Efficient loss recovery for videoconferencing via streaming codes and machine learning

Michael Rudow

CMU-CS-23-118

May 2023

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by Portuguese Foundation for Science and Technology and the National Science Foundation under award numbers 1850483, 19101813, 1901410, and 1943409, as well as the Carnegie Mellon University Graduate Student Assembly/Provost Conference Funds.

Keywords: Streaming codes, machine learning, erasure codes, learning-augmented algorithm

Abstract

Packet loss degrades the quality of experience (QoE) of live communication. The standard approach to recovering lost packets for long-distance communication is forward error correction (FEC). Conventional methods for FEC for real-time applications are highly inefficient at protecting against bursts of losses. Yet such bursts frequently arise in practice. Bursts can be tamed with less redundancy via a new class of theoretical FEC schemes, called "streaming codes," designed to communicate a sequence of frames over a bursty packet loss channel in real-time. Existing streaming codes apply when all frames are of the same fixed size. However, many applications, including videoconferencing, involve sending *compressed* frames whose sizes fluctuate dramatically. This thesis presents a generalized model for streaming codes that incorporates frames of variable sizes, studies the fundamental limits on the optimal rate for the new model, designs new high-rate streaming codes using machine learning and coding theory, and integrates streaming codes into a videoconferencing application to assess their positive impact on the QoE.

To start, we examine the fundamental limits on the rate for "offline" communication, wherein the sizes of all future frames are known. We show that the variability in the sizes of frames (a) induces a new trade-off between the rate and the decoding delay under lossless transmission and (b) impacts the optimal rate of transmission. We then design rate-optimal streaming codes for the practically relevant "online" setting (i.e., where the sizes of the future frames are unknown) for two broad parameter regimes. We show that online schemes cannot match the optimal rate of offline schemes for all remaining parameter regimes because the optimal way to spread the data over multiple transmissions to alleviate the variability depends on the future arrival pattern. To address this shortcoming, we combine algebraic coding techniques with a learning-augmented algorithm for spreading frame symbols to design the first approximately rate-optimal streaming codes for a range of important parameter regimes for practical applications.

However, many real-world applications experience what we dub "partial bursts" losses of only some packets per frame, unlike the existing model, which assumes all or no packets are lost for each frame. To address this gap, we introduce a new streaming-codes-based approach to videoconferencing called *Tambur*. When assessed over emulated networks, Tambur improves several key metrics of QoE compared to conventional methods (e.g., it reduces the median frequency of freezes by 26%). We then extend the theoretical streaming model to accommodate partial bursts and design an online approximately rate-optimal streaming code. The code combines (a) a building block construction given any choice of how much redundancy to allocate per frame with (b) a learning-augmented algorithm to allocate redundancy per frame.

Acknowledgments

I would not be earning my Ph.D. without the support of my family and friends. I want to thank my parents, Bill and Jill, my sister, Alex, and my friend, Daniel, for their love, support, and for pretending to pay attention when I told them about my research. I also would like to thank Monty Abello, Matt Butrovich, Nate Chodosh, Graham Gobieski, Isaac Grosof, Ananya Joshi, Jack Kosaian, Roger Lyengar, Francisco Maturana, Andrii Riazanov, Han Zhang, Giulio Zhou, and countless other friends for making Pittsburgh my home. This Ph.D. was an Odyssey; thank you for sailing with me into the horizon!

I would also like to acknowledge my advisor, Professor Rashmi Vinayak, for her tutelage throughout my Ph.D. journey. Rashmi taught me how to find and solve problems with real-world impact. She guided me through research across several different areas, helping me to become a well-rounded researcher capable of tackling problems in new domains. Rashmi encouraged me to persevere during research droughts and fulfill my potential during breakthroughs. She also provided me six years' worth of feedback on my oral and written communication to teach me to become a better communicator. I will carry her lessons with me beyond CMU, ever grateful for her mentorship.

My journey at CMU would not have happened without the support I received as an undergrad. I thank Eric Allender for his tutelage during a Research Experience for Undergraduates the summer after my junior year, which inspired me to pursue a Ph.D. I also would like to thank my master's thesis advisors, Brett Hemenway Falk and Nadia Heninger, for mentoring me throughout my master's thesis and encouraging me to complete a Ph.D.

I want to additionally thank Mor, Rashmi, and Anupam for their feedback when I was a TA for their classes, which has taught me to be a better teacher and communicator. I also would like to acknowledge Andrii Riazanov, Venkatesan Guruswami, and Bernhard Haeupler for serving on my writing skills committee, as well as Mor Harchol-Balter, Danny Sleator, and Pedro Paredes for serving on my speaking skills committee. I would like to thank Venkatesan Guruswami, Anupam Gupta, and Ryan O'Donnell for serving on my thesis committee.

I would like to acknowledge the current members and alumni of the TheSys lab: Jack Kosaian, Francisco Maturana, Juncheng (Jason) Yang, Sanjith Athlur, and Saurabh Kadekodi for their support and helpful feedback on my presentations. I would like to thank Jack Kosaian for providing the video trace used in Figure 1.2. I would like to thank Francisco for his helpful feedback on several of my papers and talks, which was invaluable for refining them. I would also like to acknowledge my collaborators, Ganesh Ananthanarayanan, Neophytos Charalambides, Martin Ellis, Venkatesan Guruswami, Alfred O. Hero, III, Abhishek Kumar, and Francis Y. Yan.

Finally, I would like to thank Deb Cavlovich, Matt McMonagle, Jenn Landefeld, and the rest of CMU's staff for all of their help with the administrative component of the Ph.D.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Real-time communication with a high quality of experience (QoE) is critical for many pervasive streaming applications, including VoIP and videoconferencing. For example, the quality of videoconferencing calls dictates the effectiveness of remote meetings [24], which are now ubiquitous. Video quality depends on several key performance indicators, such as freeze, bandwidth, packet loss, and latency [21, 40, 61]. At a high level, live-streaming applications such as videoconferencing involve a sender transmitting packets of information to a receiver over a lossy channel.

Recovering lost packets is crucial to the quality of communication [84]. Losing even a single packet may prevent rendering the frame. It may also prohibit rendering multiple future frames (i.e., causing freeze) because the compression introduces inter-frame dependencies. Due to this, it is common to handle packet losses at the application level. The two broad viable solutions are retransmissions and forward error correction (FEC). Both approaches transmit redundant data. Consequently, there is a trade-off between bandwidth allocated for redundancy and transmitting original data. Furthermore, real-time communication applications must recover lost packets within a strict latency—preferably less than 150 ms [84]—to meet the real-time playback requirement.

Retransmission involves minimal redundant data since it only resends the lost packets. Hence, it is preferred whenever applicable [83]. However, retransmission is suitable only for scenarios with short round trip times (e.g., short-distance communication) because of the extra delay of feedback and retransmission. When the round trip time is high (e.g., long-distance communication), the three-way delay of transmission, feedback, and retransmission exceeds the maximum tolerable latency [10]. When retransmission is not viable, applications must use Forward Erasure Correction (FEC) to recover lost packets within an acceptable latency.

One natural coding-based approach uses traditional codes, such as block codes, to recover lost packets. However, such coding schemes are ill-suited for the low-delay communication setting of live-streaming applications. The loss patterns faced in streaming applications are correlated (i.e., bursty). Nevertheless, conventional block codes, such as maximum-distance-separable (MDS) codes, are inefficient for recovering from burst losses within a strict decoding delay for the following reason. Using MDS block codes necessitates recovering all packets lost as a burst *simultaneously*. As a result, all lost packets must be decoded by the playback deadline of the first lost packet—an unnecessarily stringent requirement for most lost packets. The redundancy

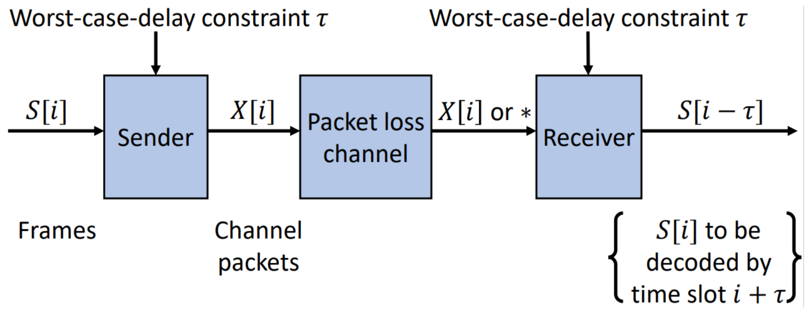

Worst-case-delay constraint τ
Worst-case-delay constraint τ
$S[i]$
Sender
$X[i]$
Packet loss channel
$X[i]$ or $*$
Receiver
$S[i-\tau]$
Frames
Channel packets
$S[i]$ to be decoded by time slot $i+\tau$

Figure 1.1: Overview of the model for streaming codes.

sent by the deadline of the final lost packet is wasted, penalizing the rate. Finally, it is possible to convert an MDS code into a burst-correcting code via the standard technique of interleaving. However, such an approach is inapplicable to the live streaming setting, as it would violate the low-delay requirement.

Convolutional coding schemes tailored to the live communication setting can outperform traditional code constructions. Martinian and Sundberg demonstrated this fact in [62] by formalizing the "streaming model" of live communication and introducing specialized codes —called "streaming codes"—with significantly higher rates than traditional block codes. During each time slot, i, a "frame," denoted as $S[i]$, of size k arrives at a sender. The sender then transmits a "channel packet," denoted as $X[i]$, of size n to a receiver over a burst-only loss channel. Bursts are of length b and are followed by guard spaces of receptions. The sender must recover $S[i]$ by time slot $(i + \tau)$. Streaming codes recover lost symbols from each frame τ time slots later, leading to significantly higher rates than alternatives that recover all lost symbols together by τ time slots after the *first* frame for which the symbols are lost [62]. Figure 1.1 presents an overview of the model, with the sender, channel, and receiver appearing in blue.

Numerous works [3, 7, 8, 9, 11, 12, 13, 27, 33, 35, 36, 37, 44, 49, 50, 52, 56, 57, 58, 63, 75, 82, 85] have designed streaming codes for various settings where the sizes of frames and channel packets are fixed in advance. These regimes are suitable for applications that send fixed quantities of data; for example, VoIP sends uncompressed audio packets. The benefits of streaming codes for *VoIP* applications have been studied using simulated losses under theoretical loss models, such as the Gilbert-Elliott channel [31] and over traces [13, 33], wherein each frame is of the same fixed size.

In contrast, many applications, such as videoconferencing, transmit *variable* amounts of data. The variability arises by compressing frames before transmission to reduce the communication load. For example, we demonstrate the variability of compressed frame sizes for a live video trace transmitted by Facebook Live in Figure 1.2. Variable-sized input data is incompatible with the previously studied streaming model, motivating the need for a new model.

Motivating the problem: Given the dual importance of bandwidth and loss recovery, streaming codes appeal to improve the QoE of live-streaming applications. However, there are two main challenges to using streaming codes.

1. Existing streaming codes are incompatible with the variable sizes of compressed frames of live-streaming applications such as videoconferencing.

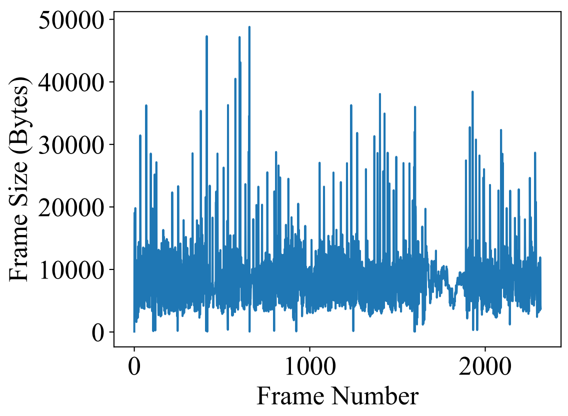


Figure 1.2: Frame size variability in a live video trace collected from the Facebook Live application (for a 2000 Kbps live video).

2. Packet loss characteristics for live streaming applications with variable-sized frames have not yet to be shown to be suitable for the general framework of streaming codes.

Contributions of the thesis : The following five contributions of this thesis address these challenges to improve the QoE of live-streaming applications via streaming codes.

1. We introduce a new model of streaming codes for variable-size frames and evaluate the fundamental limits on the optimal rate. One consequence of the new model is distinguishing between the "offline" setting, where the sizes of future frames are known, and the "online" setting, where such information is unavailable.
2. We design online rate-optimal streaming codes for the new model for two broad parameter regimes (termed Regime 1 and Regime 2) and show that online streaming codes cannot match the optimal offline rate for all other parameter regimes.
3. We present learning-augmented streaming codes with approximately optimal rates for the new model for the most practically-relevant broad parameter regime outside Regime 1 and Regime 2.
4. All data sent for a frame is lost or received under the above streaming model, but real-world networks frequently involve "partial bursts" wherein only some packets sent for a frame are lost. As a first step to exploit partial bursts, we use a heuristic to design new streaming codes for partial bursts. We then integrate the new streaming codes into the live-streaming application stack and evaluate their suitability to improve the QoE over emulated networks. We establish benefits for several key metrics of the QoE, such as reducing the median frequency of freezes by 26% compared to conventional approaches.
5. We generalize the streaming model to accommodate partial bursts and then design new approximately rate-optimal learning-augmented streaming codes for the generalized model.

1.1 Streaming codes for variable-size frames (Chapter 2)

We present a generalized model for streaming codes that accommodates frames of variable sizes. The proposed streaming model differs from that of fixed-size frames in two key ways.

First, while there are rate-optimal schemes that send each frame in the corresponding channel packet for the setting of fixed-size frames, spreading frame symbols over multiple channel packets is advantageous in the setting of variable-size frames. This is because sending a large frame within a single channel packet leads to many lost symbols when that channel packet is lost. Spreading frame symbols intelligently reduces the maximum number of frame symbols lost in a burst—a lower bound on how much redundancy is needed. In contrast, when all frames are the same size and are sent in the corresponding channel packets, all bursts drop the same number of frame symbols. As such, spreading frame symbols over multiple channel packets does not offer an advantage. We capture the new trade-off between the rate of the code and the minimum decoding delay (i.e., the delay for decoding a frame when all channel packets are received) via a new parameter in the model, which we term "lossless-delay (τ_L)." Specifically, when there are no losses, the receiver must decode each frame with a delay of τ_L.

Second, the variability in the sizes of frames *negatively* impacts the optimal rate, which is never higher than that of the setting where frames have fixed sizes. We capture the optimal rate's dependence on the variability by determining the least upper bound and greatest lower bound on the optimal rate for arbitrary sequences of sizes of frames. However, the gap between these values is wide in many settings, prompting the need to better characterize the optimal rate for any specific sequence of sizes of frames. Thus, we introduce algorithms to compute tighter upper and lower bounds on the optimal rate for any given sequence of sizes of frames.

The results of this section are published in [78].

1.2 Online rate-optimal streaming codes (Chapter 3)

The bounds on the optimal rate for any given sequence of sizes of frames apply to "offline" codes, which have access to the sizes of all frames, including future ones. In contrast, real-world streaming applications operate in an "online" setting where the sizes of future frames are unknown. By using future information, optimal offline constructions can always match and potentially significantly exceed the rate of online ones. One key challenge in realizing the benefits of spreading for online codes is determining how best to spread frame symbols over one or more channel packets, even though future frames' sizes are inherently variable and unknown.

This thesis designs the first rate-optimal online coding schemes for two classes of parameter settings. In "Regime 1," b and τ may take any values while $\tau_L = 0$, necessitating that all constructions recover each frame immediately under lossless conditions. This broad regime is well-suited for applications that require minimal latency during lossless conditions and can tolerate extra latency only during occasional losses.

The rate-optimal construction is systematic and sends each frame in the corresponding channel packet. During each time slot, i, we combine two new methodologies to alleviate the variability. (a) We apply a greedy paradigm for delaying transmitting the parity symbols associated with $S[i]$ until the time slot $(i + \tau)$. (b) We define the number of parity symbols to be sent in

$X[i+\tau]$ while deferring defining the parity symbols themselves until the time slot $(i+\tau)$ to make use of frames $S[i+1],\ldots,S[i+\tau-1]$. The construction is rate-optimal, even for the offline setting. As such, the results show that non-systematic schemes provide no advantage.

In "Regime 2," $\tau_L = (\tau - b)$ and $b|\tau$, so τ_L has its maximum value. Here, we show that a simple scheme that encodes each frame separately matches an upper bound on the rate (see Section 2.3.2). Thus, online coding schemes can match the rate of optimal offline coding schemes for two broad parameter regimes even though knowledge about the sizes of future frames appears advantageous. In addition, we demonstrate that online coding schemes necessarily have lower rates than optimal offline coding schemes for *all* remaining parameter regimes.

The results of this section are published in [79].

1.3 Learning-augmented approximately rate-optimal streaming codes (Chapter 4).

Maintaining a small value of τ_L is crucial for latency-sensitive applications, as the delay of τ_L extra time slots may be incurred for decoding *every* frame. However, Regime 1 (i.e., $\tau_L = 0$) penalizes the optimal rate by prohibiting spreading frame symbols to smooth out the variability. Next, we design approximately rate-optimal streaming codes for "Regime 3" of $\tau_L = 1$. Regime 3 imposes the minimum extra latency (i.e., one frame) to allow spreading frame symbols over *multiple* channel packets to significantly mitigate the adverse effect of variability of the sizes of frames on the rate. As such, Regime 3 reflects a good compromise between the latency and optimal rate for practical live-streaming applications.

We first consider the offline setting and decompose the code design into two distinct challenges. First, how can we best spread frame symbols over channel packets? Second, how can we send the minimum necessary number of parity symbols to ensure that each frame is decoded in time, given any choice of how to spread frame symbols? We use an integer program offline to determine how to spread frame symbols over channel packets optimally. We then introduce a building block for constructing a rate-optimal streaming code for *any given* choice of how to spread frame symbols over channel packets.

One final challenge remains: how can we construct a rate-optimal *online* streaming code? We address the problem by *combining machine learning with tools from algebraic coding theory.* We take a learning-based approach, relying on techniques similar to empirical risk minimization to convert the optimal offline solution into an approximately optimal online one that maximizes the expected rate. Our proposed method determines how to spread symbols online, and then the building block construction is applied. Our methodology can be viewed as using a "learning-augmented algorithm"—a topic that has recently surged to prominence, tackling problems in other domains, such as caching [59], metric task systems [6], bloom filters [64], learned index structures [48], scheduling [55], etc. [5, 16, 46, 47, 65]. However, to the best of our knowledge, the powerful paradigm of learning-augmented algorithms has not been applied to design coding schemes until now

The results of this section are published in [77].

1.4 Streaming codes for real-world videoconferencing (Chapter 5)

Two main obstacles prevent using the streaming codes discussed above for live-streaming applications. First, these streaming codes assume that a frame is entirely lost or received—such a loss reflects worst-case conditions. Consequently, the streaming model introduced in Chapter 2 models sending all symbols of a frame in a single channel packet. However, live-streaming applications (e.g., videoconferencing) will often distribute a large transmission over multiple packets, where only some may be lost. The pessimistic loss model demands extra redundancy to recover *all* symbols of multiple frames lost in a burst, which can negate the potential advantage in reducing the bandwidth overhead. The parameters of this pessimistic loss model are used to set the amount of redundancy employed by existing streaming codes. Second, streaming codes' viability to improve the QoE for real-world live-streaming applications has yet to be established for several key metrics of the QoE (e.g., video freeze, frequency of rendering frames, etc.). In addition, the packet loss characteristics of real-world videoconferencing calls have not been established as suitable for streaming codes.

To handle the first challenge, we will introduce Tambur, a new communication scheme for bandwidth-efficient loss recovery for live-streaming applications.[1] Tambur comprises two components:

- a new streaming code that adapts the theoretical framework discussed in Chapter 3 to overcome its limitations for real-world live-streaming applications;
- a machine learning (ML) model to take a predictive decision on the bandwidth allocated to streaming codes.

To address the second challenge, we analyze packet traces collected from thousands of video calls from *Microsoft Teams* and present three key observations:

1. Bursts of packet losses frequently arise.
2. Losses are frequently followed by a guard space of several frames with no losses.
3. Teams uses a significant bandwidth overhead to recover lost packets in real time, depleting the bandwidth left for the original data.

We implement and integrate Tambur, several baselines (Block-Within and "Block-Multi," a block code across multiple frames) and several variants of Tambur ("Tambur-full-BW," which matches the bandwidth overhead of Block-Within and "Tambur-0.9," which reduces the bandwidth overhead more at the cost of recovering fewer frames) with a videoconferencing benchmark platform. We then evaluate the schemes over an emulated network to assess the impact on the QoE (§5.4.4). Fig. 1.3 shows how Tambur, Tambur-full-BW, and Tambur-0.9 reduce the frequency of video freezes by an average of 26%, 29%, and 17%, respectively, compared with the better of Block-Within and Block-Multi. These benefits highlight that Tambur improves the QoE, as it has been shown [66, 73, 86] that video freezes have a detrimental effect on user engagement.

The results of this section are published in [81].

[1]Named to convey that streaming codes *Tam*e *Bur*sts.

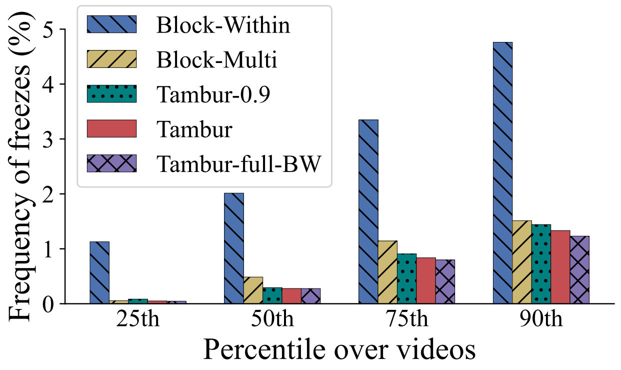


Figure 1.3: Tambur reduces the ratio of frozen frames to total frames per-video by 78% and 26% compared to Block-Within and Block-Multi, respectively, at a lower bandwidth overhead.

1.5 Learning-augmented streaming codes for variable-size frames under partial bursts (Chapter 6)

Our analysis of packet loss traces from Microsoft Teams shows that multiple packets are sent per frame and, often, only some of them are lost (as previewed in Section 1.4). While as a first step we used a heuristic-based design to demonstrate the viability and potential benefits of streaming codes in real-world applications, a larger goal is to take these learnings back into the theoretical model and improve it.

There are a few works [11, 56, 57] on theoretical streaming codes that send multiple packets per frame. However, they apply to settings where the sizes of frames are fixed and bursts where *all* consecutive packets are lost.

Chapter 6 generalizes the streaming model to accommodate sending one or more "transmitted packets" for each encoded frame where only a fraction of the transmitted packets are lost in a burst.We focus on the setting where $\tau_L = 0$; recall that this regime is important to practical videoconferencing applications because it ensures the smallest possible delay when there are no losses. We employ a two-step methodology for designing streaming codes for the new model. First, a building block construction to design a streaming code given any split of each frame 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 at its playback deadline (i.e., τ time slots later). Second, a policy for how to split each frame. This thesis uses a linear program to determine a split in the offline setting. Combining the linear program for splitting frames with the building block construction yields an approximately rate-optimal code. Online constructions that come negligibly close to matching the rate of optimal offline constructions are introduced for three parameter regimes: (a) Regime b_1 where $\tau > 1$ and $b_i = 1$ for all $i \in [t]$, (b) Regime b_τ where $b_i = \tau$ for all $i \in [t]$, and (c) Regime $b_{\tau+1}$ where $b_i > \tau$ for all $i \in [t]$ and $1 \notin \mathcal{L}$. For all parameter regimes outside of Regime b_1, Regime b_τ, and Regime $b_{\tau+1}$, we establish a nontrivial gap between the optimal rate of online codes and offline codes. The result is surprising because there is no such gap when the lossless-delay is 0 in the model without partial losses (i.e., *all* data sent during a time slot is either lost or received) (see Chapter 3). Finally, we employ a learning-augmented algorithm to determine the split to build an approximately rate-optimal construction for the online setting where the sizes of future frames are unavailable (Chapater 6.6).

The results of this section are published in [80].

1.6 Outline

The remainder of this document is summarized as follows:

- Chapter 2 introduces the new streaming model for variable-size frames, fundamental limits on the rate for the new model, and designs the first online rate-optimal streaming code for a parameter regime (i.e., Regime 2).
- Chapter 3 designs the first online rate-optimal streaming code for the most practically-relevant broad parameter regime (i.e., Regime 1) and shows that no online streaming code can match the optimal rate of offline ones in all settings outside of Regime 1 and Regime 2.
- Chapter 4 introduces the first online approximately rate-optimal streaming code for the most practically-relevant parameter regime (i.e., Regime 3) where there are no online rate-optimal streaming codes by designing a learning-augmented streaming code.
- Chapter 5 assesses a large corpus of traces from 1:1 video calls from Microsoft Teams and shows that the losses are suitable for streaming codes. It then introduces Tambur, a new communication scheme combining a new streaming code with machine learning. Finally, it demonstrates the benefits of Tambur in an offline evaluation of traces from Teams and in an online evaluation over a simulated network.
- Chapter 6 extends the streaming model to accommodate multiple packets per frame, only some of which may be lost. It introduces new offline streaming codes that are negligibly close to rate-optimal, as well as approximately-rate optimal learning-augmented streaming codes. Finally, it introduces nearly rate-optimal online streaming codes for three parameter regimes and establishes that online schemes cannot match the optimal rate of offline schemes for all other settings.
- Chapter 7 provides the conclusion and future works.

Chapter 2

Streaming codes for variable-size frames

In this chapter, we introduce a new model for streaming codes that captures the requirements of live streaming applications that send sequences of frames of varying sizes, such as videoconferencing. The variability in the sizes of frames was not present in prior works on streaming codes (see Section 2.1). We formalize the new streaming model (Section 2.2) and find that the variability in the sizes of messages leads to several unique challenges for streaming codes. Examples include a new trade-off between the rate and the decoding delay under lossless transmission and the achievable rate being a function of the sizes of frames. We examine the fundamental limits on the rate in the best-case and worst-case (Section 2.3) and find a significant gap exists between these to values for many parameter settings. Motivated by this gap, we then introduce algorithms to compute upper and lower bounds on the optimal rate for any specific sequence of sizes of frames (Section 2.4).

2.1 Background and related work

This section provides an overview of the background of streaming codes relevant to this work. First, we will describe the previously studied streaming model in which frames have the same fixed size. Second, we will detail a sliding-window adversarial channel model which captures the worst-case packet loss patterns which occur in transmissions. The sliding-window adversarial channel model will be used throughout this work. We discuss an upper bound on the rate imposed by such a channel under the previously studied setting where all frames have the same size. Third, we deconstruct a class of optimal code constructions for the previously studied fixed-size streaming model. We highlight aspects of the construction that we leverage later in this work. Fourth, we discuss alternative formulations of the streaming setting, which can form the basis for potential future studies.

2.1.1 Background

The streaming model was first introduced by Martinian and Sundberg in [62]. Under this model, at every time slot, i, the sender receives a frame, $S[i]$, comprised of k symbols from a finite field $\mathbb{F}_q$ for a natural number k. The sender transmits a channel packet, $X[i]$, consisting of

n symbols from $\mathbb{F}_q$ (for a natural number n) over a packet-loss channel to a receiver. Either $X[i]$ is received, or a unique symbol (i.e., $*$) is received, reflecting a packet loss. Packet losses can occur as isolated bursts of some maximal length, b, separated by guardspaces of successful transmissions. Such loss patterns are useful representations of real-world settings where losses occur as occasional bursts, as can be reflected by the Gilbert model [42]. In fact, burst losses occur for various reasons, including persistent Wi-Fi interference and network congestion (when applications overflow router buffers and cause correlated losses [41]).

The rate of the code is naturally defined as $\frac{k}{n}$. The encoding is *causal*, meaning that the channel packet $X[i]$ can be any function of $S[0], \ldots, S[i]$ but may not depend on any future frames. The real-time playback deadline for live communication is incorporated by requiring the receiver to recover each frame, $S[i]$, within a *worst-case-delay of* τ time slots. In other words, the received channel packets of $X[0], \ldots, X[i+\tau]$ are sufficient to decode $S[i]$.[1] Martinian and Sundberg presented an upper bound on the rate of $\frac{\tau}{\tau+b}$ as well as a rate-optimal code construction for a large class of parameter settings. Later, Martinian and Trott in [63] designed a capacity-achieving code construction for all parameter settings for this streaming model.

In certain real-world settings, burst (correlated) and isolated (uncorrelated) packet losses both occur. These loss patterns are well-approximated by statistical models like the GE channel model [31]. Yet, constructing coding schemes directly for such statistical models is believed to be hard. An analytically tractable sliding-window adversarial channel model approximating the worst-case conditions of models such as the GE model was introduced by Badr et al. in [11]. The channel model is characterized using three parameters $a, b,$ and w and is referred to as $C(a, b, w)$.[2] For every w consecutive channel packets, one burst of no more than b consecutive packets or up to a arbitrary packets may be lost.

A generalized streaming model incorporating a $C(a, b, w)$ sliding-window adversarial channel was introduced by Badr et al. in [11]. The authors designed a near-optimal streaming code construction for this streaming model. Badr et al. also showed that $\frac{\tau-a+1}{\tau+b-a+1}$ and $\frac{w-a}{w+b-a}$ are upper bounds on the rate when the worst-case-delay is $\tau < w$ and $\tau \geq w$ respectively. We will later show that the argument used to prove these bounds extends to the setting where frames have variable sizes.

Later, the above upper bound on the rate was attained by streaming code constructions designed in the two independent concurrent works [35] and [49]. An alternative explicit capacity-achieving streaming code for the model was presented by Dudzicz et al. in [30]. These constructions require an exponential field size for certain parameter settings. A capacity-achieving streaming code and an explicit capacity-achieving streaming code with quadratic field size requirements were concurrently designed by Krishnan et al. in [52] and Domanovitz et al. in [27]. The design of these streaming codes employs the technique of *diagonal interleaving* to convert the problem of constructing a rate-optimal streaming code into the more tractable challenge of designing a block code of the same rate. To design rate-optimal streaming codes for a worst-case-delay of τ and a $C(a, b, w)$ channel, one can design a block code which decodes each symbol within τ symbols when either a burst of at most b consecutive symbols or up to a arbitrary symbols are lost. The technique of first creating a block code and then applying interleaving has also

[1]In [62], the worst-case-delay parameter was called T rather than τ

[2]In [11], the parameters (a, b, w) were referred to as (N, B, W).

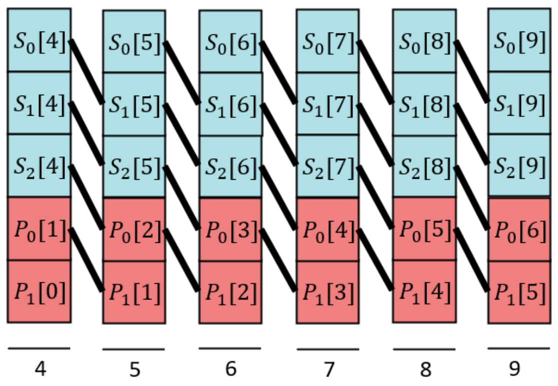


Figure 2.1: Interleaving example of a $(5, 3)$ block code. The blue boxes labeled $S_j[i]$ are symbols of frame $S[i]$, the red boxes labeled $P_j[i]$ are parity symbols, and the black lines connect the boxes which are part of the same block. The numbers under the lines indicate the time slots.

been employed in several other prior works, including [11, 38, 62, 63].

Later in this work, we will leverage existing block codes, such as from [27, 30, 35, 49, 52], as a component of our proposed code construction. Specifically, we shall consider systematic rate-optimal block codes presented in [27], whose field size requirement is quadratic in the delay parameter τ. Any block code designed for the streaming model, including those presented in [30, 35, 49, 52], could likewise be used by our proposed code construction. We refer to any such code as a Streaming Block Code (SBC). We now highlight a few relevant details for such codes, which we will use later in this work. For any parameter setting, (τ, a, b), we denote any systematic (n, k) SBC where $n = (\tau + b - a + 1)$ and $k = (\tau - a + 1)$ as $\langle s_0, \ldots, s_{\tau-a}, p_0, \ldots, p_{b-1}\rangle$. Specifically, $s_0, \ldots, s_{\tau-a}$ are the $(\tau - a + 1)$ systematic symbols and $p_0, \ldots, p_{b-1}$ are the b parity symbols. For these codes, the ith symbol for $i \in \{0, \ldots, n-1\}$ is decoded using the first $\min(i+\tau+1, n)$ symbols in the presence of a single burst loss of b consecutive symbols or the loss of a arbitrary symbols.

We now illustrate how to use interleaving to convert a block code into a streaming code. An (n, k) systematic block code which maps k systematic code symbols, $(s_0, \ldots, s_{k-1})$, into n code symbols, $(s_0, \ldots, s_{k-1}, p_0, \ldots, p_{n-k-1})$, will be used. In the ith time slot, the sender receives as input the frame $S[i] = (S_0[i], \ldots, S_{k-1}[i])$ comprising k symbols, and the channel packet $X[i] = (S_0[i], \ldots, S_{k-1}[i], P_0[i-k], \ldots, P_{n-k-1}[i-n+1])$ is sent. The symbol $P_{n-k-1}[i-n+1]$ is the final symbol of a *distinct block code* ("block") consisting of $(S_0[i-n+1], \ldots, S_{k-1}[i-n+k], P_0[i-n+1], \ldots, P_{n-k-1}[i-n+1])$. This block contains a single symbol from each of channel packets $X[i-n+1], \ldots, X[i]$. The channel packets $X[i-n+1], \ldots, X[i-n+k]$ contain $S_0[i-n+1], \ldots, S_{k-1}[i-n+k]$ respectively, and $X[i-n+k+1], \ldots, X[i]$ contain $P_0[i-n+1], \ldots, P_{n-k-1}[i-n+1]$ respectively. Next, we discuss the block which corresponds to frame $S[i]$. This block comprises (a) symbols of frames $S[i], \ldots, S[i+k-1]$ sent in channel packets $X[i], \ldots, X[i+k-1]$, and (b) parity symbols, $P_0[i], \ldots, P_{n-k-1}[i]$, sent in channel packets $X[i+k], \ldots, X[i+n-1]$. Specifically, the jth position of the block consists of the jth symbol of the corresponding channel packet for $j \in \{0, \ldots, n-1\}$. Hence, the block comprises

$$\langle S_0[j], S_1[j+1], \ldots, S_{k-1}[j+k-1], P_0[j], \ldots, P_{n-k-1}[j]\rangle.$$

We demonstrate an example of converting a block code into a streaming code with diagonal interleaving for a $(5, 3)$ block code in Figure 2.1.

During time slot i, let $i' = (i - k + 1)$. The block code

$$\langle S_0[i'], \dots, S_{k-1}[i], P_0[i'], \dots, P_{n-k-1}[i'] \rangle$$

is computed. The parity symbols

$$(P_0[i'], \dots, P_{n-k-1}[i'])$$

are defined before they are sent in channel packets $X[i + 1], \dots, X[i + n - k]$ respectively. Consequently, during time slot i, the each value $P_j[l]$ for $j \in \{0, \dots, n - k - 1\}$ is accessible for $l \leq (i - k + 1)$, since it was defined during time slot $(l + k - 1) \leq i$. Finally to handle edge conditions, for any $z < 0$ and $j \in \{0, \dots, k-1\}$, $S_j[z]$ is defined to be an arbitrary fixed symbol.

2.1.2 Other related work

Several other variants of the streaming model have been studied in the literature. We briefly discuss them below for the sake of completeness. Most of these models involve frames having fixed sizes. Under a streaming model with multiplexing, a sender receives two streams of frames as input with two different decoding delays for transmission over a burst-only channel [12, 36]. Under another model, a sender transmits a stream of frames to two different receivers over two different burst-only channels subject to two different decoding delays [7, 9]. Another variant of the streaming model includes unequal error protection wherein all symbols from each frame must be recovered in the event of short bursts, but only certain symbols need to be recovered in the presence of longer bursts [44]. Another setting considers average rather than worst-case-delay for decoding [3]. Various other streaming models incorporate multiple channel uses between every frame [11, 56, 57]. Another variation of the streaming model stipulates partial recovery of certain loss patterns wherein only some of the frames are decoded by their deadlines [8]. The setting of streaming over multi-node relay networks has been studied in several recent works [28, 37, 53]. The notion of two distinct decoding delays has also arisen in the context of VoIP in [13], which introduces codes with a shorter delay to decode a few random packet losses than that of recovering a longer burst of packet losses. A different streaming model formulation considers a channel which can induce multiple burst losses within the worst-case-delay [58]. Diverging from the above models, another streaming model considers (1) high and low priority frames, each with a (potentially different) fixed size, which occur in a fixed periodic manner, (2) channel packets of a fixed size, and (3) unequal error protection [85]. A formal study of incorporating frames with arbitrary variable sizes in these models is outside of the scope of this thesis and is a potential avenue for future work.

2.2 A model for streaming codes with frames of variable sizes

The streaming model discussed in Chapter 1 will now be generalized to incorporate frames of *variable sizes*. The variability in the sizes of the frames induces a new trade-off between the optimal rate and the decoding delay under a lossless transmission. A new delay parameter will be introduced to the model to capture this trade-off. A new definition for the rate of a code is included to reflect the varying sizes of the frames and channel packets.

Under the proposed streaming model, during the ith time slot the sender receives a frame, $S[i] = (S_0[i], \dots, S_{k_i-1}[i])$. The frame consists of k_i symbols drawn uniformly at random from a finite field, $\mathbb{F}_q$, where k_i is an arbitrary non-negative integer. A channel packet, $X[i] = (X_0[i], \dots, X_{n_i-1}[i]) \in \mathbb{F}_q^{n_i}$, is transmitted to the receiver, where n_i is an arbitrary non-negative integer. This deviates from the prior models (such as in [11, 27, 30, 35, 49, 52, 62, 63]) where each $|S[i]| = k$ and $|X[i]| = n$ for some *fixed* positive integers k and n. The channel packet $X[i]$ is a function of the current and previous frames (i.e., $X[i] = Enc\left(\langle S[j] \mid j \in \{0, \dots, i)\}\rangle\right))$. Encoding is not a function of the symbols of future frames (or their sizes), as the sender does not have access to this information.

The channel packet $X[i]$ is transmitted over a lossy channel, and the receiver obtains $Y[i] \in \{X[i], *\}$, where $*$ denotes a dropped packet. The lossy channel is denoted $C(b, \tau)$ and introduces bursts of length at most b followed by guardspaces of length at least τ.

Due to the real-time playback deadline, the receiver must decode $S[i]$ within τ time slots. We refer to τ as the "worst-case-delay" parameter and the requirement that $S[i]$ be decoded by time slot $(i + \tau)$ as the "worst-case-delay constraint." More formally, the receiver decodes $S[i]$ as

$$S[i] = Dec\big(\langle S[j] \mid j \in \{i - \tau, \dots, i - 1\}\rangle,\\ \langle Y[j] \mid j \in \{i, \dots, i + \tau\}\rangle, \langle k_j \mid j \in \{i, \dots, i + \tau'\}\rangle\big)$$

where $\tau' \leq \tau$ is the largest value such that $X[i + \tau']$ has been received (i.e., $Y[i + \tau'] = X[i + \tau']$). In other words, the receiver decodes frame $S[i]$ using the (a) previously-decoded τ frames, $(S[i-\tau], \dots, S[i-1])$, (b) already received channel packets among $(X[i], \dots, X[i+\tau])$, and (c) sizes of up to $(\tau+1)$ of the frames $(S[i], \dots, S[i+\tau])$ which may not have been decoded. In order to inform the receiver about (c) irrespective of which channel packets are lost, the sender adds the sizes of the current frame and previous b frames to a small header of each channel packet. The reason that the $C(b, \tau)$ channel is used is that a streaming code that recovers all frames within the worst-case-delay over the $C(b, \tau')$ channel for $\tau' > \tau$ recovers all lost bursts within τ time slots. Thus, such a streaming code will also satisfy the worst-case-delay over the $C(b, \tau)$ channel. The guardspaces are relaxed to only be of length at least τ.

Under the previously studied model, the rate is $\frac{k}{n}$. But $\frac{k}{n}$ is not well-defined in the proposed model. Accordingly, we introduce a suitable definition of rate for the setting of frames of variable size. To do so, we limit our attention to finite-length sequences of frames. For an arbitrary non-negative integer, t, consider an arbitrary sequence of t frames, $S[0], S[1], \dots, S[t]$. We refer to the corresponding sequence $k_0, \dots, k_t$ as the "frame-size sequence." The rate for any code construction is defined as the ratio of the number of symbols of all frames to the total number of transmitted symbols,

$$R_t = \frac{\sum_{i=0}^{t} k_i}{\sum_{i=0}^{t} n_i}. \tag{2.1}$$

For convenience of notation, we use the convention that $t \geq 4\tau$ and the sizes of each of the first and final 2τ frame is 0 (i.e., $k_0 = 0, \dots, k_{2\tau-1} = 0, k_{t-2\tau+1} = 0, \dots, k_t = 0$).[3] This convention

[3]Unless otherwise indicated.

can be met by prepending and appending 2τ frames of size 0 to any sequence of frames to ensure that it meets this convention without altering the rate.

The setting in which frames have variable sizes differs from where frames all have the same fixed size in the following critical respect. When the sizes of the frames and channel packets are fixed, there exists an optimal rate code construction in which each frame, $S[i]$, is sent as a part of the corresponding channel packet, $X[i]$ ([30, 35, 49, 52]). In other words, there are rate-optimal coding schemes where each frame $S[i]$ can be decoded without any delay under lossless transmission. However, this is *no longer true when the sizes of frames can vary*.

When the frames have variable sizes, distributing symbols of frames over multiple channel packets can lead to a higher rate than sending each frame within its corresponding channel packet. We illustrate this observation with a toy example. Consider the length $(\tau+1)$ sequence of frames where the first frame $S[0]$ is of size τ and the next τ frames have size 0. The $C(b,\tau)$ channel for $b = 1$ could drop $X[0]$. Therefore, if $S[0]$ were transmitted as part of channel packets $X[0]$, at least τ parity symbols would need to be sent in channel packets $X[1],\ldots,X[\tau]$ to decode $S[0]$ within the worst-case-delay of τ time slots. The rate for such a scheme is at most $\frac{1}{2}$. Alternatively, the symbols of frame $S[0]$ could be transmitted evenly over $X[0],\ldots,X[\tau-1]$, and a parity of the previous τ channel packets sent in $X[\tau]$ (i.e., $X[\tau] = \sum_{i=0}^{\tau-1} X[i]$). Such a scheme would have a rate of $\frac{\tau}{\tau+1}$ while satisfying the worst-case-delay constraint.

As shown above, under the setting where frames have variable sizes, distributing the symbols of a frame over multiple channel packets can lead to a higher rate. However, doing so will delay decoding the frame when there are no losses. Thus, the variability in the sizes of the frames *induces a new trade-off between the rate of the code and decoding delay when all channel packets are received*. We incorporate this new trade-off into our model via a new parameter which we call the lossless-delay, τ_L. The receiver must be able to decode every frame, $S[i]$, using channel packets $X[0],\ldots,X[i+\tau_L]$ if they are all received. In other words,

$$S[i] = Dec^{(L)}\big(\langle X[j], k_j \mid j \in \{0,\ldots,i+\tau_L\}\rangle\big).$$

The newly introduced parameter τ_L represents the tolerable decoding delay under lossless channel conditions, whereas τ reflects the worst-case delay in the presence of packet loss. The two parameters (τ_L,τ) are relevant to settings where the transmission is lossless most of the time, and the rare worst-case channel conditions are captured via the $C(b,\tau)$ channel. In such scenarios, a live streaming application may occasionally tolerate a decoding delay of τ time slots but benefit from the faster decoding of τ_L time slots most of the time.

Due to the worst-case-delay constraint, τ, for transmission over a $C(b,\tau)$ channel, each $S[i]$ must be decoded with $X[0],\ldots,X[i+\tau-b]$ when $X[i+\tau-b+1],\ldots,X[i+\tau]$ are lost. Therefore, under a lossless transmission setting, each $S[i]$ is recoverable from $X[0],\ldots,X[i+\tau-b]$. Consequently, the parameter τ_L is at most $(\tau-b)$, leading to

$$0 \le \tau_L \le (\tau-b). \tag{2.2}$$

Higher values of τ_L enable the symbols of the frames to be spread over more channel packets, thereby increasing both the rate of the code and decoding delay under lossless transmission. Moreover, if it were the case that $b > \tau$, for any time slot, i, the channel packets $X[i],\ldots,X[i+\tau]$ could all be lost. As a result, it would be impossible to decode $S[i]$ within a delay of τ, resulting in a capacity of 0. Moreover, if $b = 0$, the channel is lossless, and the capacity is trivially 1. Consequently, we restrict our attention to

$$1 \leq b \leq \tau \tag{2.3}$$

We will refer to input parameters (τ, b, τ_L) satisfying Equations 2.2 and 2.3 as *valid* throughout this work.

2.3 General bounds on rate for streaming codes with variable-size frames

This section discusses general upper and lower bounds on the rate of code constructions for the proposed model. The bounds constitute the least upper bound and greatest lower bound for arbitrary frame-size sequences. Later, Lemma 5 shows that the optimal rate depends on the frame-size sequence and can vary over the entire range between the aforementioned lower and upper bounds on the optimal rate.

2.3.1 General upper bound on the rate

The optimal rate for streaming codes that satisfy the worst-case-delay constraint τ over a $C(b, \tau)$ channel in the setting where all frames have a fixed size is $\frac{\tau}{\tau+b}$ [62, 63]. Next, we show that this quantity remains an upper bound on the rate applies under the proposed model with frames of variable sizes by using a simple extension to the proof techniques used by Martinian and Sundberg [62].

Lemma 1. *For any valid inputs (τ, b, τ_L), for any streaming code which satisfies the worst-case-delay constraint over the $C(b, \tau)$ channel, the rate is at most*

$$R^{(U)} = \frac{\tau}{\tau + b}. \tag{2.4}$$

Proof sketch. In [62], Martinian and Sundberg show that any streaming code satisfying the worst-case-delay constraint over a $C(b, \tau)$ channel must recover from *any* erasure channel which periodically introduces a burst of length b followed by a guard space of length τ. Let $C_{P,i}$ be such an erasure channel whose bursts each begin in positions $\equiv i \mod (\tau + b)$, where $i \mod j$ is defined for a non-negative integer i and positive integer j as the remainder of i divided by j. Even when the sizes of channel packets vary, $C_{P,0}, C_{P,1}, \ldots, C_{P,(\tau+b-1)}$ erase on average $\frac{b}{\tau+b}$ fraction of the transmitted symbols. Therefore, there is always some C_{P,i_*} that erases at least $\frac{b}{\tau+b}$ fraction of the transmitted symbols. Consequently, the rate cannot exceed $R^{(U)}$. Thus, the upper bound on the rate provided in [62] for fixed-size frame packets continues to hold for the model with variable-size frames as well. □

The rate $R^{(U)}$ is attained by the constructions presented in [62, 63] when the sizes of the frames are fixed. Thus, $R^{(U)}$ is the smallest general upper bound on the rate of streaming codes for arbitrary frame-size sequences.

2.3.2 General lower bound on rate

Next, we present a general lower bound on the rate. Each frame can always be encoded separately (i.e., transmitted symbols corresponding to each frame are kept independent of all other frames) for any frame-size sequence. The optimal rate of coding schemes encoding frames separately, thus, serves as a lower bound on the optimal rate. Moreover, this bound is tight for certain frame-size sequences and therefore is the greatest lower bound. For example, it is tight when the worst-case-delay is τ, and each frame of positive size is followed by at least τ frames of size 0. For such frame-size sequences, the sender *must* encode each frame separately.

Next, we will present a simple code construction with the best-possible rate among code constructions that *encode each frame separately* and identify its rate. For valid inputs (τ, a, b, w, τ_L), the proposed code construction is called the "(τ, a, b, τ_L)-separate encoding scheme." The scheme is presented in two cases.

Case 1: $\tau_L < (a-1)$. In this case, since $(\tau_L+1) < a$, all symbols used to decode a frame under lossless transmission can be lost under lossy transmission. In addition, either $(a-1-\tau_L)$ arbitrary channel packets may be lost, or the next $(b-\tau_L-1)$ channel packets may be lost. Consequently, the rate is at most 0.5 in this case. We present the scheme first using a toy example and then in detail.

Toy example. An example of the $(7, 3, 5, 1)$-separate encoding scheme is shown in Figure 2.2 for a frame $S[i] = (S_0[i], \ldots, S_5[i])$. The blue boxes contain the symbols of $S[i]$, while the red boxes contain parity symbols for a systematic $[14, 6]$ MDS code. The 6 symbols of $S[i]$ are transmitted evenly over $X[i]$ and $X[i+1]$. The 8 parity symbols are transmitted evenly over $X[i+4], \ldots, X[i+7]$. The lossless-delay constraint is satisfied, since $S[i]$ is transmitted over $X[i]$ and $X[i+1]$. The worst-case-delay constraint is met, since for any burst of length 5, or any 3 arbitrary losses, enough symbols are received by time slot 7 to decode $S[i]$ using properties of the MDS code.

Detailed description. The symbols of $S[i]$ are sent evenly over all channel packets within the lossless-delay (i.e., $X[i], \ldots, X[i+\tau_L]$). The k_i symbols corresponding to $S[i]$ are transmitted evenly over the final $(\tau-b+1)$ packets by time slot $(i+\tau)$ to cover the case of a burst of length b starting in time slot i. Parity symbols are sent over the remaining channel packets to ensure at least k_i symbols are received for a arbitrary losses. For convenience of notation, let $a' = (\tau_L + 2 + \tau - b)$. The following terms are used

$$\langle \eta, \eta' \rangle = \begin{cases} \big\langle (\tau_L+1)(\tau-b+1), & \text{if } a \le a' \\ (\tau_L+1)(\tau-b-\tau_L+a) \big\rangle & \\ \big\langle (\tau_L+1)(\tau-b+1)(\tau-a+1), & \\ \big((\tau_L+1)(\tau-b+1)(\tau-a+1)+ & \text{if } a > a' \\ (\tau_L+1)(\tau-b+1)(b-\tau_L-1)\big)\big\rangle. & \end{cases} \tag{2.5}$$

We assume that $\eta | k_i$.[4] The frame is partitioned evenly into sets of η symbols. For each such set

[4]It suffices to pad $S[i]$ with strictly fewer than η extra symbols, where $\eta \le \tau^2$ or $\eta \le \tau^3$ depending on whether $a \le (\tau_L + 2 + \tau - b)$. Typically, $\eta \ll k_i$.

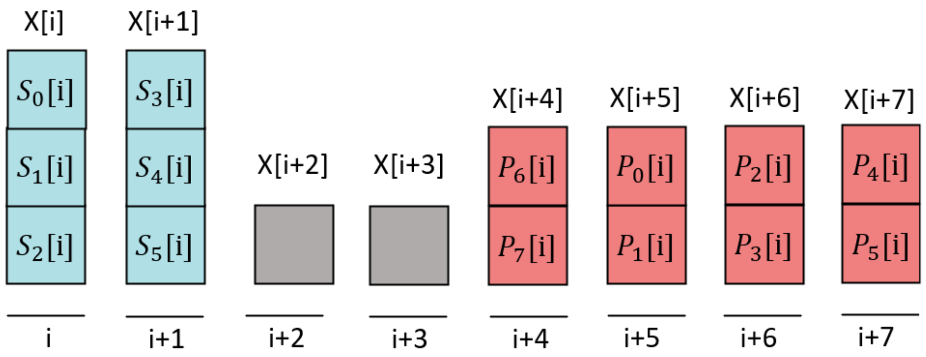


Figure 2.2: The $(7, 3, 5, 1)$-separate encoding scheme is shown for a frame $S[i] = (S_0[i], \ldots, S_5[i])$. The symbols are spread evenly over channel packets $X[i]$ and $X[i+1]$, thereby satisfying the lossless-delay constraint. Additional parity symbols $(P_0[i], \ldots, P_7[i])$ of a $[14, 6]$ systematic MDS code are distributed evenly over channel packets $X[i+4], X[i+5], X[i+6]$, and $X[i+7]$. This ensures that at least 6 out of $(S_0[i], \ldots, S_5[i], P_0[i], \ldots, P_7[i])$ are received in the event of either a burst of length at most 5 losses or the loss of any 3 arbitrary channel packets. Thus, $S[i]$ is decoded within 7 time slots by properties of the MDS code.

- A $[\eta + \eta', \eta]$ systematic MDS code is applied, leading to symbols $c_0, \ldots, c_{\eta+\eta'-1}$, where the final η' symbols are parity symbols.
- The symbols $c_0, \ldots, c_{\eta-1}$ are evenly transmitted over $X[i], \ldots, X[i + \tau_L]$.
- The symbols $c_\eta, \ldots, c_{2\eta-1}$ are evenly transmitted over $X[i + b], \ldots, X[i + \tau]$.
- The symbols $c_{2\eta}, \ldots, c_{\eta'}$ are sent evenly over $X[i + j], \ldots, X[i + b - 1]$, where $j = (\tau_L + b - a + 1)$ if $a \leq (\tau_L + 2 + \tau - b)$ and $j = (\tau_L + 1)$ otherwise.

In short, the scheme involves (a) sending $S[i]$ over (τ_L+1) channel packets to satisfy the lossless-delay constraint, (b) sending parity symbols to recover $S[i]$ when $X[i], \ldots, X[i+b-1]$ are lost, and (c) sending parity symbols to recover $S[i]$ when both $X[i], \ldots, X[i + \tau_L]$ and $(a - \tau_L - 1)$ additional channel packets of $X[i + \tau_L + 1], \ldots, X[i + \tau]$ are lost.

Remark 1. *The rate for the (τ, a, b, τ_L)-separate encoding scheme for case 1 is $\frac{\eta}{\eta'+\eta}$. This follows directly from the MDS code employed.*

The field size requirement is at most that of a $[\eta' + \eta, \eta]$ Reed-Solomon code. If $a \leq (\tau_L + 2 + \tau - b)$, the requirement is at most $(\tau_L + 1)(2\tau - 2b - \tau_L + a + 1)$, which is no more than $2\tau a$. Otherwise, the field size requirement is at most

$$(\tau_L + 1)(\tau - b + 1)\big(2(\tau - a + 1) + (b - \tau_L - 1)\big),$$

which is no more than $3a^2b$.

Case 2 : $\tau_L \geq (a - 1)$. In this case, τ_L is large enough that the symbols of $S[i]$ can be distributed over $(\tau_L + 1) \geq a$ channel packets such that at most k_i symbols are lost by making use of a buffer of $(b - a)$ channel packets in which no symbols are sent, as will be described below. This approach leads to a rate of at least 0.5 in this case. We divide the presentation of case 2 into two sub-cases.

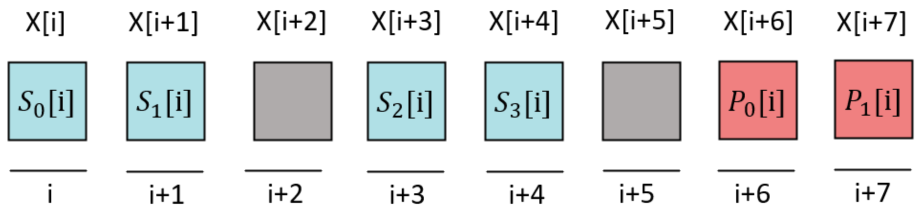


Figure 2.3: The $(7, 2, 3, 4)$-separate encoding scheme is shown for a frame $S[i] = (S_0[i], \dots, S_3[i])$. The symbols are spread evenly over channel packets $X[i], X[i+1], X[i+3]$, and $X[i+4]$. Parity symbols $(P_0[i], P_1[i])$ of a systematic $[6, 4]$ MDS code are spread evenly over channel packets $X[i+6]$ and $X[i+7]$. The lossless-delay constraint is satisfied, since the symbols of $S[i]$ are sent by time slot $X[i+4]$. At most 2 nonempty channel packets are lost with a burst of length 3 or 2 arbitrary losses. Therefore, at least 4 of $(S_0[i], \dots, S_3[i], P_0[i], P_1[i])$ are received, so $S[i]$ is decoded by time slot $(i+7)$.

Sub-case 1 : either $((\tau_L + 1) \mod b) \in \{0\} \cup \{a, \dots, b-1\}$ or $((\tau_L + 1) \mod b)\left(\left\lfloor \frac{\tau_L+1}{b} \right\rfloor + 1\right) \geq a$.

Toy example. An example of the $(7, 2, 3, 4)$-separate encoding scheme is shown in Figure 2.3 for a frame $S[i] = (S_0[i], S_1[i], S_2[i], S_3[i])$. The parity symbols, $P_0[i]$, and $P_1[i]$, are formed using a $[6, 4]$ systematic MDS code. The 6 symbols, $(S_0[i], S_1[i], S_2[i], S_3[i], P_0[i], P_1[i])$ are then periodically sent over time slots i through $(i+7)$ by transmitting one symbol for each of two consecutive time slots followed by not transmitting any symbols for one time slot. Specifically, $S_0[i], S_1[i], S_2[i], S_3[i], P_0[i]$, and $P_1[i]$ are sent over $X[i], X[i+1], X[i+3], X[i+4], X[i+6]$, and $X[i+7]$ respectively. The lossless-delay constraint is satisfied, as $S[i]$ is transmitted over $X[i], \dots, X[i+4]$ where $\tau_L = 4$. At least 4 symbols are received by time slot $(i+7)$. Hence, $S[i]$ can be decoded by properties of the MDS code.

Detailed description. The symbols of $S[i]$ are periodically spread over a channel packets followed by no symbols being sent in a buffer of $(b-a)$ channel packet until time slot $(i+\tau_L)$. Afterward, a buffer of $(b-a)$ channel packets are sent which do not include any symbols corresponding to $S[i]$. Parity symbols are sent in the next a channel packets. A similar interleaving approach with empty positions (i.e., buffers) was used in [50] and [75], albeit for the streaming model with frames all having the same fixed size, where each frame is sent in its entirety as part of the corresponding channel packet, and the parity symbols apply to multiple frames.[5] For convenience of notation, the following term is used

$$\zeta = \left(\left\lfloor \frac{\tau_L + 1}{b} \right\rfloor a + \min\left((\tau_L + 1) \mod b, a\right) \right). \tag{2.6}$$

We will assume that $\zeta | k_i$.[6] The frame is partitioned into sets of ζ symbols. For each such set:

- A $[\zeta + a, \zeta]$ systematic MDS code is applied, leading to symbols $c_0, \dots, c_{\zeta+a-1}$, where the final a symbols are parity symbols.

[5]We developed the technique first independently, although we waited to publish it until after the other work was published.

[6]It suffices to pad $S[i]$ with up to $(\zeta - 1 \leq \tau)$ extra symbols—a quantity typically negligible compared to k_i.

- For $J = \{j_0, \ldots, j_{\zeta-1}\} = \{j \mid j \in \{i, \ldots, i+\tau_L\}, j \mod b < (b-a)\}$ and $l \in \{0, \ldots, \zeta-1\}$, c_l is transmitted in $X[j_l]$.
- $c_\zeta, \ldots, c_{\zeta+a-1}$ are transmitted in $X[i+\tau_L+b-a+1], \ldots, X[i+\tau_L+b]$ respectively.

Remark 2. *The rate for the (τ, a, b, τ_L)-separate encoding scheme for case 2 sub-case 1 is $\frac{\zeta}{\zeta+a}$. This follows directly from the MDS code employed.*

The field size requirement is that of a $[\zeta+a, \zeta]$ Reed-Solomon code. The requirement is at most $(\zeta+a)$, which is no more than $(\tau+1+a)$.

Sub-case 2 : $((\tau_L+1) \mod b) \in \{1, \ldots, a-1\}$ and $((\tau_L+1) \mod b)\left(\left\lfloor \frac{\tau_L+1}{b} \right\rfloor + 1\right) < a$. The construction from sub-case 1 applies to this sub-case, but its rate does not attain the greatest lower bound on the rate. The converse proof in sub-case 1 relies on the worst-case losses corresponding to either (a0 a burst of consecutive losses over between $((\tau_L+1) \mod b)$ and b consecutive channel packets, or (b) a arbitrary losses corresponding to a set of $((\tau_L+1) \mod b)\left(\left\lfloor \frac{\tau_L+1}{b} \right\rfloor + 1\right)$ channel packets. However, in sub-case 2, both loss scenarios comprise fewer than a arbitrary losses. Hence, the worst-case a arbitrary losses cannot be limited to just one of these two quantities. In order to design a scheme that leads to the greatest lower bound for this sub-case, we introduce a construction based on a simple integer program (IP) that reflects minimizing the number of symbols sent by a coding scheme while satisfying the lossless-delay and worst-case-delay constraints. The variables of the IP are $n_0^{(c)}, \ldots, n_\tau^{(c)}$, which denote the sizes of the $(\tau+1)$ channel packets corresponding to the frame.

IP-based construction 1 Takes as input any valid parameters and k_i and uses integer programming to compute the number of symbols to be sent in the next $(\tau+1)$ channel packets.

Input: Valid values for (τ, a, b, w, τ_L) and k_i.
Minimize $\sum_{j=0}^{\tau} n_j^{(c)}$ subject to:

1. $\forall j \in \{0, \ldots, \tau\}, n_j^{(c)} \geq 0$
2. $\left(\sum_{j=0}^{\tau_L} n_j^{(c)}\right) - k_i \geq 0$.
3. $\forall l \in \{0, \ldots, \tau\} \left(\sum_{j=0}^{l-1} n_j^{(c)}\right) + \left(\sum_{j=l+b}^{\tau} n_j^{(c)}\right) - k_i \geq 0$.
4. $\forall I \subseteq \{0, \ldots, \tau\}$ such that $|I| = a$, $\left(\sum_{j \in \{0,\ldots,\tau\} \setminus I} n_j^{(c)}\right) - k_i \geq 0$

Output: $n_i^{(*)} = \left(\sum_{j=0}^{\tau} n_i^{(c)}\right)$.

The constraints of the integer program reflect the requirements that (1) the size of each channel packet is non-negative, (2) the lossless-delay constraint is met, (3) the worst-case-delay constraint is met for bursts of at most b consecutive channel packets, and (4) the worst-case-delay constraint is met for a arbitrary losses. The objective function reflects minimizing the total number of symbols which are sent. Observe that $n_i^{(*)}$ is the total number of symbols sent according to the IP subroutine of IP-based construction 1. For a frame-size sequence $k_0, \ldots, k_t$, let $n_0^{(*)}, \ldots, n_t^{(*)}$ be the outputs of IP-based construction 1 applied to each of $k_0, \ldots, k_t$. For each i where $k_i > 0$, a systematic $[n_i^{(*)}, k_i]$ MDS code is applied to encode $S[i]$ into $c_0, \ldots, c_{n^{(*)}-1}$. The symbols are distributed over channel packets $X[i], \ldots, X[i+\tau]$ so that the number sent for each channel packet $X[j]$ is $n_j^{(c)}$. The construction is systematic, as the first k_i symbols (i.e., $S[i]$) are sent over $X[i], \dots, X[i+\tau_L]$. The following terms will be used to express the rate of IP-based construction 1

$$\langle k^{(*)}, n^{(*)} \rangle = \left\langle \sum_{i=0}^{t} k_i, \sum_{i=0}^{t} n_i^{(*)} \right\rangle . \tag{2.7}$$

Remark 3. *The rate for the (τ, a, b, τ_L)-separate encoding scheme for frame-size sequence $k_0, \dots, k_t$ for case 2 sub-case 2 is $\frac{k^{(*)}}{n^{(*)}}$. This follows directly from the MDS code employed.*

Finally, we note that IP-based construction 1 can be used for any parameter settings. We provide explicit constructions for case 1 and case 2 sub-case 1 because IP-based construction 1 is not explicit.

Next, we use the (τ, a, b, τ_L)-separate encoding scheme described above to provide a general lower bound on the rate. Before doing so, we must verify that the (τ, a, b, τ_L)-separate encoding scheme satisfies the lossless-delay constraint and worst-case-delay constraint over the $C(a, b, w)$ channel for any valid parameters (τ, a, b, τ_L) and sequence of frames. This is done below.

Lemma 2. *For any valid inputs (τ, a, b, w, τ_L) and any sequence of frames $S[0], \dots, S[t]$, the (τ, a, b, τ_L)-separate encoding scheme satisfies the lossless-delay constraint τ_L and the worst-case-delay constraint τ over the $C(a, b, w)$ channel.*

Proof sketch. The proof of Lemma 2 is included in Appendix 2.5.2. □

For any valid inputs (τ, b, τ_L), as τ_L increases, the quantity $R^{(L)}$, is monotonically non-decreasing and approaches the upper bound on the rate of $R^{(U)}$.

Corollary 1. *Whenever $\tau_L = (\tau - b)$ and $b|\tau$, the rate of the (τ, b, τ_L)-separate encoding scheme matches the least upper bound on the rate of $R^{(U)} = \frac{\tau}{\tau+b} = R^{(L)}$.*

Whenever $\tau_L = (\tau - b)$ and $b|\tau$, the (τ, b, τ_L)-separate encoding scheme is also analogous to the scheme presented in [50] for the streaming model with frames of the same fixed size.

The rate of the (τ, a, b, τ_L)-separate encoding scheme constitutes a lower bound on the optimal rate. This quantity is summarized in Lemma 3.

Lemma 3. *For any valid inputs (τ, a, b, w, τ_L), the optimal rate for streaming codes that satisfy the lossless-delay constraint and worst-case-delay constraint over the $C(a, b, w)$ channel for an arbitrary frame-size sequence is at least $R^{(L)} =$*

$$\begin{cases} \frac{\eta}{\eta+\eta'} & \textit{if } \tau_L < a-1 \\ \frac{\zeta}{\zeta+a} & \textit{if } \tau_L \geq (a-1) \textit{ and } ((\tau_L+1) \mod b) \in \\ & \{0\} \cup \{a, \dots, b-1\} \textit{ or } \tau_L \geq (a-1) \textit{ and} \\ & ((\tau_L+1) \mod b)\left(\left\lfloor \frac{\tau_L+1}{b} \right\rfloor + 1\right) \geq a \\ \frac{k^{(*)}}{n^{(*)}} & \textit{if } \tau_L \geq (a-1) \textit{ and} \\ & 0 < ((\tau_L+1) \mod b) < a \textit{ and} \\ & ((\tau_L+1) \mod b)\left(\left\lfloor \frac{\tau_L+1}{b} \right\rfloor + 1\right) < a. \end{cases} \tag{2.8}$$

Proof. The (τ, a, b, τ_L)-separate encoding scheme exhibits this rate. This follows directly from the parameters of the MDS code used in each case, as is noted in Remark 1, Remark 2, and

Remark 3. The lossless-delay constraint and worst-case-delay constraint over the channel model are also satisfied by the code construction, as was shown in Lemma 2. □

For any valid inputs (τ, a, b, w, τ_L), as τ_L increases, the quantity $R^{(L)}$, is monotonically non-decreasing and approaches the upper bound on the rate of $R^{(U)}$. Whenever $\tau_L = (\tau - b)$ and either $b = a$ or $((\tau + 1) \mod b) = a$, the least upper bound on the rate of $R^{(U)}$ is equal to the greatest lower bound on the optimal rate of $R^{(L)}$. For such parameter settings, the rate of the (τ, a, b, τ_L)-separate encoding scheme matches the least upper bound on the rate of $R^{(U)}$. In such settings, the (τ, a, b, τ_L)-separate encoding scheme is also analogous to the scheme presented in [50] for the streaming model with frames of the same fixed size.

We show in Lemma 4 that $R^{(L)}$ is the greatest lower bound on the optimal rate for arbitrary frame-size sequences.

Lemma 4. *For any valid inputs (τ, a, b, w, τ_L), $R^{(L)}$ is the greatest lower bound on the optimal rate for arbitrary frame-size sequences for streaming codes that satisfy the lossless-delay constraint and worst-case-delay constraint over the $C(a, b, w)$ channel.*

Proof sketch. The proof of Lemma 4 is included in Appendix 2.5.1. □

2.4 Bounds on rate for specific frame-size sequences

In the proposed model for streaming codes with frame packets of varying sizes, the optimal rate for any transmission depends on the specific frame-size sequence. The optimal rate can be as large as $R^{(U)}$ and as small as $R^{(L)}$, as was shown in Section 2.3. These general bounds are agnostic to the sizes of the frames and apply to an *arbitrary* frame-size sequences. Next, we develop a deeper understanding for the optimal rate of a streaming code for any *specific* frame-size sequence. Recall that we refer to the setting in which the sender and receiver have access to the complete frame-size sequence as the offline setting. We consider the offline setting for the rest of this Section 2.4. This differs from the setting considered in the rest of this work (i.e., the online setting) where the sender and receiver during time slot i do not have access to $k_{i+1}, \ldots, k_t$. The optimal rate for the online setting for any specific frame-size sequence is not well-defined because there exists a coding scheme which attains the best possible rate, which is that of the offline setting. However, the rate of that coding scheme may not be optimal for other frame-size sequences, as is discussed in detail later in Chapter 3.

First, we show that the optimal rates for various frame-size sequences can take values over the entire range of $[R^{(L)}, R^{(U)}]$. Naturally, the general upper and lower bounds on the rate, i.e., $R^{(U)}$ and $R^{(L)}$, are inherently loose for many frame-size sequences, motivating the need for tighter bounds. We then present an algorithm to compute an upper bound on the rate for linear encoding schemes by imposing the lossless-delay constraint and worst-case-delay constraint over the channel model for each frame. We then present an algorithm to compute the best possible rate for a coding scheme that combines block codes such as those presented in [27, 30, 35, 49, 52] with the separate encoding scheme presented in Section 2.3.2. The so-computed rate serves as a lower bound on the optimal rate. Finally, we empirically evaluate these upper and lower bounds on the optimal rate. The empirical evaluation demonstrates that the gap between the lower and upper bounds computed by the two aforementioned algorithms is a significant improvement over the gap between the bounds agnostic to the size sequence.

Lemma 5. *For any valid inputs (τ, b, τ_L), the set of optimal rates for coding schemes that satisfy the lossless-delay constraint and worst-case-delay constraint over any $C(b,\tau)$ channel over all possible frame-size sequences are dense in $\left[R^{(L)}, R^{(U)}\right]$.*

Proof. Let $v \in \left[R^{(L)}, R^{(U)}\right]$, and $\epsilon > 0$ arbitrarily. We will show that there is a frame-size sequence for which the optimal rate is within ϵ of v. Let $p, r \in \mathbb{Z}^+ \cup \{0\}$ be chosen so that the quantity $R^{(p,r)} = \frac{p+r}{\frac{p}{R^{(L)}}+\frac{r}{R^{(U)}}}$ satisfies $|R^{(p,r)} - v| < \epsilon$. When $v \in \left(R^{(L)}, R^{(U)}\right)$, the existence of such p and r follows from the fact that $R^{(p,r)} \to v$ in the limit as $\frac{p}{r} \to \frac{R^{(L)}v - R^{(L)}R^{(U)}}{R^{(L)}R^{(U)} - R^{(U)}v}$. When $v = R^{(L)}$ or $v = R^{(U)}$ it suffices to choose $(r = 0, p > 0)$ and $(p = 0, r > 0)$ respectively. Let d be the smallest positive integer for which $\frac{d}{R^{(L)}}$ and $\frac{d}{R^{(U)}}$ are both integers. Consider the following length $(3\tau + 1)$ frame-size sequence: $k_0 = pd$, $k_j = \frac{rd}{\tau}$ for $j \in \{\tau+1, \ldots, 2\tau\}$, and $k_j = 0$ for $j \in \{1, \ldots, \tau\} \cup \{2\tau+1, \ldots, 3\tau\}$.

The proof follows from verifying that the optimal rate for this frame-size sequence is at most $R^{(p,r)}$ and presenting a coding scheme with rate $R^{(p,r)}$, which we will show below.

Upper bound. The lossless-delay constraint and worst-case-delay constraint over the $C(b,\tau)$ channel must be satisfied for frame $S[0]$. This necessitates that at least $\frac{pd}{R^{(L)}}$ symbols are sent by time slot τ (Lemma 4). The lossless-delay constraint and worst-case-delay constraint over the $C(b,\tau)$ channel must be met for the rd symbols corresponding to the remaining τ frames. Thus, at least $\frac{rd}{R^{(U)}}$ additional symbols must be sent due to the upper bound on the rate of $R^{(U)}$. A total of at least $\frac{pd}{R^{(L)}} + \frac{rd}{R^{(U)}}$ symbols are sent, leading to an upper bound on the rate of $R^{(p,r)}$.

Achievability. Applying the (τ, b, τ_L)-separate encoding scheme to frame $S[0]$ involves transmitting $\frac{pd}{R^{(L)}}$ symbol. The systematic $[\tau + b, \tau]$ block code, presented in [27] (or alternatively the block codes from [30, 35, 49, 52]), can be applied to frames $S[\tau+1], \ldots, S[2\tau]$ by sending each frame in the corresponding channel packet. Afterward, the channel packets $X[2\tau + 1], \ldots, X[2\tau + b]$ are defined to each contain $\frac{rd}{\tau}$ parity symbols of the block code. The lossless-delay constraint and worst-case-delay constraint over the $C(b,\tau)$ channel are met by the definition of the (τ, b, τ_L)-separate encoding scheme and block codes. This code construction has a rate of $R^{(p,r)}$. □

Hence, the quantities $R^{(L)}$ and $R^{(U)}$ are insufficient for understanding the best possible rate for a specific frame-size sequence. As such, Lemma 5 motivates the need to compute upper and lower bounds on the optimal rate for any specific frame-size sequence that can more tightly bound the optimal rate. A desirable property for doing so is that the upper and lower bounds on rate can likewise range from $R^{(L)}$ to $R^{(U)}$. We introduce algorithms to compute upper and lower bounds on the optimal rate for any specific frame-size sequence in Sections 2.4.1 and 2.4.2 to capture this property.

2.4.1 An upper bound on the optimal rate for specific frame-size sequences.

We now present Algorithm 1, which computes an upper bound on the rate for *linear* encoding schemes for any given frame-size sequence by imposing the lossless-delay constraint and worst-case-delay constraint over the channel model for each frame. To do so, Algorithm 1 will make use of an integer program by converting the lossless-delay and worst-case-delay constraints into constraints for the IP. In order to avoid confusion over the term "constraint," we refer to constraints of the IP as "constraints" and the lossless-delay and worst-case-delay constraints as "requirements" in this section and Section 2.4.2. Under Algorithm 1, (1) the lossless-delay and worst-case-delay requirements are converted into constraints for an integer program (IP) with a simple minimization objective function, (2) its solution is computed, and (3) its solution is converted into an upper bound on the optimal rate. In Section 2.4.3, we will show empirically that the upper bound on the optimal rate determined by Algorithm 1 can be significantly lower than $R^{(U)}$.

Consider any frame-size sequence of an arbitrary length t. Consider any valid inputs (τ, b, τ_L). We first model the sizes of the frame and channel packets, and the associated parameters will serve as the variables for the IP. Each channel packet, $X[i]$, for $i \in \{0, \dots, t\}$ comprises $(X^{(0)}[i], X^{(1)}[i])$. Under a lossless transmission, $X^{(0)}[0], \dots, X^{(0)}[i + \tau_L]$ are sufficient to recover frames s$S[0], \dots, S[i]$. In contrast, $X^{(1)}[0], \dots, X^{(1)}[i + \tau_L]$ are used for decoding only under a lossy transmission. The linear equations corresponding to the symbols of $X^{(1)}[i]$ are in the span of the linear equations corresponding to the symbols of $\langle X^{(0)}[j] \mid j \leq i \rangle$. Each quantity $|X^{(1)}[i]|$ will be a variable of the IP, whereas there will be $(\tau_L + 1)$ variables corresponding to $X^{(0)}[i]$ defined shortly. The details of how $(X^{(0)}[i], X^{(1)}[i])$ are defined are only used in the proof of Theorem 1 and can be found in the Appendix.

The symbols of $X^{(0)}[i]$ are partitioned into $X_l^{(0)}[i]$ for $l \in \{i, \dots, i - \tau_L\}$ for convenience of notation, where each quantity $|X_l^{(0)}[i]|$ will be a variable of the IP. Under a lossless transmission, the symbols sent in channel packet $X[j]$ for $j \in \{0, \dots, \tau_L\}$ used to decode $S[0]$ are called $X_0^{(0)}[j]$. Similarly, for $i = 1, \dots, t$, the symbols sent in channel packet $X[j]$ for $j \in \{i, \dots, i + \tau_L\}$ that are used to decode $S[i]$ under lossless transmission are labeled as $X_i^{(0)}[j]$. Thus, $X^{(0)}[i] = \langle X_j^{(0)}[i] j \in \{i - \tau_L, \dots, i\} \rangle$, and hence, $\sum_{j=i-\tau_L}^{i} |X_j^{(0)}[i]| = |X^{(0)}[i]|$ for any $i \in \{0, \dots, t\}$.[7]

We next outline how the constraints for the IP reflect the worst-case-delay and lossless-delay requirements of the streaming model. For ease of presentation, in this paragraph, we assume that the coding scheme is systematic. Under a systematic coding scheme, the quantity $X_j^{(0)}[i]$ for $i \in \{0, \dots, t\}, j \in \{i - \tau_L, \dots, i\}$ corresponds to $|X_j^{(0)}[i]|$ distinct symbols of $S[j]$. Each of $|X_j^{(0)}[i]|$ and $|X^{(1)}[i]|$ are non-negative integers to reflect that each channel packet consists of some non-negative quantity of symbols corresponding to frame $S[j]$ for $j \in \{i - \tau_L, \dots, i\}$, along with some non-negative number of parity symbols (constraints #1 and #2 in Algorithm 1). The lossless-delay requirement is imposed through requiring that k_i symbols for frame $S[i]$, for each $i \in \{0, \dots, t - \tau\}$, be transmitted over $X[i], \dots, X[i + \tau_L]$ (constraint #3).[8] In the proof of Lemma 1, it was shown that satisfying the worst-case-delay requirement over any $C(b, \tau)$ chan-

[7]For convenience of notation, the edge conditions are handled by modeling $S[-\tau_L], \dots, S[-1], S[t + 1], \dots, S[t + \tau]$ as frames of size 0. Furthermore, variables $|X_j^{(0)}[i]| = 0$ whenever at least one of i, j is either negative or i exceeds t. Similarly, $|X^{(1)}[i]| = 0$ whenever i is negative or exceeds t.

[8]The final τ frames are of size 0 and, therefore, no lossless-delay requirement needs to be imposed. For $i < 0$ or $j < 0$, as well as $i > t$ and $j \in \{0, \dots, \tau_L\}$, $|X_j^{(0)}[i]| = 0$ is defined only for edge conditions.

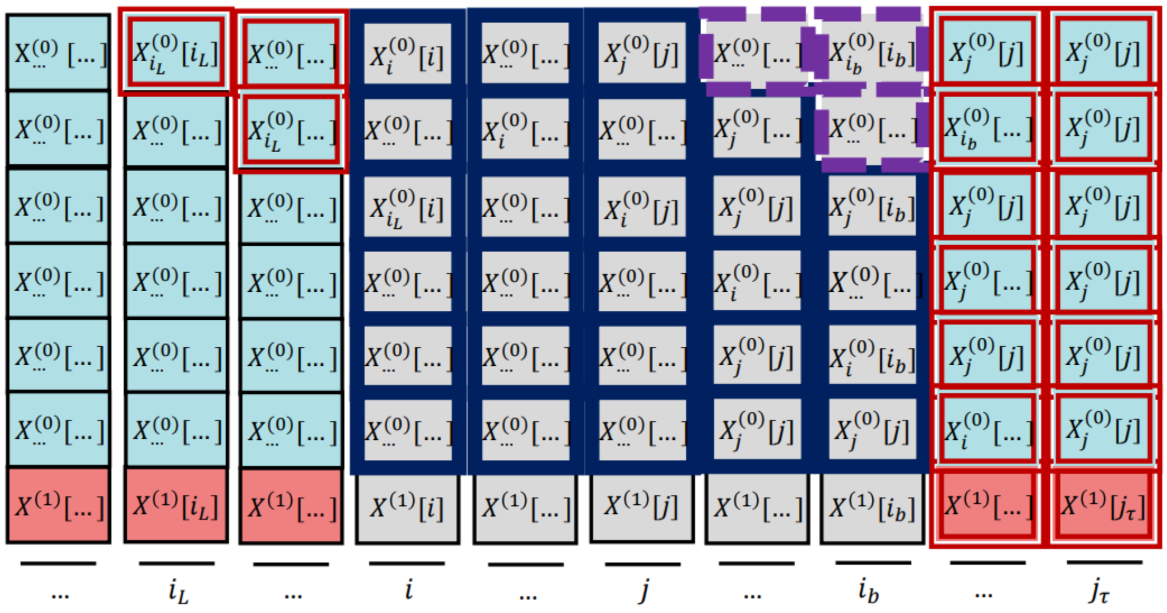


Figure 2.4: An example of imposing constraint #4 (in Algorithm 1) for $j \in \{i, \ldots, i+b-2\}$. The quantities i_L, i_b, and j_τ represent $(i - \tau_L), (i+b-1)$, and $(j+\tau)$ respectively. The gray boxes (time slots $i, \ldots, i_b$) are lost in a burst of channel packets $X[i], \ldots, X[i_b]$. The symbols in the gray boxes with thick blue outlines must be recoverable using the symbols inside boxes with double red outlines. This requirement allows for the relaxation that the symbols inside boxes with purple dashed outlines are treated as received.

nel necessitates satisfying the worst-case-delay requirement over all channels which periodically drop b channel packets and allow τ successful transmissions. This implies that for each burst of length b starting in time slot $i \in \{0, \ldots, t\}$, $S[j]$, for $j \in \{i - \tau_L, \ldots, i+b-1\}$, must be decoded by time slot $(i+\tau+b-1)$, while the worst-case-delay requirement necessitates that $S[j]$ be decoded by $(j+\tau)$ (constraint # 4). A toy example of constraint #4 is shown in Figure 2.4. Under constraint #4, for any considered burst of length b beginning in time slot $i \in \{0, \ldots, t-b\}$ and terminating in time slot $i' \in \{i, \ldots, t\}$, the following relaxation of the worst-case-delay requirement is imposed. For each frame $S[j] \in \{S[i-\tau_L], \ldots, S[i']\}$, $S[i-\tau_L], \ldots, S[j]$ must be decoded by time slot $(j+\tau)$. This relaxation is more restrictive than the relaxation which allows all lost frames to be decoded within τ time slots of the final lost frame. Finally, we consider the relaxation that each $X^{(0)}_{j'}[i']$ is received even if $X[i']$ is lost for $j' > j$.

The objective function is to minimize the sum of all variables. The summation constitutes a lower bound on the number of transmitted symbols. The solution is easily converted into an upper bound on the rate since the total number of symbols of the frames is fixed.

We now present Algorithm 1.

In Theorem 1, we verify that the output of Algorithm 1 is an upper bound on the rate.

Theorem 1. *For any valid inputs (τ, b, τ_L) and any frame-size sequence $k_0 \ldots k_t$, the value computed by Algorithm 1 is an upper bound on the rate of streaming codes that satisfy the lossless-delay requirement and worst-case-delay requirement over the $C(b, \tau)$ channel while employing*

Algorithm 1 Takes as input any valid parameters and frame-size sequence and uses integer programming to compute an upper bound on the rate of streaming codes with linear encoding schemes for the input frame-size sequence.

Input: Valid values for (τ, b, τ_L) and frame-size sequence $k_0, \ldots, k_t$.
Minimize $\sum_{i=0}^{t+\tau}\left(|X^{(1)}[i]| + \sum_{j=i-\tau_L}^{i} |X_j^{(0)}[i]|\right)$ subject to:

1. $\forall i \in \{0, \ldots, t-\tau\}, j \in \{i-\tau_L, \ldots, i\}, |X_j^{(0)}[i]| \geq 0$ and $|X_{j'}^{(0)}[i']| = 0$ when $i' < 0, i' > t$, or $j' < 0$.
2. $\forall i \in \{0, \ldots, t+\tau\}, |X^{(1)}[i]| \geq 0$.
3. $\forall i \in \{0, \ldots, t-\tau\}, \sum_{j=0}^{\tau_L} |X_i^{(0)}[i+j]| \geq k_i$ and $\sum_{j=0}^{\tau_L} |X_j^{(0)}[i+j]| \leq k_i$.
4. $\forall i \in \{0, \ldots, t-b+1\}, \forall j \in \{i-\tau_L, \ldots, i+b-1\}$

$$\sum_{z=i+b}^{\min(j+\tau, i+b+\tau-1)} |X^{(1)}[z]| - \sum_{l=i-\tau_L}^{j} k_l +$$
$$\sum_{l=i-\tau_L}^{j} \sum_{z \in \{l, \ldots, l+\tau_L\} \setminus \{i, \ldots, i+b-1\}} |X_l^{(0)}[z]| \geq 0.$$

Output: $\frac{\sum_{i=0}^{t} k_i}{\sum_{i=0}^{t+\tau}\left(|X^{(1)}[i]| + \sum_{j=i-\tau_L}^{i} |X_j^{(0)}[i]|\right)}$.

linear encoding.

Proof sketch. Follows from the high-level description presented above. The full details are shown in Appendix 2.5.3. □

Remark 4. *The value computed by Algorithm 1 is also an upper bound on the rate for streaming code constructions in an online setting since the offline setting involves providing the sender and receiver additional information not available in the online setting.*

Next, we show that the outputs of Algorithm 1 can tightly bound the optimal rate for various frame-size sequences with values ranging from $R^{(L)}$ to $R^{(U)}$. Recall from Lemma 5 that this is a desired property because the optimal rate can likewise range from $R^{(L)}$ to $R^{(U)}$. For any frame-size sequence, $(k_0, \ldots, k_t)$, let $Alg_{\tau,b,\tau_L}^{(1)}(k_0, \ldots, k_t)$ and $Opt_{\tau,b,\tau_L}(k_0, \ldots, k_t)$ denote the output of Algorithm 1 and the optimal rate respectively.

Lemma 6. *For any valid parameters (τ, b, τ_L), for all $\epsilon > 0$ and $v \in [R^{(L)}, R^{(U)}]$, there exists a sequence of frame packet sizes $(k_0, \ldots, k_t)$ such that $Alg_{\tau,b,\tau_L}^{(1)}(k_0, \ldots, k_t) = Opt_{\tau,b,\tau_L}(k_0, \ldots, k_t)$ and $|Alg_{\tau,b,\tau_L}^{(1)}(k_0, \ldots, k_t) - v| < \epsilon$.*

Proof. We now introduce a frame-size sequence for which the optimal is within ϵ of v. Let $p, r \in \mathbb{Z}^+ \cup \{0\}$ be chosen so that the quantity $R^{(p,r)} = \frac{p+r}{\frac{p}{R^{(L)}} + \frac{r}{R^{(U)}}}$ obeys the inequality $|R^{(p,r)} - v| < \epsilon$.

Let d be the smallest positive integer such that $\frac{d}{R^{(L)}}$ and $\frac{d}{R^{(U)}}$ are integers. Consider the frame-size sequence $k_0 = pd, k_j = \frac{rd}{\tau}$ for $j \in \{\tau+1, \dots, 2\tau\}$, and $k_j = 0$ for $j \in \{1, \dots, \tau\} \cup \{2\tau + 1, \dots, 3\tau\}$.

The code construction presented in the proof of Lemma 5 satisfies lossless-delay requirement and worst-case-delay requirement over the $C(b, \tau)$ channel and has rate $R^{(p,r)}$. Hence, the optimal rate is at least $R^{(p,r)}$.

The lossless-delay requirement and worst-case-delay requirement over the $C(b, \tau)$ channel are both imposed under Algorithm 1. As shown in the proof of Lemma 4, these requirements are sufficient to show that at least $\frac{pd}{R^{(L)}}$ symbols must be sent by time slot τ due to frame $S[0]$. Recall from Section 2.3 that the rate is upper bounded by $R^{(U)} = \frac{\tau}{\tau+b}$. This holds because all frames are decoded for any lossy channels $C_{P,i}$ for $i \in \{0, \dots, \tau + b - 1\}$ consisting of bursts of length b starting in positions $\equiv i \mod (\tau + b)$. Furthermore, at least one such channel drops at least $\frac{b}{\tau+b}$ fraction of the transmitted symbols. Similarly, one can show that at least one such channel drops at least $\frac{b}{\tau+b}$ fraction of the symbols sent strictly after time slot τ. All such periodic packet loss channels $C_{P,i}$ are accounted for under Algorithm 1, due to constraint #4. Hence, the output of Algorithm 1 reflects that at least $\frac{rd}{R^{(U)}}$ additional symbols are sent strictly after time slot τ. The output of Algorithm 1 is, thus, at most $R^{(p,r)}$.

The value computed by Algorithm 1 is an upper bound on the optimal rate, which is at least $R^{(p,r)}$. Therefore, Algorithm 1 must output $R^{(p,r)}$; this is a tight upper bound on the rate for the frame-size sequence, and it is within ϵ of v.

□

The value computed by Algorithm 1 is an upper bound on the rate, but the algorithm can be computationally intensive. We now discuss modifications to the algorithm that trade off tightness for runtime.

There is a simple linear program (LP) relaxation of Algorithm 1 which uses non-negative real-valued variables $|X_i^{(0)}[j]|$ and $|X^{(1)}[i]|$ rather than integral ones. A solution to this LP can be converted into an upper bound on the rate by setting each variable to be the floor of its previous value. The conversion changes the size of each channel packet by at most $(\tau_L + 2)$ which, in practice, is three orders of magnitude less than the average size of the frames. Finally, Lemma 6 would likewise apply to the LP relaxation of Algorithm 1.

Remark 5. *Modifying Algorithm 1 to solve an LP relaxation of the underlying IP has a negligible impact on its output.*

It is simple to analyze the runtime of the modified version of Algorithm 1 that uses an LP relaxation of the IP. For any valid inputs (τ, b, τ_L) and frame-size sequence $k_0 \dots k_t$, the total number of combined constraints in Algorithm 1 is at most $(b + 2\tau_L + 2)t$. Consequently, Algorithm 1 with the LP relaxation of the IP runs in $poly(\tau t)$ time.

Remark 6. *Modifying Algorithm 1 to use an LP relaxation of the underlying IP results in a polynomial-time algorithm. Its output is less than or equal to $R^{(U)}$.*

2.4.2 A lower bound on the optimal rate for specific frame-size sequences

We now present Algorithm 2, which computes a lower bound on the optimal rate for *offline* streaming codes that satisfy the lossless-delay constraint and worst-case-delay constraint over the $C(b, \tau)$ channel. Specifically, under Algorithm 2, an integer program with a simple minimization function is used to determine the minimum number of symbols which need to be transmitted using a combination of two schemes. The solution to this integer program is then converted into a lower bound on the optimal rate. The values computed by Algorithm 2 over various frame-size sequences can vary over $[R^{(L)}, R^{(U)}]$. We will later see in Section 2.4.3 that the empirically computed lower bound on the optimal rate determined by Algorithm 2 can be significantly tighter than that of $R^{(L)}$. Specifically, the gap between the output of Algorithm 2 and Algorithm 1 is shown to be small in Section 2.4.3, highlighting the utility of Algorithm 1 empirically. A high-rate offline construction (e.g., Algorithm 2) is of interest because it lays the groundwork for designing high-rate online constructions. For example, in Chapter 4, we will convert an offline rate-optimal construction into an online approximately rate-optimal online construction. A detailed discussion on the difference between the best possible rate for online and offline streaming codes is later presented in Chapter 3. We first provide an overview of Algorithm 2 before discussing its technical details. Finally, we consider the accuracy-runtime trade-off for the LP relaxation of the algorithm.

Similar to Section 2.4.1, we refer to constraints of an IP as "constraints" and the lossless-delay and worst-case-delay constraints as "requirements" for convenience of notation. Each symbol of each frame, $S[i]$, is encoded either using the (τ, b, τ_L)-separate encoding scheme or as part of a block of the systematic rate-optimal $[\tau + b, \tau]$ block codes presented in [27]. These $[\tau + b, \tau]$ block codes have a field size requirement that is quadratic in the delay parameter τ. When any b consecutive symbols are lost, they can be recovered within τ additional symbols of the block code. Any block code designed for the streaming model, including those presented in [30, 35, 49, 52], could likewise be used and is referred to as a Streaming Block Code (SBC). The number of symbols corresponding to $S[i]$ encoded using the (τ, b, τ_L)-separate encoding scheme is denoted e_i. For $j \in \{i, \ldots, i + \tau_L\}$, the variable $k_{i,j}$ will represent the number symbols corresponding to $S[i]$ sent in $X[j]$ within blocks of the SBC. Finally, p_i will reflect the number of blocks of the SBC whose first position occurs in channel packet $X[i]$.

The lossless-delay and worst-case-delay requirements are satisfied for symbols of frames encoded using the (τ, b, τ_L)-separate encoding scheme via the properties of the (τ, b, τ_L)-separate encoding scheme detailed earlier. There must be a non-negative quantity of symbols encoded in this manner (constraint #1 in Algorithm 2). The number of symbols corresponding to frame $S[i]$ sent in channel packet j is non-negative (constraint #2). Similarly, the number of blocks corresponding to each frame is non-negative (constraint #3). All symbols of frames not encoded using the (τ, b, τ_L)-separate encoding scheme must be decoded within delay τ_L under lossless transmission (constraint #4). Finally, under the two considered code constructions, all symbols of frames which are not encoded using the (τ, b, τ_L)-separate encoding scheme must be encoded via a block of SBC which ensures decoding within the worst-case-delay requirement under lossy conditions (constraints #5 and #6). Specifically, all symbols of frame $S[i]$ for i transmitted in a time slot later than i (not as part of the (τ, b, τ_L)-separate encoding scheme) must be encoded as part of blocks whose final parity symbol is transmitted by time slot $(i + \tau)$ (constraint #5).

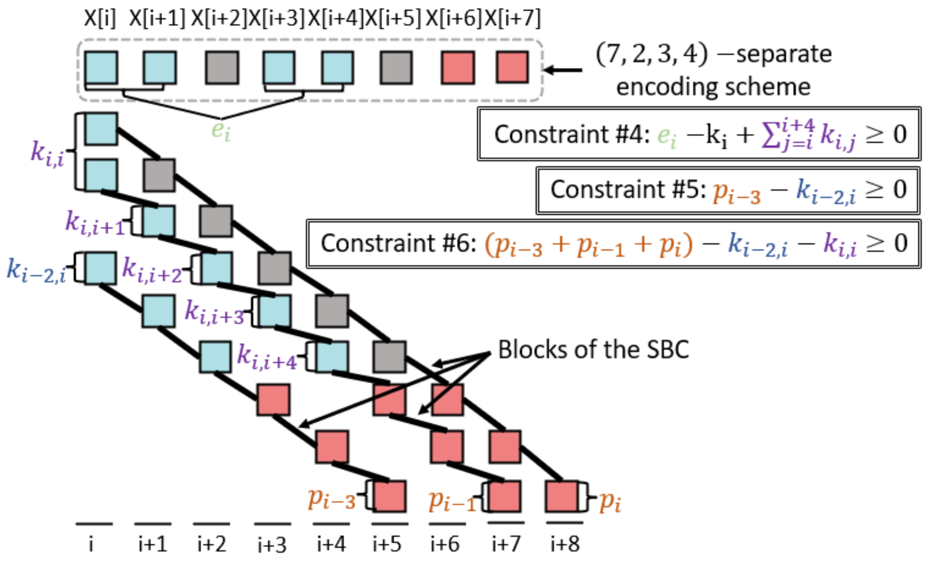

X[i] X[i+1] X[i+2] X[i+3] X[i+4] X[i+5] X[i+6] X[i+7]
(7, 2, 3, 4) −separate encoding scheme
e_i
Constraint #4: $e_i - k_i + \sum_{j=i}^{i+4} k_{i,j} \geq 0$
Constraint #5: $p_{i-3} - k_{i-2,i} \geq 0$
Constraint #6: $(p_{i-3} + p_{i-1} + p_i) - k_{i-2,i} - k_{i,i} \geq 0$
$k_{i,i}$
$k_{i,i+1}$
$k_{i-2,i}$
$k_{i,i+2}$
$k_{i,i+3}$
$k_{i,i+4}$
Blocks of the SBC
p_{i-3}
p_{i-1}
p_i
i i+1 i+2 i+3 i+4 i+5 i+6 i+7 i+8

Figure 2.5: An example imposing constraints #4, #5, and #6 for time slot i for parameters $(\tau, b, \tau_L) = (7, 3, 4)$. Blue boxes can hold symbols of frames. Red boxes hold parity symbols. Gray boxes contain no symbols. Boxes above time slot j correspond to symbols sent in channel packet $X[j]$. At least k_i symbols are sent for $S[i]$ consisting of (a) e_i symbols sent as part of the $(7, 3, 4)$-separate encoding scheme (shown at the top), and (b) $k_{i,j}$ symbols sent in channel packet $X[j]$ for $j \in \{i, \ldots, i + 4\}$ (constraint #4). There are p_{i-3} blocks of the SBC for which the final parity symbols are sent during time slot $(i + 5)$. The total number of symbols sent in channel packet $X[i]$ corresponding to frame $S[i - 2]$ (i.e., $k_{i-2,i}$) is at most p_{i-3} (constraint # 5). In addition, there are p_{i-1} and p_i blocks of the SBC for which the final parity symbols are sent during time slots $(i + 7)$ and $(i + 8)$ respectively. The number of symbols of all frames sent in channel packet $X[i]$ within blocks of the SBC (i.e., $(k_{i-2,i} + k_{i,i})$) is at most $(p_{i-3} + p_{i-1} + p_i)$ (constraint # 6).

Furthermore, all symbols for frame $S[i]$ sent in channel packet $X[i]$ are encoded via blocks of the SBC which have an open slot in position i (constraint #6). Figure 2.5 depicts a toy example of how these constraints may be applied for a single time slot, i.

The objective function of the IP used under Algorithm 2 is the total number of symbols sent via the solution to the IP. Minimizing this quantity ensures that the fewest number of symbols possible are transmitted, thereby maximizing the rate. The combined number of symbols of all frames is divided by total number of transmitted symbols to output the rate of the corresponding coding scheme.

In Theorem 2, we show that the output of Algorithm 2 is a lower bound on the optimal rate.[9]

Theorem 2. *For any valid inputs (τ, b, τ_L) and any frame-size sequence $k_0 \ldots k_t$, Algorithm 2 outputs a lower bound on the optimal rate for streaming codes that satisfy the lossless-delay requirement and worst-case-delay requirement over the $C(b, \tau)$ channel.*

[9]The extra padding symbols needed to employ the (τ, b, τ_L)-separate encoding scheme is ignored. The padding only negligibly impacts the value computed when the number of extra padding symbols is small compared to the size of each frame, as is typical.

Algorithm 2 Takes as input any valid parameters and a frame-size sequence and uses integer programming to compute a lower bound on the optimal rate for the input frame-size sequence.

```
Input: Valid $(\tau, b, \tau_L)$ and frame-size sequence $k_0 \dots k_{t-\tau}$.
Minimize $\left(\sum_{i=0}^{t-\tau} \frac{e_i}{R^{(L)}} + bp_i\right) + \left(\sum_{i=0}^{t-\tau} \sum_{j=i}^{i+\tau_L} k_{j,i}\right)$ subject to:
    1. $\forall i \in \{0, \dots, t-\tau\}, e_i \geq 0$.
    2. $\forall i \in \{0, \dots, t-\tau\}, j \in \{i, \dots, i+\tau_L\}, k_{i,j} \geq 0$.
    3. $\forall i \in \{0, \dots, t-\tau\}, p_i \geq 0$.
    4. $\forall i \in \{0, \dots t-\tau\}, e_i - k_i + \sum_{j=i}^{i+\tau_L} k_{i,j} = 0$.
    5. $\forall i \in \{0, \dots, t-\tau\}, j = \tau_L, \dots, j = 1, \sum_{z=\max(i-\tau+1,0)}^{i+1-b-j} p_z - \sum_{z=j}^{\tau_L} k_{i-z,i} \geq 0$
    6. $\forall i \in \{0, \dots, t-\tau\} \sum_{z=\max(i-\tau+1,0)}^{i} p_z - \sum_{z=0}^{\tau_L} k_{i-z,i} \geq 0$.
Output: $\frac{\sum_{i=0}^{t-\tau} k_i}{\left(\sum_{i=0}^{t-\tau} \frac{e_i}{R^{(L)}} + bp_i\right) + \left(\sum_{i=0}^{t-\tau} \sum_{j=i}^{i+\tau_L} k_{j,i}\right)}$.
```

Proof sketch. Follows from the ideas presented above. A complete proof is included in Appendix 2.5.4. □

We now demonstrate that outputs of Algorithm 2 range from $R^{(L)}$ to $R^{(U)}$. Having outputs vary over all possible values of the optimal rate for various frame-size sequences is a useful property because the optimal rate can likewise range from $R^{(L)}$ to $R^{(U)}$, as was shown in Lemma 5. For any frame-size sequence, $(k_0, \dots, k_t)$, let $Alg^{(2)}_{\tau,b,\tau_L}(k_0, \dots, k_t)$ and $Opt_{\tau,b,\tau_L}(k_0, \dots, k_t)$ denote the output of Algorithm 1 and the optimal rate respectively.

Lemma 7. *For any valid parameters (τ, b, τ_L), for all $\epsilon > 0$ and $v \in [R^{(L)}, R^{(U)}]$, there is a sequence of frame packet sizes, $(k_0, \dots, k_t)$, such that $Alg^{(2)}_{\tau,b,\tau_L}(k_0, \dots, k_t) = Opt_{\tau,b,\tau_L}(k_0, \dots, k_t)$ and $|Alg^{(2)}_{\tau,b,\tau_L}(k_0, \dots, k_t) - v| < \epsilon$.*

Proof. We will introduce a frame-size sequence whose optimal rate is within ϵ of v. Let $p, r \in \mathbb{Z}^+ \cup \{0\}$ be chosen and the quantity $R^{(p+r)} = \frac{p+r}{\frac{p}{R^{(L)}} + \frac{r}{R^{(U)}}}$ defined so that $|R^{(p+r)} - v| < \epsilon$. Let d be the smallest positive integer such that $\frac{d}{R^{(L)}}$ and $\frac{d}{R^{(U)}}$ are integers. Consider the frame-size sequence $k_0 = pd, k_j = \frac{rd}{\tau}$ for $j \in \{\tau+1, \dots, 2\tau\}$, and $k_j = 0$ for $j \in \{1, \dots, \tau\} \cup \{2\tau+1, \dots, 3\tau\}$.

The code construction presented in the proof of Lemma 5 satisfies lossless-delay requirement and worst-case-delay requirement over the $C(b, \tau)$ channel and has rate $R^{(p+r)}$. Moreover, the scheme follows from applying the (τ, b, τ_L)-separate encoding scheme to frame $S[0]$ and blocks of the SBC to frames $S[\tau+1], \dots, S[2\tau]$. Thus, the variables of the IP computed by Algorithm 2 could represent this scheme while satisfying all constraints. Therefore, Algorithm 2 will output a value of at least $R^{(p+r)}$.

As was shown in Lemma 5, $R^{(p+r)}$ is also an upper bound on the rate. Hence, the value computed by Algorithm 2 is a tight lower bound on the optimal rate. It is also within ϵ of v. □

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
τ	5	5	5	5	5	5	5	5	5
b	1	1	2	2	3	3	4	4	5
τ_L	0	4	0	3	0	2	0	1	0

Table 2.1: Parameter settings used in the empirical evaluation of the bounds on the optimal rate.

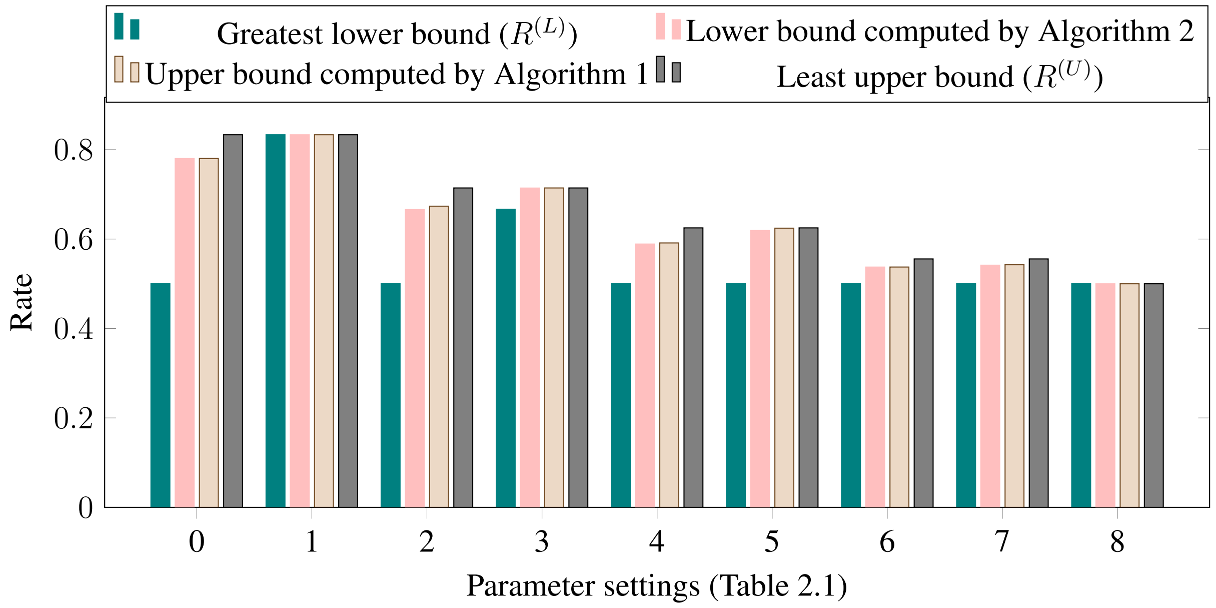


Figure 2.6: Comparison over the parameter settings listed in Table 2.1 for the live video trace shown in Figure 1.2 of the four bounds on the optimal rate for the offline setting: the greatest lower bound ($R^{(L)}$)), the lower bound computed by Algorithm 2, the upper bound computed by Algorithm 1, and the least upper bound ($R^{(U)}$).

Algorithm 2 computes a lower bound on the rate, but it can be computationally intensive. For any valid inputs (τ, b, τ_L) and frame-size sequence, consider the LP relaxation of Algorithm 2 which uses non-negative real-valued variables $e_i, k_{i,j}, p_i$ for $i \in \{\tau + b, \dots, t - (\tau + b)\}$. It is possible to transform a real-valued solution into an integral one by setting each variable to be the ceiling of its previous value. The number of symbols transmitted corresponding to each frame increases by at most $(\tau_L + 3 + b)$. In practice, $(\tau_L + 3 + b)$ is several orders of magnitude less than the average size of the frames and leads to a negligible impact on the tightness of the bound. The total number of constraints for the LP is at most $t(2\tau_L + 5)$. Hence, the number of constraints is quadratic in the input parameters (and linear in the length of the frame-size sequence).

Remark 7. *Modifying Algorithm 2 to use an LP relaxation of the underlying IP results in a polynomial-time algorithm while changing the output only negligibly.*

2.4.3 Empirical evaluation of the bounds on rate

The general upper and lower bounds on the optimal rate, $R^{(U)}$ and $R^{(L)}$, are tight for certain frame-size sequences. Yet the optimal rate for a specific frame-size sequence varies over the entire range of $[R^{(L)}, R^{(U)}]$. Thus, $R^{(U)}$ and $R^{(L)}$ can be loose depending on the frame-size sequences. In contrast, the upper and lower bounds on optimal rate computed by Algorithms 1 and 2 can range over all feasible values of the optimal rate, $[R^{(L)}, R^{(U)}]$. We evaluate the usefulness of the latter two bounds by empirically evaluating them over the live video trace shown in Figure 1.2. We show that the gap is *small in magnitude and a significant improvement over the gap between $R^{(U)}$ and $R^{(L)}$*.

We consider the setting of a small worst-case-delay (i.e., $\tau = 5$) and all parameter settings $(\tau = 5, b, \tau_L)$ where τ_L takes on its minimum and maximum values of 0 and $(\tau - b)$. Algorithms 2 and 1 bound the optimal rate significantly more tightly than $R^{(L)}$ and $R^{(U)}$ for most parameter settings. In the remaining settings, the lower and upper bounds on the optimal rate of $R^{(L)}$ and $R^{(U)}$ are tight. Specifically, the average over the parameter settings from Table 2.1 of the size of the gap between Algorithms 2 and 1 is 0.002, versus 0.106 between $R^{(U)}$ and $R^{(L)}$, as is shown in Figure 2.6. The results demonstrate the effectiveness of the algorithms in bounding the optimal rate for the offline setting.

2.5 Appendix

2.5.1 Proof of Lemma 4

Proof. The optimal rate of streaming codes that satisfy the lossless-delay and worst-case-delay constraint over the $C(a, b, w)$ model is no more than $R^{(L)}$, as was shown in Lemma 3. In order to show that $R^{(L)}$ is the greatest lower bound on rate, it suffices to show for at least one frame-size sequence that the optimal rate is at most $R^{(L)}$. We do so in this proof.

Consider the following length $(\tau + 2b)$ frame-size sequence: $k_i = 0$ for $i \in \{0, \ldots, b-1, b+1, \ldots, \tau+2b-1\}$ and $k_b = (\tau_L+1)(\tau-b+1)\left(\left\lfloor \frac{\tau_L+1}{b} \right\rfloor a + \min\left((\tau_L+1) \mod b, a\right)\right)(\tau-a+1)$.

The proof is divided into two cases to match the two cases of the construction. In both cases, we show that meeting the lossless-delay constraint and worst-case-delay constraint over the $C(a, b, w)$ requires sending at least $\frac{k_b}{R^{(L)}}$ symbols. Hence, the rate of any streaming code that satisfies the lossless-delay and worst-case-delay constraint over the $C(a, b, w)$ model is at most $R^{(L)}$ for the considered frame-size sequence. The proof will make use of the fact that $S[b]$ cannot be decoded unless at least k_b symbols are received.

Case 1 : $\tau_L < (a-1)$.

At least k_b symbols must be sent over $X[b], \ldots, X[b+\tau_L]$ to meet the lossless-delay constraint. At least k_b symbols must be sent over $X[2b], \ldots, X[b+\tau]$ to meet the worst-case-delay constraint when $X[b], \ldots, X[2b-1]$ are lost.

The average number of symbols per channel packet over $X[b], \ldots, X[b+\tau_L]$ is at least $\frac{k_b}{\tau_L+1}$. The average number of symbols per channel packet over $X[2b], \ldots, X[b+\tau]$ is at least $\frac{k_b}{\tau-b+1}$.

By definition, $(\tau_L + 1) \leq (\tau - b + 1)$.

When $a \leq (\tau_L + 1 + \tau + 1 - b)$, a arbitrary losses can result in a loss of (1) k_b symbols in $X[b], \ldots, X[b+\tau_L]$, and (2) at least $\frac{k_b}{\tau-b+1}(a - \tau_L - 1)$ symbols in $(a - \tau_L - 1)$ adversarially chosen channel packets among $X[2b], \ldots, X[b+\tau]$. Thus, at least $\frac{k_b}{\tau-b+1}(a - \tau_L - 1)$ additional symbols must be sent. Let us combine these $\frac{k_b}{\tau-b+1}(a - \tau_L - 1)$ symbols with the at least k_b symbols sent in $X[b], \ldots, X[b+\tau_L]$ and at least k_b symbols sent in $X[2b], \ldots, X[b+\tau]$. In total, at least $k_b\left(2 + \frac{a-\tau_L-1}{\tau-b+1}\right) = \frac{k_b}{R^{(L)}}$ symbols are transmitted.

When $a > (\tau_L + 1 + \tau + 1 - b)$, $(b - \tau_L - 1) > (\tau - a + 1)$. Hence, due to arbitrary losses, $X[b], \ldots, X[b+\tau_L], X[2b], \ldots, X[b+\tau]$ may all be lost. Thus, it is possible that only $(\tau - a + 1)$ arbitrary packets of $X[b+\tau_L+1], \ldots, X[2b-1]$ to be received.

As such, any $(\tau + 1 - a)$ channel packets of $X[b+\tau_L+1], \ldots, X[2b-1]$ must contain at least k_b symbols. Therefore, the channel packets $X[b+\tau_L+1], \ldots, X[2b-1]$ contain on average at least $\frac{k_b}{\tau-a+1}$ symbols. At least $(b-\tau_L-1)\frac{k_b}{\tau-a+1}$ symbols are sent over $X[b+\tau_L+1], \ldots, X[2b-1]$. In total, at least $k_b\left(2 + \frac{b-\tau_L-1}{\tau-a+1}\right)$ symbols are transmitted.

Case 2 : $\tau_L \geq (a-1)$.

Sub-case 1 : either $((\tau_L + 1) \mod b) \in \{0\} \cup \{a, \ldots, b-1\}$ or $((\tau_L + 1) \mod b)\left(\left\lfloor \frac{\tau_L+1}{b} \right\rfloor + 1\right) \geq a$.

At least k_b symbols must be sent over $X[b], \ldots, X[b+\tau_L]$ to satisfy the lossless-delay constraint. We will show in several sub-cases that at least $\frac{a}{\left\lfloor \frac{\tau_L+1}{b} \right\rfloor a + \min((\tau_L+1) \mod b, a)} k_b$ symbols could be lost by time slot $(b + \tau_L)$. At least k_b symbols must be received. Therefore, at least $k_b \frac{a}{\left\lfloor \frac{\tau_L+1}{b} \right\rfloor a + \min((\tau_L+1) \mod b, a)}$ parity symbols are transmitted. In total, $\frac{k_b}{R^{(L)}}$ symbols are sent. A final sub-case will handle the remaining parameter settings and follows from showing the correctness of IP-based construction 1.

Sub-sub-case $((\tau_L + 1) \mod b \geq a)$:

All channel packets of one of

$$\begin{aligned}
&\big(X\left[b\right], \ldots, X\left[2b-1\right]\big), \ldots, \\
&\left(X\left[b + \left(\left\lfloor \frac{\tau_L}{b} \right\rfloor - 1\right)b\right], \ldots, X\left[b + \left\lfloor \frac{\tau_L}{b} \right\rfloor b - 1\right]\right), \\
&\left(X\left[b + \left\lfloor \frac{\tau_L}{b} \right\rfloor b\right], \ldots, X\left[b + \tau_L\right]\right)
\end{aligned}$$

could be dropped as part of a single burst. There are $\left(\left\lfloor \frac{\tau_L}{b} \right\rfloor + 1\right)$ quantities, and at least k_b symbols are sent over $X[b], \ldots, X[b+\tau_L]$. By the pigeonhole principle, at least one such quantity contains at least $\frac{k_b}{\left\lfloor \frac{\tau_L+1}{b} \right\rfloor + 1} = \frac{a k_b}{\left\lfloor \frac{\tau_L+1}{b} \right\rfloor a + \min((\tau_L+1) \mod b, a)}$ symbols.

Sub-sub-case $((\tau_L + 1) \mod b \equiv 0)$:

All channel packets of one of

$$\begin{aligned}
&\big(X\left[b\right], \ldots, X\left[2b-1\right]\big), \ldots, \\
&\left(X\left[b + \left\lfloor \frac{\tau_L}{b} \right\rfloor b\right], \ldots, X\left[b + \left(\left\lfloor \frac{\tau_L}{b} \right\rfloor + 1\right)b - 1\right]\right)
\end{aligned}$$

could be dropped as part of a single burst. There are $\left(\left\lfloor \frac{\tau_L}{b} \right\rfloor + 1\right) = \frac{\tau_L+1}{b}$ such quantities. By the pigeonhole principle, at least one contains at least $\frac{k_b}{\left\lfloor \frac{\tau_L+1}{b} \right\rfloor} = \frac{a}{\left\lfloor \frac{\tau_L+1}{b} \right\rfloor a + \min((\tau_L+1) \mod b, a)} k_b$ symbols.

Sub-sub-case $0 < (\tau_L + 1) \mod b < a$ and $((\tau_L + 1) \mod b)\left(\left\lfloor\frac{\tau_L+1}{b}\right\rfloor + 1\right) \geq a$:

Note that $(\tau_L + 1 \neq b)$ by the sub case. Also, $(\tau_L + 1)$ must be strictly greater than b in accordance with $(\tau_L \geq a - 1)$.

Let $e = ((\tau_L + 1) \mod b)$. If any b consecutive channel packets of $X[b], \ldots, X[b + \tau_L]$ contains at least $k_b \frac{a}{\left\lfloor\frac{\tau_L+1}{b}\right\rfloor a+e}$ symbols the proof is immediate, since all b of them could be lost. Otherwise, let

$$X' = \bigcup_{i=0}^{\left\lfloor\frac{\tau_L+1}{b}\right\rfloor} \left\{(X[b + ib], \ldots, X[b + ib + e - 1])\right\}.$$

Consider any $(X[j], \ldots, X[j+e-1]) \in X'$. The remaining channel packets of $X[b], \ldots, X[b+\tau_L]$ can be partitioned into $\left\lfloor\frac{\tau_L+1}{b}\right\rfloor$ groups of b consecutive channel packet. Recall that each group of b consecutive packets contains at most $k_b \frac{a}{\left\lfloor\frac{\tau_L+1}{b}\right\rfloor a+e}$ symbols. In total, the $\left\lfloor\frac{\tau_L+1}{b}\right\rfloor$ groups contain at most $k_b \frac{a}{\left\lfloor\frac{\tau_L+1}{b}\right\rfloor a+e} \left\lfloor\frac{\tau_L+1}{b}\right\rfloor$ symbols. In order to satisfy the lossless-delay, at least k_b symbols must be received over $X[b], \ldots, X[b + \tau_L]$. Hence, the total combined number of symbols in $X[j], \ldots, X[j + e - 1]$ is at least the following

$$\left(1 - \frac{(\left\lfloor\frac{\tau_L+1}{b}\right\rfloor)a}{\left\lfloor\frac{\tau_L+1}{b}\right\rfloor a + e}\right) k_b = \frac{e}{\left\lfloor\frac{\tau_L+1}{b}\right\rfloor a + e} k_b.$$

There are $e\left(\left\lfloor\frac{\tau_L+1}{b}\right\rfloor + 1\right) \geq a$ channel packets in X. Each of thes channel packets lies within $X[b], \ldots, X[b + \tau_L]$. In total, these channel packets contain at least $\left(\left\lfloor\frac{\tau_L+1}{b}\right\rfloor + 1\right) \frac{e}{\left\lfloor\frac{\tau_L+1}{b}\right\rfloor a+e} k_b$ symbols. The channel packets in X contain on average at least $\frac{1}{\left\lfloor\frac{\tau_L+1}{b}\right\rfloor a+e} k_b$ symbols. In expectation, if a of these channel packets are dropped uniformly at random, at least $\frac{a}{\left\lfloor\frac{\tau_L+1}{b}\right\rfloor a+e} k_b$ symbols are lost. Thus, at least one choice of a arbitrary channel packet losses results in a total number of lost symbols of at least $\frac{a}{\left\lfloor\frac{\tau_L+1}{b}\right\rfloor a+e} k_b$.

In all above sub-cases for $\tau_L \geq (a - 1)$, at least $\frac{a}{\left\lfloor\frac{\tau_L+1}{b}\right\rfloor a+\min((\tau_L+1) \mod b, a)} k_b$ symbols could be lost. This necessitates transmitting at least $k_b(1+\frac{a}{\left\lfloor\frac{\tau_L+1}{b}\right\rfloor a+\min((\tau_L+1) \mod b, a)}) = \frac{k_b}{R^{(L)}}$ symbols.

Sub-case 2 : $((\tau_L + 1) \mod b)\left(\left\lfloor\frac{\tau_L+1}{b}\right\rfloor + 1\right) < a$.

For this case, recall that the number of symbols sent per channel packet is determined by IP-based construction 1. Each channel packet contains a non-negative number of symbols, as is imposed by constraint #1. At least k_i symbols must be received within the first τ_L channel packets due to the lossless-delay constraint. This requirement is imposed with constraint #2. For any loss pattern under the $C(a, b, w)$ channel, the total number of symbols over the received channel packets must be at least k_i. This requirement is imposed with constraints #3 and #4 of the integer program. All constraints of the integer program must be met by any code construction. The integer program solves for the minimum number of symbols to be sent subject to these three constraints. Hence, the optimal rate is attained.

$\square$

2.5.2 Proof of Lemma 2

Proof. Consider the encoding for a frame $S[i]$ for $i \in \{0,\dots,t\}$. When $i > (t-\tau)$, $k_i = 0$, and $S[i]$ is automatically known by the receiver. Otherwise, the symbols frame $S[i]$ are sent over $X[i],\dots,X[i+\tau_L]$, thereby satisfying the lossless-delay constraint.

The proof that the worst-case-delay constraint is satisfied over the $C(a,b,w)$ channel is divided into two cases depending on whether $\tau_L \geq (a-1)$.

Case 1 : $\tau_L < (a-1)$.

If $a \leq (\tau_L + 2 + \tau - b)$, let $\eta = (\tau_L+1)(\tau-b+1)$ and $\eta' = (\tau_L+1)(\tau-b+a-\tau_L)$. Otherwise, let $\eta = (\tau_L+1)(\tau-b+1)(\tau-a+1)$ and $\eta' = \eta + (\tau_L+1)(\tau-b+1)(b-\tau_L-1)$. The frame, $S[i]$, is partitioned evenly into sets of η symbols. For an arbitrary such set, $\{c_0,\dots,c_{\eta-1}\}$, we verify all symbols are decoded within τ time slots. The set is encoded as part of a $[\eta+\eta',\eta]$ systematic MDS code. Let $(c'_0,\dots,c'_{\eta+\eta'})$ be the code symbols corresponding to $\{c_0,\dots,c_\eta\}$. It suffices to show that at least η symbols of $(c'_0,\dots,c'_{\eta+\eta'})$ are received by time slot $(i+\tau)$.

We note that $(\tau_L + b \leq \tau) \implies (\tau - b + 1 \geq \tau_L + 1)$. Furthermore, $X[i],\dots,X[i+\tau_L]$ and $X[i+b],\dots,X[i+\tau]$ each contain η symbols of $(c'_0,\dots,c'_{\eta+\eta'})$. The number of symbols per channel packet in $X[i],\dots,X[i+\tau_L]$ is $\frac{\eta}{\tau_L+1}$. The number of symbols per channel packet in $X[i+b],\dots,X[i+\tau]$ is $\frac{\eta}{\tau-b+1}$. If $a \leq (\tau_L+2+\tau-b)$, the number of symbols per channel packet in $X[i+\tau_L+1],\dots,X[b-1]$ is either 0 or $\frac{\eta}{\tau-b+1}$. If $a > (\tau_L+2+\tau-b)$, the number of symbols per channel packet in $X[i+\tau_L+1],\dots,X[b-1]$ is $\frac{\eta}{\tau-a+1}$. Finally, $\frac{\eta}{\tau-a+1} \leq \frac{\eta}{\tau-b+1} \leq \frac{\eta}{\tau_L+1}$

Burst losses: For all bursts of length b starting during or after time slot $(i+\tau_L+1)$, η symbols of $(c'_0,\dots,c'_{\eta+\eta'})$ are received via $X[i],\dots,X[i+\tau_L]$. Hence, decoding within a delay of τ follows immediately. For any burst starting in channel packet $X[i+j]$ for $j \in \{0,\dots,\tau_L\}$, $(\tau_L - j + 1)$ channel packets $X[i],\dots,X[i+j-1]$ are received, each of which contain $\frac{\eta}{\tau_L+1}$ symbols of $(c'_0,\dots,c'_{\eta+\eta'})$. Moreover, channel packets $X[i+j+b],\dots,X[i+\tau]$ are received, each of which contain $\frac{\eta}{\tau-b+1}$ symbols of $(c'_0,\dots,c'_{\eta+\eta'})$. The fewest symbols are received when $j=0$, in which case exactly η symbols are received. Any η symbols are sufficient for decoding.

Arbitrary losses: The maximum number of symbols of $(c'_0,\dots,c'_{\eta+\eta'})$ are lost when the a largest channel packets are lost, such as when

$$X[i],\dots,X[i+\tau_L],X[i+\tau-a+\tau_L+2],\dots,X[i+\tau]$$

are lost. If $a \leq (\tau_L+2+\tau-b)$, the $(\tau-b+1)$ channel packets $X[i+b-a+\tau_L+1],\dots,X[i+\tau-a+\tau_L+1]$ are received, each of which contains $\frac{\eta}{\tau-b+1}$ symbols. If $a > (\tau_L+2+\tau-b)$, then the $(\tau-a+1)$ channel packets $X[i+\tau_L+1],\dots,X[i+\tau-a+\tau_L+1]$ are received, each of which contains $\frac{\eta}{\tau-a+1}$ symbols. Therefore, at least η symbols of $(c'_0,\dots,c'_{\eta+\eta'})$ are received within a delay of τ, enabling decoding.

Case 2 : $\tau_L \geq (a-1)$.

Sub-case 1 : $((\tau_L+1) \mod b) \in \{0\} \cup \{a,\dots,b-1\})$ or $(0 < (\tau_L+1) \mod b < a$ and $((\tau_L+1) \mod b)\left(\left\lfloor\frac{\tau_L+}{b}\right\rfloor + 1\right) \geq a)$.

Let $\zeta = \left(\left\lfloor\frac{\tau_L+1}{b}\right\rfloor a + \min\left((\tau_L+1) \mod b, a\right)\right)$. Recall that each $S[i]$, for $i \in \{0,\dots,t-\tau\}$, is partitioned into sets of ζ symbols. Every such set is encoded separately as part of a $[\zeta+a,\zeta]$ systematic MDS code. We verify for an arbitrary such set, $\{c_0,\dots,c_{\zeta-1}\}$, with corresponding code symbols, $(c'_0,\dots,c'_{\zeta+a})$, that at least ζ code symbols are received within τ time slots. Any ζ symbols suffice to decode $\{c_0,\dots,c_{\zeta-1}\}$.

Every burst of b consecutive channel packets eliminates at least $(b-a)$ channel packets which contain no symbols of $(c'_0, \ldots, c'_{\zeta+a})$. Thus, at most a symbols of $(c'_0, \ldots, c'_{\zeta+a})$ are lost. For any sequence of a arbitrary losses, at least ζ of the symbols of $(c'_0, \ldots, c'_{\zeta+a})$ are received within τ time slots. For either loss pattern, recovery with a delay of τ time slots follows immediately by properties of the $[\zeta + a, \zeta]$ systematic MDS code.

Sub-case 2 : $(0 < (\tau_L + 1) \mod b < a$ and $((\tau_L + 1) \mod b)\left(\left\lfloor \frac{\tau_L+1b}{b} \right\rfloor + 1\right) < a$.

Recall that each $S[i]$ is encoded according to the outputs of IP-based construction 1. Constraints #3 and #4 of IP-based construction 1 ensure that least k_i symbols of an $[n_i^{(*)}, k_i]$ MDS code are received by time slot $(i + \tau)$. Hence, $S[i]$ is recovered within τ time slots by the MDS property.

□

2.5.3 Proof of Theorem 1

Proof. We will show that each constraint corresponds to a valid requirement to impose on coding schemes. This ensures that the solution is a lower bound on the number of symbols that must be sent. As there are $\sum_{i=0}^{t} k_i$ symbols of the frames, the output must be an upper bound on the rate.

Before presenting the proof of correctness for the constraints, we will formally define how a channel packet, $X[i]$ for $i \in \{0, \ldots, t\}$, is split into $(X^{(0)}[i], X^{(1)}[i])$. Recall that each symbol of $X[i]$ comprises a linear combination of the symbols of

$$\langle S_0[0], \ldots, S_{k_0-1}[0], \ldots, S_0[i], \ldots, S_{k_i-1}[i] \rangle . \tag{2.9}$$

The symbols

$$\left\langle X_0[0], \ldots, X_{|X[0]|-1}[0], \ldots, X_0[i], \ldots, X_{|X[i-1]|-1}[i-1] \right\rangle \tag{2.10}$$

correspond to linear equations over the symbols of Equation 2.9 where the linear equations for each $j \in \{0, \ldots, i-1\}$ of $X[j]$ have 0 in positions corresponding to $\left\langle S_0[j+1], \ldots, S_{|k_i|-1}[i] \right\rangle$ due to causality. Next, the symbols of channel packet $X[i]$ are partitioned into $(X^{(0)}[i], X^{(1)}[i])$. Initially, consider $X^{(0)}[i]$ and $X^{(1)}[i]$ as being empty. The symbols of $X[i]$ are labeled as being in either $X^{(0)}[i]$ or $X^{(1)}[i]$ by iterating over $j \in \{0, \ldots, |X[i] - 1|\}$ as follows. If the set of linear equations corresponding $X_j[i]$, the symbols of $X^{(0)}[i]$, and the symbols of Equation 2.10 are linearly independent, $X_j[i]$ is added to $X^{(0)}[i]$. Otherwise, $X_j[i]$ is added to $X^{(1)}[i]$.

Constraints #1 and #2: For $i \in \{0, \ldots, t - \tau_L\}, j \in \{i - \tau_L, \ldots, i\}$, $|X_i^{(0)}[i+j]|$ reflects a number of symbols sent in channel packet $X[i+j]$ corresponding to frame $S[i]$, so it is non-negative. For $i < 0$, $i > t$, or $j < 0$, $|X_j^{(0)}[i]|$ is defined to be 0 to handle edge conditions of indexing. For $i \in \{0, \ldots, t + \tau\}$, $|X^{(1)}[i]|$ corresponds to a number of parity symbols sent in channel packet $X[i]$ and similarly is non-negative.

Constraint #3: We will show that any construction satisfying the lossless-delay requirement, even with the relaxations allowed under Algorithm 1, will satisfy constraint #3. We will prove this by induction on $i \in \{0, \ldots, t\}$. In the base case, $i = 0$ and $X_0^{(0)}[0], \ldots, X_0^{(0)}[\tau_L]$ consist of exactly k_0 symbols used to decode $S[0]$ under lossless transmission. In the inductive step, for $i = 1, \ldots, t$, $S[0], \ldots, S[i]$ can be decoded using channel packets $X[0], \ldots, X[i + \tau_L]$ by solving a system of linear equations. Only the symbols corresponding to $X^{(0)}[0], \ldots, X^{(0)}[i + \tau_L]$ need to be used, since the linear equations corresponding to $X^{(1)}[0],\dots,X^{(1)}[i+\tau_L]$ are in their span. Moreover, the linear equations corresponding to the symbols of $X^{(0)}[0],\dots,X^{(0)}[i+\tau_L]$ are linearly independent by definition. By induction, $\sum_{j=0}^{i-1}\sum_{l=0}^{\tau_L}|X_j^{(0)}[j+l]| = \sum_{j=0}^{i-1}k_j$. The symbols of $X_j^{(0)}[j+l]$ in the first term reflect the symbols sent in channel packet $X[j+l]$ used to decode frame $S[j]$. When $S[i]$ is decoded, at least $\sum_{j=0}^{i}k_j$ of the symbols of $X^{(0)}[0],\dots,X^{(0)}[i+\tau_L]$ are required to decode $S[0],\dots,S[i]$ (along with perhaps some additional symbols of $S[i+1],\dots,S[i+\tau_L]$). For $j\in\{0,\dots,i+\tau_L\}$, each symbol of $X^{(0)}[j]$ included reflects adding a linearly independent equation. $S[0],\dots,S[i]$, along with perhaps some additional symbols of $S[i+1],\dots,S[i+\tau_L]$, are decoded together. Exactly k_i equations (corresponding to symbols) used in decoding are used to decode symbols of $S[i]$. Since the encoding is causal, each of these k_i equations correspond to symbols are sent in channel packets $X[i],\dots,X[i+\tau_L]$ and are labeled $X_i^{(0)}[i],\dots,X_i^{(0)}[i+\tau_L]$.

Constraint #4: Consider any burst starting in time slot $i\in\{0,\dots,t-b+1\}$. The proof of Lemma 1 showed that satisfaction of the worst-case-delay requirement over the $C(a,b,w)$ channel implies that frames $S[i-\tau_L],\dots,S[i+b-1]$ must be decoded by time slot $(i+\tau+b-a)$. Moreover, for any burst loss of length b starting in channel packet i, for each of $j\in\{i-\tau_L,\dots,i+b-1\}$, $S[i-\tau_L],\dots,S[j]$ must be recoverable by time slots $(i-\tau_L+\tau),\dots,(j+\tau)$ respectively. Frames $S[i-\tau_L],\dots,S[j]$ are therefore decoded by time slot $(j+\tau)$. We consider the relaxation that symbols of $X_l^{(0)}[z]$ for $l\in\{j+1,\dots,i+b-1\}, z\in\{l,\dots,l+\tau_L\}$ are received.

We consider the following relaxation. Each symbol corresponding to $X_l^{(0)}[z]$ where $l\in\{i-\tau_L,\dots,j\}, z\in\{l,\dots,l+\tau_L\}$ which are received can be used to decode one symbol of $S[i-\tau_L],\dots,S[j]$. These symbols are received during time slots $(i-\tau_L),\dots,(i-1)$ and $(i+b),\dots,\min(j+\tau,i+\tau+b-a)$. Furthermore, parity symbols received during time slots $(i+b),\dots,\min(j+\tau,i+\tau+b-a)$ can be used to decode frames $S[i-\tau_L],\dots,S[j]$. However, by definition any parity symbols sent before time slot i are in the span of symbols of $X_l^{(0)}[z]$ for $l\in\{0,\dots,i-1\}, z\in\{l,\dots,\min(i-1,l+\tau_L)\}$. Therefore, given access to the symbols of $X_l^{(0)}[z]$, these parity symbols are not used to decode frames $S[i-\tau_L],\dots,S[j]$. All symbols received strictly after $\min(j+\tau,i+\tau+b-a)$ are received after frames $S[i-\tau_L],\dots,S[j]$ have already been decoded. Thus, the symbols of $S=$

$$\begin{aligned}&\big\{X_l^{(0)}[z]\mid l\in\{0,\dots,t\}, z\in\{l,\dots,l+\tau_L\}\setminus\\&\{i,\dots,i+b-1\}\big\}\cup\\&\big\{X_l^{(0)}[z]\mid l\in\{j+1,\dots,i+b-1\}, z\in\{l,\dots,l+\tau_L\}\big\}\cup\\&\big\{X^{(1)}[z]\mid z\in\{i+b,\dots,\min(j+\tau,i+\tau+b-a)\}\big\}\end{aligned}$$

must be sufficient to decode $(S[0],\dots,S[t])$. Therefore, $|S|\ge\sum_{i=0}^{t}k_i$. By constraint #3, $\sum_{z=l}^{l+\tau_L}|X_l^{(0)}[z]|=k_l$ for any $l\in\{0,\dots,t-\tau\}$. Thus, the size of

$$\begin{aligned}&\big\{X_l^{(0)}[z]\mid l\in\{i-\tau_L,\dots,j\}, z\in\{l,\dots,l+\tau_L\}\setminus\\&\{i,\dots,i+b-1\}\big\}\cup\\&\big\{X^{(1)}[z]\mid z\in\{i+b,\dots,\min(j+\tau,i+\tau+b-a)\}\big\}\end{aligned}$$

is at least $\sum_{l=i-\tau_L}^{j} k_l$.

Constraint #5: Consider any combination of a arbitrary packet losses in a sliding window of length $w = (\tau + 1)$ which begin during some time slot $i \in \{0, \ldots, t - a + 1\}$. Let the time slots of the packet losses be denoted as I, and let i' be the final time slot in I where $i' \leq t$. For any $j \in \{0, \ldots, i'\}$, each of $S[i - \tau_L], \ldots, S[j]$ must be decoded by time slot $(j + \tau)$ in order to satisfy the worst-case-delay requirement. The relaxation is used that each received symbol of $X_l^{(0)}[z]$ for $l \in \{i - \tau_L, \ldots, j\}, z \in \{l, \ldots, l + \tau_L\}$ can be used to decode one symbol of $S[i - \tau_L], \ldots, S[j]$. Furthermore, the relaxation is taken that each received symbol $X^{(1)}[l]$ for $l \in \{i + 1, \ldots, j + \tau\}$ can be used to decode one lost symbol of $S[i - \tau_L], \ldots, S[j]$. We assume $X_l^{(0)}[z]$ is received for $l \in \{0, \ldots, i-1\}, z \in \{l, \ldots, \min(l+\tau_L, i-1)\}$, so the received symbols of $X^{(1)}[0], \ldots, X^{(1)}[i-1]$ are not useful for decoding $S[i-\tau_L], \ldots, S[j]$. Moreover, all symbols received strictly after time slot $(j+\tau)$ are not used in decoding $S[i-\tau_L], \ldots, S[j]$, since they are decoded by time slot $(j+\tau)$. This follows from similar reasoning to that discussed for constraint #4. Hence, the symbols of $S =$

$$\begin{aligned}
&\big\{X_l^{(0)}[z] \mid l \in \{i - \tau_L, \ldots, j\}, z \in \{l, \ldots, l + \tau_L\} \setminus I\big\} \cup \\
&\big\{X^{(1)}[j] \mid j \in \{i + 1, \ldots, j + \tau\} \setminus I\big\} \cup \\
&\big\{X_l^{(0)}[z] \mid l \in \{0, \ldots, i - \tau_L - 1\} \cup \{i' + 1, \ldots, t\}, \\
&\quad z \in \{l, \ldots, l + \tau_L\}\big\}
\end{aligned}$$

are sufficient to decode $S[0], \ldots, S[t]$. Consequently, the size of S must be at least $\sum_{l=0}^{t} k_l$. Similar to the discussion for constraint #4, $\sum_{z=l}^{l+\tau_L} |X_l^{(0)}[z]| = k_l$ for any $l \in \{0, \ldots, t - \tau\}$. Thus, as a relaxation of the worst-case-delay requirement for at most a arbitrary losses, the size of

$$\begin{aligned}
&\big\{X_l^{(0)}[z] \mid l \in \{i - \tau_L, \ldots, j\}, z \in \{l, \ldots, l + \tau_L\} \setminus I\big\} \cup \\
&\big\{X^{(1)}[j] \mid j \in \{i + 1, \ldots, j + \tau\} \setminus I\big\}
\end{aligned}$$

must be at least $\sum_{z=i-\tau_L}^{j} k_z$. □

2.5.4 Proof of Theorem 2

Proof. The value computed by Algorithm 2 is the optimal rate for a coding scheme that combines (1) blocks of the SBC with (2) the (τ, a, b, τ_L)-separate encoding scheme. Therefore, this rate is feasible. The total number of symbols of all frames divided by the total number of symbols transmitted by the scheme is returned. Hence, the rate of the corresponding coding scheme is returned. The objective function is to compute the minimal possible value for the number of symbols transmitted over the considered coding schemes for the frame-size sequence. For a fixed total number of symbols of all frames, this minimizes the rate.

It remains to verify that the lossless-delay and worst-case-delay requirements are satisfied. We do so for (a) the symbols of frames encoded as part of the (τ, a, b, τ_L)-separate encoding scheme, and (b) for all remaining symbols. For $i > (t - \tau)$, the frame $S[i]$ is of size 0 and is automatically known by the receiver. Hence, the lossless-delay and worst-case-delay requirements are satisfied for such frames.

First, a non-negative number of symbols of any $S[i]$, for $i \in \{0,\ldots,t-\tau\}$, are modeled as being encoded using the (τ,a,b,τ_L)-separate encoding scheme for each frame due to constraint #1. The lossless-delay and worst-case-delay are met for such symbols by properties of the (τ,a,b,τ_L)-separate encoding scheme.

Second, we verify that all remaining symbols are accurately modeled as being encoded within blocks of the SBC such that the lossless-delay and worst-case-delay requirements are met.

The lossless-delay requirement: For any $S[i]$, for $i \in \{0,\ldots,t-\tau\}$, a non-negative number of symbols are modeled as sent in each of $X[i],\ldots,X[\tau_L+1]$, as is reflected by constraint #2. Moreover, the total number of such symbols is sufficient to satisfy the lossless-delay requirement by constraint #4.

The worst-case-delay requirement: A non-negative number of blocks starting in each channel packet is modeled due to constraint #3. Recall that the variables p_i reflect the quantity of blocks whose first position occurs during time slot i. The first position of blocks occur between time slot 0 and $(t-\tau)$. It remains to verify that all symbols not encoded using the (τ,a,b,τ_L)-separate encoding scheme can be modeled as being placed in the corresponding blocks which ensure that they are decoded within τ time slots. Under constraint #5, each $j \in \{\tau_L,\ldots,1\}$ is sequentially considered for each channel packet $X[i]$. Without loss of generality, each symbol sent in channel packet $X[i]$ corresponding to frame $(i-j)$ is modeled as being placed in the earliest block ending by time slot $(i-j+\tau)$ with an available position in channel packet $X[i]$. Sequentially doing so ensures that all symbols corresponding to frames $(i-\tau_L),\ldots,(i-1)$ sent in channel packet $X[i]$ are modeled as being encoded as part of by blocks of the SBC whose final position occur by the time slots $(i-\tau_L+\tau),\ldots,(i+\tau-1)$ respectively. This ensures that the worst-case-delay is satisfied over a $C(a,b,w)$ channel for such symbols. Finally, due to constraint #6, each symbol corresponding to $S[i]$ which is modeled as being transmitted without delay in $X[i]$ is modeled as being placed in an available block of the SBC. This ensures recovery within τ time slots over a $C(a,b,w)$ channel. In so doing, all blocks which have an available position in channel packet $X[i]$ are considered, as the recovery properties of the SBC ensure recovery within τ time slots. Furthermore, all symbols corresponding to frames $S[i-\tau_L],\ldots,S[i-1]$ sent in channel packet $X[i]$ still must also be modeled as being placed in available blocks of the SBC, as is reflected in constraint #6.

$\square$

Chapter 3

Online versus offline rate in streaming codes for variable-size frames

Recall that while there are rate-optimal schemes that send each frame in the corresponding channel packet for the setting of fixed-size frames, spreading frame symbols over multiple channel packets is advantageous in the setting of variable-size frames (see Section 2.2). One key challenge in realizing the benefits of spreading is determining how to best spread frame symbols over one or more channel packets despite the fact that future frames' sizes are inherently variable and unknown. For example, a large frame should be sent in the corresponding channel packet when the next several frames are even larger to reduce the variability in the sizes of channel packets. In contrast, frame symbols of a large frame should be spread over multiple channel packets when the subsequent several frames are small. Thus, the optimal strategy for encoding depends on the sizes of future frames. To capture this dependency introduced by the variability in the size of frames, the coding schemes are classified into two classes: (a)"offline" schemes and (b) "online" schemes. Offline coding schemes have access to the *sizes* of frames of *future* time slots, whereas online schemes do not have access to such information. Online constructions are of practical interest, as the sizes of future frames are typically unknown in live streaming applications. By using future information, optimal offline constructions can always match, and potentially significantly exceed, the rate of online ones. Therefore, a natural question is: "can online coding schemes match the rate of offline coding schemes?"

This question was answered for Regime 2, where the (τ, b, τ_L)-separate encoding scheme matches the optimal rate of offline constructions (see Corollary 1). More generally, we introduce the first rate-optimal online coding scheme for Regime 1 (i.e., $\tau_L = 0$ while τ and b have arbitrary values) in Section 3.2. This setting is crucial for the most latency sensitive applications, as each frame is recovered with no delay under lossless transmission. We then show in Section 3.3 that no online coding scheme can match the rate of optimal offline coding schemes for all parameter settings outside of Regime 1 and Regime 2. Before doing so, we tweak the model and provide additional notation in Section 3.1.

3.1 Modifications to model and notation

Constructions that during the time slot $i \in \{0, \ldots, t\}$ can access all future frames' sizes (i.e., $k_{i+1}, \ldots, k_t$) are called "offline." Offline schemes have access to the *sizes* but not the *symbols* of the future frames. In contrast, code constructions that do not know the sizes of the future frames are dubbed "online." Specifically, during time slot i, $(k_{i+1}, \ldots, k_t)$ are unknown for an online construction. We distinguish between the feasible rates for offline and online coding schemes. The best possible rate for offline coding schemes is called the "offline-optimal-rate" and for online coding schemes is called the "online-optimal-rate."

To distinguish between online and offline decoding, we use the following quantity to denote the last time slot for which the size of frames is available to the receiver

$$\lambda_i = \begin{cases} t & \text{if offline} \\ \arg\max_{l \in \{i, \ldots, i+\tau\}} \mathbb{1}\left[Y[l] == X[l]\right] & \text{if online.} \end{cases}$$

The decoding for frame $S[i]$ is then defined for two scenarios. First, in a lossless transmission, $S[i]$ is decoded using (a) the previously decoded frames, (b) the $(\tau_L + 1)$ channel packets received within lossless-delay, and (c) the sizes of the first $(i + \tau_L + 1)$ frames as follows:

$$S[i] = Dec^{(L)}\left(S[i-\tau], \ldots, S[i-1], X[i], \ldots, X[i+\tau_L], k_0, \ldots, k_{i+\tau_L}\right).$$

Second, when losses occur, $S[i]$ is decoded using (a) the previously decoded frames, (b) all received channel packets among the $(\tau + 1)$ sent within the worst-case-delay, and (c) the sizes of the first $(\lambda_i + 1)$ frames as follows:

$$S[i] = Dec\left(S[i-\tau], \ldots, S[i-1], Y[i], \ldots, Y[i+\tau], k_0, \ldots, k_{\lambda_{i+\tau}}\right).$$

To ensure that the receiver knows the sizes of frames, a small header containing $k_{i-b}, \ldots, k_i$ is added to $X[i]$.[1]

This work uses the following notation and conventions. The term $[n]$ denotes $\{0, \ldots, n\}$. All vectors are row vectors. A vector V has length v and is indexed as $V = (V_0, \ldots, V_{v-1})$. For $I = \{i_0, \ldots, i_l\} \subseteq [v-1]$, $V_I = (V_{i_0}, \ldots, V_{i_l})$. Let A be an $n \times n$ matrix, and $I \subseteq \{0, \ldots, n-1\}$. Then A_I is A restricted to the columns in I. For $i \in \{1-b, \ldots, -1\} \cup \{t+1, \ldots, t+b+1\}$, k_i is defined as 0. For $i \in \{1-b, \ldots, -1\}$, a burst loss of $X[i], \ldots, X[i+b-1]$ denotes a burst loss of $X[0], \ldots, X[i+b-1]$. Similarly, for $i \in \{t-b+2, \ldots, t\}$ a burst loss of $X[i], \ldots, X[i+b-1]$ denotes a burst loss of $X[i], \ldots, X[t]$.

3.2 Online Code Constructions with Optimal Rate

In this section, we present the first rate-optimal online streaming codes, as well as show that they match the offline-optimal-rate, for two broad parameter regimes: ***Regime*** 1*:* ($\tau_L = 0$ and any b and τ) and ***Regime*** 2*:* ($\tau_L = (\tau - b)$ and $b | \tau$).

[1]In the edge conditions, $(i-\tau)$ is set to 0 for $i < \tau$ and $(i+\tau)$ is set to t for $(i-\tau) > (t-\tau)$.

To begin, we consider Regime 1 (i.e., $\tau_L = 0$ and any b and τ). In this regime, the lossless-delay constraint, $\tau_L = 0$, eliminates the choice of distributing symbols corresponding to a frame over multiple channel packets. We introduce a systematic construction that sends each frame within the corresponding channel packet. The construction employs an online greedy paradigm for sending parity symbols. The approach involves (a) identifying during time slot i how many parity symbols will be sent during time slot $(i+\tau)$ (i.e., in advance τ time slots), and (b) defining the parity symbols only during time slot $(i+\tau)$ based on the sizes of $S[i+1], \ldots, S[i+\tau-1]$. To show that the construction is rate-optimal, we demonstrate via induction that the cumulative number of symbols sent by each time slot $i \in [t]$ is no more than that which is sent under an arbitrary offline construction.

We next present the rate-optimal online coding scheme for any (τ, b) under Regime 1. The scheme builds on top of the Generalized Maximally Short Codes presented in [11] in such a way so as to mitigate the adverse effects of the variability of the frame-size sequence. We call the proposed scheme the **(τ, b)-Variable-sized Generalized MS Code**. The construction is suitable for any field of size at least $2\tau m$. We first provide a high-level description, then present a toy example, and finally present the details of the code construction.

Encoding (high level description). During time slot i, each frame $S[i]$ is partitioned into two pieces: $S[i] = (U[i], V[i])$. The channel packet $X[i] = (S[i], P[i])$ is then sent, where $P[i]$ comprises parity symbols. The parity symbols are defined as $P[i] = (U[i-\tau] + P'[i])$ where $P'[i]$ consists of carefully designed linear combinations of the symbols of $(V[i-\tau], \ldots, V[i-1])$. The linear equations are defined so that that for all $i \in [t-\tau-b+1]$, $P'[i+b], \ldots, P'[i+\tau-1], V[0], \ldots, V[i-1]$ are sufficient to decode $V[i], \ldots, V[i+b-1]$, as will be fully explained in the detailed description.[2]

We set $V[i]$ to contain as many symbols of $S[i]$ as possible while meeting the following requirement. For any $j \in \{i-b+1, \ldots, i\}$ and burst loss of $X[j], \ldots, X[j+b-1]$, the sum of the sizes of $V[j], \ldots, V[i]$ is at most the number of parity symbols in $X[j+b], \ldots, X[j+\tau-1]$ (i.e., the sum of the sizes of $P[j+b], \ldots, P[j+\tau-1]$). The remaining symbols of $S[i]$ are allocated to $U[i]$. The size of $P[i]$ is set to equal that of $U[i-\tau]$.

Decoding (high level description). A burst loss of $X[i], \ldots, X[i+b-1]$ is recovered in two steps. First, for $j \in \{i+b, \ldots, i+\tau-1\}$, $U[j-\tau]$ is subtracted from $P[j]$ to obtain $P'[j]$. Then $P'[i+b], \ldots, P'[i+\tau-1]$ are used to recover $V[i], \ldots, V[i+b-1]$ during the time slot $(i+\tau-1)$. Recovery is possible because (a) $P'[i+b], \ldots, P'[i+\tau-1]$ contain at least as many symbols as $V[i], \ldots, V[i+b-1]$ by definition, and (b) the linear equations used to define $P'[i+b], \ldots, P'[i+\tau-1]$ are chosen to be linearly independent. Second, during time slot $j \in \{i+\tau, \ldots, i+\tau+b-1\}$, $V[j-\tau], \ldots, V[j-1]$ are used to compute $P'[j]$. Subtracting $P'[j]$ from $P[j]$ yields $U[j-\tau]$.

Code construction (toy example). We now present a toy example of $(\tau = 4, b = 2)-$Variable-sized Generalized MS Code for frame-size sequence $k_0 = 3, k_1 = 2, k_2 = 1, k_3 = 2, k_4 = 1$, and $k_5 = \ldots = k_8 = 0$, shown in Figure 3.1. For $i \in [4]$, $S[i]$ is sent in $X[i]$. This satisfies the lossless-delay constraint. For $i \in \{0, 1, 4\}$, $U[i]$ is defined to equal $S[i]$, and $V[i]$ is defined to be empty (i.e., of size 0). For $i \in \{2, 3\}$, $V[i]$ is set as $S[i]$, and $U[i]$ is defined to be empty. Let $P'[4] = (S_0[2], S_0[3], S_1[3])$ and $P'[5] = (S_0[3], S_1[3])$. Next, $P[4] = (S[0] + P'[4])$

[2]For $i < \tau$, $P[i]$ is empty.

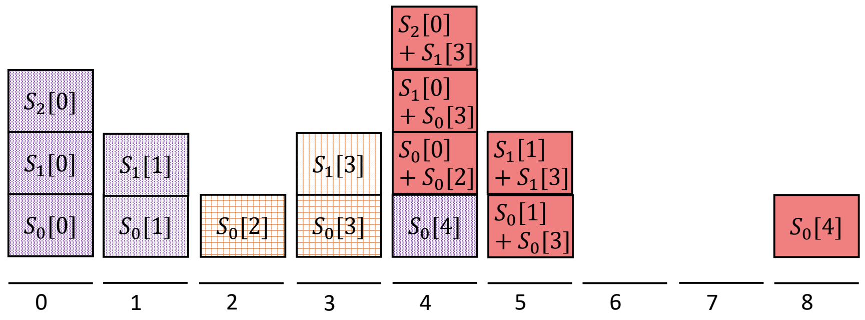


Figure 3.1: A toy example of the $(\tau = 4, b = 2)$-Variable-sized Generalized MS Code. Each frame, $S[i] = (U[i], V[i])$, is transmitted in the corresponding channel packet, $X[i]$, along with parity symbols, $P[i]$, (when applicable). White boxes with purple dots represent symbols of $U[i]$, white boxes with an orange grid represent symbols of $V[i]$, and solid red boxes represent symbols of $P[i]$. The numbers under the lines at the bottom indicate the time slots.

is transmitted in $X[4]$, and $P[5] = (S[1] + P'[5])$ is sent in $X[5]$. Finally, $P_0[8] = S_0[4]$ is transmitted in $X[8]$. The lossless-delay constraint is met, since each frame is sent within the corresponding channel packet. If any symbols of $V[2]$ and or $V[3]$ are lost, they are recovered using $P[4]$ and $P[5]$ respectively. Any lost symbols of $U[0], U[1]$, and $U[4]$ are each decoded with delay exactly 4 using $P[4], P[5]$, and $P[8]$ respectively (and subtracting $P'[4]$ and $P'[5]$ from $P[4]$ and $P[5]$ respectively). Therefore, the worst-case-delay constraint is satisfied.

Before presenting the detailed description, we introduce some notation. For any $i \leq j \in [t]$ and $Z \in \{S, X, U, V, P, P'\}$, $Z[i]$ is a vector of length $z[i]$, and $Z[i : j] = (Z[i], \dots, Z[j])$.

Code construction (detailed description). During each time slot i, the channel packet $X[i] = (S[i], P[i])$ is sent. The scheme is formally described in three parts: initialization, partitioning $S[i]$ into $(U[i], V[i])$, and defining $P[i]$.

Initialization: For $i \in [b-1]$, we set $U[i] = S[i]$ and $v[i] = 0$. For $i \in [\tau - 1]$ we set $p[i] = 0$. Let A be a $\tau m \times \tau m$ Cauchy matrix, where m is the maximum possible size of a frame.

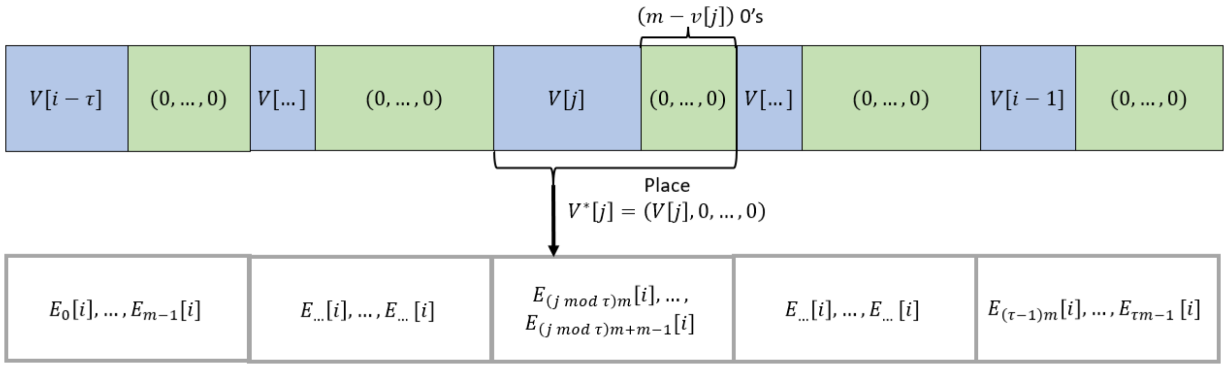
Partitioning $S[i]$: For any $i \geq b$, we partition $S[i]$ into $S[i] = (U[i], V[i])$ as follows.[3] We define an auxiliary variable z_i encapsulating the minimum number of parity symbols available for recovering $S[i]$ when $X[i]$ is dropped in a burst:

$$z_i = \min_{j \in \{i-b+1,\dots,i\}} \sum_{l=j+b}^{i+\tau-1} p[l] - \sum_{l=j}^{i-1} k_l. \tag{3.1}$$

The first $\min(k_i, z_i)$ symbols of $S[i]$ are set to $V[i]$:

$$V[i] = \left(S_0[i], \dots, S_{\min(k_i,z_i)-1}[i]\right) \tag{3.2}$$

[3]Recall that partitioning was defined for $i < b$ in initialization.



$(m - v[j])$ 0's
$V[i-\tau]$
$(0,\dots,0)$
$V[\dots]$
$(0,\dots,0)$
$V[j]$
$(0,\dots,0)$
$V[\dots]$
$(0,\dots,0)$
$V[i-1]$
$(0,\dots,0)$
Place
$V^*[j] = (V[j], 0, \dots, 0)$
$E_0[i], \dots, E_{m-1}[i]$
$E_{\dots}[i], \dots, E_{\dots}[i]$
$E_{(j \bmod \tau)m}[i], \dots, E_{(j \bmod \tau)m+m-1}[i]$
$E_{\dots}[i], \dots, E_{\dots}[i]$
$E_{(\tau-1)m}[i], \dots, E_{\tau m-1}[i]$

Figure 3.2: Illustration for defining $E[i]$, for time slot $i \in [t]$, by placing $V^*[j] = (V[j], 0, \dots, 0)$, for $j \in \{i - \tau, \dots, i - 1\}$, into m consecutive positions of $E[i]$ starting with position $(j \mod \tau)m$.

The remaining symbols of $S[i]$ are set to $U[i]$:

$$U[i] = \left(S_{\min(k_i,z_i)}[i], \dots, S_{k_i-1}[i]\right). \tag{3.3}$$

Finally,

$$p[i + \tau] = u[i] = k_i - \min(k_i, z_i) = k_i - v[i] \tag{3.4}$$

parity symbols are assigned to be sent in the channel packet $X[i+\tau]$, although the actual symbols of $P[i + \tau]$ have not yet been identified. The size of $p[i + \tau]$ is never greater than k_i (that is, the maximum possible size of $u[i]$), therefore $p[i + \tau]$ is at most m.

Defining $P[i]$*:* During time slot $(i \geq \tau)$, we set

$$P[i] = (U[i - \tau] + P'[i]) \tag{3.5}$$

where the symbols of $P'[i]$ are linear combinations of the symbols of $V[i - \tau], \dots, V[i - 1]$.[4] The linear combinations are chosen from a Cauchy matrix, as described below. Let $V^*[j]$ be the length m vector obtained by appending $(m-v[j])$ 0's to $V[j]$ for $j \in \{i-\tau, \dots, i-1\}$. We define a vector of length τm, $E[i]$, by placing $V^*[j]$, for $j \in \{i - \tau, \dots, i - 1\}$, into m consecutive positions of $E[i]$ starting with position $(j \mod \tau)m$, as is detailed in Figure 3.2. We use the Cauchy matrix A to define

$$P'[i] = E[i]A_{\{(i \mod \tau)m,\dots,(i \mod \tau)m+p[i]-1\}}. \tag{3.6}$$

The field size requirement is dictated by the Cauchy matrix and is at most $2\tau m$.

In Theorem 3 below, we verify that the Variable-sized Generalized MS Code meets the requirements of the model.

Theorem 3. *For any parameters (τ, b) and frame-size sequence $k_0, \dots, k_t$, the (τ, b)-Variable-sized Generalized MS Code satisfies the lossless-delay and worst-case-delay constraints over any $C(b, \tau)$ channel.*

[4]Recall that $p[i]$ was defined during initialization for $i < \tau$.

Proof. The lossless-delay constraint is satisfied for $i \in [t]$ by sending $X[i] = (S[i], P[i])$.

We prove that the worst-case-delay constraint is satisfied by showing for any $i \in [t-\tau]$ that each of $S[i], \ldots, S[i+b-1]$ are recovered within delay τ when $X[i], \ldots, X[i+b-1]$ are lost.[5] First, we show that $V[i], \ldots, V[i+b-1]$ are recovered by time slot $(i+\tau-1)$. Second, we show that $U[i], \ldots, U[i+b-1]$ are recovered by time slots $(i+\tau), \ldots, (i+\tau+b-1)$, respectively.

First, for $j \in \{i+b, \ldots, i+\tau-1\}$ subtracting $U[j-\tau]$ from $P[j]$ yields $P'[j]$ (by Equation 3.5). Combining Equations 3.2, 3.3, 3.4, and 3.5 shows that the total number of symbols in $P'[i+b], \ldots, P'[i+\tau-1]$ is at least as many as $V[i], \ldots, V[i+b-1]$:

$$\sum_{j=i+b}^{i+\tau+b-1} p'[j] \geq \sum_{j=i}^{i+b-1} k_j$$

$$\left(\sum_{j=i+b}^{i+\tau-1} p'[j]\right) + \left(\sum_{j=i+\tau}^{i+\tau+b-1} p'[j]\right) \geq \left(\sum_{j=i}^{i+b-1} v[j]\right) + \left(\sum_{j=i}^{i+b-1} u[j]\right)$$

$$\sum_{j=i+b}^{i+\tau-1} p'[j] \geq \sum_{j=i}^{i+b-1} v[j].$$

Next, we show that $P'[i+b], \ldots, P'[i+\tau-1]$ suffices to decode $V[i], \ldots, V[i+b-1]$. For $j \in \{i+b, \ldots, i+\tau-1\}$, recall from Equation 3.6 and Figure 3.2 that $P'[j]$ is the product of distinct columns of A with a vector consisting of (a) for $l \in \{i, \ldots, i+b-1\}$, $V[l]$ in positions $(j \mod \tau)m, \ldots, ((j \mod \tau)m + v[l] - 1)$, (b) for $l \in \{i, \ldots, i+b-1\}$, zeros in positions $((j \mod \tau)m + v[l]), \ldots, ((j \mod \tau+1)m - 1)$, and (c) a combination of symbols of $V[j-\tau], \ldots, V[i-1], V[i+b], \ldots, V[j-1]$ and zero padding in the remaining positions. For $l \in \{i+b, \ldots, i+\tau-1\}$, let $E'[l]$ be defined by first setting it equal to $E[l]$ and second replacing the symbols corresponding to $V[i], \ldots, V[i+b-1]$ with 0's. We note that the receiver can compute $E'[i+b], \ldots, E'[i+\tau-1]$ during time slot $(i+\tau-1)$. Let $P^*[i+b], \ldots, P^*[i+\tau-1]$ correspond to $(P'[i+b] - E'[i+b]A), \ldots, (P'[i+\tau-1] - E'[i+\tau-1]A)$. Then for some $l_0, \ldots, l_{b-1}$ which is a permutation of $i, \ldots, (i+b-1)$,

$$\begin{bmatrix} P^*[i+b]^T \\ \vdots \\ P^*[i+\tau-1]^T \end{bmatrix} = \begin{bmatrix} V[l_0]^T \\ \vdots \\ V[l_{b-1}]^T \end{bmatrix}^T A'$$

where T denotes transpose, and A' is a submatrix of A with $\left(\sum_{j=i}^{i+b-1} v[j]\right)$ rows and at least $\left(\sum_{j=i}^{i+b-1} v[j]\right)$ columns. As such, A' is Cauchy and thus has full rank. Hence, $P'[i+b], \ldots, P'[i+\tau-1]$ suffices to decode $V[i], \ldots, V[i+b-1]$.

Second, for $j \in \{i, \ldots, i+b-1\}$, $V[j], \ldots, V[j+\tau-1]$ are used to compute

$$P'[j+\tau] = E[j+\tau]A_{\{(j \mod \tau)m, \ldots, (j \mod \tau)m + p[j+\tau]-1\}}.$$

[5]Each frame $S[i]$ for $i > (t-\tau)$ is of size 0 and is known by the receiver due to the termination of the frame-size sequence.

During time slot $(j+\tau)$, $U[j] = (P[j+\tau] - P'[j+\tau])$ is then decoded.[6]

$\square$

The following lemma essentially shows that all parity symbols sent in any channel packet under the $(\tau, b)-$Variable-sized Generalized MS Code are needed to satisfy the worst-case-delay constraint. This property is later used to prove that the $(\tau, b)-$Variable-sized Generalized MS Code is rate-optimal in Theorem 4.

Lemma 8. *Consider any parameters (τ, b) and frame-size sequence $k_0, \ldots, k_t$. Under the (τ, b)-Variable-sized Generalized MS Code, for all $i \geq \tau$ where $p[i] > 0$, $\exists j \in \{i-\tau-b+1, \ldots, i-\tau\}$ such that $\sum_{l=j}^{i-\tau} k_l = \sum_{l=j+b}^{i} p[l]$.*

Proof. For $i \in \{\tau, \ldots, \tau+b-1\}$, consider $j = 0$. Then

$$\sum_{l=j}^{i-\tau} k_l = \sum_{l=0}^{i-\tau} u[l] = \sum_{l=\tau}^{i} p[l] = \sum_{l=b}^{i} p[l]$$

due to Equation 3.4 as well as the initialization defining (a) $p[0], \ldots, p[\tau-1]$ to each be 0, and (b) $u[0], \ldots, u[b-1]$ to be $k_0, \ldots, k_{b-1}$ respectively.

For $(i \geq \tau + b)$, if $(p[i] = u[i-\tau] > 0)$ then $(v[i-\tau] < k_{i-\tau})$. By Equations 3.1 and 3.2 and the fact that $(v[i-\tau] < k_{i-\tau})$ there is some $j \in \{i-\tau-b+1, \ldots, i-\tau\}$ for which for $i' = (i-\tau)$

$$\begin{aligned} v[i'] &= \sum_{l=j+b}^{i'+\tau-1} p[l] - \sum_{l=j}^{i'-1} k_l \\ v[i-\tau] &= \sum_{l=j+b}^{i-1} p[l] - \sum_{l=j}^{i-\tau-1} k_l \\ v[i-\tau] + u[i-\tau] + \sum_{l=j}^{i-\tau-1} k_l &= p[i] + \sum_{l=j+b}^{i-1} p[l] \\ \sum_{l=j}^{i-\tau} k_l &= \sum_{l=j+b}^{i} p[l]. \end{aligned}$$

$\square$

Next, we present Theorem 4, which shows that the (τ, b)-Variable-sized Generalized MS Code is rate-optimal for Regime 1.

The proof involves an inductive argument on the time slot. It will show that the cumulative number of symbols sent by each time slot under any code construction, even an offline one, must be at least as many as under the (τ, b)-Variable-sized Generalized MS Code to satisfy the lossless-delay and worst-case-delay constraints. The proof technique synergizes with the greedy paradigm of the (τ, b)-Variable-sized Generalized MS Code sending for each frame $S[i]$: (a) the

[6]In the edge case where $i > (t-\tau)$, $S[i]$ is known by the decoder to have size 0 and this step is not needed.

minimal number of parity symbols needed to recover $S[i]$ given any burst assuming that no future frames needs to be recovered, and (b) deferring the transmission of the parity symbols until the decoding deadline for $S[i]$ (i.e., $X[i+\tau]$). The methodology for designing a streaming code using a greedy paradigm and inductively proving that it is rate-optimal form a suitable template for designing new online coding schemes in other regimes, as is used later in Chapter 4 to design approximately rate-optimal streaming codes for Regime 3 (i.e., $\tau_L = 1$).

Theorem 4. *For any parameters $(\tau, b, \tau_L = 0)$, the (τ, b)-Variable-sized Generalized MS Codeis rate-optimal for transmission over a $C(b, \tau)$ channel.*

Proof sketch. We present the full proof in Appendix 3.7.1.

For an arbitrary frame-size sequence $k_0, k_1, \ldots, k_t$, consider any optimal offline construction O. We prove by induction on time slot $i = 0, 1, 2, \ldots, t$ that the cumulative number of symbols sent by O is at least as many as that of the (τ, b)-Variable-sized Generalized MS Code.

In the base case, for each $i \in [\tau - 1]$, the channel packet $X[i]$ under O must contain at least k_i symbols to meet the lossless-delay constraint for frame $S[i]$. Under the (τ, b)-Variable-sized Generalized MS Code, $x[i] = k_i$.

The inductive step for $i \in \{\tau, \ldots, t\}$ has two cases.

First, when no parity symbols are sent in $X[i]$ (that is, $X[i] = S[i]$) under the (τ, b)-Variable-sized Generalized MS Code, at least $s[i] = k_i$ symbols are sent in $X[i]$ under O to meet the lossless-delay constraint.

Second, suppose that $X[i] = (S[i], P[i])$ is sent under the (τ, b)-Variable-sized Generalized MS Code where $p[i] > 0$. Applying Lemma 8 shows that there is a burst loss starting at time slot $j \in \{i - \tau - b + 1, \ldots, i - \tau\}$ where the number of parity symbols received under the (τ, b)-Variable-sized Generalized MS Code in $X[j+b], \ldots, X[i]$ is the smallest for which it is possible to decode frame $S[j], \ldots, S[i-\tau]$. We combine this fact with the lossless-delay constraint for $S[j+b], \ldots, S[i]$. We then show that at least as many symbols are sent under O between time slots $(j+b)$ and i as are, respectively, sent under the (τ, b)-Variable-sized Generalized MS Code. Applying the inductive hypothesis for time slot $(j + b - 1)$ concludes the proof.

$\square$

We note that for any values of τ and b, the (τ, b)-Variable-sized Generalized MS Code's rate (i.e., the optimal rate) is highly dependent on the precise sequence of the sizes of the frames. Hence, a closed-form expression is not viable.

In this section, we presented a rate-optimal online streaming codes for Regime 1. We had previously shown a rate-optimal online streaming code for Regime 2 in Section 2.3.2. We showed in the proof of Theorem 4 that, for any (τ, b), the $(\tau, b)-$Variable-sized Generalized MS Code matches the rate of the best offline construction possible for Regime 1. The simple construction for Regime 2 matches the upper bound of the rate of $\frac{\tau}{\tau+b}$. Both of these constructions match the best possible rates of the offline setting, establishing that the online-optimal-rate equals the offline-optimal-rate in both parameter regimes. The construction for Regime 1 can be used for *any* value of τ_L, although it is not necessarily rate-optimal for $\tau_L > 0$. Next, in Section 3.3, we show that online codes cannot match the offline-optimal-rate for all other parameter settings.

3.3 Infeasiblity of offline-optimal-rate for Online Schemes

In Sections 2.3.2 and 3.2, we presented online code constructions that matched the offline-optimal-rate under the two broad settings of Regime 1 and Regime 2. A natural question is whether there are any other parameter settings where an online coding scheme can attain the offline-optimal-rate. In this section, we show that the online-optimal-rate is strictly less than the offline-optimal-rate for all other parameter settings.

At a high level, the optimal approach to spreading symbols from a frame $S[i]$ over channel packets $X[i], \ldots, X[i+\tau_L]$ depends on the sizes of future frames (i.e., $k_{i+1}, \ldots, k_t$). This dependency enables offline coding schemes to have higher rates than online coding schemes in all settings besides Regime 1 and Regime 2, as we will show in Theorem 5.

Theorem 5. *For any parameters (τ, b, τ_L) outside of Regime 1 and Regime 2, the online-optimal-rate is strictly less than offline-optimal-rate.*

Proof sketch. The proof consists of three mutually exclusive cases shown via illustrative examples in Sections 3.4, 3.5, and 3.6 and in detail in Appendix 3.7.2, 3.7.3, and 3.7.4. In each case, we present two distinct frame-size sequences of length $(t+1)$, which match for the first several time slots. We show a lower bound on the offline-optimal-rate for the two frame-size sequences by presenting an offline coding scheme with rates $R_t^{(1)}$ and $R_t^{(2)}$ on the first and second frame-size sequences, respectively. To attain a rate of at least $R_t^{(1)}$ on the first frame-size sequence requires sending symbols in a manner that leads to a lower rate than $R_t^{(2)}$ on the second. $\square$

Remark 8. *Although Theorem 5 is proven for two specific frame-size sequences, a similar proof holds if the sizes of the frames were only approximately the sizes corresponding to the frame-size sequences. As such, the result establishes a broad class of frame-size sequences for which there is a gap between the online-optimal-rate and the offline-optimal-rate.*

3.4 Case $\tau_L \geq b$ and $\tau_L = (\tau - b)$

This section presents the proof for parameters $(b, \tau_L, \tau) = (3, 4, 7)$; the general case, which builds closely on this example, is proven in Appendix 3.7.2.

Consider the following two frame-size sequences:

1. $k_0^{(1)} = 2$ and $k_j^{(1)} = 0$ for $j > 0$.
2. $k_0^{(2)} = 2, k_1^{(2)} = 2, k_2^{(2)} = 10$, and $k_j^{(1)} = 0$ for $j > 2$.

An offline construction for the two frame-size sequences is shown in Figures 3.3 and 3.4 respectively, over $\mathbb{F}_q$ for any prime $q \geq 83$.

For frame-size sequence 1, $X[0] = S_0[0]$, $X[3] = S_1[0]$, and $X[6] = (S_0[0] + S_1[0])$, as shown in Figure 3.3. The lossless-delay constraint is trivially satisfied. The worst-case-delay constraint is met, as at most one of $X[0], X[3]$, and $X[6]$ is lost.

For frame-size sequence 2, $X[0] = S[0]$ and $X[1] = S[1]$. In addition, for $i \in \{2, \ldots, 6\}$ $X[i] = \big(S_{2(i-2)}[2], S_{2(i-2)+1}[2]\big)$, $X[7] = \big(S[0] + \sum_{i=3}^{6} X[i]\big)$,$X[8] = \big(S[1] + \sum_{i=4}^{6} 2^{i-2} X[i]\big)$, and $X[9] = \sum_{i=2}^{6} 3^{i-2} X[i]$, as shown in Figure 3.4. The lossless-delay is clearly satisfied. The worst-case-delay constraint is met, as will be shown next through a comprehensive case analysis.

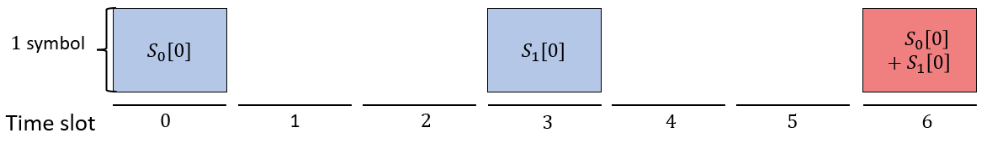


Figure 3.3: Offline construction for frame-size sequence 1 for parameters $(b, \tau_L, \tau) = (3, 4, 7)$.

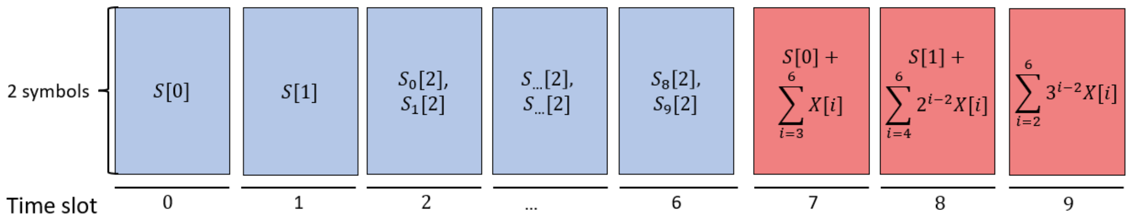

2 symbols
$S[0]$
$S[1]$
$S_0[2]$, $S_1[2]$
$S_{\dots}[2]$, $S_{\dots}[2]$
$S_8[2]$, $S_9[2]$
$S[0] + \sum_{i=3}^{6} X[i]$
$S[1] + \sum_{i=4}^{6} 2^{i-2}X[i]$
$\sum_{i=2}^{6} 3^{i-2}X[i]$
Time slot
0
1
2
...
6
7
8
9

Figure 3.4: Offline construction for frame-size sequence 2 for parameters $(b, \tau_L, \tau) = (3, 4, 7)$.

For any $l \in \{0, 1\}$ suppose that $X[l]$ is lost. Then $S[l] = \left(X[7+l] - \sum_{j=3+l}^{6}(l+1)^{j-2}X[j]\right)$ is obtained within 7 time slots. When $X[2]$ is lost, $S[0]$ and $S[1]$ are decoded. Then one can decode

$$(S_4[2], S_5[2]) = 2^{-2}\left(X[8] - S[1] - 2^3X[5] - 2^4X[6]\right)$$
$$(S_2[2], S_3[2]) = (X[7] - S[0] - X[4] - X[5] - X[6])$$
$$(S_0[2], S_1[2]) = \left(X[9] - \sum_{j=3}^{6} 3^{j-2}X[j]\right).$$

When a burst starts with $X[3]$, $S[0]$, $S[1]$, and $S_0[2]$ are decoded, and $(S_8[2], S_9[2])$ is received. Combining $S[0], S[1]$, and $X[2]$, with $X[6:9]$ yields $\sum_{l=3}^{6} X[i], \sum_{l=4}^{5} 2^{l-2}X[l]$, as well as $\sum_{l=3}^{5} 3^{l-2}X[l]$. These three equations are linearly independent and yield $X[3:5]$. Thus, $S[2]$ is decoded by time slot 9. When a burst starts with $X[4]$, $S[0], S[1], S_0[2], S_1[2], S_2[2]$, and $S_3[2]$ are received and combined with $X[7], X[8]$, and $X[9]$ to determine $\sum_{j=4}^{6} X[j], \sum_{j=4}^{6} 2^{j-2}X[j]$, and $\sum_{j=4}^{6} 3^{j-2}X[j]$. These three equations are linearly independent and yield $X[4:6]$, which consist of $S_4[2], \dots, S_9[2]$. When a burst starts with $X[5]$, $S[1], S_0[2], \dots, S_5[2]$ are received and combined with $X[8]$ and $X[9]$ to determine $\sum_{j=5}^{6} 2^{j-2}X[j]$, and $\sum_{j=5}^{6} 3^{j-2}X[j]$. These two equations are linearly independent and yield $X[5]$ and $X[6]$, which include $S_6[2], \dots, S_9[2]$. When a burst starts with $X[6]$, $S_0[2], \dots, S_7[2]$ are received, so $(S_8[2], S_9[2]) = 3^{-4}\left(\sum_{j=2}^{5} 3^{j-2}X[j]\right)$. When $X[0:6]$ are received, the frames are received.

The rate of the offline construction for frame-size sequence 1 is $2/3$, while its rate for frame-size sequence 2 is 0.7. An online construction must send at most 1 symbol in $X[0]$ to have a rate of $2/3$ on frame-size sequence 1 because $X[0]$ can be lost. We next show that any such scheme cannot attain the rate of 0.7 on frame-size sequence 2. If frame-size sequence 2 occurs, the online construction must send at least 13 symbols over $X[1:6]$ due to the lossless-delay constraint. At least one of $X[1:3]$ and $X[4:6]$ must contain at least 7 symbols and may be lost. At least 14 symbols must be received. So the rate is at most $14/21$ (i.e., less than 0.7). Therefore, any online construction with a rate of $2/3$ on frame-size sequence 1 cannot attain the rate of 0.7 on frame-size sequence 2, unlike the proposed offline construction.

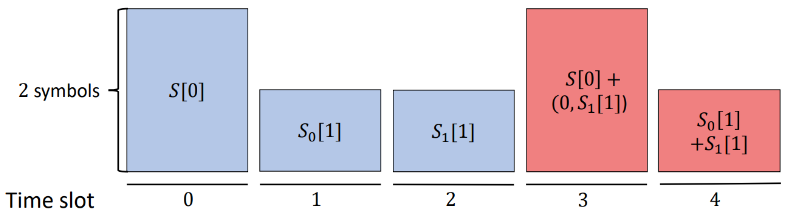


Figure 3.5: Offline construction for frame-size sequence 1 for parameters $(b, \tau_L, \tau) = (2, 1, 3)$.

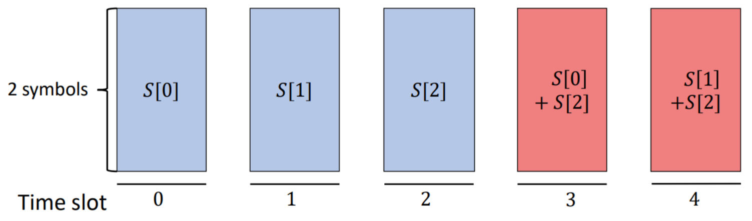


Figure 3.6: Offline construction for frame-size sequence 2 for parameters $(b, \tau_L, \tau) = (2, 1, 3)$.

3.5 Case $\tau_L < b$ and $\tau_L = (\tau - b)$

This section presents the proof for parameters $(b, \tau_L, \tau) = (2, 1, 3)$; the general case, which builds closely on this example, is proven in Appendix 3.7.3.

Consider the following two frame-size sequences:

1. $k_0^{(1)} = 2$, $k_1^{(1)} = 2$, and $k_j^{(1)} = 0$ for $j > 1$.
2. $k_0^{(1)} = 2$, $k_1^{(1)} = 2$, $k_2^{(1)} = 2$, and $k_j^{(1)} = 0$ for $j > 2$.

An offline construction for the two frame-size sequences is shown in Figures 3.5 and 3.6 respectively over any finite field, $\mathbb{F}_q$.

Figure 3.5 shows the channel packets for frame-size sequence 1; specifically, $X[0] = S[0]$, $X[1] = S_0[1]$, $X[2] = S_1[1]$, $X[3] = (S[0] + (0, S_1[1]))$, and $X[4] = (S_0[1] + S_1[1])$. The lossless-delay constraint is trivially satisfied. The worst-case-delay constraint is met for $S[0]$ because either $S[0]$ is received, or $S_1[1]$ and $X[3]$ are received, yielding $S[0]$. When $X[1]$ is lost, $(0, S_1[1]) = (X[3] - S[0])$ is obtained, leading to $S_0[1] = (X[4] - S_1[1])$. When $X[2]$ is lost, $S_0[1]$ is decoded, leading to $S_1[1] = (X[4] - S_0[1])$. As such, the worst-case-delay is satisfied for $S[1]$.

For frame-size sequence 2, $X[0] = S[0], X[1] = S[1]$, $X[2] = S[2]$, $X[3] = (S[0] + S[2])$, and $X[4] = (S[1] + S[2])$, as shown in Figure 3.6. The lossless-delay is clearly satisfied. The worst-case-delay constraint is met for $S[0]$ as either $X[0] = S[0]$ is received, or $S[0] = (X[3] - X[2])$ is obtained. The worst-case-delay constraint is satisfied for $S[1]$ since either $X[1]$ is received, or $S[0]$ is decoded, leading to $S[2] = (X[3] - S[0])$, and $S[1] = (X[4] - S[2])$. The worst-case-delay constraint is satisfied for $S[2]$ because either $X[2]$ is received, or $S[1]$ is decoded, yielding $S[2] = (X[4] - S[1])$.

The offline construction's rate for frame-size sequence 1 is $4/7$, while its rate for frame-size sequence 2 is 0.6. An online construction with a rate of $4/7$ on frame-size sequence 1 must send at most 3 symbols in $X[0 : 1]$, since at least 4 symbols are sent in $X[2 : 4]$ in case $X[0 : 1]$ is lost. Also, the construction sends at least 2 symbols over $X[0:1]$ to recover $S[0]$ under lossless transmission. Next, we show that any such scheme cannot attain the rate of 0.6 on frame-size sequence 2 due to sending fewer than 4 symbols over $X[0:2]$. Thus, any online construction with a rate of $4/7$ on frame-size sequence 1 cannot attain the rate of 0.6 on frame-size sequence 2, unlike the proposed offline construction.

First, suppose that exactly 2 symbols are sent in $X[0:1]$. Then $X[2:3]$ suffices to recover $S[0]$. Recall that the 2 symbols in $X[0:1]$ only contain information about $S[0]$, as they suffice to recover $S[0]$ under a lossless transmission. Thus, $X[0:1]$ are recovered as a function of $S[0]$, leaving the transmission lossless, so $S[1:2]$ are recovered. Thus, $X[2:3]$ contains at least 6 symbols. At least 6 symbols are sent outside of $X[2:3]$ in case $X[2:3]$ is lost, so the rate is at most $6/12$.

Second, due to the upper bound on the rate of $\frac{\tau}{\tau+b}=\frac{3}{5}$ and worst-case-delay, at least $10=6*\frac{5}{3}$ symbols must be sent by time slot 5. Suppose exactly 3 symbols are sent in $X[0:1]$. Consider the 5 periodic erasure channels, $C_0,\ldots,C_4$, where for $i\in[4]$, C_i drops packets $X[j:j+1]$ for all $j\equiv i \mod 5$. Each packet is dropped by 2 of these channels, so the channels drop at least $\frac{2}{5}*10\geq 4$ symbols on average. At least 6 symbols must be received to ensure recovery. If any channel dropped 5 or more symbols, the rate would be at most $6/11$. Thus, each channel must drop exactly 4 symbols to attain a rate of 0.6. Therefore, C_0 drops exactly 4 symbols—3 over $X[0:1]$ and 1 in $X[5]$. Each of C_4, C_3, and C_2 must drop 4 symbols (i.e., $n_4+n_5=4, n_3+n_4=4, n_2+n_3=4$). Hence, $X[4]$ contains 3 symbols, $X[3]$ contains 1 symbol, and $X[2]$ contains 3 symbols. In total, $(3+3+1+3+1)=11$ symbols are sent over $X[0:1], X[2], X[3], X[4]$, and $X[5]$, leading to a rate of $6/11$, which is less than 0.6.

Therefore, any online construction that matches the rate of $4/7$ on frame-size sequence 1 cannot attain the rate of 0.6 on frame-size sequence 2, unlike the offline construction.

3.6 Case $\tau_L < (\tau - b)$

This section presents the proof for parameter $(b,\tau_L,\tau)=(1,1,3)$; the general case, which builds closely on this example, is proven in Appendix 3.7.4.

Consider the following two frame-size sequences:

1. $k_0^{(1)}=2$ and $k_j^{(1)}=0$ for $j>0$.
2. $k_0^{(1)}=2$, $k_1^{(1)}=4$, and $k_j^{(1)}=0$ for $j>1$.

An offline construction for the two frame-size sequences is shown in Figures 3.7 and 3.8 respectively over any finite field, $\mathbb{F}_q$.

For frame-size sequence 1, $X[0]=S_0[0]$, $X[1]=S_1[0]$, and $X[2]=(S_0[0]+S_1[0])$, as is shown in Figure 3.7. The lossless-delay constraint is trivially satisfied. The worst-case-delay constraint is met because at most one of $X[0], X[1]$, or $X[2]$ is lost and $X[2]=(X[0]+X[1])$.

For frame-size sequence 2, $X[0]=S[0], X[1]=(S_0[1],S_1[1])$, $X[2]=(S_2[1],S_3[1])$, and $X[3]=(S[0]+(S_0[1],S_1[1])+(S_2[1],S_3[1]))$, as shown in Figure 3.8. The lossless-delay is clearly satisfied. The worst-case-delay constraint is met, since at most one of $X[0], X[1], X[2]$, or $X[3]=\sum_{i=0}^{2}X[i]$ is lost.

The offline construction's rate for frame-size sequence 1 is $2/3$, while its rate for frame-size

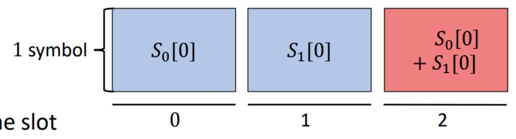


Figure 3.7: Offline construction for frame-size sequence 1 for parameters $(b, \tau_L, \tau) = (1, 1, 3)$.

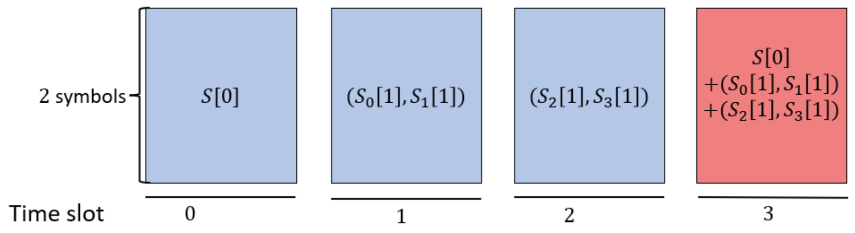


Figure 3.8: Offline construction for frame-size sequence 2 for parameters $(b, \tau_L, \tau) = (1, 1, 3)$.

sequence 2 is 0.75. For an online construction to attain a rate of $2/3$ on frame-size sequence 1, it must send exactly 1 symbol in each of $X[0]$ and $X[1]$ due to (a) the lossless-delay constraint and (b) ensuring at most 1 symbol is lost—a necessity to attain the rate of $2/3$. Next, we show that any such scheme cannot attain the rate of 0.75 on frame-size sequence 2. If frame-size sequence 2 occurs, at least 6 symbols are sent over $X[0:2]$ due to the lossless-delay constraint. The average number of symbols per packet is at least 2. If $X[0]$ contains one symbol, at least one of $X[1]$ or $X[2]$ contains at least 3 symbols. At least 6 symbols must be received to satisfy the worst-case-delay constraint. Since at least 3 symbols may be lost, at least 9 symbols must be sent in total. As such, the rate is at most $2/3$, which is less than 0.75. Therefore, any online construction that matches the rate of $2/3$ on frame-size sequence 1 cannot attain the rate of 0.75 on frame-size sequence 2, unlike the offline construction.

3.7 Appendix

3.7.1 Proof of Theorem 4

In this section, we will prove Theorem 4. At a high level, the proof is inductive and shows that the cumulative number of symbols sent by each time slot under the (τ, b)-Variable-sized Generalized MS Code is the minimum possible. For time slots where no parity symbols are sent, it follows immediately by the lossless-delay constraint. Otherwise, there is some burst for which every parity symbol in the received channel packets is needed to recover the burst within the worst-case-delay.

We begin by introducing the preliminary notation for the proof. We then include a few auxiliary Lemmas used throughout the proof. Finally, we present the full proof itself.

Let t be an arbitrary natural number, and consider any length $(t+1)$ frame-size sequence $k_0, \ldots, k_t$. Let O be an arbitrary offline code construction that satisfies the lossless-delay and worst-case-delay constraints over a $C(b, \tau)$ channel for the frame-size sequence. Let the channel packet transmitted during time slot $j \in [t]$ under construction O and under the (τ, b)-Variable-sized Generalized MS Code be labeled as $X_O[j]$ and $X_V[j]$, respectively. Let the cumulative number of symbols transmitted through time slot j under construction O and under the (τ, b)-Variable-sized Generalized MS Code be denoted $n^+_{O,j} = \sum_{i=0}^{j} x_O[i]$ and $n^+_{V,j} = \sum_{i=0}^{j} x_V[i]$, respectively. Recall that each frame comprises symbols drawn independently and uniformly at random from the finite field $\mathbb{F}_q$. Let $\mathcal{S}$ be a random variable representing a uniformly random element of $\mathbb{F}_q$.

Next, we show that the lossless-delay constraint necessitates transmitting at least as many symbols as the size of the frame for each time slot.

Lemma 9. *Consider any parameters $(\tau, b, \tau_L = 0)$, any frame-size sequence $k_0, k_1, \ldots, k_t$, and any code construction which satisfies the lossless-delay and worst-case-delay constraints over a $C(b, \tau)$ channel. For any $j \in [t]$, $n_j \geq k_j$.*

Proof. Follows directly from (a) the independence of frames, and (b) the lossless-delay constraint for $\tau_L = 0$. □

Next, we establish that whenever a burst of length b occurs, all frames from time slots before the burst must be decoded before the burst to satisfy both the lossless-delay and worst-case-delay constraints.

Lemma 10. *Consider any parameters $(\tau, b, \tau_L = 0)$, any frame-size sequence $k_0, k_1, \ldots, k_t$, $j \in [t]$, and any code construction which satisfies the lossless-delay and worst-case-delay constraints over a $C(b, \tau)$ channel. When $X[j], \ldots, X[j+b-1]$ are lost in a burst, $S[0 : j-1]$ are decoded by time slot $(j-1)$.*

Proof. By the worst-case-delay constraint, $S[0 : j-\tau-1]$ are all decoded by time slot $(j-1)$. Under the $C(b, \tau)$ channel, when $X[j], \ldots, X[j+b-1]$ are lost, $X[j-\tau], \ldots, X[j-1]$ are necessarily received.[7] By the lossless-delay constraint, $S[0 : j-\tau-1]$ and $X[j-\tau : j-1]$ suffice to decode $S[j-\tau : j-1]$. □

Finally, we prove Theorem 4 below.

Proof of Theorem 4. Let $k_0, k_1, \ldots, k_t$ be an arbitrary frame-size sequence. We will show by induction that the cumulative number of symbols sent through time slot $i \in [t]$ under an arbitrary offline construction, O, is at least as many as that of the $(\tau, b)-$Variable-sized Generalized MS Code (i.e., $n^+_{O,i} \geq n^+_{V,i}$). Consequently, the (τ, b)-Variable-sized Generalized MS Code matches the offline-optimal-rate.

In the base case, we consider $j \in [\tau - 1]$. Applying Lemma 9 determines that $x_O[j] \geq k_j = x_V[j] \; \forall j \in [\tau - 1]$.

For the inductive hypothesis, we assume that for some $(i_* \geq \tau - 1)$, for all $l \in [i_*]$, $n^+_{O,l} \geq n^+_{V,l}$.

For the inductive step, consider the time slot $(i = i_* + 1 \geq \tau)$. By the inductive hypothesis, $n^+_{O,i-1} \geq n^+_{V,i-1}$. We will show that $n^+_{O,i} \geq n^+_{V,i}$ using two cases.

Case $x_V[i] = k_i$:

Applying Lemma 9 determines that $x_O[i] \geq k_i$. Therefore, $(n^+_{O,i} = n^+_{O,i-1} + k_i \geq n^+_{V,i-1} + k_i = n^+_{V,i})$.

[7]When $j < \tau$, $X[0 : j-1]$ are received.

Case $x_V[i] > k_i$: We first provide a high-level intuition of the proof and then the detailed derivation.

High-level summary: Applying Lemma 8 shows that there is a burst starting in time slot $j \in \{i-\tau-b+1,\dots,i-\tau\}$ for which the (τ,b)-Variable-sized Generalized MS Code receives minimum required number of parity symbols to decode frames $S[j:i-\tau]$ by time slot i. Combining this fact with meeting the lossless-delay constraint for $S[j+b:i]$ shows that the number of symbols sent under O between time slots $(j+b)$ and i is at least as many as that of the (τ,b)-Variable-sized Generalized MS Code.

Detailed derivation: By Lemma 8, there is some $j \in \{i-\tau-b+1,\dots,i-\tau\}$ such that $\sum_{l=j+b}^{i} p[l] = \sum_{l=j}^{i-\tau} k_l$. Therefore,

$$\sum_{l=j+b}^{i} x_V[l] = \sum_{l=j}^{i-\tau} k_l + \sum_{l=j+b}^{i} k_l. \tag{3.7}$$

Next, we show that at least as many symbols are sent over $X_O[j+b:i]$ as are sent over $X_V[j+b:i]$. Consider a burst loss of $X[j],\dots,X[j+b-1]$. Applying Lemma 10 shows that $S[0:j-1]$ are known by the receiver by time slot $(j-1)$. By the worst-case-delay constraint,

$$H\left(S[j:i-\tau]\middle|X_O[j+b:i],S[0:j-1]\right)=0. \tag{3.8}$$

We next bound the number of symbols sent over $X_O[j+b:i]$ as

$$\begin{aligned}
&H\left(S[j:i-\tau]\right)+H\left(X_O[j+b:i]\middle|S[0:i-\tau]\right)= && (3.9)\\
&H\left(X_O[j+b:i],S[j:i-\tau]\middle|S[0:j-1]\right)= && (3.10)\\
&H\left(X_O[j+b:i]\middle|S[0:j-1]\right)+ && (3.11)\\
&H\left(S[j:i-\tau]\middle|S[0:j-1],X_O[j+b:i]\right)= && \\
&H\left(X_O[j+b:i]\middle|S[0:j-1]\right), && (3.12)
\end{aligned}$$

where Equation 3.10 follows from the chain rule and independence of frames, Equation 3.11 follows from the chain rule, and Equation 3.12 follows from Equation 3.8.

Combining Equations 3.10 and 3.12 with the fact that conditioning reduces entropy yields

$$\begin{aligned}
&H\left(X_O[j+b:i]\right)\geq H\left(X_O[j+b:i]\middle|S[0:j-1]\right)\geq \\
&H\left(S[j:i-\tau]\right)+H\left(X_O[j+b:i]\middle|S[0:j+b-1]\right).
\end{aligned} \tag{3.13}$$

Next, we evaluate the size of $H\left(X_O[j+b:i]\middle|S[0:j+b-1]\right)$ as

$$\begin{aligned}
&H\left(S[j+b:i],X_O[j+b:i]\middle|S[0:j+b-1]\right)= && (3.14)\\
&H\left(S[j+b:i]\right)+H\left(X_O[j+b:i]\middle|S[0:i]\right)= && (3.15)\\
&H\left(S[j+b:i]\right)= && (3.16)\\
&H\left(X_O[j+b:i]\middle|S[0:j+b-1]\right)+ && (3.17)\\
&H\left(S[j+b:i]\middle|S[0:j+b-1],X_O[j+b:i]\right)= && \\
&H\left(X_O[j+b:i]\middle|S[0:j+b-1]\right), && (3.18)
\end{aligned}$$

where Equation 3.15 follows from conditioning and independence of frames, Equation 3.16 follows from the fact that for $l \in [t]$, $X_O[l]$ is a function of $S[0 : l]$, Equation 3.17 follows from conditioning, and Equation 3.18 follows from the lossless-delay constraint.

For any $i \in [t]$,

$$H(S[i]) = H(\mathcal{S})k_i \tag{3.19}$$

$$H(X[i]) \leq H(\mathcal{S})n_i \tag{3.20}$$

where $\mathcal{S}$ was defined as a random variable drawn uniformly at random from the underlying field, $\mathbb{F}_q$. This follows from the definition of frames, and the fact that the maximum possible entropy of n_i symbols is $n_i H(\mathcal{S})$. Applying Equation 3.18 and 3.16 to Equations 3.13, 3.19, and 3.20 yields

$$\begin{aligned} &H(\mathcal{S}) \sum_{l=j+b}^{i} \left|X_O[l]\right| \geq H\left(X_O[j+b : i]\right) \geq \\ &H\left(S[j : i-\tau]\right) + H\left(S[j+b : i]\right) = \\ &H(\mathcal{S})\big(\sum_{l=j}^{i-\tau} k_l + \sum_{l=j+b}^{i} k_l\big). \end{aligned} \tag{3.21}$$

Combining Equations 3.21 and 3.7 determines that

$$H\left(\mathcal{S}\right) \sum_{l=j+b}^{i} x_O[l] \geq H\left(\mathcal{S}\right) \sum_{l=j+b}^{i} x_V[l]. \tag{3.22}$$

By definition, $(n^+_{O,i} = n^+_{O,j+b-1} + \sum_{l=j+b}^{i} x_O[l])$ and $(n^+_{V,i} = n^+_{V,j+b-1} + \sum_{l=j+b}^{i} x_V[l])$. Applying the inductive hypothesis to $(j + b - 1 < i)$ shows that $(n^+_{V,j+b-1} \leq n^+_{O,j+b-1})$. Combining the above equations with Equation 3.22 determines that $n^+_{V,i} \leq n^+_{O,i}$. The inductive hypothesis is proven, and the result follows immediately.

□

3.7.2 Proof of Theorem 5 case $\tau_L \geq b$ and $\tau_L = (\tau - b)$

Let $(a = \lfloor \frac{\tau_L}{b} \rfloor)$ and $(e \equiv \tau_L \mod b)$. Theorem 5 does not apply when $\tau = (\tau_L - b)$ and $b|\tau$, necessitating that $(e > 0)$. Let d be an arbitrary multiple of $(a + 1)$.

Consider the following two frame-size sequences for which the offline construction will be shown below in Figures 3.9 and 3.10 respectively:

1. $k^{(1)}_0 = \ldots = k^{(1)}_{e-1} = d$, and $k^{(1)}_e = \ldots = k^{(1)}_t = 0$.
2. $k^{(2)}_0 = \ldots = k^{(2)}_{b-2} = d$, $k^{(2)}_{b-1} = d(\tau_L + 1)$, and $k^{(2)}_b = \ldots = k^{(2)}_t = 0$.

Before going into the details of the proof, we note that the proof applies for any value of d. When d is sufficiently large, the proof could also be extended to frame-size sequences where the frames' sizes may only approximately equal the ones in the frame-size sequence. More generally, the proof also applies for any frame-size sequences for which there is a subsequence of (a) τ frames whose sizes are $\ll d$, then (b) one of the two above frame-size sequences, then (c) another τ frames whose sizes are $\ll d$.

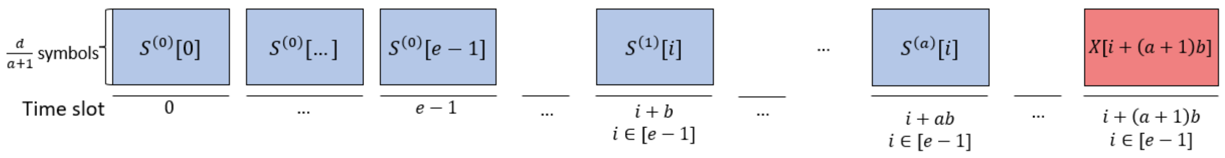


Figure 3.9: The offline scheme for frame-size sequence 1 for case $\tau_L \le b$ and $\tau_L = (\tau - b)$. Blue channel packets consist of frame symbols, and red channel packets consist of parity symbols. The numbers under the lines at the bottom indicate the time slots. The offline scheme sends $\frac{d}{a+1}$ symbols in each of the first e channel packets.

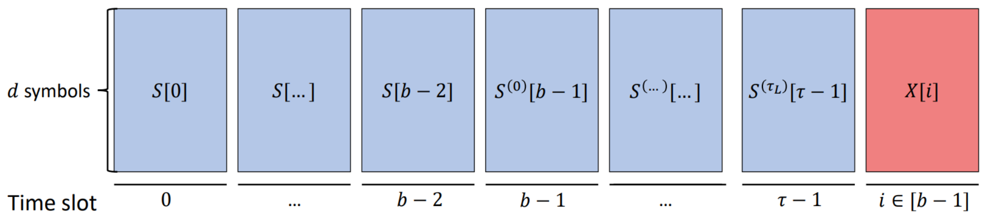


Figure 3.10: The offline scheme for frame-size sequence 2 for case $\tau_L \le b$ and $\tau_L = (\tau - b)$. Blue channel packets consist of frame symbols and red channel packets consist of parity symbols. The numbers under the lines at the bottom indicate the time slots. The offline scheme sends d symbols in each of the first e channel packets.

We present an offline coding scheme for frame-size sequences 1 and 2, which has rates

$$R_t^{(1)} = \frac{a+1}{a+2}, \qquad R_t^{(2)} = \frac{\tau}{\tau+b} \tag{3.23}$$

on the two frame-size sequences, respectively. We describe and then validate the scheme for each frame-size sequence.

Offline scheme for frame-size sequence* 1*: Each frame is encoded separately with parameters $(\tau' = \lfloor \frac{\tau}{b} \rfloor b, b' = b, \tau'_L = \tau' - b)$ as described in Section 3.2, shown in Figure 3.9, and detailed below.

- For $i \in [e-1]$, $S[i]$ is evenly divided into $(a+1)$ components of size d: $S^{(0)}[i], \dots, S^{(a)}[i]$. For $j \in [a], X[i+jb] = S^{(j)}[i]$.
- For $i \in [e-1]$, $X[i+(a+1)b] = \sum_{z=0}^{a} X[i+zb]$.

Decoding: For $i \in [e-1]$, $S[i]$ is sent evenly over $X[i], X[i+b], \dots, X[i+ab]$ where $(i+ab) \le (i+\tau_L)$ and at most one of $X[i], X[i+b], \dots, X[i+ab]$, or $X[i+(a+1)b] = \sum_{j=0}^{a} X[i+jb]$ is lost. Each frame is decoded within delay τ_L when the transmission is lossless and using a linear combination of the relevant $(a+1)$ channel packets within delay τ otherwise.

Offline scheme for frame-size sequence* 2*: The first $(b-1)$ frames are sent with no delay and the symbols of the next frame are transmitted evenly over $X[b-1], \dots, X[\tau-1]$. The symbols of $X[0 : \tau-1]$ are used to create d blocks of the rate $\frac{\tau}{\tau+b}$ systematic block code from [51]. Each of the d blocks includes b parity symbols that are sent in $X[\tau], \dots, X[\tau + b-1]$ respectively. The block code maps τ input symbols $(s_0, \dots, s_{\tau-1})$ to $(\tau+b)$ codeword symbols $(s_0, \dots, s_{\tau-1}, p_0, \dots, p_{b-1})$. For each $j \in [\tau-1]$ and any burst erasing up to b codeword symbols, the non-erased symbols of $(s_0, \ldots, s_{\tau-1}, p_0, \ldots, p_{min(b-1,j)})$ are sufficient to decode s_j. Therefore, each symbol is recovered within τ symbols. We note that although we use the block code from [51], any other block code from [27, 30, 35, 49] also works. The scheme is described in detail below and shown in Figure 3.10:

- For $j \in [b-2]$, $X[j] = S[j]$.
- $S[b-1]$ is divided evenly into $(\tau_L + 1)$ components of size d: $S^{(0)}[b-1], \ldots, S^{(\tau_L)}[b-1]$.
- For $j \in \{b-1, \ldots, b-1+\tau_L\}$, $X[j] = S^{(j-b+1)}[b-1]$.
- For each $z \in [d-1]$, an instance of the block code from [51] is created which maps $(X_z[0], \ldots, X_z[\tau-1])$ to $(X_z[0], \ldots, X_z[\tau-1], p_0^{(z)}, \ldots, p_{b-1}^{(z)})$.
- For $j \in [b-1]$, $X[\tau + j] = (p_j^{(0)}, \ldots, p_j^{(d-1)})$.

Decoding: Each frame is transmitted within the current and next τ_L channel packets and is, thus, decoded when the transmission is lossless. Each symbol $X_z[i]$ for $z \in [d-1]$ and $i \in [\tau-1]$ is decoded within the delay τ or by time slot $(\tau + b - 1)$ using the block code $(X_z[0], \ldots, X_z[\tau - 1], p_0^{(z)}, \ldots, p_{b-1}^{(z)})$. Hence, the worst-case-delay constraint is met.

Proof of the converse result : The offline-optimal-rate is at least $R_t^{(1)}$ and $R_t^{(2)}$ (that is, the rate of the offline scheme from Equation 3.23) for frame-size sequences 1 and 2, respectively. Next, we show mutually exclusive conditions for the sum of the sizes of $X[0], \ldots, X[e-1]$ to have rates at least $R_t^{(1)}$ and $R_t^{(2)}$ on frame-size sequences 1 and 2 respectively. All online coding schemes, thus, fail the condition for at least one frame-size sequence since they are identical until time slot e.

Condition for rate $R_t^{(1)}$ on frame-size sequence 1 : Consider any coding scheme for frame-size sequence 1. At least de symbols are sent over $X[b], \ldots, X[t]$ since $X[0], \ldots, X[b-1]$ could be lost. At most $d\frac{e}{a+1}$ symbols can be sent over $X[0], \ldots, X[b-1]$ if the rate is at least $R_t^{(1)}$.

Condition for rate $R_t^{(2)}$ on frame-size sequence 2 : Consider an arbitrary coding scheme for frame-size sequence 2. At least $d\tau$ symbols are sent in $X[0], \ldots, X[\tau-1]$ to meet the lossless-delay constraint. For each $i \in [a]$, at least $d\tau$ symbols are sent outside of $X[e+ib : e+(i+1)b-1]$ in case $X[e+ib : e+(i+1)b-1]$ is lost. Since the rate is $R_t^{(2)}$, at most db symbols are sent in $X[e+ib : e+(i+1)b-1]$. As such, at least $(d\tau - d(a+1)b = de)$ symbols are sent in $X[0 : e-1]$.

Summary : Any online scheme whose rate is at least $R_t^{(1)}$ on frame-size sequence 1 sends at most $d\frac{e}{a+1}$ symbols in $X[0 : b-1]$. As such, its rate is lower than $R_t^{(2)}$ on frame-size sequence 2.

3.7.3 Proof of Theorem 5 case $\tau_L < b$ and $\tau_L = (\tau - b)$

Let d be an arbitrary positive even integer. Consider the following two frame-size sequences for which the offline construction will be shown below in Figures 3.11 and 3.12 respectively:

1. $k_0^{(1)} = \ldots = k_{b-\tau_L}^{(1)} = d$, and $k_{b-\tau_L+1}^{(1)} = \ldots = k_t^{(1)} = 0$.
2. $k_0^{(2)} = \ldots = k_{b-\tau_L}^{(2)} = d$, $k_{b-\tau_L+1}^{(2)} = \ldots = k_{b-1}^{(2)} = 0$, $k_b^{(2)} = d$, and $k_{b+1}^{(2)} = \ldots = k_t^{(2)} = 0$.

Before presenting the proof in detail, we observe that the proof could also be extended to similar frame-size sequences where the sizes of each frame is perturbed by a small amount as long as

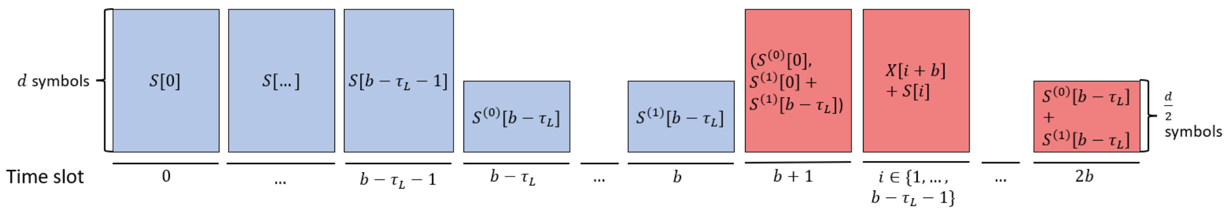


Figure 3.11: The offline scheme for frame-size sequence 1 for case $\tau_L < b$ and $\tau_L = (\tau - b)$. Blue channel packets consist of frame symbols, and red channel packets consist of parity symbols. The numbers under the lines at the bottom indicate the time slots. The offline scheme sends $\frac{d}{2}$ symbols in $X[b-\tau_L]$.

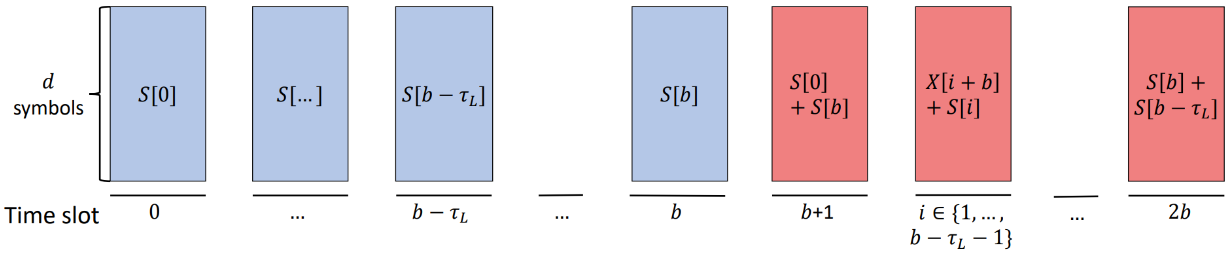


Figure 3.12: The offline scheme for frame-size sequence 2 for case $\tau_L < b$ and $\tau_L = (\tau - b)$. Blue channel packets consist of frame symbols and red channel packets consist of parity symbols. The numbers under the lines at the bottom indicate the time slots. The offline scheme sends d symbols in $X[b-\tau_L]$

d is large. More generally, the proof also applies to any frame-size sequence that contains one of the two above frame-size sequences proceeded and followed by τ frames sufficiently small relative to d.

We will present an offline coding scheme for the two frame-size sequences with rates

$$R_t^{(1)} = \frac{b-\tau_L+1}{2b-2\tau_L+1.5}, \qquad R_t^{(2)} = \frac{b-\tau_L+2}{2b-2\tau_L+3} \tag{3.24}$$

on frame-size sequence 1 and 2, respectively. After presenting the scheme for each frame-size sequence, we verify that it satisfies the lossless-delay and worst-case-delay constraints.

Offline scheme for frame-size sequence* 1*: The first $(b-\tau_L)$ frames are sent in the corresponding channel packets. The frame $S[b-\tau_L]$ is divided in half to be evenly transmitted over $X[b-\tau_L]$ and $X[b]$. Each of the next $(b-\tau_L)$ channel packets comprises d parity symbols used to decode (a) the first $(b-\tau_L)$ frames if the corresponding channel packets are lost and (b) $X[b]$ if $X[b-\tau_L]$ and $X[b]$ are both lost. The summation of $X[b-\tau_L]$ and $X[b]$ is later sent in $X[2b]$ to ensure decoding of $S[b-\tau_L]$ within delay τ. The scheme is detailed below and shown in Figure 3.11:

- $S[0]$ and $S[b-\tau_L]$ are each evenly divided into two components of $d/2$ symbols each: $S[0] = (S^{(0)}[0], S^{(1)}[0])$ and $S[b-\tau_L] = (S^{(0)}[b-\tau_L], S^{(1)}[b-\tau_L])$.
- For $i \in [b-\tau_L-1]$, $X[i] = S[i]$.
- $X[b-\tau_L] = S^{(0)}[b-\tau_L]$.
- $X[b] = S^{(1)}[b-\tau_L]$.

- $X[b+1] = (S^{(0)}[0], S^{(1)}[0] + S^{(1)}[b-\tau_L])$.
- For $i \in \{1, \ldots, b-\tau_L-1\}$, $X[i+b+1] = (X[i+b] + S[i])$.
- $X[b-\tau_L+\tau] = X[2b] = (S^{(0)}[b-\tau_L] + S^{(1)}[b-\tau_L])$.

Decoding: Each frame is sent within the current and perhaps next τ_L channel packets and is decoded when the transmission is lossless. We now discuss how frames are recovered within a delay of τ under lossy conditions. Either $X[0] = S[0]$ is received, or $X[0]$ is lost. In the latter case, both $X[b] = S^{(1)}[b-1]$ and $X[b+1] = (S^{(0)}[0], S^{(1)}[0] + S^{(1)}[b-1])$ are received. Therefore, $S[0]$ is decoded within the delay of τ. Next, for $i \in \{1, \ldots, b-\tau_L-1\}$, either $X[i] = S[i]$ is received, or both $X[i+b]$ and $X[i+b+1] = (X[i+b] + S[i])$ are received. Thus, $S[i]$ is recovered within delay $(b+1 \leq \tau)$. Either $X[b-\tau_L] = S^{(0)}[b-\tau_L]$ is received, or $X[2b-\tau_L] = \Big((S^{(0)}[0], S^{(1)}[0] + S^{(1)}[b-\tau_L]) + \sum_{i=1}^{b-\tau_L-1} S[i]\Big)$ is received. In the latter case, $S[0], \ldots, S[b-\tau_L-1]$ are decoded by time slot $(2b-1)$ and combined with $X[2b-\tau_L]$ to decode $S^{(1)}[b-\tau_L]$. $S^{(1)}[b-\tau_L]$ is then combined with $X[2b] = (S^{(0)}[b-\tau_L] + S^{(1)}[b-\tau_L])$ to recover $S^{(0)}[b-\tau_L]$ within a delay of τ. Therefore, $S^{(0)}[b-\tau_L]$ is decoded within delay τ. Either $X[b] = S^{(1)}[b-\tau_L]$ is received, or $X[2b] = (S^{(0)}[b-\tau_L] + S^{(1)}[b-\tau_L])$ is received. Recall that $S^{(0)}[b-\tau_L]$ is decoded by time slot $2b$. Thus, $S^{(1)}[b-\tau_L]$ is recovered within delay τ.

Offline scheme for frame-size sequence* 2*: Each frame $S[i]$ is transmitted in the corresponding channel packet $X[i]$. The next τ_L channel packets each comprise d parity symbols. These $d\tau_L$ symbols are used to decode (a) the first $(b-\tau_L)$ frames when the corresponding channel packets are lost, and (b) $S[b]$ when both $X[b-\tau_L] = S[b-\tau_L]$ and $X[b] = S[b]$ are lost. The sum of $S[b-\tau_L]$ and $S[b]$ is sent in $X[2b]$ to ensure that $S[b-\tau_L]$ is recovered if $X[b-\tau_L]$ is dropped. The scheme is described in full detail below and shown in Figure 3.12 :

- For $i \in [b-\tau_L] \cup \{b\}$, $S[i] = X[i]$.
- $X[b+1] = (S[0] + S[b])$.
- For $i \in \{1, \ldots, b-\tau_L-1\}$, $X[i+b+1] = (X[b+i] + S[i])$.
- $X[2b] = (S[b] + S[b-\tau_L])$.

Decoding: Each frame is transmitted within the corresponding channel packet and is decoded when the transmission is lossless. We now discuss how each frame is decoded within a delay of τ under lossy conditions. Either $X[0] = S[0]$ is received or both $X[b] = S[b]$ and $X[b+1] = (S[0] + X[b])$ are received. Consequently, $S[0]$ is decoded within delay $(b+1 \leq \tau)$. For $i \in \{1, \ldots, b-\tau_L-1\}$, either $X[i] = S[i]$ is received, or both $X[i+b]$ and $X[i+b+1] = (X[i+b] + S[i])$ is received. Therefore, each $S[i]$ is recovered within delay $(b+1 \leq \tau)$. Either $X[b-\tau_L] = S[b-\tau_L]$ is received, or $X[2b-\tau_L] = (S[b] + \sum_{i=0}^{b-\tau_L-1} S[i])$ is received. In the latter case, $S[0], \ldots, S[b-\tau_L-1]$ are decoded by time slot $(b-\tau_L+\tau)$ and combined with $X[2b-\tau_L]$ to decode $S[b]$. Then, $S[b]$ and $X[2b] = (S[b] + S[b-\tau_L])$ used to recover $S[b-\tau_L]$. Therefore, $S[b-\tau_L]$ is decoded within delay τ. Either $X[b] = S[b]$ is received, or $X[2b] = (S[b] + S[b-\tau_L])$ is received. In the latter case, subtracting $S[b-\tau_L]$ yields $S[b]$. Hence, $S[b]$ is recovered within a delay of τ.

Proof of the converse result : The offline-optimal-rate is at least $R_t^{(1)}$ and $R_t^{(2)}$ on frame-size sequences 1 and 2, respectively (i.e., the rate of the offline scheme from Equation 3.24). Next, we present necessary and mutually exclusive conditions on the total number of symbols sent in $X[0], \ldots, X[b-1]$ for a code construction to attain rates at least $R_t^{(1)}$ and $R_t^{(2)}$ on the two respective frame-size sequences. The two frame-size sequences are the same until time slot b. Therefore, no online coding scheme can satisfy the condition for both frame-size sequences.

Condition for rate $R_t^{(1)}$ on frame-size sequence 1 : Consider an arbitrary coding scheme for frame-size sequence 1. At least $d(b - \tau_L + 1)$ symbols are transmitted in $X[b], \ldots, X[t]$ since $X[0], \ldots, X[b-1]$ could be dropped in a burst. At most, an additional $d(b - \tau_L + .5)$ symbols can be sent over $X[0], \ldots, X[b-1]$ if the rate is at least $R_t^{(1)}$.

Condition for rate $R_t^{(2)}$ on frame-size sequence 2 : Consider any coding scheme for frame-size sequence 2. We will show that if

$$d' = \sum_{i=0}^{b-1} n_i \le d(b - \tau_L + .5) \tag{3.25}$$

then the rate is strictly less than $R_t^{(2)}$. At a high level, at least $d(b - \tau_L + 2)$ symbols are sent in $X[0], \ldots, X[b-1], X[2b], \ldots, X[t]$ to satisfy the worst-case-delay constraint in the event that $X[b], \ldots, X[2b-1]$ are lost. At least $d(b - \tau_L + 1.5)$ symbols must be sent in $X[b], \ldots, X[2b-1]$ for the lossless-delay and worst-case-delay constraints to be satisfied, as will be shown shortly. In total, $d(2b - 2\tau_L + 3.5)$ symbols are sent, whereas at most $d(2b - 2\tau_L + 3)$ symbols are transmitted as part of a scheme with a rate of at least $R_t^{(2)}$.

Next, the fact that sending at most $d(b - \tau_L + .5)$ symbols over $X[0], \ldots, X[b-1]$ leads to a rate of less than $R_t^{(2)}$ on frame-size sequence 2 is proven in detail. Let $\mathcal{S}$ be a random variable drawn uniformly at random from the finite field $\mathbb{F}_q$. Recall from Appendix 3.7.1 that for any $i \in [t]$, (a) $H(S[i]) = H(\mathcal{S})k_i$, and (b) $H(X[i]) \le H(\mathcal{S})n_i$ (Equations 3.19 and 3.20).

We provide an upper bound on the sizes of the channel packets as follows

$$d(b - \tau_L + 2)H(\mathcal{S}) = H(S[0:b]) \le \tag{3.26}$$

$$H(S[0:b], X[0:b-1], X[2b:b+\tau]) = \tag{3.27}$$

$$\begin{aligned} &H(X[0:b-1], X[2b:b+\tau]) + \\ &H\big(S[0:b] \big| X[0:b-1], X[2b:b+\tau]\big) = \end{aligned} \tag{3.28}$$

$$H(X[0:b-1], X[2b:b+\tau]) \le \tag{3.29}$$

$$H(\mathcal{S}) \left(\sum_{i=0}^{b-1} n_i + \sum_{i=2b}^{b+\tau} n_i \right). \tag{3.30}$$

Equation 3.26 follows from Equation 3.19, Equation 3.27 follows from the definition of entropy, Equation 3.28 follows from the chain rule, Equation 3.29 follows from the worst-case-delay constraint, and Equation 3.30 follows from Equation 3.20.

Next, we will prove that $H\left(X[b:2b-1]\right) \geq d(b-\tau_L+1.5)H\left(\mathcal{S}\right)$ as follows

$$\begin{aligned}
&H\left(X[0:b-1], S[0:b-\tau_L-1]\right) = \\
&H\left(X[0:b-1]\right) + H\left(S[0:b-\tau_L-1]\middle|X[0:b-1]\right) = \quad (3.31)\\
&H\left(X[0:b-1]\right) \leq d'H\left(\mathcal{S}\right) \quad (3.32)\\
&H\left(X[0:b-1], S[0:b-\tau_L-1]\right) = \\
&H\left(S[0:b-\tau_L-1]\right) + \quad (3.33)\\
&H\left(X[0:b-1]\middle|S[0:b-\tau_L-1]\right) = \\
&d(b-\tau_L)H\left(\mathcal{S}\right) + H\left(X[0:b-1]\middle|S[0:b-\tau_L-1]\right) \quad (3.34)
\end{aligned}$$

where Equation 3.31 follows from the chain rule, Equation 3.32 follows from the lossless-delay constraint and Equation 3.25, Equation 3.33 follows from the chain rule, and Equation 3.34 follows from applying Equation 3.19 to $S[0], \ldots, S[b-\tau-1]$.

Rearranging terms yields

$$H\left(X[0:b-1]\middle|S[0:b-\tau_L-1]\right) \leq \left(d' - d(b-\tau_L)\right)H\left(\mathcal{S}\right) \quad (3.35)$$

Next, we bound the sizes of $X[b], \ldots, X[2b-1]$ using

$$\begin{aligned}
&d(b-\tau_L+2)H\left(\mathcal{S}\right) \leq H\left(S[0:b]\right) \leq \quad (3.36)\\
&H\left(S[0:b], X[0:2b-1]\right) \leq \quad (3.37)\\
&H\left(X[b:2b-1]\right) + H\left(S[0:b-\tau_L-1]\middle|X[b:2b-1]\right) + \\
&H\left(X[0:b-1]\middle|S[0:b-\tau_L-1]\right) + \quad (3.38)\\
&H\left(S[b-\tau_L:b]\middle|X[0:2b-1]\right) = \\
&H\left(X[b:2b-1]\right) + H\left(X[0:b-1]\middle|S[0:b-\tau_L-1]\right) \quad (3.39)\\
\leq &H\left(X[b:2b-1]\right) + \left(d' - d(b-\tau_L)\right)H\left(\mathcal{S}\right), \quad (3.40)
\end{aligned}$$

where Equation 3.36 follows from Equation 3.19, Equation 3.37 follows from the definition of entropy, Equation 3.38 follows from the definition of conditioning, Equation 3.39 follows from the worst-case-delay constraint (i.e., $\tau = (\tau_L + b)$) and the lossless-delay (i.e., $\tau_L < b$), and Equation 3.40 follows from Equation 3.35.

Rearranging terms yields

$$\begin{aligned}
&\left(d(2b-2\tau_L+2) - d'\right)H\left(\mathcal{S}\right) \leq H\left(X[b:2b-1]\right) \\
&\leq H\left(\mathcal{S}\right)\sum_{i=b}^{2b-1} n_i. \quad (3.41)
\end{aligned}$$

The total number of symbols sent in $X[0:b-1]$ and $X[2b:b+\tau]$ is at least $d(b-\tau_L+2)$ by Equations 3.26 through 3.30. At least $(d(2b-2\tau_L+2)-d')$ symbols are sent in $X[b:2b-1]$ by Equation 3.41. In total, at least

$$\begin{aligned}
&d(3b-3\tau_L+4) - d' \geq \left(d(3b-3\tau_L+4) - d(b-\tau_L+.5)\right) \\
&= d(2b-2\tau_L+3.5)
\end{aligned}$$

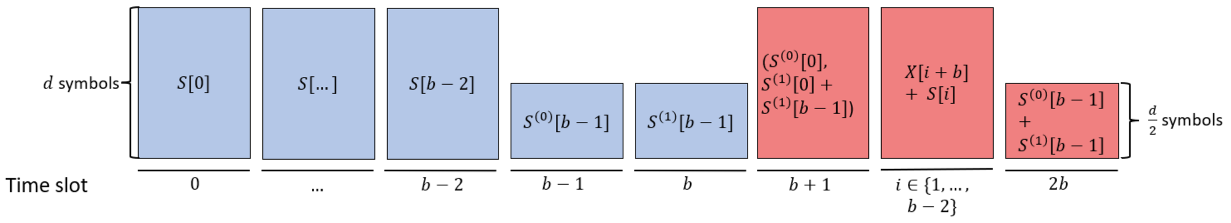


Figure 3.13: The offline scheme for frame-size sequence 1 for case $\tau_L < (\tau - b)$. Blue channel packets consist of frame symbols, and red channel packets consist of parity symbols. The numbers under the lines at the bottom indicate the time slots. The offline scheme sends $\frac{d}{2}$ symbols in $X[b-1]$.

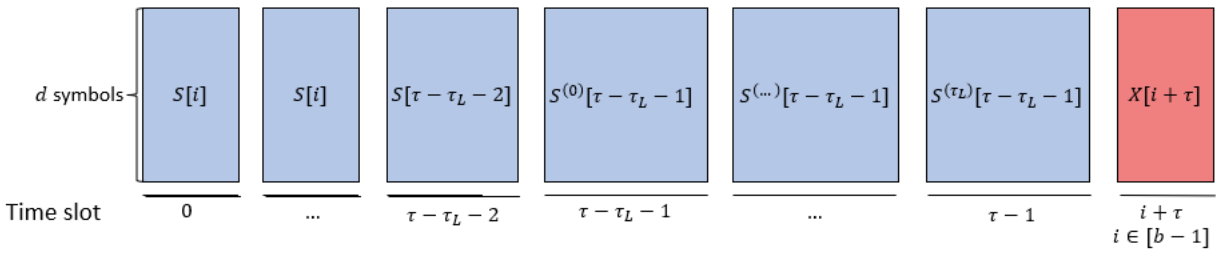


Figure 3.14: The offline scheme for frame-size sequence 2 for case $\tau_L < (\tau - b)$. Blue channel packets consist of frame symbols and red channel packets consist of parity symbols. The numbers under the lines at the bottom indicate the time slots. The offline scheme sends d symbols in $X[b-1]$

symbols are sent. Thus, the rate is strictly lower than $R_t^{(2)}$.

Summary : Any online scheme with rate at least $R_t^{(1)}$ on frame-size sequence 1 sends at most $d(b - \tau_L + .5)$ symbols over $X[0], \ldots, X[b-1]$. Consequently, its rate is strictly less than $R_t^{(2)}$ on frame-size sequence 2.

3.7.4 Proof of Theorem 5 case $\tau_L < (\tau - b)$

Let d be an arbitrary positive even integer. Consider the following two frame-size sequences for which the offline construction will be shown below in Figures 3.13 and 3.14 respectively:

1. $k_0^{(1)} = \ldots = k_{b-1}^{(1)} = d$, and $k_b^{(1)} = \ldots = k_t^{(1)} = 0$.
2. $k_0^{(2)} = \ldots = k_{\tau-\tau_L-2}^{(2)} = d$, $k_{\tau-\tau_L-1}^{(2)} = d(\tau_L + 1)$, and $k_{\tau-\tau_L}^{(2)} = \ldots = k_t^{(2)} = 0$.

Before we present the details of the proof, we point out that a similar proof applies to when the sizes of the frames are approximately equal to those of the frame-size sequences, as long as the deviation is small relative to d. In addition, the proof extends to scenarios where one of the two above frame-size sequence occurs at some point in the transmission proceeded and followed by τ frames whose sizes are much less than d.

We will describe an offline coding scheme for frame-size sequences 1 and 2 with rates

$$R_t^{(1)} = \frac{b}{2b - .5}, \qquad R_t^{(2)} = \frac{\tau}{\tau + b} \tag{3.42}$$

on the two respective frame-size sequences. We also verify that the lossless-delay and worst-case-delay constraints are satisfied.

Offline scheme for frame-size sequence 1***:*** Each of $S[0], \ldots, S[b-2]$ is transmitted immediately as part of the corresponding channel packet. Then $S[b-1]$ is divided in half and evenly sent over $X[b-1]$ and $X[b]$. The next $(b-1)$ channel packets each comprise d parity symbols. These $d(b-1)$ parity symbols are used to decode (a) frames $S[0], \ldots, S[b-2]$ when the corresponding channel packets are lost, and (b) $X[b]$ when both $X[b-1]$ and $X[b]$ are lost. The summation of $X[b-1]$ and $X[b]$ is sent in $X[2b]$ to ensure that $S[b-1]$ is decoded within a delay of τ. The scheme is described in detail below and shown in Figure 3.13 :

- The frames $S[0]$ and $S[b-1]$ are divided in half into $S[0] = (S^{(0)}[0], S^{(1)}[0])$ and $S[b-1] = (S^{(0)}[b-1], S^{(1)}[b-1])$.
- For $j \in [b-2]$, $X[j] = S[j]$.
- $X[b-1] = S^{(0)}[b-1]$.
- $X[b] = S^{(1)}[b-1]$.
- $X[b+1] = (S^{(0)}[0], S^{(1)}[0] + S^{(1)}[b-1])$.
- For $i \in \{1, \ldots, b-2\}$, $X[i+b+1] = (X[i+b] + S[i])$.
- $X[2b] = (S^{(0)}[b-1] + S^{(1)}[b-1])$.

Decoding: Each frame is sent within the current and perhaps next channel packets and is decoded when the transmission is lossless. We now discuss how each frame is decoded within delay τ under lossy conditions. Either $X[0] = S[0]$ is received, or both $X[b] = S_1[b-1]$ and $X[b+1] = (S^{(0)}[0], S^{(1)}[0] + S^{(1)}[b-1])$ are received. Thus, $S[0]$ is decoded within a delay of $(b+1 \leq \tau)$. For $j \in \{1, \ldots, b-2\}$, either $X[j] = S[j]$ is received, or both $X[j+b]$ and $X[j+b+1] = (X[j+b] + S[j])$ are received. Therefore, $S[j]$ is decoded within delay $(b+1 \leq \tau)$. Either $X[b-1] = S^{(0)}[b-1]$ is received, or $X[2b-1]$ is received. In the latter case, $S[0], \ldots, S[b-2]$ are decoded by time slot $(2b-1)$ and are combined with $X[2b-1] = \left((S^{(0)}[0], S^{(1)}[0] + S^{(1)}[b-1]) + \sum_{i=1}^{b-2} S[i]\right)$ to recover $S^{(1)}[b-1]$. The receiver then decodes $S^{(0)}[b-1] = (X[2b] - S^{(1)}[b-1])$ within delay $(b+1 \leq \tau)$. Either $X[b] = S^{(1)}[b-1]$ is received, or $X[2b] = (S^{(0)}[b-1] + S^{(1)}[b-1])$ is received and combined with $S^{(0)}[b-1]$ to recover $S^{(1)}[b-1]$ within delay τ.

Offline scheme for frame-size sequence 2***:*** Each of $S[0], \ldots, S[\tau - \tau_L - 2]$ is transmitted within the corresponding channel packet. The symbols of $S[\tau - \tau_L - 1]$ are evenly divided into $(\tau_L + 1)$ components sent over $X[\tau - \tau_L - 1], \ldots, X[\tau - 1]$ respectively. Each of $X[\tau], \ldots, X[\tau + b - 1]$ comprises d symbols, which creates d blocks of the $[\tau + b, \tau]$ systematic block codes (described in Section 3.7.2). The scheme is presented in detail below and shown in Figure 3.14 :

- For $j \in [\tau - \tau_L - 2]$, $X[j] = S[j]$.
- The frame $S[\tau - \tau_L - 1]$ is evenly divided into $(\tau_L + 1)$ components of size d: $(S^{(0)}[\tau - \tau_L - 1], \ldots, S^{(\tau_L)}[\tau - \tau_L - 1])$.
- For $j \in \{\tau - \tau_L - 1, \ldots, \tau - 1\}$, $X[j] = S^{(j - \tau + \tau_L + 1)}[\tau - \tau_L - 1]$.
- For each $z \in [d-1]$, an instance of the block code from [51] is created that maps $(X_z[0], \ldots, X_z[\tau - 1])$ to $(X_z[0], \ldots, X_z[\tau - 1], p_0^{(z)}, \ldots, p_{b-1}^{(z)})$.

- For $j \in [b-1]$, $X[\tau + j] = (p_j^{(0)}, \ldots, p_j^{(d-1)})$.

Decoding: Each frame is sent over the current and perhaps next τ_L channel packets and is decoded when the transmission is lossless. When there are losses, the block code $(X_z[0], \ldots, X_z[\tau-1], p_0^{(z)}, \ldots, p_{b-1}^{(z)})$ is used for decoding. For $z \in [d-1]$: (a) Each symbol $X_z[i]$, for $i \in [b-1]$, is decoded within a delay of τ. (b) Each symbol $X_z[i]$, for $i \in [\tau - 1] \setminus [b-1]$, is decoded by time slot $(\tau + b - 1)$. Thus, the worst-case-delay constraint is satisfied.

Proof of the converse result : The rates $R_t^{(1)}$ and $R_t^{(2)}$ of the above construction (Equation 3.42) for frame-size sequences 1 and 2, respectively, serve as a lower bound on the offline-optimal-rate for the two frame-size sequences. Next, we present mutually exclusive conditions on the number of symbols transmitted in the first b channel packets to have rates at least $R_t^{(1)}$ or $R_t^{(2)}$ on frame-size sequences 1 or 2, respectively. The online coding schemes cannot differentiate between the two frame-size sequences before the time slot b. Hence, the number of symbols sent in $X[0], \ldots, X[b-1]$ by any online scheme violates the condition for at least one frame-size sequence.

Condition for rate $R_t^{(1)}$ on frame-size sequence 1 : Consider any coding scheme for frame-size sequence 1. At least db symbols are transmitted in $X[b], \ldots, X[t]$ in case there is a burst loss of $X[0], \ldots, X[b-1]$. The rate is at least $R_t^{(1)}$, so at most $d(b - .5)$ additional symbols are sent in $X[0], \ldots, X[b-1]$.

Condition for rate $R_t^{(2)}$ on frame-size sequence 2 : Consider any coding scheme for frame-size sequence 2. We will demonstrate that if

$$\sum_{i=0}^{b-1} n_i \leq d(b - .5) \tag{3.43}$$

then the rate is strictly less than $R_t^{(2)} = \frac{\tau}{\tau+b}$ in two steps. First, we will show that all symbols are transmitted by $X[\tau + b - 1]$ without loss of generality. Second, we prove that strictly more than db symbols may be lost. At least $d\tau$ additional symbols are sent to meet the worst-case-delay constraint, leading to a lower rate than $R_t^{(2)}$.

Step 1: If $X[\tau + b - 1]$ is lost, then $X[0 : \tau - 1]$ are received, which yields $S[0 : \tau - \tau_L - 1]$ by the lossless-delay constraint. Thus, all symbols sent after the time slot $(\tau + b)$ can instead be sent in $X[\tau + b - 1]$.

Step 2: Consider the following erasure channels C_i for $i \in [\tau + b - 1]$. Each C_i introduces bursts of packet losses in $\{X[j], \ldots, X[j + b - 1] \mid j \equiv i \mod (\tau + b)\}$ and results in l_i lost (dropped) symbols.[8] At least $d(\tau + b)$ symbols are sent in total due to the upper bound on the

[8]A similar argument was used to show the upper bound on rate of $\frac{\tau}{\tau+b}$ in [63].

rate of $\frac{\tau}{\tau+b}$, leading to

$$\sum_{i=0}^{\tau+b-1} l_i \geq db(\tau+b) \tag{3.44}$$

$$\sum_{i=1}^{\tau+b-1} l_i \geq db(\tau+b-1) + .5d \tag{3.45}$$

$$\frac{1}{\tau+b-1} \sum_{i=1}^{\tau+b-1} l_i \geq db + \frac{.5d}{\tau+b-1}, \tag{3.46}$$

where Equation 3.44 follows from each packet (and hence each symbol) being dropped by exactly b channels, and Equation 3.45 follows from Equation 3.43.

Hence, there is some $i \in \{1, \ldots, \tau+b-1\}$ for which $l_i \geq (db + \frac{.5d}{\tau+b-1})$. In order to satisfy the worst-case-delay constraint over channel C_i, at least $d\tau$ symbols are received outside of the channel packets dropped by C_i. Thus, the total number of symbols sent is at least $d(\tau+b+\frac{.5}{\tau+b-1})$. In contrast, at most $d(\tau+b)$ symbols are sent if the rate is at least $R_t^{(2)}$.

Summary : Any online coding scheme with a rate of at least $R_t^{(1)}$ on frame-size sequence 1 sends at most $d(b-.5)$ symbols in $X[0], \ldots, X[b-1]$. Consequently, its rate is strictly lower than $R_t^{(2)}$ on frame-size sequence 2.

Chapter 4

Learning-augmented streaming codes are approximately optimal for variable-size frames

Recall from Chapter 3 that the optimal rate for offline schemes exceeds that of "online" schemes in all but two settings (i.e., Regime 2 and Regime 1). In Regime 2, τ_L has its maximum possible value, and a technique for spreading the symbols of each frame over several channel packets independently of all other frames is rate optimal. In Regime 1, τ_L has its minimum possible value (i.e., 0), requiring sending the symbols of each frame in the corresponding channel packet. Therefore, information about the sizes of the future frames does not help. Rate-optimal constructions, or even approximately rate-optimal constructions, are not known even for the offline setting for all remaining parameter regimes.

Inspired by the growing field of learning-augmented algorithms, this work introduces a new methodology for constructing online streaming codes that combines machine learning with algebraic coding theory tools. Using the methodology, we design an approximately rate-optimal streaming code for Regime 3 (i.e., $\tau_L = 1$). Doing so establishes that the method is viable for the key parameter regime of the smallest lossless-delay that allows the benefits of spreading. We begin in Section 4.1 by presenting the system model used in this work, which is built on top of the model of streaming codes used earlier in this work. We then present our approach in three steps: (a) isolate spreading as the component that can benefit from machine learning by designing an optimal streaming code given any choice of how to spread (Section 4.2), (b) solve the offline version of how to spread optimally using optimization (Section 4.3), and (c) convert the offline scheme into an online one using a learning-based approach (Section 4.4).

4.1 Model and background

Recall that a transmission occurs over $(t + 1)$ time slots for a non-negative integer t. During the ith time slot for $i \in \{0, \ldots, t\}$, the sender obtains a frame, $S[i] \in \mathbb{F}^{k_i}$, where $\mathbb{F}$ is a finite field, and $k_i \in \{0, \ldots, m\}$ for a positive integer m. The sender also receives "side information," $O[i]$, that captures the differences between the online and offline settings. In the offline setting, which assumes knowledge of the future, the side information is the sizes of the future frames. In the online setting, the side information is independent $\mathcal{S}$ samples from the distribution of the sizes of future frames for some positive integer $\mathcal{S}$. Let $D_{k_0,\ldots,k_i}$ be the conditional distribution of $k_{i+1},\ldots,k_t$ given $k_0,\ldots,k_i$. Then,

$$O[i] = \begin{cases} (k_{i+1},\ldots,k_t) & \text{if offline} \\ \left\langle \left(k_{i+1}^{(j)},\ldots,k_t^{(j)}\right) \sim D_{k_0,\ldots,k_i} \mid j \in \{0,\ldots,\mathcal{S}-1\} \right\rangle & \text{if online.} \end{cases} \tag{4.1}$$

Recall that this chapter considers $\tau_L = 1$.

Encoding and decoding depends on the history of the transmission, which is summarized as follows.

Definition 1 (State)**.** *For any t, τ, and $i \in \{2\tau,\ldots,t\}$, $\mathcal{X}_i = (k_0,\ldots,k_i, X[0],\ldots,X[i-1])$ denotes the state.*

This chapter considers systematic codes for clarity, but we propose to extend the results to general codes as part of the thesis. To meet the lossless-delay constraint, the symbols of $S[i]$ must be sent by time slot $(i+\tau_L)$ (i.e., in $X[i]$ and $X[i+1]$). The "policy" of a construction, as defined below, specifies how to spread the frame symbols.

Definition 2 (Policy)**.** *The policy of a construction for any $i \in [t]$ and state $\mathcal{X}_i$ is the number of symbols of $S[i]$ sent in channel packet $X[i]$. The policy is denoted as $\mathcal{F}_i(\mathcal{X}_i)$ (or f_i for conciseness) and lies in $[k_i]$.*

For any $i > 0$, $X[i]$ comprises (a) the first f_i symbols of $S[i]$, (b) the final $(k_{i-1} - f_{i-1})$ symbols of $S[i-1]$, and (c) p_i parity symbols, denoted as $P[i]$. The encoding is given by $X[i] =$

$$Enc(\mathcal{X}_i, S[i-\tau],\ldots,S[i-1], S_0[i],\ldots,S_{f_i-1}[i], O[i]) \tag{4.2}$$

for $i \geq 2\tau$, and $X[i]$ is empty for $i < 2\tau$. This section assumes that $X[i]$ is independent of the frame symbols of $S[i]$ sent in $X[i+1]$ for clarity. We propose to extend the results to hold without this assumption. The receiver obtains $Y[i] \in \{X[i], *\}$ depending on whether channel packet $X[i]$ is received or dropped.[1] Under lossless transmission, $S[i]$ is available in uncoded form. Otherwise, $S[i]$ is decoded as

$$Dec\left(\langle Y[j], k_j, f_j \mid j \in [i+\tau]\rangle\right). \tag{4.3}$$

Finally, we use the notation that a length v vector, V, is indexed for any $i \leq j \in [v-1]$ as $V_i^j = (V_i,\ldots,V_j)$.

4.2 A Building block construction

This section presents a rate-optimal construction, called the "$(\tau, b, t, \langle f_i \mid i \in [t]\rangle)$ $-$Spreading Variable-sized Generalized MS Code ," for any given policies, i.e., choice of how to spread the

[1]The receiver needs the sizes of the frames and policies to decode. Thus, a small header with up to $2\sum_{j=i-b}^{i} 2\log(k_j) \leq 2(b+1)\log(m)$ symbols containing $(k_{i-b},\ldots,k_i,\mathcal{F}_{i-b},\ldots,\mathcal{F}_i)$ is added to the header of $X[i]$.

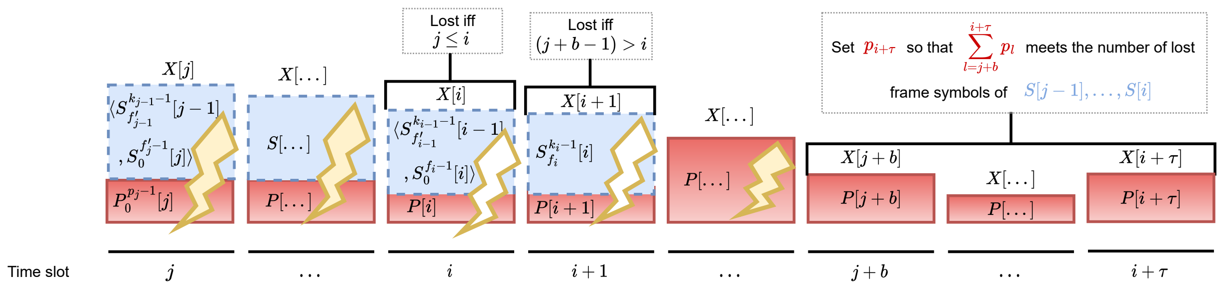


Figure 4.1: Selecting $p_{i+\tau}$ by considering each burst starting in time slot $j \in \{i-b+1, \ldots, i+1\}$ (shown with lightening bolts).

frame symbols over channel packets. Specifically, for any given $\langle f_i \mid i \in [t]\rangle$, at least f_i symbols of $S[i]$ will be sent in $X[i]$ under the construction for each $i \in [t]$.

The first 2τ channel packets are empty. For each $i \in [t] \setminus [2\tau - 1]$, $X[i]$ comprises (a) the first f'_i symbols of $S[i]$ for some $f'_i \ge f_i$, (b) the final $(k_{i-1} - f'_{i-1})$ symbols of $S[i-1]$, and (c) p_i parity symbols called $P[i]$. Next, we define f'_i, p_i, and $P[i]$ for any frame-size sequence, $k_0, \ldots, k_t$.

Defining each f'_i and p_i. For time slots $i \in [2\tau - 1] \cup \{t - 2\tau + 1, \ldots, t\}$, $f'_i = k_i = 0$. For time slots $i \in [3\tau - 1] \cup \{t - \tau + 1, \ldots, t\}$, $p_i = 0$. For all $i = 2\tau, \ldots, (t - 2\tau)$, we define $p_{i+\tau}$ to be as small as possible while ensuring that $S[i]$ is decoded by time slot $(i + \tau)$ under any lossy transmission. Specifically, $p_{i+\tau} =$

$$\max_{j \in \{i-b+1,\ldots,i+1\}} \left(0, \mathbb{1}[j+b-1 \ge i+1]\,(k_i - f_i) + \mathbb{1}[j \le i] f_i + \sum_{l=j}^{i} (k_{l-1} - f'_{l-1}) + \sum_{l=j}^{i-1} f'_l - \sum_{l=j+b}^{i+\tau-1} p_l\right), \tag{4.4}$$

as is illustrated in Figure 4.1. We then use $p_{i+\tau}$ to define

$$f'_i = \max(f_i, p_{i+\tau}). \tag{4.5}$$

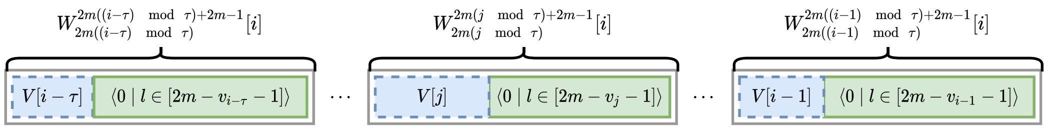
Constructing parity symbols. The parity symbols are defined analogously to those of the construction from Chapter 3 (which builds on the construction from [11]). For $i \in \{2\tau, \ldots, t - \tau\}$, the frame symbols sent in $X[i]$ are partitioned into (a) symbols of $S[i]$ that are recovered during time slot $(i + \tau)$ under a lossy transmission, and (b) symbols of $S[i-1]$ and $S[i]$ that are recovered by time slot $(i - 1 + \tau)$ under a lossy transmission. The two components are of sizes u_i and v_i and are denoted as $U[i]$ and $V[i]$, respectively. Thus, $X[i] = (U[i], V[i], P[i])$, where

$$U[i] = S_0^{p_{i+\tau}-1}[i] \tag{4.6}$$

$$V[i] = \left(S_{p_{i+\tau}}^{k_i - f'_i - 1}[i], S_{f'_{i-1}}^{k_{i-1}-1}[i-1]\right) \tag{4.7}$$

$$P[i] = U[i-\tau] + P^{(v)}[i]. \tag{4.8}$$

Each symbol of $P^{(v)}[i]$ is a linear combination of the symbols of $(V[i-\tau], \ldots, V[i-1])$, where the linear equations are chosen using a $(2m\tau) \times (2m\tau)$ Cauchy matrix, A, as follows. Let $W[i]$



$W_{2m((i-\tau) \bmod \tau)}^{2m((i-\tau) \bmod \tau)+2m-1}[i]$
$V[i-\tau]$
$\langle 0 \mid l \in [2m - v_{i-\tau} - 1]\rangle$
$W_{2m(j \bmod \tau)}^{2m(j \bmod \tau)+2m-1}[i]$
$V[j]$
$\langle 0 \mid l \in [2m - v_j - 1]\rangle$
$W_{2m((i-1) \bmod \tau)}^{2m((i-1) \bmod \tau)+2m-1}[i]$
$V[i-1]$
$\langle 0 \mid l \in [2m - v_{i-1} - 1]\rangle$

Figure 4.2: Defining $W[i]$ by placing the symbols of $V[j]$ in positions $2m(j \bmod \tau), \dots, (2m(j \bmod \tau) + v_j - 1)$ for $j \in \{i - \tau, \dots, i - 1\}$. The remaining positions are filled with 0's.

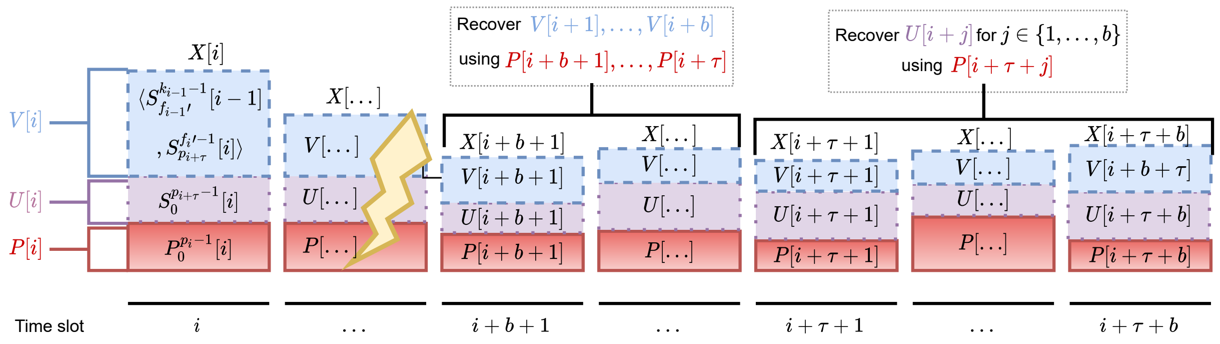


Figure 4.3: An example the $(\tau, b, t, \langle f_i \mid i \in [t]\rangle)$ −Spreading Variable-sized Generalized MS Code recovering a burst of length b starting in time slot $(i + 1)$.

be a length $2m\tau$ vector where positions $2m(j \bmod \tau), \dots, (2m(j \bmod \tau) + 2m - 1)$ comprise $V[j]$ followed by $(2m - v_j)$ 0's for $j \in \{i - \tau, \dots, i - 1\}$, as is illustrated in Figure 4.2. Finally, we define

$$P^{(v)}[i] = W[i]A_{(i)}, \tag{4.9}$$

where $A_{(i)}$ is A restricted to columns $2m(i \bmod \tau), \dots, (2m(i \bmod \tau) + p_i - 1)$. Figure 4.3 shows recovering a burst using the $\big(\tau, b, t, \langle f_i \mid i \in [t]\rangle\big)$ −Spreading Variable-sized Generalized MS Code .

Decoding. For $i \in [t]$, $S[i]$ is decoded (a) from $X[i]$ and $X[i + 1]$ under lossless conditions, and (b) by solving a system of linear equations corresponding to the symbols of $S[i - \tau], \dots, S[i - 1], Y[i], \dots, Y[i + \tau]$ when losses occur.

Next, we show the $(\tau, b, t, \langle f_i \mid i \in [t]\rangle)$ −Spreading Variable-sized Generalized MS Code meets the lossless-delay and worst-case-delay constraints.

Lemma 11. *For any parameters (τ, b), an arbitrary frame-size sequence $k_0, \dots, k_t$, and any policy f_i for $i \in [t]$, the $(\tau, b, t, \langle f_i \mid i \in [t]\rangle)$ −Spreading Variable-sized Generalized MS Code satisfies the lossless-delay constraint and worst-case-delay constraint.*

Proof sketch. The lossless-delay constraint is met by sending $S[i]$ over $X[i]$ and $X[i + 1]$. For any burst of length b starting in time slot i, $P[i + b], \dots, P[i + \tau - 1]$ are used to recover $V[i], \dots, V[i + b - 1]$. Then $U[j]$ is recovered using $P[j + \tau]$ for $j \in \{i, \dots, i + b - 1\}$. As such, $S[i], \dots, S[i + b - 1]$ are recovered by time slots $(i + \tau), \dots, (i + b - 1 + \tau)$ respectively. A complete proof is included in Appendix 4.7.1.

□

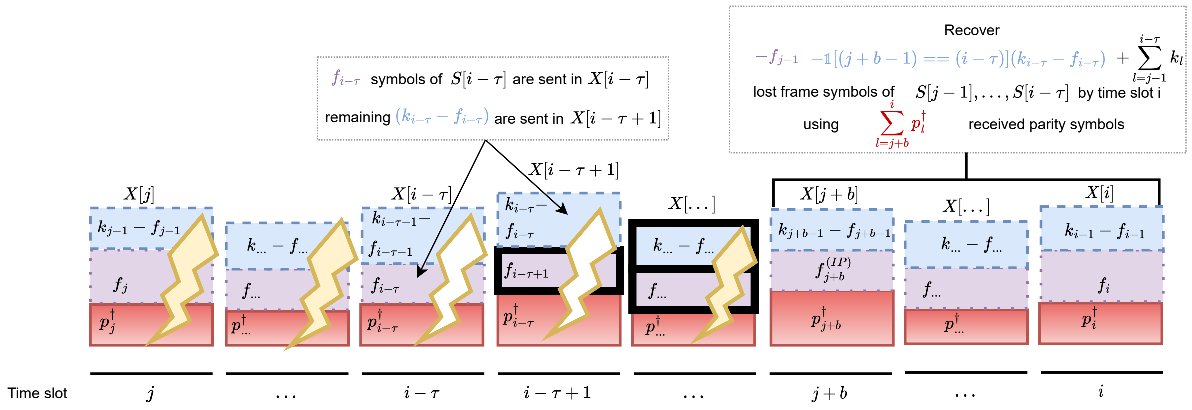

$f_{i-\tau}$ symbols of $S[i-\tau]$ are sent in $X[i-\tau]$
remaining $(k_{i-\tau}-f_{i-\tau})$ are sent in $X[i-\tau+1]$
Recover
$-f_{j-1}$ $-\mathbb{1}[(j+b-1)==(i-\tau)](k_{i-\tau}-f_{i-\tau})$ $+\sum_{l=j-1}^{i-\tau} k_l$
lost frame symbols of $S[j-1],\dots,S[i-\tau]$ by time slot i
using $\sum_{l=j+b}^{i} p_l^\dagger$ received parity symbols
$X[j]$
$X[i-\tau]$
$X[i-\tau+1]$
$X[\dots]$
$X[j+b]$
$X[\dots]$
$X[i]$
Time slot
j
$\dots$
$i-\tau$
$i-\tau+1$
$\dots$
$j+b$
$\dots$
i

Figure 4.4: Illustration of the bound on the number of symbols sent under any streaming code satisfying the delay constraints. For any $i \in \{3\tau,\dots,t\}$ and $j \in \{i-\tau-b+1,\dots,i-\tau+1\}$, $S[j-1],\dots,S[i-\tau]$ are recovered by time slot i when a burst of length b starting in time slot j (shown with lightning bolts), under the relaxation of receiving the lost symbols of $S[i-\tau+1],\dots,S[j+b-1]$ (boxes with thick black outline).

Next, we provide a lower bound on the number of parity symbols sent by any streaming code that satisfies the lossless-delay and worst-case-delay constraints. The bound is illustrated in Figure 4.4

Lemma 12. *Consider any τ, t, b, and any streaming code that satisfies the lossless-delay and worst-case-delay constraints. Suppose for $l \in [t]$, the construction sends $p_l^\dagger$ parity symbols and uses policy f_l. For any $i \in \{3\tau,\dots,t\}$ and $j \in \{i-\tau-b+1,\dots,i-\tau+1\}$, the number of parity symbols satisfies*

$$-f_{j-1} - \mathbb{1}[j+b-1 = i-\tau]\big(k_{i-\tau} - f_{i-\tau}\big) + \sum_{l=j-1}^{i-\tau} k_l \le \sum_{l=j+b}^{i} p_l^\dagger. \tag{4.10}$$

Proof sketch. Suppose $X[j],\dots,X[j+b-1]$ are lost. Due to the worst-case-delay constraint, $S[j-1],\dots,S[i-\tau]$ must be recovered by time slot i. Thus, $\sum_{l=j-1}^{i-\tau} k_l$ frame symbols must be decoded, while f_{j-1} symbols of $S[j-1]$ are received in $X[j-1]$ and, if $X[i-\tau+1]$ is received, $\big(k_{i-\tau} - f_{i-\tau}\big)$ symbols of $S[i-\tau]$ are received. By the independence of frames, the remaining frame symbols received in $X[j+b],\dots,X[i]$ contain no information about $S[j-1],\dots,S[i-\tau]$. Enough parity symbols must be received in $X[j+b],\dots,X[i]$ to recover the lost symbols of $S[j-1],\dots,S[i-\tau]$. A complete proof is included in Appendix 4.7.2. □

We show the rate of the $(\tau, b, t, \langle f_i \mid i \in [t]\rangle)$ –Spreading Variable-sized Generalized MS Code matches that of any streaming code with policy f_i for $i \in [t]$.

Lemma 13. *Consider any τ, t, b, frame-size sequence $k_0,\dots,k_t$, and any streaming code with policy f_i for $i \in [t]$ that satisfies the lossless-delay and worst-case-delay constraints. Then the streaming codes rate is no higher than the $(\tau, b, t, \langle f_i \mid i \in [t]\rangle)$ –Spreading Variable-sized Generalized MS Code .*

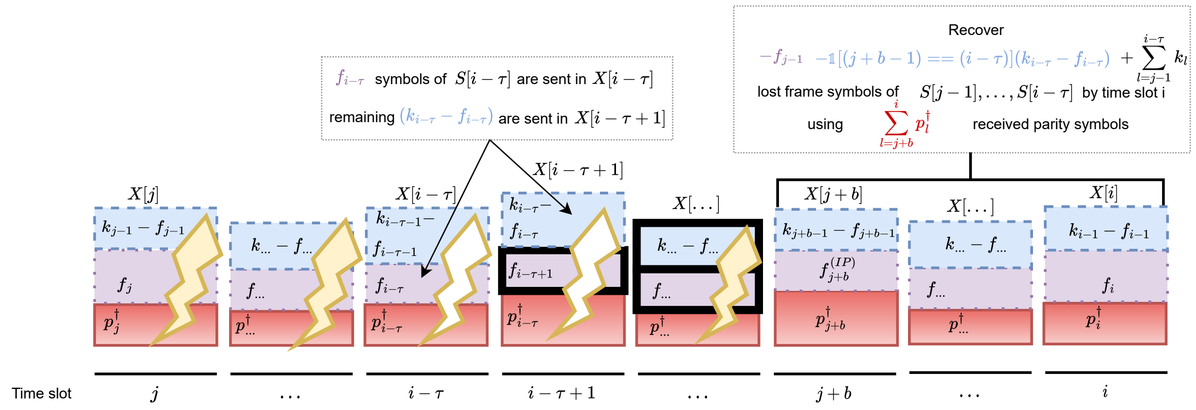


Figure 4.5: Modeling the transmission and constraints using the variables of the integer program. For any $i \in \{3\tau, \dots, t-\tau\}$ and burst (lightning bolts) of length b starting in $j \in \{i-\tau-b+1, \dots, i-\tau+1\}$, $S[j-1], \dots, S[i-\tau]$, are recovered by time slot j under the relaxation of receiving the lost symbols of $S[i-\tau+1], \dots, S[j+b-1]$ (boxes with thick black outline).

Proof sketch. Under the $(\tau, b, t, \langle f_i \mid i \in [t]\rangle)$ –Spreading Variable-sized Generalized MS Code , $\sum_{l=0}^{t}(k_l+p_l)$ symbols are sent. Consider any streaming code that satisfies the lossless-delay and worst-case-delay constraints , and, for each $i \in [t]$, employs policy f_i and sends $p_i^\dagger$ parity symbols. The code sends $\sum_{l=0}^{t}\left(k_l+p_l^\dagger\right)$ symbols.

We show by induction on $i = 0, \dots, t$ that $\sum_{l=0}^{i} p_l \le \sum_{l=0}^{i} p_l^\dagger$. The base case holds for $l < 3\tau$ because $p_0 = 0, \dots, p_{3\tau-1} = 0$. In the inductive hypothesis, for all $j < i$:

$$\sum_{l=0}^{j} p_l \le \sum_{l=0}^{j} p_l^\dagger. \tag{4.11}$$

The inductive step for $i \ge 3\tau$ holds when $p_i \le p_i^\dagger$. Otherwise, there is a burst starting in $j_* \in \{i-\tau-b+1, \dots, i-\tau+1\}$ so that the number of parity symbols sent over $X[j_*+b], \dots, X[i]$ (i.e., $\sum_{l=j_*+b}^{i} p_l^\dagger$) is at least $\sum_{l=j_*+b}^{i} p_l$. Combining this with Eq. (4.11) (i.e., $\sum_{l=0}^{j_*+b-1} p_l \le \sum_{l=0}^{j_*+b-1} p_l^\dagger$) concludes the proof. A complete proof is included in Appendix 4.7.3.

□

4.3 Offline-optimal streaming codes

In this section, we design the first rate-optimal offline construction for the setting of $\tau_L = 1$. We build the construction with two steps for an arbitrary frame-size sequence, $k_0, \dots, k_t$. First, we design an integer program (IP) to use constraints to model satisfying the lossless-delay and Lemma 12. The IP determines the optimal policy for each time slot: $\langle f_i \mid i \in [t]\rangle$, as is illustrated in Figure 4.5. Second, we employ the $(\tau, b, t, \langle f_i \mid i \in [t]\rangle)$ –Spreading Variable-sized Generalized MS Code given the polices. The objective function of the integer program is to minimize the total number of parity symbols transmitted, which maximizes the rate.

Next, we introduce Algorithm 3 to determine an optimal policy, f_i, for each time slot $i \in [t]$ and then verify that the $(\tau, b, t, \langle f_i \mid i \in [t]\rangle)$ −Spreading Variable-sized Generalized MS Code is rate optimal.

Algorithm 3 Computes $\langle f_i \mid i \in [t]\rangle$ for which the $(\tau, b, t, \langle f_i \mid i \in [t]\rangle)$ −Spreading Variable-sized Generalized MS Code matches the offline-optimal-rate.

```
Input: $(\tau, b, t, k_0, \ldots, k_t)$
Minimize $\sum_{i=0}^{t} p_i^{(IP)}$ subject to
    • $\forall i \in [t]$, $f_i^{(IP)} \geq 0, f_i^{(IP)} \leq k_i, p_i^{(IP)} \geq 0$.
    • $\forall i \in \{3\tau, \ldots, t-\tau\}, j \in \{i-\tau-b+1, \ldots, i-\tau+1\}$,
        $-f_{j-1}^{(IP)} - \mathbb{1}[j+b-1 = i-\tau]\left(k_{i-\tau} - f_{i-\tau}^{(IP)}\right) + \sum_{l=j-1}^{i-\tau} k_l \leq \sum_{l=j+b}^{i} p_l^{(IP)}$.
Output: $\langle f_i \mid i \in [t]\rangle$
```

Theorem 6. *For any (τ, b, t) and frame-size sequence $k_0, \ldots, k_t$, suppose Algorithm 3 outputs $\langle f_i \mid i \in [t]\rangle$. Then the $(\tau, b, t, \langle f_i \mid i \in [t]\rangle)$ −Spreading Variable-sized Generalized MS Code is rate optimal.*

Proof. Due to Lemma 13, the rate of the $(\tau, b, t, \langle f_i \mid i \in [t]\rangle)$ −Spreading Variable-sized Generalized MS Code is the same as a construction that for $i \in [t]$ employs the policy f_i and sends $p_i^{(IP)}$ parity symbols in $X[i]$. We will show that no coding scheme can send fewer than $\sum_{i=0}^{t} p_i^{(IP)}$ parity symbols.

An arbitrary rate-optimal construction must satisfy the first constraint because for all $i \in [t]$ between 0 and k_i symbols of $S[i]$ are sent in $X[i]$ along with a non-negative number of parity symbols. The construction must satisfy the second constraint due to Lemma 12. Using each policy of this rate-optimal construction along with the number of parity symbols it sends is a valid solution to the integer program. Correctness follows from minimization. □

Although Algorithm 3 applies to the entire frame-size sequence, it is trivial to modify the algorithm to apply to the remainder of a transmission after channel packets $X[0], \ldots, X[l]$ have been sent for some $l \in [t]$. This involves adding constraints for all $j \in [l]$ (a) $f_j^{(IP)} = f_j$ and (b) $p_j^{(IP)} = p_j$. We call the modified algorithm "Algorithm 3.A."

Corollary 2. *For any (τ, b, t), frame-size sequence $k_0, \ldots, k_t$, and $l \in [t-\tau]$, suppose that for all $j \in [l]$, policy f_j was used and p_j parity symbols were sent in $X[j]$, and Algorithm 3.A outputs $\langle f_i \mid i \in [t]\rangle$. Then the $(\tau, b, t, \langle f_i \mid i \in [t]\rangle)$ −Spreading Variable-sized Generalized MS Code attains the best possible rate given the prior transmission of $X[0], \ldots, X[l]$.*

4.4 Learning-based online streaming codes

We now present an online code construction, dubbed the "(τ, b, t)−Spread ML Code," whose expected rate is within ϵ of the online-optimal-rate. The construction uses a learning-based

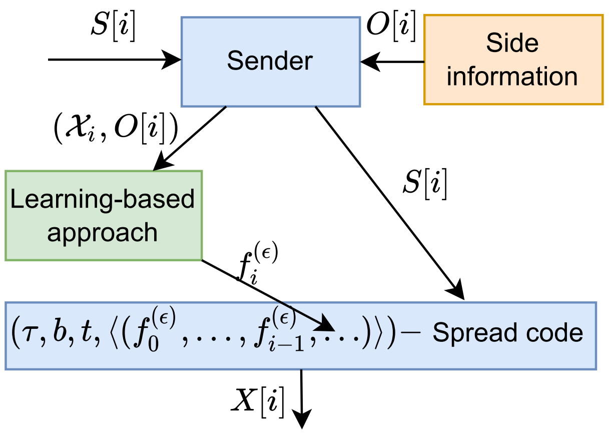


Figure 4.6: Illustration of the $(\tau, b, t)-$Spread ML Code. A learning-based approach is used to determine a policy, $f_i^{(\epsilon)}$, during the ith time slot, which is then used by the $\left(\tau, b, t, \left\langle f_i^{(\epsilon)} \mid i \in [t] \right\rangle\right)$ $-$Spreading Variable-sized Generalized MS Code .

approach to specify the policy of spreading the symbols of $S[i]$, denoted $f_i^{(\epsilon)}$, for each $i \in [t]$, and then applies the $\left(\tau, b, t, \left\langle f_i^{(\epsilon)} \mid i \in [t] \right\rangle\right)$ $-$Spreading Variable-sized Generalized MS Code , as is shown at a high level in Figure 4.6.

To determine how to spread frame symbols, we use the side information of $\mathcal{S}$ samples the distribution of the sizes of the future frames. Recall that this quantity was defined in Equation 4.1 as $\langle (k_{i+1}^{(j)}, \dots, k_t^{(j)}) \sim D_{k_0,\dots,k_i} \mid j \in \{0, \dots, \mathcal{S}-1\} \rangle$. We use a similar technique to empirical risk minimization over the $\mathcal{S}$ samples to set $f_i^{(\epsilon)}$ to the value leading to lowest expected number of symbols being sent by a rate-optimal offline code. Specifically, for any $i = 0, \dots, t$, $j \in [\mathcal{S}-1]$ and $l \in [k_i]$, let

$$z_{i,j,l} = \sum_{r=i}^{t} p_r^{(IP)},$$

where $p_i^{(IP)}, \dots, p_t^{(IP)}$ are variables used by the IP of Algorithm 3.A given $X[0], \dots, X[i-1]$, and $f_i^{(IP)} = l$. Then

$$f_i^{(\epsilon)} = \arg\min_{l \in [k_i]} \frac{1}{\mathcal{S}-1} \sum_{j \in [\mathcal{S}-1]} z_{i,j,l}. \tag{4.12}$$

We demonstrate how $f_i^{(\epsilon)}$ is defined in Figure 4.7.

The key observation to interpret the choice of $f_i^{(\epsilon)}$ is that the number of parity symbols sent corresponding to frame $S[i]$, namely $p_{i+\tau}$, is monotonically non-decreasing as $f_i^{(\epsilon)}$ increases. Thus, smaller values of $f_i^{(\epsilon)}$ lead to smaller values of $p_{i+\tau}$, which exploits the parity symbols already sent before time slot $(i+\tau)$. This strategy is effective when the next several frames are likely small. Therefore, a small $P[i+\tau]$ suffices to ensure that the next several frames are recovered when some of $X[i+2], \dots, X[i+\tau-1]$ are lost. In contrast, larger values of $f_i^{(\epsilon)}$

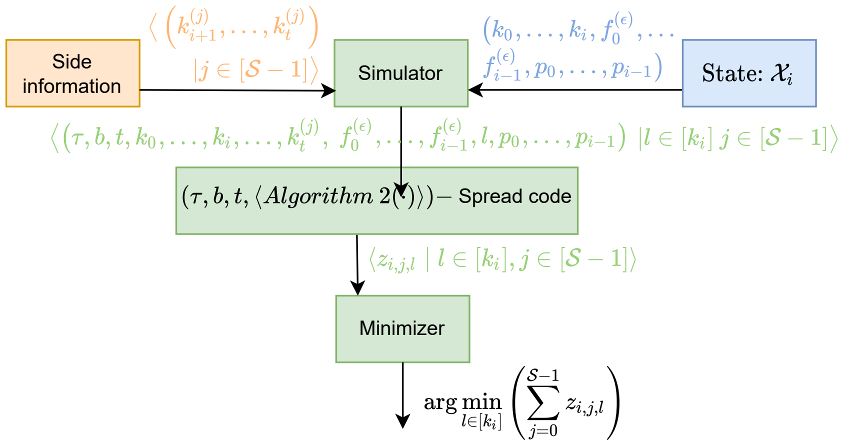

Side information
$\langle \left(k_{i+1}^{(j)}, \ldots, k_t^{(j)}\right) \mid j \in [\mathcal{S}-1]\rangle$
Simulator
$\left(k_0, \ldots, k_i, f_0^{(\epsilon)}, \ldots f_{i-1}^{(\epsilon)}, p_0, \ldots, p_{i-1}\right)$
State: $\mathcal{X}_i$
$\langle\left(\tau, b, t, k_0, \ldots, k_i, \ldots, k_t^{(j)}, f_0^{(\epsilon)}, \ldots, f_{i-1}^{(\epsilon)}, l, p_0, \ldots, p_{i-1}\right) \mid l \in [k_i]\ j \in [\mathcal{S}-1]\rangle$
$(\tau, b, t, \langle Algorithm\ 2(\cdot)\rangle)-$ Spread code
$\langle z_{i,j,l} \mid l \in [k_i], j \in [\mathcal{S}-1]\rangle$
Minimizer
$\arg\min_{l \in [k_i]} \left(\sum_{j=0}^{\mathcal{S}-1} z_{i,j,l}\right)$

Figure 4.7: Illustration of the learning-based approach (green) to determine how to spread frame symbols.

promote larger values of $p_{i+\tau}$, which is suitable when the next several frames are likely to be large. Hence, a large $P[i+\tau]$ will not go to waste even if a burst starts after receiving $S[i]$.

To show that the $(\tau, b, t)-$Spread ML Code is approximately rate optimal, we analyze the number of extra symbols it sends compared to an optimal scheme as follows.

Definition 3 (Regret)**.** *The regret, $\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right)$, for the frame-size sequence $k_0, \ldots, k_t$ is the number of extra symbols sent under Algorithm 3.A when $f_i^{(\epsilon)}$ is used compared to the best offline policy, f_i'.*

Compared to an optimal offline scheme, $\sum_{i=0}^{t} \mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right)$ extra symbols are sent; this is shown next for completeness.

Lemma 14. *Consider the frame-size sequence $k_0, \ldots, k_t$. The $(\tau, b, t)-$Spread ML Code transmits $\sum_{i=0}^{t} \mathcal{R}_{k_i,\ldots,k_t}(f_i)$ more symbols than a scheme meeting the offline-optimal-rate.*

Proof. One can sequentially improve the $(\tau, b, t)-$Spread ML Code for $i = t-2\tau, \ldots, 2\tau$. To do so, one switches $f_j^{(\epsilon)}$ for $j \in \{i, \ldots, t\}$ to the one computed by Algorithm 3.A given $f_0^{(\epsilon)}, \ldots, f_{i-1}^{(\epsilon)}, p_0, \ldots, p_{i-1}$. For each value of i, the improvement in total number of symbols sent is $\mathcal{R}_{k_i,\ldots,k_t}(f_i)$ by Definition 3. After reaching $i = 2\tau$, the output is simply Algorithm 3.A, as $k_0 = 0, \ldots, k_{2\tau-1} = 0$. □

Next, we bound the expected regret of the $(\tau, b, t)-$Spread ML Code from spreading frame symbols for any time slot.

Lemma 15. *For any (τ, b, t), $\mathcal{S} \geq \sqrt{\ln(\frac{8m^2}{\epsilon})\frac{2\sqrt{2}m^3}{\epsilon}}$ samples from the side information, where m is the maximum size of a frame, $i \in [t]$, $k_0, \ldots, k_{i-1}$, and $f_i' \in [k_i]$,*

$$\mathrm{E}_{k_{i+1},\ldots,k_t}\left[\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right) - \mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right)\right] \leq \epsilon. \tag{4.13}$$

Proof sketch. If $k_i = 0$, $f_i = f_i^{(\epsilon)}$. Otherwise, replicating $f_i^{(\epsilon)}$ symbols ensures $\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right) \leq m$. Consequently,

$$\mathrm{E}_{k_{i+1},\ldots,k_t}\left[\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right)\right] \leq m$$
$$\mathrm{Var}_{k_{i+1},\ldots,k_t}\left(\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right)\right) \leq m^2.$$

At a high level, the empirical mean over the $\mathcal{S}$ samples from the side information accurately approximates the expected regret for each policy (Hoeffding bound [45]). The best policy over the samples is suitable. A full proof is shown in Appendix 4.7.4.

□

Finally, we show that the expected online-optimal-rate, denoted as "$R^{(\mathrm{E},Opt)}$," is within ϵ of the expected rate of the $(\tau, b, t)-$Spread ML Code, denoted as

$$R^{(\mathrm{E})} = \mathrm{E}_{k_0,\ldots,k_t}\left[\frac{\sum_{i=0}^{t} k_i}{\sum_{i=0}^{t} k_i + p_i}\right]. \tag{4.14}$$

Theorem 7. *For any (τ, b, t) and for $\mathcal{S} \geq \sqrt{\ln(\frac{8m^2}{\epsilon})\frac{2\sqrt{2}m^3}{\epsilon}}$ samples from the side information, where m is the maximum size of a frame, $(R^{(\mathrm{E},Opt)} - R^{(\mathrm{E})}) < \epsilon$.*

Proof sketch. By Lemma 13, there exists some online $(\tau, b, t, \langle f_i' \mid i \in [t]\rangle)-$Spreading Variable-sized Generalized MS Code with optimal expected rate. In expectation, the $(\tau, b, t)-$Spread ML Code sends at most

$$\sum_{i=0}^{t} \mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right) - \mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right) \leq \epsilon \sum_{i=0}^{t} \mathbb{1}[k_i > 0],$$

more symbols than the optimal code does, leading to $(R^{(\mathrm{E},Opt)} - R^{(\mathrm{E})}) < \epsilon$. A full proof is shown in Appendix 4.7.5.

□

We have designed an online code that uses a black box algorithm to determine how to spread frame symbols and bounded how close the rate is to optimal based on the regret due to the choices of how to spread. To show that the code is approximately rate optimal, we presented an explicit learning-based approach of leveraging samples to the distribution of the sizes of future frames to spread frame symbols (i.e., Equation 4.12) and showed in Lemma 15 that it has a sufficiently small expected regret. More generally, any criteria with a sufficiently small expected regret could be used, leading to the following result.

Corollary 3. *Theorem 7 holds when any criteria for spreading frame symbols replaces Equation 4.12 if the criteria satisfies Equation 4.13 for all $i \in [t]$, $k_0, \ldots, k_t$, and $f_i' \in [k_i]$.*

4.5 Optimality for non-systematic constructions

So far this work has made two assumptions. First, encoding is a function of all transmitted frame symbols (i.e., encoding during time slot i is independent of the frame symbols of $S[i]$ sent in $X[i+1]$). Second, all constructions are systematic. The streaming codes presented in this work satisfy these two assumptions. We have already shown that among codes satisfying these two assumptions for any $\tau, b,$ and frame-size sequence $k_0, \ldots, k_t$ that (a) the $(\tau, b, t, \langle f_i \mid i \in [t] \rangle)-$Spreading Variable-sized Generalized MS Code is rate-optimal for any given policies and (b) the $(\tau, b, t)-$Spread ML Code is approximately rate optimal. This section extends the converse results to apply without the two assumptions.

Removing the first assumption involves redefining the encoding for $i \geq \tau$ as $X[i] =$

$$Enc(\mathcal{X}_i, S[i-\tau], \ldots, S[i], O[i]). \tag{4.15}$$

Hence, symbols of $X[i]$ can be functions of all symbols of $S[i]$ even if some of them are not sent until $X[i+1]$.[2] Throughout this section, let $e\ Uniform(\{1, \ldots, |\mathbb{F}|\})$ and $\mathcal{H}$ be the Shannon entropy. For any $i \in [t]$,

$$\mathcal{H}(e)k_i = \mathcal{H}\left(S[i]\right) \tag{4.16}$$
$$\mathcal{H}(e)n_i \geq \mathcal{H}\left(X[i]\right). \tag{4.17}$$

Without the second assumption, the notion of a policy must be adjusted to apply to non-systematic constructions, as is done next.

Definition 4 (Non-systematic policy)**.** *The non-systematic policy of a construction for any $i \in [t]$ and state $\mathcal{X}_i$ is the amount of information about $S[i]$ sent in channel packet $X[i]$. It is denoted as $\mathcal{F}_i^{\mathcal{H}}(\mathcal{X}_i) = \mathcal{H}\left(X[i] | S[0], \ldots, S[i-1]\right) / \mathcal{H}(e)$. For clarity of notation, $f_i^{\mathcal{H}}$ will represent $\mathcal{F}_i^{\mathcal{H}}(\mathcal{X}_i)$.*

By definition, $f_i^{\mathcal{H}}$ ranges in value from 0 to k_i.

The proofs in this section will refer to many different frames and channel packets, necessitating additional notation. For any frame-size sequence $k_0, \ldots, k_t$, any $i \leq j \in [t]$, and $Z \in [S, X]$, $Z[i:j] \triangleq Z[i], \ldots, Z[j]$.

Next, Lemma 16 provides a bound on the number of parity symbols sent by all streaming codes.

Lemma 16. *Consider any τ, t, b, and any streaming code satisfying the lossless-delay and worst-case-delay constraints. Suppose for $i \in [t]$, the construction sends n_i symbols and uses policy $f_i^{\mathcal{H}}$.[3] For any $i \in [t] \setminus \{0\}$,*

$$n_i \geq \left\lceil k_{i-1} - f_{i-1}^{\dagger,\mathcal{H}} + f_i^{\dagger,\mathcal{H}} \right\rceil.$$

Proof sketch. At a high level, we combine Equations 4.15 and 4.44, Definition 4, and the lossless-delay constraint to show that the amount of information that $X[i]$ contains about $S[i-1]$ is at least $(k_{i-1} - f_{i-1}^{\dagger,\mathcal{H}})$, and the amount of information bits $X[i]$ contains about $S[i]$ is at least $f_i^{\dagger,\mathcal{H}}$.

[2]No change is needed for $i < \tau$, as k_i is known to be 0 by the receiver.

[3]For the boundary condition, $n_0 \triangleq k_0 = 0$.

This requires that $n_i \geq (k_{i-1} - f_{i-1}^{\dagger,\mathcal{H}} + f_i^{\dagger,\mathcal{H}})$. In addition, n_i is an integer, concluding the result. A full proof is included in Appendix 4.7.6.

□

Similar to Lemma 12 for systematic streaming codes, Lemma 17 provides a lower bound on the number of parity symbols sent by any streaming code.

Lemma 17. *Consider any τ, t, b, and any streaming code satisfying the lossless-delay and worst-case-delay constraints. Suppose that for $i \in [t]$, the construction sends n_i symbols and uses policy $f_i^{\mathcal{H}}$, and let $p_i^{(\dagger,\mathcal{H})} \triangleq n_i - \left(k_{i-1} - f_{i-1}^{\dagger,\mathcal{H}} + f_i^{\dagger,\mathcal{H}}\right) \geq 0$. For any $j \in \{3\tau, \ldots, t\}$ and $i \in \{j-\tau-b+1, \ldots, j-\tau+1\}$,*

$$-\left\lceil f_{i-1}^{\mathcal{H}} \right\rceil - \mathbb{1}[i = j-\tau-b+1]\left(k_{j-\tau} - f_{j-\tau}^{\mathcal{H}}\right) + \sum_{l=i-1}^{j-\tau} k_l \leq \sum_{l=i+b}^{j} p_l^{(\dagger,\mathcal{H})}. \tag{4.18}$$

Proof sketch. At a high level, the proof follows from the lossless-delay constraint and the worst-case-delay constraint, similar to the proof of Lemma 12. A complete proof is included in Appendix 4.7.7.

□

Next, we show the rate of the $\left(\tau, b, t, \left\langle \left\lceil f_i^{\mathcal{H}} \right\rceil \mid i \in [t] \right\rangle\right)-$Spreading Variable-sized Generalized MS Code matches that of any streaming code with policy $f_i^{\mathcal{H}}$ for $i \in [t]$ with one extra symbol per time slot.

Lemma 18. *For any frame-size sequence $k_0, \ldots, k_t$, at most $\sum_{i=0}^{t-\tau} \mathbb{1}[k_i > 0]$ extra symbols are sent under the $\left(\tau, b, t, \left\langle \left\lceil f_i^{\mathcal{H}} \right\rceil \mid i \in [t] \right\rangle\right)-$Spreading Variable-sized Generalized MS Code than any streaming code with policy $f_i^{\mathcal{H}}$ for $i \in [t]$ satisfying the lossless-delay and worst-case-delay constraints.*

Proof sketch. At a high level, the proof follows by induction on the cumulative number of symbols sent by each time slot similar to the proof of Lemma 13. A complete proof is included in Appendix 4.7.8.

□

The results of this section extend the previous results to remove the assumptions of (a) $X[i]$ is not a function of the symbols of $S[i]$ sent in $X[i+1]$, and (b) systematic encoding, as is noted below.

Corollary 4. *Theorem 6, Corollary 2, and Theorem 7 hold with the modification of one extra symbol being sent per time slot where the frame is of a non-zero size even for non-systematic schemes where each $X[i]$ for $i \in [t]$ is an arbitrary function of $S[0:i]$.*

Proof. Replacing Lemma 13 with Lemma 18 and replacing Lemma 12 with Lemma 17 extends the proof of Theorem 6. Strengthening Corollary 2, Lemmas 15 and 14, and Theorem 7 is immediate. □

This section showed for any ϵ given sufficiently many $\mathcal{S}$ samples in the side information that the $(\tau, b, t)-$Spread ML Code has an expected rate within ϵ of the online-optimal-rate for arbitrary non-systematic online streaming codes.

4.6 LP relaxation is almost optimal

This section addresses the bottleneck to the complexity of the $(\tau, b, t)-$Spread ML Code: solving an integer program. Although solving integer programs is not tractable, a standard solution is to consider the linear program relaxation of the integer program, which runs in polynomial time. Let "Algorithm 3.B" denote the result of changing Algorithm 3 as follows: (a) replace $p^{(IP)}, p^{(IP,1)}, p^{(IP,2)}$, and $f^{(IP)}$ with $p^{(LP)}, p^{(LP,1)}, p^{(LP,2)}$, and $f^{(LP)}$, (b) relax to use real values rather than integral values, and (c) add the constraint for all $i \in [t-\tau]$ where $k_i = 0$ that $p_{i+\tau}^{(LP)} = 0$. We show that Algorithm 3.B runs in polynomial time with respect to $t\tau$ while nearly matching the offline-optimal-rate.

Lemma 19. *Suppose Algorithm 3.B for frame-size sequence $k_0, \ldots, k_t$ outputs $\left\langle f_i^{(LP)} \mid i \in [t] \right\rangle$, then the $\left(\tau, b, t, \left\langle \left\lceil f_i^{(LP)} \right\rceil \mid i \in [t] \right\rangle\right)-$Spreading Variable-sized Generalized MS Code sends at most $\sum_{i=0}^{t} \mathbb{1}[k_i > 0]$ more symbols than an offline rate-optimal code.*

Proof. Because $X[i]$ is a function of $S[0:i]$, we know $\mathcal{H}\left(X[i]|S[0], \ldots, S[i-1]\right) \leq \mathcal{H}(e)k_i$. Therefore, the first constraint is satisfied by any construction. Due to Lemma 17 (and Equation 4.60), all constructions that satisfy the worst-case-delay and lossless-delay constraints must satisfy the second constraint. The constraint that if $k_i = 0$ that $p_{i+\tau}^{(LP)} = 0$ for $i \in [t-\tau]$ holds without loss of generality; take any solution without this constraint. For $i = 0, \ldots, t-\tau$, if $k_i = 0$ and $p_{i+\tau}^{(LP)} = \gamma > 0$, simply increase $p_{i+\tau+1}^{(LP)}$ by γ and set $p_{i+\tau}^{(LP)}$ to equal 0. The objective function is unchanged and all constraints are still satisfied. Therefore, $\sum_{l=0}^{t} k_i + p_i^{(LP)}$ is at most the total number of symbols sent by a construction that matches the offline-optimal-rate.

For $i \in [t]$, let us consider

$$\begin{aligned} p_i^{(IP)} &= \left\lceil p_i^{(LP)} \right\rceil \\ f_i^{(IP)} &= \left\lceil f_i^{(LP)} \right\rceil . \end{aligned}$$

All constraints of the IP from Algorithm 3 are satisfied and only integral values are used. The change has increased the objective function by at most $\sum_{i=0}^{t} \mathbb{1}[k_i > 0]$ because (a) if $k_i = 0$ then $p_{i+\tau}^{(LP)} = f_i^{(LP)} = 0$ and (b) for all i where $k_i > 0$, $p_{i+\tau}^{(LP)}$ is increased by at most 1. Finally, we note that the objective function for Algorithm 3.B at these inputs is at most the value of Algorithm 3 at the same inputs. Thus, we have obtained a solution to the IP from Algorithm 3 that involves an extra $\sum_{i=0}^{t} \mathbb{1}[k_i > 0]$ value in the objective function. By Lemma 13, the total number of symbols sent under the proposed construction is $\sum_{i=0}^{t} n_i \leq \sum_{i=0}^{t} k_i + p_i^{(IP)}$. Combining this with the fact that at least $\sum_{i=0}^{t} k_i + p_i^{(LP)} \geq \sum_{i=0}^{t} k_i + p_i^{(IP)} - \mathbb{1}[k_i > 0]$ symbols must be sent by any scheme satisfying the lossless-delay and worst-case-delay constraints concludes the proof.

□

While Algorithm 3.B applies to the entire frame-size sequence, it is trivial to modify the algorithm to apply to the remainder of a transmission after channel packets $X[0], \ldots, X[i]$ have been sent for some $i \in [t]$. This involves adding constraints that for all $j \in [i]$ (a) $f_j^{(LP)} = f_j$ and (b) $p_j^{(LP)} = p_j$. We call the modified algorithm "Algorithm 3.C."

Corollary 5. *For any (τ, b, t) and frame-size sequence $k_0, \ldots, k_t$, $i \in [t]$, suppose channel packets $X[0], \ldots, X[i]$ have been sent such that for $j \in [i]$ policy f_j was used and Algorithm 3.C outputs $\langle f_i \mid i \in [t] \rangle$. Then the $(\tau, b, t, \langle f_i \mid i \in [t] \rangle)-$Spreading Variable-sized Generalized MS Code sends at most $\sum_{l=i+1}^{t} \mathbb{1}[k_l > 0]$ more symbols than a construction meeting the best possible rate given the prior transmission of $X[0], \ldots, X[i]$.*

One can substitute Algorithm 3.C for Algorithm 3.A in the $(\tau, b, t)-$Spread ML Code; we call the resulting code the "$(\tau, b, t)-$LP-Spread ML Code."

Remark 9. *Algorithm 3.C and Algorithm 3.B run in poly$(t\tau)$ time.*

We define a relaxed notion of regret.

Definition 5 (Relaxed-regret)**.** *The relaxed-regret, $\mathcal{R}^{(LP)}_{k_i,\ldots,k_t}(f_i^{(\epsilon)})$, for the frame-size sequence $k_0, \ldots, k_t$ is the number of extra symbols sent under Algorithm 3.C when $f_i^{(\epsilon)}$ is used compared to the best offline policy, $f_i' \in [k_i]$.*

We will show in Lemma 20 that the $(\tau, b, t)-$LP-Spread ML Code sends at most $\sum_{i=0}^{t} \mathbb{1}[k_i > 0]$ more symbols than the $(\tau, b, t)-$Spread ML Code.

Lemma 20. *For frame-size sequence $k_0, \ldots, k_t$, the $(\tau, b, t)-$Spread ML Code transmits $\leq \sum_{i=0}^{t} \mathbb{1}[k_i > 0] + \sum_{i=0}^{t} \mathcal{R}^{(LP)}_{k_i,\ldots,k_t}(f_i)$ more symbols than a scheme that meets offline-optimal-rate.*

Proof. Starting from the output of Algorithm 3.C, we can sequentially change the coding scheme for $i = 2\tau, \ldots, t - \tau$ by setting f_i to $f_i^{(\epsilon)}$. Each change leads to exactly $\mathcal{R}^{(LP)}_{k_i,\ldots,k_t}(f_i^{(\epsilon)})$ additional symbols being sent. After all changes, the $(\tau, b, t)-$Spread ML Code is used and $\sum_{i=0}^{t} \mathcal{R}^{(LP)} k_i, \ldots, k_t(f_i^{(\epsilon)})$ more "symbols" sent where non-integral numbers of symbols may be sent. Each time we take $\left\lceil f_i^{(\epsilon)} \right\rceil$ for $i \in [t]$, the objective function of Algorithm 3.C increases by at most one by the proof of Lemma 19 (i.e., it suffices to increase $p_{i+\tau}^{(LP)}$ by at most 1). When $k_i = 0$, then $f_i^{(\epsilon)} = 0$ already, so the ceiling need not be taken. The total number of extra symbols sent due to this change is at most $\sum_{i=0}^{t} \mathbb{1}[k_i > 0]$.

□

The extra at most one symbol per frame is negligible compared to the sizes of frames, which are typically on the order of thousands of symbols (e.g., bytes).

Corollary 6. *For any (τ, b, t) and for $\mathcal{S} \geq \sqrt{\ln(\frac{8m}{\epsilon})\frac{2\sqrt{2}m^2}{\epsilon}}$ samples from the side information, where m is the maximum size of a frame, the $(\tau, b, t)-$Spread ML Code transmits $\leq \sum_{i=0}^{t} \mathbb{1}[k_i > 0](1+\epsilon)$ more symbols than a scheme meeting the offline-optimal-rate in expectation.*

Proof. From Lemma 20, the number of additional symbols sent is, in expectation, no more than $\sum_{i=0}^{t} \mathbb{1}[k_i > 0] + \sum_{i=0}^{t} \mathcal{R}^{(LP)}_{k_i,\ldots,k_t}(f_i)$. For $i \in [t]$, $\mathrm{E}[\mathcal{R}^{(LP)}_{k_i,\ldots,k_t}(f_i^{(\epsilon)})] \leq \epsilon$ by the same argument as used in Theorem 7. □

4.7 Appendix

4.7.1 Proof of Lemma 11

For any $i \in [t-1]$, $S[i] = \left(S_0^{f'_i-1}[i], S_{f'_i}^{k_i-1}[i]\right)$, where $S_0^{f'_i-1}[i]$ is sent in $X[i]$, and $S_{f'_i}^{k_i-1}[i]$ is sent in $X[i+1]$. For $S[t]$, $k_t = 0$ is known. Thus, the lossless-delay constraint is met.

Next, we show that the worst-case-delay is satisfied for any burst. Satisfaction is immediate if the burst starts after $(t-\tau-b+1)$, since $k_{t-\tau-b} = 0, \dots, k_t = 0$ is known. Otherwise, suppose $X[i], \dots, X[i+b-1]$ are lost for some $i \in [t-\tau-b+1]$. We assume that $i \geq 2\tau$, since $k_0 = 0, \dots, k_{2\tau-1} = 0$ is known, and no symbols are sent in $X[0], \dots, X[2\tau-1]$. Each $P^{(v)}[i+b] = (P[i+b] - U[i+b-\tau]), \dots, P^{(v)}[i+\tau-1] = (P[i+\tau-1] - U[i-1])$ is known.

We show that enough parity symbols are received after the burst by time slot $(i+\tau-1)$ to recover $V[i], \dots, V[i+b-1]$ as follows:

$$f_{i+b-1} + \sum_{j=i}^{i+b-1} \left(k_{j-1} - f'_{j-1}\right) + \sum_{j=i}^{i+b-2} f'_j \leq \sum_{j=i+b}^{i+b+\tau-1} p_j \tag{4.19}$$

$$\sum_{j=i}^{i+b-1} \left(k_{j-1} - f'_{j-1} + f'_j\right) \leq \sum_{j=i+b}^{i+b+\tau-1} p_j \tag{4.20}$$

$$\sum_{j=i}^{i+b-1} v_j + u_j \leq \sum_{j=i+b}^{i+b+\tau-1} p_j \tag{4.21}$$

$$\sum_{j=i}^{i+b-1} v_j \leq \sum_{j=i+b}^{i+\tau-1} p_j, \tag{4.22}$$

where Equation 4.19 follows from Equation 4.4, Equation 4.20 follows from (a) $f'_{i+b-1} = f_{i+b-1}$, or (b) combining $f'_{i+b-1} = p_{i+b-1+\tau}$ with Equation 4.4 to show

$$\sum_{j=i}^{i+b-1} \left(k_{j-1} - f'_{j-1}\right) + \sum_{j=i}^{i+b-2} f'_j = k_{i-1} - f'_{i-1} + \sum_{j=i}^{i+b-2} k_j \leq \sum_{j=i+b}^{i+b+\tau-2} p_j,$$

Equation 4.21 follows from Equations 4.6 and 4.7, and Equation 4.22 follows from Equations 4.6 and 4.8.

Next, we show that $P[i+b], \dots, P[i+\tau-1]$ suffice to recover $V[i], \dots, V[i+b-1]$ Recall that $P^{(v)}[j] = W[j]A_{(j)}$ for $j \in \{i+b, \dots, i+\tau-1\}$, where $W[j]$ contains $V[l]$ in positions

$$I^{(i,j)} = \bigcup_{l \in \{i, \dots, i+b-1\}} \{2m(l \bmod \tau), \dots, (2m(l \bmod \tau) + v_l - 1)\},$$

as defined in Equation 4.9 and illustrated in Figure 4.2. Let $W'[j]$ be the vector of length $2m\tau$ with (a) 0's in positions in $I^{(i,j)}$, (b) $W_r[j]$ for positions $r \in [2m\tau - 1] \setminus I$. The receiver can compute $W'[j]$ and use it to determine $P^*[j] = \left(P^{(v)}[j] - W'[j]A_{(j)}\right)$. Let $l_0 = i, \dots, l_{b-1} =$

$(i + b - 1)$, and $r = \arg\min_{l \in \{i,\dots,i+b-1\}}(l \bmod \tau)$. Let $l'_0 = (i + b), \dots, l_{\tau-b-1} = (i + \tau - 1)$, and $r' = \arg\min_{l \in \{i+b,\dots,i+\tau-1\}}(l \bmod \tau)$. Then

$$\begin{bmatrix} P^*[l'_{r'-(i+b)}]^T \\ \vdots \\ P^*[l'_{\tau-b-1}]^T \\ P^*[l'_0]^T \\ \vdots \\ P^*[l'_{r'-(i+b)-1}]^T \end{bmatrix}^T = \begin{bmatrix} V[l_{(r-i)}]^T \\ \vdots \\ V[l_{b-1}]^T \\ V[l_0]^T \\ \vdots \\ V[l_{(r-i)-1}]^T \end{bmatrix}^T A'_{(i)}, \tag{4.23}$$

where T means transpose, and $A'_{(i)}$ is a submatrix of a Cauchy matrix of dimensions $\big(\sum_{j=i}^{i+b-1} v_j\big) \times \Big(\sum_{j=i+b}^{i+\tau-1} p_j\Big)$. As such, $A'_{(i)}$ is full rank, allowing the receiver to solve for $V[i], \dots, V[i+b-1]$.

Finally, for $j = i, \dots, (i + b - 1)$, the receiver uses the symbols of $V[j], \dots, V[j + \tau - 1]$ to compute $P^{(v)}[j + \tau]$, yielding $U[j] = \big(P[j + \tau] - P^{(v)}[j + \tau]\big)$. As $S[j]$ is sent over $V[j], U[j]$, and $V[j + 1]$, it is recovered by time slot $(j + \tau)$.

4.7.2 Proof of Lemma 12

Suppose $X[j], \dots, X[j+b-1]$ are lost. Due to the worst-case-delay constraint, $S[j-1], \dots, S[i-\tau]$ must be recovered by time slot i. Thus, $\sum_{l=j-1}^{i-\tau} k_l$ symbols need to be decoded. Because the frames are independent, the symbols of $X[0], \dots, X[j-2]$ contain no information about $S[j-1], \dots, S[i-\tau]$. By definition of encoding (that is, Equation 4.2), $X[j-1]$ contains f_{j-1} symbols of $S[j-1]$, no additional information about $S[j-1]$, and no information about $S[j], \dots, S[i-\tau]$. When $(j + b - 1) == (i - \tau)$, $X[i - \tau + 1]$ is received, and its frame symbols include $(k_{i-\tau} - f_{i-\tau})$ symbols of $S[i - \tau]$. The remaining frame symbols of $X[j + b], \dots, X[i]$ correspond to $S[i - \tau + 1], \dots, S[i]$ and cannot be used to recover $S[j - 1], \dots, X[i - \tau]$ (independence of frames). Altogether,

$$-f_{j-1} - \mathbb{1}[j + b - 1 = i - \tau]\big(k_{i-\tau} - f_{i-\tau}\big) + \sum_{l=j-1}^{i-\tau} k_l$$

symbols corresponding to $S[j - 1], \dots, S[i - \tau]$ need to be recovered by time slot i. These symbols can only be recovered using the parity symbols of $X[j + b], \dots, X[i]$, of which there are

$$\sum_{l=j+b}^{i} p_l^{\dagger}.$$

The symbols of frames are drawn uniformly at random from the underlying field. Thus, the total number of parity symbols must match the number of frame symbols to be decoded.

4.7.3 Proof of Lemma 13

Under the $(\tau, b, t, \langle f_i \mid i \in [t]\rangle)$ −Spreading Variable-sized Generalized MS Code , the total number of symbols sent is $\sum_{l=0}^{t}(k_l + p_l)$. Consider any streaming code construction that satisfies the lossless-delay and worst-case-delay constraints, and for each $i \in [t]$, employs policy f_i and sends $p_i^\dagger$ parity symbols. This streaming code construction sends $\sum_{l=0}^{t}\left(k_l + p_l^\dagger\right)$ symbols in total.

We show by induction on $i = 0, \dots, t$ that $\sum_{l=0}^{i} p_l \le \sum_{l=0}^{i} p_l^\dagger$. The base case holds for $j < 3\tau$ because $p_0 = 0, \dots, p_{3\tau-1} = 0$. For the inductive hypothesis, we note for all $j < i$:

$$\sum_{l=0}^{j} p_l \le \sum_{l=0}^{j} p_l^\dagger. \tag{4.24}$$

In the inductive step, consider $i = 3\tau, \dots, t - \tau$. By Equation 4.24, the proof holds if $p_i \le p_i^\dagger$. Otherwise, $p_i > p_i^\dagger \ge 0$. Due to Equation 4.24, we only need to show for $j \le i$ that $\sum_{l=j}^{i} p_l \le \sum_{l=j}^{i} p_l^\dagger$.

Let $i_* = (i - \tau)$. By Equation 4.4, there exists $j_* \in \{i_* - b + 1, \dots, i_* + 1\}$ (specifically, taking j_* as the value of j used to define p_i) such that

$$\sum_{l=j_*+b}^{i} p_l = \tag{4.25}$$

$$\sum_{l=j_*+b}^{i_*+\tau} p_l = \mathbb{1}[j_* + b - 1 \ge i_* + 1]\left(k_{i_*} - f_{i_*}\right) + \mathbb{1}[j_* \le i_*] f_{i_*} + \sum_{l=j_*}^{i_*}(k_{l-1} - f'_{l-1}) + \sum_{l=j_*}^{i_*-1} f'_l \tag{4.26}$$

$$- \mathbb{1}[j_* = i_* + 1] f_{j_*-1} - \mathbb{1}[j_* > (i_* + 1)] f'_{j_*-1}$$

$$= \begin{matrix} - \mathbb{1}[j_* = i_* + 1] f_{j_*-1} - \mathbb{1}[j_* > (i_* + 1)] f'_{j_*-1} \\ - \mathbb{1}[j_* + b - 1 = i_*]\left(k_{i_*} - f_{i_*}\right) + \sum_{l=j_*-1}^{i_*} k_l \end{matrix} \tag{4.27}$$

$$= - f'_{j_*-1} - \mathbb{1}[j_* + b - 1 = i_*]\left(k_{i_*} - f_{i_*}\right) + \sum_{l=j_*-1}^{i_*} k_l \tag{4.28}$$

$$= - f'_{j_*-1} - \mathbb{1}[j_* + b - 1 = i - \tau]\left(k_{i-\tau} - f_{i-\tau}\right) + \sum_{l=j_*-1}^{i-\tau} k_l \tag{4.29}$$

$$\le - f_{j_*-1} - \mathbb{1}[j_* + b - 1 = i - \tau]\left(k_{i-\tau} - f_{i-\tau}\right) + \sum_{l=j_*-1}^{i-\tau} k_l. \tag{4.30}$$

Equation 4.26 follows from the fact that $i_* = (i - \tau)$ and Equation 4.4. Equation 4.27 follows from rearranging terms. Equation 4.28 is immediate if $j_* > i_*$ and otherwise follows from $p_{i_*+\tau} \le f_{i_*}$ (by $j_* > i_*$ and Equation 4.4) leading to $f'_{i_*} = f_{i_*}$ (by Equation 4.5). Equation 4.29 follows from substituting $i_* = (i - \tau)$. Equation 4.30 follows from Equation 4.5.

By Lemma 12,

$$\sum_{l=j_*+b}^{i} p_l^\dagger \ge -f_{j_*-1} - \mathbb{1}[j_* + b - 1 = i - \tau]\big(k_{i-\tau} - f_{i-\tau}\big) + \sum_{l=j_*-1}^{i-\tau} k_l. \tag{4.31}$$

Combining Equations 4.25, 4.30 and 4.31 leads to

$$\sum_{l=j_*+b}^{i} p_l^{\dagger} \geq \sum_{l=j_*+b}^{i} p_l. \tag{4.32}$$

Applying Equation 4.24 (for $j = j_* + b - 1$) to Equation 4.32 leads to

$$\sum_{l=0}^{j_*+b-1} p_l^{\dagger} + \sum_{l=j_*+b-1}^{i} p_l^{\dagger} = \sum_{l=0}^{i} p_l^{\dagger} \geq \sum_{l=0}^{j_*+b-1} p_l + \sum_{l=j_*+b-1}^{i} p_l = \sum_{l=0}^{i} p_l, \tag{4.33}$$

proving the inductive step for $l \in \{3\tau, \dots, t-\tau\}$. Recall that $p_{t-\tau+1} = 0, \dots, p_t = 0$, leading to

$$\sum_{l=0}^{t} p_l^{\dagger} \geq \sum_{l=0}^{t-\tau} p_l^{\dagger} \geq \sum_{l=0}^{t-\tau} p_l = \sum_{l=0}^{t} p_l.$$

The $(\tau, b, t, \langle f_i \mid i \in [t] \rangle)$ $-$Spreading Variable-sized Generalized MS Code sends $\sum_{l=0}^{t}(k_l + p_l)$ symbols, which is no more than the number sent under the alternative construction (i.e., $\sum_{l=0}^{t}(k_l + p_l^{\dagger})$).

4.7.4 Proof of Lemma 15

If $k_i = 0$, $f_i = f_i^{(\epsilon)}$, concluding the proof. Otherwise, the choice of $f_i^{(\epsilon)}$ leads to sending at most k_i extra parity symbols in $X[i+\tau]$ compared to an optimal scheme, so

$$\mathcal{R}_{k_i,\dots,k_t}\left(f_i^{(\epsilon)}\right) \leq m \tag{4.34}$$

and

$$\mathrm{E}_{k_{i+1},\dots,k_t}\left[\mathcal{R}_{k_i,\dots,k_t}\left(f_i^{(\epsilon)}\right)\right] \leq m, \ \mathrm{Var}_{k_{i+1},\dots,k_t}\left(\mathcal{R}_{k_i,\dots,k_t}\left(f_i^{(\epsilon)}\right)\right) \leq m^2.$$

At a high level, we apply the Hoeffding bound [45] to show that the expected regret for each possible policy is well approximated using the empirical mean over the $\mathcal{S}$ samples from the side information. The unlikely event that the expected value deviates greatly from the mean will have negligible impact due to Equation 4.34.

Next, we use $O[i]$ to determine the values of $\mathcal{S}$ random variables, $(z_{i,j,0}, \dots, z_{i,j,\mathcal{S}-1})$, equaling $\mathcal{R}_{k_i,\dots,k_t}(j)$ in distribution. The empirical mean is $\frac{1}{\mathcal{S}}\sum_{l=0}^{\mathcal{S}-1} z_{i,j,l}$. This value is used to estimate $\mathrm{E}_{k_i,\dots,k_t}[\mathcal{R}_{k_i,\dots,k_t}(j)]$. By the Hoeffding bound [45],

$$\left|\frac{1}{\mathcal{S}}\left(\sum_{l=0}^{\mathcal{S}-1} z_{i,j,l}\right) - \mathrm{E}_{k_0,\dots,k_t}[\mathcal{R}_{k_i,\dots,k_t}(j)]\right| < \epsilon_{\dagger}$$

with probability at least

$$\left(1 - 2e^{-\frac{2\mathcal{S}^2(\epsilon_{\dagger})^2}{m^2}}\right) \geq (1-\delta)$$

as long as

$$\begin{aligned}\frac{\delta}{2} &\geq e^{-\frac{2\mathcal{S}^2(\epsilon_\dagger)^2}{m^2}}\\ \frac{2\mathcal{S}^2(\epsilon_\dagger)^2}{m^2} &\geq \ln\left(\frac{2}{\delta}\right)\\ \mathcal{S}^2 &\geq \ln\left(\frac{2}{\delta}\right)\frac{m^2}{2(\epsilon_\dagger)^2}\\ \mathcal{S} &\geq \sqrt{\ln\left(\frac{2}{\delta}\right)}\frac{m}{\sqrt{2}\epsilon_\dagger}.\end{aligned}$$

Using $\epsilon_\dagger = \delta = \frac{\epsilon}{4m^2}$ and applying the union bound over the at most m values of k_i shows with probability $(1 - m\delta)$ for all $j \in [k_i]$,

$$\left|\frac{1}{\mathcal{S}}\left(\sum_{l=0}^{\mathcal{S}-1} z_{i,j,l}\right) - \mathrm{E}_{k_0,\ldots,k_t}\left[\mathcal{R}_{k_i,\ldots,k_t}\right]\left(f_{i,j}^{(\epsilon)}\right)]\right| < \epsilon_\dagger. \tag{4.35}$$

We set $f_i^{(\epsilon)}$ according to Equation 4.12 as

$$f_i^{(\epsilon)} = \arg\min_j \frac{1}{\mathcal{S}}\sum_{l=0}^{\mathcal{S}-1} z_{i,j,l}.$$

With probability $(1 - m\delta)$ Equation 4.35 holds, leading to

$$\begin{aligned}&\mathrm{E}_{k_i,\ldots,k_t}\left[\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right) - \mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right)\right] \leq\\ &\mathrm{E}_{k_i,\ldots,k_t}\left[\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right) - \mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right)\right] + \left(\frac{1}{\mathcal{S}}\left(\sum_{l=0}^{\mathcal{S}-1} z_{i,f_i',l}\right) - \frac{1}{\mathcal{S}}\left(\sum_{l=0}^{\mathcal{S}-1} z_{i,f_i^{(\epsilon)},l}\right)\right) \leq\\ &\left(\mathrm{E}_{k_i,\ldots,k_t}\left[\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right)\right] - \frac{1}{\mathcal{S}}\left(\sum_{l=0}^{\mathcal{S}-1} z_{i,f_i^{(\epsilon)},l}\right)\right) + \left(\left(\frac{1}{\mathcal{S}}\left(\sum_{l=0}^{\mathcal{S}-1} z_{i,f_i',l}\right) - \mathrm{E}_{k_i,\ldots,k_t}\left[\mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right)\right]\right)\right) \leq\\ &\left|\mathrm{E}_{k_i,\ldots,k_t}\left[\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right)\right] - \frac{1}{\mathcal{S}}\left(\sum_{l=0}^{\mathcal{S}-1} z_{i,f_i^{(\epsilon)},l}\right)\right| + \left|\left(\frac{1}{\mathcal{S}}\left(\sum_{l=0}^{\mathcal{S}-1} z_{i,f_i',l}\right) - \mathrm{E}_{k_i,\ldots,k_t}\left[\mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right)\right]\right)\right| \leq\\ &2\epsilon_\dagger,\end{aligned} \tag{4.36}$$

which used the fact that Equation 4.12 led to

$$\frac{1}{\mathcal{S}}\left(\sum_{l=0}^{\mathcal{S}-1} z_{i,f_i',l}\right) \geq \frac{1}{\mathcal{S}}\left(\sum_{l=0}^{\mathcal{S}-1} z_{i,f_i^{(\epsilon)},l}\right).$$

Otherwise, with probability $m\delta$

$$\mathrm{E}_{k_i,\ldots,k_t}\left[\mathcal{R}_{k_i,\ldots,k_t}(f_i^{(\epsilon)}) - \mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right)\right] \leq m. \tag{4.37}$$

Combining Equations 4.36 and 4.37 leads to

$$\mathrm{E}_{k_i,\ldots,k_t}\left[\mathcal{R}_{k_i,\ldots,k_t}(f_i^{(\epsilon)}) - \mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right)\right] \le \delta m^2 + 2\epsilon_\dagger \le \epsilon.$$

4.7.5 Proof of Theorem 7

We consider an online scheme with the optimal expected rate. By Lemma 13, such a scheme of the form $(\tau, b, t, \langle f_i' \mid i \in [t]\rangle)$ −Spreading Variable-sized Generalized MS Code , which must exist. Let n_i be the number of symbols sent in $X[i]$ under the optimal scheme and

$$n_\epsilon = \sum_{i=0}^{t} \mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right) - \mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right) \tag{4.38}$$

be the number of additional symbols sent under the $(\tau, b, t, \langle f_i' \mid i \in [t]\rangle)$ −Spreading Variable-sized Generalized MS Code .

By definition

$$\begin{aligned} R^{(\mathrm{E},Opt)} - R^{(\mathrm{E})} &\le \mathrm{E}_{k_0,\ldots,k_t}\left[\left|\frac{\sum_{i=0}^t k_i}{\sum_{i=0}^t n_i} - \frac{\sum_{i=0}^t k_i}{n_\epsilon + \sum_{i=0}^t n_i}\right|\right] && (4.39)\\ &\le \mathrm{E}_{k_0,\ldots,k_t}\left[\left|\frac{|n_\epsilon|\sum_{i=0}^t k_i}{\left(\sum_{i=0}^t n_i\right)\left(n_\epsilon + \sum_{i=0}^t n_i\right)}\right|\right] \\ &\le \mathrm{E}_{k_0,\ldots,k_t}\left[\frac{|n_\epsilon|}{\sum_{i=0}^t n_i}\right] && (4.40)\\ &\le \mathrm{E}_{k_0,\ldots,k_t}\left[\frac{|n_\epsilon|}{\sum_{i=0}^t \mathbb{1}[k_i>0]}\right], && (4.41)\end{aligned}$$

where Equation 4.40 follows from sending $\sum_{i=0}^t n_i \ge \sum_{i=0}^t k_i$ symbols to satisfy the lossless-delay constraint, and Equation 4.41 follows from $\sum_{i=0}^t k_i \ge \sum_{i=0}^t \mathbb{1}[k_i > 0]$.

If $k_i = 0$, then $\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right) = \mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right) = 0$. Thus, we can simplify Equation 4.38 as

$$n_\epsilon = \sum_{i=0}^{t} \mathbb{1}[k_i>0]\left(\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right) - \mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right)\right). \tag{4.42}$$

Applying Equation 4.42 to Equations 4.39 and 4.41 leads to

$$\begin{aligned} R^{(\mathrm{E},Opt)} - R^{(\mathrm{E})} &\le \mathrm{E}_{k_0,\ldots,k_t}\left[\frac{\sum_{i=0}^t \mathbb{1}[k_i>0]\left(\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right) - \mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right)\right)}{\sum_{i=0}^t \mathbb{1}[k_i>0]}\right] \\ &\le \max_{i\in[t]} \mathrm{E}_{k_0,\ldots,k_t}\left[\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right) - \mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right)\right] \\ &= \max_{i\in[t]} \mathrm{E}_{k_0,\ldots,k_{i-1}}\left[\mathrm{E}_{k_i,\ldots,k_t}\left[\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right) - \mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right)\right]\right]\end{aligned}$$

As such, it suffices to show for all $i \in [t]$ that

$$\mathrm{E}_{k_0,\ldots,k_{i-1}}\left[\mathrm{E}_{k_i,\ldots,k_t}\left[\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right) - \mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right)\right]\right] \le \epsilon. \tag{4.43}$$

Because $\mathcal{S} \ge \sqrt{\ln(\frac{8m^2}{\epsilon})\frac{2\sqrt{2}m^3}{\epsilon}}$, Lemma 15 guarantees for any $k_0, \ldots, k_{i-1}$,

$$\mathrm{E}_{k_i,\ldots,k_t}\left[\mathcal{R}_{k_i,\ldots,k_t}\left(f_i^{(\epsilon)}\right) - \mathcal{R}_{k_i,\ldots,k_t}\left(f_i'\right)\right] \le \epsilon,$$

concluding the proof.

4.7.6 Proof of Lemma 16

By Equation 4.15 and Definition 4

$$\begin{aligned}
\mathcal{H}\left(S[i], X[i] \middle| S[0:i-1]\right) &= \mathcal{H}\left(S[i] \middle| S[0:i-1]\right) + \mathcal{H}\left(X[i] \middle| S[0:i]\right) \\
&= \mathcal{H}(e)k_i \\
\mathcal{H}\left(S[i], X[i] \middle| S[0:i-1]\right) &= \mathcal{H}\left(X[i] \middle| S[0:i-1]\right) + \mathcal{H}\left(S[i] \middle| S[0:i-1], X[i]\right) \\
&= \mathcal{H}(e)f_i^{\mathcal{H}} + \mathcal{H}\left(S[i] \middle| S[0:i-1], X[i]\right) \\
\mathcal{H}\left(S[i] \middle| S[0:i-1], X[i]\right) &= \mathcal{H}(e)(k_i - f_i^{\mathcal{H}}).
\end{aligned} \tag{4.44}$$

Applying Equations 4.15 and 4.44, Definition 4, and the lossless-delay constraint yields

$$\begin{aligned}
\mathcal{H}(e)\sum_{j=0}^{i} k_j \le\ & \mathcal{H}\left(S[0:i]\right) + \mathcal{H}\left(X[0:i] \middle| S[0:i]\right) \\
=& \mathcal{H}\left(S[0:i], X[0:i]\right) \\
\le& \mathcal{H}\left(S[0:i-2]\right) + \mathcal{H}\left(X[i-1] \middle| S[0:i-2]\right) + \\
& \mathcal{H}\left(X[i], S[i], S[i-1] \middle| S[0:i-2], X[i-1]\right) \\
\le& \mathcal{H}(e)\left(f_{i-1}^{\mathcal{H}} + \sum_{j=0}^{i-2} k_j\right) + \\
& \mathcal{H}\left(X[i] \middle| S[0:i-2], X[i-1]\right) + \mathcal{H}\left(S[i] \middle| S[0:i-1], X[i]\right) \\
\le& \mathcal{H}(e)\left(f_{i-1}^{\mathcal{H}} + k_i - f_i^{\mathcal{H}} + \sum_{j=0}^{i-2} k_j\right) + \mathcal{H}\left(X[i]\right)
\end{aligned} \tag{4.45}$$

Combining Equation 4.45 with the fact that $X[i]$ comprises n_i symbols leads to the following.

$$\begin{aligned}
\mathcal{H}(e)\left(k_{i-1} - f_{i-1}^{\mathcal{H}} + f_i^{\mathcal{H}}\right) \le \mathcal{H}\left(X[i]\right) \le \mathcal{H}(e)n_i \\
k_{i-1} - f_{i-1}^{\mathcal{H}} + f_i^{\mathcal{H}} \le n_i.
\end{aligned} \tag{4.46}$$

Finally, n_i is an integer, so it must be at least the ceiling on the left-hand side of Equation 4.46.

4.7.7 Proof of Lemma 17

We analyze the size of the channel packets

$$\begin{aligned}
\mathcal{H}(e)\sum_{l=i+b}^{j} n_l &= \mathcal{H}(e)\sum_{l=i+b}^{j}\left(k_{l-1}-f^{\mathcal{H}}_{l-1}+f^{\mathcal{H}}_{l}+p^{(\dagger,\mathcal{H})}_{l}\right) && (4.47)\\
&\geq \mathcal{H}\left(X[i+b:j]\right) && (4.48)\\
&\geq \mathcal{H}\left(X[i+b:j]\middle|S[0:i-2],X[i-1]\right) && (4.49)\\
&= \mathcal{H}\left(X[i+b:j],S[i-1:j-\tau]\middle|S[0:i-2],X[i-1]\right) && (4.50)\\
&\geq \mathcal{H}\left(S[i-1]\middle|S[0:i-2],X[i-1]\right)+\mathcal{H}\left(S[i:j-\tau]\right)+ && (4.51)\\
&\mathcal{H}\left(X[i+b:j]\middle|S[0:i+b-2+\mathbb{1}[i==j-\tau-b+1]],X[i+b-1]\right),
\end{aligned}$$

where Equations 4.47 and Equation 4.48 follows from Equation 4.17 as well as the values of $f^{\mathcal{H}}_{l}$ and $p^{(\dagger,\mathcal{H})}_{l}$ for $l\in\{i+b,\dots,j\}$, Equation 4.49 follows from conditioning not increasing entropy, Equation 4.50 follows from the worst-case-delay constraint, and Equation 4.51 follows from the chain rule, independence of frames, and conditioning not increasing entropy.

By the lossless-delay constraint,

$$\begin{aligned}
\mathcal{H}\left(S[i+b-1]\middle|S[0:i+b-2+\mathbb{1}[i==j-\tau-b+1]],X[i+b-1:i+b]\right)&=0\\
\mathcal{H}\left(S[i+b:j-1]\middle|S[0:i+b-1],X[i+b:j]\right)&=0
\end{aligned}$$

Thus,

$$\begin{aligned}
&\mathcal{H}\left(X[i+b:j]\middle|S[0:i+b-2+\mathbb{1}[i==j-\tau-b+1]],X[i+b-1]\right)=\\
&\mathcal{H}\left(X[i+b:j],S[i+b-1:j-1]\middle|S[0:i+b-2+\mathbb{1}[i==j-\tau-b+1]],X[i+b-1]\right). && (4.52)
\end{aligned}$$

Combining Equations 4.51, Equation 4.52, 4.15, and 4.44 with Definition 4 leads to

$$\begin{aligned}
&\mathcal{H}\left(S[i-1]|S[0:i-2],X[i-1]\right)+\mathcal{H}\left(S[i:j-\tau]\right)+\\
&\mathbb{1}[i>j-\tau-b+1]\mathcal{H}\left(S[i+b-1]\middle|S[0:i+b-2],X[i+b-1]\right)+\\
&\mathcal{H}\left(S[i+b:j-1]\right)+\mathcal{H}\left(X[j]\middle|S[0:j-1]\right)\\
&\geq \mathcal{H}(e)\left((k_{i-1}-f^{\mathcal{H}}_{i-1})+\mathbb{1}[i>j-\tau-b+1](k_{i+b-1}-f^{\mathcal{H}}_{i+b-1})+f^{\mathcal{H}}_{j}+\sum_{l=i}^{j-\tau}k_l+\sum_{l=i+b}^{j-1}k_l\right). && (4.53)
\end{aligned}$$

To simplify the right-hand side of Equation 4.53,

$$(k_{i-1}-f^{\mathcal{H}}_{i-1})+\sum_{l=i}^{j-\tau}k_l=-f^{\mathcal{H}}_{i-1}+\sum_{l=i-1}^{j-\tau}k_l \quad (4.54)$$

$$f^{\mathcal{H}}_{j}+\sum_{l=i+b}^{j-1}k_l=\sum_{l=i+b}^{j}f^{\mathcal{H}}_{j}+\sum_{l=i+b}^{j-1}(k_l-f^{\mathcal{H}}_{l}) \quad (4.55)$$

$$\mathbb{1}[i>j-\tau-b+1](k_{i+b-1}-f^{\mathcal{H}}_{i+b-1})=\begin{aligned}&(k_{i+b-1}-f^{\mathcal{H}}_{i+b-1})-\\&\mathbb{1}[i==j-\tau-b+1](k_{i+b-1}-f^{\mathcal{H}}_{i+b-1})\end{aligned} \quad (4.56)$$

Thus, the right-hand side of Equation 4.53, is rewritten using Equations 4.54, 4.55, and 4.56 as

$$\mathcal{H}(e)\left((k_{i-1}-f^{\mathcal{H}}_{i-1})+\mathbb{1}[i>j-\tau-b+1](k_{i+b-1}-f^{\mathcal{H}}_{i+b-1})+f^{\mathcal{H}}_j+\sum_{l=i}^{j-\tau}k_l+\sum_{l=i+b}^{j-1}k_l\right)=$$
$$\mathcal{H}(e)\left(-f^{\mathcal{H}}_{i-1}-\mathbb{1}[i==j-\tau-b+1](k_{j-\tau}-f^{\mathcal{H}}_{j-\tau})+\sum_{l=i-1}^{j-\tau}k_l+\sum_{l=i+b}^{j}f^{\mathcal{H}}_l+\sum_{l=i+b-1}^{j-1}(k_l-f^{\mathcal{H}}_l)\right)=$$
$$\mathcal{H}(e)\left(-f^{\mathcal{H}}_{i-1}-\mathbb{1}[i==j-\tau-b+1](k_{j-\tau}-f^{\mathcal{H}}_{j-\tau})+\sum_{l=i-1}^{j-\tau}k_l+\sum_{l=i+b}^{j}f^{\mathcal{H}}_l+\sum_{l=i+b}^{j}(k_{l-1}-f^{\mathcal{H}}_{l-1})\right) \tag{4.57}$$

Combining Equations 4.47 and 4.57 yields

$$\mathcal{H}(e)\sum_{l=i+b}^{j}\left(k_{l-1}-f^{\mathcal{H}}_{l-1}+f^{\mathcal{H}}_l+p^{(\dagger,\mathcal{H})}_l\right)\geq \tag{4.58}$$
$$\mathcal{H}(e)\left(-f^{\mathcal{H}}_{i-1}-\mathbb{1}[i==j-\tau-b+1](k_{j-\tau}-f^{\mathcal{H}}_{j-\tau})+\sum_{l=i-1}^{j-\tau}k_l+\sum_{l=i+b}^{j}(f^{\mathcal{H}}_l+(k_{l-1}-f^{\mathcal{H}}_{l-1}))\right), \tag{4.59}$$

which simplifies to

$$\sum_{l=i+b}^{j}p^{(\dagger,\mathcal{H})}_l\geq -f^{\mathcal{H}}_{i-1}-\mathbb{1}[i==j-\tau-b+1](k_{j-\tau}-f^{\mathcal{H}}_{j-\tau})+\sum_{l=i-1}^{j-\tau}k_l. \tag{4.60}$$

Applying the fact that $-f^{\mathcal{H}}_{i-1}\geq\lceil -f^{\mathcal{H}}_{i-1}\rceil$ to Equation 4.60 leads to

$$\sum_{l=i+b}^{j}p^{(\dagger,\mathcal{H})}_l\geq -\left\lceil f^{\mathcal{H}}_{i-1}\right\rceil-\mathbb{1}[i==j-\tau-b+1](k_{j-\tau}-f^{\mathcal{H}}_{j-\tau})+\sum_{l=i-1}^{j-\tau}k_l \tag{4.61}$$

4.7.8 Proof of Lemma 18

Under the $\left(\tau,b,t,\left\langle\left\lceil f^{\mathcal{H}}_i\right\rceil \mid i\in[t]\right\rangle\right)$ −Spreading Variable-sized Generalized MS Code , a total of $\sum_{i=2\tau}^{t-\tau}(k_i+p_i)$ symbols are sent where p_i is the number of parity symbols sent during time slot i. Consider any streaming code construction which satisfies the lossless-delay and worst-case-delay constraints, and for each $i\in[t]$, employs policy $f^{\mathcal{H}}_i$ and sends $p^{(\dagger,\mathcal{H})}_i$ parity symbols. This streaming code construction sends $\sum_{i=2\tau}^{t-\tau}\left(k_i+p^{(\dagger,\mathcal{H})}_i\right)$ symbols in total.

We show by induction on $j=3\tau,\ldots,(t-\tau)$ that $\sum_{i=0}^{j}p_i+\mathbb{1}[k_i>0]\leq\sum_{i=0}^{j}p^{(\dagger,\mathcal{H})}_i$.

For the base case, consider any $j<3\tau$. Recall that for any $l\in[3\tau-1]$, $p_l=0$. Therefore, $\sum_{i=0}^{j}p_i=0\leq\sum_{i=0}^{j}p^{(\dagger,\mathcal{H})}_i$.

For the inductive hypothesis, for all $l < j$:

$$\sum_{i=0}^{l} p_i + \mathbb{1}[k_i > 0] \leq \sum_{i=0}^{l} p_i^{(\dagger,\mathcal{H})}. \tag{4.62}$$

In the inductive step, let $j \in \{3\tau, \ldots, t-\tau\}$.

Case $p_j \leq p_j^{(\dagger,\mathcal{H})}$:

The number of parity symbols sent by time slot j can be bounded as

$$\sum_{i=0}^{j-1} p_i + \mathbb{1}[k_i > 0] \leq \sum_{i=0}^{j-1} p_i^{(\dagger,\mathcal{H})} \tag{4.63}$$

$$p_j + \sum_{i=0}^{j-1} p_i + \mathbb{1}[k_i > 0] \leq p_j^{(\dagger,\mathcal{H})} + \sum_{i=0}^{j-1} p_i^{(\dagger,\mathcal{H})} \tag{4.64}$$

$$\sum_{i=0}^{j} p_i + \mathbb{1}[k_i > 0] \leq \sum_{i=0}^{j} p_i^{(\dagger,\mathcal{H})}. \tag{4.65}$$

where Equation 4.63 follows from the IH for $l = (j-1)$, and Equation 4.64 follows from the fact that $p_j \leq p_j^{(\dagger,\mathcal{H})}$.

Case $p_j > p_j^{(\dagger,\mathcal{H})}$:

Due to Equation 4.62, it suffices to show for some for an $l < j$ that $\sum_{i=l}^{j} p_i + \mathbb{1}[k_i > 0] \leq \sum_{i=l}^{j} p_i^{(\dagger,\mathcal{H})}$.

By Equation 4.4, for $i' = (j-\tau)$ and for some $j_* \in \{i'-b+1, \ldots, i'+1\}$

$$0 <= p_j \tag{4.66}$$

$$= p_{i'+\tau} \tag{4.67}$$

$$= \mathbb{1}[j_* > i'-b+1]\left(k_{i'} - (f_{i'}^{\mathcal{H}})'\right) + \sum_{l=j_*}^{i'} \left(k_{l-1} - (f_{l-1}^{\mathcal{H}})' + (f_l^{\mathcal{H}})'\right) - \sum_{i=j_*+b}^{i'+\tau-1} p_i \tag{4.68}$$

$$= k_{j_*-1} - (f_{j_*-1}^{\mathcal{H}})' - \mathbb{1}[j_* == i'-b+1]\left(k_{i'} - (f_{i'}^{\mathcal{H}})'\right) + \sum_{l=j_*}^{i'} k_l - \sum_{i=j_*+b}^{i'+\tau-1} p_i. \tag{4.69}$$

Substituting $(j-\tau)$ for i' and rearranging terms leads to

$$\sum_{i=j_*+b}^{j} p_i = \mathbb{1}[j_* > j-\tau-b+1]\left(k_{j-\tau} - (f^{\mathcal{H}}_{j-\tau})'\right) + \sum_{l=j_*}^{j-\tau}\left(k_{l-1} - (f^{\mathcal{H}}_{l-1})' + (f^{\mathcal{H}}_{l})'\right) \tag{4.70}$$

$$= -(f^{\mathcal{H}}_{j_*-1})' - \mathbb{1}[j_* = j-\tau-b+1]\left(k_{j-\tau} - (f^{\mathcal{H}}_{j-\tau})'\right) + \sum_{l=j_*-1}^{j-\tau} k_l \tag{4.71}$$

$$\leq -\left\lceil (f^{\mathcal{H}}_{j_*-1})'\right\rceil - \mathbb{1}[j_* = j-\tau-b+1]\left(k_{j-\tau} - (f^{\mathcal{H}}_{j-\tau})'\right) + \sum_{l=j_*-1}^{j-\tau} k_l \tag{4.72}$$

$$\leq \sum_{i=j_*+b}^{j} p_i^{(\dagger,\mathcal{H})} \tag{4.73}$$

Equation 4.71 follows from rearranging and applying $(\mathbb{1}[j_* > j-\tau-b+1] = 1-\mathbb{1}[j_* = j-\tau-b+1])$. Equation 4.72 follows from the fact that $-\left\lceil (f^{\mathcal{H}}_{j_*-1})'\right\rceil \leq -f^{\mathcal{H}}_{j_*-1}$. Equation 4.73 follows from Lemma 17. Replacing $(f^{\mathcal{H}}_{j-\tau})'$ with $\left\lceil f^{\mathcal{H}}_{j-\tau}\right\rceil'$ adds at most one extra symbol. Therefore,

$$\sum_{i=j_*+b}^{j} p_i + \mathbb{1}[k_i > 0] \leq \sum_{i=j_*+b}^{j} p_i^{(\dagger,\mathcal{H})}.$$

Applying the inductive hypothesis yields

$$\sum_{i=j_*+b}^{j} p_i + \mathbb{1}[k_i > 0] + \sum_{i=0}^{j_*+b-1} p_i + \mathbb{1}[k_i > 0] \leq \sum_{i=j_*+b}^{j} p_i^{(\dagger,\mathcal{H})} + \sum_{i=0}^{j_*+b-1} p_i^{(\dagger,\mathcal{H})} \tag{4.74}$$

$$\sum_{i=0}^{j} p_i + \mathbb{1}[k_i > 0] \leq \sum_{i=0}^{j} p_i^{(\dagger,\mathcal{H})}, \tag{4.75}$$

where Equation 4.74 follows from applying Equation 4.62 (for $l = j_* + b - 1$). To do so requires $(j_* + b - 1) < j$, which holds since $1 = \tau_L \leq (\tau - b)$.

In both cases, the inductive step is proved, concluding the proof.

Chapter 5

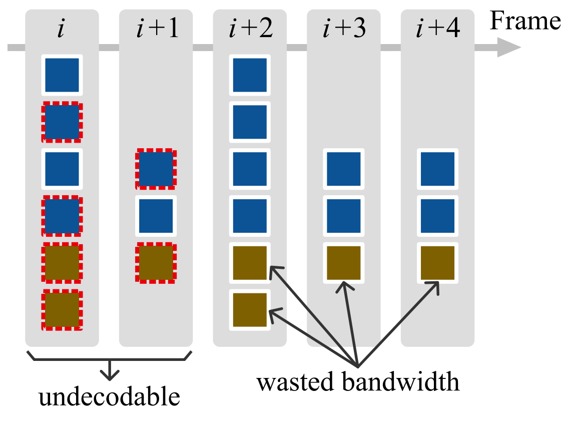
Streaming codes for real-world videoconferencing

Recall that this thesis aims to improve the QoE of real-time streaming applications like videoconferencing. As steps to attain this goal, Chapters 2, 3, and 4 involved developing streaming codes for a new theoretical model reflecting the requirement of applications like videoconferencing of sending frames of variable sizes. However, two main challenges prohibit directly applying these streaming codes to videoconferencing applications. First, significant gaps between the model and practical systems still remain, and they render the streaming codes incompatible with videoconferencing applications. Second, streaming codes' effectiveness for videoconferencing is untested on large-scale real-world traces. Hence, streaming codes' potential to improve the QoE of videoconferencing applications is yet unknown. We address both challenges in this chapter.

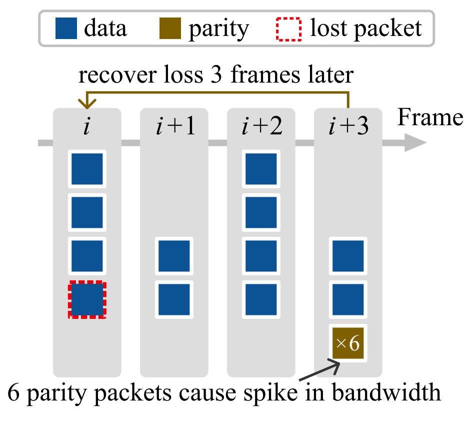
To begin, we provide the background of FEC for real-world videoconferencing applications in Section 5.1. We then assess packet-loss characteristics of real-world video calls to determine the viability of streaming codes in Section 5.2. Building upon these packet loss characteristics and the lessons we learned from designing the streaming codes from Chapter 3, we introduce a new communication scheme suitable for the videoconferencing stack in Section 5.3. Finally, we conduct an empirical evaluation of its performance in Section 5.4 and demonstrate concrete benefits for several key metrics of the QoE (e.g., reducing the median frequency of freezes by 26%).

5.1 Background on FEC for real-world videoconferencing applications

Many applications, such as Microsoft Teams, employ traditional FEC (e.g., block codes) for loss recovery. We start by explaining the limitations of this approach for videoconferencing applications. We then summarize the two challenges for using the streaming codes presented in Chapters 2, 3, and 4 for videoconferencing.



(a) Within-frame



data
parity
lost packet
recover loss 3 frames later
i
i+1
i+2
i+3
Frame
×6
6 parity packets cause spike in bandwidth

(b) Multi-frame

Figure 5.1: Two approaches for employing block codes: (a) within each frame and (b) across multiple frames.

5.1.1 Conventional FEC and its challenges in videoconferencing

Block codes. One of the most commonly used FECs is the so-called "block codes." Block codes encode k frames , $\langle S[1], \ldots, S[k] \rangle$ to r parity packets into $\langle S[1], \ldots, S[k], P[1], \ldots, P[r] \rangle$, so that the k frames can be recovered using a subset of the $(k+r)$ packets. When any k of the $(k+r)$ packets suffice for recovery, the block code is termed "maximally distance separable (MDS)." One of the best known examples of MDS codes is the Reed-Solomon (RS) block codes [76]. Other examples of block codes include fountain (i.e., rateless) codes [60], or two-dimensional block codes [87].

Traditionally, FEC applies to packets, but videoconferencing involves transmitting multiple packets for each video frame. One natural solution is to apply a block code to the frames *within each frame* (Figure 5.1a). The parity packets are sent immediately after the final frame of a frame. A second approach is to apply a block code across the frames of *multiple frames* (Figure 5.1b) by sending all parity packets after the final frame of the last frame in the block. Our analysis of the production packet loss traces (Section 5.2) from Teams shows that the packet losses are bursty. Both approaches have significant limitations for burst losses.

Limitations of block codes for videoconferencing. When packet losses occur as bursts, the within-frame approach wastes the redundancy sent in frames immediately following a burst because it is useless for recovering the lost frames. Although the multi-frame approach overcomes this problem, it has two main drawbacks. First, the latency of recovering losses is high due to waiting for the parity packets, which are sent after the final frame in the block, to recover any packets. The length of the block code must be short lest the latency exceeds the real time deadline to play a frame, leading to an increased bandwidth overhead and reduced robustness to burst losses. Second, packets sent in rapid succession may be lost if a router buffer is full. When a full router buffer coincides with the final frame of a block, no lost packets are recovered.

The bandwidth consumed by parity packets of FEC can be substantially higher than retransmission, even for modest packet loss rates. Unlike retransmission, which only resends lost packets, even an "optimal" FEC scheme does not know which packets will be lost. Hence, it must send far more parity packets than lost packets. For example, to prevent a video freeze, at least one parity packet must be sent every ≈ 150ms to cover the scenario of losing a frame . However, this parity packet is not used if there are no losses.

5.1.2 Challenges of using streaming codes for videoconferencing

Gaps between the existing model and videoconferencing applications. The existing practical work on streaming codes [13, 33], like the theoretical work they build upon [11, 62], is limited to settings where the amount of data to be transmitted at each time instant is a fixed constant. However, videoconferencing involves sending compressed video frames whose sizes vary. Only our streaming code constructions (see Chapters 2, 3, 4) can handle this variability. However, as discussed in Chapters 2, 3, and 4, existing streaming codes consider an *adversarial* loss model that imposes bursts of length b. When applied for videoconferencing, the parameter b translates into the number of consecutive frames for which *all* packets are lost. However, videoconferencing applications frequently send multiple packets per frame, and often only some of these packets are lost, as we show in greater detail in Section 5.2 for packet loss traces from production. Existing streaming codes are overly pessimistic because they can recover from losing all packets for multiple consecutive frames. This requirement imposes a significant bandwidth penalty, negating the potential bandwidth savings of streaming codes. Streaming codes are also vulnerable to recovery failures if there are *any* losses in the guard space after a burst. But, in practice, many bursts are not followed by such guard spaces (see Section 5.2.2).

Applicability of streaming codes in the wild. The benefits of streaming codes for VoIP applications have been studied using simulated losses under theoretical loss models, such as the Gilbert-Elliott channel [31] and over traces [13, 33], wherein each frame is sent in one packet and all frames/packets are of a fixed constant size. However, these results do not apply to videoconferencing applications, which send (a) multiple packets for each frame and (b) varying amounts of data per frame. Streaming codes perform best when each burst occurs across multiple frames and is followed by a guard space of several frames without losses. A natural question is whether such losses arise in videoconferencing and if they can be exploited via streaming codes. To the best of our knowledge, no study of large-scale real-world packet losses establishes the applicability of streaming codes in the wild. Furthermore, establishing that streaming codes are viable to improve the QoE hinges on improving several metrics relating to the QoE. Yet an analysis of streaming codes' impact on such metrics is similarly lacking in the existing literature. Finally, the effect of inter-frame dependencies on the benefits of streaming codes has yet to be assessed, even though inter-frame dependencies are prevalent in videoconferencing.

We are addressing these challenges in three steps:

1. Analyze thousands of packet loss logs for video calls taken from a large commercial videoconferencing application, and characterize their suitability for using streaming codes. To the best of our knowledge, this is the first work to evaluate the potential of streaming codes using large-scale, real-world traces.
2. Present *Tambur*, which bridges the gap between the theory behind streaming codes and videoconferencing applications by (a) designing a new streaming code that is well-suited to videoconferencing and (b) integrating it with a lightweight ML model to take a predictive

decision on the bandwidth allocated to streaming codes.

3. Implement a new benchmark platform to enable research on videoconferencing with an easy-to-use interface to integrate and assess new FEC schemes. In addition, implement Tambur, Block-Within, and Block-Multi in C++ and incorporate them into the benchmark platform using the interface.
4. Evaluate Tambur over a large corpus of production traces through simulation, and show that it simultaneously reduces the frequency of non-recoverable frames and bandwidth overhead by 26.5% and 35.1%, respectively.
5. Evaluate Tambur over emulated networks and show significant improvements over key metrics pertaining to end-to-end QoE (e.g., reducing the frequency of freezes by 26% and the cumulative duration of freezes by 29%).

5.2 Packet loss in the wild

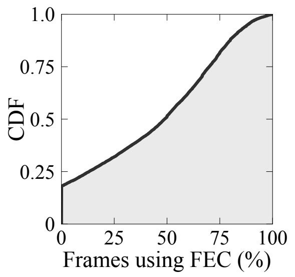
Logs (specifically, packet loss traces) from Microsoft Teams were collected from a random sample of 1:1 video calls over two weeks. One week's traces were held out as a test set for the evaluation.

Teams uses FEC only after a packet loss occurs, which is fairly standard in the industry [83] to avoid wasting bandwidth for the many video calls that do not experience any loss. We limit our study to traces with at least two instances of loss since our focus is on improving scenarios *after* FEC is activated (i.e., FEC is turned on after the first loss and then used to recover the second). Our analysis involves approximately 9700 traces, which constitute 16% of all the traces. Studying these traces sheds light on the tail performance, which is crucial for real-world commercial applications. Each trace corresponds to one video call and contains the size, sequence number, and send/receive timestamps for each received packet, as well as whether it is a parity packet or data packet; lost packets are identified via missing sequence numbers. Due to the application's data collection method, the traces are limited to the final one minute of the call. Although the logs are for packets, we approximate frame-level information by combining the logs with Teams's packetization logic and have corroborated with the Teams engineers that this approximation is good.

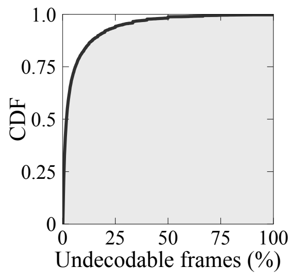
5.2.1 FEC metrics

Teams employs an RS block code within each frame and varies the bandwidth overhead based on infrequent feedback from the receiver on packet losses. We will denote the FEC scheme used by the application simply as "Block-Within."

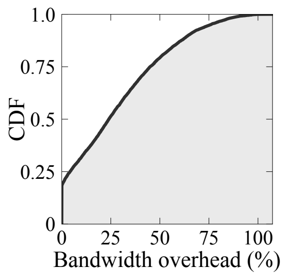
We evaluate three metrics over the traces. First, the percent of video frames using FEC for each videoconferencing call (Figure 5.2a). The 25th, 50th, and 75th percentile for the percent of video frames over each trace using FEC are 13%, 48.8%, and 70% of calls respectively, indicating that FEC is applied to a significant portion of the frames. Second, the percent of decoding failures for video frames over all frames for each videoconferencing call (Figure 5.2b). The 25th, 50th, and 75th percentile for the percent of decoding failures of frames are 0.6%, 1.8%,



(a) FEC usage



(b) Decoding failures



(c) Redundancy

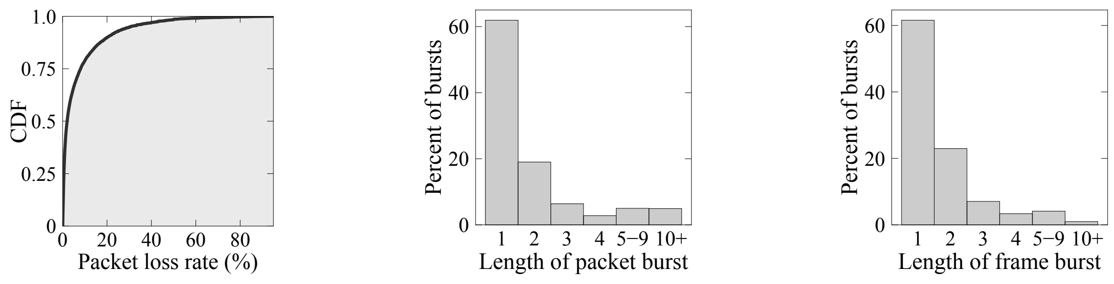
Figure 5.2: CDFs over the traces from Teams of (a) how often FEC is used to encode frames to protect against packet loss, (b) how often the lost packets are not decoded, and (c) the bandwidth overhead of parity packets.

and 6.1% of calls. Note that the decoding failures should be kept below around 1% to provide high QoE [84]. As such, decoding failures are prevalent enough to tangibly negatively impact the QoE, prompting the need for a more effective FEC mechanism. Third, the bandwidth overhead for each call (Figure 5.2c). The 25th, 50th, and 75th percentile for the bandwidth overhead are 4.2%, 24%, and 45% of calls. Thus, reducing the bandwidth overhead will free a significant portion of the bandwidth for these calls.

5.2.2 Network quality

We analyze the packet losses to assess streaming codes' suitability for real-world videoconferencing applications. To the best of our knowledge, this is the first work to analyze large-scale real-world packet loss traces from this perspective. We analyze three key metrics of losses in Figure 5.3. (1) The packet loss rate for each trace (Figure 5.3a). (2) The distribution of lengths of bursts of packets measured over all of the calls (Figure 5.3b). (3) The distribution of the lengths of bursts of frames (i.e., the number of consecutive frames with at least one packet lost) measured over all of the calls (Figure 5.3c). This metric indicates streaming codes' suitability, as they are most effective when bursts of lost packets encompass multiple frames.

The mean percent of packets lost over the traces is 7%. It is higher than the packet loss in the FCC report [22] since we focus on the traces where FEC is employed. If the other traces from Teams are also considered, the mean packet loss over all traces is 1.7%, which is comparable to the FCC measurement. In earlier studies of end-to-end Internet packet loss, loss rates tended to vary over time and between ISPs and access network technology [15, 25, 32, 74], with ISP queue management policies impacting the loss patterns seen by applications. As discussed in [32], in home broadband networks, loss rates are often less than 1% for long periods, with infrequent periods of very bursty packet loss. Similar patterns are seen in mobile networks, where loss rates tend to increase during handovers [15], and much longer packet loss bursts are seen. Our traces from Teams, described in this section, show similar behavior, with a large number showing very low loss rates, with a long-tail of traces showing extremely bursty packet loss. Specifically, 38.1% of the instances of packet loss involve at least two consecutive packets being lost (Figure 5.3b), and 38.4% of instances of packet loss encompass more than one video frame. Such loss patterns can be efficiently recovered by streaming codes (Section 5.2.3).



(a) Packet loss (b) Packet bursts (c) Frame bursts

Figure 5.3: Packet loss is prominent (e.g., Fig 5.3a shows $1 - 10\%$ packet loss for most traces) and often occurs as bursts across consecutive packets (Fig 5.3b) or frames (Fig 5.3c).

There is also a trade-off between the bandwidth overhead (i.e., the bandwidth used for parity packets) and the probability of decoding failure. The bandwidth overhead cannot be prohibitively high lest there be insufficient bandwidth for the original data. Consequently, the frequency of decoding failures for frames is non-negligible despite using FEC.

5.2.3 Potential of streaming codes

Recall that streaming codes are most effective when (a) packet loss occurs as a burst across multiple consecutive frames and (b) the burst loss is followed by a guard space of multiple consecutive frames with no losses. We formalize two metrics to capture these conditions. We then show that the packet losses in the traces exhibit these features.

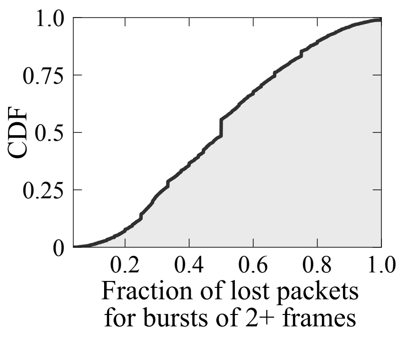
Measuring bursts. The bandwidth overhead needed to decode a burst depends on the fraction of the packets being lost when losses occur across multiple frames. We introduce a new metric to formalize this notion.

Definition 1 (Multi-frame burstiness)**.** *Suppose a burst occurs across two or more frames, i through j, over which s packets are sent. If l of the s packets are lost, the **multi-frame burstiness** is defined as l/s.*

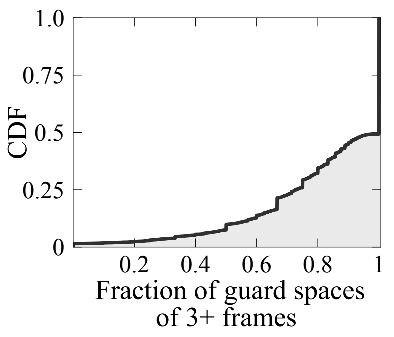
For example, suppose Tambur sends packets $(D_0[i], D_1[i], D_2[i])$ and $(D_0[i+1], D_1[i+1])$ over frames i and $(i+1)$, respectively. Suppose $D_1[i], D_2[i]$, and $D_0[i+1]$ are lost. Then the multi-frame burstiness is $3/5$. The multi-frame burstiness is always positive since at least one packet is lost for each frame in the burst. The maximum value of 1 occurs when all packets are lost for all frames in the burst. High values correspond to situations of a high percentage of the packets being lost for multiple consecutive frames. The value of the multi-frame burstiness directly relates to the minimum bandwidth overhead needed for any code to decode lossy frames in real time.

Measuring guard spaces. Streaming codes can reduce bandwidth overhead for scenarios where a burst of packet losses is followed by a guard space of at least τ frames that experience reliable transmission, where τ is the latency deadline parameter. We now introduce a new metric to measure the extent to which the guard spaces exhibit this property.

Definition 2 (Guard space sufficiency)**.** *The **τ-guard space sufficiency** is the fraction of instances in which one or more frames with packet loss are followed by at least τ consecutive frames which*



(a) Multi-frame burstiness



(b) Guard space sufficiency

Figure 5.4: The CDFs over the traces of the (a) the multi-frame burstiness (for traces with at least one burst over 2+ frames), and (b) the guard space sufficiency.

experience lossless transmission.

The value of the guard space sufficiency varies from 0 to 1. It is *negatively* related to the bandwidth overhead needed when using streaming codes. High values for the guard space sufficiency indicate that the bandwidth overhead can be reduced.

Suitability of streaming codes. The multi-frame burstiness and 3-guard space sufficiency is evaluated over the traces in Figure 5.4.[1] In Figure 5.4a, the value of the multi-frame burstiness is shown to vary over the range 0 to 1, with values at the 25th, 50th, and 75th percentiles of 0.32, 0.5, and 0.67 respectively. This indicates that the bandwidth overhead needed when using streaming codes varies over the traces, as expected. For higher values, more bandwidth must be allocated to redundancy to decode the losses. For lower values, it is possible to make do with less bandwidth used for redundancy. The guard space sufficiency is evaluated over the traces in Figure 5.4b, and its values at the 25th, 50th, and 75th percentiles are 0.73, 1.0, and 1.0 respectively. These values imply that *streaming codes are often suitable*. For example, for the traces with a value of 1.0, every single time a burst occurs across multiple frames, streaming codes could have been used to decode the losses with the optimal amount of bandwidth overhead. Yet, the low values indicate insufficient guard spaces for using existing streaming codes to reduce the bandwidth overhead, as they are vulnerable to losses in the guard space.

5.2.4 Key findings

Bursts of packet losses followed by guard spaces arise frequently and are conducive to streaming codes. However, this is not always the case. Bursts are sometimes followed by short guard spaces or involve significant packet loss, in which case the bandwidth overhead cannot be reduced via streaming codes. Hence, integrating streaming codes into real-world applications *requires (a) predicting whether the bandwidth overhead can be reduced without incurring decoding failures, (b) leveraging partial losses in a frame (i.e., losses of only some packets per frame rather than*

[1]If the maximum tolerable latency is 150 ms (a standard value for real-time video communication [84]), the one-way propagation delay is 50 ms, and a frame is encoded every 33.3 ms (i.e., at 30 fps), τ could be set as 3 $(= (150 - 50)/33.3)$. So $\tau = 3$ applies for a realistic choice of parameters, in which case a guard space of length 3 is beneficial for loss recovery with streaming codes.

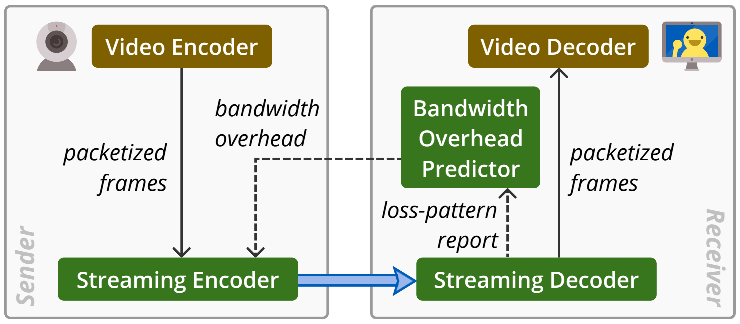


Figure 5.5: Overview of Tambur. The components in green are specific to Tambur.

all packets) and (c) adding robustness to losses in the guard space.

5.3 Tambur

We present Tambur, which exploits the potential discussed in Section 5.2.3 and addresses the challenges discussed in Section 5.1.2 by (1) using an ML model to take predictive decisions on the bandwidth overhead, and (2) designing a new streaming code suitable for videoconferencing given any setting for the bandwidth overhead.[2] First, an ML model makes a predictive decision on the number of parity symbols to allocate for each frame. This helps to set the bandwidth overhead to match the network conditions. Second, the parity symbols are defined to provide (a) sequential recovery of bursts over multiple frames while exploiting partial losses, (b) recovery of occasional losses within a single frame immediately, and (c) robustness to a small amount of loss in the guard space after a burst. Third, a new methodology is employed to distribute each frame's data and parity symbols over multiple packets. The design of the parity symbols and their distribution across packets constitute Tambur's streaming code. During loss recovery, Tambur uses the received packets from the frames involved in a burst (i.e., partial losses), which allows for a lower bandwidth overhead than is possible for existing streaming codes that ignore such packets.

Figure 5.5 shows how Tambur fits into the stack of a videoconferencing application. The *streaming encoder* encodes data from compressed frames into frames and parity packets. A *Bandwidth Overhead Predictor* periodically selects bandwidth overhead for each frame using a predictive (ML) model based on the losses observed at the decoder and sends the value to the encoder. The *streaming decoder* uses parity packets to recover lost frames . We will now describe these components in detail.

5.3.1 Tambur's streaming code

We present the code in two parts: encoding and decoding.

[2]The new streaming code builds upon the theoretical streaming code from Chapter 3 while overcoming the limitations discussed in Section 5.1.2.

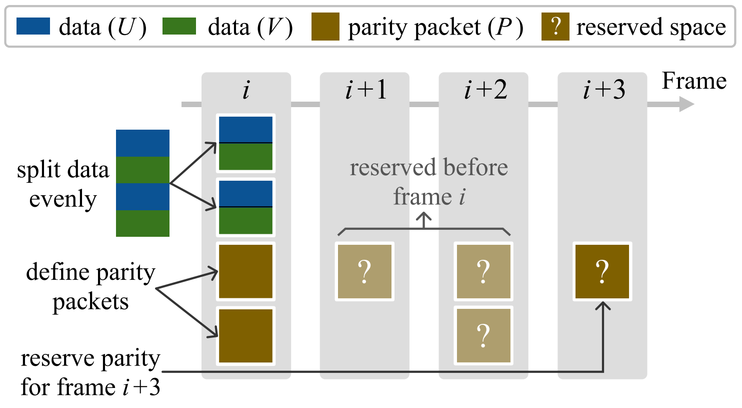


Figure 5.6: Encoding for $\tau = 3$. Tambur splits frame i evenly into $(V[i], U[i])$ and sends them over frames . Also, Tambur sends parity packets for recovering $V[i-3], \ldots, V[i], U[i]$ and $U[i-3]$ and reserves space for parity symbols of frame $(i+3)$.

Encoding. We illustrate how Tambur encodes the ith frame. Figure 5.6 shows an example of encoding for $\tau = 3$. The frame symbols of this frame, $S[i]$, are sent in frames , and the parity symbols, $P[i]$, are sent in parity packets. The sizes of the packets are maximized subject to (a) not exceeding an MTU (for example, 1500 bytes in our experiments) to be equal. The previous value of the Bandwidth Overhead Predictor determines how many parity symbols are allocated for frame i. These parity symbols will be sent τ frames later (see "reserved space" in Figure 5.6). The number of parity symbols sent for frame i was determined by the size of frame $(i - \tau)$.

Next, we describe how parity symbols are formed. The symbols of $P[i]$ are linear combinations of the symbols of the $(\tau + 1)$ frames, $\{S[i], \ldots, S[i - \tau]\}$. To define the parity symbols, it helps to view the frame symbols of the associated $(\tau + 1)$ frames as being divided evenly into two parts as $S[j] = (V[j], U[j])$, for $j \in \{i, \ldots, i - \tau\}$. Figure 5.6 shows these components in blue and green, respectively.

The symbols of $P[i]$ are designed linear combinations of the symbols of the following quantities: $V[i], \ldots, V[i - \tau], U[i]$, and $U[i - \tau]$. Specifically, $P[i]$ is sum of three quantities: $P[i] := P_1[i] + P_2[i] + P_3[i]$. The symbols of $P_1[i]$ are linear combinations of the symbols of $V[i - \tau], \ldots, V[i - 1]$. The symbols of $P_2[i]$ are linear combinations of the symbols of $U[i - \tau]$. The symbols of $P_3[i]$ are linear combinations of the symbols of $U[i]$ and $V[i]$. All linear combinations are carefully chosen to be *linearly independent* linear equations.[3]

Decoding. We describe the decoding process in two parts: (1) occasional packet losses and (2) burst of packet losses. Let all frames before the loss be decoded. First, suppose that packet loss is rare, and the size of $P[i]$ exceeds the number of symbols lost for frame i. Then $P[i]$ suffices to decode the ith frame (specifically, by solving a system of linear equations).

Second, consider a burst of packet losses across two consecutive frames for $\tau = 3$, as is shown in Figure 5.7. Packet losses (red-dashed border) span frames i and $(i+1)$. For each frame i, the blue, green, and brown parts represent $U[i]$, $V[i]$, and $P[i]$, respectively. First, $V[i]$ and $V[i + 1]$ are both decoded using $P[i + 2]$, which consists of independent linear combinations of (a) the symbols of $V[i]$ and $V[i + 1]$, and (b) the (received) symbols of $V[i - 1]$, $U[i - 1]$,

[3]It suffices to take linear equations from three different Cauchy matrices.

$U[i+2]$, and $V[i+2]$. Second, for $j \in \{i, i+1\}$, $U[j]$ is decoded using $P[j+3]$, which consists of independent linear combinations of (a) the symbols of $U[j]$, and (b) the (available) symbols of $V[j-3], \ldots, V[j]$, and $U[j]$. The key to this methodology is that $U[i+1]$ is *not* recovered by the latency deadline of $S[i]$ (i.e., $(i+3)$). This enables using extra parity symbols (i.e., $P[i+4]$) to recover $U[i+1]$ while still decoding each frame within $\tau = 3$ frames. Appendix 5.5.1 presents the general case. If decoding fails, the receiver queries the sender to generate a new keyframe (i.e., a self-sufficient frame) to handle inter-frame dependencies.

There are three key differences from existing streaming codes for videoconferencing: (1) The frame symbols and parity symbols of a frame are sent over multiple packets instead of a single packet. (2) Each frame's parity packets are designed such that they are useful in recovering its lost data packets (in addition to being useful in recovering previously sent frames). (3) The code is flexible enough to allow per-frame bandwidth overhead to be set using the Bandwidth Overhead Predictor.

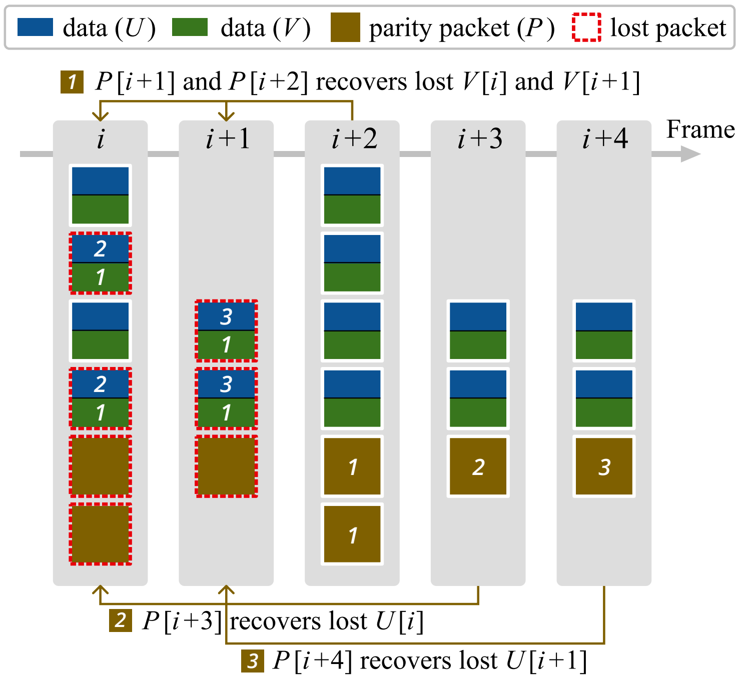

data (U)
data (V)
parity packet (P)
lost packet
1 P[i+1] and P[i+2] recovers lost V[i] and V[i+1]
i
i+1
i+2
i+3
i+4
Frame
2 P[i+3] recovers lost U[i]
3 P[i+4] recovers lost U[i+1]

Figure 5.7: Decoding a burst across 2 frames within $\tau = 3$ frames delay using Tambur's streaming code. Data symbols labeled 1, 2, and 3 are decoded using the parity packets with the same label.

5.3.2 Bandwidth overhead predictor

At a high level, Tambur makes use of a predictive model to determine the bandwidth overhead employed by its streaming code (i.e., the amount of "reserved space" in Figure 5.6). This predictive model takes as input a feature vector computed by the receiver periodically (e.g., every two seconds), dubbed a *loss-pattern report*. The predictive model's output is then sent to the sender to set the bandwidth overhead for each frame for Tambur's streaming code until the next loss-pattern report is received. For example, a bandwidth overhead of 50% means that if frame i comprises 1000 bytes, 500 bytes of parity symbols are allocated.

Loss-pattern report. Let P be the bitmap of packet losses since the last loss-pattern report, where 1 denotes a loss and 0 is a reception. Let F be a bitmap over all frames since the last loss-pattern report of whether at least one of the frame's packets was lost. The loss-pattern report consists of the following 13 quantities, all of which can be computed in linear time with a single sequential pass over F and P.

- Multi-frame burstiness and guard space sufficiency (Section 5.2).
- Fraction of losses for P and F.
- Mean number of consecutive losses for P and F.
- Mean length of guard spaces for P and F.
- Burst density [19] and gap density [19] for P and F[4].
- A score employed by Teams to choose its bandwidth overhead, which is based on the observed fraction of packet losses and lengths of bursts.

Bandwidth overhead prediction via weighted classification. Tambur uses an ML model to determine the bandwidth overhead allocated per frame based on the recent loss conditions. As discussed above in Section 5.3.1, Tambur's streaming code enables such an approach by allowing fine-grained tuning of the bandwidth overhead. To keep the model simple, we select two options for the bandwidth overhead. This approach easily generalizes to more than two values for bandwidth overhead by using a multiclass classifier to enable tuning the bandwidth overhead used by Tambur. In our implementation, we use a small neural network (discussed further in Section 5.3.3), although any methodology could be substituted.[5]

The ML model is trained with different *weights* for the two classes based on prioritization of bandwidth savings versus minimizing decoding failures. Essentially, the higher the weight for the class corresponding to the higher bandwidth overhead, the greater the frequency of decoding frames, but the lower the reduction in bandwidth overhead. Videoconferencing service operators can use these weights as a knob to prioritize reducing decoding failures or bandwidth overhead.

Neural network details. Binary classification is conducted using a small fully connected neural network with one hidden layer. The input is the values of the 13 metrics for the previous 3 loss-pattern reports. The cross-entropy loss is applied, and by default the weights for mistakenly reducing the bandwidth overhead (i.e., causing a decoding failure) and not reducing the bandwidth overhead by half (i.e., failing to save bandwidth) are 0.999 and 0.001, respectively. We tested various number of hidden neurons (e.g., 100, 1000, and 10000) and selected 1000 as the smallest option to reach the point of diminishing returns. The model is implemented and trained in PyTorch offline using the traces based on the optimal decision for reducing the bandwidth overhead without causing decoding failures. During inference, it is instantiated in C++.

[4]The parameter gMin [19] is set to be 1 and τ for P and F respectively.
[5]We found ML models to outperform heuristics empirically.

5.3.3 Implementation

We implemented Tambur in C++ as part of a new independent library called Tambur that any videoconferencing application can use.[6] At the sender, Tambur takes successive compressed frames as input and outputs frames and parity packets. At the receiver, Tambur decodes lost packets by solving a system of linear equations using the symbols of the received packets. When packets are lost, we combine properties of streaming codes with an open-source min-cut/max-flow algorithm [18] to determine which frame symbols can be decoded using which parity symbols in negligible time (see Appendix 5.5.2). Data is then decoded by solving the smallest full-rank systems of linear equations.

We use a small header to provide frame-level information needed for decoding. This includes sequence numbers for packets and frames and relative positions of a packet within a frame and amongst parity packets. The streaming decoder also needs the size of the lost frame in order to decode it (even when all packets corresponding to the frame are lost); hence, we encode the sequence of frame sizes using a streaming code and send one parity symbol of this code in each packet.

The library provides an interface for rapidly prototyping new FEC schemes. We used this interface to implement the baselines from Section 1 (i.e., Block-Within and Block-Multi).

The core arithmetic of linear encoding and solving a system of linear equations for decoding is done using Jerasure 2.0 [70], an open-source library in C/C++ with modules for key operations of erasure coding. Jerasure 2.0 is built on top of the GF-Complete library [71], which uses Intel SIMD instructions to perform Galois Field arithmetic quickly. Tambur involves encoding data into “coding blocks” of 256 bytes, each of which uses the same linear equations. Extending Tambur to use hardware offload to encode and decode frames is a potential avenue of future work.

Integration with videoconferencing. To validate Tambur’s effectiveness in the real world, we integrate it with *Ringmaster*[7], a new videoconferencing platform that emulates one-on-one video calls for benchmarking FEC schemes. Ringmaster is implemented in $\sim$4000 lines of C++. Its video sender reads raw frames from an input Y4M video file on disk at a precise frame rate (e.g., 30 fps) and compresses them with the VP9 encoder in the `libvpx` [1] library using similar codec configuration as in WebRTC [2]. A user-provided FEC scheme provides parity data for the encoded frames, which is sent over UDP after packetization to the video receiver. Upon receiving the frames, the video receiver applies the FEC decoder and VP9 decoder sequentially to decode and render the original video frames. In addition, Ringmaster allows for requesting new keyframes, e.g., when the receiver fails to recover a video frame due to excessive loss of packets and thus requests the sender to encode a new keyframe so as to resume the video. At the end of the automated call, QoE metrics are computed by aggregating logs from both endpoints, which record the timestamps when each frame is encoded or decoded, along with its frame ID, size, FEC bandwidth overhead, etc.

Ringmaster provides clean and modular interfaces that we use to integrate it into Tambur. Combining Ringmaster with Tambur enables benchmarks of FEC schemes’ performance featuring QoE metrics, e.g., video freezes, per-frame delay, rendered frame rate, for FEC schemes im-

[6]`https://github.com/Thesys-lab/tambur`
[7]`https://github.com/microsoft/ringmaster`

plemented via Tambur's interface. Furthermore, Ringmaster also allows researchers to isolate the impact of FEC and disable modules that interfere with FEC, such as bandwidth estimation [20] and packet retransmission.

5.4 Evaluation

To assess whether Tambur can improve the QoE, we ask:

- Can Tambur provide significant benefits for metrics relating to FEC on real-world losses?
- Do the benefits of Tambur lead to a higher QoE?

5.4.1 Experimental methodology and highlights

Videoconferencing application parameters. In our experiments, we aim for a maximum tolerable latency of 150 ms to meet industry recommendations [84], which is a fairly standard value for interactive video. The frame rate is taken to be 30 fps, which is a typical value in videoconferencing. The inter-frame arrival time for 30 fps is 33.3ms. Allowing for a one-way frame delay of 50 ms leaves room for a decoding delay of around 100ms. Thus, the parameter τ can be at most 3 (frames) for the end-to-end latency (i.e., $33.3\tau + 50$) to be at most ≈ 150 ms. The two options for the bandwidth overhead of Tambur are to match or use half of the bandwidth overhead of the baseline coding scheme, Block-Within, which is introduced next.

Coding schemes. We evaluate six coding schemes. (1) **Block-Within** (Figure 5.1a), which applies RS codes within a frame. This scheme is employed in production by Teams. (2) **Block-Multi** (Figure 5.1b) which applies RS codes across $(\tau + 1) = 4$ frames. RS codes are *optimal* block codes, and hence the above two baselines outperform other block codes such as fountain or rateless codes in recovering losses and bandwidth overhead. (3) **Tambur-full-BW**, which is a variant of Tambur that matches Block-Within's bandwidth overhead. (4) **Tambur-0.9**, which is Tambur with the neural network trained to prioritize bandwidth savings more by decreasing the weight of misclassification from 0.999 to 0.9 in the loss function. Thus, Tambur-0.9 prioritizes reducing the bandwidth overhead more than Tambur. (5) **Tambur-low-BW**, which is a variant of Tambur that uses 50% of the bandwidth overhead of Block-Within. (6) **O**, which optimally selects between Tambur-full-BW, Tambur-low-BW, or Block-Within. Each time the sender obtains feedback from the receiver, the Oselects the scheme with the smallest bandwidth overhead among the scheme(s) that recover the most frames. This choice never causes a non-recoverable loss. Consequently, the Oalways recovers at least as many frames as Block-Within, Tambur, and Tambur-full-BW. The bandwidth overhead of Block-Within and Block-Multi is never reduced to ensure a fair comparison of Tambur's loss recovery capabilities and because both baselines already perform worse than Tambur despite using the full bandwidth overhead. Like Tambur, Block-Within and Block-Multi send feedback to the sender once FEC decoding has failed to trigger a new keyframe as a fallback mechanism to handle inter-frame dependencies.

Metrics. We evaluate the following metrics: (1) Percent of non-recoverable frames, which is the percentage of compressed frames that are not recovered. (2) Bandwidth overhead for FEC. (3) Percent of non-rendered frames, which is the percent of frames that are not played by the receiver—this includes non-recovered frames and recovered frames that depend on non-recovered frames. (4) Latency, which is the duration between a frame being created and rendered. (5) Frequency of freezes, which is the number of times the receiver's video is frozen. (6) Duration of freezes, which is the cumulative length of time where the receiver's video is frozen.[8] We calculate these metrics only for the frames where FEC is applied (i.e., where FEC affects the quality). We compute one value per call (e.g., median duration of freezes, bandwidth overhead, etc.) and then consider the percentiles over these values. For latency, we consider all frames over all calls.

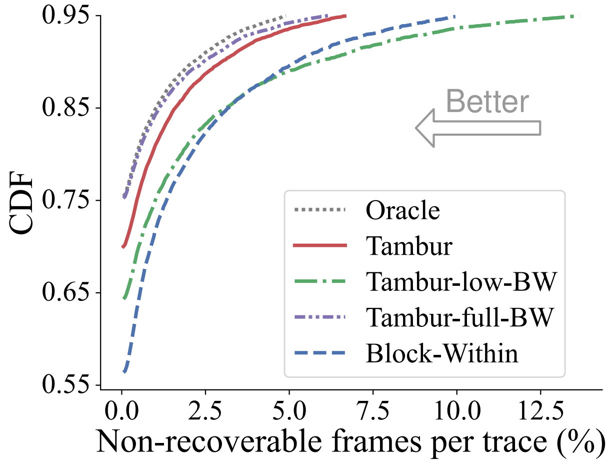
QoE is difficult to measure precisely with so-called "QoE models" [88] because it depends on video-specific properties (e.g., in sports, video quality during gameplay matters more than during timeouts). But several works [14, 29, 54] have shown that key metrics for QoE (e.g., frequency of freezes, duration of freezes, bandwidth, etc.) impact the mean opinion score—the gold standard measure of QoE. These metrics also affect user interactions (e.g., users watch more video when there are fewer freezes). In fact, [26] showed that cumulative freeze duration is crucial for QoE, as well as the importance of bitrate and frequency of video freezes for live video.

Offline evaluation. We evaluate the performance of Block-Within, Tambur, Tambur-full-BW, Tambur-0.9, Tambur-low-BW, and Oover the test set of traces from Teams described in Section 5.2, which was held out from the previous analyses. The packet logs provide the performance of Block-Within. We make two safe assumptions to evaluate the remaining schemes over the traces: (a) modifying the payload of a packet, but not its size, would not change whether it is lost or received; (b) reducing the size of a packet's payloads would not incur any new packet losses. Each frame is sent identically as in the trace, the payloads for the parity packets are changed, the sizes of the parity packets are sometimes reduced, and the bitmap of packet losses from the trace is used. To satisfy the assumptions, we must force Tambur to send the number of parity symbols allocated for each frame within the frame (rather than delayed by τ frames), which we expect to *degrade Tambur's performance*. This enforcement alters the number of parity packets sent under Tambur but not how their symbols are defined. Block-Multi is excluded because it sends all parity packets after the final frame of the final frame of the block, so its performance cannot be fairly simulated using the production traces.

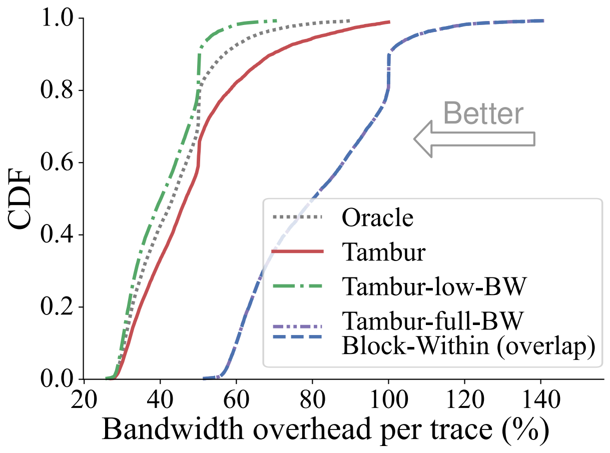
Online evaluation. We evaluate prototype implementations of Block-Within, Block-Multi, Tambur, Tambur-full-BW, and Tambur-0.9 integrated with Ringmaster (the videoconferencing benchmark platform described in Section 5.3.3) via network emulation using Mahimahi [67] while simulating a Gilbert-Elliott (GE) [31] loss model over a dataset of 80 videos. Specifically, we evaluate 20 video calls from [23, 68] at four constant bitrates each (namely, $500, 1000, 1500,$ and 2000 kbps) to isolate the effect of FEC. The bandwidth overhead is set to 50% for Block-Within (likewise, for Block-Multi and Tambur-full-BW).[9] The GE loss model is a standard loss model which is a Markov model with two states: "good" and "bad," each with an associated probability of packet loss. For a fair and realistic comparison, different coding schemes must experience the same distribution of burst losses at the frame level even though they send differing

[8]We use the definition of freezes and duration of freezes from the most recent (unofficial) draft of identifiers for WebRTC's statistics [17].

[9]The bandwidth overhead is sometimes slightly higher for *all* schemes due to rounding and ensuring at least one parity packet is sent per frame.



(a) Non-recoverable frames



(b) Bandwidth overhead

Figure 5.8: CDFs for the percent of non-recoverable frames for the 55th through 95th percentiles and the bandwidth overhead for the offline evaluation.

numbers of packets per frame. Therefore, we consider transitions between the states occurring once at the start of every frame (i.e., once every 33.3 ms) rather than a transition between states every packet, which is commonly used in the literature when only one packet is sent per frame. Packets within each frame are lost independently with the same probability. The modified GE channel can be viewed as a buffer overflowing for a short period, as can arise from on/off characteristics of traffic [69]. Appendix 5.5.3 details how we set the parameters of the GE model based on the losses from the traces.

Result highlights.

- In offline evaluation, Tambur reduces the frequency of non-recoverable frames by 26.5% while using 35.1% less bandwidth overhead.
- In online evaluation, Tambur reduces frequency of non-rendered frames, frequency of freezes, and duration of freezes by 28%, 26%, and 29%, respectively compared to Block-Multi, and by 73%, 78%, and 77% compared to those of Block-Within. Block-Multi has a significantly higher latency than Block-Within (see Figure 5.11b).
- Modest memory overhead and median encoding and decoding times of 575 KB, 1.7ms, and 3.4ms, respectively.

5.4.2 Offline evaluation

We assess only the frequency of non-recoverable frames and the bandwidth overhead for offline traces because the remaining metrics are unavailable. Figure 5.8a shows the CDF of the percent of non-recoverable frames from 55th to 95th percentiles over the traces. These percentiles correspond roughly to the 92nd to 99th percentile over all traces. The Oreduces the total number of non-recoverable frames by 44.2% compared to Block-Within and reflects an upper bound on performance. Tambur-full-BW reduces the frequency of non-recoverable frames by 33% compared to Block-Within, indicating the potential improvements of using streaming codes. In contrast, Tambur-low-BW *increases* the frequency of non-recoverable frames by 34.7% compared

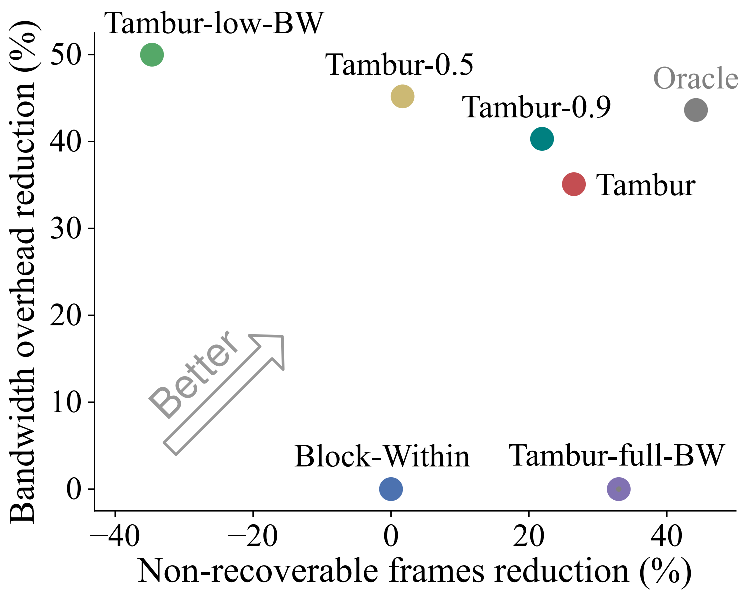
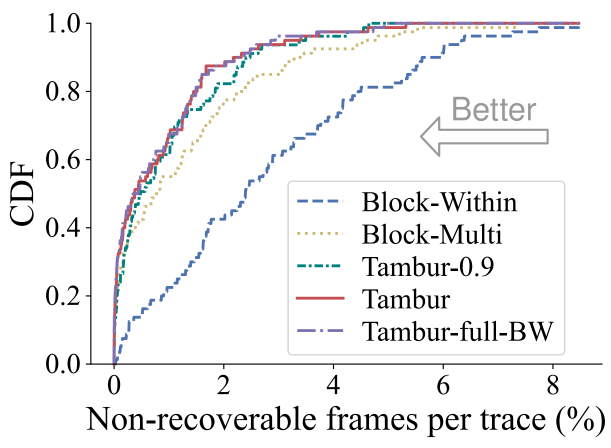


Figure 5.9: Sensitivity analysis of the weights for the classes used in the predictive model for the frequency of non-recoverable frames and bandwidth overhead over all of the frames where FEC is used in the traces.

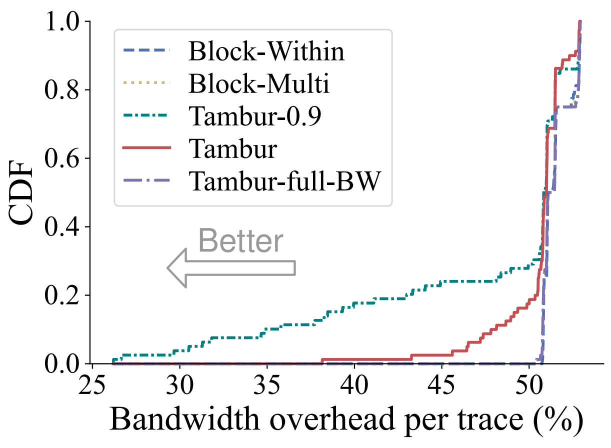
to Block-Within, indicating the need for more sophisticated methods to reduce the bandwidth overhead without incurring a significant penalty in non-recoverable frames. By using a predictive model to determine the bandwidth overhead, Tambur reduces the bandwidth overhead by 35% while simultaneously reducing the number of non-recoverable frames by 26.5% compared to Block-Within (Figure 5.8b). Section 5.4.3 summarizes the spectrum of bandwidth savings versus recovering frames for Tambur based on tuning the associated weight parameter.

5.4.3 Sensitivity analysis

There is an inherent trade-off in performance between the non-recoverable frames and bandwidth overhead metrics. The ML model for Tambur is trained using a loss function with a weight of 0.999 on avoiding recovery failures and the remaining weight (i.e., 0.001) on saving bandwidth overhead (Section 5.3.3). Figure 5.9 shows the impact of this parameter on the frame recovery performance of Tambur with the weight set to 0.9 (i.e., Tambur-0.9) and to 0.5. The improvement in non-recoverable frames for the two schemes are respectively 21.9% and 1.7%. The reduction in the bandwidth overhead is respectively 40.3% and 45.2%. By contrast, recall that Tambur leads to a 26.5% improvement in non-recoverable frames and reduces the bandwidth overhead by 35.1%. Reducing the value of the parameter reduces the frequency of recovering frames and increases the reduction in the bandwidth overhead. Videoconferencing service operators can use these weights as a knob to prioritize one metric over another.



(a) Non-recoverable frames



(b) Bandwidth overhead

Figure 5.10: CDFs for the percent of non-recoverable frames and the bandwidth overhead for the online evaluation.

5.4.4 Online evaluation

Next, we establish Tambur's potential to improve the QoE. To facilitate an easy comparison with the offline evaluation, we show the frequency of non-recoverable losses and the bandwidth overhead (as in Section 5.4.2) in Figure 5.10. On average, Tambur reduces the number of non-recoverable frames by 69% compared to Block-Within and 34% compared to Block-Multi. Tambur-0.9 reduces the number of non-recoverable frames by 65% compared to Block-Within and 26% compared to Block-Multi despite Block-Multi's much higher latency (Figure 5.11b). The results differ slightly from the offline evaluation at the lower percentiles because of setting the parameters of the channel based on average loss statistics over all the traces. This significantly reduced the frequency of calls with low loss rates where any coding scheme suffices to recover nearly all frames (i.e., sophisticated FEC schemes are unnecessary).

Tambur—which is conservative in risking recovery failures to save bandwidth—reduces the bandwidth overhead by 3% on average of the calls. In contrast, Tambur-0.9 reduces the bandwidth overhead by an average of over 8%. These results reflect both schemes reducing the bandwidth overhead significantly on some calls but only negligibly on many others, which is expected given the loss rates of most calls. Tambur-0.9's bandwidth savings are especially pronounced at the lower percentiles (e.g., 31% at the 10th percentile and 15% at the 20th percentile). Tambur-0.9 provides a win-win by both recovering more frames and saving bandwidth despite the online evaluation reflecting out-of-sample performance for its neural network, which was trained offline over the production traces. The results further validate the trade-off between the bandwidth overhead and recovering frames discussed in Section 5.4.3.

Next, we examine the percent of non-rendered frames in Figure 5.11a; recall that fewer frames are rendered than recovered due to inter-frame dependencies. Tambur reduces the frequency of failing to render framescompared to Block-Multi and Block-Within by an average of 28% and 73%, respectively. Tambur does worse than Block-Multi at the tail, but this only occurs after all schemes have a failure rate above 23%. Thus, all schemes should employ more redundancy. Tambur-0.9 decreases the frequency of failing to render frames by an average of

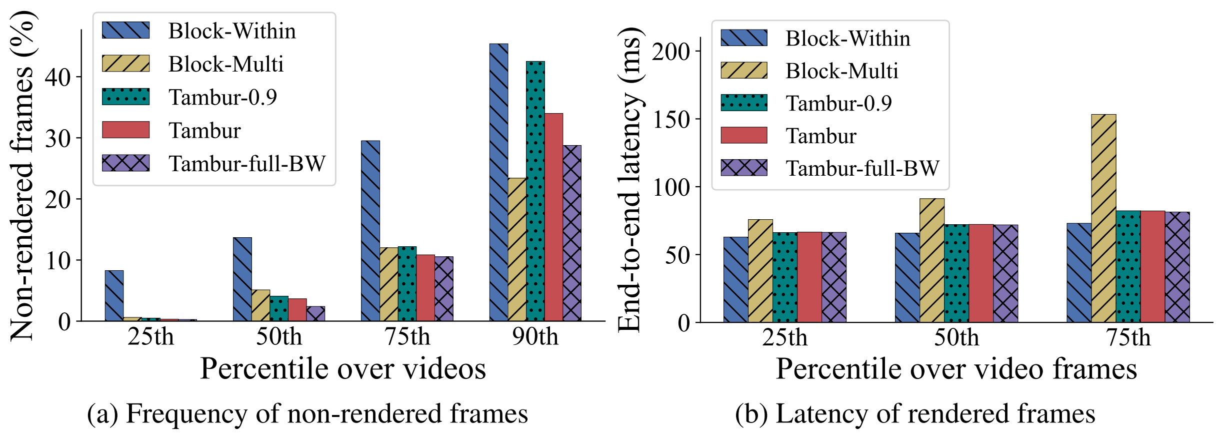


Figure 5.11: Tambur renders significantly more frames than Block-Multi and with lower latency. Tambur's modestly higher latency than Block-Within is more than offset by the improvement in rendering frames.[10]

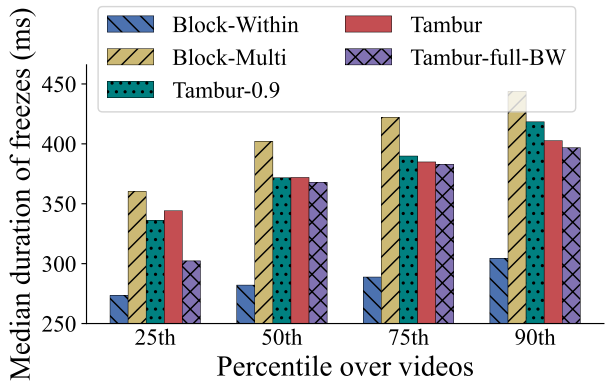
70% and 20% compared to Block-Within and Block-Multi, respectively. Tambur-0.9 modestly increases the frequency by 1% at the 75th percentile compared to Block-Multi. Overall, the rate of rendering frames can be improved while simultaneously reducing the bandwidth overhead for most calls. The results are the first to establish the benefits of streaming codes when there are inter-frame dependencies.

Figure 5.11b shows that the end-to-end latency is within the upper limit of approximately 150ms for all schemes. Block-Within's latency is slightly lower due to a shorter encode/decode time and always recovering rendered frames using the parity of the same frame (see Figure 5.13 and Figure 5.14 in Appendix 5.5.4); Tambur decodes 87% of frames without extra frames versus 88% for Block-Within, so the extra latency from the waiting for extra frames should really be compared to Block-Within failing to decode at all. We argue that Tambur's small cost (e.g., an extra 1.7ms to encode and 3.4ms to decode at the median) is worthwhile due to substantial improvements across the remaining QoE metrics. We also note that our implementation of Tambur's streaming code is not yet optimized for fast encoding/decoding; hence, we believe it can be significantly faster. Our goal is to establish that Tambur's streaming code is practical enough for videoconferencing applications.

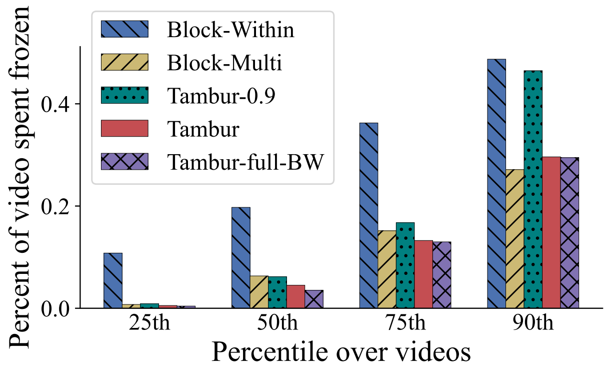
Recall from Figure 1.3 that Tambur reduces the frequency of freezes by 78% and 26% compared to Block-Within and Block-Multi, respectively, and Tambur-0.9 reduces the frequency of freezes by 75% and 17% compared to the two respective baselines. Figure 5.12a shows that Tambur and Tambur-0.9 each reduce the median duration of freezes compared to Block-Multi by 30ms on average. Tambur and Tambur-0.9 each have a 90ms longer median duration of freezes than theBlock-Within because Block-Within has over 300% more freezes than Tambur does. Many of the extra freezes are short, reducing Block-Within's median value to below Tambur's.

Tambur-0.9 reduces the cumulative duration of freezes by an average of 69% compared to Block-Within. The cumulative duration of freezes is 17% lower for Block-Multi than for Tambur-0.9 despite Tambur-0.9 having on average 11% shorter median durations of freezes and 17% fewer freezes. While the combined effect of the frequency and duration of freezes on the QoE

[10]We omit the 90th percentile since over 10% of frames are not rendered.



(a) Median duration of freezes



(b) Percent of video spent frozen

Figure 5.12: Tambur has a higher median duration of freezes than Block-Within but a significantly smaller cumulative duration of freezes because Tambur has 78% fewer freezes than Block-Within (Figure 1.3). Tambur has a lower cumulative and median duration of freezes than Block-Multi.

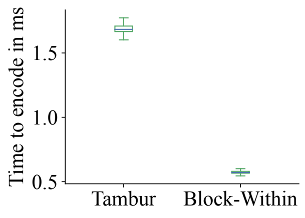
for Block-Multi and Tambur-0.9 are similar, recall that Tambur-0.9 also improves the bandwidth overhead and renders more frames for most traces. As such, we expect Tambur-0.9 to provide an overall higher QoE. Tambur has an average of 77% and 28% shorter cumulative durations of freezes than Block-Within and Block-Multi, respectively, which is a clear win. Tambur, Tambur-0.9, and Tambur-full-BW exhibit higher cumulative durations of freezes at the tail than Block-Multi. We argue that this matters less because the tail QoE is already bad, indicating that all schemes needed more bandwidth overhead. Appendix 5.5.5 explains how this phenomenon is an artifact of the implementation and includes our proposed a solution.

The benefits across QoE metrics of Tambur, Tambur-full-BW, and Tambur-0.9 suggest a markedly improved QoE compared to Block-Within and Block-Multi. Without using ML to reduce the bandwidth overhead, Tambur-full-BW offers a substantial improvement over the two baselines. Tambur and Tambur-0.9 progressively trade off improvements in loss recovery with bandwidth overhead. Overall, the results illustrate a Pareto frontier of the benefits of streaming codes across the QoE metrics that could be studied further in future work.

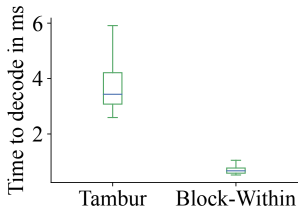
5.5 Appendix

5.5.1 Recovering a burst with Tambur's streaming code

Consider a burst of length b starting in frames i and delay constraint τ. Suppose all frames before the burst have been decoded. First, the received symbols of $P[i], \ldots, P[i+b-1]$ as well as $P[i+b], \ldots, P[i+\tau]$ are used to decode the lost symbols of $V[i], \ldots, V[i+b-1]$. Second, each $U[j]$ for $j \in \{i, \ldots, i+b\}$ is decoded using $P[j+\tau]$. In both steps, decoding follows from solving a system of linear equations.



(a) Encoding



(b) Decoding

Figure 5.13: The encoding and decoding times are modest.

5.5.2 Tambur's streaming code's flow network

The graph of the flow network at a high level represents each $P[i]$ that may be used in decoding with a node with an edge into nodes corresponding to each of $U[i]$, $U[i-\tau], V[i], \ldots, V[i-\tau]$, where one unit of flow represents decoding one symbol. The flow network is small (i.e., at most $(5\tau+3)$ vertices and $(2\tau^2+11\tau+5)$ edges for $\tau = 3$). Therefore, the time to solve it is negligible compared to solving the system of linear equations.

5.5.3 Parameters of the GE channel

To set the parameters of the GE channel for the offline evaluation, we first identify settings that match several aggregate statistics of the production traces as follows. The probability of transitioning from the bad state to the good state (respectively, vice versa) is the mean over traces of one divided by the mean length of bursts (respectively, guard spaces) in frames. The probability of loss in the bad state equals the mean over traces of the multi-frame burstiness. The probability of loss in the good state is then set so that the expected loss rate matches the mean loss rate over traces given the other three parameters. To ensure our results hold for varying network conditions, we then draw the values for each of the four parameters uniformly at random from intervals around these values (rounded to increments of 0.05) as follows. The probability of transitioning from the good state to the bad state and vice versa are distributed as Uniform(0, 0.05) and Uniform(.75, .9), respectively. The probability of loss in the good and bad states are distributed as Uniform(0, 0.05) and Uniform(0.05, 1), respectively.[11]

5.5.4 Encoding and decoding overheads

We compare the encoding and decoding time for Tambur with that of Block-Within, which is the fastest of all the baselines (Figure 5.13). As seen in Figure 5.13, the time to encode and decode is comparable to Block-Within and is only a small fraction of the end-to-end latency budget of 150 ms. The median times for encoding are 1.7ms and .6ms for Tambur and Block-Within, respectively, whereas decoding takes 3.4ms and .7ms for Tambur and Block-Within, respectively. Because Tambur operates over multiple frames of varying sizes, encoding and decoding times are slightly longer and more variable. Our implementation of Tambur requires a fixed amount of memory of approximately 575 KB during encoding and decoding.

[11]The results were similar when we varied the ranges.

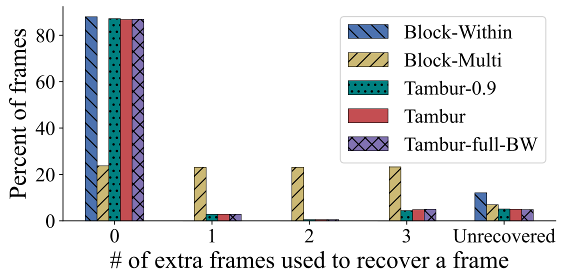


Figure 5.14: Tambur recovers nearly as many frames as Block-Within using no extra frames and also recovers more overall. Block-Multi recovers an approximately equal number of frames using 0, 1, 2, and 3 extra frames.

But times for encoding and decoding are just a small component of the end-to-end latency. The 50ms one-way delay and the number of extra frames used in decoding (see Figure 5.14) have more pronounced effects. Recall that each additional frame used in adds approximately 33 ms to the end-to-end latency, so using fewer extra frames is faster. Tambur does not decode within the same frame only 1% more frequently than Block-Within, which cannot use extra frames in decoding. Tambur uses extra frames to decode only 8% of the time. Block-Multi decodes 24%, 23%, 23%, and 23% of frames with $0, 1, 2$, and 3 extra frames, respectively. Each extra frame adds ≈ 33 ms to the end-to-end latency.

5.5.5 Tail duration of freezes

Recall from Figure 5.12b that Tambur, Tambur-0.9, and Tambur-full-BW have higher tail durations of freezes than Block-Multi. The reason for the poor performance is threefold. First, Tambur, Tambur-0.9, and Tambur-full-BW fail to render more frames at the tail, as was discussed in Section 5.4.2. Second, the sender generates a keyframe (often ending a freeze) once it learns of recovery failures. Because Block-Within can only recover a frame using the parity packets within the same frame, a keyframe is requested 3 frames sooner (i.e., ≈ 100 ms faster) than when Tambur (or Tambur-0.9) is used. Many of the 78% of freezes under the Block-Within where Tambur does not freeze are therefore short and shift the entire distribution of cumulative duration of freezes for Block-Within, including the tail; if we added 0ms freezes for Tambur (or Tambur-0.9) for these instances, their distributions would likewise shift. Third, encoding across multiple frames can make it harder to recover a keyframe triggered by a freeze of several lost frames. This phenomenon does not impact Block-Within and affects Block-Multi less than any of Tambur, Tambur-0.9, and Tambur-full-BW (e.g., does not affect on Block-Multi whenever the keyframe is in the first position within the block of $(\tau + 1) = 4$ frames). The phenomenon also contributes to a difference in the frequency of recovered frames (Figure 5.10a) and rendered frames (Figure 5.11a). There is a natural solution that is outside of the scope of this work. When the sender triggers a new keyframe due to a loss, it should stop taking linear combinations of frames from before the new keyframe. Doing so will strictly (a) increase the frequency of displaying frames and (b) decrease the mean and median duration of freezes. It will benefit Tambur, Tambur-0.9, and Tambur-full-BW the most, but it will also improve Block-Multi to a lesser extent.

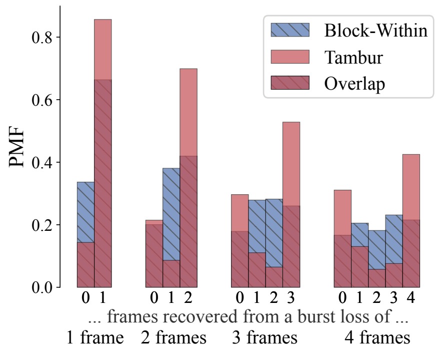


Figure 5.15: Given the same bandwidth budget as Block-Within, Tambur is more likely to recover all or zero frames from a burst loss over production traces.

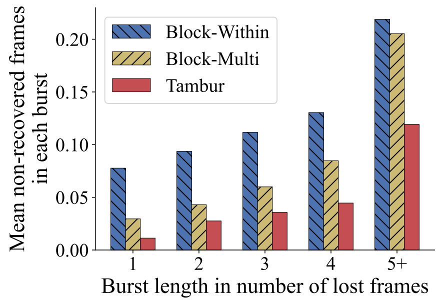


Figure 5.16: Given the same bandwidth budget as Block-Within/Block-Multi, Tambur provides greater improvement for longer bursts over an emulated network.

5.5.6 Analysis of recovering bursts

Next, we evaluate Tambur's capabilities for recovering bursts of packets across multiple frames; to do so fairly, we must fix the bandwidth overhead, so "Tambur" refers to Tambur-full-BW for the remainder of Section 5.5.6. Figure 5.15 shows the distribution of the number of packets recovered for each burst length (in frames) for the offline evaluation. In Figure 5.15, the distribution of the number of packets recovered for each burst length (in frames) is shown. Bursts encompassing 2, 3, and 4 frames constitute 23%, 7%, and 3.3% of all lossy events, respectively. For these losses, Tambur recovers all lossy frames 66.8%, 103%, and 97.3% more frequently than Block-Within. For the longer (less frequent) bursts of lengths 3 and 4, when the bandwidth overhead is insufficient, Tambur fails to recover any frames 65.9% and 87% more frequently than the Block-Within. This follows from the Block-Within being more likely to recover some (but not all) of the frames when there is insufficient bandwidth overhead to recover all losses. In contrast, when the bandwidth overhead is insufficient to recover a burst in its entirety, streaming codes are likely to fail to recover all of the frames. However, note that the overall performance of Tambur is still better than the Block-Within: Tambur recovers 21.8%, 12.4%, and 2.3% more frames than the Block-Within for bursts of 2,3, and 4 frames, respectively. Tambur also outperforms Block-Within in recovering losses limited to a single frame, as parity packets sent with later frames can be used in recovery. In short, Tambur performs significantly better for bursts across up to 3 frames than Block-Within and offers more modest gains for bursts across 4 frames.

We also evaluate Tambur's effectiveness at recovering bursts in the online evaluation. Because the loss of a single packet of a frame means that the frame is "lost" under our definition of a burst, longer bursts usually only involve being in the bad state for one, two, or sometimes three frames. We consider the mean number of frames recovered among a burst encompassing 1, 2, 3, 4, or greater than 4 frames in Figure 5.16. Tambur reduces the frequency of non-recoverable frames by 70.5%, 68.0%, and 65.8% compared to Block-Within over bursts of 2, 3, and 4 frames respectively. Tambur reduces the frequency of non-recoverable frames by 35.8%, 40.3%, and 47.4% compared to Block-Multi over bursts of 2, 3, and 4, respectively.

Chapter 6

Learning-augmented streaming codes for variable-size frames under partial bursts

Recall that the streaming codes designed in Chapters 2, 3, and 4 applied to burst losses of entire frames. However, Chapter 5 showed that network conditions that lose only some packets for each frame in a burst are also prevalent. The streaming model for variable-size frames presented in Chapter 2 is incompatible with such losses, motivating the need for updating the model. Thus, we generalize the streaming model for variable-size frames to accommodate such losses in Section 6.1. We aim to design high-rate streaming codes for the new model. To do so, we decompose the code design into two components. First, a building block construction for a streaming code given any choice of how to split a frame into a component recovered at its deadline and a component recovered strictly before its deadline. We introduce such an approximately rate-optimal building block construction in Section 6.2 Second, a policy to determine how to split the frames. We use a linear program to determine how to optimally split frame symbols in the offline setting in Section 6.3. Combining the linear program with the building block construction leads to an approximately rate-optimal offline code. But the goal is to design online codes. Thus, we present online streaming codes for three parameter regimes that are optimal up to a trivial factor in Section 6.4. However, Section 6.5 then establishes for all remaining parameter regimes that there is a nontrivial gap between the rate of optimal online codes and the offline-optimal-rate despite the lossless-delay being zero. This deviates from the model studied in Chapter 3 (i.e., without partial bursts), where we presented an online code matching the optimal rate of offline codes for the setting where the lossless-delay is zero. To construct an online code for the remaining parameter regimes, we replace the linear program with a learning-based approach to determine how to split frame symbols in Section 6.6. Doing so yield an approximately rate-optimal online code. Finally, Section 6.7 adds a constraint into the model to capture a requirement of real-world systems: a maximum size of a transmitted packet. Section 6.7 then illustrates how to adjust our constructions to satisfy this requirement with minimal changes in rate.

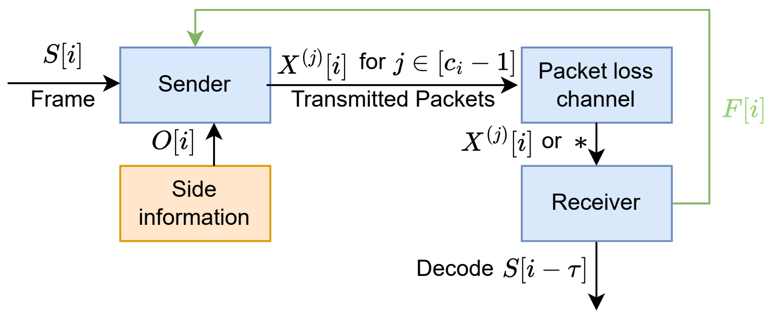


Figure 6.1: Overview of the proposed streaming model. Multiple packets are transmitted over the channel for each frame. The packet loss channel allows for partial bursts.

6.1 System model

We now extend the streaming model for variable-size frames from Chapter 2, as illustrated in Figure 6.1. There are a positive number, t, of time slots. During the ith time slot, the sender obtains a frame, $S[i]$, of k_i independent random symbols of a finite field, $\mathbb{F}$, where k_i is a non-negative integer between 0 and a maximum value, m. We refer to $k_0, \ldots, k_t$ as the "frame-size sequence." The sender sends c_i *transmitted packets*, $X^{(0)}[i], \ldots, X^{(c_i-1)}[i]$, consisting of $n_i^{(0)}, \ldots, n_i^{(c_i-1)}$ symbols, respectively. This change to the model allowing multiple packets to be transmitted over the channel for each frame 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

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

respectively. The rate is defined as in Chapter 2 as the ratio of frame symbols to transmitted symbols:

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

The transmitted packets are sent over the following lossy channel.

Loss model: The loss model comprises bursty losses (affecting one or more consecutive time slots) followed by a guard space 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, $l \in \{i, \ldots, i+b-1\}$, a $\ell_l \in (0, 1]$ fraction of the transmitted packets can be lost. That is, an arbitrary $\lceil \ell_l c_l \rceil$ transmitted packets of $X[l]$ are lost.

Further, the length and the fraction of packets lost of 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. 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

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

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

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

Feedback: During any time slot, i, the sender may obtain feedback from the receiver for updating the length of a burst starting in time slot i and the fraction of transmitted packets lost during each time slot of the burst (i.e., b_i and $(\ell_i, \ldots, \ell_{i+b_i-1})$). The feedback can be viewed as the receiver conservatively estimating how lossy the network conditions will be based on prior losses. For most time slots, no feedback is received. In this case, the parameters are retained as is. At times, there could be an underestimation of the losses, and that could lead to frames not being recovered. In videoconferencing, due to compression, video frames are typically dependent on each other. Hence not recovering a frame 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. This is modeled via a binary value ζ_i. It is 0 by default and set to 1 to indicate that the τ frames 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 frames.

6.1.1 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 frames 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, $O_i = \langle k_l, b_l, \ell_l, \zeta_l \big| l \in \{i+1, \ldots, t\} \rangle$. In the online setting, side information is the output of a predictive model (see Section 6.6 for details). During time slot i, the sender uses the prior frames and side information, O_i, to encode as

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

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 frame, $S[i]$, 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 frame is recovered within τ time slots. Specifically, for any burst starting in time slot j of length b_j that encompasses time slot i,

$$S[i] = Dec\big(S[0], \ldots, S[j-1], Y[j], \ldots, Y[i+\tau], \\ k_0, \ldots, k_{i+\tau}\big). \tag{6.1}$$

We note that under variable-size frames, the sizes of the frames are needed for decoding (see Chapter 2, 3, and 4). This is handled by adding a small header containing the sizes of the previous τ frames. We also point out that our constructions do not require the full memory allowed under the model because they do not use any information about frames and transmitted packets more than 2τ time slots in the past.

6.1.2 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.

Finally, we define a notation related to burst losses which will be used in the construction and multiple proofs. 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 will next define some conventions followed in the rest of the paper. The final $(\tau+1)$ frames are assumed to be of size 0, and t is at least $(\tau+1)$; this can be satisfied by appending $(\tau+1)$ frames of size 0 without affecting the optimal rate.

The number of transmitted packets for each frame is always an integer between 0 and $2m$. Thus, for any time slot, i, ℓ_i can be restricted to be a rational number. We then define natural numbers q_i and h_i so that $\ell_i = q_i/h_i$ is in simplest form. We expect $(h_i - 1)$ to be negligible for practical settings; this can be guaranteed by considering higher values of ℓ_i with smaller values for h_i. 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 $(h_i - 1)$. The cost of zero-padding is at most $\sum_{i=0}^{t} 2(h_i - 1)$ (because replication could be used).

6.2 A Building Block Construction

This section develops an approximately rate-optimal construction for any parameters, τ and t, frame-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 frames 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 the ith time slot, $S[i]$ 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 $V[i-\tau], \dots, V[i]$ 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 $V[j], \dots, V[j+b_j-1]$ 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 6.2 provides an overview of encoding.

Recovery (high-level description). Consider a burst of length b_i starts in time slot i where $Y[i : i+b_i-1]$ are received. First, for $j \in \{i, \dots, i+\tau-1\}$, 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 $j \in \{i+\tau, \dots, i+\tau+b_i-1\}$, $P'[j]$ 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 $U[j-\tau]$. Figure 6.3 provides an overview of loss recovery.

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 $i \in \{\tau, \dots, t-\tau\}$, $S[i]$ 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 $j \in \mathbb{B}_i, l \in \{j, \dots, j+b_j-1\}$, 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 frames are recovered using parity symbols sent after time slot $(i+\tau)$ by setting $u_l = k_l = 0$ (for Equations 6.2 and 6.3 below), leading to

$$d^{(i,j,l)} = \min\left((1-\ell_l)n_l,\ k_l - u_l\ell_l + \sum_{r=j}^{l-1}\left(k_l - u_l\ell_r - d^{(i,j,r)}\right)\right). \tag{6.2}$$

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

$$\sum_{l=j+b_j}^{j+\tau-1} p_l + \sum_{l=j}^{j+b_j-1} d^{(i,j,l)} \geq \sum_{l=j}^{j+b_j-1} (k_l - u_l\ell_l). \tag{6.3}$$

Next, u_i is increased until Equation 6.3 is satisfied and $h_i \big| u_i$. Then $S[i]$ is split into:

$$U[i] = S_{0:u_i-1}[i] \tag{6.4}$$
$$V[i] = S_{u_i:k_i-1}[i]. \tag{6.5}$$

The number of parity symbols of $X[i+\tau]$ is defined using as

$$p_{i+\tau} = \ell_i u_i + pad_{i+\tau}, \tag{6.6}$$

where $pad_{i+\tau}$ is the smallest integer to ensure $h_{i+\tau} \big| p_{i+\tau}$. The symbols of $P[i+\tau]$ themselves 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 [39, 43] (as from [11]). 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 $U[i-\tau]$:

$$\begin{aligned}
U^*[i-\tau] &= (U[i-\tau], 0^{\langle m-u_{i-\tau}\rangle},) \\
V^*[j] &= (V[j], 0^{\langle m-v_j\rangle}) \\
P^{(*)}[i] &= (AU^*[i-\tau])_{0:p_i-1} \\
P'[i] &= \sum_{j=0}^{\tau} H_j V^*[i-\tau+j] \\
P[i] &= (P^{(*)}[i] + P'[i]). \qquad (6.7)
\end{aligned}$$

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 $j \in [c_i - 1]$, let the jth v_i/c_i, u_i/c_i, and p_i/c_i symbols of $V[i], U[i]$, and $P[i]$, be denoted as $V^{(j)}[i], U^{(j)}[i]$, 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 frame of a length $(t-i+1)$ call and completes initialization.

Next, Theorem 8 shows that the lossless-delay and worst-case-delay constraints are met by the building block construction.

Theorem 8. *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.*

Proof. At a high level, for any time slot i and burst over $X[i : i+b_i-1]$, we show that $V[i : i+b_i-1]$ are recovered by time slot $(i+\tau-1)$ using the received frame symbols and received symbols of $P'[i : i+\tau-1]$. We then show for $l \in \{i, \ldots, i+b_i-1\}$ that $U[l]$ is recovered during time slot $(l+\tau)$ using the received symbols of $U[l]$ and $P^{(*)}[l+\tau]$.

Appendix 6.8.2 has a detailed proof.

□

Recall that the field size requirement of the code design is based on two components. First, it is at least $2m$ to construct A. Second, it needs to be large enough for the m-MDS code, and this field size may be large. Next, we present a randomized construction with a small field size. When a burst loss occurs during time slot i, the probability that $S[i : i+b_i]$ will not be recovered will be at most ϵ for a small $\epsilon > 0$. The construction involves choosing each entry of $H_0, \ldots, H_{\tau-1}$ independently and uniformly at random from a field of size $\big|\mathbb{F}\big| \geq (\tau m/\epsilon)$ (i.e., polynomial in the input parameters and $1/\epsilon$). We call this construction $\big(\tau, t, K, \mathrm{Z}, \mathcal{L}, B\big)$-RandomCode($\epsilon$) and show it satisfies the lossless-delay and worst-case-delay constraints next.

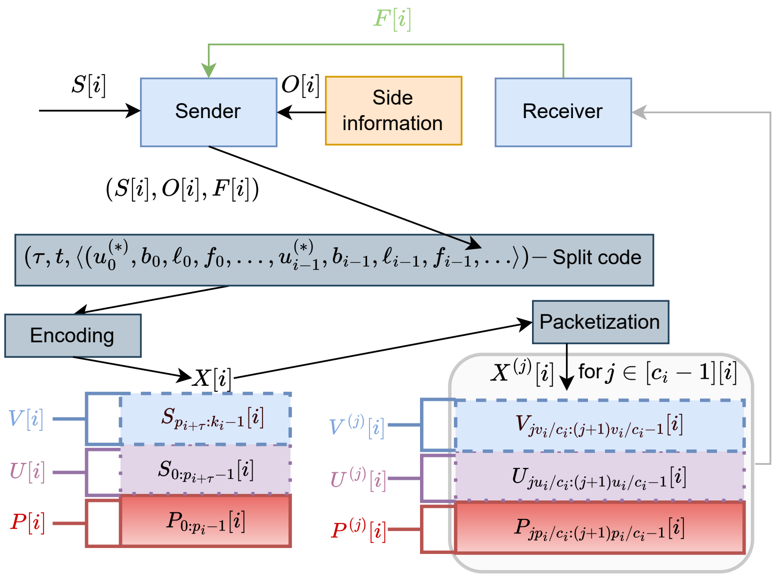

$F[i]$
$S[i]$
Sender
$O[i]$
Side information
Receiver
$(S[i], O[i], F[i])$
$(\tau, t, \langle (u_0^{(*)}, b_0, \ell_0, f_0, \dots, u_{i-1}^{(*)}, b_{i-1}, \ell_{i-1}, f_{i-1}, \dots \rangle)-$ Split code
Encoding
Packetization
$X[i]$
$X^{(j)}[i]$ for $j \in [c_i - 1][i]$
$V[i]$
$U[i]$
$P[i]$
$S_{p_{i+\tau}:k_i-1}[i]$
$S_{0:p_{i+\tau}-1}[i]$
$P_{0:p_i-1}[i]$
$V^{(j)}[i]$
$U^{(j)}[i]$
$P^{(j)}[i]$
$V_{jv_i/c_i:(j+1)v_i/c_i-1}[i]$
$U_{ju_i/c_i:(j+1)u_i/c_i-1}[i]$
$P_{jp_i/c_i:(j+1)p_i/c_i-1}[i]$

Figure 6.2: Overview of encoding.

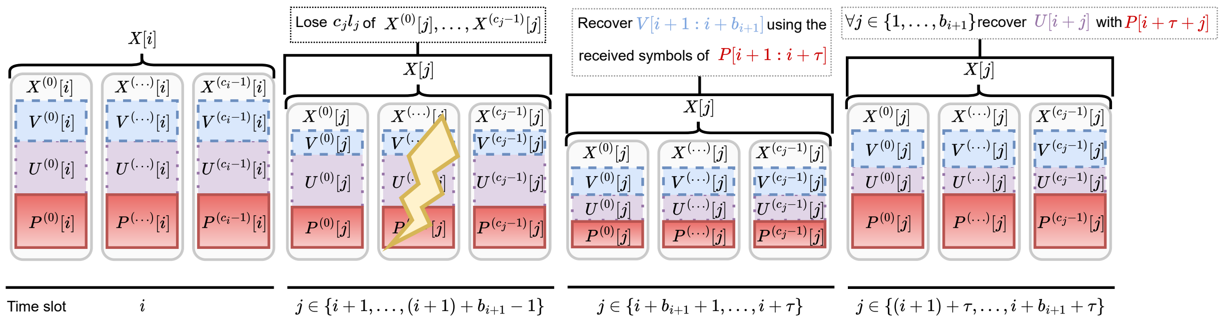

$X[i]$
$X^{(0)}[i]$ $X^{(\dots)}[i]$ $X^{(c_i-1)}[i]$
$V^{(0)}[i]$ $V^{(\dots)}[i]$ $V^{(c_i-1)}[i]$
$U^{(0)}[i]$ $U^{(\dots)}[i]$ $U^{(c_i-1)}[i]$
$P^{(0)}[i]$ $P^{(\dots)}[i]$ $P^{(c_i-1)}[i]$
Lose $c_j l_j$ of $X^{(0)}[j], \dots, X^{(c_j-1)}[j]$
Recover $V[i+1 : i+b_{i+1}]$ using the received symbols of $P[i+1 : i+\tau]$
$\forall j \in \{1, \dots, b_{i+1}\}$ recover $U[i+j]$ with $P[i+\tau+j]$
$X[j]$
$X^{(0)}[j]$ $X^{(\dots)}[j]$ $X^{(c_j-1)}[j]$
$V^{(0)}[j]$ $V^{(\dots)}[j]$ $V^{(c_j-1)}[j]$
$U^{(0)}[j]$ $U^{(\dots)}[j]$ $U^{(c_j-1)}[j]$
$P^{(0)}[j]$ $P^{(\dots)}[j]$ $P^{(c_j-1)}[j]$
Time slot
i
$j \in \{i+1, \dots, (i+1)+b_{i+1}-1\}$
$j \in \{i+b_{i+1}+1, \dots, i+\tau\}$
$j \in \{(i+1)+\tau, \dots, i+b_{i+1}+\tau\}$

Figure 6.3: Illustration of loss recovery under the $(\tau, t, K, Z, \mathcal{L}, B, W)$-Split Code for a burst over $X[i+1 : i+b_{i+1}-1]$.

Lemma 21. *Consider any $\tau, t, K, Z, \mathcal{L}, B, W$. The $(\tau, t, K, Z, \mathcal{L}, B)$-RandomCode($\epsilon$) (a) satisfies the lossless-delay constraint for all frames, and (b) for any burst starting in time slot $i \in [t]$ satisfies the worst-case-delay constraint for $S[i : i + b_i - 1$ with probability at least $(1 - \epsilon)$.*

Proof sketch. At a high level, the proof follows from showing that the set of symbols from the system of linear equations used to recover $V[i : i+b_i-1]$ under $(\tau, t, K, Z, \mathcal{L}, B, W)$-Split Code by time slot $(i + \tau - 1)$ constraints remain full rank, thus sufficing to recover $S[i : i + b_i - 1]$ under $(\tau, t, K, Z, \mathcal{L}, B)$-RandomCode($\epsilon$). Then recovery of $U[i : i + b_i - 1]$ follows identically to Theorem 8. A complete proof is presented in Appendix 6.8.3

□

Next, we use a simple example to highlight how $(\tau, t, K, Z, \mathcal{L}, B)$-RandomCode($\epsilon$) is likely to require a reasonable field size and provide sufficient loss-recovery capabilities for practical videoconferencing applications. For a 2000 kbps video 30 fps, a reasonable setting of m may be

2^{16}. Suppose there is a a 50 ms one-way delay, then the end-to-end latency is $(33.3\tau + 50)$ ms. One may set $\tau = 3$ so that this latency is ≈ 150 ms to satisfy industry recommendations [84]. Therefore, if we choose $\epsilon = 2^{14}$, we obtain a field size of 2^{32}, which requires only 4 bytes. The recovery error probability is then less than 0.01% due to using random matrices. We argue that 0.01% is likely to be negligible compared to the small error of unpredictable changes to the network conditions due to real-world events.

6.3 Offline codes

In this section, we design an offline approximately rate-optimal construction in three steps. First, we present Algorithm 4. 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 4 is combined with $\big(\tau, t, K, \mathrm{Z}, \mathcal{L}, B, W\big)$-Split Code.

At a high level, the variables of the LP used in Algorithm 4 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 frames 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 during 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 decoding deadlines. Finally, Constraint 5 reflects that w_i never exceeds the number of lost symbols of $S[i]$.

We will show that Algorithm 4 yields an upper bound on the optimal rate subject to a certain condition on feedback introduced below. Then Theorem 9 shows that this upper bound is nearly tight.

Condition on feedback : A reset must occur whenever increasing the fraction of transmitted packets that could be lost. Formally, for any time slot $i \in [t - \tau] \setminus \{0\}$ where $\ell_i > \ell_{i-1}$, ζ_i must be set to 1.

Lemma 22. *For any $\tau, t, K, \mathrm{Z}, \mathcal{L}, B$, if Algorithm 4outputs $\big\langle w_i \big| i \in [t]\big\rangle$, the offline optimal rate under the condition on feedback is at most*

$$\big(\sum_{i=0}^{t} k_i\big) / \big(-2(t-\tau)(\tau-1) + \sum_{i=0}^{t-\tau} k_i + w_i\big). \tag{6.9}$$

Proof. The proof is shown in Appendix 6.8.4. □

Remark 10. *Algorithm 4 runs in $O(poly(t\tau))$ time since the LP has $O(t\tau)$ constraints and variables.*

Combining Algorithm 4 with the building block construction (Section 6.2) yields an approximately rate-optimal code.

Algorithm 4 Computes $\langle w_i \big| i \in [t] \rangle$ of an approximately rate optimal code.

```
Input: $(\tau, t, K, \mathcal{L}, B)$
For $i \in [t-\tau]$:
  If $\sum_{j=i+1}^{i+\tau} \zeta_i > 0$:
    Set $k_i = 0$.
Minimize $\sum_{i=0}^{t-\tau} p_{i+\tau}^{(IP)}$ subject to:
  1. $\forall j \in [\tau - 1], p_j^{(IP)} = 0$.
  2. $\forall j \in [t-\tau], p_{j+\tau}^{(IP)} \geq 0$.
  3. $\forall i \in [t-\tau], l \in \{i, \dots, i + b_i - 1\}$,
```

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

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

$$\sum_{l=i+b_i}^{i+\tau-1} p_l^{(IP)} + \sum_{l=i}^{i+b_i-1} d_{i,l} \geq \sum_{l=i}^{i+b_i-1} (k_l - p_{l+\tau}^{(IP)}) \tag{6.8}$$

```
  5. $\forall j \in [t-\tau], k_j \ell_j \geq p_{j+\tau}^{(IP)}$.
Output: $\left\langle p_i^{(IP)} \big| i \in [t] \right\rangle$
```

Theorem 9. *For any $\tau, t, K, Z, \mathcal{L}$, suppose Algorithm 4 outputs $\langle w_i | i \in [t] \rangle$. For any $i \in [t-\tau]$ where $\sum_{j=i+1}^{i+\tau} \zeta_j \geq 1$, let $w_i = k_i \ell_i$. Then the rate of the corresponding Split Code is less than the optimal rate under the condition on feedback by at most*

$$\big(\sum_{i=0}^{t-\tau}(2\tau + q_i + h_{i+\tau} - 4)\big)/\big(\sum_{i=0}^{t} k_i\big). \tag{6.10}$$

Proof. The proof is shown in Appendix 6.8.5. □

As an example of applying Theorem 9, 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.

Finally, we bound the cost of removing the condition on feedback. For each time slot i that the condition is violated (i.e., $\ell_i > \ell_{i-1}$ without a rest), the denominator of Equation 6.9 decreases and numerator of Equation 6.10 increases by up to $\big((\ell_i - \ell_{i-1}) \sum_{j=i-\tau+1}^{i} k_j \ell_j\big)$.

6.4 Online rate-optimal codes

This section presents online rate-optimal constructions for three parameter regimes when there are no resets (i.e., $\zeta_i = 0$ for all $i \in [t]$): (a) Regime b_1 where $\tau > 1$ and $b_i = 1$ for all $i \in [t]$, (b) Regime b_τ where $b_i = \tau$ for all $i \in [t]$, and (c) Regime $b_{\tau+1}$ where $b_i > \tau$ and $\ell_i < 1$ for all $i \in [t]$. Later, Section 6.5 shows in all other parameter regimes that online rate-optimal codes cannot match the optimal rate of offline ones, even without resets.

6.4.1 Online optimal codes for Regime b_1

We introduce a systematic construction called "Regime b_1-Code." During the ith time slot, $X[i] = (S[i], P[i])$. One symbol is sent per transmitted packet (i.e., $c_i = n_i$). All that remains is to define $P[i]$. At a high level, $P[i]$ is chosen to be full-rank linear combinations of all of the symbols of $S[i-\tau : i]$. Its size, p_i, is as small as possible to ensure $S[i-\tau]$ is recovered by time slot i.

Next, we define some notation and then use it to define $P[i]$. Let $[I_m | A^T]^T$ be the generator matrix of a $[m(\tau+2), m]$ systematic MDS code, where T denotes transpose. Let A comprise $m \times m$ block submatrices, $A_0, \ldots, A_\tau$. For $i < \tau, p_i = 0$, and no parity symbols are sent. For $i \in [t-\tau] \setminus [\tau - 1]$, during time slot i, we define

$$p_i = \max\big(0, n_{i-\tau} - \lceil n_{i-\tau}\ell_{i-\tau}\rceil - k_{i-\tau} - \sum_{l=i-\tau+1}^{i-1} p_l\big) \tag{6.11}$$

$$V[j] = (S[j], 0^{\langle m-k_j \rangle}) \tag{6.12}$$

$$P[i] = \big(\sum_{j=i-\tau}^{i} A_{j-(i-\tau)} V[j]\big)_{0:p_i - 1}. \tag{6.13}$$

Decoding involves solving a full-rank system of linear equations.

Next, we establish that the proposed construction satisfies the lossless-delay and worst-case-delay constraints.

Lemma 23. *For any $\tau, t, K, \mathcal{L}$ under Regime b_1, the Regime b_1-Code lossless-delay and worst-case-delay constraints over the channel.*

Proof. The lossless-delay constraint is met by sending $S[i]$ in $X[i]$.

The final τ frames are known to be of size 0. Consider any burst loss in $X[i]$ for some $i \in [t-\tau]$. For each $l \in \{\max(\tau, i), \dots, i+\tau\}$ and $j \in \{l-\tau, \dots, l\} \setminus \{i\}$, $S[j]$ is received and used to compute $V[j]$.[1] This yields

$$P^*[l] = \sum_{j \in \{l-\tau,\dots,l\}\setminus\{i\}} A_{j-(l-\tau)} V[j].$$

Recall that

$$A_{i-(l-\tau)} V[i] = P[l] - P^*[l].$$

Overall, for $l \in \{i+1, \dots, i+\tau\}$, the first p_l symbols of $A_{i-(l-\tau)}V[i]$ are available. In addition, some $(n_i - \lceil n_i \ell_i \rceil)$ symbols of $[I, A_\tau^T]^T V[i]$ are received. Therefore,

$$(n_i - \lceil n_i \ell_i \rceil) + \sum_{l=i+1}^{i+\tau} p_l$$

symbols of $[I, A^T]^T V[i]$ are received and $(m - k_i)$ symbols of $V[i]$ are known in advance to be zero-padding.

By Equation 6.11 that

$$(p_i - \lceil p_i \ell_i \rceil) + k_i(1-\ell_i) + \sum_{l=i+1}^{i+\tau} p_i \geq k_i.$$

By the MDS property, $V[i]$ can be recovered, yielding $S[i]$. □

Finally, we show that the proposed construction is trivially close to having an optimal rate.

Lemma 24. *Consider any $\tau, t, K, \mathcal{L}$ under Regime b_1, and let $N^{(OPT)}$ be the number of symbols sent under a rate-optimal offline code. Then the Regime b_1-Code's rate is at least*

$$\Big(\sum_{i=0}^{t} k_i\Big) / (N^{(OPT)} + 3\tau(t-\tau) + 1).$$

Proof sketch. At a high level, we compare the number of symbols sent under Regime b_1-Code to the number modeled as being sent under Algorithm 4. We demonstrate how to modify the values of $p_i^{(IP)}$ for $i \in [t]$ to match p_i for $i \in [t]$ while increasing $\sum_{i=0}^{t} p_i^{(IP)}$ by at most $(t - \tau + 1)$. A complete proof is included in Appendix 6.8.6.

□

[1]For $j \in [\tau - 1]$, we consider $V[-j] = 0^{\langle m \rangle}$.

6.4.2 Online optimal codes for Regime b_τ

We introduce a systematic construction called "Regime b_τ-Code." During the ith time slot, $(S[i], P[i])$. One symbol is sent per transmitted packet, leading to $c_i = n_i$. It suffices to define $P[i]$. At a high level, $P[i]$ is chosen to be full-rank linear combinations of the symbols of $S[i-\tau]$ and $S[i]$. Its size, p_i, is as small as possible to ensure $S[i-\tau]$ is recovered by time slot i.

Next, we define some notation and then use it to define $P[i]$. Let $[I_m | A^T]^T$ be the generator matrix of a $[3m, m]$ systematic MDS code. Let A_0 and A_1 be disjoint $m \times m$ block submatrices of A. For $i < \tau, p_i = 0$, and no parity symbols are sent. For $i \in [t-\tau] \setminus [\tau - 1]$, during time slot i, we define

$$p_i' = \max\big(0, n_{i-\tau} - \ell_{i-\tau} n_{i-\tau} - k_{i-\tau}\big) \tag{6.14}$$

$$p_i = p_i' + \mathbb{1}[p_i' \mod h_i](h_i - (p_i' \mod h_i)) \tag{6.15}$$

$$V[i] = (S[i], 0, \ldots, 0)$$

$$P[i] = \big(A_0 V[i-\tau] + A_i V[i]\big)_{0:p_i - 1}. \tag{6.16}$$

Decoding follows from solving a full-rank system of linear equations. Clearly, $w_i = p_i$.

Next, we establish that the proposed construction satisfies the lossless-delay and worst-case-delay constraints.

Lemma 25. *For any $\tau, t, K, \mathcal{L}$ under Regime b_τ, the Regime b_τ-Code lossless-delay and worst-case-delay constraints over the channel.*

Proof. The lossless-delay constraint is met by sending $S[i]$ in $X[i]$.

The final τ frames are known to be of size 0. Consider any burst loss in $X[i]$ for some $i \in [t-\tau]$. Then $S[i+\tau]$ is received and used to compute $V[i+\tau]$ and $A_1 V[i+\tau]$. Thus,

$$\big(A_0 V[i]\big)_{0:p_{i+\tau}} = P[i+\tau] - \big(A_1 V[i+\tau\big)_{0:p_{i+\tau}}.$$

Additionally, $S[i-\tau]$ is known and used to compute $V[i-\tau]$ and $V[i-\tau]A_0$. Suppose p_i' parity symbols of $P[i]$ are received; for some $J \subseteq [p_i - 1]$ of size p_i', for all $j \in J$, $P_j[i]$ is received and used to compute

$$\big(A_1 V[i]\big)_j = P_j[i] - \big(A_0 V[i-\tau]\big)_j.$$

Finally, $k_i' = (n_i - \lceil n_i \ell_i \rceil - p_i')$ frame symbols of $S[i]$ are received. Recall by Equations 6.14 and 6.15 that

$$(p_i - \lceil p_i \ell_i \rceil) + k_i(1 - \ell_i) + p_{i+\tau} \geq k_i.$$

By the MDS property, $V[i]$ can be recovered, yielding $S[i]$. □

Finally, we show that the proposed construction is trivially close to having an optimal rate.

Lemma 26. *Consider any $\tau, t, K, \mathcal{L}$ under Regime b_τ, $N^{(OPT)}$ be the number of symbols sent under a rate-optimal offline code. Then the Regime b_τ-Code's rate is at least*

$$\Big(\sum_{i=0}^{t} k_i\Big) / \Big(N^{(OPT)} + 2(t-\tau)(\tau - 1) + \sum_{l=0}^{t-\tau} 2(q_l + h_{l+\tau} - 2)\Big).$$

Proof sketch. Let C be the code created by $\big(\tau, t, K, Z, \mathcal{L}, B, (w_0, \ldots, w_{t-\tau})\big)$-Split Code. By Theorem 9, it sends at most $2(t-\tau)(\tau-1) + \sum_{l=0}^{t-\tau}(q_l + h_{l+\tau} - 2)$ extra symbols compared to an offline rate-optimal code. We iteratively adjust C so that each $p_i^{(C)} = p_i$ while increasing the number of symbols sent for each time slot i by at most h_i. This bounds the difference in the rates of C and the Regime b_τ-Code. A complete proof is included in Appendix 6.8.7.

□

6.4.3 Online optimal codes for Regime $b_{\tau+1}$

We introduce a systematic construction called "Regime $b_{\tau+1}$-Code." During the ith time slot, $X[i] = (S[i], P[i])$. Next, we will define $P[i]$. No parity symbols are sent for the first τ time slots (i.e., $p_i = 0$ for $i < \tau$). For each $i \in [t] \setminus [\tau - 1]$, we define the number of parity symbols as

$$p_i' = k_{i-\tau}\ell_{i-\tau}/(1-\ell_i) \tag{6.17}$$

$$p_i = \lceil p_i \rceil + \mathbb{1}[\lceil p_i \rceil \mod h_i](h_i - \lceil p_i \mod h_i \rceil)). \tag{6.18}$$

A $[p_i + k_{i-\tau}, k_{i-\tau}]$ systematic MDS code is constructed for $S[i-\tau]$, leading to p_i parity symbols. These parity symbols are evenly spread over h_i transmitted packets of $X[i]$. The frame symbols of $S[i]$ are also evenly spread over these transmitted packets. Next, we show the proposed construction satisfies the lossless-delay and worst-case-delay constraints.

Lemma 27. *For any $\tau, t, K, \mathcal{L}$ under Regime $b_{\tau+1}$, the Regime $b_{\tau+1}$-Code lossless-delay and worst-case-delay constraints over the channel.*

Proof. The lossless-delay constraint is satisfied by sending $S[i]$ in $X[i]$.

Consider any $i \in [t]$. If $X[i]$ is lost in part of a burst, at least $(1-\ell_i)k_i$ frame symbols are recovered in $S[i]$ and at least $(1-\ell_{i+\tau})p_{i+\tau}$ parity symbols are received τ time slots later. As such, at least k_i symbols are received of an the $[p_{i+\tau} + k_i, k_i]$ MDS encoding of $S[i]$. By the MDS property, these symbols to recover $S[i]$. □

Next, we show that the proposed construction sends at most $(h_i - 1)$ extra symbols per time slot compared to a rate-optimal code (i.e., it is trivially close to being rate-optimal).

Lemma 28. *Consider any $\tau, t, K, \mathcal{L}$ under Regime $b_{\tau+1}$, and let $N^{(opt)}$ be the number of symbols sent under a rate-optimal offline code. Then the Regime $b_{\tau+1}$-Code's rate is at least*

$$\big(\sum_{i=0}^{t} k_i\big)/(N^{(opt)} + \sum_{l=0}^{t-\tau} h_i).$$

Proof. Recall that under Regime $b_{\tau+1}$, for any $i \in [t-\tau+1]$, $b_i > \tau$. Any code construction satisfying the lossless-delay and worst-case-delay constraints must recover from each $S[i]$ within τ time slots. So if a burst starts in time slot 0, $S[0]$ is recovered by time slot τ. Therefore, the code can recover from a burst of length $(\tau + 2)$ starting in time slot 0. If the code must recover from a burst of length $l \geq (\tau + 1)$ starting in time slot 0, then $S[0 : l - \tau - 1]$ must be recovered by time slot $(l-1)$. Therefore, it is as if nothing was lost until time slot $(l - \tau)$. Therefore, the code must be able to recover from a burst of length $(l+1)$ starting in time slot 0. Thus, inductively, the code must be able to recover from a burst of length t starting in time slot 0.

This establishes that any code, C, that sends $n_{C,i}$ symbols during time slot i must be able to recover when ℓ_i fraction of transmitted packets are dropped adversarially for any $i \in [t]$. The total number of symbols lost is at least $\sum_{i=0}^{t} \ell_i n_{C,i}$. At least $\sum_{i=0}^{t} k_i$ symbols must be received. Therefore, $\sum_{i=0}^{t}(1-\ell_i)n_{C,i} \geq \sum_{i=0}^{t} k_i$.

Combining Equations 6.17, 6.18, and $p_i = 0$ for $i < \tau$, and $k_i = 0$ for $i > (t-\tau)$ establishes

$$\begin{aligned}\sum_{i=0}^{t}(1-\ell_i)n_i &= \sum_{i=0}^{t}(1-\ell_i)(k_i+p_i)\\ &\leq \sum_{l=0}^{\tau-1} k_i + \sum_{i=\tau}^{t} h_i + (1-\ell_i)(k_i+p_i')\\ &= \sum_{l=0}^{\tau-1} k_i\\ &+ \sum_{i=\tau}^{t}(h_i + (1-\ell_i)(k_i + k_{i-\tau}\ell_{i-\tau}/(1-\ell_i)))\\ &\leq \sum_{i=0}^{t-\tau}(h_i+k_i) + \sum_{i=t-\tau+1}^{t} k_i\\ &= \sum_{i=0}^{t-\tau}(h_i+k_i).\end{aligned}$$

This concludes the result.

□

The proof of Lemma 28 also establishes that Regime $b_{\tau+1}$ is not bursty. Specifically, there are not bursts of losses across only a few consecutive time slots followed by guard spaces. Instead, arbitrary losses occur during every single time slot.

6.5 Gap between online and offline codes

Recall from Theorem 4 that in the setting where an entire channel packet (i.e., all transmitted packets) is lost or received, an online code exists that matches the optimal rate of offline codes. However, this is no longer true when only some transmitted packets are lost. The key distinction is that the optimal choice for how much information the received symbols of a transmitted packets should contain about the corresponding frame depends on the sizes of *future* frames. Consequently, even when there are no resets, there is a gap between the optimal online rate and offline rate for all but the three parameter regimes discussed in Section 6.4: Regime b_1, Regime b_τ, and Regime $b_{\tau+1}$. We formalize this result next in Theorem 10

Theorem 10. *For any τ and B outside of Regime b_1, Regime b_τ, and Regime $b_{\tau+1}$, the online-optimal-rate is strictly less than offline-optimal-rate.*

Proof sketch. We introduce two possible frame-size sequences. We show that there exists a non-negligible $\epsilon > 0$ so that in order to be within ϵ of the optimal offline rate on one sequence, an online code's rate must be lower than the optimal offline rate by at least ϵ on the second. A full proof is included in Appendix 6.8.8. □

6.6 Online approximately rate-optimal codes

We now present an online approximately rate-optimal construction. 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 6.2). 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 6.3), and (b) padded to be divisible by q_i. The construction is dubbed the "$\big(\tau, t, K, \mathrm{Z}, \mathcal{L}, B, W^{(O)}\big)$-Split ML Code."

Our result requires a few terms. Let the outputs of the ML model over time slots $0, \dots, (t-\tau)$ be $W^{(O)} = O_0, \dots, O_{t-\tau}$. For $i = 0, \dots, (t-\tau)$, let $W_i^{(Opt)}$ be the set of optimal values for $p_{i+\tau}^{(IP)}$ in Algorithm 4 with additional constraints that the variables for variables corresponding to earlier time slots are set according to $W^{(O)}$ (i.e., for $j \in [i-1]$ $p_{j+\tau}^{(IP)} = W_j^{(O)}$). For $i \in [t-\tau]$, the regret of the outputs of the ML model compared to the optimal values is

$$\mathcal{R}_i = \min_{w^{(Opt)} \in W_i^{(Opt)}} \big|O_i - w^{(Opt)}\big|, \mathcal{R}_{[t]} = (\mathcal{R}_0, \dots, \mathcal{R}_t) \tag{6.19}$$

For an arbitrary frame-size sequence, $K = (k_0, \dots, k_t)$, and feedback, $\mathcal{L} = (\ell_0, \dots, \ell_t), B = (b_0, \dots, b_t)$, and $\mathrm{Z} = (\zeta_0, \dots, \zeta_t)$, (all chosen offline without access to $W^{(O)}$), let $R^{(opt)}$ be the offline optimal rate under the condition on feedback from Section 6.3 and $R^{(on)}$ be a random variable (over the predictions of the ML model) reflecting the rate of the Split ML Code.

Theorem 11. *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 $\mathrm{E}[\mathcal{R}_i] \le \epsilon k_i$ and $t > log(1/\delta)/(2\epsilon_\dagger^2)$. Then with probability at least $(1-\delta)$,*

$$R^{(opt)} - R^{(on)} \le \epsilon + \big(\sum_{i=0}^{t} \epsilon_\dagger + 2\tau + h_i + q_i - 4\big)/\big(\sum_{i=0}^{t} k_i\big). \tag{6.20}$$

Proof. The proof is shown in Appendix 6.8.9. □

Consider the example of a videoconferencing call discussed after Theorem 9. If the call is sufficiently long, with probability $(1-\delta)$, $R^{(on)}$ is within $(0.01 + \epsilon + 0.00048 \cdot \epsilon_\dagger)$ of optimal.

Finally, we note that removing the condition on feedback leads to increasing Equation 6.20 for each $i \in \{\tau, \dots, t\}$ were $\ell_i > \ell_{i-1}$ by at most

$$\big((\ell_i - \ell_{i-1}) \sum_{j=i-\tau+1}^{i} k_j \ell_j\big)/\big(\sum_{j=0}^{t} k_j\big).$$

6.7 Maximum transmittable unit

To match the conventions used by real-world multimedia streaming applications, we can add a maximum transmittable unit (MTU) to the model. Specifically, we introduce a new parameter, "M," wherein each transmitted packet is of size at most M.

One can modify $\big(\tau, t, K, \mathrm{Z}, \mathcal{L}, B, W\big)$-Split Code to handle this extra requirement while sending negligibly few extra symbols per time slot as follows: Consider the ith time slot, wherein $X^{(j)}[i]$ is sent for $j \in [c_i - 1]$ for $c_i = h_i$,. Suppose any transmitted packet contains more than M symbols. Let c_i' be the smallest multiple of h_i so that $\lceil n_i/c_i' \rceil \leq M$. The symbols of $X[i]$ are spread over $(c_i' - 1)$ transmitted packets so that each transmitted packet contains the same ratio of symbols of $U[i], V[i], P[i]$ as were originally sent. This may require decreasing $V[i]$ to equal the nearest multiple of c_i', increasing $U[i]$ by the same amount, and then padding $U[i]$ until it is a multiple of c_i'. Similarly, $P[i]$ will be padded until it is a multiple of c_i'.

Next, we provide a simple example of the number of extra symbols needed to satisfy the requirement of the M. A typical value of M may be 1500 bytes. Suppose $\big|\mathbb{F}\big| \geq 8$. Consider videoconferencing at a standard 30 frames per second where the maximum possible size of a video frame is 2^{14} bytes. Then c_i' is never more than 11. Thus, the amount of extra padding is at most $2c_i' \leq 22$ bytes per frame. The overhead per frame is at most 22 bytes in this example; for a 2000 kbps videoconferencing call, this leads to an overhead of less than 0.27%.

6.8 Appendix

6.8.1 Additional notation

We define the set of possibilities for the received packets of $X[i]$ as $\mathcal{Y}_i$. For a burst starting in time slot j of length b_j and any $i \in \{j, \ldots, j + b_j - 1\}$, we define the set of possible received packets of $X[j:i]$ as $\mathcal{Y}_{j:i} = \mathcal{Y}_j \times \ldots \times \mathcal{Y}_i$.

To simplify the presentation of proofs, we also require that a burst starting in time slot i does not end sooner than a burst starting before time slot i. Formally, we require $(i + b_i) \geq (j + b_j)$ for all $j \in \mathbb{B}_i$. This requirement holds without loss of generality by the satisfaction of the worst-case-delay for any $j \in \mathbb{B}_i$ and any burst loss leading to $Y[j : j + b_j - 1] \in \mathcal{Y}_{j:j+b_j-1}$. Specifically, $S[j : i-1]$ are recovered by time slot $(i + \tau - 1)$ and suffice to obtain $X[j : i-1]$. The worst-case-delay is still satisfied even though now the burst spans time slots i through $(j + b_j - 1)$.

Next, we formalize decoding in terms of the normalized Shannon Entropy (i.e., the Shannon Entropy divided by the entropy of a random field element, $log(\big|\mathbb{F}\big|)$):

$$\mathcal{H}(S[i] \big| S[0], \ldots, S[i-1], k_i, Y[i]) = 0 \tag{6.21}$$

$$\begin{aligned}\mathcal{H}\big(S[i] \big| S[0], \ldots, S[j-1], Y[j], \ldots, Y[i+\tau],& \\ k_j, \ldots, k_{i+\tau}\big) = 0.&\end{aligned} \tag{6.22}$$

Equations 6.21 and 6.22 follow directly from the lossless-delay and worst-case-delay constraints, respectively. The sizes of the frames are assumed to be known for loss recovery (from the header), so they will be omitted from the entropy function henceforth. Let $\mathcal{I}(\cdot)$ be the normalized Mutual Information (MI): $\mathcal{I}(W; Z) = \mathcal{H}(W) - \mathcal{H}(W \big| Z)$.

Finally, formally define the term to capture how frames are split into a component recovered before its deadline and a component recovered at its deadline. For any $i \in [t-\tau]$, let $w_i =$

$$\max_{Y[i]\in\mathcal{Y}_i} \big(\mathcal{H}(S[i] \big| S[0:i-1], X[i+1], \dots, X[i+\tau-1], \\ Y[i], k_0, \dots, k_{i+\tau-1})\big) \tag{6.23}$$

6.8.2 Proof of Theorem 8

We start with an auxiliary Lemma.

Lemma 29. *For any $i \in [t-\tau]$, all $j \in \mathbb{B}_i$ where $\sum_{r=j}^{j+\tau-1} \zeta_r = 0$, and any $l \in \{j, \dots, i\}$, $d^{(i,j,l)} \geq (1-\ell_l)u_l$.*

Proof. First, $(1-\ell_l)n_l \geq (1-\ell_l)k_l \geq (1-\ell_l)u_l$.

For $l = j$, combining Equation 6.2 with $k_l \geq u_l$ yields

$$d^{(i,j,l)} \geq (k_l - u_l\ell_l) \geq (u_l - u_l\ell_l) = u_l(1-\ell_l) \geq 0. \tag{6.24}$$

For $l \in \{j+1, \dots, i\}$, the case necessitates that $j < i$. First, by the inductive hypothesis on $(l-1)$,

$$d^{(i,j,l-1)} \leq k_{l-1} - u_{l-1}\ell_{l-1} + \\ \sum_{r=j}^{l-2} k_r - u_r\ell_r - d^{(i,j,r)} \tag{6.25}$$

$$\sum_{r=j}^{l-1} d^{(i,j,r)} \leq \sum_{r=j}^{l-1} k_r - u_r\ell_r \tag{6.26}$$

where Equation 6.26 follows from rearranging terms. Thus, $d^{(i,j,l)} \geq (1-\ell_l)u_l$ by Equation 6.2 and the logic for $l = j$.

$\square$

For any $i \in [t]$, the lossless-delay is met as $S[i]$ is in $X[i]$.

Next, to show satisfaction of the worst-case-delay, we consider any burst starting in $i \leq [t-\tau]$. We need not consider $i > (t-\tau)$, as the final τ frames are known to be 0. If $\sum_{l=i+1}^{i+\tau-1} \zeta_l > 0$ then $S[i : i+b_i-1]$ need not be recovered, and the proof is concluded. Otherwise, $\sum_{l=i+1}^{i+\tau-1} \zeta_l = 0$. We show in two steps that each $S[i : i+b_i-1]$ is recovered within τ timeslots. First, the received symbols of $Y[i : i+b_i-1]$ and $P[i+b_i : i+\tau-1]$ are used to recover $V[i : i+b_i-1]$. Second, for $j \in \{i, \dots, i+b_i-1\}$ where $\sum_{l=i+1}^{j+\tau-1} \zeta_l = 0$, $U[j]$ is recovered in time slot $(i+\tau)$ with $P[i+\tau]$.

First, for $j \in \{i, \ldots, i+\tau-1\}$, $U[j-\tau]$ is used to compute $P^{(*)}[j]$ to determine $P'[j] = (P[j] - P^{(*)}[j])$ (by Equation 6.7). Recall from Equation 6.3 (and $v_l = (k_l - u_l)$)

$$\sum_{l=i+b_i}^{i+\tau-1} p'[l] + \sum_{l=i}^{i+b_i-1} d^{(j,i,l)} \geq \sum_{l=i}^{i+b_i-1} k_l - u_l \ell_l \tag{6.27}$$

$$\sum_{l=i+b_i}^{i+\tau-1} p'[l] + \sum_{l=i}^{i+b_i-1} d^{(j,i,l)} - (1-\ell_l)u_l \geq \sum_{l=i}^{i+b_i-1} v_l \tag{6.28}$$

where for $l \in \{i, \ldots, i+b_i-1\}$ the $(1-\ell_l)u_l$ symbols of $U[l]$ that are received are subtracted out (valid by Lemma 29).

Without loss of generality, we pretend that each $P[j]$ is padded with extra parity symbols to be $P^{(pad)}[j]$ of size m but these extra $(m - p[j])$ symbols are all lost. Then each $(V^*[j], P^{(pad)}[j])$ comprises $2m$ symbols. The number of received parity symbols exceeds the number of missing frame symbols. Thus, combining Equation 6.28 with Lemma 1 L1 of [11] shows $V[i]$ is recovered by time slot $(i+\tau-1)$ (e.g., by solving a system of linear equations). For $r = (i+1), \ldots, (i+b_i-1)$, by Lemma 29 and Equation 6.2,

$$\sum_{l=i}^{r-1} d^{(i,i+b_i-1,l)} \leq \sum_{l=i}^{r-1} k_l - u_l \ell_l \tag{6.29}$$

$$\sum_{l=i}^{r-1} d^{(i,i+b_i-1,l)} - u_l(1-\ell_l) \leq \sum_{l=i}^{r-1} v_l \tag{6.30}$$

Combining Equations 6.28 and 6.30 shows

$$\sum_{l=i+b_i}^{i+\tau-1} p'[l] + \sum_{l=r}^{i+b_i-1} (d^{(j,i,l)} - u_l(1-\ell_l) \geq \sum_{l=r}^{i+b_i-1} v_l \ell_l \tag{6.31}$$

Combining Equation 6.31 with Lemma 1 L1 of [11], $V[r]$ can be recovered by time slot $(i+\tau-1)$ (e.g., by solving a system of linear equations). After finishing iteration $r = (i+b_i-1)$, $V[i : i+b_i-1]$ have been recovered.

For $l = i, \ldots, (i+b_i-1)$ where $\sum_{r=i+1}^{l+\tau-1} \zeta_r = 0$, we now show that $U[l]$ is recovered by time slot $(i+\tau)$. During time slot $(l+\tau)$, $V[l : l+\tau]$ are available and are used to compute $P'[l+\tau]$, yielding $P^{(*)}[l+\tau] = (P[l+\tau] - P'[l+\tau])$. Then $P^{(*)}[l+\tau]$ comprises $p[l+\tau] \geq U[l]\ell_l$ linearly independent linear equations of the symbols of $U[l]$. Combining $P^{(*)}[l+\tau]$, the $(1-\ell_i)u[i]$ received symbols of $U[l]$, and the $(m-u[l])$ zeroes of $U^*[l]$ (padding) provides at least m symbols of the $[2m, m]$ maximum distance separable linear code. Thus, $U[l]$, is obtained by solving a system of linear equations. Both $V[l]$ and $U[l]$ are recovered by time slot $(l+\tau)$ for any $l \in [i, \ldots, i+b_i-1]$, so the worst-case-delay constraint is satisfied.

6.8.3 Proof of Lemma 21

For $i \in \{t-\tau+1, \ldots, t\}$, $k_i = 0$. Hence, $S[i]$ is known. Otherwise, consider $i \in [t-\tau]$. The lossless-delay constraint is satisfied by $S[i]$ being sent uncoded in $X[i]$. If $\sum_{l=i+1}^{i+\tau-1} \zeta_l > 0$ then

$S[i:i+b_i-1]$ need not be recovered, and the proof is concluded. Otherwise, $\sum_{l=i+1}^{i+\tau-1}\zeta_l=0$. We will show for any burst starting in time slot i of length b_i and any $j\in\{i,\dots,i+b_i-1\}$ such that $\sum_{l=i+1}^{j+\tau-1}\zeta_l=0$ that the probability of correctly recovering $S[i:j]$ within the worst-case-delay using $S[0:i-1],Y^{(C)}[i:j+\tau]$ is at least $(1-\epsilon)$. Recall that if $\sum_{l=i+1}^{j+\tau-1}\zeta_l>0$, $S[j]$ need not be recovered. It suffices to show that $V[i:i+b_i-1]$ are correctly recovered by time slot $(i+\tau-1)$; afterwards, $U[j]$ for $j\in\{i,\dots,i+b_i-1\}$ is recovered identically to the proof of Theorem 8 (respectively, not recovered if $\sum_{l=i+1}^{j+\tau-1}\zeta_l>0$).

If $\big(\tau,t,K,\mathrm{Z},\mathcal{L},B,W\big)$-Split Code were used, the only difference is the linear equations used to generate the parity symbols. Recall from the proof of Theorem 8 the missing symbols of $V[i:i+b_i-1]$ are obtained by solving a system of linear equations combining (a) received symbols of $S[0:i-1]$, (b) received symbols of $V[i:i+b_i-1]$, and (c) a set of $L=\sum_{j=i}^{i+b_i-1}\ell_j v_j$ parity symbols. The first two quantities are available to $\big(\tau,t,K,\mathrm{Z},\mathcal{L},B\big)$-RandomCode($\epsilon$). We will show that set of l corresponding parity symbols for $\big(\tau,t,K,\mathrm{Z},\mathcal{L},B\big)$-RandomCode($\epsilon$) can be used to recover the l missing symbols of $V[i:i+b_i-1]$. Without loss of generality, when $\big(\tau,t,K,\mathrm{Z},\mathcal{L},B,W\big)$-Split Code is instead used, $L=\{r_1,\dots,r_l\}$ where r_j is used as a pivot column in Gaussian Elimination to recover the jth missing symbol of $V[i+i+b_i-1]$. Suppose under $\big(\tau,t,K,\mathrm{Z},\mathcal{L},B\big)$-RandomCode($\epsilon$) Gaussian Elimination is used and sequentially $r_1,\dots,r_l$ is chosen as a pivot column to recover the next missing symbol. We bound the probability of failing to recover at least one symbol. To fail, the pivot column would need to be 0 in the position corresponding to the missing symbol. For each symbol, the probability of failure (i.e., that it cannot be used) is the same as the probability that it is 0 which occurs with probability at most $1/|\mathbb{F}|$. Applying a union bound shows that the probability of failing for at least one symbol is at most $l/|\mathbb{F}|$. Noting that $l\le b_i m\le\tau m$ and $|\mathbb{F}|\ge(\tau m/\epsilon)$ leads to a probability of failing of at most ϵ.

6.8.4 Proof of Lemma 22

Throughout Section 6.8.4, we assume for all $i\in[t]$ that $b_i\le\tau$. We begin by proving the result when there are no resets (i.e., $\sum_{i=0}^{t}\zeta_i=0$). Let code construction, C, be any offline construction that satisfies the lossless-delay and worst-case-delay constraints. Under C, $X^{(C)}[i]$ is sent during time slot $i\in[t]$ of size $n_{C,i}$ comprising $(c_{C,i}+1)$ transmitted packets. Then $Y^{(C)}[i]$ is the vector of received channel packets, and the total number of received symbols is $n_{C,i,Y^{(C)}}$. Appendix 6.8.4 shows some preliminary results for the proof. Appendix 6.8.4 incorporates relaxations to C and verifies their correctness. Appendix 6.8.4 proves additional properties due to the relaxations. Finally, Appendix 6.8.4 ties the results together to finish the proof

Proof preliminaries

We begin with defining a few terms for time slot $i\in[t-\tau]$. Will use the following terms:

$$w_i^{(C)}=\max_{Y^{(C)}[i]\in\mathcal{Y}_i}\big(\mathcal{H}(S[i]\big|S[0:i-1],\\ Y^{(C)}[i],X^{(C)}[i+1:i+\tau-1])\big) \tag{6.32}$$

$$p_i^{(C)}=n_i-k_i. \tag{6.33}$$

Lemma 30. *For any $i \in [t-\tau]$ such that $\mathbb{B}_i \neq \emptyset, \ell_i > 0$ the number of symbols sent during the $(i+\tau)$th time slot is at least*

$$n_{C,i+\tau} \geq \left\lceil w_i^{(C)} + k_{i+\tau} \right\rceil .$$

Proof. At a high level, by Equation 6.32, at least $w^{(C)}[i]$ symbols' worth of information are needed to recover $S[i]$ that are unavailable prior to time slot $(i+\tau)$. They must be recovered during time slot $(i+\tau)$ due to the worst-case-delay, leading to at least $\lceil w^{(C)}[i] \rceil$ symbols being sent in $X^{(C)}[i+\tau]$. The lossless-delay constraint for $S[i+\tau]$ necessitates an additional $k_{i+\tau}$ symbols be sent in $X^{(C)}[i+\tau]$.

By Equation 6.32, there is a $Y^{(C)}[i] \in \mathcal{Y}_i$ so that

$$\left(\mathcal{H}(S[i] \big| S[0:i-1], Y^{(C)}[i], X^{(C)}[i+1:i+\tau-1])\right) = w_i^{(C)}. \tag{6.34}$$

Recall that $S[0:i-1], Y^{(C)}[i:i+\tau-1]$ are available by time slot $(i+\tau-1)$. By Equation 6.22 and the chain rule,

$$\mathcal{H}(S[i] \big| S[0:i-1], Y^{(C)}[i], X^{(C)}[i+1:i+\tau]) = 0. \tag{6.35}$$

Thus for some $j \in \mathbb{B}_i$ and $Y^{(C)}[j:j+b_j-1] \in \mathcal{Y}_{j:j+b_j-1}$,

$$n_{C,i+\tau} \geq \mathcal{H}(X^{(C)}[i+\tau]) \geq \tag{6.36}$$
$$\mathcal{H}(X^{(C)}[i+\tau] \big| S[0:i-1], Y^{(C)}[i:j+b_j-1], X^{(C)}[j+b_j:i+\tau-1]) = \tag{6.37}$$
$$\begin{aligned}&\mathcal{H}(X^{(C)}[i+\tau] \big| S[0:i-1], Y^{(C)}[i:j+b_j-1], X^{(C)}[j+b_j:i+\tau-1]) + \\ &\mathcal{H}(S[i] \big| S[0:i-1], Y^{(C)}[i:j+b_j-1], X^{(C)}[j+b_j:i+\tau]) =\end{aligned} \tag{6.38}$$
$$\mathcal{H}(S[i], X^{(C)}[i+\tau] \big| S[0:i-1], Y^{(C)}[i:j+b_j-1], X^{(C)}[j+b_j:i+\tau-1]) \geq \tag{6.39}$$
$$\begin{aligned}&\mathcal{H}(S[i] \big| S[0:i-1], Y^{(C)}[i:j+b_j-1], X^{(C)}[j+b_j:i+\tau-1]) + \\ &\mathcal{H}(X^{(C)}[i+\tau] \big| S[0:i], Y^{(C)}[i:j+b_j-1], X^{(C)}[j+b_j:i+\tau-1]) \geq\end{aligned} \tag{6.40}$$
$$\begin{aligned}&\mathcal{H}(S[i] \big| S[0:i-1], Y^{(C)}[i], X^{(C)}[i+1:i+\tau-1]) + \\ &\mathcal{H}(X^{(C)}[i+\tau] \big| S[0:i], X^{(C)}[i+1:i+\tau-1]) \geq\end{aligned} \tag{6.41}$$
$$w_i^{(C)} + \mathcal{H}(X^{(C)}[i+\tau] \big| S[0:i], X^{(C)}[i:i+\tau-1]) \geq \tag{6.42}$$
$$w_i^{(C)} + \mathcal{H}(X^{(C)}[i+\tau] \big| S[0:i+\tau-1]) \geq \tag{6.43}$$
$$w_i^{(C)} + k_{i+\tau} \tag{6.44}$$

Equation 6.37 comes from conditioning reducing entropy; in Equation 6.38, the extra added term is 0 due to Equation 6.35; Equation 6.39 comes from applying the chain rule to Equation 6.38; Equation 6.40 comes from applying the chain rule to Equation 6.39; Equation 6.41 comes from conditioning reducing entropy; Equation 6.42 comes from conditioning reducing entropy Equation 6.34; Equation 6.43 comes from $X^{(C)}[i:i+\tau-1]$ being a function of $S[0:i+\tau-1]$ and conditioning reducing entropy; Equation 6.44 comes from Equation 6.21. Finally, combining Equations 6.36 and 6.44 establishes the result.

□

Next, we bound the amount of information the parity symbols of each transmitted packet provide about earlier frames.

Lemma 31. *For any $i \in [t], j \in [i-1]$, the amount of information $X^{(C)}[i]$ can provide about dropped symbols of earlier frames is bounded by*

$$\begin{aligned}&\mathcal{I}(X^{(C)}[i]; S[j : \min(j+b_j, i)-1] \big| S[0 : j-1])\\ &\leq (n_{C,i} - k_i)\end{aligned}$$

Proof.

$$\begin{aligned}
&\mathcal{I}(X^{(C)}[i]; S[j : \min(j+b_j, i)-1] \big| S[0 : j-1]) = && (6.45)\\
&\mathcal{H}(X^{(C)}[i] \big| S[0 : j-1]) - \\
&\mathcal{H}(X^{(C)}[i] \big| \min(j+b_j, i)-1) \leq && (6.46)\\
&\mathcal{H}(X^{(C)}[i] \big| S[0 : j-1]) - \mathcal{H}(X^{(C)}[i] \big| S[0 : i-1]) \leq && (6.47)\\
&(n_{C,i} - k_i) && (6.48)
\end{aligned}$$

where Equation 6.46 comes from the definition of Mutual Information; Equation 6.47 comes from conditioning reducing information; Equation 6.48 comes from $X^{(C)}[i]$ having at most $n_{C,i}$ symbols and Equation 6.21. □

Lemma 31 will later be used to show for a burst starting in time slot j how much redundancy transmitted packets received after the burst can provide to help in loss recovery.

We show for any $i \in [t-\tau]$ there is a burst starting in time slot $j \in \mathbb{B}_i$ such that all symbols of $S[j : i-1]$ and all but $w_i^{(C)}$ symbols of $S[i]$ must be recovered by time slot $(i+\tau-1)$.

Lemma 32. *Consider any $i \in [t-\tau]$ and any $j \in \mathbb{B}_i$. Then*

$$\begin{aligned}\min_{Y^{(C)}[j:j+b_j-1] \in \mathcal{Y}_{j:j+b_j-1}} &\big(\mathcal{I}(S[j : i]; Y^{(C)}[j : j+\tau-1] \big| \\ &S[0 : j-1])\big) \leq \sum_{l=j}^{i} k_l - w_l^{(C)}.\end{aligned}$$

Proof.

$$\begin{aligned}
&\mathcal{H}(S[j : i] \big| S[0 : j-1], Y^{(C)}[j : j+\tau-1]) = && (6.49)\\
&\sum_{l=j}^{i} \mathcal{H}(S[l] \big| S[0 : l-1], Y^{(C)}[j : j+\tau-1]) \geq && (6.50)\\
&\sum_{l=j}^{i} \mathcal{H}(S[l] \big| S[0 : l-1], Y^{(C)}[l : l+\tau-1]) \geq && (6.51)\\
&\sum_{l=j}^{i} \mathcal{H}(S[l] \big| S[0 : l-1], Y^{(C)}[l], X^{(C)}[l+1 : l+\tau-1]) \geq && (6.52)\\
&\sum_{l=j}^{i} w_l^{(C)}. && (6.53)
\end{aligned}$$

where Equation 6.50 follows from the chain rule; Equation 6.51 comes from conditioning reducing entropy; Equation 6.52 comes from conditioning reducing entropy; Equation 6.53 follows from Equation 6.32.

We note that

$$\mathcal{I}(S[j:i]; Y^{(C)}[j:j+\tau-1] \big| S[0:j-1]) = \tag{6.54}$$

$$\mathcal{H}(S[j:i]\big|S[0:j-1]) - \mathcal{H}(S[j:i]\big|S[0:j-1], Y^{(C)}[j:j+\tau-1]) \tag{6.55}$$

$$\geq \sum_{l=j}^{i} k_l - w_l^{(C)} \tag{6.56}$$

where Equation 6.55 follows from the definition of Mutual Information; Equation 6.56 follows from the sizes of $S[j:i]$/independence of frames and Equation 6.53. $\square$

We rewrite the worst-case-delay constraint in terms of mutual information as follows.

Lemma 33. *Consider any* $i \in [t-\tau]$ *and any* $j \in \mathbb{B}_i$. *Then*

$$\min_{Y^{(C)}[j:j+b_j-1] \in \mathcal{Y}_{j:j+b_j-1}} \big(\mathcal{I}(S[j:i]; Y^{(C)}[j:i+\tau]\big| S[0:j-1])\big) = \sum_{l=j}^{i} k_l.$$

Proof.

$$\mathcal{I}(S[j:i]; Y^{(C)}[j:i+\tau]\big|S[0:j-1]) = \tag{6.57}$$

$$\mathcal{H}(S[j:i]\big|S[0:j-1]) - \mathcal{H}(S[j:i], \big|S[0:j-1], Y^{(C)}[j:i+\tau]) = \tag{6.58}$$

$$\sum_{l=j}^{i} k_l - \sum_{l=j}^{i} \mathcal{H}(S[l], \big|S[0:j-1], Y^{(C)}[l:l+\tau]) = \tag{6.59}$$

$$\sum_{l=j}^{i} k_l. \tag{6.60}$$

Where Equation 6.58 follows from the definition of mutual information; Equation 6.59 follows from independence of frames and the fact that the Equation 6.22 shows for any $l \in \{j, \dots, i\}$, $\mathcal{H}(S[l], \big|S[0:j-1], Y^{(C)}[l:l+\tau]) = 0$, leading to Equation 6.60. $\square$

Relaxations

We will use the following relaxations which may increase (but never decrease) the mutual information between received symbols under C and missing information. Consider any $i \in [t-\tau]$ and any $j \in \mathbb{B}_i$.

Relaxation 1.

$$\min_{Y^{(C)}[j:j+b_j-1]\in\mathcal{Y}_{j:j+b_j-1}} \big(\mathcal{I}(S[j:i];Y^{(C)}[i]\big| S[0:j-1],Y^{(C)}[j:i-1])\big) =$$

$$\min\big(n_{C,i,Y^{(C)}}, k_i - w_i^{(C)} + \sum_{r=j}^{i-1} k_r - w_r^{(C)} - \mathcal{I}(S[j:r];Y^{(C)}[r]\big|S[0:j-1],Y^{(C)}[j:r-1])\big).$$

Relaxation 2. For any $l \in \{j+b_j,\dots,j+\tau-1\}$,

$$\min_{Y^{(C)}[j:j+b_j-1]\in\mathcal{Y}_{j:j+b_j-1}} \big(\mathcal{I}(S[j:i];X^{(C)}[l]\big| S[0:j-1],Y^{(C)}[j:l-1])\big) =$$

$$\min\big(n_{C,l}-k_l, \sum_{r=j}^{i} k_r - w_r^{(C)} - \sum_{r=j}^{l-1} \mathcal{I}(S[j:\min(r,i)];Y^{(C)}[r]\big|S[0:j-1],Y^{(C)}[j:r-1])\big).$$

Lemma 34. *Relaxations 1 and 2 do not cause C to send extra symbols or violate any constraints.*

Proof. **Relaxation 1** . By the sizes of transmitted packets and conditioning reducing entropy, we know $\mathcal{I}(S[j:i];Y^{(C)}[i]\big|S[0:j-1],Y^{(C)}[j:i-1]) \le \mathcal{H}(Y^{(C)}[i]) \le n_{C,i,Y^{(C)}}$ (i.e., at most the size of $Y^{(C)}[i]$). We note

$$\sum_{l=j}^{i}\mathcal{I}(S[j:i];Y^{(C)}[l]\big|S[0:l-1],Y^{(C)}[j:l-1]) = \tag{6.61}$$

$$\sum_{l=j}^{i}\mathcal{I}(S[j:l];Y^{(C)}[l]\big|S[0:l-1],Y^{(C)}[j:l-1]) = \tag{6.62}$$

$$\mathcal{I}(S[j:i];Y^{(C)}[j:i]\big|S[0:j-1]) \le \tag{6.63}$$

$$\mathcal{I}(S[j:i];Y^{(C)}[j:j+\tau-1]\big|S[0:j-1]) \le \tag{6.64}$$

$$\left(\sum_{l=j}^{i} k_l - w_i^{(C)}\right) \tag{6.65}$$

where Equation 6.62 comes from the independence of $S[l+1:i]$ from $S[0:l], Y^{(C)}[j:l]$; Equation 6.63 holds because applying the chain rule for mutual information to it yields Equation 6.61; Equation 6.64 comes from $(j+\tau-1)\ge i$, the chain rule for mutual information, and non-negativity of mutual information; Equation 6.65 comes from Lemma 32. Combining

Equation 6.61 with Equation 6.65 establishes that the relaxation only maintains or increases the mutual information.

Relaxation 2 . By Lemma 31, $\mathcal{I}(S[j:i]; X^{(C)}[l] \big| S[0:j-1], Y^{(C)}[j:l-1])$ is at most $(n_{C,l} - k_l) = p_l^{(C)}$. Also,

$$\sum_{r=j}^{l} \big(\mathcal{I}(S[j:i]; Y^{(C)}[r] \big| S[0:j-1], Y^{(C)}[j:r-1]) = \tag{6.66}$$

$$\sum_{r=j}^{i} \big(\mathcal{I}(S[j:r]; Y^{(C)}[r] \big| S[0:j-1], Y^{(C)}[j:r-1]) + \sum_{r=i+1}^{l} \big(\mathcal{I}(S[j:i]; Y^{(C)}[r] \big| S[0:j-1], Y^{(C)}[j:r-1]) = \tag{6.67}$$

$$\mathcal{I}(S[j:i]; Y^{(C)}[j:l] \big| S[0:j-1]) \leq \tag{6.68}$$

$$\mathcal{I}(S[j:i]; Y^{(C)}[j:j+\tau-1] \big| S[0:j-1]) \leq \tag{6.69}$$

$$\left(\sum_{r=j}^{i} k_r - w_r^{(C)}\right) \tag{6.70}$$

where Equation 6.67 comes from the independence of $S[l+1:i]$ from $S[0:l], Y^{(C)}[j:l]$; Equation 6.68 comes from the chain rule for mutual information; Equation 6.69 comes from the chain rule of mutual information, the fact that $l \leq (j+\tau-1)$, and the non-negativity of mutual information. Equation 6.70 comes from Lemma 32.

□

The relaxations lead to a mutual information that depends on the sizes of frames and transmitted packets, not the symbols that are sent under C themselves.

Relaxation 3. For any $i \in [t], (c_{C,i} + 1) = n_{C,i}$.

Lemma 35. *Relaxation 3 does not cause C to send extra symbols or violate any constraints.*

Proof. If there are no losses, this change is irrelevant. Otherwise, all considered bounds (under the relaxation) when $X^{(C)}[i]$ experiences loss apply to any choice for packetization. The only way that packetization effects the bounds is if it changes the total number of received symbols, with the greedy choice being to receive as many symbols as possible for each transmitted packet.

Relaxation 3 has no change for $i > (t-\tau)$. We apply the change for $i = 0, \ldots, (t-\tau)$; this does not alter the total number of symbols sent (or even the total number sent during any time slot).

Next, we show that this change does not effect decoding. When $\ell_i n_{C,i}$ is an integer, then the minimum possible number of symbols lost over $X^{(C)}[i]$ is $\ell_i n_{C,i}$ and that is what is lost in the worst case.

Otherwise, exactly $\lceil \ell_i n_{C,i} \rceil$ symbols are lost in the worst case. For any $(c_{C,i}+1)$ transmitted packets sent in $X^{(C)}[i]$, the largest $l = \lceil \ell_i (c_{C,i}+1) \rceil$ could be lost, which contain in total at least $\lceil n_{C,i} l / (c_{C,i}+1) \rceil \geq \lceil \ell_i n_{C,i} \rceil$ symbols by the pigeonhole principle and transmitted packets containing an integral number of symbols. Thus, one symbol per transmitted packet leads to as many symbols being received as possible, which maximizes their utility. $\square$

Relaxation 4 . $\forall l \in [t], \mathcal{I}(S[l]; X^{(C)}[l] \big| S[0:l-1]) = k_l$ whenever $n_{C,l} \geq k_l$.
Relaxation 5 . $\forall l \in \{\tau, \ldots, t\}$,

$$\mathcal{I}(S[l-\tau]; X^{(C)}[l] \big| S[0:l-\tau-1], Y^{(C)}[l-\tau:l-1]) = \\ \min(p_l^{(C)}, \mathcal{H}(S[l-\tau] \big| S[0:l-\tau-1], Y^{(C)}[l-\tau:l-1])$$

This only increases the mutual information by Lemma 31.

Useful identities of relaxed code

Corollary 7. *Adjusting C so that $p_j^{(C)} = 0$ for $j < \tau$ and for any $i \in [t-\tau]$,*

$$p_{i+\tau}^{(C)} = w_i^{(C)} \tag{6.71}$$

leads to at most $2(t-\tau)(\tau-1)$ extra symbols being sent.

Proof. By Equation 6.21, for all $i \in [t], n_{C,i} \geq k_i$. For $i \in [t]$, we will adjust $n_{C,i}$ but maintain $n_{C,i} \geq k_i$. This ensures that the lossless-delay constraint is met under relaxation 4.

When a burst of length b_0 starts in $X^{(C)}[0]$, it is recovered by time slot $(\tau + b_0 - 1)$ then $S[0:\tau+b_0-1]$ are available by the same time slot $(\tau+b_0-1)$ by Lemma 33 and relaxation 4. In addition, $n_{C,l} \geq k_l$ for $l \in [b_0-1]$ by Equation 6.21, so a worst case burst drops at least $k_l \ell_l$ symbols of $X^{(C)}[l]$. We apply relaxations 1-5.

For $l \in [\tau-1]$ we move all but k_l symbols of $X^{(C)}[l]$ to $X^{(C)}[\tau]$. If $X^{(C)}[\tau]$ is received, this move only improves loss recovery. If $X^{(C)}[\tau]$ is involved in a loss starting during time slot τ, $S[0:\tau-1]$ are available, so the move does not harm loss recovery. If a burst starts in time slot $j \in \mathbb{B}_\tau$ then the move only improves loss recovery; if an extra symbol is lost in $X^{(C)}[\tau]$ due to rounding, this can only occur if (a) at least one extra symbol would have been lost in $X^{(C)}[j:\tau-1]$ anyway or (b) $\ell_\tau < \ell_j$, in which case moving more symbols of $X^{(C)}[\tau]$ reduces the number lost. Given relaxation 1, the change only improves loss recovery. If a burst starts in $X^{(C)}[0]$, the changes have only improved loss recovery because strictly more symbols are received; for $j \in [b_0]$, any $\ell_j k_j$ symbols that are lost could have been lost before and now additional symbols are received. This also updates $w_0^{(C)} = \ell_0 k_0$ but does not change $w_j^{(C)}$ for any $j > 0$.

For $z = 0, \ldots, b_0 - 1$, we move all but $(k_{\tau+z} + \ell_z k_z)$ symbols of $X^{(C)}[\tau+z]$ to $X^{(C)}[\tau+z+1]$. If a burst starts during or before time slot z then this does not change which symbols are received.[2]. So loss recovery is unchanged for $S[0:z-1], S[z+1:z+b_z-1]$. By Relaxation 5, the worst-case-delay is satisfied for $S[z]$. If a burst starts during time slot $(z+1)$ then $S[z]$ is recovered before the burst. If $b_z = \tau$ then the change causes strictly more symbols to be received and otherwise the exact same symbols are received. Either way, loss recovery will

[2]Except for another burst that may perhaps start in time slot $(z+\tau+1)$, but for that symbols sent before the burst are useless for loss recovery

proceed unchanged for $S[z+1:z+b_0]$. If a burst starts strictly after time slot $(z+1)$ during time slot l, then by relaxations 1 and 2 all received symbols of $X^{(C)}[l:l+\tau-1]$, including the received symbols of $X^{(C)}[z+\tau:z+\tau+1]$, are maximally useful for loss recovery. The only way fewer symbols are available compared to before the change is if the burst starts by time slot $(z+\tau)$ and is of length at least 2 time slots and extra losses occur due to rounding for how many transmitted packets are lost in partial bursts. So we send two extra symbols in $X^{(C)}[r]$ for $r \in \{z+\tau+2,\ldots,z+2\tau\}$ so that each provides one symbol's worth of information about $S[r-\tau]$ under relaxation 5.

The changes may have increased $w_{b_0}^{(C)}$ but not its loss recovery (relaxation 5). They also have not increased $w_j^{(C)}$ for $j > b_0$.

We prove by induction that we can alter C to obey Equation 6.71 at the cost of $2(\tau-1)$ symbols per time slot; there is no cost for the final $(\tau+1)$ time slots where the frames are each of size 0.

Suppose for the inductive hypothesis that

$$\forall j \in [i_*], p_{j+\tau}^{(C)} = w_j^{(C)} \tag{6.72}$$

and loss recovery for $S[0:i_*]$ within τ time slots is guaranteed . The inductive hypothesis has been shown to hold for $i_* = (b_0-1)$.

Now we apply induction for $i=(i_*+1)$. We know by Lemma 30 that $p_{i+\tau}^{(C)} \geq w_i^{(C)}$ before we altered C. During earlier time slot $i' < i$, having moved symbols to $X^{(C)}[i'+\tau]$ could only decrease $w_i^{(C)}$. Only moving symbols from $X^{(C)}[i+\tau-1]$ to $X^{(C)}[i+\tau]$ may have increased $w_i^{(C)}$, but doing so would maintain the inequality $p_{i+\tau}^{(C)} \geq w_i^{(C)}$. Recall that by the IH $S[0:i-1]$ are still recovered by time slot $(i+\tau-1)$. We set

$$w_i^{(C)} = \max_{Y^{(C)}[i:i+b_i-1]\in\mathcal{Y}_{i:i+b_i-1}} \left(\mathcal{H}(S[i] \middle| S[0:i-1], Y^{(C)}[i:i+b_i-1], X^{(C)}[i+b_i:i+\tau-1])\right) \tag{6.73}$$

under the relaxations, which can only increase $w_i^{(C)}$, ensuring $S[0:i-1]$ are still recovered by time slot $(i+\tau-1)$. This involves letting C potentially remove information about $S[i]$ from $X^{(C)}[i+1:i+\tau-1]$. We will show that loss recovery is still accomplished despite the change under the considered relaxations.

For any burst starting in time slot $j \in \mathbb{B}_i$ where $j < i$ the change only increases loss recovery capabilities of $S[j:i-1]$ by time slot $(i+\tau-1)$ under our relaxations, so they are still recovered by then. To show loss recovery for $S[i:j+b_j-1]$, it suffices to show loss recovery for the worst-case burst starting in time slot i, as $(j+b_j) \leq (i+b_i)$. Under such a burst, all received symbols of $Y^{(C)}[i:i+\tau-1]$ are maximally useful under relaxations 1 and 2. Recall that by "worst-case" we just mean "drops as many symbols as possible" due to relaxation 1. Before resetting $w_i^{(C)}$, $S[i]$ was recovered by time slot $(i+\tau)$ and $w_i^{(C)}$ symbols must have been recovered at time slot $(i+\tau)$, so $p_{i+\tau}^{(C)} \geq w_i^{(C)}$. After the change, all but $w_i^{(C)}$ symbols of $S[i]$ are recovered before time slot $(i+\tau)$. Then $S[i]$ is still recoverable by time slot $(i+\tau)$ by relaxation 5. Now we can condition on $S[0:i]$ being recovered. We want to show worst-case loss recovery for $S[i+1:i+b_i-1]$.

Before the change, for $l \in \{i, \ldots, i + b_i - 1\}$

$$\mathcal{I}(S[i:l]; Y^{(C)}[i:i+b_i-1], X^{(C)}[i+b_i:l+\tau] \big| S[0:i-1]) = \sum_{r=i}^{l} k_r$$

due to loss recovery under the IH. Recall

$$\mathcal{I}(S[i:l]; Y^{(C)}[i:i+b_i-1], X^{(C)}[i+b_i:l+\tau] \big| S[0:i-1]) = \tag{6.74}$$
$$\mathcal{I}(S[i]; Y^{(C)}[i:i+b_i-1], X^{(C)}[i+b_i:i+\tau] \big| S[0:i-1]) + \tag{6.75}$$
$$\mathcal{I}(S[i+1:l]; Y^{(C)}[i+1:i+b_i-1], X^{(C)}[i+b_i:l+\tau] \big| S[0:i]) \tag{6.76}$$

where we have already shown Equation 6.75 remains k_i after the change.

Based on Equation 6.73 and Lemma 31 and under the relaxations,

$$\mathcal{I}(S[i+1:l]; Y^{(C)}[i+1:i+b_i-1], X^{(C)}[i+b_i:i+\tau-1] \big| S[0:i])$$

is not decreased (i.e., removing information about $S[i]$ from $Y^{(C)}[i+1:i+\tau-1]$ does not reduce the above expression). Both before and after the change, $X^{(C)}[i+\tau]$ provided $w_i^{(C)}$ symbols worth of information about $S[i]$. The change does not effect

$$\mathcal{I}(S[i+1:l]; X^{(C)}[i+\tau:l+\tau:i+b_i-1] \big| S[0:i])$$

So there is still enough information to ensure $Y^{(C)}[i+1:i+b_i-1]$ are recoverable. So loss recovery for $S[i+1:i+b_i-1]$ by their deadlines is still satisfied for this burst; For any burst starting in time slot $j > i$, loss recovery for $S[j:j+b_j-1]$ is similarly not affected.

Let $\delta_i = (p_{i+\tau}^{(C)} - w_i^{(C)})$ We now move δ symbols from $X^{(C)}[i+\tau]$ to $X^{(C)}[i+\tau+1]$, which increases $p_{i+\tau+1}^{(C)}$ by δ and decreases $p_{i+\tau}^{(C)}$ by δ.[3]

Loss recovery for $S[0:i-1]$ is the same because the changes occur after the deadline of $(i-1+\tau)$. Before the change by Equation 6.73 for any $j \in \mathbb{B}_i$ and $Y^{(C)}[j:j+b_j-1] \in \mathcal{Y}_{j:j+b_j-1}$, $\mathcal{H}(S[i] \big| Y^{(C)}[j:i+\tau-1], S[0:j-1]) \leq w_i^{(C)}$. After the change, the $p_{i+\tau}^{(C)} = w_i^{(C)}$ symbols of $X^{(C)}[i+\tau]$ are still available to be used to recover $S[i]$. So $S[i]$ is recovered by time slot $(i+\tau)$ (relaxation 5).

For $j = (i+1)$, the total amount of information available to recover $S[j]$ by time slot $(j+\tau)$ from received symbols has perhaps increased but not decreased from our alterations to C. It is possible that $w_j^{(C)}$ has increased as a result; however, the at most δ symbols of $X^{(C)}[i+\tau]$ that would have been used to recover $S[j]$ are now available in $X^{(C)}[j+\tau]$ to recover $S[j]$. So $S[j]$ is still recovered within τ time slots, and $p_{j+\tau}^{(C)} \geq w_j^{(C)}$.

For a burst starting in $j \in \mathbb{B}_i \cup \{i+1\}$, or any other burst that ends during or before $(i+\tau)$, the total number of symbols received over $X^{(C)}[i+\tau:i+\tau+1]$ is not reduced, so loss recovery of $S[i+1:j+b_j-1]$ is not negatively impacted. Either $X^{(C)}[i+\tau:i+\tau+1]$ are received (no change) or the information about lost packets is not lowered by the relaxations. For a burst

[3]We do not move the extra symbols we sent due to rounding issues (if we needed to move these to move δ symbols, then these symbols were never needed and are simply not sent).

starting during or after $(i+\tau+1)$, the symbols of $X[i+\tau]$ would not be used anyway because $S[0:i+\tau]$ are available. For bursts that include $X[i+\tau:i+\tau+1]$—therefore, starting no sooner than $X[i+2]$ and also ensuring relaxation 1 applies—up to 2 extra symbols may be lost due to a rounding issue of $\lceil \ell_l n_{C,l} \rceil$ symbols being lost in $X[l]$ for $l \in \{i+2, i+\tau+1\}$. This can be mitigated by sending two extra symbols in $X[r+\tau]$ for $r \in \{i+2, \ldots, i+\tau\}$ that is used to recover $S[r]$ (relaxation 5) and increasing $w_r^{(C)}$ by two (up to a max of k_r). in other words, all information about up to two symbols of $S[r]$ are removed from the transmission, C pretends k_r was two symbols smaller, and the extra two symbols are sent via replication.

Iterating over $i_* = b_0, \ldots, (t-\tau-1)$, we add at most $2(\tau-1)$ symbols for each value of i_*.

Finally, we note how the proof changes without the condition on feedback. For each $i \in [t-\tau]$ where $\ell_{i+\tau} < \ell_{i+\tau+1}$, the number of extra symbols sent associated with step i will increase by up to $(\ell_{i+\tau+1} - \ell_{i+\tau})$ times the number of parity symbols moved from $X^{(C)}[i+\tau]$ to $X^{(C)}[i+\tau+1]$. Consider a relaxed code, C, that was originally rate optimal. If more than $\sum_{j=i+1}^{i+\tau} k_j \ell_j$ parity symbols are moved, instead they can be deleted, and for $j \in \{i+1, \ldots, i+\tau\}$, $w_j^{(C)}$ could be set to $\ell_j k_j$ and an extra $w_j^{(C)}$ parity symbols be sent in $X^{(C)}[j+\tau]$. Thus, the number of symbols moved during time slot i is at most $\sum_{j=i+1}^{i+\tau} k_j \ell_j$. so the increase during iteration i is at most

$$(\ell_{i+\tau+1} - \ell_{i+\tau}) \sum_{j=i+1}^{i+\tau} k_j \ell_j$$

□

Corollary 8. *For any $i \in [t-\tau]$, any $j \in \mathbb{B}_i$, and any construction C adjusted under Corollary 7*

$$\min_{Y^{(C)}[j:j+b_j-1] \in \mathcal{Y}_{j:j+b_j-1}} \big(\mathcal{I}(S[j:i]; Y^{(C)}[j:j+\tau-1] \big| S[0:j-1])\big) = \sum_{l=j}^{i} k_l - w_l^{(C)}.$$

Proof. Follows from Lemma 32, Lemma 33, Corollary 7, and Equation 6.32. □

Finishing the proof with no resets

Let C be an offline rate-optimal construction under the relaxations of Appendix 6.8.4. Recall that without loss of generality, for all $i \in [t-\tau]$ then $w_i^{(C)} = p_{i+\tau}^{(C)}$ by Corollary 7. This causes C to send an extra at most $2(t-\tau)(\tau-1)$ symbols compared to a rate-optimal code.

We will show that C satisfies the constraints where $p_{i+\tau}^{(IP)} = w_i^{(C)}$ for all $i \in [t-\tau]$. So the objective function is at most the number of symbols sent by C; this in turn is at most

$$2(t-\tau)(\tau+1)$$

more than a rate-optimal code. Essentially, we show that Constraint 4 is analogous to the worst-case-delay constraint for a burst starting in time slot $i \in [t-\tau]$ where (a) for $j \in \{i, \ldots, i+b_i-1\}$, $w_j^{(C)}$ symbols of $S[j]$ are recovered during time slot $(j+\tau)$, (b) $d_{i,j}$ reflects the number of useful symbols of $Y^{(C)}[j]$ for recovering the remaining symbols of $S[i:j]$ not recovered in (a), and (c) for all $l \in \{i+b_i, \ldots, i+\tau-1\}$, at most $p_l^{(IP)}$ symbols of $X^{(C)}[l]$ are useful for recovering the remaining symbols of $S[i:i+b_i-1]$ not recovered in (a). Ultimately, because prove C satisfies the constraints, the value of the objective function of the algorithm is smaller than the number of symbols sent by C which is at most

$$2(t-\tau)(\tau+1)$$

more than the that of a rate-optimal offline code.

We begin by noting that Constraint 1 holds by the proof of Corollary 7. By definition, $w_i^{(C)} \le \ell_i k_i$, so Constraint 5 is satisfied. Also, Constraint 2 holds by Equation 6.33 and Equation 6.21.

By Corollary 8, for any burst starting in $i \in [t-\tau]$ and $j \in \{i, \ldots, i+b_i-1\}$

$$\begin{aligned}\min_{Y^{(C)}[i:i+b_i-1] \in \mathcal{Y}_{i:i+b_i-1}} &\big(\mathcal{I}\big(S[i:j];\\ &Y^{(C)}[i:i+\tau-1]\big|S[0:i-1]\big)\big) = \sum_{l=i}^{j} k_l - w_l^{(C)}.\end{aligned} \tag{6.77}$$

Consequently,

$$\begin{aligned}
&\mathcal{I}\big(S[i:i+b_i-1];Y^{(C)}[i:i+\tau-1]\big|S[0:i-1]\big)\big) = && (6.78)\\
&\mathcal{I}\big(S[i:i+b_i-1];Y^{(C)}[i:i+b_i-1]\big|S[0:i-1]\big)+ \\
&\mathcal{I}\big(S[i:i+b_i-1];Y^{(C)}[i+b_i:i+\tau-1]\big| && (6.79)\\
&S[0:i-1],Y^{(C)}[i:i+b_i-1]\big)\big) \le \\
&\sum_{l=i}^{i+b_i-1} \min\big((1-\ell_l)(p_l^{(IP)}+k_l), k_l - w_l^{(C)} + \sum_{r=i}^{l-1} k_r - w_r^{(C)} - \\
&\mathcal{I}(S[i:r];Y^{(C)}[r]\big|S[0:i-1],Y^{(C)}[i:r-1])\big)+ && (6.80)\\
&\sum_{l=i+b_i}^{i+\tau} \min\big(p_i^{(C)}, \sum_{r=i}^{j} k_r - w_r^{(C)} - \sum_{r=i}^{l-1} \\
&\mathcal{I}(S[i:j];Y^{(C)}[r]\big|S[0:i-1],Y^{(C)}[i:r-1])\big) \le \\
&\sum_{l=i}^{i+b_i-1} \min\big((1-\ell_l)(p_l^{(IP)}+k_l), k_l - w_l^{(C)} + \sum_{r=i}^{l-1} k_r - w_r^{(C)} - \\
&\mathcal{I}(S[i:r];Y^{(C)}[r]\big|S[0:i-1],Y^{(C)}[i:r-1])\big)+ && (6.81)\\
&+\sum_{l=i+b_i}^{i+\tau} p_i^{(C)}
\end{aligned}$$

where Equation 6.79 follows from the chain rule for MI; Equations 6.80 follows from relaxations 1 and 2 as well as Lemma 31 and Corollary 7; Equation 6.81 follows from the definition of the minimum function.

Therefore, by Equation 6.77, for any $i \in [t-\tau], j \in \{i, \ldots, i+b_i-1\}$ Constraints 3 and 4 are satisfied.

By the minimization of Algorithm 4, the values for $p_i^{(IP)}$ lead to

$$\sum_{i=0}^{t-\tau} w_{i+\tau}^{(C)} \geq \sum_{i=0}^{t} p_{i+\tau}^{(IP)}$$

where the number of symbols sent by a rate-optimal code is by Corollary 7 is at most

$$-2(t-\tau)(\tau-1) + \sum_{i=0}^{t-\tau} w_{i+\tau}^{(C)} \geq -2(t-\tau)(\tau-1) + \sum_{i=0}^{t} p_{i+\tau}^{(IP)}.$$

Extending to resets

At a high level, we prove the result by induction on the number of resets. The base case of no resets has already been shown. If the first reset occurs in time slot r, we show that the objective function is only $2(r)(\tau-1)$ more than the number of symbols sent under a rate optimal code by time slot r. We then apply the inductive hypothesis to prove closeness to optimality for the rest of the transmission.

We prove by induction on $\zeta = \sum_{l=0}^{t} \zeta_l$. The base case of $\zeta = 0$ has already been proven.

For the inductive step, we prove the result for ζ assuming that it is been proven for all $j \in \{0, \ldots, \zeta-1\}$. Let r be the smallest value so that $\zeta_r = 1$. If $r < \tau$ then only the lossless-delay constraint is imposed for $S[0:r-1]$ so $\sum_{l=0}^{r-1} k_l$ symbols are sent and no parity need to be sent. Otherwise, $S[r-\tau : r-1]$ need not be recovered under lossy conditions. By considering $k_{r-\tau}, \ldots, k_{r-1}$ to all be 0, the output of the algorithm through time slot $(r-1)$ can be viewed as how many parity symbols to send per time slot for a code that sends at most $2r(\tau-1)$ more symbols than a rate-optimal code for the transmission of $S[0:r-1]$ after appending $(\tau+1)$ time slots with size 0 frames. Then an extra $\sum_{l=r-\tau}^{r-1} k_l$ symbols must also be sent under C for the lossless-delay constraint (as with the offline code).

Either way, for time slot r and above, we can view Algorithm 4 as being applied a second time on the remainder of the transmission given $p_l^{(IP)} = 0$ for $l \in \{r, \ldots, r+\tau-1\}$ and the worst-case-delay is not imposed for $S[r-\min(r,\tau) : r-1]$. Then $\sum_{l=r}^{t} \zeta_l = (\zeta-1)$, so the correctness of Algorithm 4 holds by the inductive hypothesis. We note at most $(2r(\tau-1)+2(t-r-\tau)(\tau-1)) = 2(t-\tau)(\tau-1)$ extra symbols are sent overall compared to a rate-optimal offline code.

6.8.5 Proof of Theorem 9

At a high level, the constraints of the algorithm ensure that Equation 6.3 is satisfied. For $i \in [t-\tau]$, the sizes of u_i and $p_{i+\tau}$ are then slightly increased to ensure they are divisible by q_i and $h_{i+\tau}$ respectively, leading to the slightly lower rate than the upper bound from Lemma 22.

Equation 6.3 need only be satisfied with the correct values of u_i and k_i in it and Equation 6.2 (due to the structure of the proof of Theorem 8). By the minimization and Constraints 3, 4 and Equations 6.2 (and $u_i = k_i$ if $\sum_{j=i+1}^{i+\tau} \zeta_i > 0$), Equation 6.3 is always satisfied. Thus, the construction's requirements are met.

For $j \in [t-\tau]$, if $\sum_{l=j+1}^{j+\tau} \zeta_l > 0$ then $p_{j+\tau} = 0$ (by the "resets" case of the construction); also, $u_i = k_i$, so failing to decode $S[i]$ will not hinder using parity symbols for recovering earlier frames. Recall for any $u_j^{(LP)}$ that increasing $u_j^{(LP)} = p_{j+\tau}^{(IP)}$ to a quantity no more than $\ell_j k_j$ retains satisfaction of all constraints of the LP. So increasing w_i from $p_{i+\tau}^{(IP)}$ for $i \in [t-\tau]$ (to ensure $q_i \big| w_i$) and likewise increasing $p_{i+\tau}$ (i.e., $p_{i+\tau}^{(IP)}$) is still a valid solution to the LP. Increasing $p_{i+\tau}$ (similarly $p_{i+\tau}^{(IP)}$) to be divisible by $h_{i+\tau}$ for $i \in [t-\tau]$ likewise retains the satisfaction of all constraints (with Constraint 5 removed). After these changes to the values of $p_j^{(IP)}$, $\big(\tau, t, K, \mathrm{Z}, \mathcal{L}, B, W\big)$-Split Code is fully specified and sends $\sum_{l=0}^{t-\tau} k_l + p_{l+\tau}^{(IP)}$. The total increase in $\sum_{l=0}^{t-\tau} p_{l+\tau}^{(IP)}$ due to the change is at most $\sum_{l=0}^{t-\tau} \mathbb{1}[p_{l+\tau}^{(IP)} \neq 0](q_l + h_{l+\tau} - 2)$.

We know the rate is

$$\big(\sum_{i=0}^{t} k_i\big) / \big(2(t-\tau)(\tau-1) + \sum_{i=0}^{t-\tau} k_i + w_i + q_l + h_{l+\tau} - 2\big).$$

It also sends at most

$$\epsilon = 2(t-\tau)(\tau-1) + \sum_{i=0}^{t-\tau} k_i + w_i + q_l + h_{l+\tau} - 2$$

more symbols than a rate-optimal code.

Let us label the number of symbols sent as N, so the rate is $(\sum_{i=0}^{t} k_i)/N$. Then the optimal rate is at most $(\sum_{i=0}^{t} k_i)/(N-\epsilon)$ where $(N-\epsilon) \geq \sum_{i=0}^{t} k_i$.

The difference to the optimal rate is at most

$$\begin{aligned}
&(\sum_{i=0}^{t} k_i)/(N-\epsilon) - (\sum_{i=0}^{t} k_i)/N \leq \\
&(\epsilon \sum_{i=0}^{t} k_i)/(N(N-\epsilon)) \leq \\
&\epsilon/(\sum_{i=0}^{t} k_i).
\end{aligned}$$

6.8.6 Proof of Lemma 24

Suppose Algorithm 4$(\tau, t, K, \mathcal{L})$ outputs $\Big\langle p_i^{(IP)} \big| i \in [t-\tau] \Big\rangle$. Recall from Lemma 22 that the offline optimal rate is at most

$$\sum_{i=0}^{t-\tau} k_i / \left(-2(t-\tau)(\tau-1) + \sum_{i=0}^{t-\tau} k_i + p_i^{(IP)} \right)$$

We will adjust the values of the $p_i^{(IP)}$ to be $p_i^{(IP,1)} = p_i$ while maintaining satisfaction of all constraints and show

$$\sum_{i=0}^{t-\tau} p_i - p_i^{(IP)} \leq (t-\tau+1).$$

We start by setting $p_i^{(IP,1)} = p_i^{(IP)}$ for $i \in [t]$.

We prove the result by induction on i. We start with the base case. For $i \in [\tau - 1]$, $p_i = p_i^{(IP,1)} = p_i^{(IP)} = 0$ by Constraint 1. Due to Constraint 4, $p_\tau^{(IP)} = p_\tau^{(IP,1)} = p_\tau = \ell_0 k_0$.

For the inductive hypothesis, we assume that for some $i_* \geq \tau$ for all $i \in [i_*]$ that $p_i^{(IP,1)} = p_i$, all constraints are satisfied (when applying $p^{(IP,1)}$ instead of $p^{(IP)}$), and

$$\sum_{i=0}^{i_*} p_i - p_i^{(IP)} \leq (i_* + 1).$$

For the inductive step, we consider $l = (i_* + 1)$.

Case $p_l \leq p_l^{(IP,1)}$. Then let $\delta = p_l^{(IP,1)} - p_l$. We set $p_l^{(IP,1)} = p_l$ and increase $p_{l+1}^{(IP,1)}$ by δ. By Lemma 23, the constraints are clearly still satisfied for a burst starting in time slot $(l - \tau)$. For a burst starting in time slot $(l - \tau + 1), \ldots, l - 1$, the same number of parity symbols are received, so all constraints remain satisfied. For a loss in time slot l, the constraints are satisfied because more parity symbols are available. For a burst starting after time slot l, the change cannot hurt the satisfaction of the constraints, as parity symbols sent in time slot l would not be useful. In this case, $\sum_{j=0}^{l+1} p_l^{(IP,1)}$ did not change.

Case $p_l > p_l^{(IP,1)}$. By Equation 6.11, $p_r^{(IP,1)} = p_r$ for $r \in \{l - \tau, \ldots, l - 1\}$, and satisfaction of Constraint 4 (for $p_r^{(IP,1)}$), it must be that $\left|p_l - p_l^{(IP,1)}\right| < 1$. The discrepancy comes from taking a ceiling in Equation 6.11. Therefore, it suffices to increase $p_l^{(IP,1)}$ by $(p_l - p_l^{(IP,1)}) \leq 1$. After the change, all constraints are satisfied, $p_l^{(IP,1)} = p_l$, and

$$\sum_{i=0}^{l} p_i - p_i^{(IP)} \leq (l + 1).$$

6.8.7 Proof of Lemma 26

Let us define $U[i]$, and $V[i]$, for C according to Section 6.2. We define the number of parity symbols as $p_i^{(C)}$ and the parity symbols as $P^{(C)}[i]$ where $w_i^{(C)} = p_{i+\tau}^{(C)}$ under C.

Recall that $b_0 = \tau$. By definition of the Regime b_τ-Code and C, for $i \in [\tau - 1]$, $p_i = p_i^{(C)} = 0$. For $i \in \{\tau, \ldots, 2\tau - 1\}$, $p_i = p_i^{(C)} = \ell_{i-\tau} k_{i-\tau}$.

Let us assume that $p_i = p_i^{(C)}$ for all $i \in [i_* - 1]$ for some $i_* \geq 2\tau$. Now we consider i_*.

Under C, by Equation 6.2 (burst ending in i_*) and Equation 6.3, $\ell_{i_*}(k_{i_*} - w_{i_*}^{(C)}) \leq \ell_{i_*} p_{i_*}^{(C)}$. Therefore, for any burst starting in time slot $j \in \mathbb{B}_{i_*}$, for any $l \in \{j, \ldots, j + \tau - 1\}$ we know $P^{(C)}[j]$ is used to recover all missing symbols of $V[j]$.

Suppose $p_{i_*} < p_{i_*}^{(C)}$. Let $\delta = (p_{i_*}^{(C)} - p_{i_*})$. We reset $w_{i_*-\tau}^{(C)}$ to equal $p_{i_*}/\ell_{i_*-\tau}$. We reduce $p_{i_*}^{(C)}$ by δ to equal p_{i_*}. By Equation 6.14, $(\ell_{i_*-\tau} k_{i_*-\tau} - p_{i_*}) \leq \ell_{i_*-\tau} p_{i_*-\tau}^{(C)}$. Therefore, enough parity symbols of $X^{(C)}[i_* - \tau]$ are received to recover $V[i_* - \tau]$. We must increase $w_{i_*-\tau}^{(C)}$ to be divisible by $q_{i_*-\tau}$ (so that $U[i_* - \tau]$ has $w_{i_*-\tau}^{(C)}/\ell_{i_*-\tau}$ symbols). This causes $p_{i_*}^{(C)}$ to be increased by up to $(q_{i_*-\tau} - 1)$ symbols. We then pad $p_{i_*}^{(C)}$ with up to $(h_{i_*} - 1)$ symbols to be divisible by h_{i_*}. The total number of extra symbols sent is at most $(q_{i_*-\tau} + h_{i_*} - 2)$.

After the changes, $V[i_* - \tau]$ is still recoverable with the received symbols of $P^{(C)}[i_* - \tau]$. Therefore, loss recovery for $S[0 : i_* - \tau - 1]$ is unchanged. Adjusting the size of $U[i_* - \tau]$ does not effect loss recovery for $V[i_* - \tau + 1 : i_* - 1]$ because of $V[i_* - \tau]$ still being recovered, $U[i_* - \tau]$ being independent of the symbols of $P^{(C)}[i_* - \tau + 1 : i_* - 1]$, and $V[j]$ being recovered in $P^{(C)}[j]$ for $j \in \{i_* - \tau + 1, \ldots, i_* - 1\}$. Then $U[i_* - \tau]$ is recoverable using $P^{(C)}[i_*]$. In addition, $P^{(C)}[i_*]$ is independent of the symbols of $U[i_* - \tau + 1 : i_*]$, so loss recovery for these quantities is unaffected. The change ensures that the size of $V[i_*]$ is at most $p_{i_*}^{(C)}$, so $V[i_*]$ is recoverable using the received symbols of $Y^{(C)}[i_*]$ for any burst where $X^{(C)}[i_*]$ experiences loss. For any burst where $X^{(C)}[i_*]$ experiences loss for $j \in \{i_* + 1, \ldots, i_* + \tau - 1\}$, $V[j]$ can still be recovered with the received symbols of $Y^{(C)}[j]$. Then $U[i_*]$ is recovered with $P[i_* + \tau]$, and $U[j]$ is recovered with $P[j + \tau]$ for $j \in \{i_* + 1, \ldots, i_* + \tau - 1\}$. Therefore, after the change, the lossless-delay and worst-case-delay constraints are still met. At most $(q_{i_*} + h_{i_* + \tau} - 1)$ extra symbols were sent.

Suppose $p_{i_*} = p_{i_*}^{(C)}$. No change is needed, and we move on to $(i_* + 1)$.

Suppose $p_{i_*} > p_{i_*}^{(C)}$. This case cannot occur. Recall that the changes from earlier time slots only increased $p_{i_*}^{(C)}$. Recall $n_{i_* - \tau} = n_{C, i_* - \tau}$. All but $w_{i_* - \tau}^{(C)} = u_{i_* - \tau} \ell_{i_* - \tau}$ symbols of $S[i_* - \tau]$ are recovered during time slot $(i_* - \tau)$. In order to have $p_{i_*}^{(C)} < p_{i_*}$, this means that $w_{i_* - \tau}^{(C)} < p'_{i_*}$ by Equation 6.15. But by Equation 6.14, only $n_{i_* - \tau}(1 - \ell_{i_* - \tau}) < k_{i_* - \tau} - w_{i_*}^{(C)}$ symbols were received during time slot $(i_* - \tau)$, which is a contradiction

Applying the change over all time slots leads to an extra $\sum_{i=0}^{t-\tau}(q_i + h_{i+\tau} - 2)$ extra symbols being sent. The resulting scheme, C, sends the same number of symbols as the Regime b_τ-Code after the change. This concludes the result.

6.8.8 Proof of Theorem 10

Let b_i be fixed as b for all $i \in [t]$. Let $\ell_i \in \ell$ be fixed as $\ell = q/h \in (0, 1)$ for $i \in [t - \tau]$. Recall that $\tau > b \geq 2$. Let $\zeta_i = 0$ for all $i \in [t]$. Let d be an arbitrary positive integer where $h^3(2h - q) \big| d$. We consider two frame-size sequences:

1. $k_0^{(1)} = k_1^{(1)} = d, k_\tau^{(1)} = 2d(1 - \ell), k_{\tau+1}^{(1)} = d(1 - \ell), k_{2\tau+1}^{(1)} = d(1 - \ell)$, and for all $j \in \{2, \ldots, \tau - 1, \tau + 2, \ldots, 2\tau, 2\tau + 2, \ldots, t\}$ $k_j^{(1)} = 0$.
2. $k_0^{(2)} = k_1^{(2)} = d, k_\tau^{(2)} = 2d(1 - \ell), k_{\tau+1}^{(2)} = d(1 - \ell), k_{\tau+b} = d(1 - \ell)$, and for all $j \in \{2, \ldots, \tau - 1, \tau + 2, \ldots, \tau + b - 1, \tau + b + 1, \ldots, t\}$ $k_j^{(2)} = 0$.

For large d, the proof extends to frame-size sequences where the frames' sizes approximately equal the ones above. It also applies to longer frame-size sequences that contain the above frame-size sequences (or approximations of them).

Let code construction, C, be any offline rate-optimal construction that satisfies the lossless-delay and worst-case-delay constraints and $w_i^{(C)} = p_{i+\tau}^{(C)}$ for all i.[4] Suppose C sends $X^{(C)}[i]$ during time slot $i \in [t]$, where $X^{(C)}[i]$ comprises $n_{C,i}$ symbols and $(c_{C,i} + 1)$ transmitted packets.

[4]The construction may be slightly below rate optimal to put in the form $w_i^{(C)} = p_{i+\tau}^{(C)}$. By Theorem 9, it sends at most $2(t - \tau)(\tau - 1) + \sum_{l=0}^{t-\tau}(q_l + h_{l+\tau} - 2))$ more symbols than a rate optimal code. This will have a negligible effect on the rate for sufficiently large d.

Let us define $w_i^{(C)}$ and $p_i^{(C)}$ as in Equations 6.32 and 6.33.

We begin by proving several results towards proving the converses. We will include a few relaxations that can only increase the rate when proving the converse.

For either frame-size sequence

$$w_0^{(C)} = w_1^{(C)} = \ell d = p_\tau^{(C)} = p_{\tau+1}^{(C)} \tag{6.82}$$

$$\forall i \in [\tau] \cup \{2+\tau, \ldots, 2\tau-1\}, p_i^{(C)} = 0 \tag{6.83}$$

In addition, $w_\tau^{(C)} \leq d(1-\ell)\ell$ without loss of generality because we use relaxation

$$\mathcal{I}(S[\tau]; Y^{(C)}[\tau] \big| S[0:1]) = n_{C,\tau}(1-\ell)$$

(i.e., all parity symbols of $X[\tau]$ that are received are useful). Also, $S[0:1]$ are received if $X[\tau]$ is involved in the burst. So we take the relaxation that

$$\mathcal{I}(S[\tau:\tau+1]; Y^{(C)}[\tau:\tau+1] \big| S[0:1]) =$$
$$(n_{C,\tau} + n_{C,\tau+1})(1-\ell).$$

As such,

$$w_\tau^{(C)} + w_{\tau+1}^{(C)} = 3d(1-\ell)\ell - d(1-\ell)\ell = d(1-\ell)\ell \tag{6.84}$$

(i.e., of the $3d(1-\ell)\ell$ lost frame symbols of $S[\tau:\tau+1]$, $2d(1-\ell)\ell$ are recovered using the received parity symbols of $X^{(C)}[\tau:\tau+1]$). By Corollary 7 for any $i \in \{\tau+2, \ldots, 2\tau-1\}$ either $n_{C,i} = 0$ or if it is the second frame-size sequence $n_{C,\tau+b} = k_{\tau+b} > 0$. For the second frame-size sequence

$$\begin{aligned}
&n_{C,\tau+b} \geq \\
&\mathcal{I}(S[\tau:\tau+1], S[\tau+b]; X^{(C)}[\tau+b] \big| S[0:\tau-1]) = \\
&\mathcal{I}(S[\tau:\tau+1]; X^{(C)}[\tau+b] \big| S[0:\tau-1]) + \\
&\mathcal{I}(S[\tau+b]; X^{(C)}[\tau+b] \big| S[0:\tau+1]) = \\
&\mathcal{I}(S[\tau:\tau+1]; X^{(C)}[\tau+b] \big| S[0:\tau-1]) + \\
&k_{\tau+b}
\end{aligned}$$

Therefore,

$$\mathcal{I}(S[\tau:\tau+1]; X^{(C)}[\tau+b] \big| S[0:\tau-1]) = 0.$$

Also,

$$\begin{aligned}
&n_{C,2\tau} = w_\tau^{(C)} \geq \\
&\mathcal{I}(S[\tau:\tau+1]; X^{(C)}[2\tau] \big| S[0:\tau-1], Y^{(C)}[\tau:2\tau-1]) = \\
&\mathcal{I}(S[\tau]; X^{(C)}[2\tau] \big| S[0:\tau-1], Y^{(C)}[\tau:2\tau-1]) + \\
&\mathcal{I}(S[\tau+1]; X^{(C)}[2\tau] \big| S[0:\tau], Y^{(C)}[\tau:2\tau-1]) = \\
&w_\tau^{(C)} + \mathcal{I}(S[\tau+1]; X^{(C)}[2\tau] \big| S[0:\tau], Y^{(C)}[\tau:2\tau-1])
\end{aligned}$$

Therefore, $\mathcal{I}(S[\tau+1]; X^{(C)}[2\tau] \big| S[0:\tau], Y^{(C)}[\tau : 2\tau-1]) = 0$. For $j \in \{0,1\}$ no information about $S[\tau+j]$ is available after $X^{(C)}[\tau+1]$ until $X^{(C)}[2\tau+j]$. Therefore, without loss of generality

$$(w_\tau^{(C)} + w_{\tau+1}^{(C)}) = d(1-\ell)\ell. \tag{6.85}$$

By the Equations 6.82 and 6.83 $n_{C,0} = n_{C,1} = d$ and $p_\tau^{(C)} = p_{\tau+1}^{(C)} = d\ell$. By Equation 6.21 and Lemma 33, for a burst starting in $X[0]$, $\mathcal{H}(S[0:1], S[\tau:\tau+1] \big| Y^{(C)}[0:1], X[\tau:\tau+1]) = 0$ so for $i \in \{\tau, \tau+1\}$, at least $(k_i + d\ell)$ symbols are sent in $X[i]$. Thus,

$$d(2 + 3(1-\ell) + 2\ell) \tag{6.86}$$

symbols are sent by time slot $(\tau+1)$. And the total number of frame symbols is

$$K' = 2d(1 + 2(1-\ell)). \tag{6.87}$$

Offline scheme for frame-size sequence 1. Select

$$\begin{aligned} w_0 &= d\ell \\ w_1 &= d\ell \\ w_\tau &= 0 \\ w_{\tau+1} &= d(1-\ell)\ell \\ w_{2\tau+1} &= d(1-\ell)\ell - d(1-\ell)^2\ell = d(1-\ell)\ell^2. \end{aligned}$$

The total number of parity symbols sent is

$$2d\ell + d(1-\ell)\ell + d(1-\ell)\ell^2.$$

We apply $\big(\tau, t, K, Z, \mathcal{L}, B, W\big)$-Split Code. The lossless-delay constraint is clearly satisfied. Next, we show the worst-case-delay constraint is met. For any burst starting in $i \in \{0,1\}$, $(i+b-1) \le b < \tau$, so $X[\tau]$ and $X[\tau+1]$ are received. Thus, $S[0:1]$ are recovered. Since $X[2:\tau-1]$ are empty, we need not consider another burst until one that starts in $X[\tau]$. For a burst in $X[\tau]$, $2d(1-\ell)\ell$ symbols of $S[\tau]$ are lost, $d\ell(1-\ell)$ parity symbols are received in each of $X[\tau]$ and $X[\tau+1]$, and $u[\tau+1] = k_{\tau+1}$, so $X[\tau]$ is recovered by time slot $(\tau+1)$. For a burst starting in $X[\tau]$ or $X[\tau+1]$, the $d(1-\ell)\ell$ parity symbols of $X[2\tau+1]$ recover $S[\tau+1]$. So $S[\tau:\tau+1]$ are recovered in time. Finally, for any $i \in \mathbb{B}_{2\tau+1}$, $d(1-\ell)^2\ell$ missing symbols of $S[2\tau+1]$ are recovered using the $d(1-\ell)^2\ell$ received parity symbols of $X[2\tau+1]$. The remainder are recovered using $X[3\tau+1]$. In total,

Converse for frame-size sequence 1. Recall that $(p_\tau^{(C)} + p_{\tau+1}^{(C)}) = 2d\ell$ and $(p_{2\tau}^{(C)} + p_{2\tau+1}^{(C)}) \ge d(1-\ell)\ell$. By Corollary 7, $n_{C,2\tau} = w_\tau^{(C)}$ and $n_{C,2\tau+1} = w_{\tau+1}^{(C)} + d(1-\ell)$. At least $(d(1-\ell)\ell - w_{\tau+1}^{(C)}(1-\ell)))$ parity symbols are sent in $X[3\tau+1]$. Therefore, the total number of symbols sent is at least

$$\begin{aligned} &K' + 2d\ell + d(1-\ell) + d(1-\ell)\ell - w_{\tau+1}^{(C)}(1-\ell) = \\ &K' + 2d\ell + d(1-\ell) + d(1-\ell)\ell^2 + w_\tau^{(C)}(1-\ell). \end{aligned}$$

Converse for frame-size sequence 2. By Corollary 7, $n_{C,2\tau} = w_{\tau}^{(C)}$ and $n_{C,2\tau+1} = w_{\tau+1}^{(C)}$. By definition of $w_{\tau+1}^{(C)}$ there is a burst starting in time slot $(\tau + 1)$ so that

$$\begin{aligned}&\mathcal{H}(S[\tau+1]\big|S[0:\tau], Y^{(C)}[\tau+1:\tau+b],\\ &X^{(C)}[\tau+b+1:2\tau]) = w_{\tau+1}^{(C)} =\\ &\mathcal{H}(S[\tau+1]\big|S[0:\tau], Y^{(C)}[\tau+1], Y^{(C)}[\tau+b], X^{(C)}[2\tau])\end{aligned} \tag{6.88}$$

since for $j \in \{i+\tau+2, \ldots, i+\tau+b-1, i+\tau+b+1, \ldots, i+2\tau-1\}$, $n_{C,j} = 0$. Combining Equation 6.88,Lemma 33, and the definition of Mutual Information shows

$$\begin{aligned}&w_{\tau+1}^{(C)} = n_{C,2\tau+1} \geq\\ &\mathcal{I}(S[\tau+1]; X^{(C)}[2\tau+1]\big|\\ &S[0:\tau], Y^{(C)}[\tau+1:\tau+b], X^{(C)}[\tau+b+1:2\tau]) =\\ &\mathcal{I}(S[\tau+1]; X^{(C)}[2\tau+1]\big|\\ &S[0:\tau], Y^{(C)}[\tau+1], Y^{(C)}[\tau+b], X^{(C)}[2\tau]).\end{aligned} \tag{6.89}$$

Also, by the fact that $n_{C,j} = 0 \; j \in \{i+\tau+2, \ldots, i+\tau+b-1, i+\tau+b+1, \ldots, i+2\tau-1\}$,

$$\begin{aligned}&w_{\tau+1}^{(C)} = n_{C,2\tau+1} \geq\\ &\mathcal{I}(S[\tau+1], S[\tau+b]; X^{(C)}[2\tau+1]\big|\\ &S[0:\tau], Y^{(C)}[\tau+1:\tau+b], X^{(C)}[\tau+b+1:2\tau]) =\\ &\mathcal{I}(S[\tau+1], S[\tau+b]; X^{(C)}[2\tau+1]\big|\\ &S[0:\tau], Y^{(C)}[\tau+1], Y^{(C)}[\tau+b], X^{(C)}[2\tau]) =\\ &\mathcal{I}(S[\tau+1]; X^{(C)}[2\tau+1]\big|\\ &S[0:\tau], Y^{(C)}[\tau+1], Y^{(C)}[\tau+b], X^{(C)}[2\tau]) +\\ &\mathcal{I}(S[\tau+b]; X^{(C)}[2\tau+1]\big|\\ &S[0:\tau+1], Y^{(C)}[\tau+b], X^{(C)}[2\tau])\end{aligned} \tag{6.90}$$

using the fact that $Y^{(C)}[\tau+1]$ is a function of $S[0:\tau+1]$. Combining Equations 6.89 and 6.90 shows $0 =$

$$\mathcal{I}(S[\tau+b]; X^{(C)}[2\tau+1]\big|S[0:\tau+1], Y^{(C)}[\tau+b], X^{(C)}[2\tau]). \tag{6.91}$$

Finally, we combine Equation 6.91 with the chain rule for mutual information to show

$$\begin{aligned}&\mathcal{I}(S[\tau+b]; X^{(C)}[2\tau:2\tau+1]\big|S[0:\tau+1], Y^{(C)}[\tau+b]) =\\ &\mathcal{I}(S[\tau+b]; X^{(C)}[2\tau]\big|S[0:\tau+1], Y^{(C)}[\tau+b]) \leq\\ &n_{C,2\tau} = w_{\tau}^{(C)}.\end{aligned} \tag{6.92}$$

We have already established that

$$M = (k_0 + k_1)(1+\ell) + k_\tau + k_{\tau+1} + w_{\tau}^{(C)} + w_{\tau+1}^{(C)} + k_{\tau+b} \tag{6.93}$$

symbols are sent by time slot $(2\tau+1)$ to ensure the lossless-delay constraint is met and $S[0:\tau+1]$ are recovered within the worst-case-delay. In addition, in case of a burst starting in time slot $(\tau+b)$, we bound the amount of information about $S[\tau+b]$ that is available; the remainder must also be sent, and can be sent in $X[2\tau+b]$.

If $\tau+b+b-1 < 2\tau$ (i.e., $2b-1 < \tau$) then $X^{(C)}[2\tau]$ is received if $X[\tau+b]$ is lost; so at least $E = (k_{\tau+b}\ell - w_{\tau}^{(C)})$ extra symbols must be sent. So an extra $d(1-\ell)\ell - w_{\tau}^{(C)}$ symbols are sent compared to the optimal value; the optimal value is 0 and is obtained when $w_{\tau}^{(C)} = d(1-\ell)\ell$.

Suppose $\tau+b+b-1 > 2\tau$ (i.e., $2b-1 > \tau$) then we note

$$\begin{aligned}
&\mathcal{I}(S[\tau+b]; Y^{(C)}[2\tau:2\tau+1] \big| S[0:\tau+1], Y^{(C)}[\tau+b]) \leq \\
&\mathcal{H}(Y^{(C)}[2\tau:2\tau+1] \big| S[0:\tau+1], Y^{(C)}[\tau+b]) \leq \\
&\mathcal{H}(Y^{(C)}[2\tau:2\tau+1]) \leq \\
&(n_{C,2\tau} + n_{C,2\tau+1})\ell = (w_{\tau}^{(C)} + w_{\tau+1}^{(C)})\ell = d(1-\ell)\ell^2
\end{aligned} \tag{6.94}$$

by Equation 6.84. $X^{(C)}[2\tau:2\tau+1]$ is in loss if $X[\tau+b]$ is lost. Therefore, by Equation 6.94 and Equation 6.92, $E = (k_{\tau+b}\ell - \min(w_{\tau}^{(C)}, d(1-\ell)\ell^2))$ symbols must be sent.

If $\tau+b+b-1 = 2\tau$ (i.e., $2b-1 = \tau$) then $X^{(C)}[2\tau]$ is lost if $X[\tau+b]$ is lost but $X[2\tau+1]$ is not. We note

$$\begin{aligned}
&\mathcal{I}(S[\tau+b]; Y^{(C)}[2\tau], X^{(C)}[2\tau+1] \\
&\qquad\qquad \big| S[0:\tau+1], Y^{(C)}[\tau+b]) \leq \\
&\mathcal{H}(Y^{(C)}[2\tau], X^{(C)}[2\tau+1] \big| S[0:\tau+1], Y^{(C)}[\tau+b]) \leq \\
&\mathcal{H}(Y^{(C)}[2\tau], X^{(C)}[2\tau+1]) \leq \\
n_{C,2\tau}\ell + n_{C,2\tau+1} = {} &w_{\tau}^{(C)}\ell + w_{\tau+1}^{(C)}.
\end{aligned} \tag{6.95}$$

Combining Equations 6.95 and 6.92 shows at least $E = (k_{\tau+b}\ell - \min(w_{\tau}^{(C)}, w_{\tau}^{(C)}\ell + w_{\tau+1}^{(C)})))$ symbols must be sent.

To assess this term, we observe that

$$\begin{aligned}
&\min\left(w_{\tau}^{(C)}, w_{\tau}^{(C)}\ell + w_{\tau+1}^{(C)}\right) = \\
&\min\left(w_{\tau}^{(C)}, d(1-\ell)\ell - w_{\tau}^{(C)}(1-\ell)\right)
\end{aligned}$$

by Equation 6.84. If the first term is larger, the min is increased by decreasing $w_{\tau}^{(C)}$. If the first term is smaller, the min is increased by increasing $w_{\tau}^{(C)}$. The slope of the change is linear except where the two quantities equal. So the maximum occurs at the endpoints or where the quantities meet; recall that the maximum value minimizes E. if $w_{\tau}^{(C)} = d(1-\ell)\ell$, the quantity is $d(1-\ell)\ell^2$. If $w_{\tau}^{(C)} = 0$ then the quantity is 0 To meet in the middle

$$\begin{aligned}
w_{\tau}^{(C)} &= d(1-\ell)\ell - w_{\tau}^{(C)}(1-\ell) \\
w_{\tau}^{(C)} &= d(1-\ell)\ell/(2-\ell)
\end{aligned}$$

We note that $\ell/(1/(2-\ell)) = \ell(2-\ell)$; this quantity is less than $1/2$ for $\ell = 1/3$ and it is greater than $1/2$ for $\ell = 2/3$. Therefore, the value of $w_{\tau}^{(C)}$ that maximizes the desired quantity depends on ℓ. Let $w^{(C)}_{\tau,max} \in \{d(1-\ell)\ell, d(1-\ell)\ell/(2-\ell)\}$ be chosen to minimize E. So total sent is $(k_{\tau+b}\ell - e'$ for $e' \in \{d(1-\ell)\ell/(2-\ell), d(1-\ell)\ell^2\}$

Offline scheme for frame-size sequence 2. We set

$$\begin{aligned}
w^{(C)}_0 &= d\ell \\
w^{(C)}_1 &= d\ell \\
w^{(C)}_{\tau+1} &= \begin{cases} d(1-\ell)\ell - w^{(C)}_{\tau,max} \text{ if } 2b-1=\tau \\ 0 \text{ otherwise} \end{cases} \\
w^{(C)}_{\tau} &= d\ell(1-\ell) - w_{\tau+1} \\
w^{(C)}_{\tau+b} &= \begin{cases} k_{\tau+b}\ell - d\ell(1-\ell) \text{ if } 2b-1<\tau \\ k_{\tau+b}\ell - e'\ell \text{ if } 2b-1=\tau \\ k_{\tau+b}\ell - d\ell(1-\ell)^2 \text{ if } 2b-1>\tau. \end{cases}
\end{aligned}$$

We apply $\big(\tau, t, K, \mathrm{Z}, \mathcal{L}, B, W\big)$-Split Code. The lossless-delay constraint is clearly satisfied. Next, we show satisfaction of the worst-case-delay. For any burst starting in $i \in \{0,1\}$, $(i+b-1) \le b < \tau$, so $X[\tau]$ and $X[\tau+1]$ are received. Thus, $S[0:1]$ are recovered. Since $X[2:\tau-1]$ are empty, we need not consider another burst until one that starts in $X[\tau]$. For a burst in $X[\tau]$, $2d(1-\ell)\ell$ symbols of $S[\tau]$ are lost, $d\ell(1-\ell)$ parity symbols are received in each of $X[\tau]$ and $X[\tau+1]$, and $v[\tau]+v[\tau+1] = 2d(1-\ell)$, so $V[\tau]$ and $V[\tau+1]$ are recovered by time slot $(\tau+1)$. Then $U[\tau+j]$ is recovered with $P[2\tau+j]$ for $j \in \{0,1\}$. For a burst starting in $X[\tau+1]$, the received parity symbols of $X[\tau+1]$ recover $V[\tau+1]$. The symbols of $X[2\tau+1]$ provide no information on $S[\tau+b]$ given $X[2\tau]$ so they are used to recover $X[\tau+1]$. Then it immediately. So it suffices to consider a burst dropping $X[\tau+b]$. Sufficiently many symbols are available to recover all but $w^{(C)}_{\tau+b}$ by time slot $(2\tau+1)$. The remainder are recovered with $X[2\tau+b]$. The total number of symbols sent is the term from Equation 6.93 plus E where E is minimized.

Finally, we show in a case analysis over the three possible cases that the choice for $w^{(C)}_{\tau}$ during time slot $(\tau+1)$ must be suboptimal for one of the two frame-size sequences. This causes a nontrivial gap between the optimal offline and online rates, concluding the proof.

Case $2b-1<\tau$

Suppose $w^{(C)}_{\tau} \le d(1-\ell)\ell^2/2$ is chosen during time slot $(\tau+1)$ and frame-size sequence 2 happens. Then the rate is at most

$$\big(K'\big)/ \\ \big(K' + 2d\ell + d\ell(1-\ell) + d(1-\ell)\ell/2\big)$$

versus an optimal value of at least

$$\big(K'\big)/ \\ \big(K' + 2d\ell + d\ell(1-\ell)\big)$$

Otherwise, suppose $w_\tau^{(C)} \geq d(1-\ell)\ell^2)/2$ is chosen during time slot $(\tau+1)$ and frame-size sequence 1 happens. Then the rate is at most

$$\big(K'\big)/ \\ \big(K' + 2d\ell + d\ell(1-\ell) + d(1-\ell)\ell^2 + d(1-\ell)^2\ell^2)/2\big)$$

whereas the optimal rate is at least

$$\big(K'\big)/ \\ \big(K' + 2d\ell + d\ell(1-\ell) + d(1-\ell)\ell^2\big)$$

.

Case $2b-1=\tau$

Suppose $w_\tau^{(C)} \leq w_{\tau,max}^{(C)}/2$ is chosen during time slot $(\tau+1)$. Then the rate is at most

$$\big(K'\big)/ \\ \big(K' + 2d\ell + d\ell(1-\ell) - e'/2\big)$$

versus an optimal value of at least

$$\big(K'\big)/ \\ \big(K' + 2d\ell + d\ell(1-\ell) - e'\big)$$

Otherwise, suppose $w_\tau^{(C)} \geq w_{\tau,max}^{(C)}/2$ is chosen during time slot $(\tau+1)$ and frame-size sequence 1 happens. Then $w_{\tau+1}^{(C)} \leq d(1-\ell)\ell - w_{\tau,max}^{(C)}/2$ Then the rate is at most

$$\big(K'\big)/ \\ \big(K' + 2d\ell + d\ell(1-\ell)) + d(1-\ell)\ell^2 + w_{\tau,max}^{(C)}/2(1-\ell)\big)$$

whereas the optimal rate is at least

$$\big(K'\big)/ \\ \big(K' + 2d\ell + d\ell(1-\ell) + d(1-\ell)\ell^2)\big)$$

.

Case $2b-1>\tau$

Suppose $w_\tau^{(C)} \leq d(1-\ell)\ell^2/2$ is chosen during time slot $(\tau+1)$. Then the rate is at most

$$\big(K'\big)/ \\ \big(K' + 2d\ell + d\ell(1-\ell) - d(1-\ell)\ell^2/2\big)$$

versus an optimal value of at least

$$\begin{aligned}&\big(K'\big)/\\&\big(K' + 2d\ell + d\ell(1-\ell) - d(1-\ell)\ell^2\big)\end{aligned}$$

Otherwise, suppose $w_{\tau}^{(C)} \geq d(1-\ell)\ell^2/2$ is chosen during time slot $(\tau+1)$ and frame-size sequence 1 happens. Then $w_{\tau+1}^{(C)} \leq d(1-\ell)\ell - d(1-\ell)\ell^2/2$ Then the rate is at most

$$\begin{aligned}&\big(K'\big)/\\&\big(K' + 2d\ell + d\ell(1-\ell) + d(1-\ell)\ell^2 + d(1-\ell)^2\ell^2/2\big)\end{aligned}$$

whereas the optimal rate is at least

$$\begin{aligned}&\big(K'\big)/\\&\big(K' + 2d\ell + d\ell(1-\ell) + d(1-\ell)\ell^2\big)\end{aligned}$$

.

6.8.9 Proof of Theorem 11

At a high level, the proof is divided into three steps. First, we bound how many extra symbols are modeled as being sent under Algorithm 4 in terms of $\mathcal{R}_0, \ldots, \mathcal{R}_{t-\tau}$ by adding constraints for $p_{\tau}^{(IP)}, \ldots, p_{t}^{(IP)}$ to equal $O_0, \ldots, O_{t-\tau}$, respectively (Appendix 6.8.9). We then bound the probability that $\sum_{l=0}^{t-\tau} \mathcal{R}_i$ exceeds its mean by a significant amount (Appendix 6.8.9). Finally, we establish the rate in terms of these quantities (Appendix 6.8.9).

Extra symbols sent under Algorithm 4

First, we show that the increase in $\sum_{l=0}^{t} p_l^{(IP)}$ due to adding the constraint $p_{i+\tau}^{(IP)} = O_i$ for $i \in [t-\tau]$ is at most $\mathcal{R}_i$.

Let $w_i^{(Opt)} \in W_i^{(Opt)}$ be the value that minimizes $\big|w_i^{(Opt)} - O_i\big|$.

Suppose $O_i \geq w_i^{(Opt)}$. Then using the values of $p_j^{(IP)}$ for all $j > (i+\tau)$ still satisfy all constraints if $p_{i+\tau}^{(IP)}$ is set to equal O_i and $p_{i+\tau,opt}^{(IP)}$

Otherwise, suppose $O_i < w_i^{(Opt)}$. Let $\delta = \delta' = w_i^{(Opt)} - O_i$. Let us set $w_i^{(Opt)} = O_i$. While $\delta > 0$ let $j = \min_{l \in \{i,\ldots,i+b_i-1 \big| p_{l+\tau}^{(IP)} < \ell_l k_l, \sum_{r=l+1}^{l+\tau} \zeta_r = 0\}}(l)$. At least one such j exists, since otherwise $p_{i+\tau}^{(IP)}$ could be reduced, violating the minimizing the objective function. Increase $p_{j+\tau}^{(IP)}$ by $\min(\ell_j k_j - w_j^{(Opt)}, \delta)$ and decrement δ by the changed amount. The changes ensure that all constraints are satisfied. The total number of extra symbols sent is at most δ'.

Bounding the regret

First, suppose we start with $\big(\tau, t, K, \mathrm{Z}, \mathcal{L}, B\big)$-Split ML Code and then incrementally for $j = 0, \ldots, (t-\tau)$ switch to $\big(\tau, t, K, \mathrm{Z}, \mathcal{L}, B, W^{(O,j)}\big)$-Split ML Code. With each switch, the total number of extra symbols sent is at most $\mathcal{R}_j \leq \ell_j k_j \leq \ell_j m$ (Appendix 6.8.9). In total, the number of extra symbols sent compared to $\big(\tau, t, K, \mathrm{Z}, \mathcal{L}, B\big)$-Split ML Code is $\sum_{l=0}^{t-\tau} \mathcal{R}_j$. The proof follows from the Hoeffding Bound [45]. Formally,

$$\mathbb{P}\big[1/(t+1)\sum_{i=0}^{t}(\mathcal{R}_i - \mathrm{E}[\mathcal{R}_i]) \geq \epsilon_\dagger\big] \leq \qquad e^{-2(t+1)\epsilon_\dagger^2/m}$$

To ensure this probability is at most δ, we require

$$\begin{aligned} \delta &\geq e^{-2(t+1)\epsilon_\dagger^2/m} \\ e^{2(t+1)\epsilon_\dagger^2/m} &\geq 1/\delta \\ 2(t+1)\epsilon_\dagger^2/m &\geq log(1/\delta) \\ t &> log(1/\delta)/(2\epsilon_\dagger^2). \end{aligned}$$

Online approximately optimality

Let

$$\mathcal{R}_{[t]}^{(+)} = \sum_{i=0}^{t} \mathcal{R}_i$$

Let the total number of symbols modeled as being sent under Algorithm 4 be $N_{\tau,t,K,\mathrm{Z},\mathcal{L},B} = \sum_{l=0}^{t-\tau} k_i + p_{l+\tau}^{(IP)}$. Then let

$$N' = N_{\tau,t,K,\mathrm{Z},\mathcal{L},B} - 2(t-\tau)(\tau-1) - \sum_{l=0}^{t}(h_i + q_i - 2)$$

$$\gamma = 2(t-\tau)(\tau-1) + \big(\sum_{l=0}^{t} h_i + q_i - 2\big) + \mathcal{R}_{[t]}^{(+)}.$$

Restricting to using the building block construction causes sending at most an extra up to $(h_i + q_i - 2)$ symbols per time slot i compared to an optimal scheme for the choice of $O_0, \ldots, O_t$. By Lemma 22, defining $W_i^{(Opt)}$ for $i \in [t]$ as we do costs an additional at most $2(t-\tau)(\tau-1)$. So N' is at most the number of symbols sent by an offline rate-optimal scheme. By Appendix 6.8.9, the number of extra symbols sent during time slot i due to sub optimal choice of O_i is $\mathcal{R}_i$. So the total number of extra symbols sent over an optimal coding scheme is at most γ.

Let $K^{(+)} = \sum_{i=0}^{t} k_i$. Then,

$$R^{(opt)} - R^{(on)} \leq \tag{6.96}$$

$$K^{(+)}/N' - K^{(+)}/(N'+\gamma) \leq \qquad K^{(+)}\gamma/\big(N'(N'+\gamma)\big) \leq \gamma/N' \tag{6.97}$$

Recall from Appendix 6.8.9 that with probability at least $(1-\delta)$,

$$\mathcal{R}_{[t]}^{(+)} \leq \sum_{i=0}^{t}(\epsilon k_i + \epsilon_\dagger)$$

Therefore, by the definition of $K^{(+)}$ and the fact that $K^{(+)} \leq N'$,

$$\begin{aligned}
R^{(opt)} - R^{(on)} &\leq \big(2(t-\tau)(\tau-1) + \sum_{l=0}^{t}(h_i + q_i - 2 + \epsilon k_i + \epsilon_\dagger)\big)/N' \\
&\leq \big(2(t-\tau)(\tau-1) + \sum_{l=0}^{t}(h_i + q_i - 2 + \epsilon k_i + \epsilon_\dagger)\big)/K^{(+)} \\
&= \epsilon + \epsilon_\dagger(t+1)/K^{(+)} + \sum_{l=0}^{t}(2\tau + h_i + q_i - 4)/K^{(+)}
\end{aligned}$$

Chapter 7

Conclusion and future directions

Real-time video communication applications, like videoconferencing and online gaming, are becoming the mainstay of communication over the Internet. Prior work on streaming codes is insufficient to provide bandwidth-efficient loss recovery for these applications because existing works assume the sizes of frames are fixed in advance. In contrast, real-time video communication involves transmitting a sequence of frames of varying sizes unknown in advance. Streaming codes that can support varying frame sizes well can help such applications improve the quality of service.

This thesis introduces a new model of streaming codes for variable-size frames and identifies the key challenges for designing high-rate streaming codes. In particular, streaming codes operate in an "online" setting where the amount of data to be transmitted varies over time and is not known in advance. Mitigating the adverse effects of variability requires spreading the data that arrives at a time slot over multiple future packets and determining in real-time how much redundancy to allocate for each frame. The optimal strategy depends on the arrival pattern. Algebraic coding techniques alone are, therefore, insufficient for designing rate-optimal codes.

We address these challenges in several steps. First, we introduce a simplified model where each frame is sent in one packet that is lost or received and analyze fundamental limits on the rate for (a) arbitrary frame-size sequences and (b) any given frame-size sequence. Second, we consider the regime where each frame must be sent immediately (i.e., $\tau_L = 0$). We propose a new framework for designing online rate-optimal constructions using a greedy paradigm for sending parity symbols. Third, we introduce a new methodology for constructing online streaming codes to tackle spreading frame symbols in real-time (specifically, for $\tau_L = 1$) under the same model. The approach combines machine learning with algebraic coding theory tools by (a) isolating the component that can benefit from machine learning, (b) solving the offline version of the problem by integrating optimization with algebraic coding theory techniques, and (c) converting the offline scheme into an online one using a learning-based approach. Fourth, we establish that these theoretical results can translate into practical settings. To do so, we analyze a large corpus of traces from Microsoft Teams to determine the suitability of streaming codes. We then design Tambur, a new communication scheme for bandwidth-efficient loss recovery for videoconferencing comprising two components: (a) A new streaming code that bridges the gap between theoretical streaming codes and videoconferencing applications, which takes as input any given bandwidth overhead; (b) a learning-based predictive model to set the bandwidth overhead. We assessed Tambur offline over a dataset of traces from Teams and online over a simulated network. We showed improvements in QoE metrics, including 26% fewer freezes and 28% fewer non-rendered frames. The benefits establish streaming codes as a viable solution to recovering lost packets for videoconferencing applications. The results thus also show the promise of streaming codes for other live-streaming applications like cloud gaming. We released our framework as open-source codebase. The framework enables easy evaluation of the QoE benefits of new communication schemes by providing a simple interface to incorporate (a) new FEC schemes and (b) new learning-based predictive models. Fifth, using the learnings from our analysis of real-world loss traces, we introduce a generalized streaming model with partial bursts. We then design offline and online approximately rate-optimal streaming codes by using a linear program and learning-augmented algorithm, respectively, to determine how to split frames into two components. One component is recovered using the parity symbols sent for prior frames, and additional parity symbols are sent to recover the other component. A building block construction is then presented to design an approximately rate-optimal code given how frame symbols are split.

Overall, this thesis expanded the toolkit for real-time communication to include new interdisciplinary techniques combining algebraic coding theory with algorithms, optimization, and machine learning. We end by presenting a few potential avenues of future direction for this toolkit.

7.1 Competitive analysis

One of this thesis' contributions was to show that online streaming codes cannot match the offline-optimal-rate in several settings. Specifically, Chapter 3 (respectively, Chapter 6) showed that the best way to spread (respectively, split) frame symbols depends on the sizes of future frames for all but a few parameter regimes. The result was shown via a case analysis using the following argument. Two possible frame-size sequences were introduced that were identical for the first several time slots. An offline coding scheme was presented for the frame-size sequences. We showed that matching the rate of the offline scheme on the first frame-size sequence required spreading (respectively, splitting) at least some number of frame symbols during the time slots where the two frame-size sequences were indistinguishable. Then we showed that this choice of spreading (respectively, splitting) penalized the rate on the second frame-size sequence. This accomplished the objective of establishing a gap between the online-optimal-rate and the offline-optimal-rate. But several questions remain, including:

1. How large is the gap in the worst case?
2. What is the worst-case gap for a given policy for spreading (respectively, splitting) symbols?
3. What policy spreading (respectively, splitting) symbols leads to the smallest gap?

To answer these open questions, one could leverage the existing literature [4, 34] on online algorithms' performance, which is typically called "competitive analysis." To facilitate future research, we translate the nomenclature of this thesis into the terminology used under competitive analysis for the model considered in Chapter 3 for $\tau_L = 1$ and any valid parameters b and τ.

An online streaming algorithm, A, receives a request sequence, $\sigma = \sigma(1), \sigma(2), \ldots, \sigma(t)$ of non-negative integers reflecting the sizes of frames. For request, $i \in [m]$, the online algorithm does not have access to $\sigma(i')$ for $i' > i$. Algorithm A determines how many symbols of each frame to spread. Let the values A computes on request sequence σ be $A_\sigma(\sigma_1), \ldots, A_\sigma(\sigma_t) \in [m]$. Suppose the $(\tau, b, t, \langle A_\sigma(\sigma_1), \ldots, A_\sigma(\sigma_t) \mid i \in [t] \rangle)$ −Spreading Variable-sized Generalized MS Code (from Chapter 4.2) sends channel packets of sizes $n_0, \ldots, n_t$. The cost of A on request sequence σ is given by $\text{cost}_A(\sigma) = \sum_{i=0}^{t} n_i$. The cost represents the total number of symbols sent by a rate-optimal construction given A's choices of how to spread frame symbols. The competitive ratio of A is given by

$$\inf\{c \mid \text{cost}_A(\sigma) \leq c \cdot \text{cost}_B(\sigma), \forall \sigma \in [m]^t, \forall B\}.$$

For any given request sequence, σ it may be useful to know the optimal cost. The optimal cost can be exactly computed using Algorithm 3 and approximately computed using Algorithm 3.B.

7.2 Stochastic optimization

One of the core problems tackled by this thesis is communicating in the face of uncertainty in the sizes of future frames. Chapter 3 used a greedy paradigm to circumvent this challenge for certain parameter regimes and showed that no such workaround exists for all remaining settings. Instead, Chapter 4 introduced a learning-based approach to design online approximately rate-optimal streaming codes. Under this approach, a predictive decision is employed for the one component that fundamentally depends on future frames' sizes: how to spread frame symbols. We developed an explicit approach for making these predictive decisions with sufficient accuracy; this approach can be viewed as stochastic optimization.

Future work might further study how to use stochastic optimization to best make the decision for spreading frame symbols. To make this direction more concrete, we discuss how this thesis' terminology relates to an established framework for stochastic optimization [72].

Let the state, $\mathbb{S}_i$, comprise (a) the sizes of frames 0 through i, (b) how the symbols of the prior τ frames were spread, (c) the number of parity symbols sent during the prior τ time slots, and (d) the number of parity symbols allocated to be sent during time slot i through $(i + \tau - 1)$. The exogenous information, $W_i \in [m]$ is the size of the ith frame. The decision variable is $a_t \in [k_i]$ and reflects how to spread the symbols of $S[i]$. The decision is made with a policy $A^\pi(\mathbb{S}_i) = a_i$ where π contains the information about the function and parameters of the policy. The definition of the transition function, $\mathbb{S}^M(\mathbb{S}_i, a_i, W_{i+1})$, follows immediately from the definition of the state. The objective function is

$$\min_\pi \left[\mathrm{E}_{W_1,\ldots,W_t} \left\{ \sum_{i=0}^{t} C(\mathbb{S}_{i+1}, A^\pi(\mathbb{S}_{i+1}), W_{i+1}) \right\} \right],$$

where

$$\begin{aligned} \mathbb{S}_{i+1} &= \mathbb{S}^M(\mathbb{S}_i, a_i, W_{i+1}), \\ C(\mathbb{S}_i, a_i, W_{i+1}) &= n_i, \end{aligned}$$

and n_i is the size of the ith channel packet. The objective function reflects the expected number of symbols sent used assuming an optimal policy is applied; hence, minimizing the objective function maximizes the communication rate.

Finally, we list a few open problems where stochastic optimization may be suitable. The number of samples to the distribution of the sizes of future frames that are needed to design a policy for spreading frame symbols that is within ϵ of accurate in expectation is currently unknown. In fact, whether this number of samples can be reduced from polynomial (Chapter 4) to linear or sublinear is as yet unknown. Recall in Chapter 4 that the choice of how to *spread* $S[i]$'s symbols over $X[i]$ and $X[i+1]$ is determined during time slot i. In contrast, how to *split* $S[i]$'s symbols depends on the parity symbols sent during time slots i through $(i+\tau-1)$. For example, if no parity symbols are sent during time slots i through $(i+\tau-1)$, then all of $S[i]$'s symbols are recovered at their deadline. However, if sufficiently many parity symbols are sent during any of time slots i through $(i+\tau-1)$, all of $S[i]$'s symbols can be recovered within $(\tau-1)$ time slots. Thus, the split for frame i is not determined until *after* time slot i. For this reason, we did not use stochastic optimization in Chapter 6 to determine how to split frame symbols. How to combine stochastic optimization and coding theory to determine the best way to split frames remains an open problem.

7.3 Data-driven coding theory in new domains

Recall that our goal was to improve the QoE for real-time streaming applications like videoconferencing that involve sending variable-size data. Thus, we introduced a new streaming model that is closely related to prior models studied in a series of related works [11, 27, 30, 35, 49, 52, 62, 63]. Unlike these prior models, our model incorporated (a) variable-size frames, and (b) partial bursts. Addressing these two unique aspects was our main challenge.

To do so, we developed an interdisciplinary toolkit to design and analyze codes using stochastic optimization, online algorithms, coding theory, and machine learning. These methods could be applied beyond the specific domain of live communication considered in this thesis. For example, one could expand our model to consider unequal error protection for frames using a similar methodology to that of [44]. Two loss modes would be considered: (a) short partial bursts where a small fraction of packets are lost per frame, and (b) long partial bursts where a large fraction of packets are lost per frame. All frames would need to be recovered after a short partial burst, but only keyframes would need to be recovered after a long one. Unequal error protection may be useful in applications like videoconferencing where one type of frame (specifically, a keyframe) is extra important to the QoE.

Future work could also expand this thesis' streaming model to include aspects of the various other streaming models discussed in Chapter 2.1.2 (e.g., multi-hop networks [28, 37, 53]) then apply our toolkit to design and analyze new erasure codes. More generally, the methods developed in this thesis could be used in any scenario where coding theory is applied online over variable-size data.

7.4 Improvements to Tambur

One of the contributions of Chapter 5 was to release Tambur, an open-source library integrated with Ringmaster that handles real-time packet-loss recovery for videoconferencing applications. There are several possible future directions here. For concreteness, we will list three.

First, recall that Tambur comprises two components: (a) a new streaming code, and (b) an ML model to take predictive decisions on the bandwidth overhead. The second component was a simple ML model trained offline on a dataset of traces from Microsoft Teams 1:1 video calls. For simplicity, the ML model uses binary classification to set the bandwidth overhead. Instead, a multi-class classification model would allow for greater flexibility to tune the bandwidth overhead. Second, Tambur could be combined with bandwidth estimation. Using a variable bitrate for video encoding to leverage Tambur's bandwidth savings may enable communicating the video at a higher resolution. Third, one could use the library's interface to implement the learning-augmented streaming code introduced in Chapter 6. Doing requires (a) replacing the feedback with estimates of the two channel parameters, and (b) a new predictive model to determine how to split frames.

Bibliography

[1] libvpx. `https://chromium.googlesource.com/webm/libvpx/`. 5.3.3

[2] WebRTC. `https://webrtc.org/`. 5.3.3

[3] N. Adler and Y. Cassuto. Burst-erasure correcting codes with optimal average delay. *IEEE Transactions on Information Theory*, 63(5):2848–2865, May 2017. ISSN 1557-9654. doi: 10.1109/TIT.2017.2663110. 1, 2.1.2

[4] Susanne Albers. Online algorithms: a survey. *Mathematical Programming*, 97:3–26, 2003. 7.1

[5] Keerti Anand, Rong Ge, and Debmalya Panigrahi. Customizing ml predictions for online algorithms. In *International Conference on Machine Learning*, pages 303–313. PMLR, 2020. 1.3

[6] Antonios Antoniadis, Christian Coester, Marek Elias, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. In *International Conference on Machine Learning*, pages 345–355. PMLR, 2020. 1.3

[7] A. Badr, A. Khisti, and E. Martinian. Diversity embedded streaming erasure codes (desco): Constructions and optimality. *IEEE J. Sel. Areas Inf. Theory*, 29(5):1042–1054, May 2011. ISSN 1558-0008. doi: 10.1109/JSAC.2011.110514. 1, 2.1.2

[8] A. Badr, A. Khisti, W. Tan, and J. Apostolopoulos. Streaming codes with partial recovery over channels with burst and isolated erasures. *IEEE Journal of Selected Topics in Signal Processing*, 9(3):501–516, April 2015. ISSN 1941-0484. doi: 10.1109/JSTSP.2014.2388191. 1, 2.1.2

[9] A. Badr, D. Lui, and A. Khisti. Streaming codes for multicast over burst erasure channels. *IEEE Trans. Inf. Theory*, 61(8):4181–4208, Aug 2015. ISSN 1557-9654. doi: 10.1109/TIT.2015.2445753. 1, 2.1.2

[10] A. Badr, A. Khisti, W. Tan, and J. Apostolopoulos. Perfecting protection for interactive multimedia: A survey of forward error correction for low-delay interactive applications. *IEEE Signal Processing Magazine*, 34(2):95–113, March 2017. ISSN 1558-0792. doi: 10.1109/MSP.2016.2639062. 1

[11] A. Badr, P. Patil, A. Khisti, W. Tan, and J. Apostolopoulos. Layered constructions for low-delay streaming codes. *IEEE Transactions on Information Theory*, 63(1):111–141, Jan 2017. ISSN 1557-9654. doi: 10.1109/TIT.2016.2618924. 1, 1.5, 2.1.1, 2, 2.1.2, 2.2, 3.2, 4.2, 5.1.2, 6.2, 6.8.2, 6.8.2, 7.3

[12] A. Badr, D. Lui, A. Khisti, W. Tan, X. Zhu, and J. Apostolopoulos. Multiplexed coding for multiple streams with different decoding delays. *IEEE Trans. Inf. Theory*, 64(6):4365–4378, June 2018. ISSN 1557-9654. doi: 10.1109/TIT.2018.2827367. 1, 2.1.2

[13] Ahmed Badr, Ashish Khisti, Wai-tian Tan, Xiaoqing Zhu, and John Apostolopoulos. FEC for VoIP using dual-delay streaming codes. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017. 1, 2.1.2, 5.1.2

[14] Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. Developing a predictive model of quality of experience for internet video. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, page 339–350, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320566. doi: 10.1145/2486001.2486025. URL `https://doi.org/10.1145/2486001.2486025`. 5.4.1

[15] Dziugas Baltrunas, Ahmed Elmokashfi, Amund Kvalbein, and Özgü Alay. Investigating packet loss in mobile broadband networks under mobility. In *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 225–233. IEEE, 2016. 5.2.2

[16] Étienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms. *Advances in Neural Information Processing Systems*, 33:20083–20094, 2020. 1.3

[17] Henrik Boström, Harald Alvestrand, and Varun Singh. Provisional identifiers for WebRTC's statistics API unofficial draft. Draft of a potential specification, W3C, July 2022. `https://w3c.github.io/webrtc-provisional-stats/#RTCVideoReceiverStats-dict`. 8

[18] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1124–1137, 2004. 5.3.3

[19] Ramon Caceres, Alan Clark, and Timur Friedman. RTP Control Protocol Extended Reports (RTCP XR). RFC 3611, November 2003. URL `https://rfc-editor.org/rfc/rfc3611.txt`. 5.3.2, 4

[20] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. Analysis and design of the Google congestion control for web real-time communication (WebRTC). In *Proceedings of the 7th International Conference on Multimedia Systems*, pages 1–12, 2016. 5.3.3

[21] Hyunseok Chang, Matteo Varvello, Fang Hao, and Sarit Mukherjee. Can you see me now? A measurement study of Zoom, Webex, and Meet. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 216–228, 2021. 1

[22] Federal Communications Commission. Measuring Broadband America, 2021. `https://www.fcc.gov/reports-research/reports/measuring-broadband-america/measuring-fixed-broadband-eleventh-report` (Last accessed: 2022-02-02). 5.2.2

[23] Mauro Conti, Simone Milani, Ehsan Nowroozi, and Gabriele Orazi. Do not deceive your employer with a virtual background: A video conferencing manipulation-detection system. *CoRR*, abs/2106.15130, 2021. URL `https://arxiv.org/abs/2106.15130`. 5.4.1

[24] Ross Cutler, Yasaman Hosseinkashi, Jamie Pool, Senja Filipi, Robert Aichner, Yuan Tu, and Johannes Gehrke. Meeting effectiveness and inclusiveness in remote collaboration. *Proc. ACM Hum.-Comput. Interact.*, 5(CSCW1), apr 2021. doi: 10.1145/3449247. URL `https://doi.org/10.1145/3449247`. 1

[25] Marcel Dischinger, Andreas Haeberlen, Krishna P. Gummadi, and Stefan Saroiu. Characterizing residential broadband networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 43–56, 2007. 5.2.2

[26] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. Understanding the impact of video quality on user engagement. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, page 362–373, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450307970. doi: 10.1145/2018436.2018478. URL `https://doi.org/10.1145/2018436.2018478`. 5.4.1

[27] E. Domanovitz, S. L. Fong, and A. Khisti. An explicit rate-optimal streaming code for channels with burst and arbitrary erasures. In *2019 IEEE Information Theory Workshop (ITW)*, pages 1–5, 2019. 1, 2.1.1, 2.2, 2.4, 2.4, 2.4.2, 3.7.2, 7.3

[28] Elad Domanovitz, Ashish Khisti, Wai-Tian Tan, Xiaoqing Zhu, and John Apostolopoulos. Streaming erasure codes over multi-hop relay network. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 497–502. IEEE, 2020. 2.1.2, 7.3

[29] Zhengfang Duanmu, Kai Zeng, Kede Ma, Abdul Rehman, and Zhou Wang. A quality-of-experience index for streaming video. *IEEE Journal of Selected Topics in Signal Processing*, 11(1):154–166, 2017. doi: 10.1109/JSTSP.2016.2608329. 5.4.1

[30] D. Dudzicz, S. L. Fong, and A. Khisti. An explicit construction of optimal streaming codes for channels with burst and arbitrary erasures. *IEEE Transactions on Communications*, 68 (1):12–25, 2020. 2.1.1, 2.2, 2.2, 2.4, 2.4, 2.4.2, 3.7.2, 7.3

[31] E. O. Elliott. Estimates of error rates for codes on burst-noise channels. *The Bell System Technical Journal*, 42(5):1977–1997, Sep. 1963. ISSN 0005-8580. doi: 10.1002/j.1538-7305.1963.tb00955.x. 1, 2.1.1, 5.1.2, 5.4.1

[32] Martin Ellis. *Understanding the performance of Internet video over residential networks*. PhD thesis, University of Glasgow, 2012. 5.2.2

[33] Salma Shukry Emara, Silas Fong, Baochun Li, Ashish Khisti, Wai-Tian Tan, Xiaoqing Zhu, and John Apostolopoulos. Low-latency network-adaptive error control for interactive streaming. *IEEE Transactions on Multimedia*, pages 1–1, 2021. doi: 10.1109/TMM.2021.3070134. 1, 5.1.2

[34] Amos Fiat and Gerhard J Woeginger. *Online algorithms: The state of the art*, volume 1442. Springer, 1998. 7.1

[35] 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 Trans. Inf. Theory*, 65(7):4274–4292, July 2019. ISSN 1557-9654. doi: 10.1109/TIT.2019.2894124. 1, 2.1.1, 2.2, 2.2, 2.4, 2.4, 2.4.2, 3.7.2, 7.3

[36] S. L. Fong, A. Khisti, B. Li, W. Tan, X. Zhu, and J. Apostolopoulos. Optimal multiplexed erasure codes for streaming messages with different decoding delays. *IEEE Trans. Inf. Theory*, 66(7):4007–4018, 2020. 1, 2.1.2

[37] Silas L. Fong, Ashish Khisti, Baochun Li, Wai-Tian Tan, Xiaoqing Zhu, and John Apostolopoulos. Optimal streaming erasure codes over the three-node relay network. *IEEE Trans. Inf. Theory*, 66(5):2696–2712, 2020. doi: 10.1109/TIT.2019.2940833. 1, 2.1.2, 7.3

[38] G. Forney. Burst-correcting codes for the classic bursty channel. *IEEE Transactions on Communication Technology*, 19(5):772–781, October 1971. ISSN 2162-2175. doi: 10.1109/TCOM.1971.1090719. 2.1.1

[39] EM Gabidulin. Convolutional codes over large alphabets. In *Proc. Int. Workshop on Algebraic Combinatorial and Coding Theory*, pages 80–84, 1988. 6.2

[40] Boni García, Micael Gallego, Francisco Gortázar, and Antonia Bertolino. Understanding and estimating quality of experience in WebRTC applications. *Computing*, 101(11):1585–1607, 2019. 1

[41] Jim Gettys and Kathleen Nichols. Bufferbloat: Dark buffers in the internet. *Queue*, 9(11): 40–54, 2011. 2.1.1

[42] E. N. Gilbert. Capacity of a burst-noise channel. *Bell System Technical Journal*, 39(5):1253–1265, 1960. doi: 10.1002/j.1538-7305.1960.tb03959.x. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1960.tb03959.x`. 2.1.1

[43] Heide Gluesing-Luerssen, Joachim Rosenthal, and Roxana Smarandache. Strongly-mds convolutional codes. *IEEE Transactions on Information Theory*, 52(2):584–598, 2006. 6.2

[44] Mahdi Haghifam, M Nikhil Krishnan, Ashish Khisti, Xiaoqing Zhu, Wai-Tian Tan, and John Apostolopoulos. On streaming codes with unequal error protection. *IEEE J. Sel. Areas Inf. Theory*, 2021. 1, 2.1.2, 7.3

[45] Wassily Hoeffding. *Probability Inequalities for sums of Bounded Random Variables*, pages 409–426. Springer New York, New York, NY, 1994. ISBN 978-1-4612-0865-5. doi: 10.1007/978-1-4612-0865-5_26. URL `https://doi.org/10.1007/978-1-4612-0865-5_26`. 4.4, 4.7.4, 6.8.9

[46] Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *International Conference on Learning Representations*, 2019. 1.3

[47] Tanqiu Jiang, Yi Li, Honghao Lin, Yisong Ruan, and David P Woodruff. Learning-augmented data stream algorithms. In *International Conference on Learning Representations*, 2019. 1.3

[48] Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 international conference on management of data*, pages 489–504, 2018. 1.3

[49] M. N. Krishnan and P. V. Kumar. Rate-optimal streaming codes for channels with burst and isolated erasures. In *ISIT*, pages 1809–1813, June 2018. doi: 10.1109/ISIT.2018.8437570. 1, 2.1.1, 2.2, 2.2, 2.4, 2.4, 2.4.2, 3.7.2, 7.3

[50] M. N. Krishnan, V. Ramkumar, M. Vajha, and P. V. Kumar. Simple streaming codes for reliable, low-latency communication. *IEEE Communications Letters*, pages 1–1, 2019. ISSN 2373-7891. doi: 10.1109/LCOMM.2019.2956500. 1, 2.3.2, 2.3.2, 2.3.2

[51] M. N. Krishnan, D. Shukla, and P. V. Kumar. A quadratic field-size rate-optimal streaming code for channels with burst and random erasures. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 852–856, 2019. 3.7.2, 3.7.4

[52] M Nikhil Krishnan, Deeptanshu Shukla, and P Vijay Kumar. Rate-optimal streaming codes for channels with burst and random erasures. *IEEE Trans. Inf. Theory*, 66(8):4869–4891, 2020. 1, 2.1.1, 2.2, 2.2, 2.4, 2.4, 2.4.2, 7.3

[53] M Nikhil Krishnan, Gustavo Kasper Facenda, Elad Domanovitz, Ashish Khisti, Wai-Tian Tan, and John Apostolopoulos. High rate streaming codes over the three-node relay network. In *2021 IEEE Information Theory Workshop (ITW)*, pages 1–6. IEEE, 2021. 2.1.2, 7.3

[54] S. Shunmuga Krishnan and Ramesh K. Sitaraman. Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs. In *Proceedings of the 2012 Internet Measurement Conference*, IMC '12, page 211–224, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450317054. doi: 10.1145/2398776.2398799. URL `https://doi.org/10.1145/2398776.2398799`. 5.4.1

[55] Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1859–1877. SIAM, 2020. 1.3

[56] D. Leong and T. Ho. Erasure coding for real-time streaming. In *ISIT*, pages 289–293, July 2012. doi: 10.1109/ISIT.2012.6284055. 1, 1.5, 2.1.2

[57] D. Leong, A. Qureshi, and T. Ho. On coding for real-time streaming under packet erasures. In *ISIT*, pages 1012–1016, July 2013. doi: 10.1109/ISIT.2013.6620379. 1, 1.5, 2.1.2

[58] Z. Li, A. Khisti, and B. Girod. Correcting erasure bursts with minimum decoding delay. In *Conf. Rec. Asilomar Conf. Signals Syst. Comput.*, pages 33–39, Nov 2011. doi: 10.1109/ACSSC.2011.6189949. 1, 2.1.2

[59] Thodoris Lykouris and Sergei Vassilvtiskii. Competitive caching with machine learned advice. In *International Conference on Machine Learning*, pages 3296–3305. PMLR, 2018. 1.3

[60] David J. C. MacKay. Fountain codes. *IEE Proceedings-Communications*, 152(6):1062–1068, 2005. 5.1.1

[61] Kyle MacMillan, Tarun Mangla, James Saxon, and Nick Feamster. Measuring the performance and network utilization of popular video conferencing applications. In *Proceedings of the 21st ACM Internet Measurement Conference*, IMC '21, page 229–244, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450391290. doi: 10.

1145/3487552.3487842. URL `https://doi.org/10.1145/3487552.3487842`. 1

[62] E. Martinian and C. . W. Sundberg. Burst erasure correction codes with low decoding delay. *IEEE Trans. Inf. Theory*, 50(10):2494–2502, Oct 2004. ISSN 1557-9654. doi: 10.1109/TIT.2004.834844. 1, 2.1.1, 1, 2.2, 2.3.1, 2.3.1, 5.1.2, 7.3

[63] E. Martinian and M. Trott. Delay-optimal burst erasure code construction. In *ISIT*, pages 1006–1010, June 2007. doi: 10.1109/ISIT.2007.4557355. 1, 2.1.1, 2.2, 2.3.1, 2.3.1, 8, 7.3

[64] Michael Mitzenmacher. A model for learned bloom filters and optimizing by sandwiching. *Advances in Neural Information Processing Systems*, 31, 2018. 1.3

[65] Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. *arXiv preprint arXiv:2006.09123*, 2020. 1.3

[66] Anush Krishna Moorthy, Lark Kwon Choi, Alan Conrad Bovik, and Gustavo de Veciana. Video quality assessment on mobile devices: Subjective, behavioral and objective studies. *IEEE Journal of Selected Topics in Signal Processing*, 6(6):652–671, 2012. doi: 10.1109/JSTSP.2012.2212417. 1.4

[67] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. Mahimahi: Accurate Record-and-Replay for HTTP. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*, pages 417–429, Santa Clara, CA, July 2015. USENIX Association. ISBN 978-1-931971-225. URL `https://www.usenix.org/conference/atc15/technical-session/presentation/netravali`. 5.4.1

[68] Ehsan Nowroozi, Ali Dehghantanha, Reza M Parizi, and Kim-Kwang Raymond Choo. A survey of machine learning techniques in adversarial image forensics. *Computers & Security*, page 102092, 2020. 5.4.1

[69] Kohong Park and Walter Willinger. *Sele-Similar network traffic and performance evaluation*. Wiley & Son, 2000. 5.4.1

[70] James S. Plank, Scott Simmerman, and Catherine D. Schuman. Jerasure: A library in C/C++ facilitating erasure coding for storage applications-version 1.2. *University of Tennessee, Tech. Rep. CS-08-627*, 23, 2008. 5.3.3

[71] James S. Plank, Ethan L. Miller, Kevin M. Greenan, Benjamin A. Arnold, John A. Burnum, Adam W. Disney, and Allen C. McBride. GF-Complete: A comprehensive open source library for galois field arithmetic version 1.02, 2014. 5.3.3

[72] Warren B. Powell. A unified framework for stochastic optimization. *European Journal of Operational Research*, 275(3):795–821, 2019. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2018.07.014. URL `https://www.sciencedirect.com/science/article/pii/S0377221718306192`. 7.2

[73] Yining Qi and Mingyuan Dai. The effect of frame freezing and frame skipping on video quality. In *2006 International Conference on Intelligent Information Hiding and Multimedia*, pages 423–426, 2006. doi: 10.1109/IIH-MSP.2006.265032. 1.4

[74] Ramya Raghavendra and Elizabeth M. Belding. Characterizing high-bandwidth real-time

video traffic in residential broadband networks. In *8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, pages 597–602. IEEE, 2010. 5.2.2

[75] Vinayak Ramkumar, Myna Vajha, M. Nikhil Krishnan, and P. Vijay Kumar. Staggered diagonal embedding based linear field size streaming codes, 2020. 1, 2.3.2

[76] Irving S. Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960. 5.1.1

[77] Michael Rudow and K. V. Rashmi. Learning-augmented streaming codes are approximately optimal for variable-size messages. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pages 474–479, 2022. doi: 10.1109/ISIT50566.2022.9834539. 1.3

[78] Michael Rudow and K. V. Rashmi. Streaming codes for variable-size messages. *IEEE Transactions on Information Theory*, 68(9):5823–5849, 2022. doi: 10.1109/TIT.2022.3170895. 1.1

[79] Michael Rudow and K.V. Rashmi. Online versus offline rate in streaming codes for variable-size messages. *IEEE Transactions on Information Theory*, pages 1–1, 2023. doi: 10.1109/TIT.2023.3244799. 1.2

[80] Michael Rudow and K.V. Rashmi. Learning-augmented streaming codes for variable-size messages under partial burst losses. In *2023 IEEE International Symposium on Information Theory (ISIT)*, page to appear, 2023. 1.5

[81] Michael Rudow, Francis Y. Yan, Abhishek Kumar, Ganesh Ananthanarayanan, Martin Ellis, and K.V. Rashmi. Tambur: Efficient loss recovery for videoconferencing via streaming codes. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 953–971, Boston, MA, April 2023. USENIX Association. ISBN 978-1-939133-33-5. URL `https://www.usenix.org/conference/nsdi23/presentation/rudow`. 1.4

[82] Pin-Wen Su, Yu-Chih Huang, Shih-Chun Lin, I-Hsiang Wang, and Chih-Chun Wang. Random linear streaming codes in the finite memory length and decoding deadline regime. In *ISIT*, pages 730–735, 2021. doi: 10.1109/ISIT45174.2021.9518162. 1

[83] Justin Uberti. WebRTC Forward Error Correction Requirements. RFC 8854, January 2021. URL `https://rfc-editor.org/rfc/rfc8854.txt`. 1, 5.2

[84] IT Union. ITU-T G. 1010: End-User Multimedia Qos Categories. *G SERIES: Transmission Systems and Media, Digital System and Networks-Multimedia Quality of Service and Performance Generic and User-Related Aspects*, 2001. 1, 5.2.1, 1, 5.4.1, 6.2

[85] Yunkai Wei and Tracey Ho. On prioritized coding for real-time streaming under packet erasures. In *Allerton*, pages 327–334. IEEE, 2013. 1, 2.1.2

[86] Francis Y. Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. Learning *in situ*: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 495–511, Santa Clara, CA, February 2020. USENIX Association. ISBN 978-1-939133-13-7. URL `https://www.usenix.org/conference/`

nsdi20/presentation/yan. 1.4

[87] Mo Zanaty, Varun Singh, Ali C. Begen, and Giridhar Mandyam. RTP Payload Format for Flexible Forward Error Correction (FEC). RFC 8627, July 2019. URL https://rfc-editor.org/rfc/rfc8627.txt. 5.1.1

[88] Xu Zhang, Yiyang Ou, Siddhartha Sen, and Junchen Jiang. SENSEI: Aligning video streaming quality with dynamic user sensitivity. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 303–320. USENIX Association, April 2021. ISBN 978-1-939133-21-2. URL https://www.usenix.org/conference/nsdi21/presentation/zhang-xu. 5.4.1