Learning-augmented streaming codes for variable-size messages under partial burst losses

Michael Rudow and K.V. Rashmi

2023 IEEE International Symposium on Information Theory (ISIT) | 978-1-6654-7554-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISIT54713.2023.10206559

***Abstract*—Recovering bursts of lost packets in real-time is crucial to multimedia live-streaming applications' quality-of-experience (QoE). Streaming codes optimally handle the unique aspects of loss recovery for live streaming, including (a) variable-size messages, (b) a real-time playback deadline, and (c) burst losses across multiple frames. However, existing models for streaming codes in this setting only apply to bursts that drop *all* data sent for each message. Yet in many real-world applications only some packets are lost for each message in what we call a "partial burst." We introduce a new streaming model to accommodate partial bursts. We then design a building block to construct a streaming code given any choice of how much parity to allocate for each message. Next, we present a streaming code in an offline setting (i.e., where the sizes of future messages are known) by combining (a) the building block with (b) a linear program to set the number of parity symbols per message. We then design a streaming code in an online setting (i.e., without knowledge of the future) by combining (a) the building block with (b) a learning-augmented algorithm to set the number of parity symbols per message. The constructions are approximately rate-optimal under a natural condition on the nature of feedback.**

I. Introduction

Live streaming is crucial to numerous popular applications ranging from videoconferencing to cloud gaming. The QoE of such applications depends on several factors, including bandwidth, latency, and packet loss recovery. Retransmission can recover lost packets using minimal redundancy in three steps: (a) transmission, (b) feedback, and (c) retransmission. But the latency of three one-way delays is prohibitively high for long-distance communication. Instead, erasure codes can be used for loss recovery.

The main drawback of traditional erasure codes like block codes or random linear convolutional codes is that they are inefficient at recovering burst losses in real-time. This is problematic for the many live-streaming applications that experience bursty losses. The weakness arises from the fact that conventional approaches recover all lost packets simultaneously. If a burst loss encompasses multiple messages, the first message of the burst is unavailable until after recovering the lost packets of the final message. This property can cause the latency to exceed the playback deadline of the first message. In contrast, "streaming codes" are designed to recover each message in the burst sequentially. Hence, earlier messages are available to be played sooner. This reduces the latency for recovering earlier messages in a burst. As such, streaming codes are better suited for real-time communication.

Streaming codes were first introduced by Martinian and Sundberg in [1]. Under their streaming model, during each time slot, i, a *message packet*, $S[i]$, of k symbols arrives at a sender. A *channel packet*, $X[i]$, of n symbols is sent over a burst-only packet loss model. Losses occur as bursts of b consecutive channel packets followed by guard spaces of at least τ consecutive receptions. Each message packet must be recovered within τ time slots to satisfy the playback requirement. Martinian and Sundberg introduced rate-optimal streaming codes for several parameters, then Martinian and Trott presented rate-optimal codes for the remaining parameters in [2]. Many other works have studied streaming codes for various models of communication that fix the sizes of message packets and channel packets in advance [2]–[12], [12]–[21].

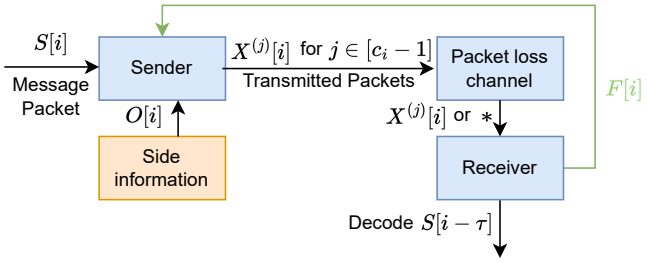

Fig. 1: Overview of the proposed streaming model. Multiple packets are transmitted over the channel for each message packet. The packet loss channel allows for partial bursts.

Motivated by the varying sizes of message packets in many live-streaming applications, such as videoconferencing, a new streaming model was introduced in [22]. Our paper focuses on this setting of variable-size message packets. Under this model, $S[i]$ and $X[i]$ are of sizes k_i and n_i, respectively, for non-negative integers k_i and n_i. Under this model, the optimal rate is adversely affected by the variability in the sizes of message packets. Spreading message symbols over multiple channel packets can mitigate the negative impact of the variability. However, spreading message symbols increases the latency to recover message packets when there are no losses—this is termed the "lossless-delay." As such, spreading across several channel packets may not be acceptable for many live-streaming applications. Rate-optimal and approximately rate-optimal streaming codes have been introduced when the lossless-delay is zero and one in [23] and [24], respectively.

The streaming model for variable message-sizes involves sending one channel packet per frame which is either received or lost. The prior works [3], [10], [11] on streaming codes that send multiple packets per message packet apply to settings where the sizes of message packets are fixed and bursts where *all* consecutive packets are lost. Unlike the existing literature on theoretical streaming codes, real-world applications frequently experience what we dub "partial bursts"

978-1-6654-7554-9/23/$31.00 ©2023 IEEE

losses of only some packets per frame. This was shown in a recent work [25] by assessing a large corpus of packet loss traces from Microsoft Teams 1:1 video calls and showing the prevalence of bursts where only some packets are lost per frame. The authors also proposed a heuristic-based design of new streaming codes suitable for such losses, and used these constructions to showcase the viability and potential benefits of streaming codes for improving the QoE for real-world live-streaming applications. The promising observations in [25] motivate a formal study of streaming codes for partial bursts.

In this paper, we generalize the streaming model for variable message-sizes to accommodate sending one or more "transmitted packets" per time slot, where only a fraction of the transmitted packets may be lost in a burst (termed *partial bursts*). The receiver sends occasional feedback to the sender to inform the choice of parameters associated with partial bursts. The model is shown in Figure 1—the component "side information" will be introduced later in Section II. In the literature on streaming codes, arbitrary losses are often considered along with burst losses. While one could proceed to account for partial bursts by introducing arbitrary losses into the loss model, doing so would allow arbitrary losses over *across multiple time slots*. Allowing such arbitrary losses would reduce streaming codes' potential to exploit the bursty structure of losses clustered within a few consecutive frames observed in [25]. Instead, this burst structure can be leveraged under our new model with partial bursts.

We apply a two-step methodology to design streaming codes for the new model. First, a building block construction to design a streaming code given any split of each message packet into (a) one component guaranteed to be recovered strictly before its playback deadline (i.e., within $(\tau-1)$ time slots), and (b) another component guaranteed to be recovered by its playback deadline (i.e., τ time slots later). Second, a policy for how to split each message packet into these two components. Our work uses a linear program to determine how to split message packets in the offline setting where the sizes of future message packets are available. Combining the linear program for splitting frames with the building block construction yields an approximately rate-optimal code under a natural condition on the nature of feedback (termed the "reset condition"). Finally, we employ a learning-augmented algorithm to determine the split to build a construction for the online setting where the sizes of future message packets are unavailable which is approximately rate-optimal under the reset condition. All proofs are available in [26].

II. System model

We now extend the streaming model with variable-size message packets from [22], as illustrated in Figure 1. There are a positive number, t, of time slots. During the ith time slot, the sender obtains a message packet, $S[i]$, of k_i independent random symbols of a finite field, $\mathbb{F}$, where $k_i \in \{0,\dots,m\}$ for some maximum value, m. We call $k_0,\dots,k_t$ the "message packet size sequence." The sender sends c_i *transmitted packets*, $X^{(0)}[i],\dots,X^{(c_i-1)}[i]$, consisting of $n_i^{(0)},\dots,n_i^{(c_i-1)}$ symbols, respectively. This change to the model allowing multiple packets to be transmitted over the channel for each message packet is a stepping stone toward adding partial bursts to the loss model. We denote the transmitted packets, number of symbols sent, and number of parity symbols as

$$X[i] = \left\langle X^{(0)}[i],\dots,X^{(c_i-1)}[i]\right\rangle,$$
$$n_i = \sum_{j=0}^{c_i-1} n_i^{(j)}$$
$$p_i = n_i - k_i,$$

respectively. The rate is defined as in [22] as the ratio of message symbols to transmitted symbols:

$$R_t = \frac{\sum_{i=0}^{t} k_i}{\sum_{i=0}^{t} n_i}.$$

Loss model: The transmitted packets are sent over a channel with bursty losses (affecting one or more consecutive time slots) followed by guard spaces where there are no losses. We introduce a new type of burst loss, called a *partial burst*. In each time slot within a partial burst, only a fraction of the transmitted packets are lost. Formally, for a partial burst of length b starting at time slot i, for each time slot l within the partial burst, $l \in \{i,\dots,i+b-1\}$, a $\ell \in (0,1]$ fraction of the transmitted packets can be lost. That is, an arbitrary $\lceil \ell c_l \rceil$ transmitted packets of $X[l]$ are lost.

Further, the length and the fraction of packets lost in partial bursts are allowed to vary over time in order to enable using feedback (based on network changes) to tune the code. Formally, a partial burst starting in time slot i encompasses b_i consecutive time slots, where b_i is a positive integer. During each time slot of the partial burst, $j \in \{i,\dots,i+b_i-1\}$, ℓ_j fraction of the transmitted packets are lost. Other than perhaps the few time slots after receiving feedback, $\ell_j = \ell_i$. The partial burst is followed by a guard space of at least τ time slots where all transmitted packets are received.

For any time slot i, we denote the c_i received packets as

$$Y[i] = \left\langle Y^{(0)}[i],\dots,Y^{(c_i-1)}[i]\right\rangle,$$

where each received packet corresponds to either receiving the corresponding transmitted packet intact or it being lost. That is, for $j \in \{0,\dots,c_i-1\}$,

$$Y^{(j)}[i] = \begin{cases} X^{(j)}[i] & \text{if } X^{(j)}[i] \text{ is received} \\ * & \text{if } X^{(j)}[i] \text{ is lost} \end{cases}.$$

Feedback: The sender occasionally receives feedback from the receiver for updating the burst length and the fraction of transmitted packets lost. During any time slot i where feedback is received, b_i and all undefined ℓ_j for $j \in \{i,\dots,i+b_i-1\}$ are updated accordingly. If there is no feedback, the parameters do not change (i.e., b_i and ℓ_{i+b_i-1} are set based on

the last received feedback). The feedback can be viewed as the receiver conservatively estimating how lossy the network conditions will be based on prior losses. At times, there could be an underestimation of the losses, and that could lead to message packets not being recovered. In videoconferencing, due to compression, video frames are typically dependent on each other. Hence not recovering a message packet can lead to several subsequent packets not being useful even though they are received intact. Thus, the receiver can send additional feedback to signal that a *reset* is needed during any time slot. This is modeled via a binary value ζ_i. It is 0 by default and set to 1 to indicate that the τ message packets before the reset need not be recovered if their transmitted packets are lost; this ensures that loss recovery does not rely on having already decoded these previous message packets. Whenever a reset is triggered, the values of b_i and $\ell_i, \ldots, \ell_{i+b_i-1}$ are also updated.

A. Encoding and Decoding

Defining encoding and decoding requires understanding what information is available during the ith time slot. In the "offline" setting, the sizes of future message packets and future feedback from the receiver are assumed to be known in advance. In contrast, the setting where this information is unavailable is dubbed "online." We introduce side information, O_i, to capture the available information. Thus, in the offline setting, O_i comprises the sizes of future message packets and future feedback. In the online setting, side information is the output of a predictive model (see Section V for details). During time slot i, the sender uses the prior message packets and side information, O_i, to encode as

$$X[i] = Enc(S[0], \ldots, S[i], O_i).$$

We consider two types of decoding: (a) decoding when there are no losses, and (b) decoding when there are losses. First, when there are no losses (or all losses have already been recovered), the *lossless-delay constraint* requires decoding each message packet, $S[i]$, within the same time slot:

$$S[i] = Dec^{(L)}\big(S[0], \ldots, S[i-1], Y[i], k_i\big).$$

Second, when there are losses, the *worst-case-delay constraint* stipulates that each message packet is recovered within τ time slots. Specifically, for any burst starting in time slot j of length b_j that encompasses time slot i,

$$S[i] = Dec\big(S[0], \ldots, S[j-1], Y[j], \ldots, Y[i+\tau], k_0, \ldots, k_{i+\tau}\big). \quad (1)$$

We note that under variable-size message packets, the sizes of the message packets are needed for decoding [22]–[24]. This is handled by adding a small header containing the sizes of the previous τ message packets. Also, our constructions do not require the full memory allowed under the model because they do not use any information about message packets and transmitted packets more than 2τ time slots in the past.

B. Notation and Conventions

Let $[n]$ denote $\{0, \ldots, n\}$. Any vector, V, is a column vector of length v. For any $I = \{j_1, \ldots, j_i\} \subseteq [n]$ where $j_1 < \ldots < j_i$, the values of V in the positions of I are denoted as $V_I = V_{j_1:j_i}$. For any time slots $i \leq j \in [t]$ and vectors $Z[i], \ldots, Z[j]$, let $Z[i:j] = Z[i], \ldots, Z[j]$, and $z_i, \ldots, z_j$ denote their sizes. Let $0^{\langle j \rangle}$ be a vector of j zeros.

Next, we define extra notation for bursts. For any time slot, i, let $\mathbb{B}_i$ be the set of time slots, j, for which a burst starting in time slot j includes time slot i (i.e., $i \in \{j, \ldots, j + b_j - 1\}$).

We now introduce some conventions followed in the rest of the paper. The final $(\tau + 1)$ message packets are assumed to be of size 0, and t is at least $(\tau + 1)$; this can be satisfied by appending $(\tau+1)$ message packets of size 0 without affecting the optimal rate. Because an integral number of transmitted packets are always sent, for each time slot, i, ℓ_i can be restricted to be a rational number q_i/h_i in simplest form. To simplify our presentation of constructions and proofs, we require $h_i \big| k_i$ and $k_i \leq m - h_i$; this can be accomplished by zero-padding $S[i]$ and increasing m by at most $(h_i - 1)$.

III. A Building Block Construction

This section develops an approximately rate-optimal construction for any parameters, τ and t, message packet size sequence, $K = (k_0, \ldots, k_t)$, and feedback, $\mathcal{L} = (\ell_0, \ldots, \ell_t), B = (b_0, \ldots, b_t)$, and $\mathrm{Z} = (\zeta_0, \ldots, \zeta_t)$. We present a building block to construct a code given any splits of the message packets into (a) a component recovered within $(\tau-1)$ time slots, and (b) a component recovered τ time slots later. Specifically, for any time slot $i \in [t - \tau]$, let w_i be the number of symbols of $S[i]$ to be recovered during time slot $(i + \tau)$, and let $W = (w_0, \ldots, w_{t-\tau})$. At a high level, $(k_i - w_i)$ symbols of $S[i]$ are received or recovered using the parity symbols of $X[i : i+\tau-1]$. Then w_i parity symbols are sent in $X[i+\tau]$ to recover the remaining lost symbols of $S[i]$. The construction is called "$\big(\tau, t, K, \mathrm{Z}, \mathcal{L}, B, W\big)$-Split Code."

Encoding (high-level description). During time slot i, $S[i]$ is partitioned into $S[i] = (U[i], V[i])$. Parity symbols $P[i]$ are defined as $P[i] = (P^{(*)}[i] + P'[i])$ where $P'[i]$ comprises symbols that are full-rank linear combinations of the symbols of $V[i - \tau], \ldots, V[i]$ and $P^{(*)}[i]$ comprises full-rank linear combinations of the symbols of $U[i - \tau]$. The key property of the linear equations and choices of how to split is that for any $j \in [i]$ and burst of length b_j starting in time slot j, the symbols of $V[j], \ldots, V[j + b_j - 1]$ can be recovered by time slot $(j+\tau-1)$. Finally, the symbols of $U[i], V[i], P[i]$ are each evenly spread over h_i transmitted packets. Figure 2 provides an overview of encoding.

Recovery (high-level description). Consider a burst starting in time slot i of length b_i where $Y[i : i + b_i - 1]$ are received. First, for $j \in \mathbb{B}_i$, the received symbols of $P[j]$ are combined with $U[j - \tau]$ (which would have been already received) to determine $P'[j]$. Then the received symbols of $P'[i : i+\tau-1]$ are used to recover $V[i : i+b_i-1]$ by solving a system of linear equations. Second, for each $j \in \{i + \tau, \ldots, i + \tau + b_i - 1\}$,

$P'[j]$ is computed using $V[j-\tau:j]$, yielding $P^{(*)}[j] = (P[j] - P'[j])$. Combining $P^{(*)}[j]$ with the received symbols of $U[j-\tau]$ suffices to recover $U[j-\tau]$.

Code construction (detailed description) time slot i. The five-step encoding process comprises: (a) initialization, (b) splitting $S[i]$ into $V[i]$ and $U[i]$, (c) defining $P[i]$ given $V[j], U[j]$ for $j < i$, (d) allocating symbols to transmitted packets, and (e) handling resets from $\zeta_i = 1$.

Initialization: For any $i \in [\tau-1]$, $U[i] = S[i], v_i = 0, p_{i+\tau} = k_i\ell_i$, and $p_i = 0$.

Splitting $S[i]$: For $i \in \{\tau, \ldots, t-\tau\}$, $S[i]$ splits into $S[i] = (U[i], V[i])$ where $u_i = 0$ if $\ell_i = 0$ and otherwise $u_i = w_i/\ell_i$. For each $j \in \mathbb{B}_i, l \in \{j, \ldots, j+b_j-1\}$, we define the number of received parity symbols for recovering $V[j : j+b_j-1]$ as $d^{(i,j,l)}$ next. Since for any $l > i$ k_l is not available, we pretend that all future message packets are recovered using parity symbols sent after time slot $(i+\tau)$ by setting $u_l = k_l = 0$ (for Equations 2 and 3 below), leading to

$$d^{(i,j,l)} = \min\left((1-\ell_l)n_l,\ k_l - u_l\ell + \sum_{r=j}^{l-1}\left(k_l - u_l\ell_r - d^{(i,j,r)}\right)\right). \tag{2}$$

To ensure $V[j : j+b_j-1]$ are recovered by time slot $(j+\tau-1)$, we require

$$\sum_{l=j+b_j}^{j+\tau-1} p_l + \sum_{l=j}^{j+b_j-1} d^{(i,j,l)} \geq \sum_{l=j}^{j+b_j-1} (k_l - u_l\ell_l). \tag{3}$$

Next, u_i is increased until Equation 3 is satisfied and $h_i | u_i$. Then $U[i]$ comprises the first u_i symbols of $S[i]$, and $V[i]$ comprises the remaining symbols. The number of parity symbols of $X[i+\tau]$ is defined as

$$p_{i+\tau} = \ell_i u_i + pad_{i+\tau}, \tag{4}$$

where $pad_{i+\tau}$ is the smallest integer to ensure $h_i | p_{i+\tau}$.[1] The symbols of $P[i+\tau]$ are not defined until time slot $(i+\tau)$.

Defining $P[i]$: To start, we define matrices that we use to define parity symbols. Let $H_0, \ldots, H_\tau$ be the parity check matrices of a systematic $[m(\tau+1), m\tau]$ m-MDS convolutional code [27], [28] (as from [3]). Let A be a $m \times m$ parity check matrix of a $[2m, m]$ systematic MDS code (e.g., Reed-Solomon). For any $i \in [\tau-1], p_i = 0$ by initialization. For $i \geq \tau$, $P[i]$ is full-rank linear combinations of the symbols of $V[i-\tau:i]$ and $U[i-\tau]$:

$$\begin{aligned} U^*[i-\tau] &= (U[i-\tau], 0^{\langle m-u_{i-\tau}\rangle},) \\ V^*[j] &= (V[j], 0^{\langle m-v_j\rangle}) \\ P^{(*)}[i] &= (AU^*[i-\tau])_{0:p_i-1} \\ P'[i] &= \sum_{j=0}^{\tau} H_j V^*[i-\tau+j] \end{aligned}$$

[1]We define $p_{i+\tau}$ once $\ell_{i+\tau}$ is known and until then pretend there is no padding for future parity symbols in Equations 2 and 3.

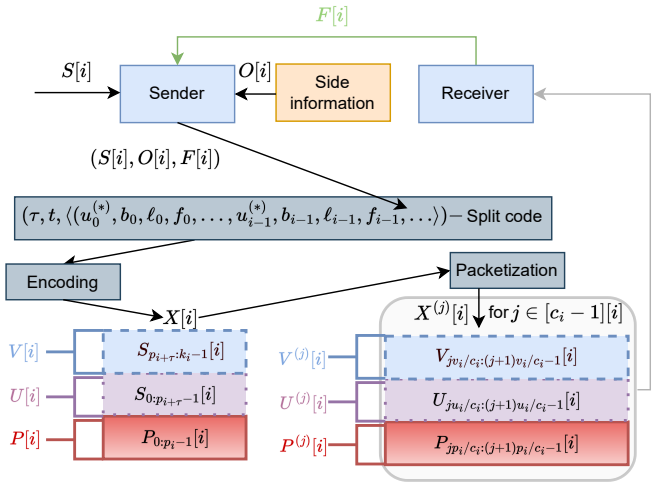

Fig. 2: Overview of encoding.

$$P[i] = (P^{(*)}[i] + P'[i]). \tag{5}$$

Allocating symbols to transmitted packets. Let $c_i = h_i$. The symbols of each of $V[i], U[i]$, and $P[i]$ are evenly allocated over c_i transmitted packets. Formally, for each $j \in [c_i - 1]$, the jth v_i/c_i, u_i/c_i, and p_i/c_i symbols of $V[i], U[i]$, and $P[i]$, are called $V^{(j)}[i], U^{(j)}[i]$, and $P^{(j)}[i]$,respectively. Then let

$$X^{(j)}[i] = (V^{(j)}[i], U^{(j)}[i], P^{(j)}[i]).$$

Resets . When $\zeta_i = 1$ the sender treats $S[i]$ as the first message packet of a length $(t-i+1)$ call and completes initialization.

Next, Theorem 1 shows that the building block construction satisfies the lossless-delay and worst-case-delay constraints.

Theorem 1: For any $\tau, t, K, \mathrm{Z}, \mathcal{L}, B, W$, the Split Code satisfies the lossless-delay and worst-case-delay constraints over the channel.

IV. Offline codes

In this section, we design an offline approximately rate-optimal construction in three steps. First, we present Algorithm 1. The algorithm identifies suitable choices for $w_0, \ldots, w_t$ using a linear program (LP) whose objective function is to minimize the number of parity symbols sent, which maximizes the rate. Second, Algorithm 1 is combined with $(\tau, t, K, \mathrm{Z}, \mathcal{L}, B, W)$-Split Code.

At a high level, the variables of the LP used in Algorithm 1 represent $w_0, \ldots, w_{t-\tau}$, which equal the number of parity symbols sent during time slots $\tau, \ldots, t$, respectively. Then $(k_i + w_{i-\tau})$ symbols are modeled as being sent during time slot i (satisfying the lossless-delay constraint). The message packets that need not be recovered due to resets are modeled as having size zero. The LP's constraints impose the worst-case-delay constraint as follows. Constraint 1 ensures that no parity symbols are sent until time slot τ. Constraint 2 ensures that a non-negative number of parity symbols are sent. For any burst starting in time slot i, Constraint 3 bounds how much useful information is received during the burst. Constraint 4 ensures recovery of enough symbols of $S[i : i+b_i-1]$ by time slot $(i+\tau-1)$ that the remaining symbols are recoverable

at their respective deadlines. Finally, Constraint 5 reflects that w_i never exceeds the number of lost symbols of $S[i]$.

Algorithm 1 Computes $\langle w_i | i \in [t] \rangle$ of an approximately rate optimal code.

Input: (τ, t, K, Z)

For $i \in [t-\tau]$:

If a reset occurs between time slot $(i+1)$ and $(i+\tau)$:

Treat k_i as 0 in the below LP.

Minimize $\sum_{i=0}^{t-\tau} p_{i+\tau}^{(LP)}$ subject to:

1) $\forall j \in [\tau-1], p_j^{(LP)} = 0.$
2) $\forall j \in [t-\tau], p_{j+\tau}^{(LP)} \geq 0.$
3) $\forall i \in [t-\tau], l \in \{i, \ldots, i+b_i-1\}$,

$$0 \leq d_{i,l} \leq \min((p_l^{(LP)} + k_l)(1-\ell_l), \quad k_l - p_{l+\tau}^{(LP)} + \sum_{r=i}^{l-1}(k_r - p_{r+\tau}^{(LP)} - d_{i,r})$$

4) $\forall i \in [t-\tau]$,

$$\sum_{l=i+b_i}^{i+\tau-1} p_l^{(LP)} + \sum_{l=i}^{i+b_i-1} d_{i,l} \geq \sum_{l=i}^{i+b_i-1} (k_l - p_{l+\tau}^{(LP)}) \quad (6)$$

5) $\forall j \in [t-\tau], k_j \ell_j \geq p_{j+\tau}^{(LP)}.$

For $i \in [t-\tau]$:

If a reset occurs between time slot $(i+1)$ and $(i+\tau)$:

Set $p_{i+\tau}^{(LP)} = \ell_i k_i$.

Output: $\left\langle p_i^{(LP)} \middle| i \in [t] \right\rangle$

Theorem 2 below shows that combining Algorithm 1 with the building block construction (Section III) yields an approximately rate-optimal code subject to the following condition on the reset bit in feedback.

Reset condition on feedback : A reset must occur whenever an increasing fraction of transmitted packets could be lost. Formally, for any $i \in [t-\tau] \setminus \{0\}$ where feedback increases ℓ_i, ζ_i must be set to 1.

Theorem 2: For any $\tau, t, K, \mathrm{Z}, \mathcal{L}, B$, if Algorithm 1 outputs $\langle w_i | i \in [t] \rangle$, then the rate of the corresponding Split Code is less than the optimal rate under the reset condition on feedback by at most

$$\big(\sum_{i=0}^{t-\tau}(2\tau + q_i + h_{i+\tau} - 4)\big)/\big(\sum_{i=0}^{t} k_i\big). \quad (7)$$

As an example of applying Theorem 2, consider a videoconferencing call at 2000 kbps and 30 fps. Suppose the field size is 2^{32}, for $i \in [t] \ell_i \in \{j/8 \mid j \in [8]\}$, and $\tau \leq 5$. Then the rate of the Split Code is within 0.01 of optimal.

V. Online approximately rate-optimal codes

We now present an online approximately rate-optimal construction, dubbed the "$\big(\tau, t, K, \mathrm{Z}, \mathcal{L}, B, W^{(O)}\big)$-Split ML Code." During the ith time slot, an ML model provides side information, $O_i = w_i$, to determine how to split the ith frame in the building block construction (Section III). If $\ell_i = 0$ then $X[i]$ is received, so O_i must be 0. Otherwise, to ensure O_i can be used by the building block construction, we require it to be (a) sufficiently large (i.e., setting $u_i = O_i/\ell_i$ satisfies Equation 3), and (b) padded to be divisible by q_i.

Our result requires a few terms. Let the outputs of the ML model over time slots $0, \ldots, (t-\tau)$ be $W^{(O)} = O_0, \ldots, O_{t-\tau}$. For $i = 0, \ldots, (t-\tau)$, let $W_i^{(Opt)}$ be the set of optimal values for $p_{i+\tau}^{(LP)}$ in Algorithm 1 with additional constraints that the variables corresponding to earlier time slots are set according to $W^{(O)}$ (i.e., for $j \in [i-1]$ $p_{j+\tau}^{(LP)} = W_j^{(O)}$). For $i \in [t-\tau]$, the regret of the outputs of the ML model compared to the optimal values is

$$\mathcal{R}_i = \min_{w^{(Opt)} \in W_i^{(Opt)}} \big|O_i - w^{(Opt)}\big|, \mathcal{R}_{[t]} = (\mathcal{R}_0, \ldots, \mathcal{R}_t) \quad (8)$$

For an arbitrary message packet size sequence and feedback chosen offline without access to $W^{(O)}$, let $R^{(opt)}$ be the offline optimal rate under the reset condition on feedback from Section IV and $R^{(on)}$ be a random variable (over the predictions of the ML model) reflecting the rate of the Split ML Code.

Theorem 3: Consider any $\tau, t, K, \mathrm{Z}, \mathcal{L}, B, W^{(O)}$ and $\epsilon, \delta, \epsilon_\dagger \in (0,1)$. Suppose for $i \in [t]$ that $\mathrm{E}[\mathcal{R}_i] \leq \epsilon k_i$ and $t > log(1/\delta)/(2\epsilon_\dagger^2)$. Then with probability at least $(1-\delta)$,

$$R^{(opt)} - R^{(on)} \leq \epsilon + \big(\sum_{i=0}^{t} \epsilon_\dagger + 2\tau + h_i + q_i - 4\big)/\big(\sum_{i=0}^{t} k_i\big). \quad (9)$$

Consider the example of a videoconferencing call discussed after Theorem 2. If the call is sufficiently long, with probability $(1-\delta)$, $R^{(on)}$ is within $(0.01 + \epsilon + 0.00048 \cdot \epsilon_\dagger)$ of optimal.

VI. Conclusion

Motivated by live-streaming applications experiencing partial bursts of only some packets per message, our work introduces the first streaming model to accommodate such losses. We then present a streaming code construction that is approximately rate-optimal under a natural (reset) condition in two steps. First, we use a learning-augmented algorithm to split message packets into (a) a component recovered strictly before its decoding deadline, and (b) a component recovered at its decoding deadline. Second, we introduce a building block construction to design a code given the choice of how to split message symbols that is approximately rate-optimal code under the reset condition. Future work can build upon our results in three main directions: (a) construct explicit predictive models to split message packets, (b) construct explicit predictive models to estimate the channel parameters (i.e., the feedback), and (c) combine our methodology with that of [24] to spread message symbols over $(\tau_L + 1) \geq 2$ time slots to alleviate the negative impact on the rate of the variability in the sizes of the message packets.

Acknowledgment

This work was funded in part by an NSF grant (CCF-1910813).

REFERENCES

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

[2] E. Martinian and M. Trott, "Delay-optimal burst erasure code construction," in *2007 IEEE International Symposium on Information Theory*, June 2007, pp. 1006–1010.

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

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

[5] M. N. Krishnan and P. V. Kumar, "Rate-optimal streaming codes for channels with burst and isolated erasures," in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 1809–1813.

[6] 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, 2020.

[7] E. Domanovitz, S. L. Fong, and A. Khisti, "An explicit rate-optimal streaming code for channels with burst and arbitrary erasures," vol. 68, no. 1, 2022, pp. 47–65.

[8] A. Badr, A. Khisti, and E. Martinian, "Diversity embedded streaming erasure codes (de-sco): Constructions and optimality," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 5, pp. 1042–1054, May 2011.

[9] N. Adler and Y. Cassuto, "Burst-erasure correcting codes with optimal average delay," *IEEE Transactions on Information Theory*, vol. 63, no. 5, pp. 2848–2865, May 2017.

[10] D. Leong and T. Ho, "Erasure coding for real-time streaming," in *2012 IEEE International Symposium on Information Theory Proceedings*, July 2012, pp. 289–293.

[11] D. Leong, A. Qureshi, and T. Ho, "On coding for real-time streaming under packet erasures," in *2013 IEEE International Symposium on Information Theory*, July 2013, pp. 1012–1016.

[12] S. L. Fong, A. Khisti, B. Li, W.-T. Tan, X. Zhu, and J. Apostolopoulos, "Optimal streaming erasure codes over the three-node relay network," *IEEE Transactions on Information Theory*, vol. 66, no. 5, pp. 2696–2712, 2020.

[13] A. Badr, A. Khisti, W.-t. Tan, X. Zhu, and J. Apostolopoulos, "Fec for voip using dual-delay streaming codes," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[14] Z. Li, A. Khisti, and B. Girod, "Correcting erasure bursts with minimum decoding delay," in *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Nov 2011, pp. 33–39.

[15] Y. Wei and T. Ho, "On prioritized coding for real-time streaming under packet erasures," in *Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on*. IEEE, 2013, pp. 327–334.

[16] P.-W. Su, Y.-C. Huang, S.-C. Lin, I.-H. Wang, and C.-C. Wang, "Random linear streaming codes in the finite memory length and decoding deadline regime—part i: Exact analysis," *IEEE Transactions on Information Theory*, vol. 68, no. 10, pp. 6356–6387, 2022.

[17] S. Emara, F. Wang, I. Kaplan, and B. Li, "Ivory: Learning network adaptive streaming codes," pp. 1–10, 2022.

[18] S. Emara, S. L. Fong, B. Li, A. Khisti, W.-T. Tan, X. Zhu, and J. Apostolopoulos, "Low-latency network-adaptive error control for interactive streaming," *IEEE Transactions on Multimedia*, vol. 24, pp. 1691–1706, 2021.

[19] M. N. Krishnan, G. K. Facenda, E. Domanovitz, A. Khisti, W.-T. Tan, and J. Apostolopoulos, "High rate streaming codes over the three-node relay network," in *2021 IEEE Information Theory Workshop (ITW)*. IEEE, 2021, pp. 1–6.

[20] E. Domanovitz, A. Khisti, W.-T. Tan, X. Zhu, and J. Apostolopoulos, "Streaming erasure codes over multi-hop relay network," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 497–502.

[21] M. Haghifam, M. N. Krishnan, A. Khisti, X. Zhu, W.-T. Tan, and J. Apostolopoulos, "On streaming codes with unequal error protection," *IEEE J. Sel. Areas Inf. Theory*, 2021.

[22] M. Rudow and K. Rashmi, "Streaming codes for variable-size messages," *IEEE Transactions on Information Theory*, pp. 1–1, 2022.

[23] ——, "Online versus offline rate in streaming codes for variable-size messages," 2023, pp. 1–1.

[24] ——, "Learning-augmented streaming codes are approximately optimal for variable-size messages," in *2022 IEEE International Symposium on Information Theory (ISIT)*, 2022, pp. 474–479.

[25] M. Rudow, F. Y. Yan, A. Kumar, G. Ananthanarayanan, M. Ellis, and K. Rashmi, "Tambur: Efficient loss recovery for videoconferencing via streaming codes," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. Boston, MA: USENIX Association, Apr. 2023, pp. 953–971. [Online]. Available: https://www.usenix.org/conference/nsdi23/presentation/rudow

[26] M. Rudow and K. Rashmi, "Learning-augmented streaming codes for variable-size messages under partial burst losses," http://www.cs.cmu.edu/~rvinayak/papers/learning_augment_streaming_codes_under_partial_bursts_ISIT_2023.pdf, 2023.

[27] E. Gabidulin, "Convolutional codes over large alphabets," in *Proc. Int. Workshop on Algebraic Combinatorial and Coding Theory*, 1988, pp. 80–84.

[28] H. Gluesing-Luerssen, J. Rosenthal, and R. Smarandache, "Strongly-mds convolutional codes," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 584–598, 2006.