of 1500 bytes, a delay $T \approx 20$ may be considered. In terms of the code rate, the rate of $2/3$ is typically used in VoIP applications [18], [19].

Remark 9: Unlike MDS codes, where only the total number of erasures matter, the performance of MS and UEP streaming codes are strongly affected by how erasures are distributed. A realization of a stochastic erasure channel need not satisfy the constraints in Definition 2. If long bursts or short guard intervals (see Remark 1) occur, recovery of an erased source symbol/packet using an MS code or a UEP streaming code may become impossible. In Appendix C, we provide some additional numerical results in this regard.

REFERENCES

[1] M. Haghifam, A. Badr, A. Khisti, X. Zhu, W.-T. Dan, and J. Apostolopoulos, "Streaming codes with unequal error protection against burst losses," in *Proc. 29th Biennial Symp. Commun. (BSC)*, 2018, pp. 1–5.

[2] E. Martinian and C.-E. W. Sundberg, "Burst erasure correction codes with low decoding delay," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2494–2502, Oct. 2004.

[3] A. Badr, P. Patil, A. Khisti, W.-T. Tan, and J. Apostolopoulos, "Layered constructions for low-delay streaming codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 1, pp. 111–141, Jan. 2017.

[4] S. L. Fong, A. Khisti, B. Li, W.-T. Tan, X. Zhu, and J. Apostolopoulos, "Optimal streaming codes for channels with burst and arbitrary erasures," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 4274–4292, Jul. 2019.

[5] M. N. Krishnan, D. Shukla, and P. V. Kumar, "Rate-optimal streaming codes for channels with burst and random erasures," *IEEE Trans. Inf. Theory*, vol. 66, no. 8, pp. 4869–4891, Aug. 2020.

[6] P. Patil, A. Badr, A. Khisti, and W.-T. Tan, "Delay-optimal streaming codes under source-channel rate mismatch," in *Proc. Asilomar Conf. Signals Syst. Comput.*, 2013, pp. 2094–2099.

[7] D. Leong and T. Ho, "Erasure coding for real-time streaming," in *Proc. Int. Symp. Inf. Theory (ISIT)*, 2012, pp. 289–293.

[8] A. Badr, A. Khisti, and E. Martinian, "Diversity embedded streaming erasure codes (DE-SCo): Constructions and optimality," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 5, pp. 1042–1054, May 2011.

[9] A. Badr, A. Khisti, W.-T. Tan, and J. Apostolopoulos, "Perfecting Protection for Interactive Multimedia: A survey of forward error correction for low-delay interactive applications," *IEEE Signal Process. Mag.*, vol. 34, no. 2, pp. 95–113, Mar. 2017.

[10] M. Rudow and K. V. Rashmi, "Online versus offline rate in streaming codes for variable-size messages," in *Proc. Int. Symp. Inf. Theory (ISIT)*, 2020, pp. 509–514.

[11] J. Apostolopoulos, W.-T. Tan, and S. J. Wee, "Video streaming: Concepts, algorithms, and systems," HP Laboratories, Palo Alto, CA, USA, Rep. HPL-2002-260, 2002.

[12] E. Bedrosian, "Weighted PCM," *IRE Trans. Inf. Theory*, vol. 4, no. 1, pp. 45–49, 1958.

[13] B. Masnick and J. Wolf, "On linear unequal error protection codes," *IEEE Trans. Inf. Theory*, vol. 13, no. 4, pp. 600–607, Oct. 1967.

[14] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 1737–1744, Nov. 1996.

[15] Y. Geng, X. Zhang, C. Zhou, and Z. Guo, "Unequal error protection for real-time video streaming using expanding window Reed-Solomon code," in *Proc. Int. Conf. Image Process. (ICIP)*, 2015, pp. 3763–3767.

[16] E. Martinian and M. Trott, "Delay-optimal burst erasure code construction," in *Proc. Int. Symp. Inf. Theory (ISIT)*, 2007, pp. 1006–1010.

[17] H. D. L. Hollmann and L. M. G. M. Tolhuizen, "Optimal codes for correcting a single (wrap-around) burst of erasures," *IEEE Trans. Inf. Theory*, vol. 54, no. 9, pp. 4361–4364, Sep. 2008.

[18] J. S. Marcus, *Designing Wide Area Networks and Internetworks: A Practical Guide*. Boston, MA, USA: Addison-Wesley Professional, 1999.

[19] A. Badr, A. Khisti, W.-T. Tan, and J. Apostolopoulos, "Perfecting protection for interactive multimedia: A survey of forward error correction for low-delay interactive applications," *IEEE Signal Process. Mag.*, vol. 34, no. 2, pp. 95–113, Mar. 2017.